Aplikacija za anonimnu razmjenu datoteka

Cvitković, Simon

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:311835

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJCSKIH TEHNOLOGIJA

Sveučilišni studij

APLIKACIJA ZA ANONIMNU RAZMJENU DATOTEKA

Završni rad

Simon Cvitković

Osijek, 2017.

SADRŽAJ:

1. UVOD 1

1.1. Zadatak završnog rada 2

2. TEORIJSKE OSNOVE KORIŠTENIH TEHNOLOGIJA 3

2.1. Raspberry Pi 3

2.2. Tor 3

2.3. SSH 5

2.3.1. SFTP 6

2.4. Kriptografski alati i tehnologije 7

2.4.1. GNU Privacy Guard 7

2.4.2. AES 7

2.5. Python 8

2.6. Qt Designer 8

2.7. PyQt 9

2.8. PyCharm 9

3. MODEL APLIKACIJE 10

3.1. Koncept aplikacije 10

3.2. Model poslužiteljske aplikacije 10

3.3. Model klijentske aplikacije 11

4. OPIS KREIRANE APLIKACIJE 13

4.1. Poslužitelj 13

4.1.1. Skripta *mail-test.py* 13

4.1.2. Skripta *dir-watcher.py* 16

4.1.3. Skripta *new-piranoid-launcher.sh* 19

4.2. Klijent 20

4.2.1. Skripta *main.py* 21

5. ZAKLJUČAK 28

LITERATURA 30

SAŽETAK 31

ABSTRACT 32

ŽIVOTOPIS 33

1. UVOD

Tema završnog rada je aplikacija za anonimnu razmjenu datoteka. Potrebno je osmisliti mehanizam za anonimnu razmjenu datoteka u Linux okruženju. Na osnovu modela je bilo potrebno izraditi aplikaciju implementacijom klijentske i poslužiteljske strane čijom primjenom bi se ostvarila anonimnost na internetu koristeći Tor program , podaci zaštitili nekom od kriptografskih metoda. Potrebno je i omogućiti instalaciju poslužiteljske strane na SoC uređaje. Inspiracija za izradu ovog rada je program OnionShare, tj. njegova kritička analiza, koji omogućava anonimnu razmjenu datoteka. Uočeno je da OnionShare ne nudi kriptiranje datoteka i ovisi o potencijalno nesigurnoj razmjeni datoteka koji izlaže korisnike opasnosti od krađe podataka te deanonimizacije. Iz tog razloga se problem anonimne razmjene datoteka promatra iz drugog kuta te se traže drugačija rješenja za isti problem.

Postoje brojni scenariji u kojima osobe žele anonimno razmjenjivati datoteke. To može biti skupina razvojnih programera koji rade na inovativnom proizvodu koji ne smije pasti u ruke konkurenciji. Drugi primjer korištenja je grupa znanstvenika koja radi na tajnom projektu čija aktivnost nikako ne smije biti povezana s njima samima, dok s druge strane to mogu biti vojska i državni agenti koji rukuju s izuzetno osjetljivim informacijama koje moraju ostati tajne i dijeliti se tajnim komunikacijskim kanalima. U takvim, a i u mnogim ostalim slučajevima važno je osigurati anonimnu i sigurnu komunikaciju, te samim time očuvati i sigurnost ljudskih života.

Na početku završnog rada govorit će se o teorijskim osnovama korištenih tehnologija: Raspberry Pi računala kao primjera SoC uređaja i poslužitelja u ovom radu, Tor programa za anonimizaciju internet prometa i upoznavanje s funkcionalnostima korištenima u radu, više informacija o SSH protokolu te njegovoj nadogradnji u obliku SFTP protokola koji omogućava prijenos datoteka. Zatim će biti govora o osnovnim informacijama kriptografskih tehnologija korištenima za kriptiranje podataka u komunikaciji klijenta i poslužitelja. Na kraju opisa korištenih tehnologija bit će opisani: programski jezik Python, Qt Designer program za dizajn i izradu grafičkog korisničkog sučelja, PyQt programsko proširenje za implementaciju sučelja u Pythonu i PyCharm radni okvir za kvalitetnije programiranje u Pythonu. Glavni dio rada podijeljen je na dva dijela - model aplikacije koji objašnjava logiku funkcionalnosti osmišljenog mehanizma i opis aplikacije u kojem se opisuje kod aplikacije te prikazuje njen izgled kroz korištenje.

1.1. Zadatak završnog rada

Potrebno je osmisliti i modelirati mehanizam za anonimnu razmjenu datoteka u Linux okruženju te na osnovu modela izraditi aplikaciju implementacijom klijentske i poslužiteljske strane aplikacije koristeći skriptne programske jezike poput Pythona i koristeći Tor program. Za poslužiteljski dio implementacije omogućiti instalaciju na SoC uređaje. Podatke koji se izmjenjuju zaštiti nekim od oblika kriptiranja.

2. TEORIJSKE OSNOVE KORIŠTENIH TEHNOLOGIJA

U nastavku su ukratko objašnjene tehnologije korištene za realizaciju završnog rada. Tehnologije korištene u radu vezane su za sklopovlje, mreže i mrežne protokole, kriptografske algoritme te programske jezike i alate odabrane za izvedbu aplikacije za anonimno dijeljenje datoteka.

2.1. Raspberry Pi

Raspberry Pi je računalo veličine kreditne kartice koje ulazi u kategoriju SoC uređaja (engl. *System on a Chip*). Raspberry Pi Fundacija iz Ujedinjenog Kraljevstva razvila je istoimeno računalo u svrhu popularizacije računarstva u školama i zemljama u razvoju. Zbog povoljne cijene, postalo je najpopularnije malo računalo u svijetu i najprodavanije britansko računalo u povijesti. Do pisanja ovog rada izašle su četiri serije računala, a po svojim performansama Raspberry Pi 3 Model B je najmoćnije računalo od ostala tri, prema [1]

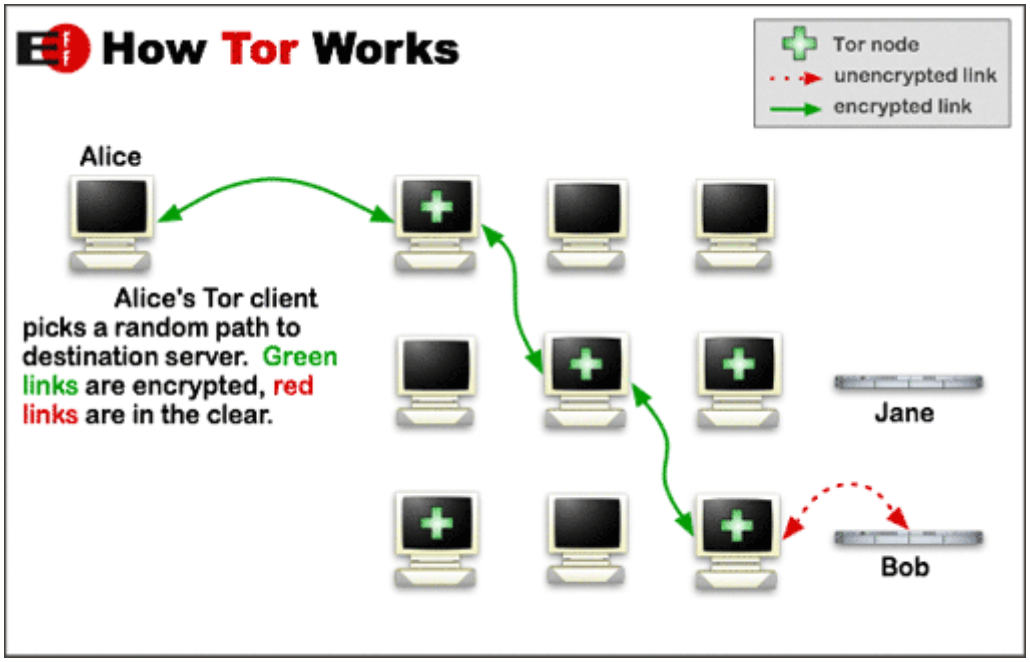
U ovom radu, Raspberry Pi koristi se kao poslužitelj za aplikaciju izrađenu u završnom radu. Operacijski sustav koji pokreće Raspberry Pi mora biti dizajniran za ARM arhitekturu (engl. *Advanced RISC Machine Architecture*). Između velike ponude raznih Linux distribucija za ARM arhitekturu, odabran je Raspbian Jessie – Linux distribucija koju je razvila Raspberry Pi Fundacija, a derivacija je vrlo poznate Debian Linux distribucije. Raspberry Pi Fundacija podržava Jessie što omogućava redovita ažuriranja i stabilnost operacijskog sustava.

2.2. Tor

Tor je program za anonimizaciju aktivnosti na internetu. Razvili su ga matematičar Paul Syverson te tadašnji studenti računarstva na MIT-u Roger Dingledine i Robert Mathewson. Ime Tor potječe iz kratice TOR (engl. *The Onion Router*), a *onion routing* je bila tehnika osiguravanja obavještajnih komunikacija na kojem je radila američka ratna mornarica. Tor project, organizacija osnovana 2002. godine, i danas omogućava podršku i razvoj Tor programa te ostalih aplikacija baziranih na funkcionalnosti Tora, prema [1]

Tor radi na principu usmjeravanja paketa preko tri čvora posrednika (engl. *relay)*. Čvorove može podesiti i pokretati svaka osoba s računalom i pristupom internetu. Danas Tor broji oko 7000 čvorova, što osigurava anonimnost i dobru otpornost na napade. Tor program na klijentskoj strani uspostavlja sigurnu, kriptiranu vezu do prvog čvora – ulaznog čvora. Zatim se ulazni čvor veže na srednji čvor i na kraju na izlazni čvor. Kada je put kreiran, paketi se mogu slati preko Tor mreže.

Svaki paket je omotan u više kriptiranih slojeva, a svaki sloj nosi informaciju idućeg čvora. Na svakom čvoru se dekriptira jedan sloj, čvor čita adresu idućeg čvora te prema njemu usmjerava paket. Na izlaznom čvoru se dekriptira posljednji sloj i tada se paket šalje na odredišnu adresu poslužitelja. U tom trenutku promet više nije nužno kriptiran, tj. ovisi o tipu zaštite prijenosa podataka na internetu, prema [2]. Osnovna funkcionalnost Tor-a prikazana je na slici 2.1. na kojoj se vidi kako klijent imena Alice uspostavlja vezu do poslužitelja pod nazivom Bob. Kriptirana veza označena je zelenim strelicama dok je nekriptirana veza označena crvenim isprekidanim strelicama. Čvorovi u Tor mreži označeni su zelenim križićima.



Slika 2.1. Prikaz rada Tora, iz [1]

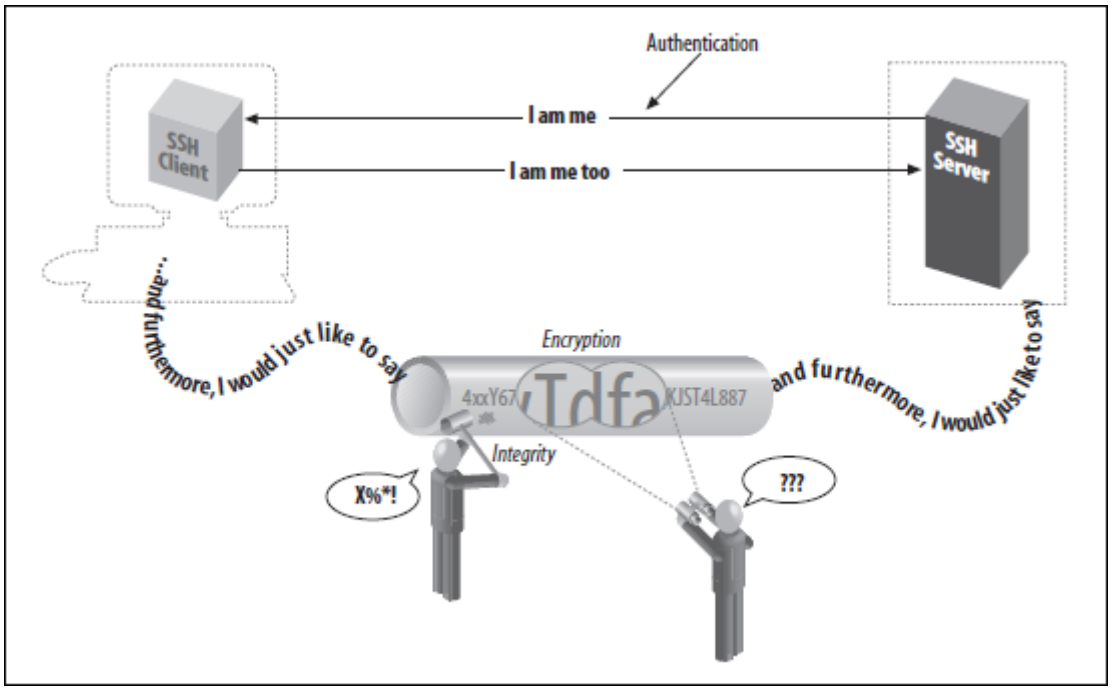
Tor, osim što omogućava anonimnost koristeći tri čvora prije dostizanja do poslužitelja na internetu, omogućava kreiranje i korištenje skrivenih usluga (engl. *hidden service*). Skrivene usluge su zapravo aplikacije poslužitelja koje su dostupne samo unutar Tor mreže. To znači da računalo koje ne koristi Tor, ne može saznati da je skrivena usluga dostupna. Skrivene usluge omogućavaju i klijentu i poslužitelju komunikaciju bez da jedan o drugome znaju informacije koje mogu odati identitet, npr. IP adresu. To je moguće zbog točaka sastanaka (engl. *rendezvous points*). Poslužitelj najprije mora oglasiti svoje postojanje čvorovima. Ti čvorovi se zovu točke predstavljanja (engl. *introduction points*). Zatim poslužitelj objedinjava informacije o točkama sastanaka i svoj javni ključ te informacije – opisne informacije, potpisuje privatnim ključem i šalje u distribuiranu *hash* tablicu. Klijenti opisne informacije mogu naći korištenjem XYZ.onion adrese gdje XYZ može biti adresa koja se sastoji od 16 alfanumeričkih znakova izvedenih iz javnog ključa

poslužitelja. Tada je skrivena usluga podešena i spremna za funkciju. Klijent pri pronalasku skrivene usluge, tj. njenih opisnih informacija, uspostavlja točku sastanka preko koje kontaktira poslužitelja jednokratnom tajnom kriptiranom javnim ključem poslužitelja i čuva svoju anonimnost. Sva komunikacija se odvija u Tor mreži, tako da klijent i poslužitelj ne znaju međusobne IP adrese, a mogu izmjenjivati podatke, prema [3].

Skrivene usluge, iako skrivene svima izvan Tor mreže, korisnicima Tor-a su dostupne ukoliko korisnik zna *.onion* adresu. Da bi određene skrivene usluge zaista i bile skrivene od svih potencijalnih uljeza, postoje nevidljive skrivene usluge (engl. *stealth hidden service*). Razlika od klasičnih skrivenih usluga je u činjenici što poslužitelj skrivene usluge šalje dodatnu šifru koja se uparuje s *.onion* adresom, i za svakog klijenta se radi novi par adrese i šifre. Klijent koji nema šifru, ne može kontaktirati točke upoznavanja preko *.onion* adrese pa za njega ta skrivena usluga ne postoji. Na taj način se postiže autorizacija provjerenih korisnika. Nevidljive skrivene usluge pružaju mogućnost kreiranja do 16 različitih parova adresa i šifri, prema [4].

2.3. SSH

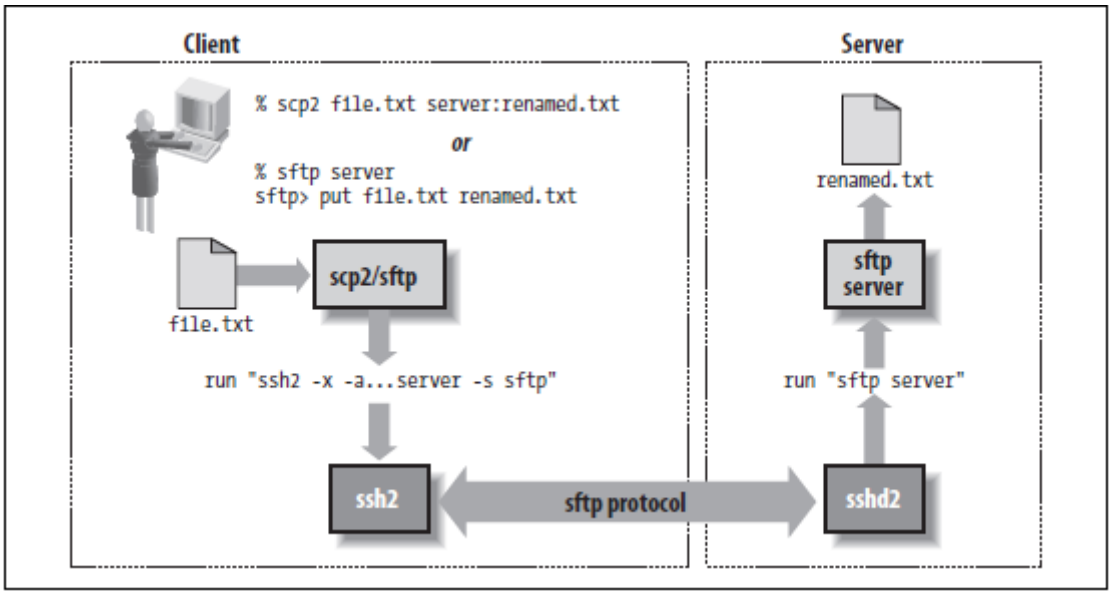
SSH *(engl. Secure Shell)* je protokol za rad na udaljenom računalu. Razvio ga je Tatu Ylönen, finski znanstvenik na Helsinškom Sveučilištu Tehnologije 1995. godine. Protokol je nastao kao rezultat sigurnosnih nedostataka postojećeg Telnet protokola. Tada se protokol zvao SSH-1, danas je standard SSH-2 ili SSH 2.0 protokol. SSH omogućava rad na udaljenom računalu, ali ne omogućava lokalnu ljusku niti ne nudi prevoditelja naredbi već osigurava siguran prijenos naredbi do udaljenog računala na kojem se potom naredbe izvršavaju. Veza do udaljenog računala je kriptirana, korisniku provjeren identitet te protokol sprječava izmjenu podataka u transportu kao što je to učestalo u MITM napadima (engl. *Man In The Middle*), iako sam protokol nije otporan na napade takvog tipa, prema [5]. Provjera identiteta, kriptiranje i integritet su tri stupa sigurnosti koji ga čine sigurnim i pouzdanim protokolom za komunikaciju s udaljenim računalom, a prikaz se može vidjeti na slici 2.2. gdje SSH klijent na lijevoj strani slike uspostavlja vezu s poslužiteljem na desnoj strani. Strelice označavaju proces provjere identiteta, a ispod je prikazan integritet podataka u obliku tunela koji osigurava da se podaci ne izmjenjuju prilikom transporta. Kriptiranje je prikazano kao niz nasumičnih znakova koji maskiraju pravu poruku.



Slika 2.2. Provjera identiteta, kriptiranje i integritet podataka u SSH vezi, iz [5]

2.3.1. SFTP

SSH protokol nema u sebi implementirane mogućnosti za prijenos datoteka na udaljeno računalo. Postojeći protokol za prijenos datoteka na udaljeno računalo – FTP (engl. *File Transfer Protocol*) nije kompatibilan sa SSH protokolom, a sam nema implementirane sigurnosne mehanizme poput kriptiranja. Zbog nepostojanja rješenja za sigurnu razmjenu datoteka, stvorila se potreba za novim protokolima. SFTP (engl. *SSH File Transfer Protocol*) nema implementirane sigurnosne mehanizme, ali svoju sigurnost oslanja na SSH protokol što znači da koristi SSH protokol kao transportnu podlogu i koristi njegove mehanizme. Prije nego se omogući slanje, preuzimanje ili izmjena datoteka na udaljenom računalu, SFTP koristi SSH protokol za slanje naredbi sigurnim načinom. Jedna od prednosti SFTP protokola je sposobnost nastavka preuzimanja datoteke nakon zastoja uzrokovanih nekim smetnjama u prijenosu, prema [5]. Funkcioniranje SFTP protokola prikazano je na slici 2.3. gdje klijent unosi SFTP naredbe za slanje datoteke *file.txt* poslužitelju, a naredbom se također određuje ime datoteke na strani poslužitelja. SFTP protokol koristi SSH za uspostavljanje veze s poslužiteljem. Nakon uspostavljanja veze s poslužiteljem, SSH javlja poslužitelju da se radi o SFTP vezi te poslužitelj pokreće SFTP verziju poslužitelja. Tada se pokreće prijenos datoteke i pri završetku prijenosa na poslužitelju se pojavljuje datoteka *renamed.txt* identičnog sadržaja kao datoteka *file.txt* čime se održava integritet podataka.



Slika 2.3. Prikaz rada SFTP protokola, iz [5]

2.4.Kriptografski alati i tehnologije

U nastavku su ukratko objašnjene tehnologije i alati korišteni za kriptiranje podataka koji se prenose putem aplikacije za anonimnu razmjenu datoteka. Tehnologije se oslanjaju na simetričnu odnosno asimetričnu kriptografiju.

2.4.1. GNU Privacy Guard

GNU Privacy Guard (skraćeno GPG) je programska implementacija Open PGP standarda za kriptiranje elektroničke pošte. Open PGP definira format i pravila kriptiranja elektroničke pošte, digitalne potpise te certifikate za razmjenu javnih ključeva. GPG je komandno interpreterski program i dostupan je za Windows, UNIX/Linux i macOS operacijske sustave. Podržava više algoritama kriptiranja - većinu algoritama za koje ne postoji patent. Licenciran je pod GPL licencom što ga čini slobodnim i besplatnim programom. Dizajniran je da radi kao *backend* aplikacija, pa je raspoloživ pozivima raznih skriptnih jezika što mu primjenu znatno proširuje u razne sfere uporabe. GPG-om se elektronička pošta brzo i efikasno kriptira asimetričnom kriptografijom (kriptografijom javnog ključa) što omogućava privatnost u komunikaciji elektroničkom poštom, prema [6].

2.4.2. AES

AES (engl. *Advanced Encryption Standard*) je jedna od najraširenijih kriptografskih tehnika današnjice. AES je podskup originalnog Rijndael algoritma. Rijndael je ime dobio po kreatorima-

belgijskim kriptografima Vincentu Rijmenu i Joanu Daemenu. AES radi na istom principu kao i Rijndael, ali je standardiziran, tj. ograničen veličinom bloka na 128 bita i duljinom ključeva na 128, 192 ili 256 bita. AES pripada skupu simetrične kriptografije, što znači da se podaci kriptiraju i dekriptiraju istim ključem za razliku od asimetrične kriptografije u kojoj postoje parovi javni-tajni ključevi. AES je popularan zbog svog algoritma kriptiranja koji do danas nije kriptografski slomljen tj. uspješni kriptografski napadi nisu ostvareni, prema [7].

2.5.Python

Python je programski jezik više razine namijenjen generalnom programiranju. Prvi put se pojavio 1991. godine, a tvorac je Guido Van Rossum. Spada u grupu interpreterskih jezika, a dizajniran je s ciljem da svojom sintaksom i skupom pravila olakša programiranje. Tako postoje dinamički tipovi podataka koji smanjuju broj linija koda jer se programer ne mora zamarati pretvaranjem podataka iz jednog tipa u drugi. Blokovi koda ne zatvaraju se ključnim riječima ili vitičastim zagradama kao što je to slučaj kod drugih jezika već „uvlačenjem“ redova što rezultira čitkijim kodom. Podržava širok skup programerskih paradigmi kao što su: objektno orijentirano programiranje, strukturno, proceduralno i MVC (engl. *Model View Controller*). MVC predstavlja programersku paradigmu razvoja aplikacija u kojoj se sve klase mogu podijeliti u: modele koji predstavljaju objektnu reprezentaciju relacija iz baze podataka, upravljače koji manipuliraju objektima modela i poglede koji prikazuju podatke manipulirane upravljačima. Zbog široke podrške biblioteka popularan je u znanosti, financijskom svijetu, automatizaciji poslova i strojnom učenju. Danas postoje dvije inačice Pythona u koegzistenciji, a to su verzija 2 i verzija 3. Verzija 3 je dizajnirana kao poboljšanje inačice 2 i nije kompatibilna s njom. Težnja cijele Python zajednice je da se verzija 2 postupno napusti, ali zbog inercije industrije i podrške biblioteka, taj proces će potrajati, prema [1].

2.6. Qt Designer

Qt Designer je alat tvrtke Qt namijenjen izradi grafičkog korisničkog sučelja (engl. *Graphical User Interface*). GUI se sastavlja od gotovih Qt elemenata koji su pretprogramirani. Forma GUI-a je ostvarena kroz XML jezik što ga čini neovisnim o platformi na kojoj se pokreće. Važna stavka je logika povezivanja elemenata, a to je npr. povezivanje gumba „Izlaz“ s funkcijom izlaska iz programa. Qt je uveo pojam signala i pretinaca (engl. *signals and slots*). Na taj način, događaj na nekom elementu emitira signal i taj signal programer može povezati s određenom funkcijom,

neovisno o tome je li ona ugrađena ili kreirana. Takav pristup olakšava programiranje ponašanja GUI-a, prema [8].

2.7.PyQt

PyQt je jedno od dva najpopularnija proširenja Qt radnog okvira (engl. *Framework*) za Python programski jezik. Drugo proširenje je PySide. PyQt omogućava korištenje Qt grafičkih elemenata u Pythonu, ali i ostalih dostupnih u Qt-u kao što su elementi za mrežnu komunikaciju, rad s bazama podataka, alati za pregledavanje Web-a itd. Kao što je ranije spomenuto, Qt GUI je izvorno ostvaren u XML jeziku, a PyQt je ostvarenje GUI-a u Pythonu. Određena forma ili prozor zapisana u XML formatu s ekstenzijom *.ui* se u Python prevodi s *pyuic* naredbom u konzoli. Kao rezultat dobije se datoteka s ekstenzijom *.py* koja je zapravo Python skripta, a njenim pozivanjem iscrtava se GUI ekvivalentan onom napisanom u XML-u, prema [9].

2.8. PyCharm

PyCharm je integrirano razvojno okruženje (engl. *Integrated Development Environment*) za Python programski jezik. Razvila ga je češka tvrtka JetBrains. Dostupan je za Windows, Linux i macOS operacijske sustave. Napisan je u Pythonu i Javi, a odličan je za razvoj Python aplikacija zbog svojih mogućnosti. Ima ugrađen *debugger*, automatsko završavanje koda (kod pozivanja varijabli, metoda ili klasa), prepoznavanje sintaktičkih pogrešaka. Podržava razvoj u radnim okvirima za razvoj Web aplikacija kao što su Django i Flask. Olakšava navigaciju kroz kod i testiranje aplikacije. Dostupan je kao besplatna verzija i profesionalna koja se plaća. Studenti imaju pravo na profesionalnu verziju tijekom svog studiranja, prema [1].

3. MODEL APLIKACIJE

U idućim potpoglavljima objašnjava se model aplikacije. Koncept aplikacije objašnjava njen smisao i osnovnu funkcionalnost. Zatim se opisuju uloge i funkcionalnosti aplikacije na strani poslužitelja, odnosno na strani klijenta.

3.1. Koncept aplikacije

Postoje razne pojave i stanja koje se javljaju u današnjoj internetskoj komunikaciji kao potencijalni problem. Jedan od njih je i kontrola nad infrastrukturom za komunikaciju. Činjenica je da je skoro nemoguće alocirati sve resurse interneta za određene partikularne potrebe, a i da je moguće, vrlo vjerojatno bi bilo ekonomski neisplativo. Može se zaključiti da valja pronaći rješenja za efikasnu uporabu internetske infrastrukture u postojećim okvirima.

Model za aplikaciju izrađenu u ovome završnom radu temelji se na komunikaciji klijent-poslužitelj. Pretpostavka je da postoji interesna skupina koja želi anonimno razmjenjivati datoteke. Jedan od članova skupine treba posjedovati poslužitelj u fizičkom obliku. Iz tog razloga se koristi Raspberry Pi. Jednostavan za instalaciju, lako prenosiv, a sadrži i Wi-Fi modul koji olakšava povezivanje na internet. Poslužitelj služi kao spremnik za datoteke koje se mogu dijeliti s ostalim korisnicima. Kako bi anonimizacija bila uspješna, i klijenti i poslužitelj moraju koristiti Tor za komunikaciju. Uz Tor koji anonimizira promet podataka, potreban je i protokol kojim će računala komunicirati te vršiti prijenos datoteka. Odabran je SSH protokol za komunikaciju i pristup datotekama, odnosno SFTP protokol za postavljanje i preuzimanje datoteka. Od važnosti je i očuvati tajnost sadržaja datoteka na poslužitelju, a to se ostvaruje kriptiranjem datoteka. Kako bi korisnici znali kada se koja datoteka pojavljuje na poslužitelju ili kada je poslužitelj uopće dostupan, koristi se komunikacija elektroničkom poštom. Aplikacija za anonimno dijeljenje datoteka će se u nastavku nazivati PiRanoid zbog jednostavnosti i kraćeg naziva. Detaljnije o modelu mehanizma za anonimnu razmjenu datoteka će biti rečeno u idućim potpoglavljima.

3.2. Model poslužiteljske aplikacije

Poslužitelj za aplikaciju zamišljen je kao malo, prenosivo računalo. Stoga je izabran Raspberry Pi kao što je spomenuto u potpoglavlju 3.1. Preduvjet je da poslužitelj ima instaliran operacijski sustav (Raspbian Linux u ovom slučaju) te vezu na internet. Poslužitelj mora imati instaliran Tor i SSH program, te stvorene Linux korisničke račune koji imaju pravo pisanja i otvaranja datoteka u direktoriju namijenjenom za razmjenu datoteka. Poslužitelj pokreće Tor i istovremeno otvara nevidljivu skrivenu uslugu. Poslužiteljev Tor klijent generira adrese za korisnike i pripadne šifre

s kojima će korisnici moći pristupiti. Ako osoba koja nije u interesnoj skupini koja koristi aplikaciju odluči pristupiti poslužitelju preko interneta, to je moguće samo ako osoba dođe do *.onion* adrese i njene pripadne šifre za korisnika. Na taj način nevidljiva skrivena usluga je vidljiva samo odabranim korisnicima, ostalima je sasvim nevidljiva i naizgled nepostojeća. Na taj se način može izbjeći razne skenere i enumeratore koji analiziraju dizajn i usluge poslužitelja diljem interneta, a služe kao priprema za napade. Poslužitelj zatim mora imati pokrenut SSH poslužitelj kako bi mogao osluškivati zahtjeve korisnika. Kada se korisnik spoji na poslužitelja, otvara se SFTP sjednica i tada je moguć prijenos datoteka. Poslužitelj ima implementiranu funkcionalnost slanja elektroničke pošte adresama korisnika. Također, instaliran program poput GPG-a, koji može kriptirati elektroničku poštu te ju poslati preko određenog SMTP (engl. *Simple Mail Trasnfer Protocol*) poslužitelja. Poslužitelj elektroničku poštu ne šalje preko Tor mreže, već kontaktira SMTP poslužitelj preko interneta. Kako ta veza potencijalno može otkriti lokaciju poslužitelja, a i njegovu IP adresu, potrebno je odabrati pružatelja usluga elektroničke pošte koji brine o privatnosti korisnika. U ovom slučaju je odabran Riseup kolektiv pa se i koristi njihov SMTP poslužitelj. Može se postaviti pitanje „Zašto baš Riseup?“. Naravno, ne mora biti Riseup, ali bi trebao biti sličan pružatelj usluga. Jedno od svojstava je što elektronička pošta poslana preko njihovog SMTP poslužitelja ne sadrži informacije u zaglavlju koje bi mogle odati identitet pošiljatelja niti Riseup čuva te informacije pa su stoga korištenje elektroničke pošte i opasnosti od deanonimiziranja svedene na minimum, prema [10]. Elektroničku poštu potrebno je slati u više navrata. Svaki puta kada poslužitelj bude aktivan potrebno je poslati elektroničku poštu korisnicima, kriptirajući svaku poruku javnim ključem korisnika. Ostali slučajevi su u svakom trenutku kada datoteka bude postavljena na poslužiteljev dijeljeni direktorij (jer se tada i datoteka s ključem postavlja u direktorij za ključeve). Tada se elektronička šalje kao obavijest da je dodana nova datoteka, datoteka s ključem se kriptira javnim ključem korisnika te se šalje kao privitak u elektroničkoj pošti. Nakon što poslužitelj pošalje elektroničku poštu svim korisnicima, briše datoteku s ključem iz direktorija s ključevima. Ključevi se sa poslužitelja brišu kako bi, u slučaju kompromitacije poslužitelja, datoteke ostale zaštićene kriptiranjem.

3.3. Model klijentske aplikacije

Model klijentske aplikacije predstavlja aplikaciju na klijentskoj strani, tj. na računalu klijenta. Aplikacija na klijentskoj strani je zamišljena da radi u Linux okruženju. Aplikacija treba imati GUI kako bi korištenje bilo jednostavno i intuitivno. Korisnik prije korištenja aplikacije treba instalirati Tor i SSH program, te konfigurirati SSH da stvori priključnicu (engl. *socket*) kojom se omogućava

usmjeravanje veze preko Tora. Konfiguracija Tora mora imati u sebi podatke (adresu i šifru) o nevidljivoj skrivenoj usluzi na koju se spaja, što omogućava Toru da uspije pronaći poslužitelja u Tor mreži. Klijent prvo mora ostvariti vezu- SSH tunel do poslužitelja, zatim slijedi otvaranje SFTP sesije, a potom mu mora biti omogućen prijenos datoteka. Kada korisnik odabere opciju postavljanja datoteke, generira se nasumični ključ za AES kriptiranje, datoteka se kriptira i ta datoteka se postavlja na poslužitelj umjesto originalne nekriptirane datoteke. Prije postavljanja kriptirane datoteke, postavlja se datoteka u koju je zapisan ključ. U slučaju preuzimanja datoteke, nakon završetka preuzimanja korisnika se traži da otvori datoteku s ključem kako bi aplikacija dekriptirala datoteku. Nakon dekriptiranja, datoteka se sprema u direktorij za preuzimanja. Kriptirane datoteke imaju isti naziv kao dekriptirane, samo im je dodana ekstenzija *.enc*. Datoteke s ključem također imaju isti naziv kao dekriptirana datoteka, samo im je dodana ekstenzija *.key*.

4. OPIS KREIRANE APLIKACIJE

U nastavku će biti opisana programska implementacija aplikacije za anonimnu razmjenu datoteka. Kod aplikacije se može podijeliti na stranu poslužitelja i na stranu klijenta. Svaka strana aplikacije ima pojedine skripte ili dijelove većeg programa koji će biti detaljnije opisani.

4.1. Poslužitelj

U idućim potpoglavljima se opisuje programska izvedba na poslužitelju, tj. skriptni programi koji definiraju odgovore poslužitelja na zahtjeve klijenta.

4.1.1. Skripta *mail-test.py*

Skripta *mail-test.py* je Python skripta koja šalje elektroničku poštu kriptiranu javnim ključevima korisnika. Prema programskom kodu 4.1. u skripti se koriste biblioteke *gnupg*, *smtlib* i *email*. Najprije se određuje instanca klase *gnupg* preko putanje do direktorija u kojem se nalazi GPG program. Na taj način se koristi GPG kao vanjski program. Zatim se određuje kodiranje znakova, koji se koriste za kriptiranje poruke, preko *encoding* atributa. Varijabla *unencrypted_string* predstavlja poruku koja se šalje u tijelu pošte - još uvijek čitljivi niz znakova bez kriptiranja. Iduća naredba otvara datoteku u istom direktoriju u kojem se nalazi i skripta, a datoteka se zove *email_list* i ona sadrži popis adresa elektroničke pošte. U datoteci su adrese zapisane svaka u svom redu, jedna ispod druge. Kada se datoteka otvori, adrese se spremaju u listu *recepients* preko metode *splitlines()* koja pročitani niz znakova, ako joj se ne proslijedi argumente, rastavlja niz znakova kada naiđe na kraj reda. Na taj način lista *recepients* sadržava adrese korisnika. Objekt *server* predstavlja klijenta koji obavlja povezivanje na SMTP poslužitelj i slanje elektroničke pošte. Pri inicijalizaciji objekta *server* u konstruktor se proslijeđuju podaci o SMTP poslužitelju koji se koristi, a to su adresa poslužitelja i port. Port 587 se koriste kod kriptirane TLS veze, a upravo se metodom *starttls()* pokreće sigurna veza prema poslužitelju. Metoda *login()* proslijeđuje podatke za provjeru identiteta klijenta, i ako je ona uspješna, metoda ne vraća pogrešku te se ispisuje poruka o uspješnoj provjeri identiteta. Kada je klijent uspješno ostvario konekciju na poslužitelj, započinje petlja koja za svaku adresu korisnika aplikacije radi iduće korake. Varijabla *encrypted_data* služi za spremanje niza znakova koji se generiraju kriptiranjem poruke prema javnom ključu korisnika povezanog s adresom elektroničke pošte. Da bi metoda *encrypt* radila na ovakav način, javni ključevi moraju biti uvezeni u GPG program. Nakon kriptiranja poruke, stvara se struktura elektroničke pošte pomoću *MIMEMultipart()* metode, ispunjavaju se podaci i na koncu se poziva metoda *sendmail()* kojoj se proslijeđuju adresa

pošiljatelja, adresa primatelja i varijabla text koja je tekstualni zapis kriptirane poruke prilagođena MIME standardu kako bi format elektroničke pošte bio ispravan. Skripta završava metodom *quit()* kojom se konekcija s SMTP poslužiteljem završava.

```
#!/usr/bin/env python3

import gnupg
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText


gpg = gnupg.GPG(gnupghome='/home/pi/.gnupg')
gpg.encoding = 'utf-8'

unencrypted_string = "Posluzitelj je dostupan!"

with open ("email_list") as file:
    recepients = file.read().splitlines()

server = smtplib.SMTP('smtp.riseup.net', 587)
server.starttls()
server.login("latordis", "password")
print("Authenticated user!")

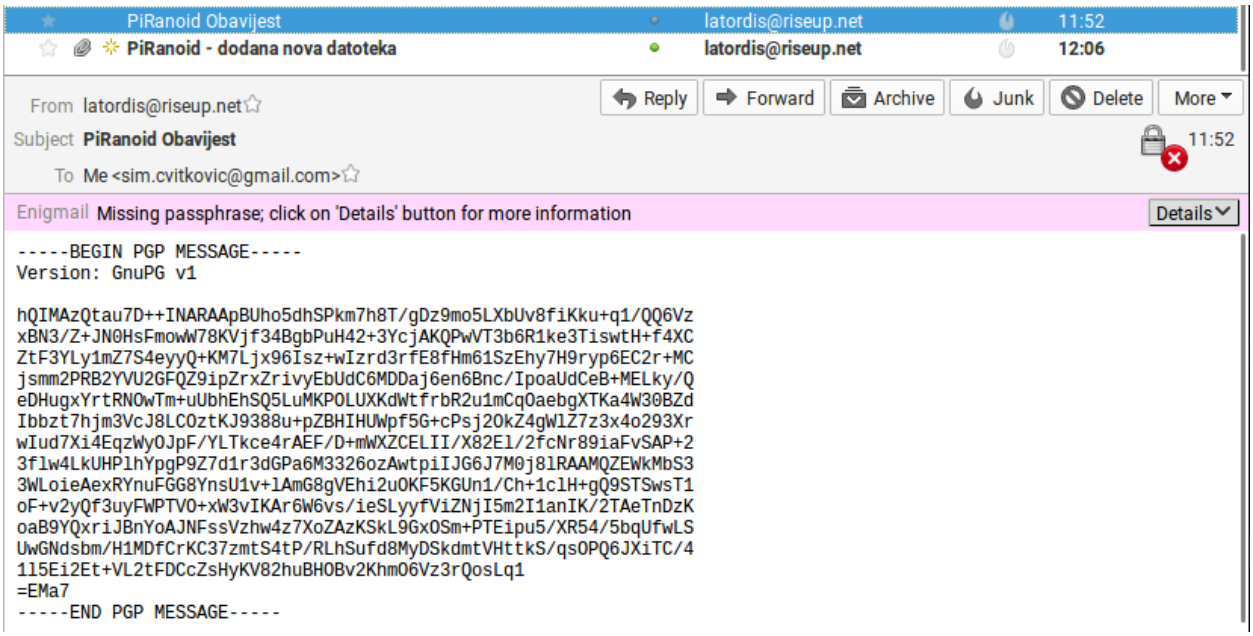
for recepient in recepients:
    encrypted_data = gpg.encrypt(unencrypted_string, recepient)
    fromaddr = "latordis@riseup.net"
    toaddr = recepient
    msg = MIMEMultipart()
    msg['From'] = fromaddr
    msg['To'] = toaddr
    msg['Subject'] = "PiRanoid Obavijest"

    body = str(encrypted_data)
    msg.attach(MIMEText(body, 'plain'))


    text = msg.as_string()
    server.sendmail(fromaddr, toaddr, text)
    print("Message sent")
server.quit()
```

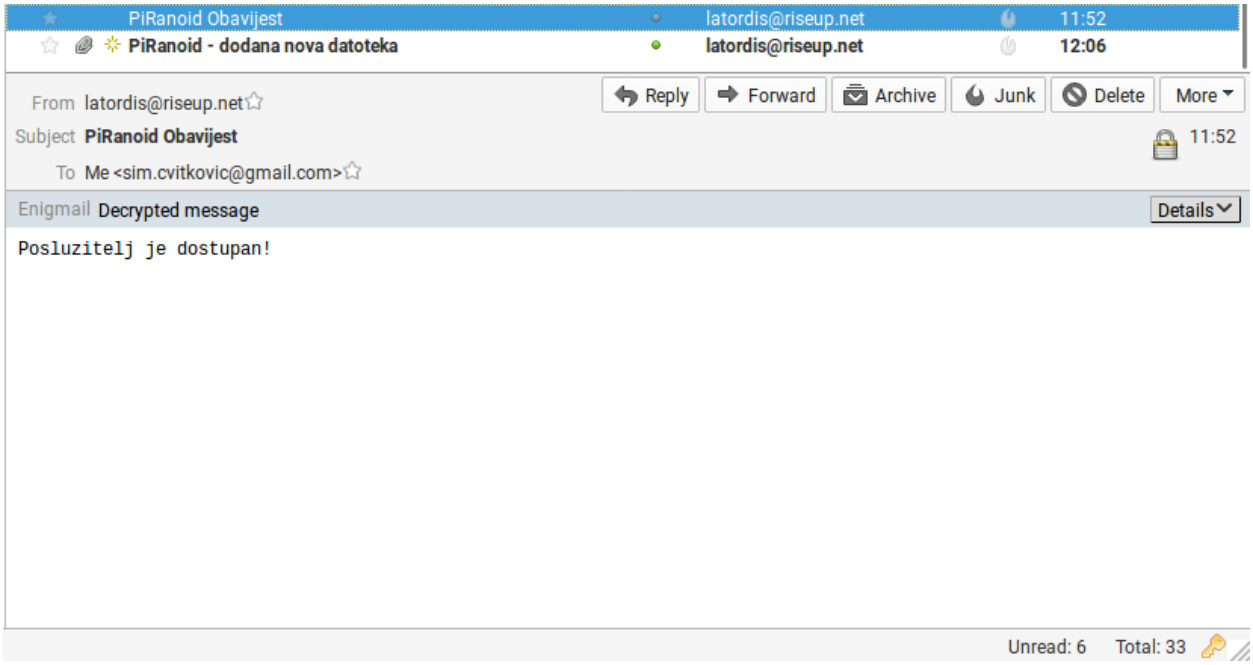
Programski kod 4.1. Skripta *mail-test.py*

Na slici 4.1. može se vidjeti uspješno primljena poruka na Gmail korisnički račun, a preuzeta s Mozilla Thunderbird klijentom za korisničku poštu. Iz slike je vidljivo da je poruka kriptirana.



Slika 4.1. Prikaz kriptiranog sadržaja elektroničke pošte

Nakon što korisnik unese privatni ključ, Mozilla Thunderbird, tj programsko proširenje Enigmail dekriptira sadržaj poruke i pokazuje izvornu poruku. Rezultat dekriptiranja može se vidjeti na slici 4.2.



Slika 4.2. Prikaz dekriptiranog sadržaja elektroničke pošte

Na ranije opisani način poslužiteljska strana aplikacije PiRanoid može razašiljati elektroničku poštu koristeći SMTP poslužitelj i kriptografiju javnog ključa Open PGP standarda koja se uspješno može dekriptirati na strani klijenta.

4.1.2. Skripta *dir-watcher.py*

Glavni dio programa, prema programskom kodu 4.2., pokazuje neke od funkcionalnosti koje PyQt nudi. Inicijalizira se objekt *app* koji označava aplikaciju. Varijabla *dirpath* označava putanju do direktorija u koji sadržava kriptirane datoteke, tj. direktorij u koji korisnici datoteke postavljaju i iz kojeg iste preuzimaju. Objekt *dir_watcher* je instanca klase *QfileSystemWatcher* koja prima varijablu *dirpath*. *dir_watcher* nakon inicijalizacije promatra promjene nad direktorijem. Svaki puta kada se pojavi nova datoteka, objekt će to opaziti i emitirati signal. U idućem koraku signal *directoryChanged* povezuje se s pretincem, tj. funkcijom *directory_changed*. Zadnja naredba pokreće *app*.

```
1.  app = QtCore.QCoreApplication(sys.argv)
2.  dirpath = "/var/piranoid/"
3.  dir_watcher = QtCore.QFileSystemWatcher(dirpath)
4.  dir_watcher.directoryChanged.connect(directory_changed)
5.  sys.exit(app.exec_())
```

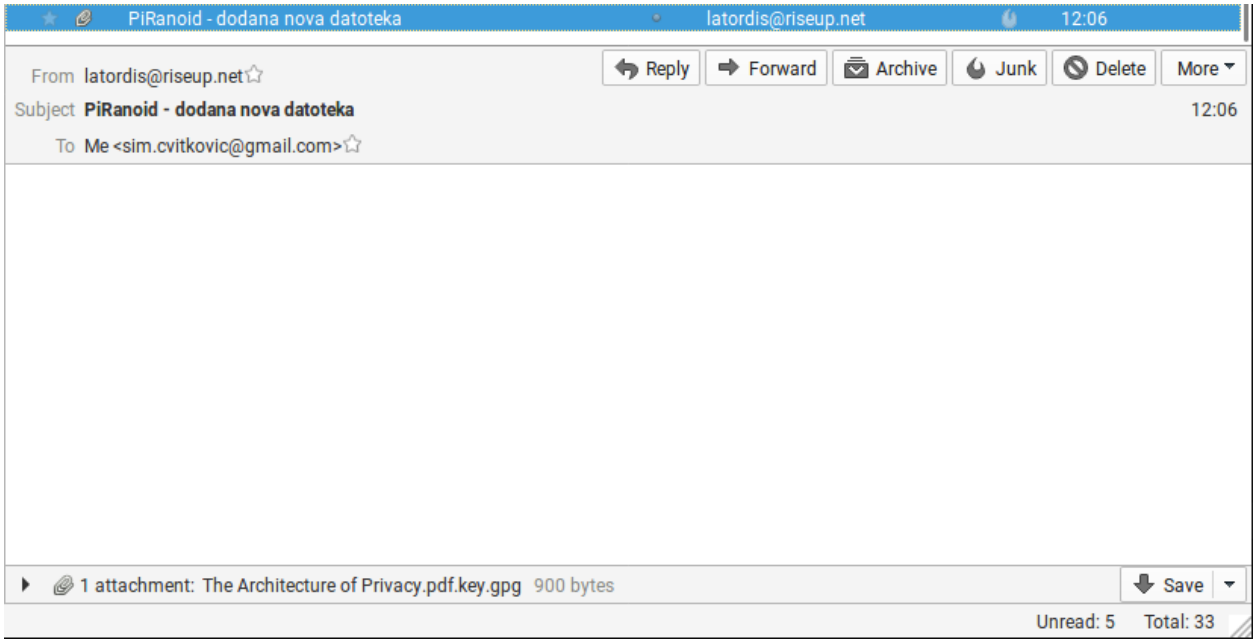
Programski kod 4.2. glavni dio programa

```
1.  def directory_changed(path):
2.
3.      print('Directory Changed: %s' % path)
4.      gpg = gnupg.GPG(gnupghome='/home/pi/.gnupg')
5.      gpg.encoding = 'utf-8'
6.      filekeys = os.listdir('/var/piranoidKeys/')
7.
8.      with open ("email_list") as file:
9.          recepients = file.read().splitlines()
10.
11.     server = smtplib.SMTP('smtp.riseup.net', 587)
12.     server.starttls()
13.     server.login("latordis", "password")
14.     print("Authenticated user!")
15.
16.     for filekey in filekeys:
17.         filekeypath = str('/var/piranoidKeys/'+filekey)
18.         print (filekeypath)
19.         filekeyinfo = open(filekeypath, "rb")
20.         filekeygpg = filekey + '.gpg'
21.         filekeygpgpath = filekeypath + '.gpg'
22.         for recepient in recepients:
23.
24.             encrypted_data= gpg.encrypt_file(filekeyinfo, recepient, output=filekeyg
    pgpath)
25.             fromaddr = "latordis@riseup.net"
26.             toaddr = recepient
27.             msg = MIMEMultipart()
28.             msg['From'] = fromaddr
29.             msg['To'] = toaddr
30.             msg['Subject'] = "PiRanoid - dodana nova datoteka"
31.
32.             fp = open(filekeygpgpath, "rb")
33.             attachment = MIMEApplication(fp.read(), Name=filekeygpg)
34.             fp.close()
35.
36.             attachment.add_header("Content-
    Disposition", "attachment", filename = filekeygpg)
37.
38.             msg.attach(attachment)
39.             text = msg.as_string()
40.             server.sendmail(fromaddr, toaddr, text)
41.             print("Message sent")
42.
43.     filekeyinfo.close()
44.     server.quit()
45.     os.remove(filekeypath)
46.     os.remove(filekeygpgpath)
```

Programski kod 4.3. funkcija *directory_changed*

Prema programskom kodu 4.3. korištenjem PyQt-ovih signala i pretinaca može se vidjeti da se emitiranjem signala poziva funkcija *directory_changed*. U funkciji se izvršava serija naredbi, a mnoge su slične kao u skripti *mail-test.py*. Ova skripta opaža kada je nova datoteka postavljena u direktoriju *piranoid*, a zatim u direktoriju za ključeve *piranoidKeys* sprema popis datoteka u listu *filekeys*. Zatim u petlji za svaki ključ formira putanje i imena datoteka. Datoteka s ključem u

direktoriju ima ekstenziju *.key*, a kriptirana datoteka s ključem ima još dodanu ekstenziju *.gpg*. Datoteka s ključem se kriptira javnim ključem primatelja i zapisuje se u isti direktorij u istoimenu datoteku s dodatkom *.gpg* u imenu kako bi se razlikovala od datoteke s ključem. Kriptirana datoteka se šalje kao privitak, i korisnik dobiva elektroničku poštu koja ga obavještava da je dodana nova datoteka. Nakon što se svima pošalje elektronička pošta, skripta briše datoteku s ključem i zadnju verziju kriptirane datoteke.



Slika 4.2. Prikaz elektroničke pošte s kriptiranim privitkom ključa

4.1.3. Skripta *new-piranoid-launcher.sh*

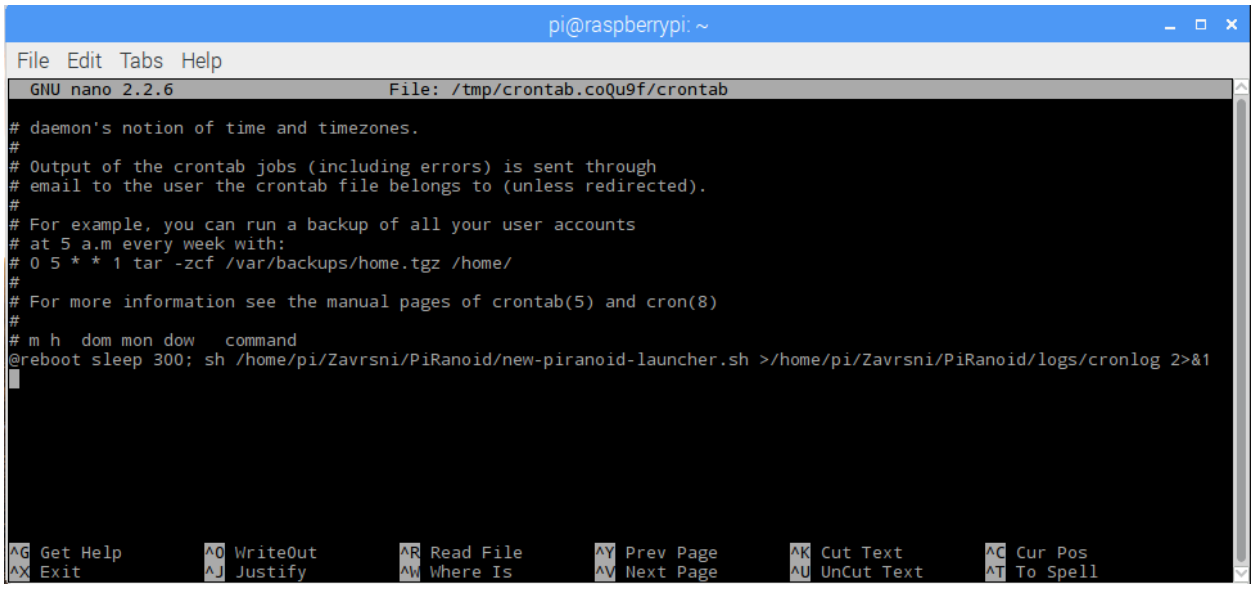
Kako je aplikacija na strani poslužitelja, poželjno je automatizirati proces njenog pokretanja. Za automatizaciju naredbi na Linux operacijskim sustavima postoji skriptni jezik Bash. Napisana je skripta koja izvršava set naredbi kako bi poslužitelj bio spreman i dostupan za korištenje na klijentskoj strani.

```
1.  #!/bin/sh
2.  #new-piranoid-launcher.sh
3.  cd /home/pi
4.  sudo service tor start
5.  cd /home/pi/Zavrsni/PiRanoid
6.  sudo python3 mail-test.py
7.  sudo python3 dir-watcher.py
8.  cd /home/pi
```

Programski kod 4.4. *new-piranoid-launcher.sh*

U Bash skripti, prema programskom kodu 4.4., vidljiv je set instrukcija za Linux operacijski sustav koje omogućavaju dostupnost i pravilan rad poslužitelja. Prvo se mijenja direktorij u početni, zatim se pokreće Tor. Nakon pokretanja Tora skripta mijenja direktorij u *PiRanoid* koji sadržava prije opisane skripte. Zatim se skripte pokreću u Python3 interpreteru. Valja opaziti *sudo* naredbu. Ona označava da će se naredba izvršiti kao *root* korisnik, što znači da izvršavanje naredbe kao *root* ima sve ovlasti nad datotekama i direktorijima te procesima u cijelom datotečnom sustavu, odnosno operacijskom sustavu. Po završetku, skripta mijenja direktorij u početni. Direktorij *pi* je početni jer je osnovni korisnik *pi* preko kojeg je sav posao na poslužiteljskoj strani obavljen.

Samim stvaranjem skripte proces automatizacije nije završen. Zato se uređuje datoteka *crontab* u koju se mogu upisivati naredbe koje se izvršavaju ovisno o vremenu.



Slika 4.3. izgled *crontab* datoteke

Naredba zapisana na kraju datoteke govori operacijskom sustavu da pokrene skriptu nakon ponovnog pokretanja sustava, ali nakon što prođe 300 sekundi. To daje vremena operacijskom sustavu da pokrene sve potrebne procese, pa se zatim skripta izvršava. *2>&1* označava da *cron* ne šalje elektroničku poštu u slučaju zapisa izlaza, već je navedeno da se standardni izlaz zapisuje u datoteku *cronlog* koja se nalazi u direktoriju *logs* kako bi administratoru poslužitelja bilo lakše pratiti rad skripti, ili eventualne pogreške.

4.2. Klijent

Aplikacija na strani klijenta koristi četiri Python datoteke. Datoteke *login.py* i *main_window.py* su rezultati kreiranih formi u Qt Designer programu te prevedene iz XML oblika u Python. O njima neće biti detaljnije analize jer su dizajn grafičkog sučelja koje se iscrtava pokretanjem aplikacije, ali se njihovi elementi koriste u glavnoj datoteci *main.py*. Zadnja datoteka je *encode.py* i ona je programsko rješenje s implementiranim funkcijama za AES kriptiranje datoteka te je preuzeta iz literature, prema [11]

4.2.1. Skripta *main.py*

Prema programskom kodu 4.5. pokazana je deklaracija dvije klase i njihovi konstruktori. Klasa *LoginDialog* označava klasu kojom je opisana forma za prijavu korisnika. Atribut *.ui* označava grafičko sučelje, a on je zapravo objekt klase *Ui_LoginDialog* koji preuzima sve atribute i metode iz *login.py* kao što se može vidjeti iz linije 4. U konstruktoru se još poziva i metoda *setupUi()* kojom se postavljaju pravila položaja elemenata u prozoru, veličina prozora itd. Klasa *mainPiranoid* ima analognu deklaraciju, samo se ne radi o prozoru za prijavu korisnika, već o glavnom prozoru preko kojeg se razmjenjuju datoteke.

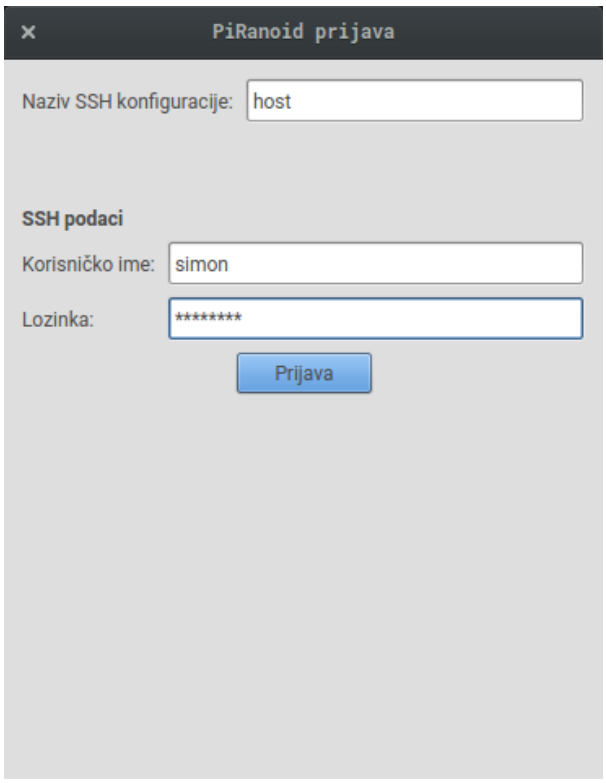
```
1.  import sys, paramiko, getpass, encode, os
2.  from PyQt4 import QtCore, QtGui
3.  from login import Ui_LoginDialog
4.  from main_window import Ui_MainWindow
5.
6.  class LoginDialog(QtGui.QDialog):
7.      def __init__(self, parent=None):
8.          super(LoginDialog, self).__init__(parent)
9.
10.         self.ui = Ui_LoginDialog()
11.         self.ui.setupUi(self)
12.
13.
14.
15. class mainPiranoid(QtGui.QMainWindow):
16.     def __init__(self, parent=None):
17.         super(mainPiranoid, self).__init__(parent)
18.
19.         self.ui = Ui_MainWindow()
20.         self.ui.setupUi(self)
```

Programski kod 4.5. deklariranje klasa *LoginDialog* i *mainPiranoid*

Prema programskom kodu 4.6. prikazan je konstruktor za klasu *SftpClient*. Klasa u konstruktor prima parametre *loginDialog* i *mainWindow* koji označavaju objekte prozora ranije spomenutih klasa u programskom kodu 4.5. Na taj način moguće je unutar klase inicijalizirati objekte te kasnije pristupati njihovim atributima. Postoji niz atributa koji se inicijaliziraju unutar konstruktora. Atribut *user* koji dobiva vrijednost imena trenutnog Linux korisnika, *ssh* postaje objekt za uspostavljanje i održavanje veze čijim atributima se kasnije manipulira preko metoda. *conf je* atribut koji postaje objekt za manipulaciju SSH konfiguracijskim datotekama. Atributi *logindialog* i *mainwindow* su objekti koji predstavljaju prozore. Metoda *show()* iscrtava grafičko sučelje, u ovom slučaju iscrtava se prozor za prijavu korisnika. Navedeni su i signali i pretinci koji određene objekte, tj. njihovo emitiranje signala povezuju s metodama klase koji su opisani u nastavku.

```
1.  class SftpClient:
2.      def __init__(self, loginDialog, mainWindow):
3.          self.srv_address = ""
4.          self.uname  = ""
5.          self.passwd = ""
6.          self.conf = paramiko.SSHConfig()
7.          self.user = getpass.getuser()
8.          self.ssh = paramiko.SSHClient()
9.          self.logindialog = loginDialog
10.         self.mainwindow = mainWindow
11.         self.logindialog.show()
12.
13. ###### SIGNALS && SLOTS #######
14.         self.logindialog.ui.loginButton.clicked.connect(self.start)
15.         self.mainwindow.ui.downloadButton.clicked.connect(self.file_download)
16.         self.mainwindow.ui.chooseFileButton.clicked.connect(self.file_choose)
17.         self.mainwindow.ui.uploadButton.clicked.connect(self.file_upload)
18.         self.mainwindow.ui.quitButton.clicked.connect(self.quit)
```

Programski kod 4.6. Konstruktor klase *SftpClient*



Slika 4.4. Prozor za prijavu korisnika

Na slici 4.4. prikazan je prozor za prijavu korisnika. Korisnik ispunjava podatke potrebne za spajanje s poslužiteljem. Kako bi prijava bila uspješna, na računalu klijenta Tor program mora biti pokrenut. Prva linija označava ime konfiguracije u *config* datoteci SSH programa. Na taj način se ova konfiguracija razlikuje od ostalih SSH konfiguracija za spajanje na neka druga korisnička

računala. Primjer ispravne konfiguracije SSH programa za ovu aplikaciju može se vidjeti na slici 4.5.

```
1  host host
2  user -
3  hostname jbrrdr75vaeivqgw.onion
4  CheckHostIP no
5  Compression yes
6  Protocol 2
7  ProxyCommand connect-proxy -S localhost:9050 %h %p
8
```

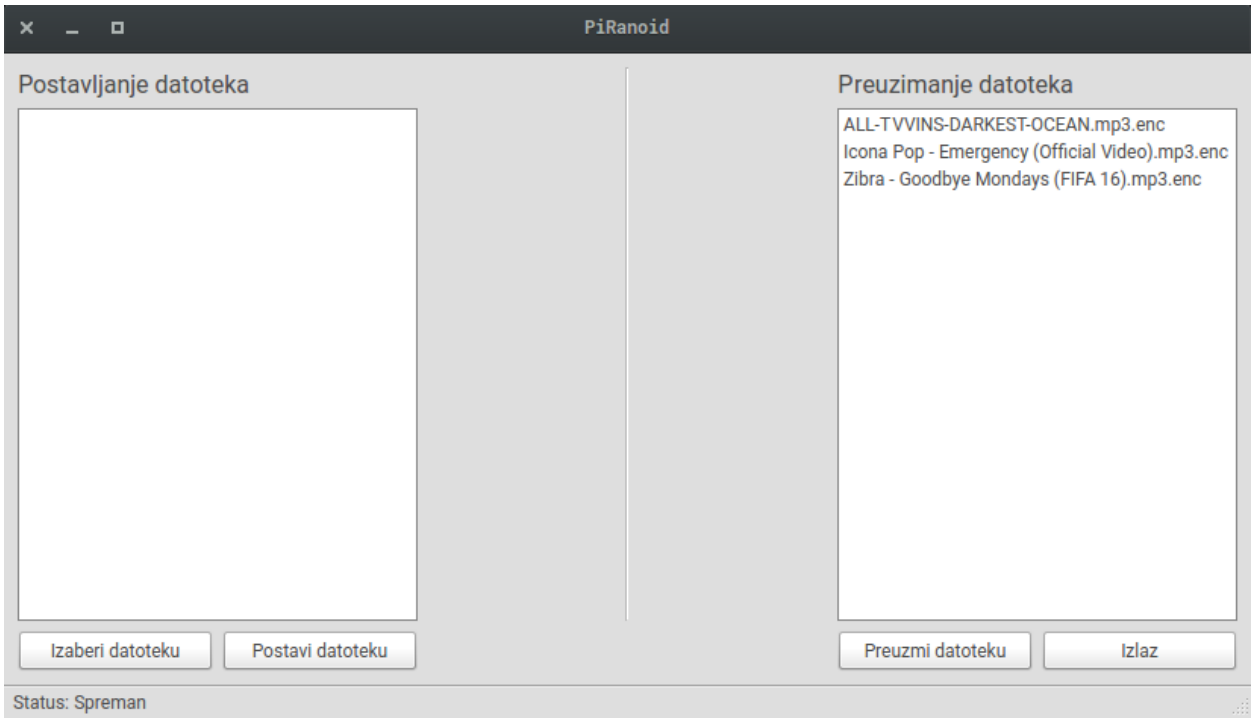
Slika 4.5. Izgled *config* datoteke.

Kao što je ranije spomenuto, SSH protokol po dogovoru koristi port 22 za konekciju i prijenos podataka. Programi koji koriste Tor mrežu promet moraju preusmjeriti na port 9050 koja su standardna za Tor promet. Adresa *jbrrdr75vaeivqgw.onion* je adresa na kojoj se može pronaći poslužitelj aplikacije (nevidljiva, skrivena usluga), uz posjedovanje šifre koja je pohranjena u *torrc* datoteci – datoteci za konfiguraciju Tor programa.

```
1.  def start(self):
2.          self.configName = str(self.logindialog.ui.srvAddrLine.text())
3.          self.uname = str(self.logindialog.ui.usernameLine.text())
4.          self.passwd = str(self.logindialog.ui.passwordLine.text())
5.          self.remotepath = "/var/piranoid"
6.          self.conf.parse(open('/home/' + self.user + '/.ssh/config'))
7.          self.host = self.conf.lookup(self.configName)
8.          self.proxy = paramiko.ProxyCommand(self.host['proxycommand'])
9.
10.         self.ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
11.         self.ssh.connect(self.srv_address, username=self.uname, password=self.passwd
    ,  sock=self.proxy)
12.         self.sftp = paramiko.SFTPClient.from_transport(self.ssh.get_transport())
13.         if ( self.ssh.get_transport().is_active()):
14.             self.initialize_interface()
```

Programski kod 4.7. Metoda *start*

U programskom kodu 4.6. na liniji 14. povezani su signal i pretinac. Signal se emitira nakon klika mišom na gumb *Prijava* i on pokreće metodu u pretincu, a ta metoda je *start*. Prva tri atributa u metodi dobivaju vrijednosti iz polja za unos teksta u prozoru za prijavu korisnika. Atribut *remotepath* označava direktorij na poslužitelju namijenjenom za dijeljenje datoteka. Zatim slijedi rastavljanje teksta iz datoteke *config* i spremanje pojedinih parametara koji služe za povezivanje SSH protokolom. Nakon prijave na poslužitelja pokreće se SFTP sjednica koristeći SSH protokol. Ukoliko je veza aktivna poziva se metoda *initialize_interface()* koja zatvara prozor za prijavu i otvara glavni prozor, osvježen s trenutnim stanjem direktorija na poslužitelju, koji se može vidjeti na slici 4.6.



Slika 4.6. Prikaz glavnog prozora aplikacije

U glavnom prozoru postoje dvije liste: jedna koja prikazuje odabrane datoteke koje korisnik zatim može postaviti na poslužitelj, i jedna koja izlistava kriptirane datoteke na poslužitelju, a koje korisnik može preuzeti. Na traci za status prikazuje se stanje aplikacije koje može biti *Spreman*- označava da je aplikacija slobodna za odabir i prijenos datoteka ili *Prijenos* koji označava da se datoteka prenosi, a ima dodatne informacije koje prikazuju koliko bajta aplikacije je prenešeno od ukupnog broja bajta koji se moraju prenijeti.

Klikom na gumb *Izaberi datoteku* otvara se prozor koji prikazuje datotečni sustav s direktorijima. Korisnik tada odabire datoteku i njena putanja se dodaje u listu za postavljanje datoteka. Korisnik zatim treba odabrati datoteku na listi ili dodati još datoteka. Moguće je prenositi jednu datoteku istovemeno tako da postavljanje datoteke ovisi o odabiru s liste. Postavljanje datoteke na poslužitelj započinje klikom na gumb *Postavi datoteku* kojim se poziva metoda *file_upload.*

```
1.  def file_upload(self):
2.          localpath = str(self.mainwindow.ui.uploadList.currentItem().text())
3.
4.          path_split = localpath.split("/")
5.          listlen = len(path_split)
6.          upfilename = path_split[listlen - 1]
7.
8.          encode.encrypt_file(encode.key, localpath, None)
9.
10.         keyfilename = upfilename + '.key'
11.         keyfile = open(keyfilename, "w")
12.         keyfile.write(encode.key)
```

```
            keyfile.close()

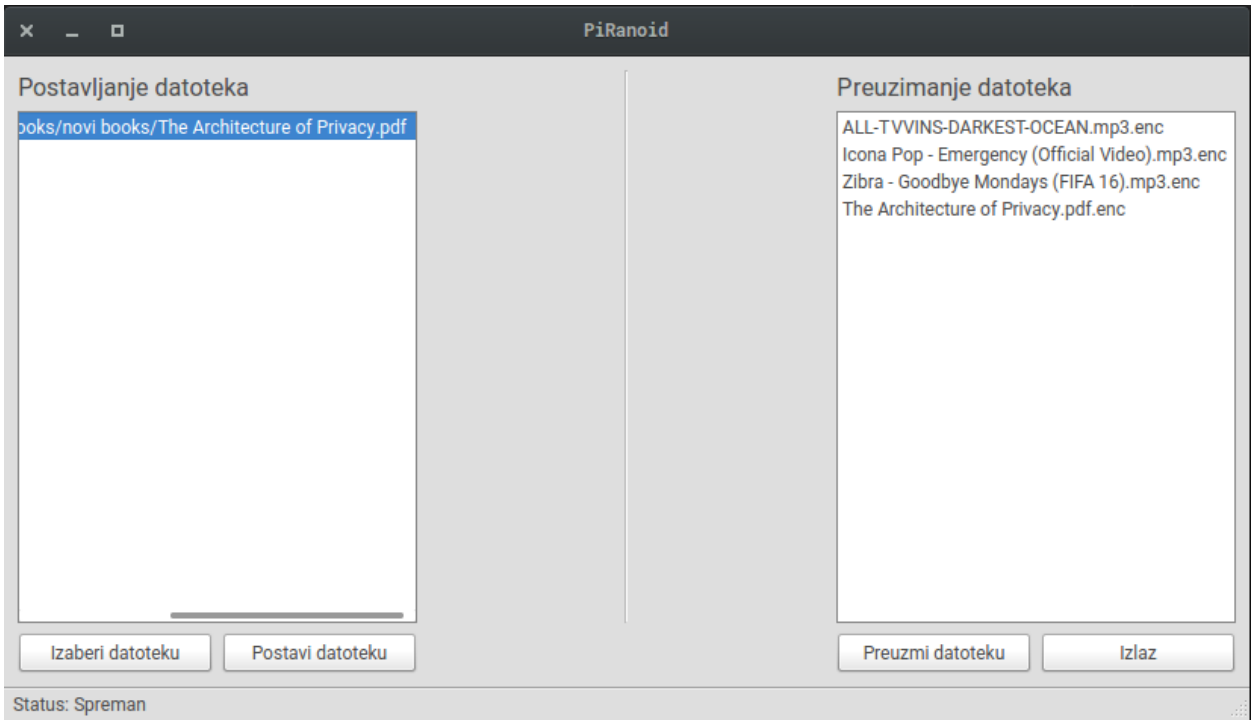
            currentpath = os.getcwd() + '/' + keyfilename
            self.sftp.put(currentpath, '/var/piranoidKeys/' + keyfilename, callback=sel
    f.print_totals)
            self.sftp.put(localpath + '.enc', self.remotepath + '/' + upfilename + '.en
    c', callback=self.print_totals)


            self.mainwindow.ui.statusbar.showMessage("Status: Spreman")

            file_list = self.sftp.listdir(self.remotepath)
            self.mainwindow.ui.downloadList.clear()
            self.mainwindow.ui.downloadList.addItems(file_list)
```

Programski kod 4.8 Metoda *file_upload*

Prije prijenosa datoteke na poslužitelj, potrebno je odrediti parametre koji se koriste kod postavljanja na poslužitelj. Odabrana putanja do datoteke s liste se sprema u varijablu localpath i onda se nizom naredbi za rastavljanje putanje dolazi do imena datoteke. Poziva se funkcija iz datoteke *encode.py* naziva *encrypt_file* koja obavlja AES-128 bitno kriptiranje odabrane datoteke te stvara novu datoteku istog naziva s dodatnom ekstenzijom *.enc*. Zatim se stvara nova datoteka za pohranu generiranog ključa za dekriptiranje datoteke. Ključ se zapisuje u datoteku istog naziva kao datoteke koju korisnik želi postaviti na poslužitelja samo što se dodaje ekstenzija *.key*. Nakon što postoji kriptirana verzija datoteke i datoteka s ključem, započinje njihovo postavljanje. Prvo se postavlja datoteka s ključem u direktorij *piranoidKeys*, a zatim kriptirana datoteka. Kada postavljanje završi, osvježava se lista s datotekama na poslužitelju, a poslužitelj obavlja proces opisan u potpoglavlju 4.1.2., tj. korisnici će elektroničkom poštom dobiti kriptiranu verziju ključa, i datoteke s ključevima će se obrisati s poslužitelja.

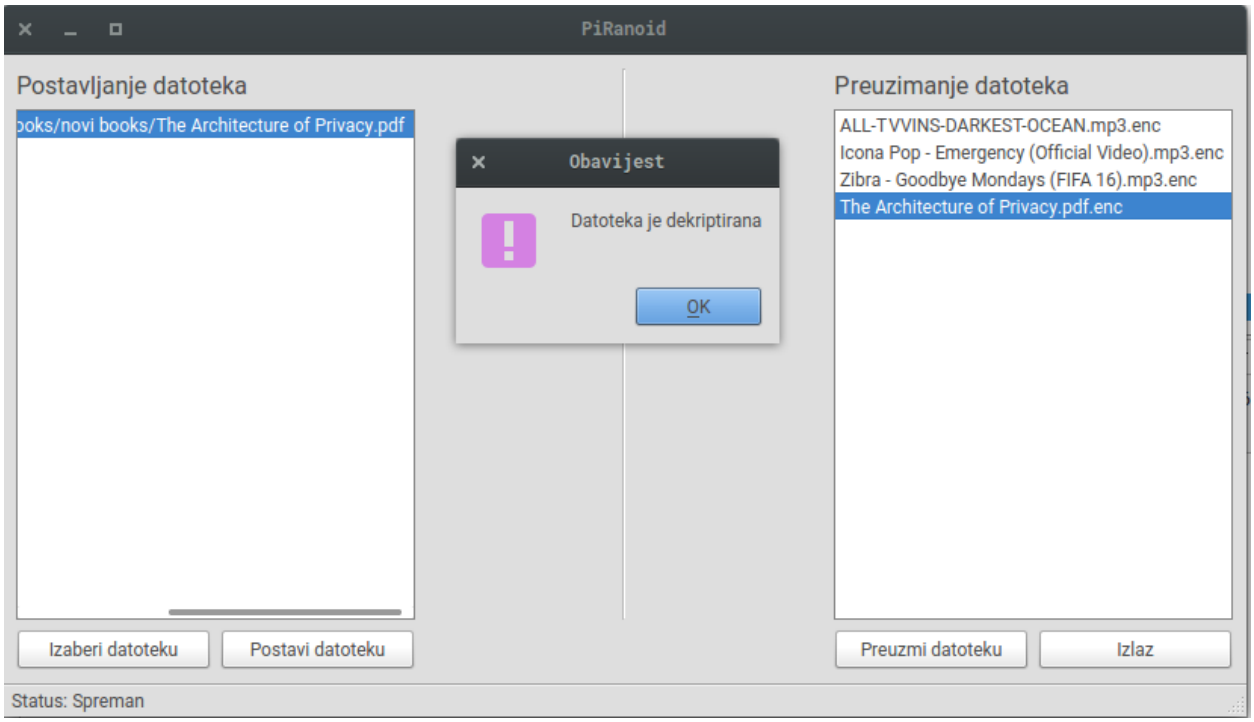


Slika 4.7. Prikaz stanja aplikacije nakon postavljanja datoteke

```
1.  def file_download(self):
2.          fileName = str(self.mainwindow.ui.downloadList.currentItem().text())
3.
4.
5.          self.sftp.get(self.remotepath + '/' + fileName, '/home/' + self.user + '/Dow
    nloads/' + fileName, callback=self.print_totals)
6.          self.mainwindow.ui.statusbar.showMessage(„Status: Spreman“)
7.          self.localencfile = '/home/' + self.user + '/Downloads/' + fileName
8.          self.key_choose()
9.
10. def key_choose(self):
11.         fileDialog= QtGui.QWidget()
12.
13.         fpath = QtGui.QFileDialog.getOpenFileName(fileDialog, 'Izaberi kljuc', '/hom
    e')
14.         localpath = str(fpath)
15.
16.         fopen = open(localpath,"r")
17.         fkey = fopen.read()
18.         fopen.close()
19.         encode.decrypt_file(fkey, self.localencfile, None)
20.         self.showMessageBox('Obavijest', 'Datoteka je dekriptirana')
```

Programski kod 4.9. Metode *file_download* i *key_choose*

Odabirom datoteke s liste za preuzimanje i klikom na gumb *Preuzmi datoteku*, sprema se ime datoteke. Zatim se obavlja preuzimanje datoteke. Nakon što preuzimanje datoteke završi, korisniku se otvara prozor za izbor datoteke s ključem, a to je rezultat poziva metode *key_choose*. Nakon izbora datoteke s ključem poziva se funkcija *decrypt_file* iz datoteke *encode.py* koja dekriptira datoteku i sprema u isti direktorij gdje je spremljena i preuzeta kriptirana datoteka-direktorij *Downloads*.



Slika 4.8. Prikaz stanja aplikacije nakon preuzimanja i dekriptiranja datoteke

5. ZAKLJUČAK

U radu su opisane tehnologije potrebne za rad i razvoj aplikacije za anonimnu razmjenu datoteka, mehanizam za anonimnu razmjenu datoteka te programska implementacija iste aplikacije na strani poslužitelja i klijenta u Linux okruženju. Poslužiteljsku stranu aplikacije za anonimnu razmjenu datoteka moguće je instalirati na SoC uređaj– što je dokazano korištenjem Raspberry Pi-a kao poslužitelja.

Na temelju mehanizma za anonimnu razmjenu datoteka, osmišljen je model aplikacije na poslužiteljskoj i na klijentskoj strani. Programska implementacija je izvršena koristeći Tor program za anonimizaciju prometa na internetu i Python programski jezik za implementaciju na poslužiteljskoj strani u obliku skripti koje su automatizirane na način da se pokreću pri podizanju operacijskog sustava. Na klijentskoj strani aplikacija implementirana pomoću Pythona koristi grafičko korisničko sučelje za lakšu i intuitivniju uporabu. Protokol koji se preko Tora koristi za spajanje klijenta na poslužitelj je SSH, a za prijenos datoteka SFTP koji se temelji na SSH protokolu. Datoteke su kriptirane 128-bitnim simetričnim AES algoritmom, a razmjena ključeva se vrši elektroničkom poštom kriptiranom Open PGP standardom koji se temelji na asimetričnom kriptiranju.

Pogodnosti korištenja aplikacije su brojne. Besplatno, jednostavno i sigurno postavljanje poslužitelja na internet. Nemogućnost pristupa poslužitelju svima osim ciljanim korisnicima usluge. Dostupnost poslužitelja na bilo kojoj geolokaciji na kojoj je dostupan pristup internetu. Anonimna razmjena datoteka, sigurno okruženje rada za korisnike te zaštita datoteka kriptiranjem za slučaj kompromitiranja poslužitelja. Aplikaciju može koristiti svaka grupa ljudi koja ima potrebu za korištenjem anonimizacije aktivnosti na internetu ili sigurnim prijenosom datoteka u grupi.

Iako aplikacija radi bez kritičnih pogrešaka koje bi onemogućavale nastavak rada ili dovele do urušavanja aplikacije, postoje nedostaci koji se mogu programski izbjeći. Jedan nedostatatak aplikacije je nemogućnost razmjena datoteka čiji naziv nije kompatibilan s ASCII standardom pa korisnici moraju paziti na nazive datoteka koje žele razmjenjivati. Drugi nedostatak aplikacije je praćenje pogrešaka u SSH konekciji s poslužiteljem. U slučaju prekida veze s poslužiteljem, iznimke će se ispisati u konzoli, ali ne i u korisničkom sučelju što može otežati rad korisniku. U slučaju prekida veze, korisnik mora zatvoriti aplikaciju te ju ponovno pokrenuti. Integritet

podataka se podrazumijeva korištenjem Tor-a i SSH veze koje su kriptirane, ali bilo bi korisno implementirati funkcije za provjeru integriteta datoteka na temelju *hash* algoritama.

LITERATURA

[1] Wikimedia Foundation, [online] , Wikimedia Foundation, 28. srpnja 2017. dostupno na: https://en.wikipedia.org [28. srpnja 2017.]

[2] Tor Project, Tor: Overview [online], Tor Project, dostupno na: https://www.torproject.org/about/overview.html.en [27. srpnja 2017.]

[3] Tor Project, Tor: Hiden Service Protocol [online], Tor, dostupno na: https://www.torproject.org/docs/hidden-services.html.en [27. srpnja 2017.]

[4] Tor Project, Tor Rendezvous Specification [online], Tor, dostupno na: https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt#n928

[5] D. J. Barret, R. E. Silverman, R. G. Byrnes, SSH, The Secure Shell: The Definitive Guide, O'Reilly, 2005.

[6] GNU Project, GNUPG – The Universal Crypto Engine [online], GNU Project, dostupno na: https://www.gnupg.org/software/index.html , [23. srpnja 2017.]

[7] L. V. Houtven, Crypto 101, 2017.

[8] The Qt Company Ltd., Qt Documentation, Qt Designer Manual [online], The Qt Company Ltd., dostupno na: http://doc.qt.io/qt-4.8/designer-manual.html [25. lipnja 2017.]

[9] D. Boodie About PyQt [online], Python Software Foundation, 8. Lipnja 2015., dostupno na: https://wiki.python.org/moin/PyQt , 17. srpnja 2017.

[10] Riseup Collective, Privacy Policy [online], Riseup Collective, dostupno na: https://riseup.net/en/privacy-policy [23. srpnja 2017.]

[11] A. Asayev, Encode.py [online], Github, 2015., dostupno na: https://gist.github.com/dreikanter/2780734 [17. srpnja 2017.]

SAŽETAK

Glavni problem rada je bio osmisliti mehanizam za anonimnu razmjenu datoteka te modelirati i programski implementirati što bi rezultiralo aplikacijom na klijentskoj i poslužiteljskoj strani u Linux okruženju. Anonimizacija je postignuta korištenjem Tor programa, a siguran i pouzdan prijenos podataka omogućen je SSH protokolom. Prijenos datoteka omogućen je SFTP protokolom. Aplikacija na poslužiteljskoj strani obavještava klijente kada je dostupna, preuzima datoteke s ključevima za dekriptiranje kriptiranih datoteka koje klijenti postavljaju te ih razašilje klijentima kriptiranom elektroničkom poštom i briše ključeve kako bi kriptirane datoteke ostale kriptirane i nakon eventualne kompromitacije poslužitelja. Klijentska strana aplikacije omogućava korisnicima postavljanje datoteka koje se prije postavljanja kriptiraju. Također im omogućava preuzimanje datoteka koje su kriptirane na poslužitelju i dekriptiraju se na klijentskoj strani odabirom ispravnog ključa koji je klijent dobio elektroničkom poštom poslanom od poslužitelja. Instalacija poslužiteljska strane aplikacije je omogućena za SoC uređaje.

Ključne riječi: anonimnost, Linux, Tor, SSH, Python

ABSTRACT

Application for anonymous file trasnfer

The main problem of this thesis was to design a mechanism for anonymous file transfer and programmatically implement this mechanism. Solution of the problem resulted in a client-side and server-side application in Linux environment. Anonymisation was achieved by using Tor software, and secure and reliable data transfer was achieved by using SSH protocol. File transfer was enabled by using SFTP protocol. The server-side application informs clients when it is available, handles files with keys for decryption of files which clients upload, encrypts files with keys and sends them to clients using encrypted e-mails and then deletes files with keys so that encrypted files stay encrypted in case of server compromisation. Client-side application enables its users to upload files which are being encrypted before upload. It also enables its users to download files which are encrypted on the server, and are decrypted on the client-side by choosing the corresponding key for decrypting which user had got by e-mail sent by the server. Installation of the server-side application is enabled for SoC devices.

Keywords: anonymity, Linux, Tor, SSH, Python

ŽIVOTOPIS

Simon Cvitković rođen je 25. Srpnja 1994. godine u Zagrebu. Od rođenja živi u gradu Kutini, a osnovnoškolsko obrazovanje stječe u OŠ Stjepan Kefelja. Godine 2009. upisuje prirodoslovno-matematičku gimnaziju SŠ Tina Ujevića u kojoj sve razrede prolazi s odličnim uspjehom. Maturira 2013. godine i iste godine upisuje Fakultet strojarstva i brodogradnje, Sveučilišta u Zagrebu. Iduće godine ispisuje se s fakulteta u Zagrebu te upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.