Akustična kriptoanaliza tipkovnica

Breš, Barbara

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni diplomski studij računarstva

Diplomski rad

Akustična kriptoanaliza tipkovnica

Rijeka, rujan 2023.

Barbara Breš
0069085153

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni diplomski studij računarstva

Diplomski rad

Akustična kriptoanaliza tipkovnica

Mentor: izv. prof. dr. sc. Jonatan Lerga

Rijeka, rujan 2023.

Barbara Breš
0069085153

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 20. ožujka 2023.

Zavod: **Zavod za računarstvo**
Predmet: **Kodiranje i kriptografija**
Grana: **2.09.02 informacijski sustavi**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Barbara Breš (0069085153)**
Studij: Sveučilišni diplomski studij računarstva
Modul: Računalni sustavi

Zadatak: **Akustična kriptoanaliza / Acoustic Cryptanalysis**

Opis zadatka:

Potrebno je dati pregled poznatih akustičnih kriptoanaliza. U ovom diplomskom radu potrebno je fokusirati se na kriptoanalizu iz zvukova nastalih tijekom tipkanja po tipkovnici. Završetkom snimanja i prikupljanja zvukova potrebno je provesti analizu istih kako bi se izvukle korisne informacije o tekstu koji tipkan. Potrebno je primjeniti metode strojnog učenja i testirati točnost takve analize.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Izv. prof. dr. sc. Jonatan Lerga

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Miroslav Joler

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, rujan 2023.

Barbara Breš

Zahvala

Htjela bih se zahvaliti svojem mentoru izv. prof. dr. sc. Jonatan Lerga na podršci prilikom cijelog ciklusa izrade ovog rada, od kreiranja same ideje do njezine realizacije. Vaša podrška mi je pomogla da uspješno savladam sve prepreke na koje sam naletjela putem.

Sadržaj

Popis slika

Poglavlje 1

Uvod

Sigurnost je veliki faktor današnjeg društva, no osim u stvarnom svijetu veliku ulogu igra i u računalnom svijetu. Kibernetički napadi sve su poznatiji i popularniji danas, no postoje mnoge preventive protiv istih. Napadi koji predstavljaju jednaku razinu opasnosti kao i kibernetički, a nisu toliko poznati široj populaciji su napadi sporednim kanalom.

Rad kriptografskog sustava uzrokuje mnoge fizičke učinke kao što su: količina potrošnje energije, vrijeme obavljanja procesa, emitirani zvuk, itd. Napadi sporednim kanalom vrsta su napada kojima je cilj prikupiti informacije ili utjecati na izvođenje programa sustava, mjerenjem ili iskorištavanjem navedenih fizičkih učinaka koje emitira kriptosustav, umjesto izravnog ciljanja programa ili njegovog koda, kao što to kibernetički napadi rade. Napadači iskorištavaju prikupljene informacije o fizičkim učincima kriptosustava za dobivanje informacija o probijanju kriptosustava[4].

Postoje razni tipovi napada sporednim kanalom, a neki od njih su: napadi provedeni analizom snage i potrošnjom struje [5], tempirani napadi [6], elektromagnetska analiza [7], akustična kriptoanaliza [8], optička kriptoanaliza [9], napadi sporednim kanalom preko analize temperature [10], itd.

U ovom radu pobliže će se analizirati akustična kriptoanaliza. Akustična kriptoanaliza je napad sporednim kanalom koji se bavi snimanjem, prikupljanjem i analizi-

ranjem zvukova koje proizvodi kriptosustav prilikom izvođenja računalnih operacija. Analizom prikupljenih zvukova, napadači mogu izvući osjetljive podatke o kriptosustavu kao što su kriptografski ključevi ili korisničke lozinke i pinovi [11].

Korištenjem akustične kriptoanalize, po uzoru na istraživački rad istraživača Li Zhuang, Feng Zhou, J. D. Tygar sa Sveučilišta u Kaliforniji [12], u nastavku rada bit će predstavljena sigurnosna prijetnja današnjih tipkovnica te način na koji je moguće koristeći akustičnu kriptoanalizu i metode strojnog učenja, dešifrirati, s visokim postotkom točnosti, tekst tipkan na tipkovnici iz audio snimki.

Poglavlje 2

Akustična kriptoanaliza

2.1 Definicija i provođenje napada

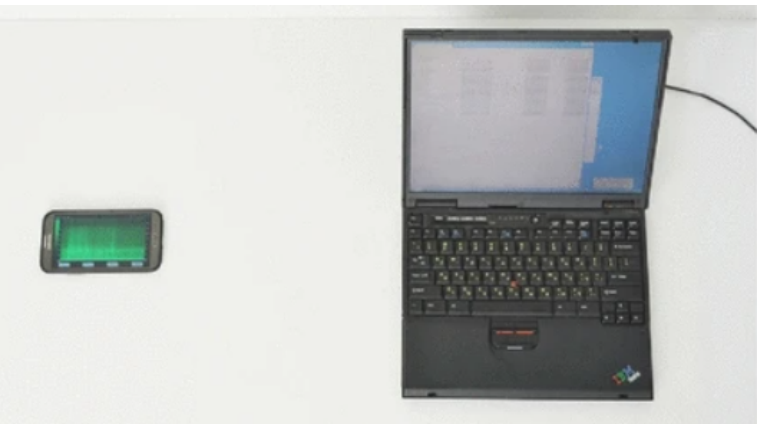
Akustična kriptoanaliza, kao što je i spomenuto već u uvodu, je napad sporednim kanalom u kojem napadač prikuplja zvukove koje proizvodi kriptosustav prilikom određivanja operacija. Te operacije mogu biti: izvođenje procesa, kriptiranje i dekriptiranje podataka, utipkavanje lozinke ili pina, itd. Iz navedenog vidljivo je da napadač može koristiti unutarnje zvukove koje proizvodi kriptosustav, kao što su ultrazvučni šumovi koji izlaze iz kondenzatora i induktora na matičnoj ploči računala, ili vanjske zvukove, kao što je zvuk tipkanja na tipkovnici, kako bi prikupio osjetljive informacije i iskoristio ih za probijanje kriptosustava [13].

Provođenje napada akustičnom kriptoanalizom najčešće se provodi u **četiri** koraka.

1. **Identificiranje mete**
2. **Snimanje zvuka**
3. **Analiza snimljenog zvuka**
4. **Dekripcija osjetljivih informacija**

Kao prvi korak napadač mora identificirati metu koju će pratiti i osluškivati kako bi izvukao osjetljive informacije iz iste. Ove mete mogu biti određeni uređaj ili sustav za koji se zna da se koristi za komunikaciju, poput telefona ili računala.

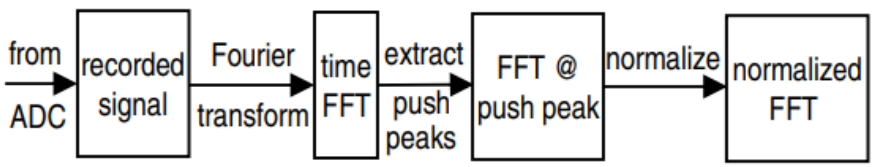
Nakon odabira mete, napadač snima zvukove koje odabrana meta proizvodi, bilo to unutarnje, vanjske ili oboje. Navedeni zvukovi se mogu snimati koristeći mikrofone ili neke druge uređaje. Korištenjem osjetljivijih mikrofona moguće je prikupiti više informacija, no istraživači *Dmitri Asonov i Rakesh Agrawal* dokazali su u svom istraživačkom radu da i korištenjem jeftinije opreme je moguće izvući potrebne informacije i s dovoljno visokom točnošću identificirati korisničke lozinke i pinove, kao i običan tekst, utipkane na tipkovnici [2]. U istraživačkom radu *Daniel Genkina, Adi Shamira i Eran Tromera* također je dokazano da se pomoću pametnog telefona, ako se postavi na točnu udaljenost od ventilacijskih otvora prijenosnog računala, vidljivo na slici 2.1, mogu snimiti te dešifrirati RSA kriptografski ključevi [1].



Slika 2.1 Snimanje zvukova iz ventilacijskih otvora prijenosnog računala koristeći pametni telefon (*Acoustic Cryptanalysis - Daniel Genkin, Adi Shamir i Eran Tromer* [1])

Završetkom snimanja i prikupljanja zvukova odabrane mete, napadač obavlja analizu istih kako bi izvukao korisne informacije pomoću kojih može dešifrirati osjetljive podatke. Analiza može uključivati traženje specifičnih uzoraka ili frekvencija u zvuku, ili traženje bilo kakvih anomalija ili nepravilnosti unutar snimljenog zvuka. Neke od korištenih tehnika analize zvuka su **obrada signala** i **statistička analiza**.

Tehnike obrade signala koriste se za izdvajanje specifičnih značajki ili uzoraka iz snimljenog zvuka, kao što su filtriranje signala kako bi se uklonio šum iz snimljenog zvuka, primjena Fourierove analize za identificiranje specifičnih frekvencija ili korištenje algoritama za prepoznavanje uzoraka kako bi se otkrili specifični uzorci u snimljenom zvuku. U istraživačkom radu *Dmitri Asonova i Rakesh Agrawala* [2] korištena je još i brza Fourierova transformacija [14] za ekstrakciju značajki unutar snimljenog zvuka, a cijeli proces je prikazan na slici 2.2. Za utvrđivanje korelacija ili anomalija u snimljenom zvuku koriste se tehnike statističke analize. One uključuju izračun srednje vrijednosti, varijance ili drugih statističkih mjera zvuka, ili korištenje algoritama strojnog učenja za otkrivanje uzoraka koji nisu odmah vidljivi ljudskom oku.



Slika 2.2 Korištenje brze Fourierove transformacije za ekstrakciju značajki iz snimljenog zvuka (*Keyboard Acoustic Emanations - Dmitri Asonov i Rakesh Agrawal* [2])

Na kraju kada su korisne informacije izvučene iz zvuka, napadač može upotrijebiti te informacije za dešifriranje šifriranih poruka kao i probijanje kriptosustava. Također napadač može otkriti korisničke lozinke kao i pinove nakon cijelog procesa akustične kriptoanalize. Dešifriranje poruka i tajnih informacija je moguće korištenjem različitih tehnika kao što su tehnike statističke analize ili algoritmi strojnog učenja korišteni upravo u te svrhe.

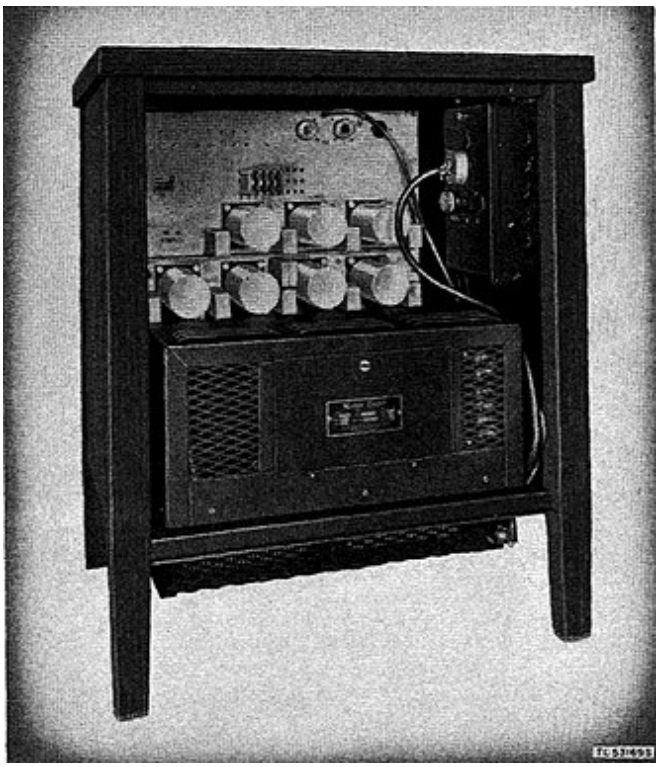
2.2 Povijest

Povijest napada akustičnom kriptoanalizom datira još od ranih dana primjene računala. Pedesetih i šezdesetih godina prošlog stoljeća istraživači su otkrili da elektronički sklopovi emitiraju zvučne valove prilikom obavljanja raznih računalnih operacija. Ti su zvučni valovi obično na frekvencijama van raspona ljudskog sluha, koji je otprilike od 20 Hz do 20 kHz, ali se mogu detektirati i analizirati pomoću specijalizirane opreme.

Prvi dokumentirani primjer korištenja akustične kriptoanalize za izvlačenje informacija iz elektroničkih uređaja zabilježen je 1950-ih kada su inženjeri, radeći na UNIVAC računalu, u Agenciji za nacionalnu sigurnost (skraćeno NSA) slučajno otkrili da mogu prisluškivati izvršavanje operacija na računalu pomoću mikrofona. U to vrijeme, inženjeri su pratili temperature vakuumskih cijevi računala koristeći specijalizirani mikrofon za tu namjenu. Osim uobičajenih podataka o temperaturama vakuumskih cijevi, inženjeri su primijetili da mikrofon također hvata zvukove koje proizvode sklopovi računala. Pobližom opservacijom prikupljenih zvukova, otkrili su da zvuk koji proizvode sklopovi varira ovisno o operacijama koje izvodi računalo, te analizom tih varijacija moguće je zaključiti o prirodi operacija koje se izvode. Inženjeri su shvatili da se ova tehnika može koristiti za izvlačenje osjetljivih informacija iz računala, poput sadržaja njegove memorije ili instrukcija koje se izvršavaju. Novootkrivenu tehniku nazvali su "akustički nadzor" te su počeli eksperimentirati s njom. Međutim, brzo su shvatili da njihova novootkrivena tehnika ima i značajna ograničenja. Za provođenje tehnike potreban je visokoosjetljivi mikrofon kako bi se uhvatili slabi zvukovi koje proizvode sklopovi unutar računala te je također te zvukove često bilo i teško protumačiti. Osim navedenog, tehnika je uvelike ovisila o hardveru koji se koristio, a ako bi se napravile nekakve promjene na hardveru, tehnika bi postala beskorisna. Unatoč navedenim ograničenjima, NSA je nastavila eksperimentirati s akustičkim nadzorom te je razvila niz tehnika za analizu zvukova koje proizvode elektronički uređaji. Te tehnike uključivale su korištenje analizatora spektra za vizualizaciju samog frekvencijskog spektra prikupljenih zvukova i korištenje algoritama za prepoznavanje uzoraka kako bi se identificirali specifični uzorci u prikupljenim zvukovima koji odgovaraju određenim operacijama [15].

Iako raniji napad akustične kriptoanalize nije bio izveden u zlonamjerne svrhe, pokazao je potencijal korištenja zvučnih emisija za izvlačenje informacija iz elektroničkih uređaja. Osim toga, godine 1964. američka protuobavještajna služba pronašla je 64 mikrofona i veliku metalnu rešetku na stropu svoje ambasade u Moskvi. Sve navedeno je dovelo do razvoja napada, ali i danas poznate tehnologije koja uključuje nadzor i zaštitu uređaja koji emitiraju elektromagnetsko zračenje, na način koji se može koristiti za rekonstrukciju razumljivih podataka, pod kodnim imenom TEMPEST (*Transient Electromagnetic Pulse Emanation STandard*). Napad se sastoji od više vrsta napada sporednim kanalom od kojih su najznačajniji akustična kriptoanaliza, kojom se analiziraju dohvaćene akustične emanacije i vibracije koje proizvodi elektronički uređaj, i elektromagnetska analiza, koja se smatra glavom vrstom napada za izvršavanje TEMPEST napada. Američka vlada prvi put je upotrijebila akronim 1960-ih kako bi opisala problematiku elektroničkih uređaja da emitiraju radio valove (elektromagnetska zračenja) koji se mogu presresti i analizirati za potrebe izvlačenja osjetljivih informacija. Već 1940-ih koncept TEMPEST napada bio je otkriven kada su istraživači otkrili da elektronički uređaji poput radija i televizije mogu emitirati neželjeno elektromagnetsko zračenje koje se može prikupiti i analizirati kako bi se izvukli audio ili video signali, koji bi se mogli akustičnom kriptoanalizom dalje analizirati kako bi se dodatno izvukle osjetljive informacije [3]. Kako je američka vlada s vremenom postala sve zabrinutija zbog mogućnosti da TEMPEST napadi ugroze nacionalnu sigurnost, osobito u kontekstu vojnih komunikacija i prikupljanja obavještajnih podataka, razvili su niz protumjera koje uključuju tehnike zaštite i filtriranja kako bi se spriječilo neželjeno zračenje elektroničkih uređaja. Uz to, razvili su i specijalizirane programe testiranja i certifikacije kako bi se osiguralo da elektronički uređaji, koje koristi vlada, zadovoljavaju određene sigurnosne standarde. Kako je već prije navedeno, ove vrste zaštite također su dobile ime TEMPEST [16].
U 1950-ima, CIA (*Central Intelligence Agency*) otkrila je da može oporaviti originalni tekst navodno 'sigurnih' vojnih komunikacija na modelu Bell 131-B2 prikazanom na slici 2.3, sigurnom stroju za enkripciju komunikacije vojske i mornarice koji koristi 'neraskidive' jednokratne trake, osluškujući preko 400 metara daleke električne vodove. To je bilo nakon što je stroj već modificiran 1940-ih. Izvorna verzija stroja je pokazala do 75% pritisnutih tipki te ih se moglo dešifrirati na osciloskopu koji se nalazio u blizini, unutar laboratoriju. Curenje podataka je bilo uzrokovano ra-

diofrekvencijskim emisijama iz električnih kontakata u relejima, indukcijskim signalima na komunikacijskoj mreži koji su se mogli uhvatiti i na udaljenosti do 1,6 km, te elektromagnetskim curenjem iz zavojnica u relejima, na koje se moglo uhvatiti i na udaljenosti od oko 9 metra od samog stroja [11].



Slika 2.3 Stroj za enkripciju Bell 131-B2 [3]

ENGULF je bilo kodno ime za niz operacija u kojima je britanska sigurnosna služba, MI5 (*Millitary Intelligence, Section 5*), presrela egipatske i francuske šifre u razdoblju od sredine 1950-ih do sredine 1960-ih. Prvi napad dogodio se tijekom Sueske krize, godine 1956. Britanska sigurnosna služba MI5 izvela je napad na egipatske strojeve za šifriranje, poznate kao Hagelin, koje je egipatska vlada koristila za osiguranje svojih diplomatskih komunikacija, kako bi presrela i dešifrirala komunikaciju u tijeku. Napad je bio dio zajedničke operacije britanskog Vladinog komunikacijskog stožera i američke Nacionalne sigurnosne agencije (skraćeno NSA), poznate kao Operacija RAFTER. Egipatski strojevi za šifriranje bili su vrsta elektromehaničkog

uređaja koji je koristio rotirajuće kotače za šifriranje i dešifriranje poruka. Strojevi su se u to vrijeme smatrali vrlo sigurnima, ali su bili ranjivi na tehniku poznatu kao napad sporednim kanalom, koja je uključivala analizu neželjenih signala koje je stroj emitirao tijekom rada. Napad MI5 uključivao je instaliranje mikrofona unutar zidova egipatskog veleposlanstva u Londonu, gdje su se koristili strojevi za šifriranje. Mikrofoni su korišteni za snimanje zvukova koje proizvode strojevi dok rade, uključujući zvukove rotirajućih kotača i električne signale koje stvara stroj, ali i slabe zvukove poput škljocanja i zujanja unutarnjih zupčanika i mehanizama. Koristeći tehnike obrade signala, analitičari MI5 uspjeli su izvući informacije o internom radu strojeva za šifriranje, uključujući položaje rotirajućih kotača i električne signale koje stroj generira. S tim informacijama uspjeli su razviti matematički model stroja za šifriranje koji se onda koristio za dešifriranje presretnutih poruka. Napad MI5 na egipatske strojeve za šifriranje Hagelin bio je značajno postignuće u povijesti kriptoanalize, budući da je pokazao potencijal korištenja napada sporednim kanalom, pogotovo akustične kriptoanalize, za razbijanje sustava šifriranja. Napad je također istaknuo važnost fizičke sigurnosti u zaštiti osjetljivih informacija, budući da je ranjivost strojeva za šifriranje bila posljedica neželjenih signala koje su emitirali tijekom rada, a ne bilo kakve slabosti u samom algoritmu šifriranja. Međutim, operacija je također izazvala etičku zabrinutost u vezi s korištenjem tajnih tehnika prikupljanja obavještajnih podataka, osobito u kontekstu diplomatskih odnosa između zemalja. Upotreba akustične kriptoanalize za prisluškivanje komunikacija egipatskog veleposlanstva bila je posebno kontroverzna, budući da je uključivala ugrožavanje sigurnosti komunikacija strane vlade i potencijalno kršenje međunarodnih zakona i normi [17].

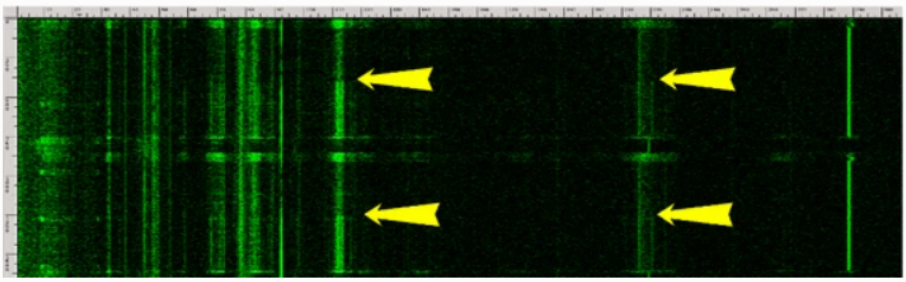
Dmitri Asonov i Rakesh Agrawal 2004. godine su objavili rad u kojem su dokazali ranjivost računalnih, ali i ostalih, tipkovnica. Koristeći jeftinu opremu (mikrofon i parabolički mikrofon) za snimanje tipkanja na računalnoj tipkovnici, kao i tipkovnici na bankomatu, te koristeći neuronsku mrežu za analiziranje snimljenog zvuk tipkanja kao i Fourierove transformacije za ekstrakciju značajki unutar zvuka, uspjeli su dokazati da postoji razlika u zvuku koji proizvodi svaka tipka na tipkovnici. Također njihov eksperiment je sa 79% uspješnosti uspio vratiti originalno upisan tekst, no veći uspjeh je imao pri kraćim riječima kao što su lozinke ili pinovi, nego s ci-

jelim tekstom. Također mane njihovog eksperimenta su neprimjenjivost istrenirane neuronske mreže na druge modele tipkovnica ili čak druge tipkovnice istog modela. Također, ukoliko bi se promijenila osoba koja bi obavljala tipkanje na tipkovnici, na kojoj je provedeno testiranje, model bi vraćao slabije rezultate [2].

Povedeni radom Asonova i Agrawala, Li Zhuang, Feng Zhou i J. D. Tygar su 2005. godine nadogradili Asonov i Agrawalov eksperiment. U svojem istraživanju, umjesto 300 klikova po tipki, odlučili su snimiti 10-minutnu snimku korisnika koji upisuje engleski tekst na tipkovnici. Zatim su tu istu snimku dali kao ulaz u neuronsku mrežu koja je uspjela u 96% točnosti vratiti originalno upisan tekst. Također, istraživanjem su utvrdili da njihov model može s 90% vratiti nasumičnu zaporku od 5 znakova koji su samo slova te može biti generirana u manje od 20 pokušaja, dok se s 80% točnosti mogu generirati 10-znakovne lozinke u manje od 75 pokušaja. Njihov napad koristi se i pravilima engleskog jezika (gramatika), kako bi rekonstruirao tekst iz zvučnih snimaka bez označenih trening podataka. Napad također koristi kombinaciju standardnih tehnika strojnog učenja i prepoznavanja govora, uključujući značajke kepstruma, skrivene Markovljeve modele, linearnu klasifikaciju i inkrementalno učenje temeljeno na povratnim informacijama [12].

Godine 2013. Daniel Genkin, Adi Shamir i Eran Tromer objavili su rad u kojem su demonstrirali svoj napad na implementaciju RSA u GnuPG-u (*GNU Privacy Guard*) koji radi na prijenosnom računalu. Postavili su osjetljivi mikrofon blizu računala i snimili zvukove proizvedene tijekom operacija dešifriranja. Zatim su analizirali snimljene zvukove kako bi izvukli informacije o privatnom ključu korištenom u enkripciji. Prikaz snimke zvuka i njegove analize vidljiv je na slici 2.4. Istraživači su također 2016. godine ponovili eksperiment. Iako su bolje rezultate dobili s osjetljivijim mikrofonima, isti su napad proveli i s mobilnim uređajem smještenim u blizini ventilacijskih otvora prijenosnog računala. Rezultati su vratili da se korištenjem i mobilnog uređaja, ako je postavljen na točnu udaljenost, u potpunosti mogu prikupiti informacije o privatnom ključu te ga se može i rekonstruirati. Također su primijetili da je napad bio uspješan čak i kada je prijenosno računalo bilo smješteno u zvučno izoliranu kutiju ili serversku sobu, što ukazuje da same fizičke sigurnosne mjere možda neće biti dovoljne za zaštitu od napada akustičnom kriptoanalizom.

Napad je bio uspješan u izvlačenju tajnog ključa korištenog u enkripciji s točnošću od 96%, što pokazuje ranjivost RSA enkripcije na napade akustičnom kriptoanalizom [1].



Slika 2.4 Akustični potpis dva GnuPG RSA potpisa (žutim strelicama označeni su prijelazi između p i q) [1]

U novije vrijeme mobilni telefoni su sve rašireniji i čak više korišteni od računala, no ni oni nisu sigurni od napada akustičnom kriptoanalizom. Sa svojim ogromnim rasponom uvijek uključenih senzora, posebno su izloženi riziku kada je riječ o napadima sporednim kanalom koji ugrožavaju privatnost. Zapravo sve veće kompanije koriste upravo te napade kako bi strateški preporučili stvari svakom korisniku ili prikazali točno željene reklame. Često se koristi zvuk u rasponu između 18 i 20 kHz (otprilike čujan za mlađe ljude kad nema pozadinske buke) upravo kako bi se prikupile informacije o korisniku. Akcelerometri se također koriste u napadima akustičnom kriptoanalizom. Upravo oni se mogu koristiti kao mikrofoni u samom napadu, ali također mogu se i koristiti za ometanje sustava. Nedavno su istraživači pokazali da se akustična kriptoanaliza može koristiti za kompromitiranje mobilnih uređaja, poput pametnih telefona i tableta. Godine 2014. tim istraživača s kalifornijskog sveučilišta Berkeley pokazao je da mogu koristiti akcelerometar ugrađen u pametni telefon za detektiranje vibracija uzrokovanih pritiskom na tipku i korištenje tih vibracija kako bi zaključili koji tekst je bio tipkan. Istraživači su pokazali da zvuk koji proizvodi telefon varira ovisno o pritisku na tipku te da je analizom tih varijacija moguće izvući osjetljive informacije [8].

Poglavlje 3

Metodologija

3.1 Oprema i metoda tipkanja

Za odradu napada korištene su mehaničke tipkovnice ***Akko World Tour-Tokyo R1 3087 s Akko 2nd Gen Pink*** **prekidačima** i ***SHINOBI White s OUTEMU Blue*** **prekidačima**. Za snimanje tipkanja na obje tipkovnice korišten je ***Audio-Technica AT 2035 kondenzatorski*** mikrofon, ***Mackie ProFX6*** analogna mikseta s USB konekcijom te ***Reaper*** aplikacija za snimanje i obradu audio podataka. Snimke tipkanja su prikupljene od jednog tipkača. Ukupno je prikupljeno **66,53 minuta** tipkanja na obje tipkovnice te je isto odrađeno u relativno idealnim uvjetima: snimanje u zatvorenoj sobi s minimalnim smetnjama zvukova iz okoline. Postava opreme za izvođenje snimanja tipkovnica, kao i tipkovnice, prikazane su na slici 3.1.

Metoda tipkanja odabrana u ovom radu je **metoda traženja i kuckanja** *(eng. hunt-and-peck)*. U ovoj metodi slušani tipkač tipka na tipkovnici koristeći po jedan prst sa svake ruke, prvo tražeći željeno slovo na tipkovnici, a nakon njegovog pronalaska, tipkač klikne prstom željeno slovo [18]. Ova metoda je odabrana kako bi se lakše odradila obrada snimki te se zvukovi pojedinih tipki ne bi previše preklapali, što bi također otežalo samu analizu krajnjih snimki.

Slika 3.1 Postava opreme za snimanje tipkovnica

3.2 Podaci

Kao što je već prethodno spomenuto, prikupljeno je 66,53 minuta snimki tipkanja. Ovaj broj uključuje ukupno vrijeme tipkanja na obje spomenute mehaničke tipkovnice. Na svakoj od tipkovnica je otipkan isti tekst s otprilike istom brzinom kako bi se rezultati mogli lakše usporediti. Osim .wav datoteke, potrebno je bilo spremiti i .txt datoteke originalnoga teksta koji je bio pisan prilikom snimanja tipkanja. Tipkane tekst datoteke se mogu razdvojiti u dvije grupe: **qwerty** i **rečenice**. Raspodjela količine podataka, izražena u minutama, za obje grupe prikazana je u tablici 3.1.

Tablica 3.1 Raspodjela podataka unutar grupa

Grupa podataka	**Količina podataka [minute]**
qwery	43,13
rečenice	23,4

U grupi **qwerty** nalaze se snimke i tekstualne datoteke u kojima se na razne načine tipkao cijeli abecedni dio tipkovnice, uz to uključujući i tipku za razmak, dok se unutar grupe **rečenice** nalaze razne tekstualne datoteke, kao i njihove pripadajuće snimke, napisane na engleskom jeziku. Odabran je engleski jezik, ne samo zbog manjeg broja slova u samoj abecedi, već i zbog pokušaja rekreacije uspješnosti rada Li Zhuang, Feng Zhou i J. D. Tygara [12], ali postoji velika vjerojatnost da bi isti napad mogao biti uspješno odrađen i za hrvatski jezik i abecedu. Potrebno je naglasiti da grupu **qwerty** je bilo potrebno kreirati pošto vjerojatnosti korištenja slova engleske abecede unutar samog jezika nisu podjednake, što je vidljivo iz tablice 3.2, te ukoliko bi se samo rečenice koristile za učenje neuronske mreže, koja će biti predstavljena kasnije u radu, vjerojatnosti raspoznavanja slova na tipkovnici ne bi bile ispravne tj. slova s manjom frekvencijom pojavljivanja u engleskom jeziku ne bi bila ispravno predviđena.

Tablica 3.2 Vjerojatnosti pojavljivanja slova u engleskom jeziku [19]

Slovo	**Vjerojatnost pojavljivanja (izražena u %)**
A	8.34
B	1.54
C	2.73
D	4.14
E	12.60
F	2.03
G	1.92
H	6.11
I	6.71
J	0.23
K	0.87
L	4.24
M	2.53
N	6.80
O	7.70
P	1.66
Q	0.09
R	5.68
S	6.11
T	9.37
U	2.85
V	1.06
W	2.34
X	0.20
Y	2.04
Z	0.06

3.3 Priprema okruženja za eksperiment

Ovaj eksperiment odrađen je na operacijskom sustavu **Windows** te je napisan u programskom jeziku **Python** unutar **Anaconda** virtualnog okruženja kako ne bi bilo problema s verzijama Pythona i paketa koji se već nalaze na računalu. Odabrana je 3.7 verzija Pythona jer na toj verziji sve potrebne knjižnice rade bez ikakvih problema s verzijama paketa s kojima imaju zavisnost. Kod eksperimenta dostupan je na poveznici: **https://github.com/baobab1808/Diplomski-Rad.git**. Prije kreiranja

Anaconda virtualnog okruženja, potrebno je preuzeti kod na vlastito računalo.

Kako bi se napravilo Anaconda virtualno okruženje, potrebno je napisati sljedeću naredbu unutar naredbenog retka *(eng. command prompt)*:

```
conda create -n audiocrypt python=3.7
```

Poslije uspješno kreiranog virtualnog okruženja, potrebno je ući u novokreirano okruženje naredbom:

```
conda activate audiocrypt
```

Nakon ulaska u virtualno okruženje, potrebno je instalirati sve potrebne Python knjižnice koje se koriste unutar koda. Lista knjižnica prikazana s naredbom ***conda list*** prikazana je u tablici 3.3.

Tablica 3.3 Popis knjižnica i njihovih verzija

Ime	**Verzija**	**Ime**	**Verzija**
appdirs	1.4.4	audioread	3.0.0
bracex	2.3.post1	ca-certificates	2023.05.30
certifi	2022.12.7	click	8.1.6
greenlet	2.0.2	html5lib	1.1
inexactsearch	1.0.2	lazy-loader	0.3
librosa	0.10.0.post2	llvmlite	0.39.1
msgpack	1.0.5	numba	0.56.4
numpy	1.21.6	matplotlib	3.5.3
openssl	1.1.1v	path	16.6.0
pathlib	1.0.1	pip	23.2.1
pooch	1.6.0	python	3.7.16
pyyaml	6.0.1	regex	2023.8.8
scipy	1.7.3	setuptools	68.0.0
silpa-common	0.3	soundex	1.1.3
soundfile	0.12.1	soxr	0.3.2
torch	1.13.1	torchsummary	1.5.1
torchviz	0.0.2	tqdm	4.66.1
typing-extensions	4.7.1	vc	14.2
vs2015_runtime	14.27.29016	wavio	0.0.7
wcmatch	8.4.1	wheel	0.41.1
wincertstore	0.2	wrapt	1.14.1

Svaka knjižnica se instalira pomoću ***pip install*** naredbe, a u primjeru je naredba napisana za instalaciju *wavio* knjižnice:

```
pip install wavio
```

Nakon instalacija potrebnih knjižnica moguće je pokrenuti kod. Primjeri pokretanja koda prikazani su u nastavku:

1. Pokretanje datoteke za učitavanje .wav datoteke šaljući i .txt datoteku s oznakama:

```
    python readKeystrokes.py ..\recordings\sentence9_w.wav
                                ..\recordings\sentence9.txt

```

2. Pokretanje datoteke za učitavanje .wav datoteke bez slanja .txt datoteke s oznakama:

```
    python readKeystrokes.py ..\recordings\sentence9_w.wav
                                ..\recordings\sentence9.txt

```

3. Pokretanje datoteke za klasifikaciju i treniranje neuronske mreže s dobivenim podacima iz prve ili druge točke:

```
    python classifyKeystrokes.py ..\out\keystrokes\
                                  keyboards\p\sentence1_p.
                                  wav_out

```

4. Pokretanje datoteke za klasifikaciju i treniranje neuronske mreže iz željene mape koja je definirana unutar koda:

```
    python classifyKeystrokes.py

```

Pri završetku korištenja virtualnog okruženja potrebno ga je isključiti naredbom:

```
conda deactivate
```

3.4 Algoritam napada

Algoritam napada akustičnom kriptoanalizom u ovom će radu biti predstavljen kroz svoje tri faze:

1. **Ekstrakcija zvukova tipki iz .wav snimki na kojima tipka osoba**
2. **Ekstrakcija značajki iz individualnih zvukova tipki**
3. **Treniranje neuronske mreže s ciljem raspoznavanja razlike između zvukova različitih tipki te vraćanje predviđenog teksta kao izlaz**

Faze algoritma su slikovno prikazne na slici 3.2.

Osim demonstracije napada, napravljena je i usporedba između dvije mehaničke tipkovnice kako bi se provjerilo ako korištenje drugačijega prekidača tipkovnice *(eng. switch)* utječe na uspješnost napada.

Slika 3.2 Algoritam faza napada

3.4.1 Ekstrakcija zvukova tipki iz .wav snimki

Za ekstrakciju zvukova tipki iz .wav snimke potrebno je snimiti korisnika kako tipka na tipkovnici.
Ekstrakcija zvukova tipki odrađuje se po sljedećim koracima:

1. **Čitanje .wav datoteke i obrada audio podataka**
2. **Otkrivanje zvukova tipki unutar obrađenih audio podataka pomoću vrhova *(eng.peaks)***

Kako bi se mogla pročitati .wav datoteka i na njoj odraditi sva potrebna pre-

dobrada za daljnje korištenje, kreirana je ***wav_read*** funkcija koja je prikazana u nastavku, a sam opis rada funkcije je opisan u daljnjem tekstu.

```
def wav_read(filepath):

    data = wavio.read(filepath)
    data = data.data
    if type(data[0]) == np.ndarray:
        data = data[:,0]
    data = data.astype(np.float32)
    data = (data / np.max(np.abs(data)))
    data -= np.mean(data)

    return data
```

Čitanje .wav datoteka u Pythonu odrađuje se pomoću modula ***wavio***. Ovaj Python modul sadrži u sebi funkciju ***wavio.read*** koja čita WAV datoteku i vraća objekt koji sadrži brzinu uzorkovanja, širinu uzorka u bajtovima i NumPy polje u kojem se nalaze podaci iz WAV datoteke [20]. Nakon čitanja WAV datoteke, podaci su zapisani kao 2D NumPy polje gdje prva dimenzija predstavlja broj audio kanala koji govori na koliko je kanala audio signal snimljen dok druga dimenzija koja predstavlja audio uzorke. Prije daljnje obrade podataka potrebno je provjeriti jesu li podaci jednokanalni *(eng. mono)* ili višekanalni *(eng. stereo)*. Ako su višekanalni, potrebno je uzeti samo jedan kanal iz podataka pošto daljnja analiza audio podataka bi se poprilično otežala ako bi se ekstrakcija zvukova tipki radila na dva odvojena audio kanala i iz dva audio signala u isto vrijeme, što je glavna značajka višekanalnog signala [21]. Uz sve navedeno, potrebno je još dodatno postaviti da svi podaci budu tipa ***np.float32*** s obzirom na to da nije potrebna velika preciznost u decimalama te se taj tip podatka često koristi u radu s audio signalima. Nakon toga, podaci trebaju biti normalizirani u rasponu od [-1, 1] te centrirani oko nule kako kasnije obrade podataka ne bi preferirale jedne značajke nad drugima ili kako se kasnije u neuronskoj mreži ne bi nulti gradijenti natrag propagirali u mrežu prije nego je to očekivano.

Nakon uspješnog učitavanja audio podataka i njihove predobrade, moguće je pri-

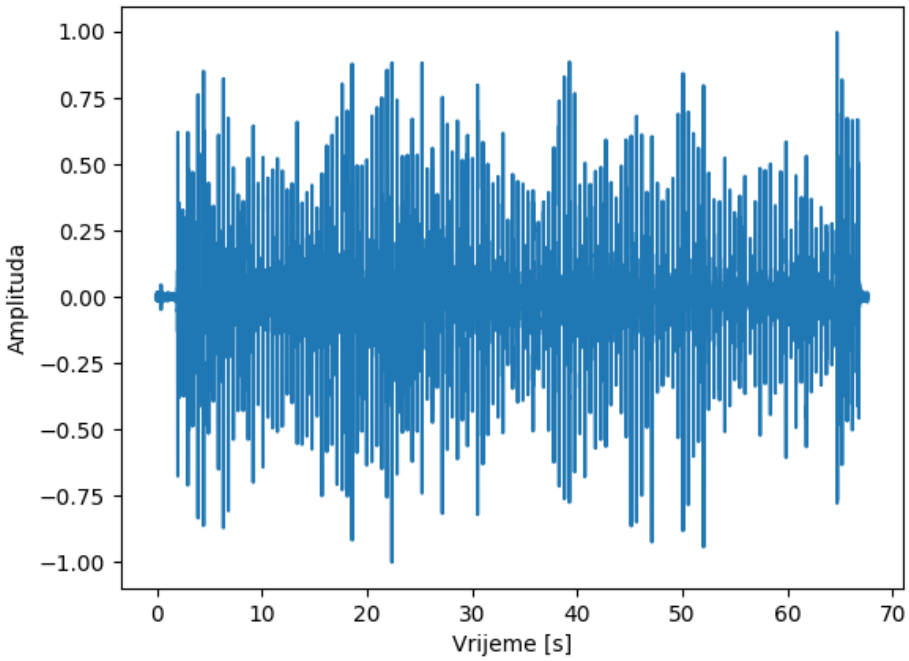
jeći na idući korak koji je otkrivanje zvukova tipki unutar obrađenih audio podataka. Prema radu L. Zhuang, F. Zhou i J.D. Tygara, prosječan korisnik može natipkati oko 300 znakova u minuti. Sam pritisak tipke se može podijeliti na dva glavna zvučna dijela: pritisni vrh i otpuštani vrh *(eng. push peak and release peak)* [12]. Ono što su L. Zhuang, F. Zhou i J.D. Tygar potvrdili u svom radu, a što su D. Asonov i R. Agrawal dokazali u svom radu, je da vremenski razmak između navedenih vrhova traje otprilike 100 ms [2]. Iz toga je moguće zaključiti da pritisak tipke mora minimalno trajati 100 ms, a u ovom radu eksperimentom je utvrđeno da razmak između dva različita pritiska iznosi minimalno 200 ms, stoga se za razliku od navedenih prijašnjih radova, gdje se koristio samo pritisni vrh kako bi se izvukle karakteristike pojedine tipke, u ovom radu koriste oba vrha jer pridodaje informacijama o samim karakteristika tipke. Kako bi se otkrili zvukovi tipki, potrebno je prvo odrediti gdje se nalaze pritisni vrhovi koji su ujedno i maksimalne vrijednosti amplitude te tipke. Kako bi se pronašle navedene maksimalne amplitude, potrebno je ponašati se prema njima kao prema lokalnim maksimumima. Prolaskom kroz podatke potrebno je usporediti svaku vrijednost podatka na pronađenom lokalnom maximumu sa zadanim pragom *(eng. threshold)* te ako je veća, potrebno je lokaciju lokalnog maksimuma nadodati u listu svih trenutno pronađenih vrhova. Prag je ovdje igrao veliku ulogu jer pomoću njega se moglo odvojiti koje vrijednosti zvukova su proizvedene od strane tipke na tipkovnici, a koje su samo pozadinska buka, što uvelike olakšava daljnji proces otkrivanja zvukova tipki. Nakon prolaska kroz sve dane podatke i prikupljenih svih pozicija lokalnih maksimuma vrhova, potrebno je proći po svim lokalnim maksimumima, umanjiti poziciju za 1440 kako bi se dobila početna pozicija početka tipke u snimci. Eksperimentom je određeno da većina maksimalnih amplituda zvukova tipki, što je trenutak pritiska tipke, se nalazi oko 1440 pozicije, stoga je odlučeno da je ovaj broj univerzalan za dobivanje početne pozicije tj. početka zvuka tipke na snimci. Nakon postavljene početne pozicije zvuka tipke u podacima, potrebno je izračunati poziciju kraja zvuka tipke. Kao što je već ranije spomenuto, vremenski razmak između dva pritiska različitih tipki iznosi oko 200 ms, stoga se krajnja pozicija računa tako da početnoj poziciji pribrojimo umnožak između vremenskog razmaka između dvije tipke i brzine uzorkovanja, a koja je postavljena na standardnu vrijednost od 44100 Hz. Ova brzina uzorkovanja se koristi zato što čovjekovo uho može čuti frekvencije između 20 i 20000 Hz, kao što je već prije u radu spomenuto, no

da bi računalo rekreiralo te frekvencije, mora koristiti dvostruko veću brzinu uzorkovanja. Ta minimalna brzina uzorkovanja se zove **Nyquistova brzina** i prema kriteriju, ako brzina uzorkovanja nije veća od Nyquistove brzine, pojavit će se pojava zvana preklapanje frekvencija komponenti signala *(eng. aliasing)*, što dovodi do izobličenja signala prilikom rekonstrukcije iz uzoraka, rezultirajući razlikom između rekonstruiranog i izvornog signala [22]. Nakon postavljenih početnih i krajnjih pozicija zvukova tipki u podacima, potrebno je spremiti svaki od zvukova, u rasponima od početne do krajnje pozicije, kao zasebnu listu podataka koji sačinjavaju jedan zvuk tipke. Na kraju potrebno je povezati sve zvukove tipki, kao i njihove oznake tj. koji znak je bio pritisnut u tom trenutku, u jednu listu koja će sadržavati sve tipke i sve oznake unutar te audio datoteke. Funkcije ovih dvaju opisanih postupaka prikazane su u nastavku. Funkcija ***find_peaks*** pronalazi maksimalne lokalne vrhove, dok funkcija ***detect_keystrokes*** otkriva zvukove tipki i izdvaja ih van.

```
def find_peaks(data, distance=1, threshold=0.1):
    peaks = []
    i = 0
    while i < len(data):
        max_loc = np.argmax(data[i:i+distance]) + i
        if data[max_loc] > threshold:
            peaks.append(max_loc)
            i = max_loc + distance
        else:
            i += 1000
    return peaks
```

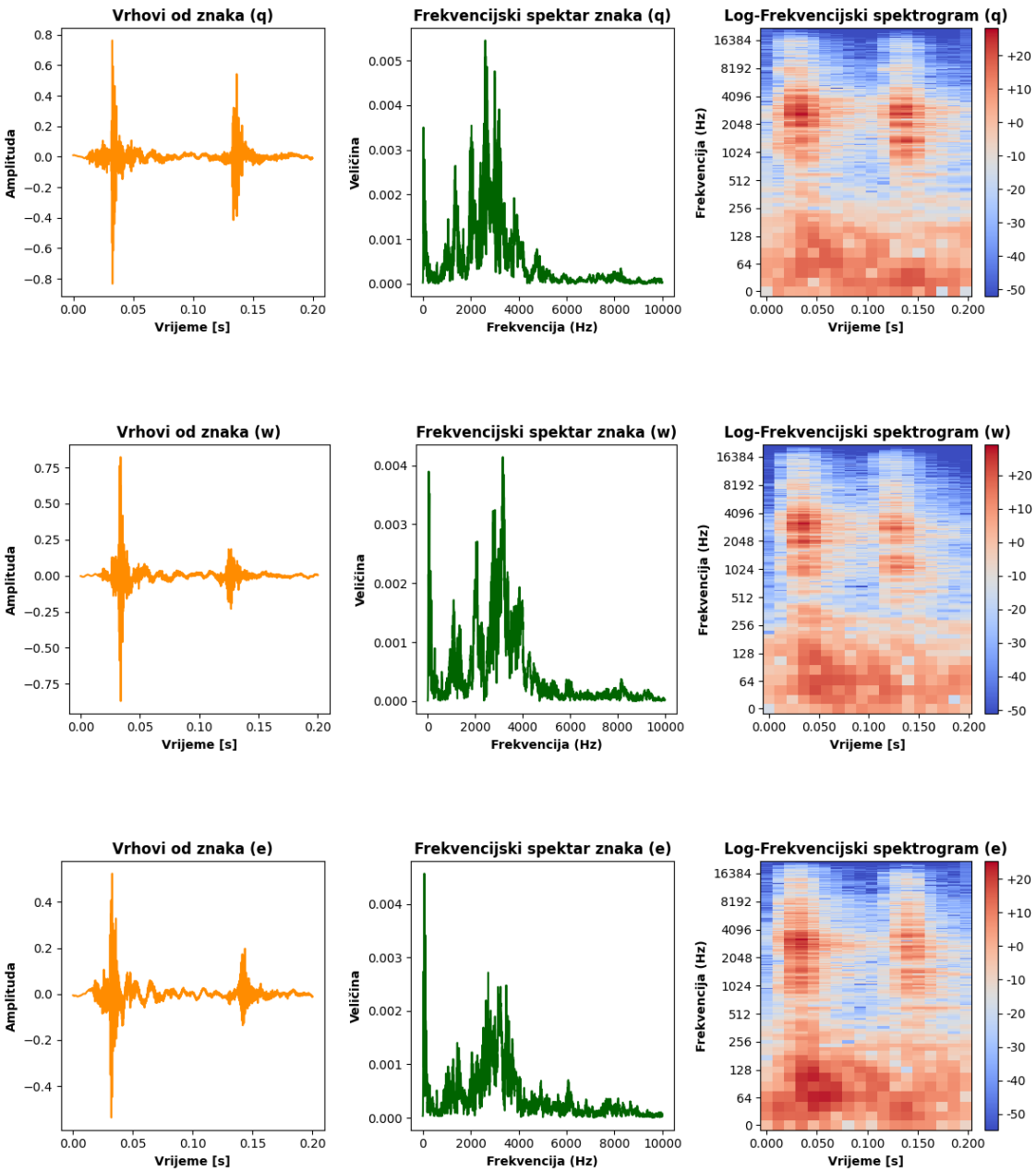
```
def detect_keystrokes(sound_data, sample_rate=SAMPLE_RATE,
                                  output=True, num_peaks = None,
                                  labels = None):
    keystroke_duration = 0.2
    len_sample = int(sample_rate * keystroke_duration)
    keystrokes = []

    peaks = find_peaks(sound_data, threshold=0.06, distance=
                                  len_sample)
    print(f"Found {len(peaks)} keystrokes in data")

    if not num_peaks:
        labels = [None for i in range(len(peaks))]

    for i, peak in enumerate(peaks):
        peak = peak - 1440
        start, finish = peak, peak + int(0.2 * sample_rate)

        if finish > len(sound_data):
            finish = len(sound_data)

        keystroke = sound_data[start:finish]
        keystrokes.append((keystroke.tolist(), labels[i]))

    return keystrokes
```

Iako kroz objašnjenu teoriju možda i dalje nije najjasnije kako je moguće da postoje razlike u zvuku između svake tipke, vizualizacija pritisnog i otpusnog vrha, kao i frekvencijskog spektra, svake tipke bolje dočarava tu razliku. U sljedećih nekoliko slika prikazana će biti vizualna reprezentacija tipkanja slova QWERTY tipkovnice. Na slici 3.3 prikazan je cijeli audio signal kroz vrijeme sa svim svojim amplitudama. Kao što je iz slike vidljivo, samo promatranje cijelog signala ne govori puno o razlikama između pritisnih i otpusnih vrhova svake tipke jer su zbijeni jedni do drugih.

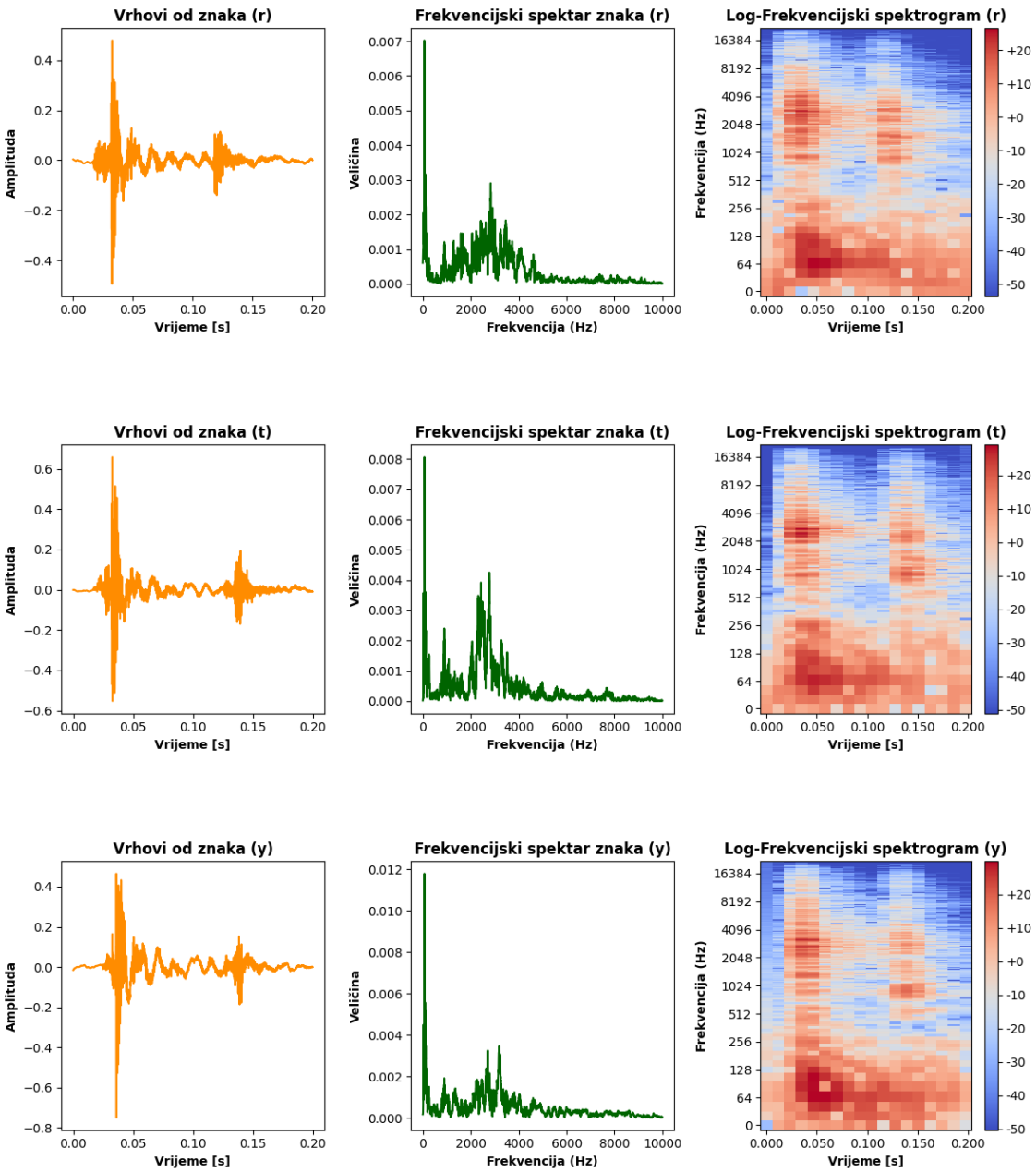


Slika 3.3 QWERTY audio signal - Amplitude signala kroz vrijeme

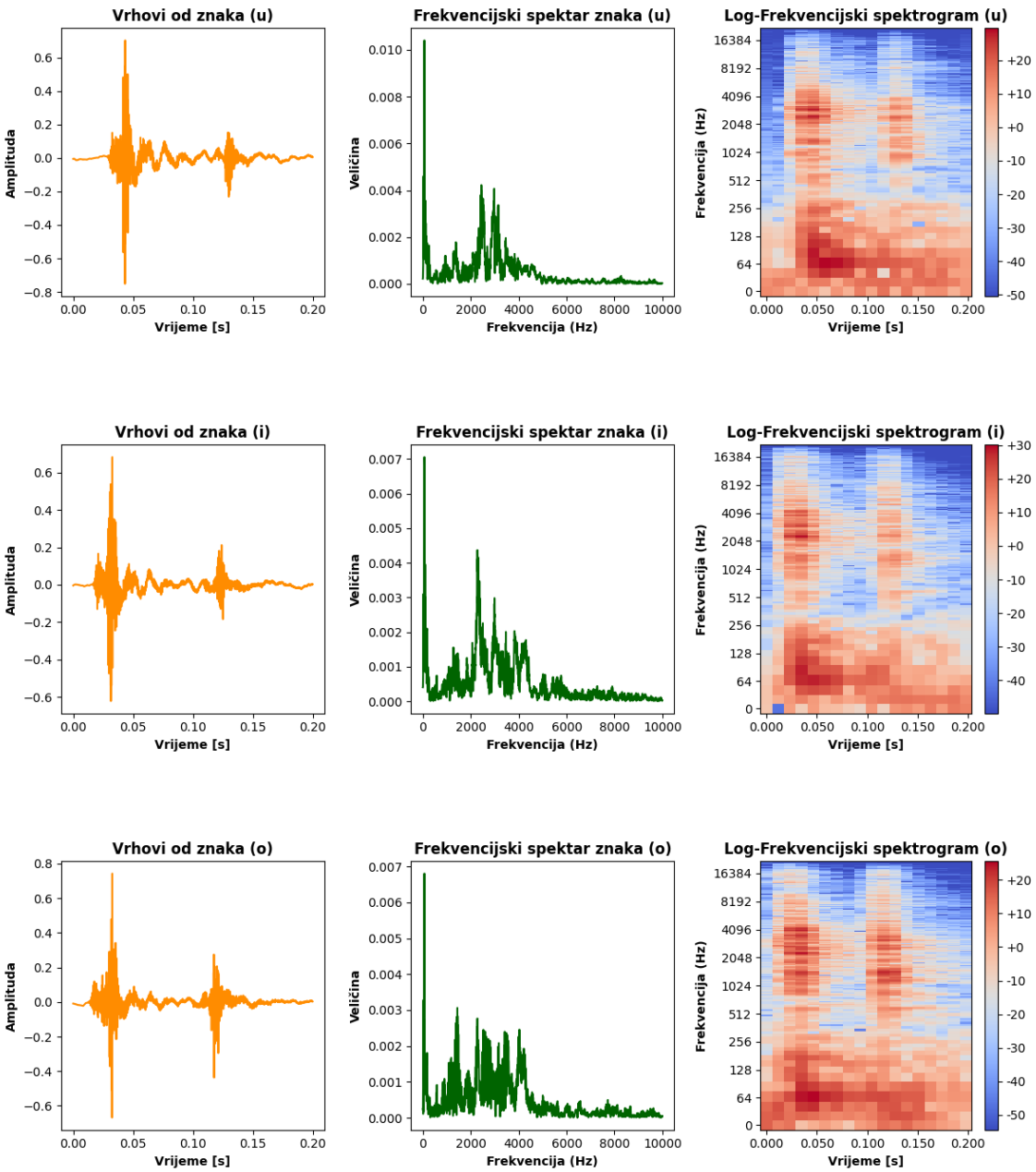
Iz tog razloga, vizualizacija pritisnih i otpusnih vrhova svakog slova engleske abecede na QWERTY tipkovnici, kao i njihovi frekvencijski spektri te log-frekvencijski spektrogrami prikazat će jasnije razliku između zvuka svake tipke. Na slikama 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 i 3.12 prikazano je svih 26 vrhova slova engleske, kao i znaka za razmak.



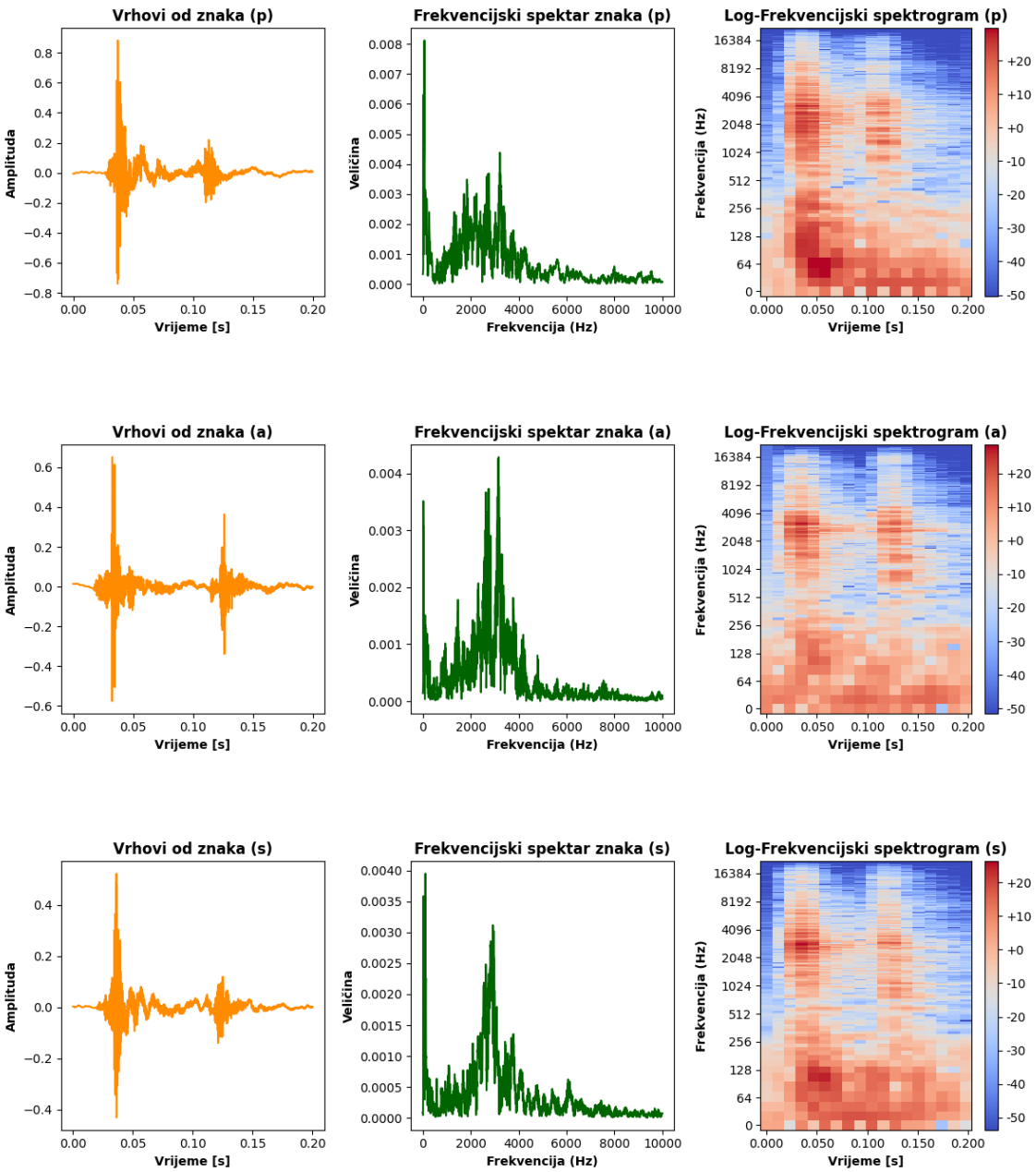
Slika 3.4 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: Q, W, E



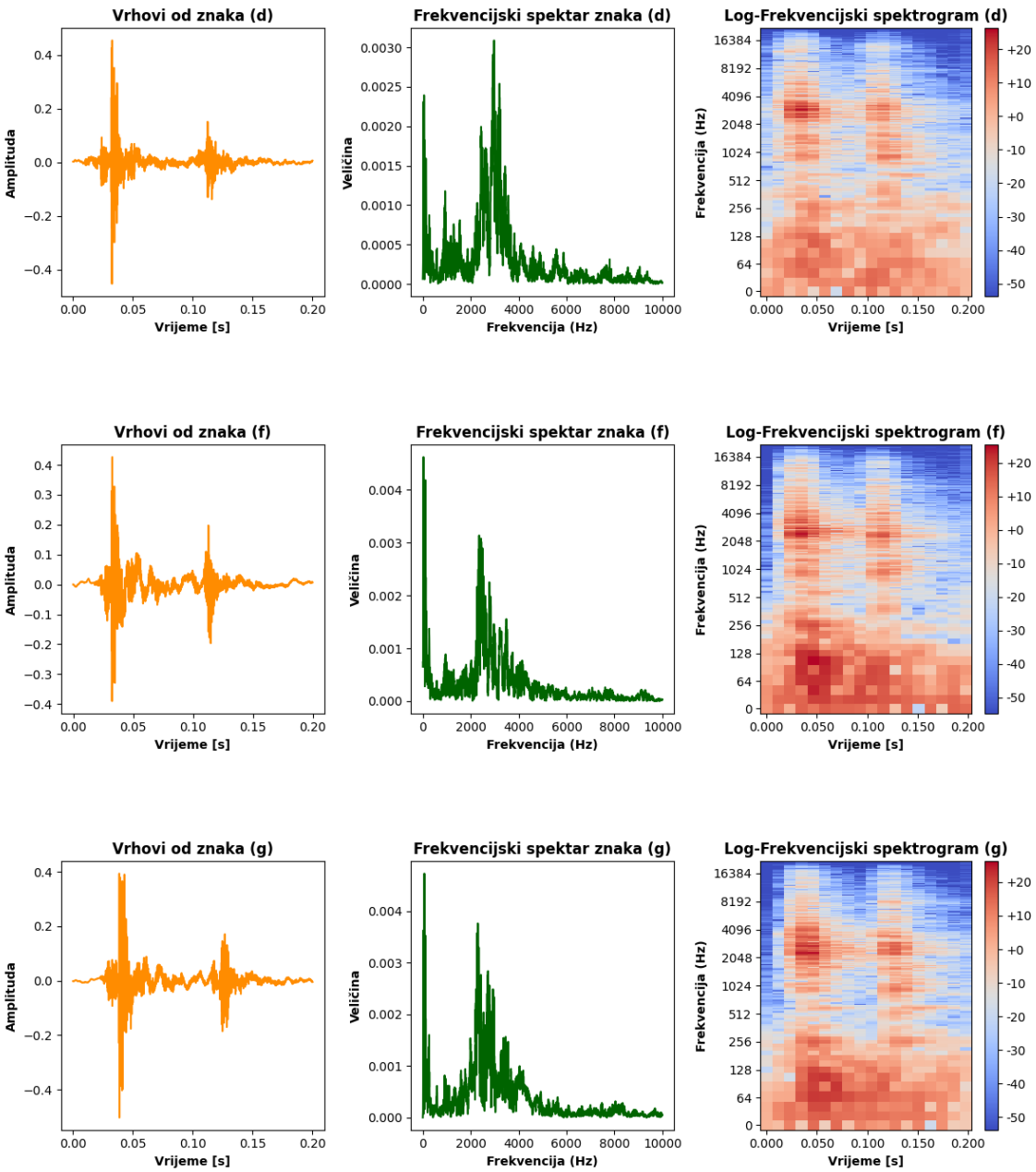
Slika 3.5 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: R, T, Y



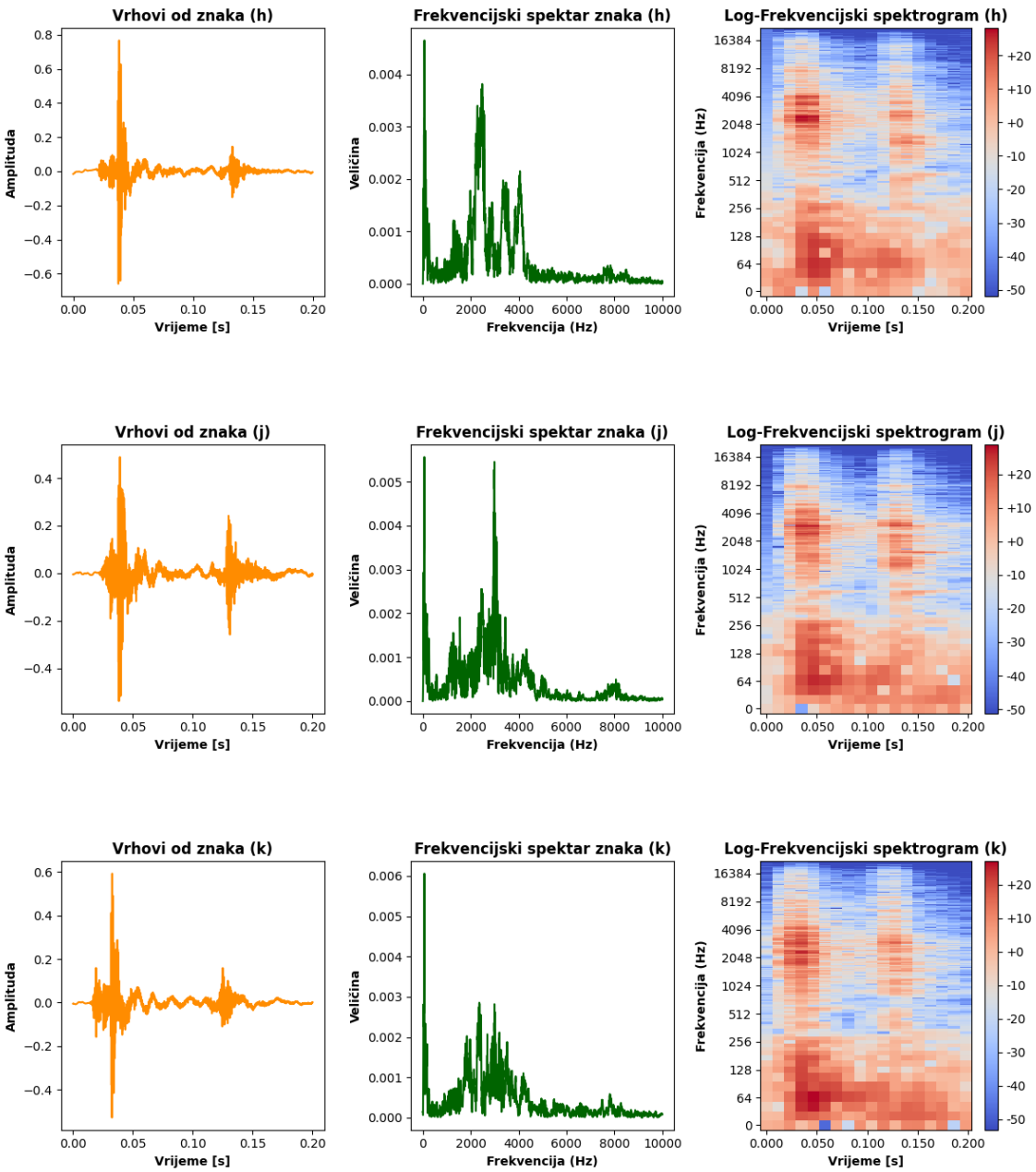
Slika 3.6 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: U, I, O



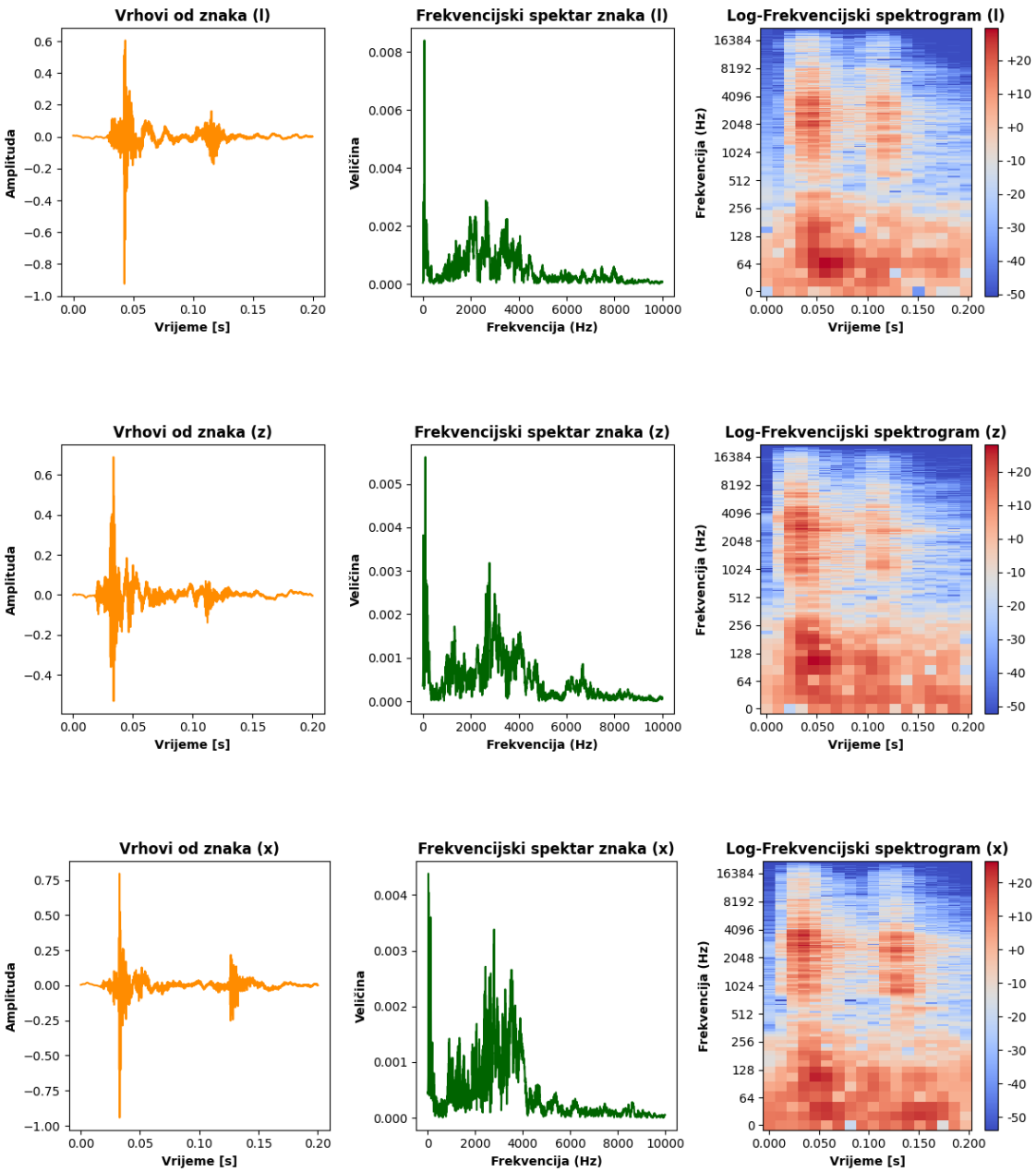
Slika 3.7 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: P, A, S



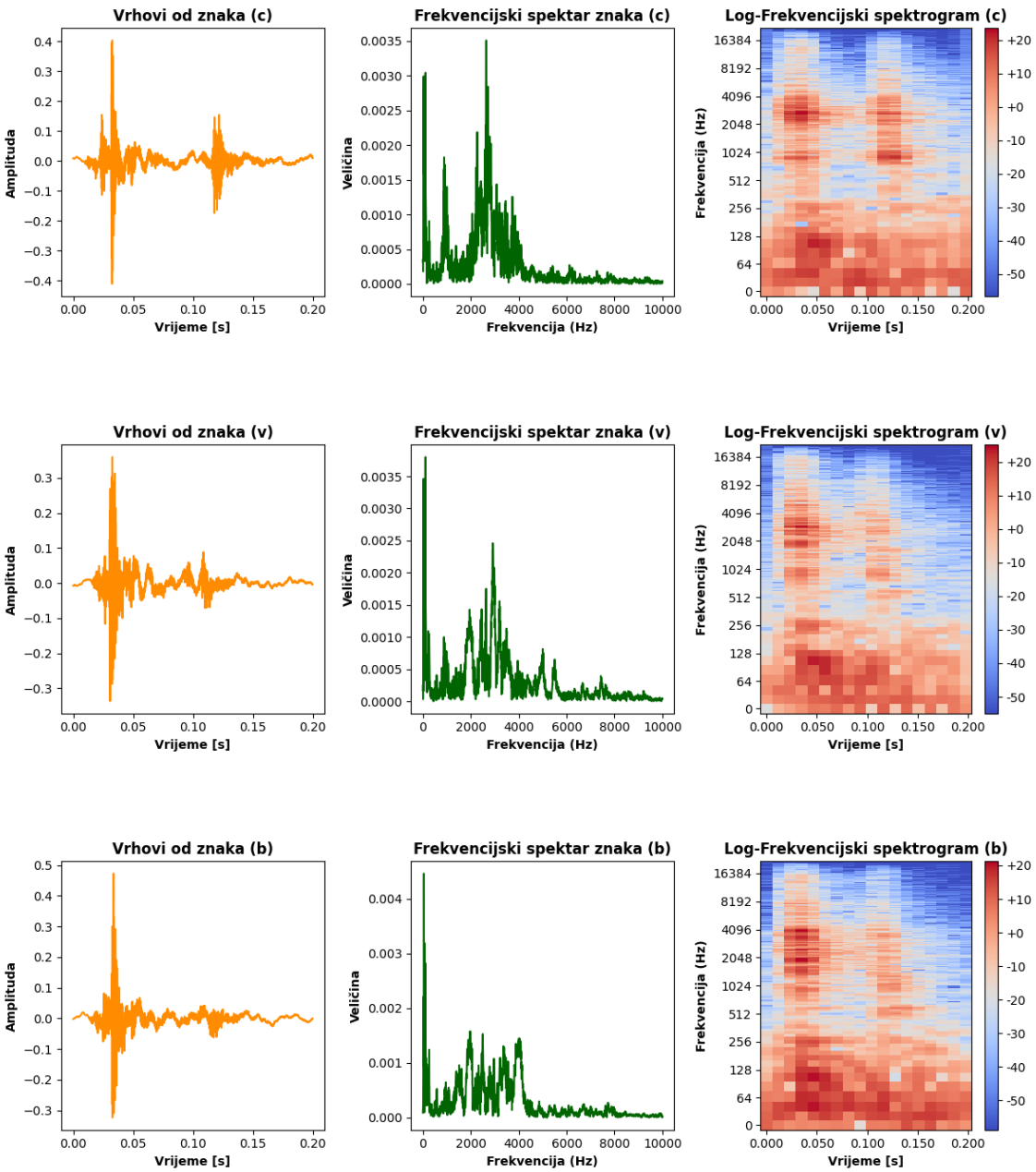
Slika 3.8 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: D, F, G



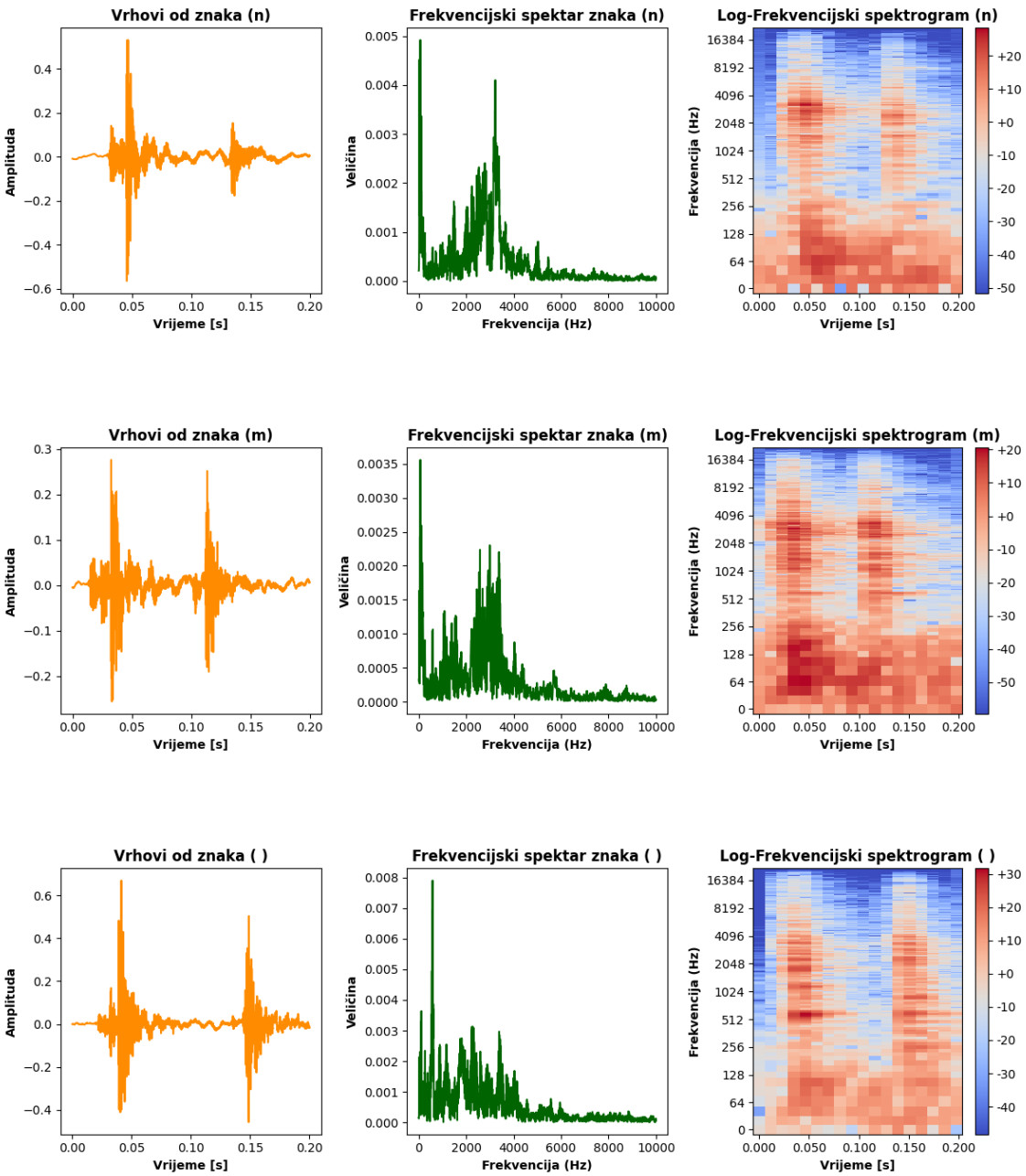
Slika 3.9 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: H, J, K



Slika 3.10 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: L, Z, X



Slika 3.11 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: C, V, B



Slika 3.12 Frekvencijski spektri, vrhovi i log-frekvencijski spektrogrami za: N, M, razmak

Kao što je iz grafova vidljivo, svaka tipka se razlikuje u svojim obilježjima vrhova, frekvencijskom spektru kao i log-frekvencijskom spektrogramu. Iako se vrhovi čine relativno slični između slova, a pogotovo susjednih slova, veliki posao u raspoznavanju takvih slova odradit će frekvencijski spektar kao i log-frekvencijski spektrogram koji je jedinstven za svako slovo. Iz log-frekvencijskog spektrograma, koji je dobiven korištenjem **kratkotrajne Fourierove transformacije** *(eng. STFT - Short-time Fourier transform)* nad podacima, jasno se mogu raspoznati frekvencije pritisnog i otpusnog vrha. Kratkotrajna Fourierova transformacija je tehnika vremensko-frekvencijske analize koja se koristi u obradi signala za razgradnju signala vremenske domene u njegove sastavne frekvencijske komponente tijekom malih, preklapajućih vremenskih intervala [23]. Trajanje zvukova tipki, kao i njihovi vrhovi i amplitude, koji spadaju u vremenske značajke, kao i frekvencijski spektar koji spada u frekvencijsku značajku će pridonijeti treniranju neuronske mreže da izvuče i najsitnije razlike u zvukovima svake tipke kako bi se dobila čim bolja predikcija natipkane rečenice.

3.4.2 Ekstrakcija značajki iz individualnih zvukova tipki

U prijašnjem koraku generirana je datoteka u kojoj su spremljene sve vrijednosti za svaki zvuk tipke, kao i njihove oznake. Primjer kako izgledaju te vrijednosti u umanjenoj veličini prikazan je u nastavku:


```
[[0.00014506028674077243, 0.0002906312874983996, 0.0003634167951531708,
0.0003634167951531708, 0.0003634167951531708, 0.00014506028674077243,
0.00014506028674077243, 0.00021784579439554363, 0.00021784579439554363,
0.00021784579439554363, 7.227477908600122e-05, -5.10719758040068e-07,
..., -0.2218507081270218, 0.007205253932625055, 0.2446315586566925,
0.46429821848869324, 0.5918911695480347, 0.6208598017692566,
0.5629225373268127, 0.4168420732021332, 0.20918501913547516, ...,
0.0049489033408463, 0.00560397282242775, 0.00545840198174119], "q"]
```


Kao što je iz podataka vidljivo, sve vrijednosti su u rasponu od $[-\mathbf{1}, \mathbf{1}]$, što je već

prije bilo i rečeno. Jedna tipka sadrži 8820 numeričkih vrijednosti koje su spremljene unutar polja, a u tim vrijednostima spada i vrijednost lokalnog maksimuma, koja za ovaj primjer iznosi **0.6208598017692566**, te oznaku koja je bila otipkana i koja se veže uz sve te vrijednosti, u ovom slučaju to je slovo **q**. Ove informacije su korisne neuronskoj mreži da izvuče značajke za svaku tipku.

Prvi korak u ekstrakciji značajki iz zvukova tipki je čitanje datoteka koje sadrže podatke kao primjer prikazan iznad. Oznaka iz podataka se prebacuje iz znaka u brojevnu reprezentaciju tog znaka. Svaki znak ima brojevnu vrijednost od 0 do 26 koja je ekvivalentna njegovoj poziciji u engleskoj abecedi (slovo a je predstavljeno s brojem 0, slovo b s brojem 1, itd.). S brojem 26 označen je znak za razmak, te bilo koji drugi znak koji u ovom kontekstu možemo smatrati zabranjenim. Ovu radnju je potrebno odraditi jer neuronska mreža radi s brojevnim podacima te kao izlaz će dati brojevni podatak tj. kojoj klasi (slovu) smatra da pripada dani ulazni podatak.

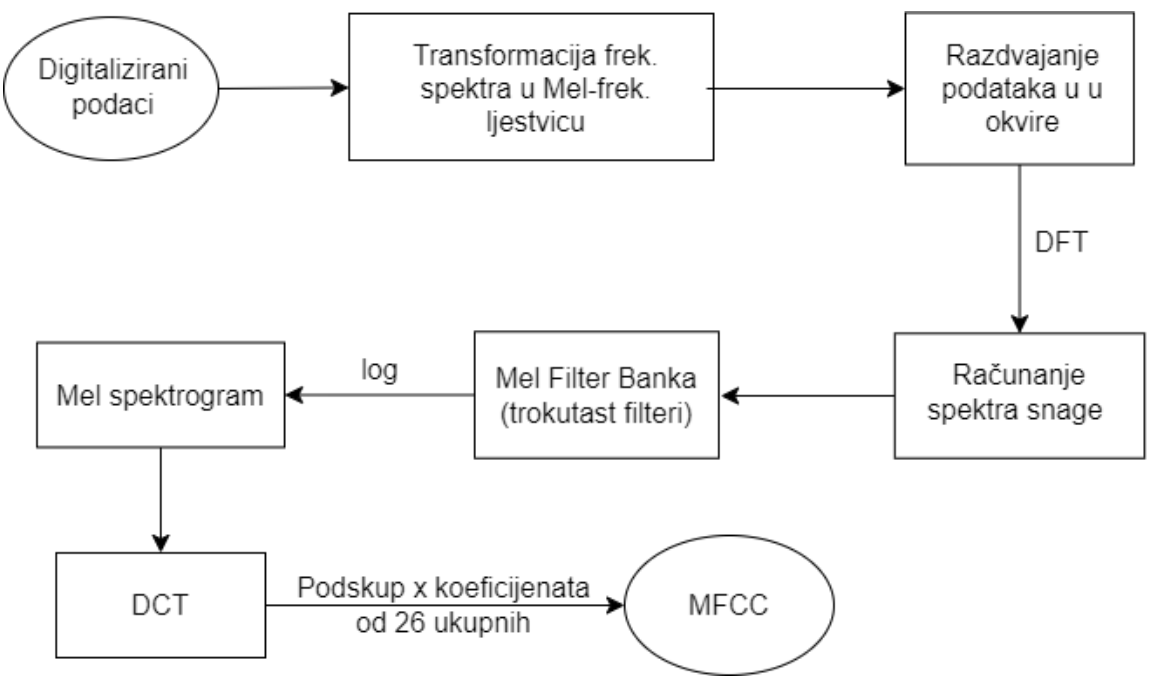
Iz brojevnih podataka datoteke odrađuje se ekstrakcija značajki pomoću Mel-frekvencijskih kepstrum koeficijenata. **Mel-frekvencijski kepstrum koeficijenti**, skraćeno **MFCC**, predstavljaju temeljnu metodu ekstrakcije značajki u domeni obrade audio i govornog signala. Inspirirani nelinearnim perceptivnim karakteristikama ljudskog sluha, MFCC služe kao sredstvo za ekstrakciju vitalnih spektralnih i timbralnih atributa iz audio signala. Postupak dobivanja MFCC-ova bit će opisan ukratko u nastavku. Računanje MFCC-ova počinje transformacijom linearnog frekvencijskog spektra u **Mel-frekvenciju ljestvicu**, logaritamsku frekvencijsku prezentaciju osmišljenu za oponašanje osjetljivosti ljudskog slušnog sustava na određene frekvencijske raspone. Formula po kojoj se obavlja konverzija iz frekvencija u Mel-frekvencijsku ljestvicu prikazana je formulom 3.1.

$$\mathbf{m(f) = 1125*ln(1+f/700)} \tag{3.1}$$

Mel-frekvencijska ljestvica služi kako bi se pojačale niže frekvencije na kojima ljudsko uho može primijeniti čak i male promjene u visini tona *(eng. pitch)*.

Nakon konverzije u Mel-frekvencijsku ljestvicu, signal je potrebno razdvojiti u kratke, preklapajuće okvire *(eng. frames)* kako bi se uhvatile vremensko-lokalizirane značajke. Kada se signal podijeli u okvire, potrebno je izračunati spektar snage *(eng. power spectrum)* tj. spektralnu gustoću *(eng. spectral density)* koristeći **diskretnu**

Fourierovu transformaciju, skraćeno **DFT**. Spektar snage opisuje distribuciju snage ili energije signala kao funkciju frekvencije. Na izračunati spektar snage primjenjuje se **Mel Filter Banka** *(eng. Mel Filter Bank)*. To su trokutasti filteri koji se primjenjuju kako bi se dobila bolja rezolucija na niskim frekvencijama te grupirale energije u različitim frekvencijskim područjima, slično kao što ljudski sluh naglašava određene frekvencijske raspone. Nakon primjene trokutastih filtera, potrebno je uzeti logaritam energija koje su dobivene njihovom primjenom. Ovim korakom sažima se dinamički raspon energije filter banke kako bi odgovarao ljudskoj percepciji koja ne percipira glasnoću na linearnoj skali. Zatim je potrebno izračunati **diskretnu kosinusnu transformaciju**, skraćeno **DCT**, na logaritamskim energijama kako bi transformirala energije iz vremenske domene u frekvencijsku domenu. Na kraju Mel-frekvencijski kepstrum koeficijenti zapravo čine podskup DCT koeficijenata, najčešće 12 koeficijenata od ukupnih 26. Kao što je već prije rečeno, ovi koeficijenti izvlače bitne spektralne i timbralne karakteristike iz podataka [24]. Opisani postupak prikazan je vizualno dijagramom, koji se nalazi na slici 3.13.



Slika 3.13 Dijagram postupka računanja MFCC-ova

Osim Mel-frekvencijskih kepstrum koeficijenata, podatke je moguće prikazati i na Mel spektrogramu koji prikazuje energiju signala kako varira tijekom vremena na različitim frekvencijama koje su razmaknute prema Mel ljestvici. Mel spektrogram je vizualna reprezentacija svih koraka računanja MFCC-ova prije izračuna diskretne kosinusne transformacije na logaritamskim energijama.

Mel spektrogram kao i Mel-frekvencijski kepstrum koeficijente moguće je implementirati u Pythonu koristeći **librosa** Python paket. Unutar paketa postoje razni dodatni paketi, ali za izračun Mel spektrograma i MFCC-ova potreban je **feature** paket koji sadrži funkcije **melspectrogram** i **mfcc**. Kako bi se Mel spektrogram i MFCC-ovi mogli izračunati, potrebno je definirati nekoliko varijabli:

1. **sr** - označava brzinu uzimanja uzoraka, te kako je već prije spomenuto, stavljena je na 44100 Hz

2. **n_mfcc** - broj MFCC-ova koji će biti izračunati, odabrano je 16

3. **n_fft** - broj uzoraka po okviru u STFT-u, odabrana vrijednost je 220, što odgovara prozoru od 10 ms za brzinu uzorkovanja od 44100 Hz

4. **hop_length** - broj audio uzoraka između susjednih STFT okvira, odabrana vrijednost je 110, što odgovara približno 2.5 ms "skokovima" za brzinu uzorkovanja od 44100 Hz

5. **n_mels** - broj Mel traka *(eng. Mel bands)* tj. broj frekvencija u rasponu od minimalne do maksimalne, odabrana vrijednost je 40

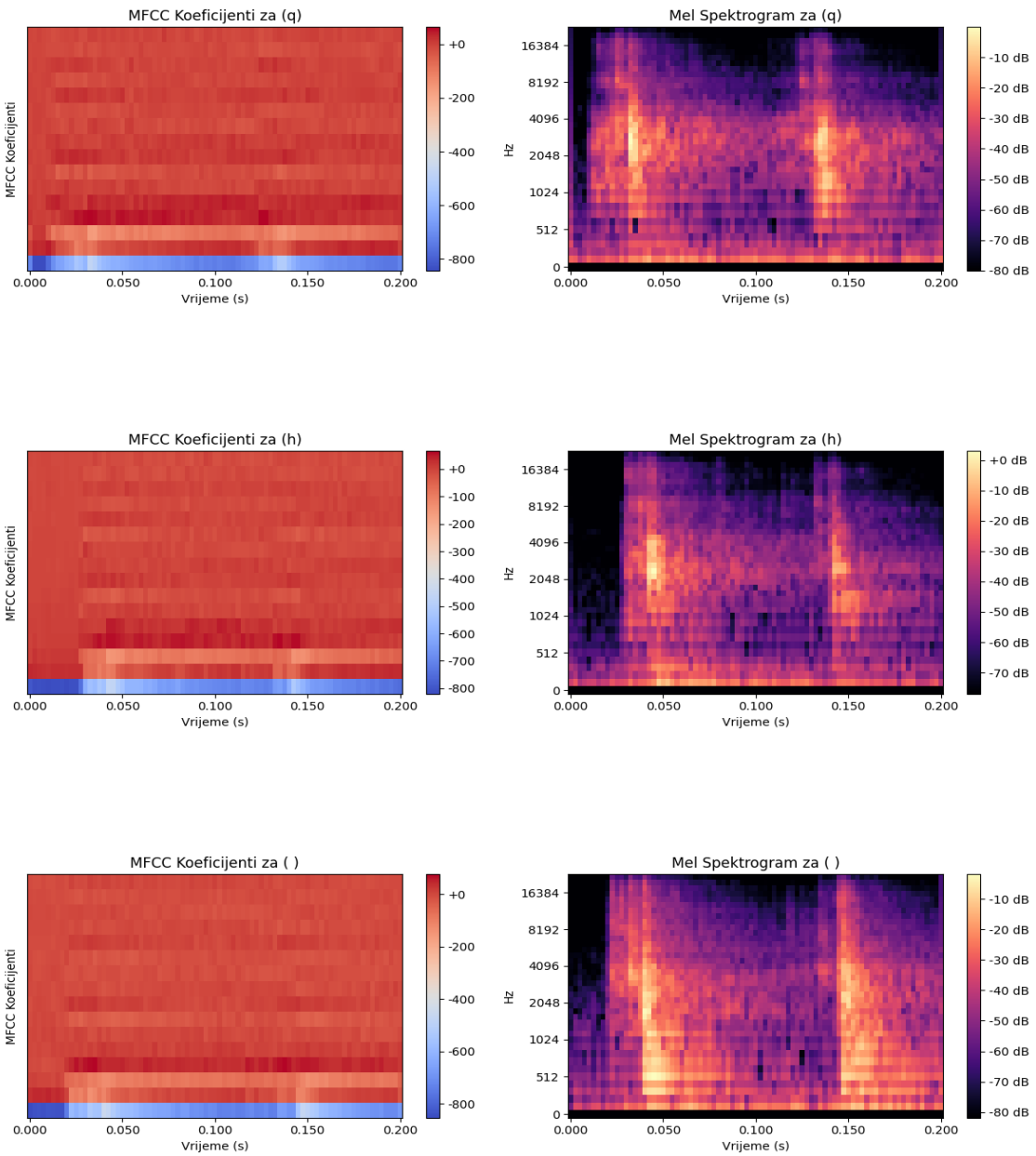
Vrijednosti za **n_ftt** i **hop_legth** su odabrane prema radu L. Zhuanga, F. Zhoua i J.D. Tygara iz 2009. godine, u kojem su s odabranim vrijednostima ostvarili dobre rezultate [25].

Unutar funkcije ***extract_features***, prikazan je Python kod za izračun Mel spektrograma i MFCC-ova.

```
    def extract_features(self, keystroke, label, sr=44100,
                                n_mfcc=16, n_fft=220, hop_len=
                                110):
        spec = mfcc(y=keystroke.astype(float),
                    sr=sr,
                    n_mfcc=n_mfcc,
                    n_fft=n_fft,
                    hop_length=hop_len,
                    )
        mel_spectrogram = melspectrogram(
            y=keystroke.astype(float),
            sr=sr,
            n_fft=n_fft,
            hop_length=hop_len,
            n_mels=40
            )
        # Convert to dB scale
        mel_spectrogram_db = power_to_db(mel_spectrogram)

        # return flattened one-dimensional tensor containing
                                the MFCC feature vector
        return torch.tensor(spec.flatten())
```

Svaka značajka predstavljena je 2D poljem koje se sastoji od **16 redaka**, koji predstavljaju 16 MFCC-ova, i **81 stupca**, unutar kojih se nalaze vrijednosti svakog koeficijenta za svaki od 81 okvira. Tako prikazanu značajku potrebno je pretvoriti u 1D vektor, veličine umnoška redaka i stupaca 2D polja, koji sadrži sve vrijednosti. Na kraju potrebno je 1D vektor pretvoriti u tenzor, jednostavni matematički objekt koji se koristi za opisivanje fizičkih svojstava, kako bi se podaci mogli proslijediti neuronskoj mreži.

Kao mali primjer kako podaci iz Mel spektrograma i MFCC-ova izgledaju, njihova vizualizacija za slova **Q** i **H** i znak **razmak** prikazana su na slici 3.14.



Slika 3.14 Mel-frekvencijski kepstrum koeficijenti i Mel spektrogram za: Q, H, razmak

3.4.3 Treniranje neuronske mreže s ciljem raspoznavanja razlike između zvukova različitih tipki

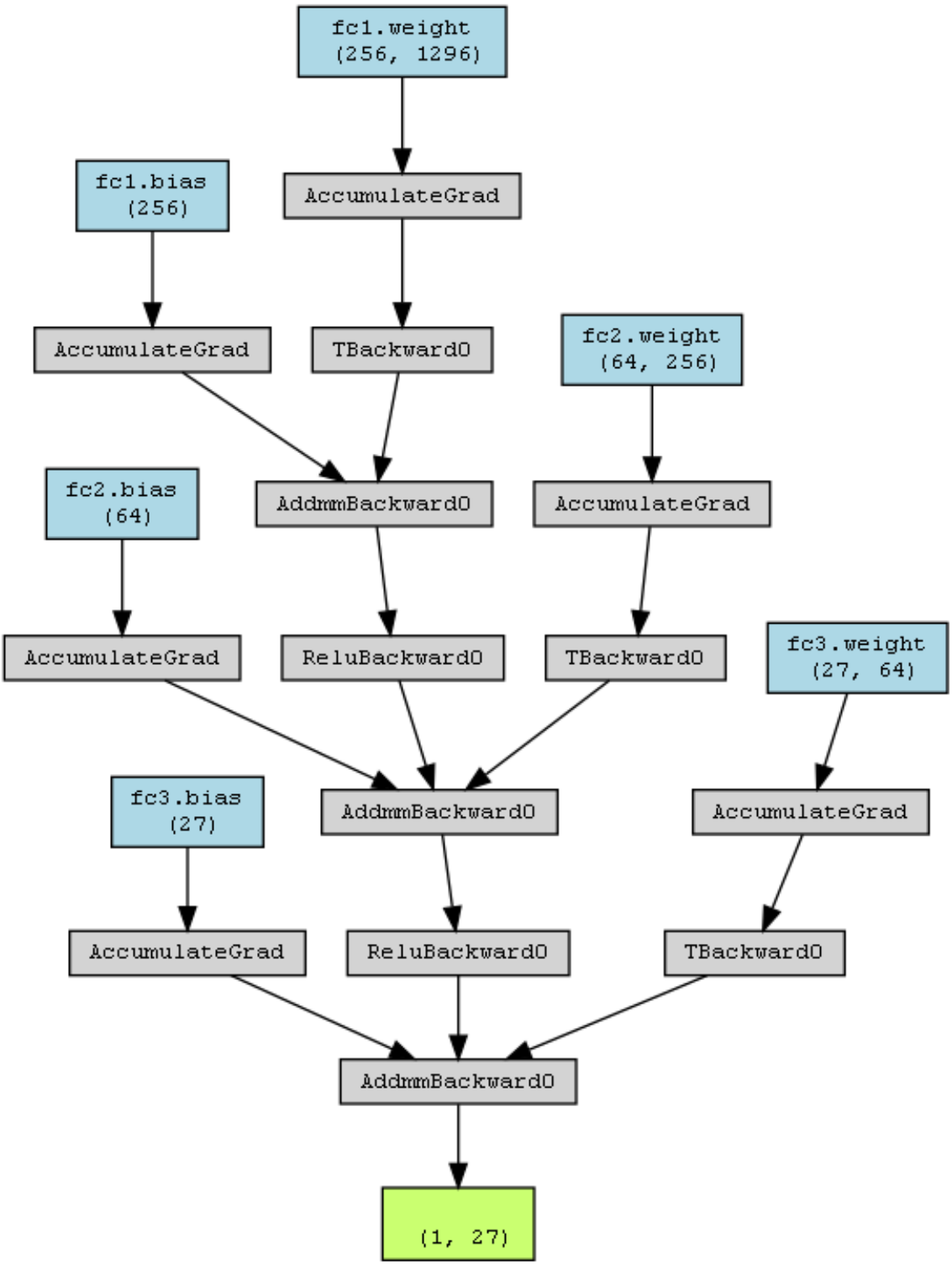
Podaci koji se šalju neuronskoj mreži prikazani su kao par (2-torka) *(eng. tuple)* tenzora. Prvi tenzor para sadrži vektor značajki zvuka tipke s ukupno **1296** vrijednosti, koji je dobiven izračunom Mel-frekvencijskih kepstrum koeficijenata. Drugi tenzor para sadrži brojevnu vrijednost otipkanog znaka za čiju je snimku računan MFCC. Primjer izgleda podatka prikazan je u nastavku:

```
(tensor([-858.1145, -858.1989, -856.4804,  ...,    6.2316,  -11.1975,
         -2.8058], dtype=torch.float64), tensor(26))
```

Prije samog treniranja neuronske mreže, potrebno je podijeliti dobivene podatke na podatke za treniranje i podatke za validaciju. Podaci su podijeljeni prema standardnoj podjeli 80/20, 80% podataka je uzeto za treniranje mreže dok na ostalih 20% će se mreža validirati. To bi značilo da od ukupno **4033** znakova tipkanih na Akko tipkovnici, njih **3226** bi bilo uzeto za trening podatke, a **807** za validacijske podatke.

Nakon raspodjele podataka, potrebno je kreirati neuronsku mrežu. Dijagram neuronske mreže prikazan je na slici 3.15. Kao što je na slici 3.15 vidljivo, neuronska mreža se sastoji od 3 linearna sloja. Na izlazima iz prva 2 sloja nalazi se odabrana aktivacijska funkcija ReLU. ReLU je uobičajeni tip aktivacijske funkcije koja se koristi u neuronskim mrežama za postizanje nelinearnih transformacija podataka. Nelinearne transformacije se primjenjuju na podatke u nadi da će podaci biti, u ovom slučaju, linearno odvojivi kako bi se mogli klasificirati kao određena tipka. ReLU prihvaća izlaz iz sloja te na ulaz u idući sloj prosljeđuje izlaznu vrijednost prijašnjeg sloja ako je ona pozitivna ili nulu ako je izlazna vrijednost negativna [26]. Za optimizaciju parametara neuronske mreže, kao što su težine, korišten je standardni optimizacijski algoritam Adam.

Kao ulaz u neuronsku mrežu šalje se prvi tenzor para vrijednosti. Ovaj tenzor sadrži, kao što je već prije spomenuto, vektor značajki znaka s 1296 vrijednosti, što se uzima kao broj ulaznih parametara neuronske mreže, a vidljivo je na slici 3.15.



Slika 3.15 Dijagram neuronske mreže

Učenje neuronske mreže na podacima je standardno, za zadani broj epoha, u ovom slučaju je 250, potrebno je proći kroz cijeli trening set podataka, što će se odraditi više puta. Neuronska mreža će na temelju značajki, koje su poslane kao ulazni podaci, dati predikciju o kojem slovu se radi na snimci, ali kao izlaz će poslati redni broj slova u abecedi što je prije već bilo naglašeno. Nakon predikcije, potrebno je izračunati koliki gubitak *(eng. loss)* postoji u točnosti predikcije podataka. Ovaj izračun radi se između predikcije izlaza neuronske mreže i stvarne vrijednosti, koja je sadržana unutar drugog tenzora para, pomoću funkcije unakrsne entropije gubitka *(eng. Cross-Entropy Loss)*. Ova funkcija mjeri performanse modela uz pomoć razlike između previđene i stvarne vrijednosti, što je razlika veća to je i gubitak veći [27]. Nakon izračuna gubitka, odrađuje se propagacija unatrag *(eng. backpropagation)* kako bi se izračunali gradijenti neuronske mreže s kojima će se na kraju optimizacijski algoritam Adam ažurirati težine u neuronskoj mreži.

Nakon svake epohe određuje se validacija trenutnog modela. Iako ovo nije standard, jer inače se odrađuje cijeli trening modela, a zatim validacija, u ovom slučaju je napravljeno kako bi se mogli uspoređivati razni modeli i na kraju spremiti najbolji. Validacija modela se kao i trening dio, sastoji od toga da neuronska mreža izbaci predikciju znaka za koji misli da je dobila podatke. Nakon toga traži se klasa s najvećom predikcijskom vjerojatnosti za svaku predikciju. Na kraju se ažuriraju varijable za ukupni broj instanci kao i brojač točno predviđenih klasa koji se povećava samo ako se predviđena klasa, u ovom slučaju znak, podudara s istinitom klasom tj. znakom. Pomoću ove dvije vrijednosti, izračunava se točnost modela koja se kasnije uspoređuje s točnostima drugih modela kako bi se najbolji mogao spremiti i koristiti u daljnjim provjerama. Python kod opisanog postupka validacije prikazan je u nastavku unutar funkcije ***validate***.

```
def validate(self):
    correct = 0
    total = 0
    # disable gradient computation
    with torch.no_grad():
        # iterate over mini-batches of data
        for data in self.validloader:

            # extract the input data and true labels
            images, labels = data

            outputs = self.net(images)
            _, predicted = torch.max(outputs.data, 1)
            total += 1
            correct += (predicted == labels).sum().item()

    acc = 100 * correct / total
    print(f'Accuracy of the network on {len(self.validloader)}
                                test keys: {acc}\%')
    return acc
```

Kada se završi zadnja validacija modela i dobio se model s najboljom točnosti, postupak validacije se ponavlja zatim na cijelim podacima. Ovdje se model više ne trenira niti ažurira, nego se samo gleda koliko točno može predviditi slova ukoliko mu se da veća baza podataka.

Krajnja validacija točnosti modela odradila se na svakoj datoteci posebno s podacima samo iz te datoteke. Ovo je odrađeno iz razloga da možemo dobiti predviđenu rečenicu za određenu datoteku, ali i vidjeti statistiku točnosti određivanja pojedinih slova kao i koje su česte zamjene slova bile. Praćenje točnosti predviđenih slova odrađeno je pomoću kreiranog rječnika u koji bi se upisivala originalna oznaka tj. znak i predviđeni znak od strane neuronske mreže te njihova frekvencija pojavljivanja. Dio koda koji predstavlja ovaj postupak nalazi se unutar funkcije ***validate_sentence*** koja neće biti prikazana u cijelosti jer znatni dio koda se podudara s funkcijom ***va-***

lidate, stoga će biti prikazan samo dio za statistiku točnosti određivanja pojedinih slova u nastavku.

```
def validate_sentence(self, path):
    char_predict = defaultdict(lambda: defaultdict(int))
    correctly_predicted_chars = {}

    with torch.no_grad():
        for data in self.validloader:

            images, labels = data
            outputs = self.net(images)
            _, predicted = torch.max(outputs.data, 1)

            # convertnumber function returns a character for
                                the given number
            char_predict[convertnumber(labels.data)][
                                convertnumber(predicted)] += 1

    # iterate through char_predict dictionary
    for char in char_predict:
        sorted_char = sorted(char_predict[char].items(), key=
                                operator.itemgetter(1))
        if(len(sorted_char) > 1):
            print(f"Most common mistake for char {char}: {
                                sorted_char}")
        elif(len(sorted_char) == 1):
            correctly_predicted_chars[char] = sorted_char
    for char, sorted_char in correctly_predicted_chars.items():
        print(f"Character has been 100\% correctly predicted {
                                char}: {sorted_char}")
```

Rezultati točnosti modela, kao i prikazi predviđenih rečenica i statistike točnosti predviđanja svakog znaka u datoteci, bit će opisani u poglavlju 4.

Poglavlje 4

Rezultati

4.1 Rezultati neuronske mreže za Akko World Tour-Tokyo R1 3087 tipkovnicu

Potrebno je još jednom napomenuti da su ovi rezultati dobiveni za tipkanje jednog tipkača sa snimkama snimljenim u gotovo savršenim uvjetima i metodom tipkanja traženja i kuckanja.

Iako su korištene iste vrijednosti parametara za računanje Mel-frekvencijskih kepstrum koeficijenata koje su J. D. Tygara, Li Zhuanga i Feng Zhoua u radovima iz 2005. godine [12] kao i iz 2009. godine koristili [25], u ovom radu korišteno je samo **nadzirano strojno učenje** *(eng. supervised machine learning)*, način učenja neuronske mreže koji koristi označene skupove podataka za treniranje algoritma koji klasificiraju podatke ili točno predviđaju ishode. Razlog odabiru ovog pristupa učenja neuronske mreže je jer implementacijom metode učenja bez nadzora, koja je korištena u navedenim radovima, dobiveni rezultati nisu mogli biti reproducirani već su uvijek dolazili nasumice i s poprilično lošom točnosti modela. Iz tog razloga, ovaj pristup neće ni biti opisan u ovom radu kao ni njegovi rezultati.

Nakon treninga i validacije rezultata neuronske mreže, točnost najboljeg modela dosegla je **97,02233%** što su značajno bolji rezultati nego u radovima J. D. Tygara, Li Zhuanga i Feng Zhoua iz 2005. godine [12] kao i iz 2009. godine [25].

Značajno bolji rezultat postignut je povećanjem baze podataka. U gore navedenim radovima korišteni su skupovi podataka u duljinama od 12 do 26 minuta, no kroz eksperiment se dokazalo da je to premala količina podataka da bi neuronska mreža mogla dobro odrediti koja tipka je bila pritisnuta po podacima koje je dobila. Opće poznato je da za treniranje neuronske mreže i dobivanje dobrih rezultata je potrebno imati veći skup podataka, a kako ovaj napad u praksi bi se trebao fokusirati na samo jednog tipkača, logičan odabir je bio povećati bazu podataka.

Iako točnost modela iznosi navedenih 97,02233%, točnost predviđenog teksta od strane neuronske mreže iznosi između 98% i 100%.

U nastavku je prikazan izlaz iz neuronske mreže, kao i statistika točnosti predviđanja svakog znaka za jednu snimku iz **qwerty** grupe i jednu snimku iz **rečenice** grupe.

1. Snimka iz **qwerty** grupe:

```
6. ..\out\keystrokes\keyboards\qwerty13_p.wav_out
Accuracy of the network for one file
on the 270 test keys: 99.62962962962963%

Predicted sentence for ..\out\keystrokes\keyboards\qwerty13_p.wav_out:
mmmmmmmmmmnnnnnnnnnnbbbbbbbbbbvvvvvvvvvvccccccccccxxxxxxxxxxzzzzzzzzz
’ ’’ ’’ ’’ ’’ ’’ ’’ ’’ ’’ ’’ ’llllllllllkkkkkkkkkkjjjjjjjjjjhhhhhhhhhh
ggggggggggffffffffffddddddddddssssssssssaaaaaaaaaappppppppppoooooooooo
iiiiiiiiiiuuuuuuuuuuyyyyyyyyyyttttttttttrrrrrrrrrreeeeeeeewewwwwwwwwww
qqqqqqqqqq

Most common mistake for char e: [(’w’, 1), (’e’, 9)]
Character has been 100% correctly predicted m: [(’m’, 10)]
Character has been 100% correctly predicted n: [(’n’, 10)]
Character has been 100% correctly predicted b: [(’b’, 10)]
Character has been 100% correctly predicted v: [(’v’, 10)]
Character has been 100% correctly predicted c: [(’c’, 10)]
Character has been 100% correctly predicted x: [(’x’, 10)]
```

```
Character has been 100% correctly predicted z: [('z', 10)]
Character has been 100% correctly predicted  : [(' ', 10)]
Character has been 100% correctly predicted l: [('l', 10)]
Character has been 100% correctly predicted k: [('k', 10)]
Character has been 100% correctly predicted j: [('j', 10)]
Character has been 100% correctly predicted h: [('h', 10)]
Character has been 100% correctly predicted g: [('g', 10)]
Character has been 100% correctly predicted f: [('f', 10)]
Character has been 100% correctly predicted d: [('d', 10)]
Character has been 100% correctly predicted s: [('s', 10)]
Character has been 100% correctly predicted a: [('a', 10)]
Character has been 100% correctly predicted p: [('p', 10)]
Character has been 100% correctly predicted o: [('o', 10)]
Character has been 100% correctly predicted i: [('i', 10)]
Character has been 100% correctly predicted u: [('u', 10)]
Character has been 100% correctly predicted y: [('y', 10)]
Character has been 100% correctly predicted t: [('t', 10)]
Character has been 100% correctly predicted r: [('r', 10)]
Character has been 100% correctly predicted w: [('w', 10)]
Character has been 100% correctly predicted q: [('q', 10)]
```

2. Snimka iz **rečenice** grupe:

```
20.  ..\out\keystrokes\keyboards\sentence5_p.wav_out
Accuracy of the network for one
file on the 139 test keys: 99.28057553956835%

Predicted sentence for
..\out\keystrokes\keyboards\sentence5_p.wav_out:
amidst the bustling city a charming musician olayed jazz saxophone
captivating the audience with lively soulful melodies
and rhythmic tunes
```

```
Most common mistake for char p: [('o', 1), ('p', 2)]
Character has been 100% correctly predicted a: [('a', 11)]
Character has been 100% correctly predicted m: [('m', 5)]
Character has been 100% correctly predicted i: [('i', 13)]
Character has been 100% correctly predicted d: [('d', 5)]
Character has been 100% correctly predicted s: [('s', 7)]
Character has been 100% correctly predicted t: [('t', 10)]
Character has been 100% correctly predicted  : [(' ', 19)]
Character has been 100% correctly predicted h: [('h', 7)]
Character has been 100% correctly predicted e: [('e', 10)]
Character has been 100% correctly predicted b: [('b', 1)]
Character has been 100% correctly predicted u: [('u', 6)]
Character has been 100% correctly predicted l: [('l', 7)]
Character has been 100% correctly predicted n: [('n', 8)]
Character has been 100% correctly predicted g: [('g', 3)]
Character has been 100% correctly predicted c: [('c', 6)]
Character has been 100% correctly predicted y: [('y', 4)]
Character has been 100% correctly predicted r: [('r', 2)]
Character has been 100% correctly predicted j: [('j', 1)]
Character has been 100% correctly predicted z: [('z', 2)]
Character has been 100% correctly predicted x: [('x', 1)]
Character has been 100% correctly predicted o: [('o', 4)]
Character has been 100% correctly predicted v: [('v', 2)]
Character has been 100% correctly predicted w: [('w', 1)]
Character has been 100% correctly predicted f: [('f', 1)]
```

Nakon provedenog treniranja i validacije, statistika predikcije znakova, broj ukupnog pojavljivanja znaka kao i česte zamjene prilikom krive predikcije, prikazane su u tablici 4.1.

Iz tablice 4.1 vidljivo je da je točnost predikcije slova po svakom slovu poprilično visoka te da iako postoji pogrešaka, one nisu toliko velike. Najčešće zamjene događaju se između slova koja su susjedna ili su jednako udaljena od sredine tipkovnice.

Tablica 4.1 Statistika predikcije znakova za Akko World Tour-Tokyo R1 3087 tipkovnicu

Slovo	**Točna predikcija znaka (%)**	**Ukupan broj pojavljivanja**	**Česte zamjene (broj zamjena)**
A	100	179	-
B	100	114	-
C	100	138	-
D	100	151	-
E	97,9	236	W(1), I(4)
F	99,2	123	G(1)
G	99,2	129	F(1)
H	100	139	-
I	98,4	187	E(3)
J	100	107	-
K	100	109	-
L	100	155	-
M	100	122	-
N	100	175	-
O	97,6	166	W(4)
P	96,9	128	O(1), Q(3)
Q	97,3	111	W(1), P(1), I(1)
R	99,4	170	I(1)
S	100	169	-
T	100	182	-
U	100	150	-
V	100	125	-
W	98,4	122	O(2)
X	100	110	-
Y	97,6	126	T(3)
Z	100	115	-
’ ’	100	295	-

4.1.1 Rezultati neuronske mreže za SHINOBI White tipkovnicu

Ovaj eksperiment je rađen i na drugoj tipkovnici s pretpostavkom da će se dobiti model s većom razlikom u točnosti ako mu se kao ulazne vrijednosti daju snimke

tipkovnice s drugačijim prekidačem koji ima svoje karakteristike. Kod nije nimalo mijenjan kako bi se čim bolje mogla provjeriti pretpostavka.

No pretpostavka je brzo bila dokazana krivom jer je točnost najboljeg modela dosegla **97,0203%** što je minimalna razlika u točnosti u odnosu na najbolji model Akko tipkovnice.

Kao i za Akko tipkovnicu, u tablici 4.2 prikazana je statistika predikcije znakova.

Tablica 4.2 Statistika predikcije znakova za SHINOBI White tipkovnicu

Slovo	**Točna predikcija znaka (%)**	**Ukupan broj pojavljivanja**	**Česte zamjene (broj zamjena)**
A	99,0	191	E(1), X(1)
B	98,3	117	V(1), A(1)
C	100	143	-
D	97,5	158	G(1), J(1), X(1), C(1)
E	99,6	253	W(1)
F	100	124	-
G	99,2	132	K(1)
H	100	144	-
I	98,5	196	K(1), Y(1), Z(1)
J	99,1	107	N(1)
K	100	111	-
L	100	162	-
M	99,2	128	N(1)
N	100	182	-
O	98,9	176	I(1), K(1)
P	100	131	-
Q	100	111	-
R	100	179	-
S	99,4	180	E(1)
T	99,0	194	Z(1), K(1)
U	100	152	-
V	100	129	-
W	98,4	123	E(1), Q(1)
X	98,2	111	Z(1), S(1)
Y	100	130	-
Z	99,1	116	X(1)
’ ’	100	318	-

Kao kod statistike za Akko tipkovnicu, vidljivo je iz tablice 4.2 da postoje zamjene slova s drugim slovom, ili slovima, prilikom predikcije, no postotak točne predikcije je i dalje poprilično visok. Vidljivo je da su najčešće zamjene između slova koja su susjedna na tipkovnici ili na jednakoj udaljenosti od sredine tipkovnice, kao što je to bio slučaj s rezultatima Akko tipkovnice.

4.1.2 Rezultati neuronske mreže trenirane na obje tipkovnice i uspješnost predikcije ne označenih podataka

Završetkom treniranja i validacije neuronske mreže za svaku tipkovnicu posebno, odlučeno je napraviti zajednički model koji će biti treniran na snimkama obje tipkovnice. Pretpostavka uspješnosti ovog modela bila je niska s obzirom na to da tipkovnice se jako razlikuju u zvuku te iz tog razloga smatralo se da neuronska mreža neće moći točno naučiti niti predvidjeti slova jer postoji velika vjerojatnost da bude zbunjena s dva različita zvuka za isto slovo. No iznenađujuće, najbolji model je imao točnost od **95,62576%** što je malo manje od 2% pogoršanja u odnosu na najbolje modele svake posebne tipkovnice.

Dobiveni najbolji model dodatno je testiran s ne označenim podacima koji nisu bili korišteni ni za testiranje niti za validaciju modela, a dobiveni su na isti način kao i označeni podaci u koraku **Ekstrakcija zvukova tipki iz .wav snimki**, samo bez dodatnog slanja .txt datoteke. U nastavku će biti prikazane 3 rečenice koje su se koristile za dodatno testiranje. Za svaku od 3 rečenica, bit će prikazan originalni tekst, zatim tekst tipkan na Akko tipkovnici, a zatim tekst tipkan na SHINOBI tipkovnici.

1. **Originalna rečenica:**
   the joyful chef prepared exquisite gourmet meals sizzling with flavor using a unique blend of spices and fresh quality ingredients

   (a) **Akko tipkovnica:**
   the joyful chef prepared exquisite gourmet meals sizzling with flavor using a unique blend of spices and fresh quality ingredients

(b) **SHINOBI tipkovnica:**
the joyful chef prepared exquisite gourmeh meals sizzling with flavor using a unique blend of spices and fresh quality ingredients

2. **Originalna rečenica:**
with cautious eyes the detective uncovered a cryptic conspiracy solving mysteries exposing villains and preserving justice with zealous fervor

(a) **Akko tipkovnica:**
witd cautious eyes the detective uncovered a cryptic conspiracy solving mysteries exposing villains and preservinh justice with zealous fervor

(b) **SHINOBI tipkovnica:**
with cautious eyes the detective uncovered a cryptic conspiracy solving mysteries exposing villains and preserving justice with zealous fervor

3. **Originalna rečenica:**
beneath the twinkling stars a poet wrote verses of love invoking passion longing and tender emotions with eloquent heartfelt words

(a) **Akko tipkovnica:**
beneath the twinkling stars a poet wrote verses of love invoking passion longing and tender emotions with eloquent heartfelt words

(b) **SHINOBI tipkovnica:**
beneath the twinkling stars a ooet wrote verses of love invoking passion longing and tender emotions with eloquent heartfelt words

Unutar rečenica crvenom bojom označena su slova koja su krivo predviđena i razlikuju se od slova u originalnoj rečenici.

Kao što se iz rečenica može vidjeti, postotak pogrešno predviđenog znaka za prikazane rečenice iznosi **0,00332%**, što je jako malen postotak. Iako postoji krivo slovo unutar riječi, ona je i dalje čitljiva i lako ispravljiva od strane čovjeka, a ako se i dogodi neka veća pogreška gdje jedna riječ može biti predviđena kao više riječi, iz konteksta rečenice moguće je odrediti točnu riječ.

Poglavlje 5

Zaključak

Kroz rad prikazana je akustična kriptoanaliza kroz teoriju, ali i kod. Akustična kriptoanaliza predstavlja veliku sigurnosnu prijetnju, no u obliku u kojem je implementirana i prikazana u ovom radu, ta sigurnost je ugrožena samo ako se ciljani tipkač nalazi u savršenim uvjetima.

Rezultati, iako su dobiveni u relativno savršenim uvjetima, pokazali su da je moguće rekreirati i predvidjeti s visokom točnošću tekst čija je snimka uzeta kao ulaz u neuronsku mrežu. Također pokazano je da koristeći dvije poprilično različite tipkovnice, njihovi zasebni modeli, ali kao i kombinirani model, dati će odlične rezultate i na označenim i na neoznačenim podacima te će predvidjele rečenice biti izrazito čitljive.

Pozitivne strane ove implementacije su što se visoka točnost uspjela dobiti proširivanjem baze podataka snimki, bez da se morao uvoditi dodatno jezični model. Ovakav pristup ima smisla jer za odradu akustične kriptoanalize kroz povijest, napadači su kroz neki vremenski period skupljali podatke od ciljane žrtve kako bi mogli čim bolje istrenirati svoje modele, koji bi u ključnom trenutku mogli s visokim postotkom točnosti tada dati napisani tekst.

Mane ove implementacije, ali i općenito akustične kriptoanalize, su velike buke i smetnje u audio snimkama. Kao poboljšanje ovog rada predlažem daljnje testiranje u raznim okruženjima s različitim razinama pozadinske buke. U tim situacijama, pretprocesiranje audio signala bilo bi puno kompliciranije jer bi se audio snimka

trebala procesirati tako da se prvo izdvoji što je više moguće okolinske buke i smetnje iz nje, a zatim ili pojačati zvuk tipkanja na tipkovnici ako je to potrebno ili koristiti pretprocesiranje audio snimke, bez buke i smetnji, kako je to odrađeno u ovom radu.

Još jedan prijedlog za poboljšanje rada bilo bi korištenje snimki na kojima tipkač koristi različitu metodu tipkanja od traženja i kuckanja koje je korišteno u ovom radu. Uz to bilo bi dobro proširiti napad na cijelu tipkovnicu kako bi se čim više teksta i lozinki moglo predvidjeti.

Kako je eksperiment odrađen na mehaničkim tipkovnicama, bilo bi zanimljivo vidjeti kakve rezultate bi neuronska mreža dala za snimke tipkanja na membranskim tipkovnicama pošto je njihov zvuk prilikom tipkanja više prigušen. Također buduće implementacije eksperimenta mogle bi biti trenirane na hrvatskom jeziku i slovima kako bi se izračunala točnost predviđanja i na drugim jezicima.

Bibliografija

[1] A. S. i. E. T. Daniel Genkin, "Acoustic cryptanalysis," *Journal of Cryptology*, 2016.

[2] R. A. Dmitri Asonov, "Keyboard acoustic emanations," *IBM Almaden Research Center*, 2004.

[3] TEMPEST. , s Interneta, https://en.wikipedia.org/wiki/Tempest_(codename) , veljača 2023.

[4] side-channel attack. , s Interneta, https://www.techtarget.com/searchsecurity/definition/side-channel-attack , veljača 2023.

[5] Power analysis. , s Interneta, https://en.wikipedia.org/wiki/Power_analysis , veljača 2023.

[6] Timing attack. , s Interneta, https://en.wikipedia.org/wiki/Timing_attack , vljača 2023.

[7] Electromagnetic attack. , s Interneta, https://en.wikipedia.org/wiki/Electromagnetic_attack , veljača 2023.

[8] Acoustic cryptanalysis. , s Interneta, https://en.wikipedia.org/wiki/Acoustic_cryptanalysis , veljača 2023.

[9] Optical Side Channel Attacks on Singlechip. , s Interneta, https://www.atlantis-press.com/proceedings/itms-15/25842897 , veljača 2023.

[10] The Temperature Side Channel and Heating Fault Attacks. , s Interneta, https://eprint.iacr.org/2014/190.pdf , veljača 2023.

[11] What is a side-channel attack and how do they work? , s Interneta, https://www.comparitech.com/blog/information-security/side-channel-attack/ , veljača 2023.

[12] J. D. T. Li Zhuang, Feng Zhou, "Keyboard acoustic emanations revisited," *University of California, Berkeley*, 2005.

[13] Acoustic Cryptanalysis — A Side-Channel Attack. , s Interneta, https://kindawingingit.medium.com/acoustic-cryptanalysis-a-side-channel-attack-f864851be2c4 , veljača 2023.

[14] Fast Fourier transform. , s Interneta, https://en.wikipedia.org/wiki/Fast_Fourier_transform , veljača 2023.

[15] NSA. , s Interneta, https://en.wikipedia.org/wiki/NSA_encryption_systems , veljača 2023.

[16] TEMPEST 2. , s Interneta, https://stationhypo.com/2021/07/06/tempest-and-side-channel-attacks-1-of-5/ , veljača 2023.

[17] Engulf Operation. , s Interneta, http://www.faqs.org/espionage/Ec-Ep/Engulf-Operation.html , veljača 2023.

[18] Typing: hunt and peck. , s Interneta, https://en.wikipedia.org/wiki/Typing , kolovoz 2023.

[19] Alphabet and Character Frequency: English. , s Interneta, https://www.sttmedia.com/characterfrequency-english , kolovoz 2023.

[20] Python module: wavio. , s Interneta, https://pypi.org/project/wavio/ , kolovoz 2023.

[21] Mono vs Stereo. , s Interneta, https://www.headphonesty.com/2022/01/what-is-the-difference-between-mono-and-stereo/ , kolovoz 2023.

[22] Digital Audio Basics: Audio Sample Rate and Bit Depth. , s Interneta, https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html , kolovoz 2023.

[23] Toggle the table of contents Short-time Fourier transform. , s Interneta, https://en.wikipedia.org/wiki/Short-time_Fourier_transform , kolovoz 2023.

[24] Mel Frequency Cepstral Coefficient (MFCC) tutorial, note = , kolovoz 2023., url = http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/, owner = James Lyons, timestamp = 2023.08.29.

[25] J. D. T. Li Zhuang, Feng Zhou, "Keyboard acoustic emanations revisited," *University of California, Berkeley*, 2009.

[26] ReLU in neural networks. , s Interneta, https://stats.stackexchange.com/questions/226923/why-do-we-use-relu-in-neural-networks-and-how-do-we-use-it , kolovoz 2023.

[27] Cross-Entropy Loss Function. , s Interneta, https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/ , kolovoz 2023.

Sažetak

U ovom radu predstavljena je teorija i povijest akustične kriptoanalize kao i praktičan primjer primjene akustične kriptoanalize kroz kod pisan u Pythonu. Korištenjem Mel-frekvencijskih kepstrum koeficijenata za izvlačenje značajki iz snimki tipkanja te linearne neuronske mreže, model je uspio s oko 97% točnosti predvidjeti tipkani znak iz označenih podataka dvije različite tipkovnice. S visokom točnosti najbolji model je uspio rekonstruirati natipkani tekst iz neoznačenih podataka tipkanih na obje tipkovnice.

Ključne riječi — **Akustična kriptoanaliza, tipkovnice, linearna neuronska mreža, Mel-frekvencijski kepstrum koeficijenti**

Abstract

In this paper, the theory and history of acoustic cryptanalysis is presented, as well as a practical example of the application of acoustic cryptanalysis through code written in Python. Using Mel-frequency cepstral coefficients to extract features from typing recordings and a linear neural network, the model was able to predict the typed character from the labeled data of two different keyboards with about 97% accuracy. With high accuracy, the best model was able to reconstruct typed text from unlabeled data typed on both keyboards.

Keywords — **Acoustic cryptanalysis, keyboards, linear neural network, Mel-frequency cepstral coefficients**