Sigurnosna nadogradnja sustava elektroničke pošte

Tomšić, Dominik

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:549607

Rights / Prava: Attribution-NonCommercial-ShareAlike 3.0 Unported/Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 3.0

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

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Tomšić

SIGURNOSNA NADOGRADNJA SUSTAVA ELEKTRONIČKE POŠTE

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Tomšić

Matični broj: 0016129175

Studij: Informacijski sustavi

SIGURNOSNA NADOGRADNJA SUSTAVA ELEKTRONIČKE POŠTE

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Nikola Ivković

Varaždin, rujan 2020.

Dominik Tomšić

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovaj rad bavi se praktičnom temom poboljšanja sigurnosti poruka poslanih putem elektroničke pošte. Kako bi se postiglo željeno sigurnosno poboljšanje korištena je Diffie-Hellmanova razmjena ključeva, a za kriptiranje sadržaja e-pisma algoritam AES. Razmjena ključeva implementirana je pomoću udaljenog poslužitelja na kojem se spremaju potrebni javni ključevi. Na udaljenom poslužitelju pokrenut je program koji automatski sprema pristigle javne ključeve i također na zahtjeve za određenim ključevima odgovara putem elektroničke pošte. Kreiranje ključeva i kriptiranje poruka implementirano je u vlastitom klijentu za e-poštu. Vlastiti klijent za e-poštu također ima implementirane funkcije potrebne za komunikaciju s udaljenim poslužiteljem. Željeni cilj uspješno je postignut te je sustav implementiran.

Ključne riječi: Diffie-Hellman; AES; elektronička pošta; računalna sigurnost

Sadržaj

Sadržaj v
1. Uvod 1
2. Metode i tehnike rada 2
3. Koncepti potrebni za implementaciju sigurnosne nadogradnje elektroničke pošte 3
3.1. Jednostavni protokol za prijenos pošte 3
3.2. Napredni standard za enkripciju 3
3.3. Diffie-Hellmanova razmjena ključeva 4
4. Implementacija sigurnosne nadogradnje sustava elektroničke pošte 6
4.1. Skica funkcionalnosti 6
4.2. Implementacija klijenta e-pošte 8
4.2.1. Klijent e-pošte grafički 8
4.2.2. Pojašnjenje programskog koda klijenta e-pošte 11
4.2.2.1. Klasa „Person“ 11
4.2.2.2. Klasa „ReadMail“ 12
4.2.2.3. Klasa „SendMail“ 14
4.3. Implementacija poslužitelja 16
4.3.1. Instalacija poslužitelja e-pošte 16
4.3.2. Program za automatsko odgovaranje na zahtjeve – autores 17
4.4. Implementacija kriptiranja i dekriptiranja poruka 22
4.4.1. Stvaranje i spremanje para privatnog i javnog ključa primatelja 22
4.4.2. Enkripcija poruke 24
4.4.3. Dekripcija poruke 28
4.5. Prikaz rada sustava 30
5. Zaključak 35
Popis literature 36
Popis slika 37

1. Uvod

Tajne su nešto prirodno za svakog čovjeka, tako su ljudi već u antičkim vremenima pokušavali otkriti načine kako napisane poruke učiniti neshvatljivima za sve osim namijenjenih čitatelja. Kroz povijest su se tako razvile razne metode za prikrivanje pravih značenja pisama, od jednostavnih zamjena određenih pisanih znakova drugima i zaključavanja pisama u prenosive kutijice do vrlo složenih uređaja poput enigme koju je koristila Njemačka strana za šifriranje vojnih tajni u vrijeme Drugog svjetskog rata.

U današnje vrijeme računala i računalnih mreža privatne i poslovne poruke se sve više šalju elektroničkim putevima, pa se tako moderna kriptografija primarno bavi računalnim šifriranjima poruka. Elektronička pošta ne osigurava buduću tajnost (eng. *Forward secrecy*) te ovaj rad obrađuje i implementira jedan od mogućih načina osiguravanja buduće tajnosti.

Područje sigurnosti meni kao studentu vrlo je zanimljivo te me je tema ovog rada potaknula na dodatno proučavanje načina za sigurni prijenos informacija od jedne osobe do druge, bez da ih treća strana može pročitati.

2. Metode i tehnike rada

Prilikom izrade ovog rada, najprije je provedeno teoretsko istraživanje teme kroz knjige i web izvore navedene u literaturi.

Za izradu potrebnih programskih rješenja korišteni su programski jezici C# i C++. Za programiranje u jeziku C# korišten je „Visual Studio 2019“, dok je za programiranje u jeziku C++ korišten Linux uređivač teksta „nano“.

Za izradu klijenta za e-poštu nisu bili potrebni dodatni resursi osim osobnog računala, no za izradu udaljenog poslužitelja bilo je potrebno nabaviti pristup udaljenom poslužitelju i nabaviti domenu kako bi on imao adresu.

Također potrebno je napomenuti kako je korišten „Postfix“ poslužitelj e-pošte na udaljenom poslužitelju i „OpenPop.Net“ NuGet paket u „Visual Studio 2019“ okruženju koji omogućava spajanje na poslužitelj elektroničke pošte te čitanje i slanje poruka.

3. Koncepti potrebni za implementaciju sigurnosne nadogradnje elektroničke pošte

Ovo poglavlje sadrži informacije o najvažnijim konceptima koji se koriste u kasnijem dijelu rada prilikom implementacije sustava za sigurnosnu nadogradnju elektroničke pošte.

3.1. Jednostavni protokol za prijenos pošte

S obzirom da je cilj ovog rada poboljšati sigurnost elektroničke pošte, najprije je potrebno spomenuti jednostavni protokol za prijenos pošte (eng. *Simple Mail Transfer Protocol*, kraće SMTP) koji je dominantni protokol za slanje elektroničke pošte. SMTP protokol specificiran je 1982. u RFC 821. SMTP je niz naredbi koje se šalju od jednog poslužitelja e-pošte do drugog kako bi se omogućilo slanje elektroničke pošte gdje pošiljatelj primatelju šalje naredbe s informacijama o svojim namjerama, a primatelj mu odgovara ovisno o stanju pristiglih podataka [1]. Sadržaj poruka poslanih koristeći SMTP nije posebno kriptiran te bi ga mogao dokučiti netko tko prisluškuje mrežni promet i ovdje nastaje potreba za sigurnosnim poboljšanjem.

3.2. Napredni standard za enkripciju

Kako bi se poruke poslane koristeći SMTP mogle smatrati sigurnima potrebno je njihov sadržaj na neki način kriptirati. U ovom radu će se za tu enkripciju koristiti napredni standard za enkripciju (eng. *Advanced Encryption Standard*, kraće AES).

AES kao standard omogućava kriptiranje podataka koristeći ključeve veličina 128, 192 i 256 bitova. AES je simetričan algoritam, što znači da željene podatke kriptira i dekriptira koristeći isti ključ. AES je kružni algoritam koji početni zapis kriptira kroz nekoliko krugova, ovisno o veličini ključa te je za ključ od 256 bitova, koji će biti korišten u ovom radu, broj krugova 14. Podaci se dijele na blokove veličine ključa i zatim se svaki blok kriptira zasebno. Također ključ se u svakom krugu matematički mijenja te se za početak ciklusa koristi inicijalizacijski vektor, vrijednost koja ne mora biti tajna, ali mora biti unikatna za svaku primjenu ključa [2].

Proces kriptiranja AES-om može se opisati slijedećim koracima, s time da se podcrtani koraci ponavljaju u slijedu 9, 11 ili 13 puta, ovisno o veličini ključa koji se koristi [3]:

- Proširenje ključa

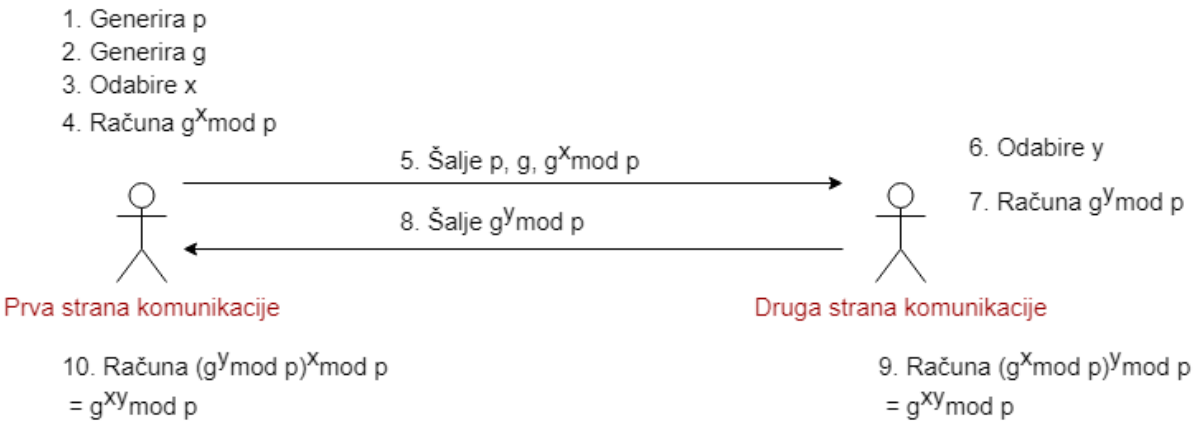
- Dodavanje kružnog ključa
- Zamjena bajtova
- Pomicanje redova
- Miješanje stupaca
- Dodavanje kružnog ključa
- Zamjena bajtova
- Pomicanje redova
- Dodavanje kružnog ključa

3.3. Diffie-Hellmanova razmjena ključeva

S obzirom da je AES simetričan algoritam za kriptiranje poruka, potrebno je osigurati da primatelj i pošiljatelj imaju pristup jedinstvenom ključu. Iz sigurnosnih razloga taj ključ ne bi se smio slati preko interneta kako ga netko tko prisluškuje mrežu ne bi pribavio. Problem razmjene tajnog ključa bez slanja konkretnog ključa riješili su Whitfield Diffie i Martin Hellman. Oni su osmislili sustav koji omogućava da dvije osobe koje žele komunicirati s kriptiranim porukama ne moraju razmijeniti konkretan ključ kojim se poruka kriptira. Koristeći njihovu metodu potrebno je razmijeniti određene komponente iz kojih se može izračunati zajednički ključ, ali samo uz komponente koje ostaju tajne [4]. Ovakva razmjena funkcionira na slijedeći način:

- Prva strana generira prosti broj p
- Prva strana generira bazu g
- Prva strana odabire svoj tajni broj x
- Prva strana računa $g^x \bmod p$
- Prva strana drugoj strani šalje p, g i $g^x \bmod p$
- Druga strana odabire svoj tajni broj y
- Druga strana računa $g^y \bmod p$
- Druga strana šalje prvoj $g^y \bmod p$
- Druga strana računa $(g^x \bmod p)^y \bmod p$ što je jednako $g^{xy} \bmod p$
- Prva strana računa $(g^y \bmod p)^x \bmod p$ što je jednako $g^{xy} \bmod p$

Može se primijetiti da obje strane komunikacije na kraju procesa imaju izračunatu vrijednost ekvivalentnu $g^{xy} \bmod p$ bez da prva strana zna tajni broj y i bez da druga strana zna tajni broj x [5]. Vrijednost $g^{xy} \bmod p$ sada se može koristiti kao ključ za enkripciju podataka npr. Algoritmom AES. Navedeni postupak također je skiciran na Slici 1.



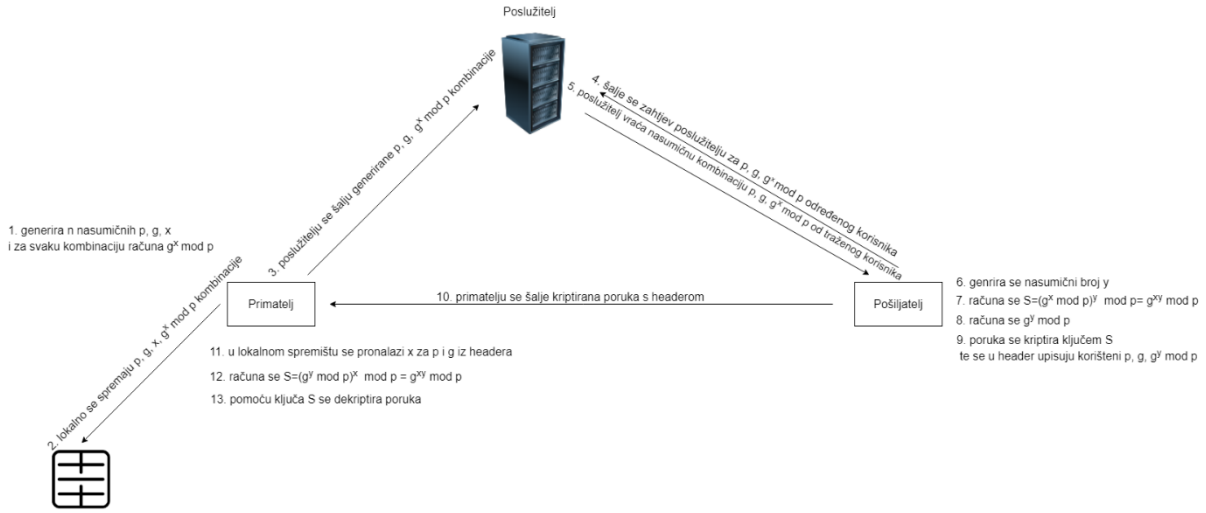
Slika 1: Princip rada Diffie-Hellmanove razmjene ključeva, autorski rad

4. Implementacija sigurnosne nadogradnje sustava elektroničke pošte

U ovom poglavlju prikazan je praktični dio rada, odnosno kako je implementiran sustav koji bi trebao omogućiti razmjenu kriptiranih poruka s budućom sigurnosti.

4.1. Skica funkcionalnosti

Nakon teoretske analize potrebnih dijelova, bilo je potrebno napraviti skicu na koji način će se koristiti Diffie-Hellmanova razmjena ključeva za poboljšanje sigurnosti elektroničke pošte. Početna teoretska skica funkcionalnost prikazana je na slici 2 .

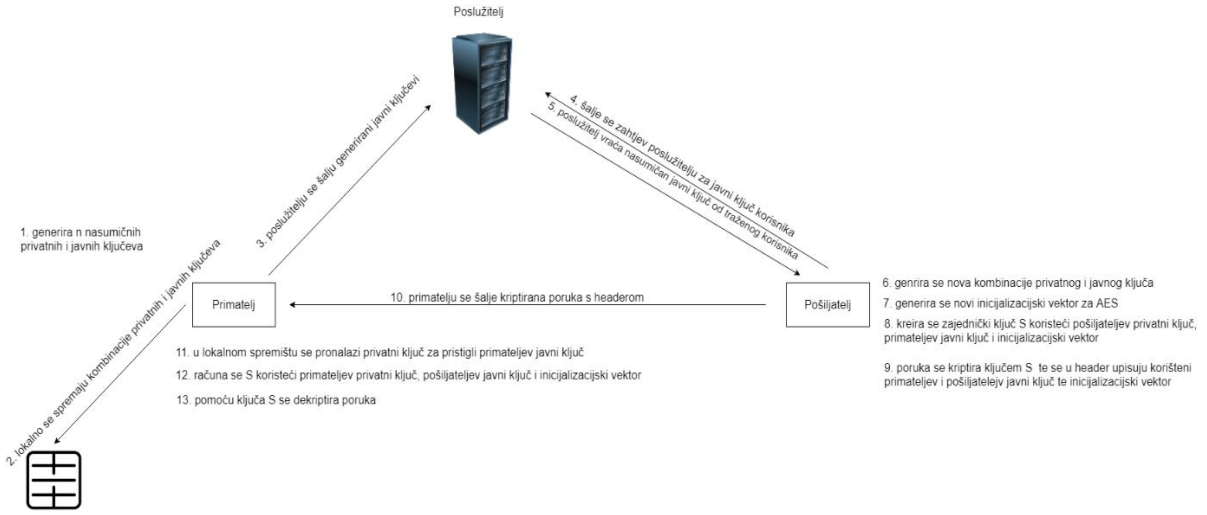


Slika 2: Početna skica funkcionalnosti sustava za sigurnosnu nadogradnju elektroničke pošte, autorski rad

Način na koji je sustav zamišljen prikazan je u točkama 1-13. Primatelj poruke prilikom registracije u sustav generira *p*, *g* i *x* vrijednosti te računa $g^x \bmod p$ koji se spremaju lokalno i zatim se poslužitelju sustava šalju samo *p*, *g* i $g^x \bmod p$ vrijednosti, dok *x* ostaje tajan i spremljen samo u lokalnoj bazi podataka primatelja poruka. Kada pošiljatelj registriranom primatelju želi poslati kriptiranu poruku mora poslužitelju sustava za sigurnosnu nadogradnju elektroničke pošte poslati zahtjev za *p*, *g* i $g^x \bmod p$ željenog primatelja. Poslužitelj na taj zahtjev odgovara na način da pošiljatelju vrati nasumično odabranu kombinaciju *p*, *g* i $g^x \bmod p$ od svih

kombinacija koje ima spremljene za traženog primatelja poruke. Pošiljatelj zatim generira nasumični broj *y* te računa ključ *S* po formuli $(g^x \bmod p)^y \bmod p$ što je jednako vrijednosti koja se dobije računanjem $g^{xy} \bmod p$, a bez da pošiljatelj zna broj *x*. Nakon toga pošiljatelj računa $g^y \bmod p$, kriptira poruku ključem *S* koristeći algoritam AES i u zaglavlje poruke sprema *p*, *g*, $g^y \bmod p$ te ju šalje primatelju. Prilikom čitanja poruke sustav primatelja u lokalnoj bazi podataka traži *x* vezan uz pristigle *p* i *g* i zatim računa ključ *S* po formuli $(g^y \bmod p)^x \bmod p$ što opet poprima vrijednost jednaku $g^{xy} \bmod p$, ali ovaj put bez da primatelj zna nasumični broj *y*. Pomoću izračunatog ključa S primatelj koristeći AES dekripciju može pročitati sadržaj primljene elektroničke pošte.

Tijekom implementacije, zbog korištenih određenih gotovih rješenja, pokazalo se kako sustav mora raditi malo drugačije od originalne skice funkcionalnosti te tako nastaje skica funkcionalnosti koja više odgovara konačnom rješenju i prikazana je na slici 3.



Slika 3: Prepravljena skica funkcionalnosti sustava za sigurnosnu nadogradnju elektroničke pošte, autorski rad

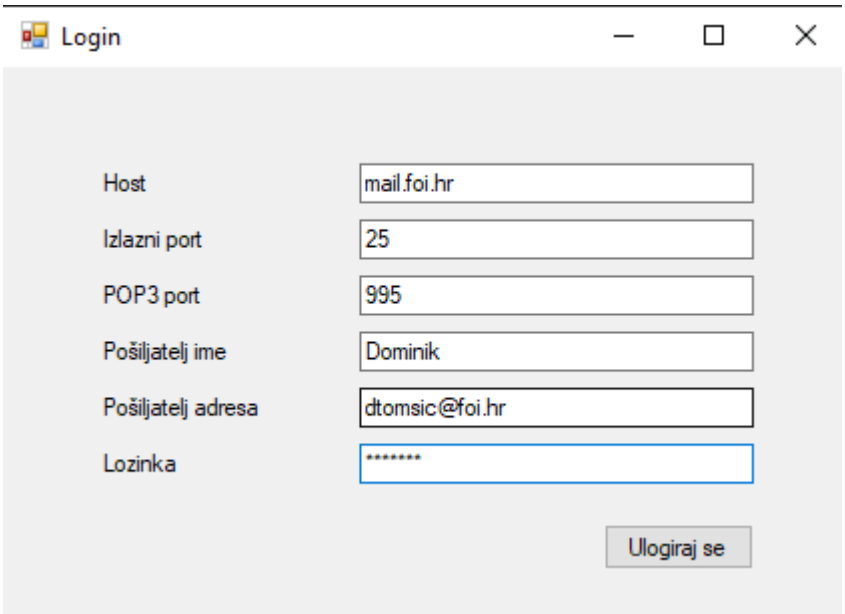
Koristeći gotovu klasu „ECDiffieHellmanCng“ koja je dio „System.Security.Cryptography“ dolazi do potrebe za malo drugačijom implementacijom od početne skice funkcionalnosti. Prema ovoj skici funkcionalnosti može se vidjeti kako se ovdje koristeći navedenu klasu, generiraju parovi privatnih i javnih ključeva, gdje javni ključ kao jedna vrijednost unutar sebe sadrži sve pošiljatelju potrebne informacije za stvaranje zajedničkog ključa uz vlastiti privatni ključ. Također može se primijetiti kako se ovdje pojavljuje inicijalizacijski vektor koji je potreban kod AES enkripcije te je i njega bitno proslijediti primatelju.

4.2. Implementacija klijenta e-pošte

Kako bi se implementirao sustav za enkripciju elektroničke pošte koji koristi Diffie-Hellmanovu razmjenu ključeva, prvo je napravljen jednostavan klijent za e-poštu koji omogućava čitanje i slanje elektroničke pošte. Napravljeni klijent e-pošte sastoji se od tri Windows Forms obrasca, obrazac za prijavu, obrazac koji omogućava pregled pristiglih poruka s informacijom o pošiljatelju i naslovom poruke, također omogućava slanje nove poruke i zadnji obrazac koji prikazuje odabranu pristiglu poruku u cijelosti.

4.2.1. Klijent e-pošte grafički

Kao što je ranije napisano, klijent e-pošte sastoji se od tri obrasca. Pokretanjem aplikacije otvara se obrazac „Login“ u koji korisnik upisuje potrebne podatke vezane uz svoj poslužitelj e-pošte te adresu e-pošte i lozinku kako bi se prijavio u svoj korisnički račun. U primjeru na slici 4 prikazan je spomenuti obrazac.

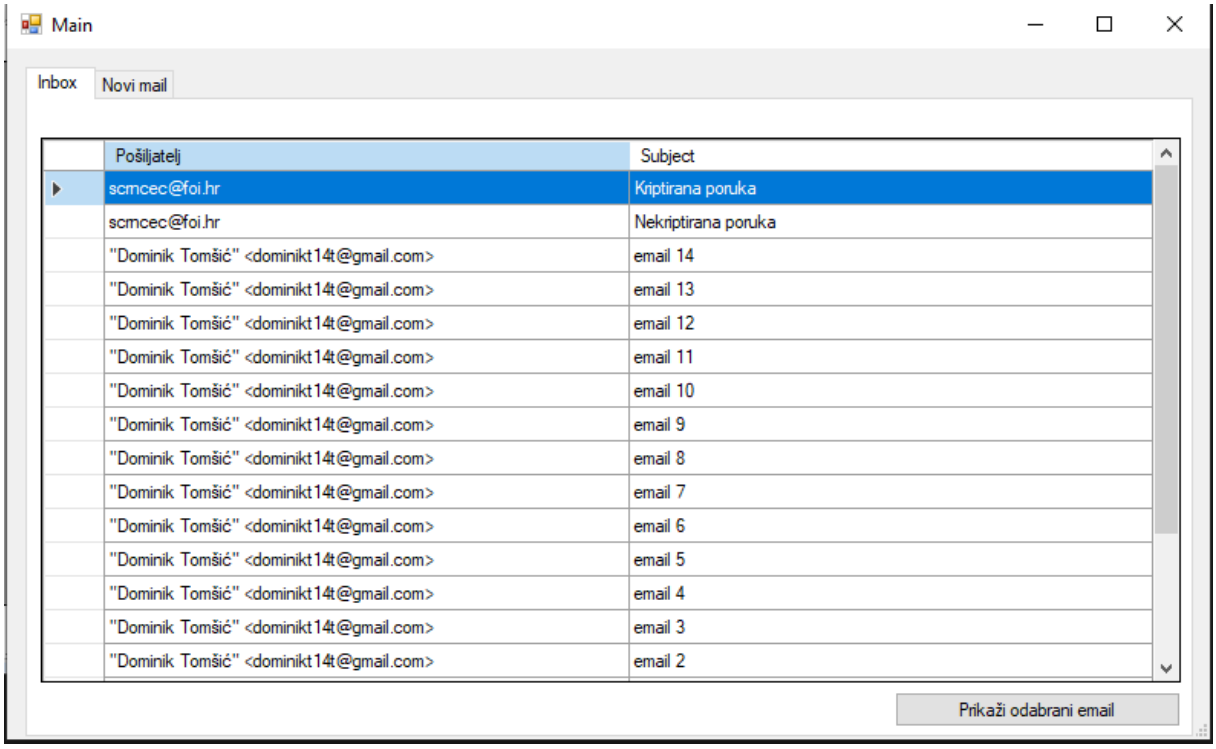


Slika 4: Obrazac „Login“, autorski rad

Kako bi se korisnik uspješno prijavio u svoj korisnički račun, potrebni podaci su adresa poslužitelja e-pošte, SMTP port za izlaznu poštu, POP3 port za ulaznu poštu, ime korisnika, adresa e-pošte korisnika i valjana lozinka za taj račun. Nakon što korisnik unese sve potrebne

podatke, klikom na gumb „Ulogiraj se“ pokreće se proces autentifikacije kod poslužitelja e-pošte.

Nakon što se korisnik uspješno prijavi u sustav prikazuje mu se slijedeći obrazac, pod nazivom „Main“, na kojem se nalaze izlistane pristigle poruke, počevši od najnovije, s podatkom o pošiljatelju i naslovu poruke. Ovaj obrazac može se vidjeti na slici 5.



Slika 5: Obrazac „Main“ s prikazanim ulaznim pretincem, autorski rad

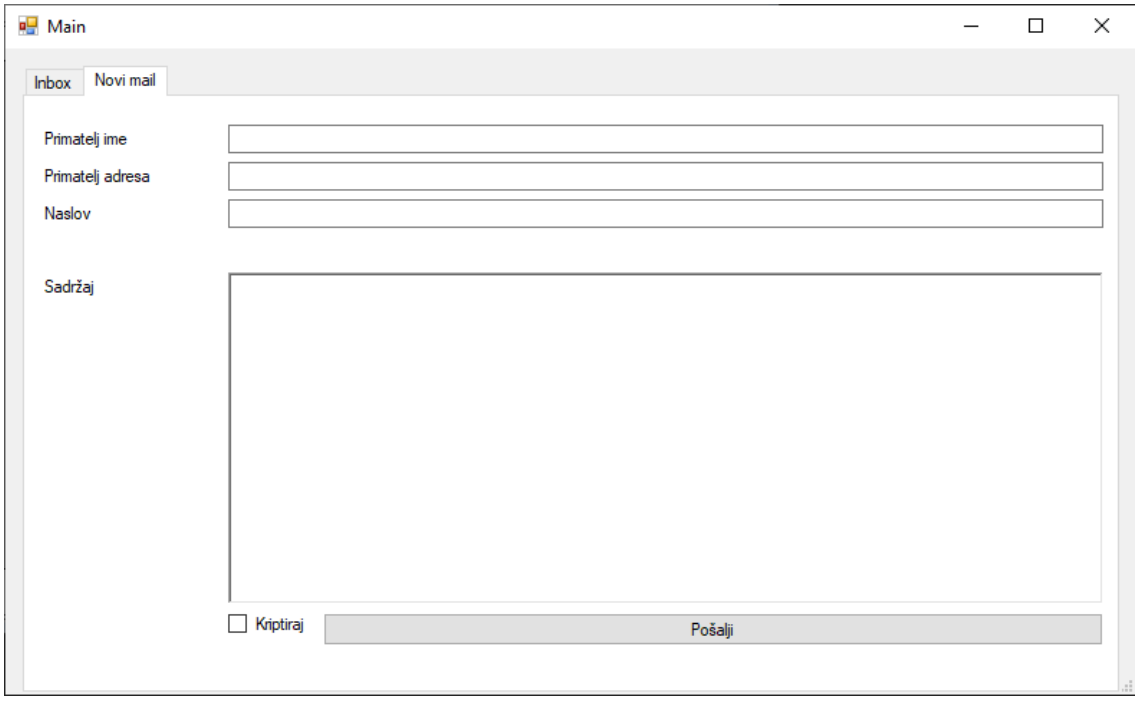
Korisnik može kliknuti na željenu poruku kako bi ju odabrao te nakon toga kliknuti na gumb „Prikaži odabrani email“ čime se otvara novi obrazac „ShowMail“ u kojem je prikazana cijela željena poruka, što se može vidjeti na slici 6. Za odabranu poruku prikazuju se informacije o pošiljatelju, naslov poruke i sadržaj poruke.



Slika 6: Obrazac „ShowMail“, autorski rad

Dok je obrazac „ShowMail“ otvoren, obrazac „Main“ ostaje otvoren u pozadini, ali on je zablokiran te se daljnja interakcija na tom obrascu može odvijati tek nakon što se obrazac „ShowMail“ zatvori.

Na obrascu „Main“ klikom na „Novi mail“ otvara se drugi prikaz tog obrasca koji korisnicima omogućava sastavljanje nove poruke što je prikazano na slici 7.



Slika 7: Obrazac „Main“ s prikazanim pogledom za kreiranje nove poruke, autorski rad

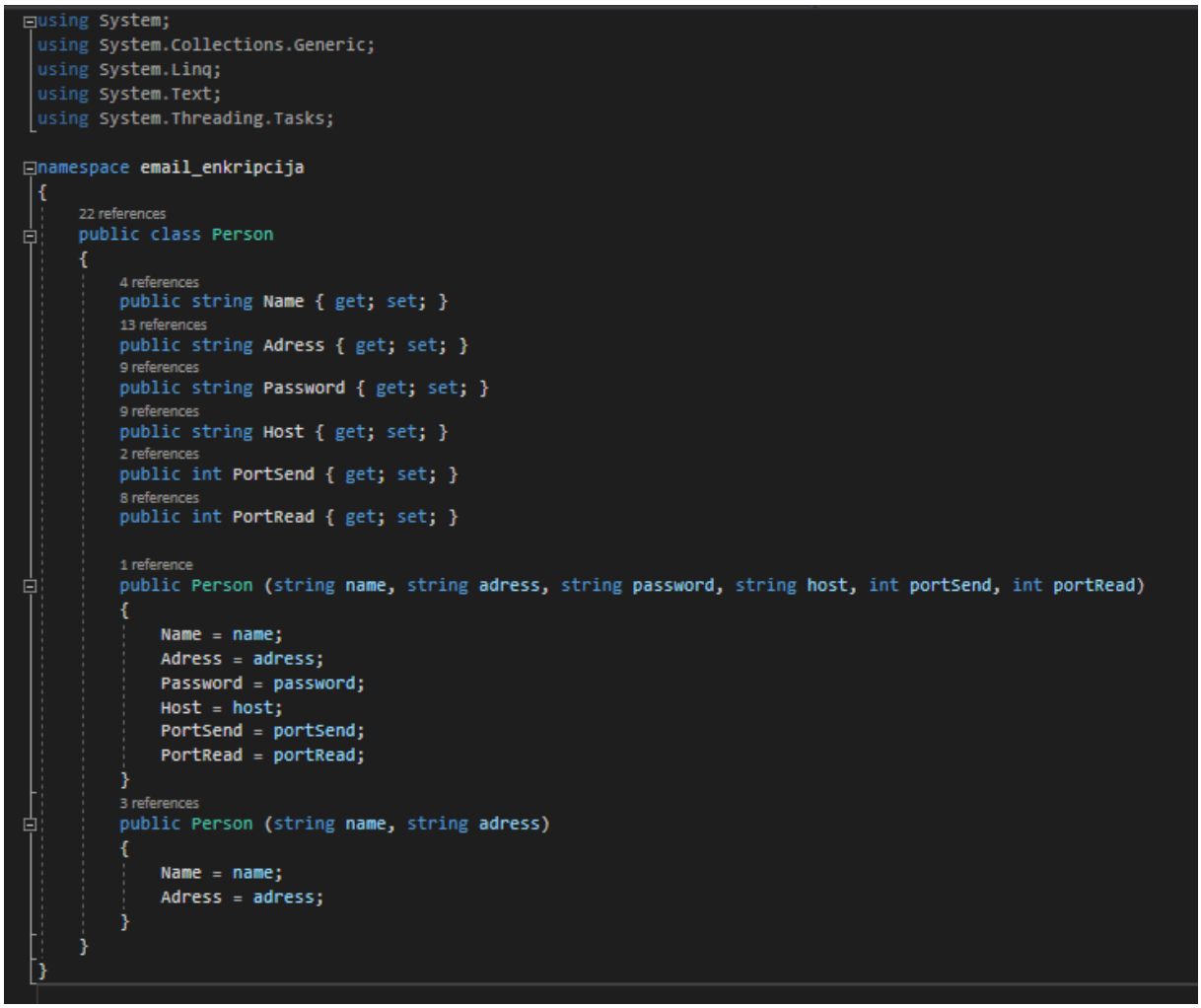
Za slanje nove poruke potrebno je ispuniti sva polja: „Primatelj ime“, „Primatelj adresa“, „Naslov“ i „Sadržaj“. Također ovdje se može vidjeti checkbox „Kriptiraj“ čijim označavanjem se napisana poruka šalje u kriptiranom obliku, no detaljnije o tome napisano je kasnije u tekstu.

4.2.2. Pojašnjenje programskog koda klijenta e-pošte

Za izgradnju klijenta e-pošte korišten je programski jezik C# te je kao razvojno okruženje odabran Visual Studio 2019. Ranije prikazani obrasci vizualno su uređeni koristeći dizajnerski pogled u razvojnom okruženju koji omogućava dodavanje i uređivanje određenih unaprijed pripremljenih elemenata. Što se tiče programskog koda za spajanje na poslužitelj e-pošte, čitanje pristiglih i slanje novih poruka korišten je NuGet paket OpenPop.NET. U implementaciji napravljene su dvije klase: klasa „ReadMail“ omogućava sve potrebne radnje vezane uz čitanje poruka, a klasa „SendMail“ omogućava sve potrebne radnje vezane uz slanje poruka. Za izgradnju ovih dviju klasa znatno su pomogli primjeri s izvora [6]. Također je napravljena još jedna klasa nazvana „Person“ koja sadrži potrebne podatke o pošiljateljima i primateljima poruka.

4.2.2.1. Klasa „Person“

Ova klasa vrlo je jednostavna, sastoji se od 6 svojstava i 2 konstruktora koji se koriste u dvije različite svrhe, konstruktor sa 6 ulaznih parametara stvara objekt koji u radu ovog klijenta sadrži informacije o prijavljenom korisniku određenog poslužitelja e-pošte, a konstruktor s 2 ulazna parametra sadrži podatke o primatelju određene poruke. Svrha ove klase je smanjiti broj potrebnih varijabli koje se prosljeđuju određenim drugim klasama tako što se dalje može prenositi samo jedan objekt tipa „Person“. Implementacija ove klase prikazana je na slici 8.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace email_enkripcija
{
    22 references
    public class Person
    {
        4 references
        public string Name { get; set; }
        13 references
        public string Adress { get; set; }
        9 references
        public string Password { get; set; }
        9 references
        public string Host { get; set; }
        2 references
        public int PortSend { get; set; }
        8 references
        public int PortRead { get; set; }

        1 reference
        public Person (string name, string adress, string password, string host, int portSend, int portRead)
        {
            Name = name;
            Adress = adress;
            Password = password;
            Host = host;
            PortSend = portSend;
            PortRead = portRead;
        }
        3 references
        public Person (string name, string adress)
        {
            Name = name;
            Adress = adress;
        }
    }
}
```

Slika 8: Klasa „Person“, autorski rad

4.2.2.2. Klasa „ReadMail“

Klasa „ReadMail“ sadrži funkcije potrebne za prikaz pristigle pošte. Ova klasa sastoji se od 4 metode:

- „GetNumberOFMessages“
- „FetchAllHeaders“
- „FetchMailByIndex“
- „DeleteMessageOnServer“.

Metoda „GetNumberOFMessages“ za prijavljenog korisnika od poslužitelja e-pošte saznaje informaciju o ukupnom broju poruka u ulaznom pretincu. Ova metoda prikazana je na slici 9.

```
public int GetNumberOFMessages(string hostname, int port, bool useSsl, string username, string password)
{
    int count=0;
    using (Pop3Client client = new Pop3Client())
    {
        client.Connect(hostname, port, useSsl);
        client.Authenticate(username, password);
        count = client.GetMessageCount();
    }

        return count;
}
```

Slika 9: Metoda „GetNumberOFMessages“, autorski rad

Metoda „FetchAllHeaders“ od poslužitelja e-pošte pribavlja zaglavlja svih poruka u korisnikovom ulaznom pretincu te se na osnovi tih informacija na obrascu „Main“ prikazuju informacije o pošiljateljima i naslovima svih poruka. Implementacija ove metode vidljiva je na slici 10.

```
public List<MessageHeader>  FetchAllHeaders(string hostname, int port, bool useSsl, string username, string password)
{
    using (Pop3Client client = new Pop3Client())
    {
        client.Connect(hostname, port, useSsl);
        client.Authenticate(username, password);
        int messageCount = client.GetMessageCount();
        List<MessageHeader> allMessages = new List<MessageHeader>(messageCount);
        for (int i = messageCount; i > 0; i--)
        {
            MessageHeader headers = client.GetMessageHeaders(i);
            allMessages.Add(headers);
        }
        return allMessages;



    }
}
```

Slika 10: Metoda „FetchAllHeaders“, autorski rad

„FetchMailByIndex“ je metoda koja na osnovi rednog broja određene poruke od poslužitelja e-pošte zahtijeva cijelu poruku kako bi se u obrascu „ShowMail“ osim pošiljatelja i naslova poruke mogao prikazati i sadržaj poruke. Implementacija ove metode prikazana je na slici 11.

```
public Message FetchMailByIndex(int index, string hostname, int port, bool useSsl, string username, string password)
{
    using (Pop3Client client = new Pop3Client())
    {
        client.Connect(hostname, port, useSsl);
        client.Authenticate(username, password);
        int messageCount = client.GetMessageCount();
        Message Read = client.GetMessage(index);
        return Read;
    }
}
```

Slika 11: Metoda „FetchMailByIndex“, autorski rad

Zadnja metoda u ovoj klasi je „DeleteMessageOnServer“ te ona na osnovi rednog broja poruke briše poruku na poslužitelju, što je također važna funkcija za cijeli sustav te u daljnjem tekstu piše više o tome. Ova metoda prikazana je na slici 12.

```
public static void DeleteMessageOnServer(string hostname, int port, bool useSsl, string username, string password, int messageNumber)
{
    using (Pop3Client client = new Pop3Client())
    {
        client.Connect(hostname, port, useSsl);
        client.Authenticate(username, password);
        client.DeleteMessage(messageNumber);
    }
}
```

Slika 12: Metoda „DeleteMessageOnServer“, autorski rad

4.2.2.3. Klasa „SendMail“

Klasa „SendMail“ je klasa koja ovom klijentu e-pošte omogućava slanje poruka. Ova klasa sadrži 6 svojstava: Person Sender, Person Reciever, string Subject, string Body, bool SSL i bool Crypto te ima jedan konstruktor koji prima po jedan parametar za svako od navedenih svojstava što je prikazano na slikama 13 i 14.

```
class SendMail
{
    8 references
    public Person Sender { get; set; }
    6 references
    public Person Reciever { get; set; }
    2 references
    public string Subject { get; set; }
    5 references
    public string Body { get; set; }
    2 references
    public bool SSL { get; set; }
    2 references
    public bool Crypto { get; set; }
```

Slika 13: Svojstva klase „SendMail“, autorski rad

```
public SendMail(Person sender, Person reciever, string subject, string body, bool ssl, bool crypto)
{
    Sender = sender;
    Reciever = reciever;
    Subject = subject;
    Body = body;
    SSL = ssl;
    Crypto = crypto;
}
```

Slika 14: Konstruktor klase „SendMail“

Nadalje, ova klasa sadrži jednu metodu pod nazivom „Send“. Navedena metoda prvo provjerava je li svojstvo Crypto za konkretan objekt vrijednosti 1 ili 0 te ako je 1 instancira objekt klase Crypto, koja je kasnije detaljno objašnjena, te kriptira poruku koju je potrebno poslati. Ukoliko poruka u svojstvu Crypto ima 0, tada se ranije navedeni korak preskače i nastavlja se sa slanjem poruke koristeći System.Net.Mail.SmtpClient i metodu Send. Na kraju se korisniku prikazuje poruka o uspješno poslanoj elektroničkoj pošti, no ta poruka se ne prikazuje u slučaju slanja pošte na adresu „root@dodocrypts.com“ zbog određenih automatskih poruka potrebnih za funkcioniranje sustava za kriptiranje, također objašnjenih u daljnjem dijelu rada. Metoda „Send“ klase „SendMail“ prikazana je na slici 15.

```
public bool Send()
{
    if (Crypto)
    {

        Crypto crypto = new Crypto(Body);
        Body = crypto.Encrypt( Sender, Reciever);
        if (Body == "")
        {
            MessageBox.Show("Primatelj nije registriran u Crypto sustav!");
            return false;
        }

    }
    MailAddress from_address = new MailAddress(Sender.Adress, Sender.Name);
    MailAddress to_address = new MailAddress(Reciever.Adress, Reciever.Name);
    MailMessage message = new MailMessage(Sender.Adress, Reciever.Adress);
    message.Subject = Subject;
    message.Body = Body;
    System.Net.Mail.SmtpClient client = new System.Net.Mail.SmtpClient()
    {
        Host = Sender.Host,
        Port = Sender.PortSend,
        EnableSsl = SSL,
        DeliveryMethod = SmtpDeliveryMethod.Network,
        UseDefaultCredentials = false,
        Credentials = new NetworkCredential(from_address.Address, Sender.Password),
    };
    client.Send(message);
    if (Reciever.Adress != "root@dodocrypts.com")
    {
        MessageBox.Show("Poruka je poslana");
    }
    return true;
}
```

Slika 15: Metoda „Send“, autorski rad

4.3. Implementacija poslužitelja

Kao poslužitelj zakupljen je udaljeni poslužitelj kojem se može pristupiti pomoću programa „Putty“. Poslužitelj radi na operacijskom sustavu Linux Debian 10.4. Za potrebe instalacije poslužitelja e-pošte na udaljenom poslužitelj na vremenski period od jedne godine pribavljena je web adresa „dodocrypts.com“.

4.3.1. Instalacija poslužitelja e-pošte

Kako bi se poslužitelju mogle slati važne poruke vezane uz rad sustava za enkripciju poruka e-pošte i kako bi on mogao odgovarati na njih bilo je potrebno instalirati poslužitelja e-pošte na ranije navedeni Linux sustav. Za poslužitelja je odabran „Postfix Mail Server“ koji

se praćenjem koraka s izvora [7] vrlo jednostavno instalira na Linux sustav. Nakon što je Postfix postavljen on omogućava da svaki korisnik poslužitelja ima svoju adresu koja u ovom slučaju ima nastavak „@dodocrypts.com“. Korisnik koji se koristi kako bi se posložio poslužitelj potreban za ovaj projekt je „root“, odnosno glavni korisnik udaljenog poslužitelja te kao takav dobiva svoju odgovarajuću adresu koja glasi „root@dodocrypts.com“ te se koristi za komunikaciju s poslužiteljem od strane ranije opisanog klijenta e-pošte prilikom kriptiranja poruka.

Poslužitelj e-pošte instaliran na ovaj način još uvijek nema sve potrebne certifikate i stoga će poruke pristigle s adresa ovog poslužitelja e-pošte većina ostalih poslužitelja prikazati kao neželjenu poštu. S obzirom da instalacija poslužitelja e-pošte nije glavna tema ovog rada, ovako posložen poslužitelj zadovoljava potrebe onoga za što je namijenjen: postizanje funkcionalnosti razmjene javnih ključeva.

4.3.2. Program za automatsko odgovaranje na zahtjeve – autores

Kako bi poslužitelj mogao automatski čitati pristigle poruke i adekvatno reagirati na njih, bilo je potrebno napraviti program koji čita svu pristiglu poštu i ovisno o sadržaju poruke spremi pristigli javni ključ određenog korisnika ili na zahtjev šalje nasumični javni ključ određenog korisnika na adresu e-pošte pošiljatelja zahtjeva. Program koji to radi dobio je naziv „autores“. Program je napisan u programskom jeziku C++ koristeći Linuxov uređivač teksta nano.

Autores je napravljen kao jedna beskonačna petlja koja svoj krug započinje otvaranjem datoteke s nepročitanom pristiglom poštom, koja je u mbox formatu, koji je strukturiran tako da je najstarija poruka na početku datoteke. Zatim, ukoliko datoteka nije prazna, počinje čitanje datoteke red po red dok se iz nje ne pročitaju potrebne informacije o pošiljatelju pristigle poruke i sadržaju pristigle poruke.

S obzirom na sadržaj pristiglih poruka, pristigle poruke program tumači na tri različita načina:

- Poruke s pristiglim javnim ključem pošiljatelja
- Poruke koje zahtijevaju javni ključ određenog korisnika
- Sve ostale poruke

Poruke koje u sebi sadrže javni ključ pošiljatelja, u tijelu poruke započinju s oznakom „#reg#“ nakon koje slijedi javni ključ zapisan kao tekst u heksadekatskom obliku. Ukoliko

program nađe takvu poruku, on sprema ostatak tijela poruke, bez oznake „#reg#“, u mapu „keys“ u datoteku koja ima naziv jednak pošiljateljevoj adresi s nastavkom „.txt“. Na primjer, ukoliko bi takva poruka došla s adrese „mail@mail.com“, ključ bi bio spremljen u datoteku „mail@mail.com.txt“. Program novo pristigli ključ sprema na kraj datoteke ukoliko ona već postoji ili stvara novu datoteku s nazivom definiranim na ovdje opisan način.

Poruke koje su zahtjevi za javnim ključem određenog korisnika u svom sadržajnom dijelu započinju i završavaju oznakom „#!#“, a između tih oznaka nalazi se adresa e-pošte željenog primatelja kriptirane poruke, npr. „#!#mail@mail.com#!#“. Kada program prepozna ovakvu poruku on u mapi „keys“ traži datoteku koja ima naziv jednak sadržaju između oznaka i nastavak „.txt“, znači u ovom primjeru „mail@mail.com.txt“. Ukoliko program ne pronađe takvu datoteku kao sadržaj povratne poruke se određuje tekst "Trazeni korisnik nije registriran u crypto sustav", no ukoliko se pronađe datoteka s traženim nazivom, program nasumično odabire jedan od redova upisanih u toj datoteci te to odredi kao sadržaj odgovora na zahtjev.

Ukoliko nepročitana poruka ne sadrži niti jednu od ranije navedenih oznaka, program ne poduzima nikakve specifične akcije.

Nakon što program „pročita“ najstariju nepročitanu poruku, ukoliko je poruka bila registracija novog javnog ključa ne šalje nikakav odgovor. Ukoliko je poruka bila zahtjev za javnim ključem određenog korisnika, tada se pošiljatelju zahtjeva šalje povratna poruka sa sadržajem "Trazeni korisnik nije registriran u crypto sustav" ako traženi korisnik nema spremljenih javnih ključeva ili sadržajem koji odgovara jednom javnom ključu traženog korisnika ako je korisnik registriran u sustav. Za slanje ovakve e-poruke koristi se naredba „system()“ koja omogućava C++ programu slanje naredbi komandnoj liniji Linux operacijskog sustava te se u ovom slučaju šalje naredba koja šalje željenu poruku.

Na kraju ciklusa ove beskonačne petlje, ukoliko je na početku ustanovljeno kako je postojala nepročitana poruka, opet se koristi naredba „system()“, koja ovoga puta kao argument dobiva naredbu, kojom operacijski sustav iz mbox datoteke s nepročitanim porukama najstariju nepročitanu poruku premješta u mbox datoteku s pročitanim porukama.

U nastavku se nalazi cijeli programski kod programa „autores“:

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime>
#include <random>

using namespace std;
```

```
int main() {
    while (true) {
      string sendTo = "";
      string Msg = "";
      ifstream mail;
      mail.open("/var/mail/root");
      string line;
      std::string delimiter = "<";
      bool youGotMail = false;

      if (mail.is_open()) {
        bool from = false;
        bool foundKey = false;
        int brojiRedove = 0;
        while (getline(mail, line)) {
          if (!from) {
            if (line.find("Return-Path") != string::npos) {
              youGotMail = true;
              size_t pos = line.find(delimiter);
              line.erase(0, pos + delimiter.length());
              sendTo = line.substr(0, line.find(delimiter));
              sendTo = sendTo.substr(0, sendTo.size() - 1);
              from = true;
            }
          } else {
            string del = "#!#";
            string reg = "#reg#";
            if (line.find(del) != string::npos) {
              size_t pos = line.find(del);
              line.erase(0, pos + del.length());
              string searchFor = line.substr(0, line.find(del));
              searchFor = "keys/" + searchFor + ".txt";
              std::ifstream infile(searchFor);

              if (infile.good()) {
                int count = -1;
                if (infile.is_open()) {
                  while (getline(infile, line)) {
                    count++;
                  }
```

```
                infile.close();
                std::ifstream pick(searchFor);
                if (pick.is_open()) {
                  srand(time(NULL));
                  count = rand() % (count + 1);
                  for (int i = 0; i <= count; i++) {
                    getline(pick, line);
                    Msg = line;
                  }
                } else Msg = "Greska!";
              } else Msg = "Greska!";
            } else {
              Msg = "Trazeni korisnik nije registriran u crypto
sustav";
            }

            break;
          }
          if (line.find(del) != string::npos) {
            size_t pos = line.find(del);
            line.erase(0, pos + del.length());
            string searchFor = line.substr(0, line.find(del));
            searchFor = "keys/" + searchFor + ".txt";
            searchFor = "keys/" + searchFor + ".txt";
            std::ifstream infile(searchFor);

            if (infile.good()) {
              int count = -1;
              if (infile.is_open()) {
                while (getline(infile, line)) {
                  count++;
                }
                infile.close();
                std::ifstream pick(searchFor);
                if (pick.is_open()) {
                  srand(time(NULL));
                  count = rand() % (count + 1);
                  for (int i = 0; i <= count; i++) {
                    getline(pick, line);
                    Msg = line;
                  }
```

```
                } else Msg = "Greska!";
              } else Msg = "Greska!";
            } else {
              Msg = "Trazeni korisnik nije registriran u crypto
sustav";
            }
            break;
            else if ((line.find(reg) != string::npos || foundKey) &&
brojiRedove < 3) {
              foundKey = false;
              string Write = "";
              if (line.find("=")) {
                brojiRedove++;
                foundKey = true;
              }
              if (line.find(reg) != string::npos) {
                size_t pos = line.find(reg);
                line.erase(0, pos + reg.length());
                Write = line.substr(0, line.find(reg));
              } else {
                Write = line;
              }
              if (brojiRedove < 3) {
                Write = Write.substr(0, Write.length() - 1);
              } else {
                Write = Write + "\n";
              }
              string searchFor = "keys/" + sendTo + ".txt";
              Msg = "";
              std::ifstream infile(searchFor);
              if (infile.good()) {
                infile.close();
                std::ofstream outfile;
                outfile.open(searchFor, std::ios_base::app);
                outfile << Write;
                outfile.close();
              } else {
                std::ofstream outfile(searchFor);
                outfile << Write;
                outfile.close();
              }
```

```
            } else if (line.find("Return-Path") != string::npos) {
              break;
            }
          }
        }
      }
      mail.close();
      if (sendTo != "" && Msg != "") {
          string response = "echo \"" + Msg + "\" | mail -s \"AUTO\" "
+ sendTo;
          system(response.c_str());
      }
      if (youGotMail){
                system("echo p | mail");
          }
    }
    return 0;
 }
```

Za pokretanje ovog programa u terminalu Linux poslužitelja koristi se „nohup“ funkcija kako bi program radio u pozadini, odnosno kako se gašenjem daljinske veze do poslužitelja njegov rad ne bi prekinuo. Konkretna naredba za pokretanje „autores“ programa izgleda ovako: „nohup "./autores" &“.

4.4. Implementacija kriptiranja i dekriptiranja poruka

Nakon što su pripremljeni klijent e-pošte koji omogućava čitanje i slanje poruka i poslužitelj koji omogućava spremanje i distribuciju javnih ključeva potrebnih za enkripiciju poruka, slijedeći korak bio je implementirati kriptiranje i dekriptiranje poruka. Za potrebe ovog dijela ranije opisanom klijentu, dodana je nova klasa pod nazivom „Crypto“ koja ima jedno svojstvo: „Poruka“ i 5 metoda: „Encrypt“, „Decrypt“, „Register“, „EncryptStringToBytes_Aes“ i „DecryptStringFromBytes_Aes“. U ovom poglavlju detaljno je objašnjen način rada i primjene ovih funkcija.

4.4.1. Stvaranje i spremanje para privatnog i javnog ključa primatelja

Prilikom svake prijave u ranije opisani klijent e-pošte poziva se metoda „Register“ novo stvorenog objekta tipa „Crypto“, koja kao argument prima objekt tipa „Person“. Ova klasa koristeći klasu „ECDiffieHellmanCng“ uz potrebne parametre stvara par privatnog i javnog

ključa u obliku niza bajtova. Ta dva niza bajtova pretvaraju se u heksadekatske vrijednosti koje se pamte kao stringovi te se zajedno s adresom e-pošte prijavljenog korisnika spremaju u lokalnu datoteku „LocalKeys.txt“ i javni ključ se također šalje i ranije opisanom poslužitelju koji ga sprema u svoju bazu podataka.

U datoteci „LocalKeys.txt“ podaci se spremaju redom: adresa e-pošte korisnika, znakovni niz „!#!“ koji označava početak slijedeće stavke, javni ključ u heksadekatskom obliku pa ponovno znakovni niz „!#!“ i na kraju privatni ključ u heksadekatskom obliku. Primjer jednog takvog zapisa izgleda ovako:

„dtomsic@foi.hr!#!45434b3120000000b286d219d8c6a220fe1466d07152d9048b7ae58642fb8630eff270c8410dc9be17b2122b3dd1d020eeebd95a698b12f80b5ee705469c2153c5ba752812de0a62!#!45434b3220000000b286d219d8c6a220fe1466d07152d9048b7ae58642fb8630eff270c8410dc9be17b2122b3dd1d020eeebd95a698b12f80b5ee705469c2153c5ba752812de0a6219e337a70d1a191577ce085299dc7c1db22d12c97d85d4596ef4213559fe8140“.

Kao što je navedeno, javni ključ se također šalje poslužitelju ovog sustava na adresu „root@dodocrypts.com“ u sadržajnom dijelu poruke koji ima slijedeći format: oznaka „#reg#“ kako bi program „autores“ na poslužitelju prepoznao da se radi o novom javnom ključu i nakon te oznake dolazi javni ključ zapisan u heksadekatskom obliku, za gore prikazan lokalno spremljen zapis, sadržaj poruke koja bi se slala na adresu „root@dodocrypts.com“ izgleda ovako:

„#reg#45434b3120000000b286d219d8c6a220fe1466d07152d9048b7ae58642fb8630eff270c8410dc9be17b2122b3dd1d020eeebd95a698b12f80b5ee705469c2153c5ba752812de0a62“.

Cijela klasa „Register“ prikazana je na slici 16.

```
1 reference
public bool Register(Person Sender)
{
    byte[] bobPublicKey;
    byte[] bobPrivateKey;

    string keystring;

    using (ECDiffieHellmanCng cng = new ECDiffieHellmanCng(
        CngKey.Create(
            CngAlgorithm.ECDiffieHellmanP256,
            null,
            new CngKeyCreationParameters
    { ExportPolicy = CngExportPolicies.AllowPlaintextExport })))
    {
        cng.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;
        cng.HashAlgorithm = CngAlgorithm.Sha256;
        bobPrivateKey = cng.Key.Export(CngKeyBlobFormat.EccPrivateBlob);
        bobPublicKey = cng.PublicKey.ToByteArray();

    }

    StringBuilder hexPublic = new StringBuilder(bobPublicKey.Length * 2);
    foreach (byte b in bobPublicKey)
    hexPublic.AppendFormat("{0:x2}", b);
    keystring = hexPublic.ToString();

    StringBuilder exPrivate = new StringBuilder(bobPrivateKey.Length * 2);
    foreach (byte b in bobPrivateKey)
    exPrivate.AppendFormat("{0:x2}", b);
    string privatekeystring = exPrivate.ToString();

    string body = "#reg#" + keystring;
    Person Dodocrypts = new Person("root", "root@dodocrypts.com");

    SendMail posalji = new SendMail(Sender, Dodocrypts, "register", body, true, false);
    posalji.Send();

    using (StreamWriter writer = new StreamWriter("LocalKeys.txt", true))
    {
        writer.WriteLine(Sender.Adress+"!#!"+keystring + "!#!" + privatekeystring);
    }
    return true;
}
```

Slika 16: Metoda „Register“, autorski rad

4.4.2. Enkripcija poruke

Klasa „Crypto“ ima dva konstruktora koji su prikazani na slici 17. Jedan od njih ne prima argumente te se koristi za stvaranje objekta kojem je svrha pokretanje „Register“ metode, dok se konstruktor koji prima argument tipa string koristi kod kreiranja objekta kojem je svrha pokretanje metoda „Encrypt“ ili „Decrypt.

```
1 reference
public Crypto()
{

}

2 references
public Crypto(string poruka)
{
    Poruka = poruka;
}
```

Slika 17: konstruktori klase „Crypto“, autorski rad

Metoda „Encrypt“ prilikom svojeg poziva prima 2 objekta tipa „Person“, a to su „Sender“ i „Reciever“ koji sadrže podatke o pošiljatelju, odnosno primatelju željene kriptirane poruke. Ova metoda na početku, poput metode „Register“, kreira novi par javnog i privatnog ključa te javni ključ dodatno pretvara i u heksadekatski oblik koji pamti kao string. Slijedeća stvar koju ova metoda radi jest slanje e-pošte s adrese pošiljatelja prema poslužitelju te je sadržaj te poruke formatiran na ovaj način: na početku i na kraju sadržaja nalaze se oznake „#!#“, a među njima zapisana je adresa željenog primatelja kriptirane poruke, npr.: „#!#mail@mail.com#!#“. Dosad objašnjen dio metode prikazan je na slici 18.

```
1 reference
public string Encrypt(Person Sender, Person Reciever)
{
    byte[] alicePublicKey;
    byte[] alicePrivateKey;
    byte[] bobPublicKey;
    byte[] aliceKey;

    string alicekeystring;
    string IVstring;

    using (ECDiffieHellmanCng cng = new ECDiffieHellmanCng(
        CngKey.Create(
            CngAlgorithm.ECDiffieHellmanP256,
            null,
            new CngKeyCreationParameters
            { ExportPolicy = CngExportPolicies.AllowPlaintextExport })))
    {
        cng.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;
        cng.HashAlgorithm = CngAlgorithm.Sha512;
        alicePublicKey = cng.PublicKey.ToByteArray();
        alicePrivateKey = cng.Key.Export(CngKeyBlobFormat.EccPrivateBlob);

    }

    StringBuilder hexPublic = new StringBuilder(alicePublicKey.Length * 2);
    foreach (byte b in alicePublicKey)
        hexPublic.AppendFormat("{0:x2}", b);
    alicekeystring = hexPublic.ToString();

    ReadMail reader = new ReadMail();
    string keystring = "";
    string user = Sender.Adress;
    int index = user.IndexOf("@");
    if (index > 0)
        user = user.Substring(0, index);
    int before = reader.GetNumberOfMessages(Sender.Host, Sender.PortRead, true, user, Sender.Password);
    string body = "#!#" + Reciever.Adress + "#!#";
    Person Dodocrypts = new Person("root", "root@dodocrypts.com");
    SendMail posalji = new SendMail(Sender, Dodocrypts, "request keys", body, true, false);
    posalji.Send();
```

Slika 18: Metoda „Encrypt“ prvi dio, autorski rad

Na slici 19 prikazan je dio ove metode koji čeka e-poštu kao odgovor na poslani zahtjev poslužitelju. Poslužitelju se daje 30 sekundi vremena da odgovori povratnom porukom, koja sadrži nasumičan javni ključ željenog primatelja kriptirane poruke ili informaciju o tome da željeni primatelj nije registriran u sustav. Ukoliko se u poruci nalazi ključ, on se za sada sprema kao string, a ako korisnik nije registriran, metoda vraća prazan string nakon što obriše dobivenu poruku kako ona ne bi ostala korisniku sustava u dolaznom pretincu elektroničke pošte.

```
int after = before;
int timer = 0;
while (after == before && timer < 30)
{
    System.Threading.Thread.Sleep(1000);
    timer++;
    after = reader.GetNumberOfMessages(Sender.Host, Sender.PortRead, true, user, Sender.Password);
    if (after > before)
    {
        OpenPop.Mime.Message procitano = reader.FetchMailByIndex(after, Sender.Host, Sender.PortRead, true, user, Sender.Password);
        if (procitano.Headers.From.ToString().Contains("root@mail.dodocrypts.com"))
        {
            OpenPop.Mime.MessagePart sadrzaj = procitano.FindFirstPlainTextVersion();
            if (sadrzaj != null)
            {
                keystring = sadrzaj.GetBodyAsText();
                if (keystring.Contains("Trazeni korisnik"))
                {
                    return "";
                }
                ReadMail.DeleteMessageOnServer(Sender.Host, Sender.PortRead, true, user, Sender.Password, after);
            }
        }
        else
        {
            before++;
        }
    }
}
```

Slika 19: Metoda „Encrypt“ drugi dio, autorski rad

U zadnjem dijelu ove metode, prikazanom na slici 20, prvo se dobiveni ključ iz heksadekatskog oblika spremljenog tekstualno, pretvara u niz bajtova kako bi se zajedno s ranije novo stvorenim privatnim ključem pošiljatelj mogao koristiti za sastavljanje onoga što je cijeli bit ovog sustava, ključa kojim se šifrira poruka. Nakon toga se uz pomoć klase „Aes“ kreira novi inicijalizacijski vektor potreban metodi „EncryptStringToBytes_Aes“, prikazanoj na slici 21 i preuzetoj s izvora [8]. Ova metoda kao argumente prima poruku koju treba kriptirati u tekstualnom obliku, ranije kreirani ključ kao niz bajtova i spomenuti inicijalizacijski vektor te vraća kriptiranu poruku u obliku niza bitova. Taj niz bitova se tada pretvara u heksadekatski oblik i sprema kao string. Na kraju ove metode stvara se string koji sadrži redom: oznaku „#CRYPTO#“ kako bi klijent e-pošte prilikom čitanja poruke znao da je kriptirana, zatim niz znakova „#!#“ kao razdjelnik pa korišteni javni ključ primatelja u heksadekatskom obliku, kako bi se mogao pronaći spareni privatni ključ prilikom dešifriranja pa ponovno razdjelnik „#!#“ nakon kojeg slijede javni ključ pošiljatelja, inicijalizacijski vektor i šifrirana poruka, svi u heksadekatskom obliku i razdvojeni znakovnim nizovima „#!#“. Jedan takav string u konačnici izgleda ovako:

„#CRYPTO#45434b31200000000994e4300181b027f044ff07bbd8847ea710b303e350aa6a8ec280e6d5136595632b76614f64b0f3b6342c1f7f3c2581f52b186900bc06919a86344f3cc43e53#!#45434b312000000019ef2927d64f5c485362f10923ec217fadd60c3d14ee1c9fbf2141ff734f22911b4a40caefa3bdceea0bf0063f080e45c57f05bf18cb0b70e0b59d1451dd9968#!#72076d8d54578d2de7eba687e1f40193#!#cecbb1360eb078fbcd299093afb0313ece5170c892c3a590418b421ce022badb5d751fb413394f25b59ef2a093086578bdd54b3400d20ced7c8cc85ca0754c56a6aec146f56b739ed4df94e7fb7391ebcef29c8ad74dae6cafcfb9e902f2be0ed3ef0569a167a3aa023e15b0cbfe783a“

Na kraju se ovakav string vraća kao rezultat poziva metode „Encrypt“.

```
    bobPublicKey = Enumerable.Range(0, keystring.Length)
             .Where(x => x % 2 == 0)
             .Select(x => Convert.ToByte(keystring.Substring(x, 2), 16))
             .ToArray();

    using (ECDiffieHellmanCng cng = new ECDiffieHellmanCng(CngKey.Import(alicePrivateKey, CngKeyBlobFormat.EccPrivateBlob)))
    {
        cng.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;
        cng.HashAlgorithm = CngAlgorithm.Sha256;
        aliceKey = cng.DeriveKeyMaterial(CngKey.Import(bobPublicKey, CngKeyBlobFormat.EccPublicBlob));
    }

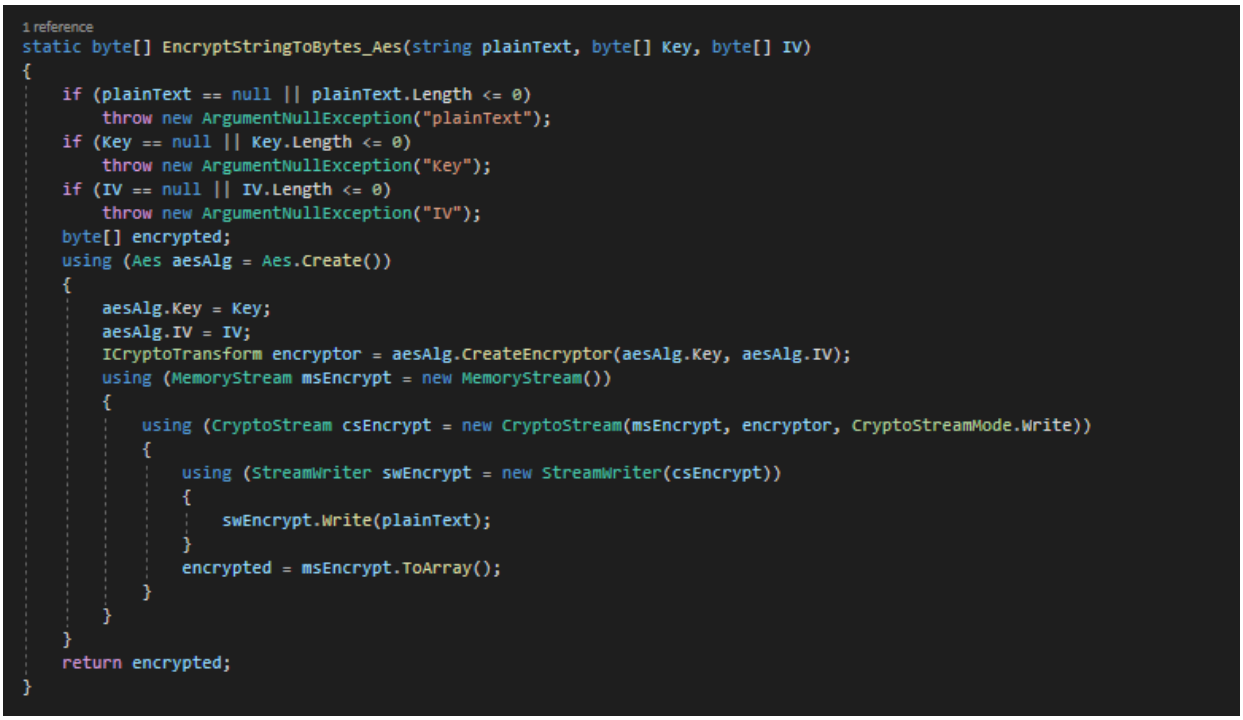
    using (Aes myAes = Aes.Create())
    {
        StringBuilder hexIV = new StringBuilder(myAes.IV.Length * 2);
        foreach (byte b in myAes.IV)
            hexIV.AppendFormat("{0:x2}", b);
        IVstring = hexIV.ToString();

        byte[] encrypted = EncryptStringToBytes_Aes(Poruka, aliceKey, myAes.IV);

        StringBuilder hexPoruka = new StringBuilder(encrypted.Length * 2);
        foreach (byte b in encrypted)
            hexPoruka.AppendFormat("{0:x2}", b);
        Poruka = hexPoruka.ToString();

    }
    Poruka = "#CRYPTO#" + keystring + "#!#" + alicekeystring + "#!#" + IVstring + "#!#" + Poruka;
    return Poruka;
}
```

Slika 20: Metoda „Encrypt“ treći dio, autorski rad



```
1 reference
static byte[] EncryptStringToBytes_Aes(string plainText, byte[] Key, byte[] IV)
{
    if (plainText == null || plainText.Length <= 0)
        throw new ArgumentNullException("plainText");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Key");
    if (IV == null || IV.Length <= 0)
        throw new ArgumentNullException("IV");
    byte[] encrypted;
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = Key;
        aesAlg.IV = IV;
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
        using (MemoryStream msEncrypt = new MemoryStream())
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {
                    swEncrypt.Write(plainText);
                }
                encrypted = msEncrypt.ToArray();
            }
        }
    }
    return encrypted;
}
```

Slika 21: Metoda „EncryptStringToBytes_Aes“, Prema izvoru [8]

4.4.3. Dekripcija poruke

Kada klijent e-pošte, opisan ranije u radu, prilikom otvaranja pristigle poruke u njenom sadržaju pronađe niz znakova „#CRYPTO#“, tada se pretpostavlja da poruka strukturom odgovara ranije opisanoj strukturi za slanje kriptiranih poruka i kreira se novi objekt klase „Crypto“, kojem u konstruktoru kao argument daje sadržaj poruke i nad tim objektom poziva metoda „Decrypt“ kojoj je argument objekt tipa Person i sadrži podatke o prijavljenom korisniku. Ova metoda najprije pristiglu poruku dijeli po separatoru „#!#“ te njene dijelove stavlja u polje stringova. Nakon toga se u lokalno spremljenoj datoteci „LocalKeys.txt“ provjerava postojanje zapisa s pristiglim javnim ključem primatelja i adresom primatelja. Ukoliko se takav zapis ne pronađe, kao rezultat ove metode vraća se string „Poruka je enkriptirana koristeći CRYPTO, no ključ koji je korišten nije spremljen na ovom računalu!“. Ukoliko je odgovarajući zapis pronađen, iz njega se uzima pripadajući privatni ključ primatelja. Nakon toga se pronađeni privatni ključ primatelja, pristigli javni ključ pošiljatelja i pristigli inicijalizacijski vektori, iz tekstualno spremljenog heksadekatskog oblika, pretvaraju u niz bajtova. Slijedeća stvar je ponovno kreiranje ključa kojim je pristigla poruka šifrirana te se on na ranije opisan način kreira iz javnog ključa pošiljatelja i privatnog ključa primatelja, a on svojom vrijednošću odgovara ključu kreiranom korištenjem privatnog ključa pošiljatelja i javnog ključa primatelja. Pozivom metode „DecryptStringFromBytes_Aes“, preuzete s izvora [8] i prikazane na slici 22, kojoj se kao argumenti daju kriptirani sadržaj poruke, izračunati ključ i

pristigli inicijalizacijski vektor, kao rezultat se dobiva dekriptirana poruka koja se vraća kao rezultat metode i prikazuje korisniku. Metoda „Decrypt“ prikazana je na slici 23.



```
1 reference
static string DecryptStringFromBytes_Aes(byte[] cipherText, byte[] Key, byte[] IV)
{

    if (cipherText == null || cipherText.Length <= 0)
        throw new ArgumentNullException("cipherText");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Key");
    if (IV == null || IV.Length <= 0)
        throw new ArgumentNullException("IV");
    string plaintext = null;
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = Key;
        aesAlg.IV = IV;
        ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
        using (MemoryStream msDecrypt = new MemoryStream(cipherText))
        {
            using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
            {
                using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                {
                    plaintext = srDecrypt.ReadToEnd();
                }
            }
        }
    }
    return plaintext;
}
```

Slika 22: Metoda „DecryptStringFromBytes_Aes“, Prema izvoru [8]

```
1 reference
public string Decrypt(Person Reciever)
{
    Poruka = Poruka.Substring(8);
    string[] separatingStrings = { "#!#","!#!" };
    string[] keysAndContent = Poruka.Split(separatingStrings, System.StringSplitOptions.RemoveEmptyEntries);
    byte[] alicePublicKey;
    byte[] IV;
    byte[] Content;
    byte[] bobPrivateKey = null;
    byte[] bobKey;

    string line = "";
    StreamReader file = new StreamReader("LocalKeys.txt");
    bool found = false;
    while ((line = file.ReadLine()) != null && !found)
    {
        string[] LocalKeys = line.Split(separatingStrings, System.StringSplitOptions.RemoveEmptyEntries);

        if(LocalKeys[0] == Reciever.Adress && LocalKeys[1] == keysAndContent[0])
        {
            found = true;
            bobPrivateKey = Enumerable.Range(0, LocalKeys[2].Length)
             .Where(x => x % 2 == 0)
             .Select(x => Convert.ToByte(LocalKeys[2].Substring(x, 2), 16))
             .ToArray();
        }
    }
    file.Close();
    if (found)
    {
        alicePublicKey = Enumerable.Range(0, keysAndContent[1].Length)
             .Where(x => x % 2 == 0)
             .Select(x => Convert.ToByte(keysAndContent[1].Substring(x, 2), 16))
             .ToArray();

        IV = Enumerable.Range(0, keysAndContent[2].Length)
             .Where(x => x % 2 == 0)
             .Select(x => Convert.ToByte(keysAndContent[2].Substring(x, 2), 16))
             .ToArray();

        Content = Enumerable.Range(0, keysAndContent[3].Length)
             .Where(x => x % 2 == 0)
             .Select(x => Convert.ToByte(keysAndContent[3].Substring(x, 2), 16))
             .ToArray();

        using (ECDiffieHellmanCng cng = new ECDiffieHellmanCng(CngKey.Import(bobPrivateKey, CngKeyBlobFormat.EccPrivateBlob)))
        {
            cng.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;
            cng.HashAlgorithm = CngAlgorithm.Sha256;

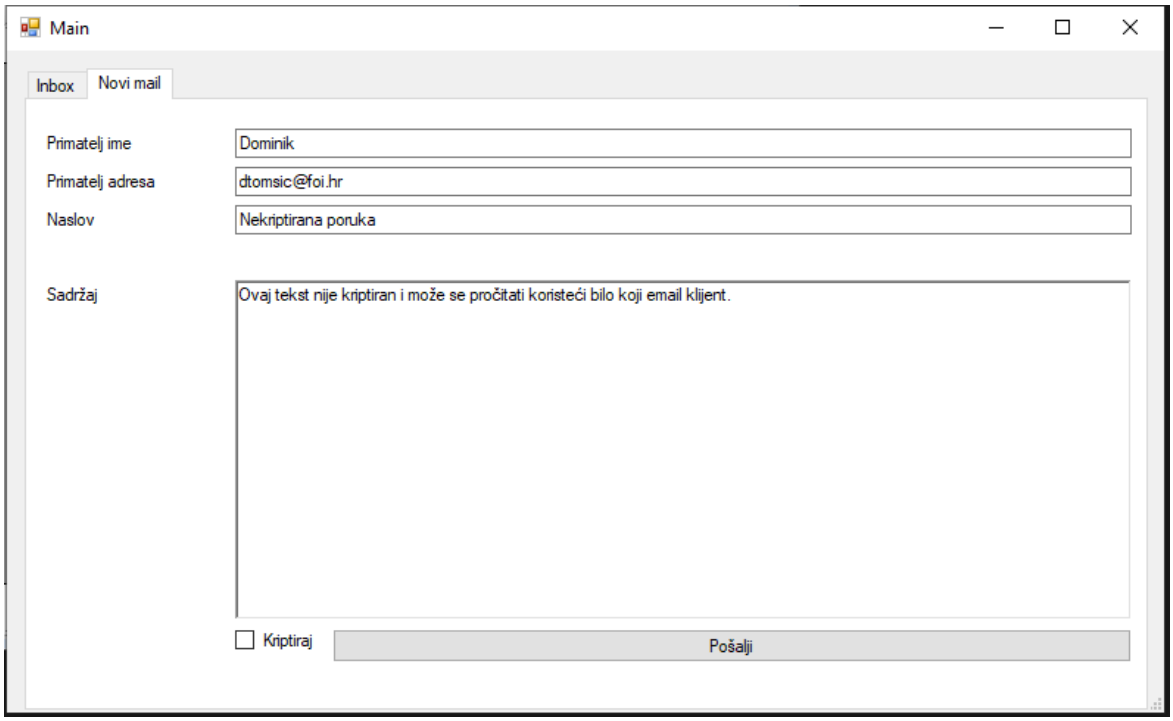
            bobKey = cng.DeriveKeyMaterial(CngKey.Import(alicePublicKey, CngKeyBlobFormat.EccPublicBlob));
        }
        Poruka = DecryptStringFromBytes_Aes(Content, bobKey, IV);
    }
    else
    {
        Poruka = "Poruka je enkriptirana koristeći CRYPTO, no ključ koji je korišten nije spremljen na ovom računalu!";
    }
    return Poruka;
}
```

Slika 23: Metoda „Decrypt“, autorski rad

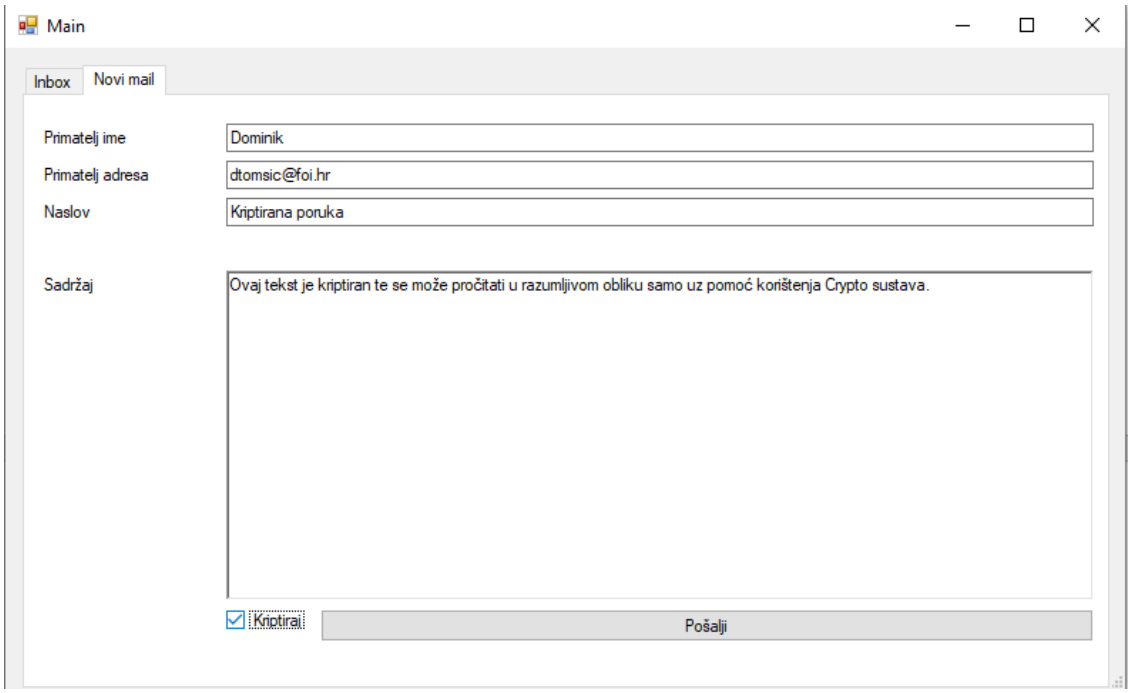
4.5. Prikaz rada sustava

U nastavku je prikazano kako se koristeći ovaj sustav za poboljšanje sigurnosti elektroničke pošte mogu poslati ne kriptirane i kriptirane poruke, kako pristigle poruke izgledaju u klijentu e-pošte s implementiranim sustavom i klijentu bez njega.

Na slici 24 je prikazano slanje ne kriptirane poruke primatelju „dtomsic@foi.hr“ i može se vidjeti kako checkbox „Kriptiraj“ nije označen, dok je na slici 25 prikazano slanje kriptirane poruke istom primatelju te je checkbox „Kriptiraj“ označen.

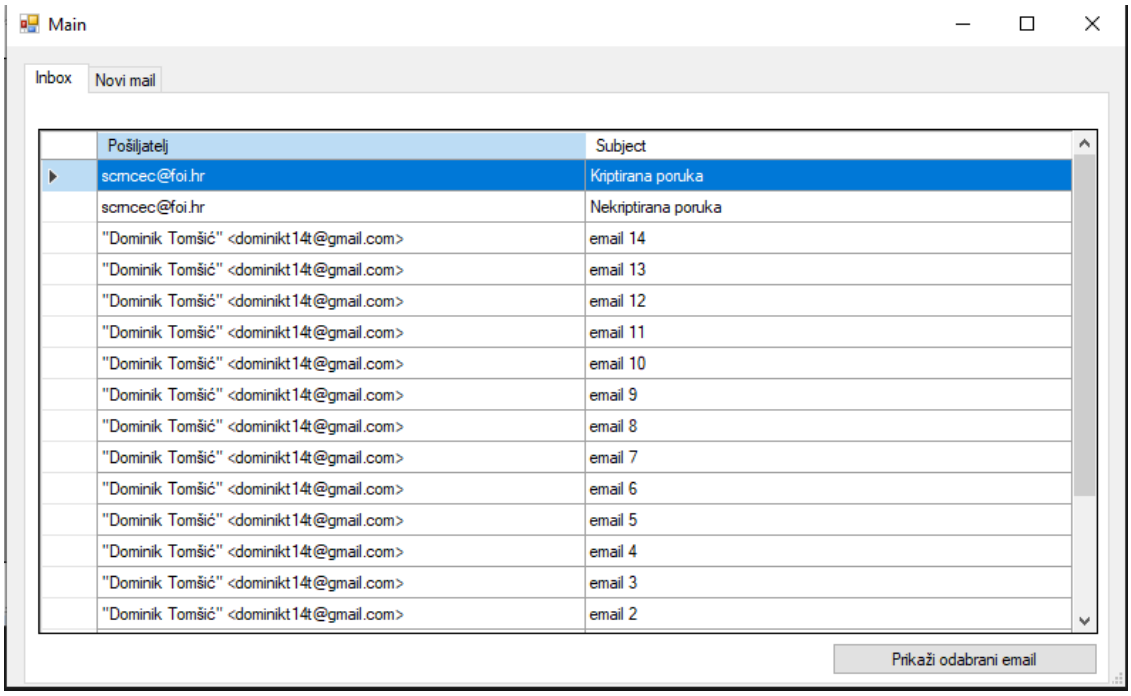


Slika 24: Slanje ne kriptirane poruke, autorski rad



Slika 25: Slanje kriptirane poruke, autorski rad

Kako se može vidjeti na slici 26, koja prikazuje ulazni pretinac korisnika „dtomsic@foi.hr“ učitan u klijentu napravljenom za potrebe ovog rada, obje poruke su pristigle i vidi se tko je pošiljatelj i koji je naslov.



Slika 26: Ulazni pretinac prikazan u kreiranom klijentu e-pošte, autorski rad

Na slikama 27 i 28 prikazane su poruke poslane na slikama 24 i 25 otvorene u klijentu e-pošte, koji ima implementiran sustav za sigurnosnu nadogradnju elektroničke pošte gdje se može vidjeti kako je primatelju u oba slučaja prikazan tekst kojeg je pošiljatelj napisao u sadržaje poruka.



Slika 27: Ne kriptirana poruka otvorena u vlastitom klijentu e-pošte, autorski rad



Slika 28: Kriptirana poruka otvorena u vlastitom klijentu e-pošte, autorski rad

Analizom ovih slika može se ustanoviti kako primatelj može razumljivo pročitati sadržaj obiju poslanih poruka, koristeći klijent koji ima implementiran sustav za sigurnosnu nadogradnju elektroničke pošte. Sada će biti prikazano kako te poruke izgledaju kada se prikazuju u klijentu e-pošte koji nema implementiran gore navedeni sustav te će za tu potrebu biti korišten „SquirrelMail“. Na slici 29 prikazan je ulazni pretinac gdje se opet mogu vidjeti pošiljatelj i naslov obiju poruka.



Slika 29: Ulazni pretinac otvoren u SquirrelMail klijentu, autorski rad

Na slici 30 prikazana je otvorena ne kriptirana poruka gdje se jasno može razumjeti njen sadržaj koji je jednak onome što je pošiljatelj napisao,



Slika 30: Ne kriptirana poruka otvorena u SquirrelMail klijentu, autorski rad

dok je na slici 31 prikazana otvorena kriptirana poruka čiji sadržaj nije jasan tekst, jednak onome što je pošiljatelj napisao, već niz znakova koji je formuliran na način opisan ranije u radu gdje se govori o tome kako se kriptira poruka u ovom sustavu.



Slika 31: Kriptirana poruka otvorena u SquirrelMail klijentu, autorski rad

Radi čitljivosti zbog veličine slike u nastavku je iz ovog klijenta e-pošte kopiran cijeli sadržaj poruke:
„#CRYPTO#45434b31200000000994e4300181b027f044ff07bbd8847ea710b303e350aa6a8ec280e6d5136595632b76614f64b0f3b6342c1f7f3c2581f52b186900bc06919a86344f3cc43e53#!#45434b312000000019ef2927d64f5c485362f10923ec217fadd60c3d14ee1c9fbf2141ff734f22911b4a40caefa3bdceea0bf0063f080e45c57f05bf18cb0b70e0b59d1451dd9968#!#72076d8d54578d2de7eba687e1f40193#!#cecbb1360eb078fbcd299093afb0313ece5170c892c3a590418b421ce022badb5d751fb413394f25b59ef2a093086578bdd54b3400d20ced7c8cc85ca0754c56a6aec146f56b739ed4df94e7fb7391ebcef29c8ad74dae6cafcfb9e902f2be0ed3ef0569a167a3aa023e15b0cbfe783a“

5. Zaključak

Tema ovog rada je sigurnosna nadogradnja sustava elektroničke pošte. Željena sigurnosna nadogradnja postiže se implementacijom sustava koji koristi Diffie-Hellmanovu metodu za razmjenu ključeva i algoritam AES za kriptiranje poruka. Sustav je implementiran kao dva zasebna programska rješenja. Prvo je jednostavan klijent e-pošte koji omogućava slanje kriptiranih i ne kriptiranih poruka te kreiranje ključeva potrebnih za to. Spomenuti klijent izrađen je u programskom jeziku C# korištenjem „Visual Studio 2019“ alata. Kako bi se spomenuta razmjena ključeva mogla odvijati, također je napravljen udaljeni poslužitelj s kojim ovaj klijent e-pošte komunicira putem elektroničke pošte. Svrha udaljenog poslužitelja je spremanje javnih ključeva kreiranih u klijentu e-pošte. Udaljeni poslužitelj pokretan je Linux Debian operacijskim sustavom i na njemu je instaliran „Postfix“ poslužitelj e-pošte. Na poslužitelju je konstantno pokrenut program izrađen u programskom jeziku C++. Taj program čita pristiglu elektroničku poštu s adresom poslužitelja. Pristigla pošta može biti javni ključ koji treba spremiti te program to i čini, a druga opcija je zahtjev za javnim ključem određenog korisnika te u tom slučaju poslužitelj odgovara na zahtjev elektroničkom poštom u kojoj se nalazi traženi javni ključ.

Opisani sustav uspješno je implementiran, no kako bi se takav sustav mogao komercijalno koristiti potrebno je još dorade, gdje bi kao najbitniji prvi korak naveo instalaciju potrebnih certifikata, kako se elektronička pošta pristigla od udaljenog poslužitelja ne bi smatrala neželjenom poštom.

Popis literature

[1] J. Rhoton, X.400 and Smtp: Battle of the E-Mail Protocols, Boston: Digital Press, 1997.

[2] H. Dobbertin, V. Rijmen i A. Sowa, Advanced Encryption Standard – AES, Berlin: Springer-Verlag , 2005.

[3] J. Lake, »What is AES encryption and how does it work?,« comparitech, 17. 2. 2020.. [Mrežno]. Available: https://www.comparitech.com/blog/information-security/what-is-aes-encryption/. [Pokušaj pristupa 29. 8. 2020.].

[4] C. P. Bauer, Secret History The Story of Cryptography, Boca Raton: Chapman and Hall/CRC, 2013.

[5] J. Lake, »What is the Diffie–Hellman key exchange and how does it work?,« comparitech, 15. 3. 2019.. [Mrežno]. Available: https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/. [Pokušaj pristupa 29. 8. 2020.].

[6] »OpenPop.Net,« [Mrežno]. Available: http://hpop.sourceforge.net/examples.php. [Pokušaj pristupa 1. 8. 2020.].

[7] J. Wallen, »Install and Configure a Postfix Mail Server,« Linux.com, 19. 5. 2010.. [Mrežno]. Available: https://www.linux.com/training-tutorials/install-and-configure-postfix-mail-server/. [Pokušaj pristupa 12. 8. 2020.].

[8] »Aes Class,« Microsoft, [Mrežno]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes?view=netcore-3.1. [Pokušaj pristupa 12. 8. 2020.].

Popis slika

Slika 1: Princip rada Diffie-Hellmanove razmjene ključeva, autorski rad 5
Slika 2: Početna skica funkcionalnosti sustava za sigurnosnu nadogradnju elektroničke pošte, autorski rad 6
Slika 3: Prepravljena skica funkcionalnosti sustava za sigurnosnu nadogradnju elektroničke pošte, autorski rad 7
Slika 4: Obrazac „Login“, autorski rad 8
Slika 5: Obrazac „Main“ s prikazanim ulaznim pretincem, autorski rad 9
Slika 6: Obrazac „ShowMail“, autorski rad 10
Slika 7: Obrazac „Main“ s prikazanim pogledom za kreiranje nove poruke, autorski rad 10
Slika 8: Klasa „Person“, autorski rad 12
Slika 9: Metoda „GetNumberOFMessages“, autorski rad 13
Slika 10: Metoda „FetchAllHeaders“, autorski rad 13
Slika 11: Metoda „FetchMailByIndex“, autorski rad 14
Slika 12: Metoda „DeleteMessageOnServer“, autorski rad 14
Slika 13: Svojstva klase „SendMail“, autorski rad 14
Slika 14: Konstruktor klase „SendMail“ 15
Slika 15: Metoda „Send“, autorski rad 16
Slika 16: Metoda „Register“, autorski rad 24
Slika 17: konstruktori klase „Crypto“, autorski rad 25
Slika 18: Metoda „Encrypt“ prvi dio, autorski rad 25
Slika 19: Metoda „Encrypt“ drugi dio, autorski rad 26
Slika 20: Metoda „Encrypt“ treći dio, autorski rad 27
Slika 21: Metoda „EncryptStringToBytes_Aes“, Prema izvoru [8] 28
Slika 22: Metoda „DecryptStringFromBytes_Aes“, Prema izvoru [8] 29
Slika 23: Metoda „Decrypt“, autorski rad 30
Slika 24: Slanje ne kriptirane poruke, autorski rad 31
Slika 25: Slanje kriptirane poruke, autorski rad 31
Slika 26: Ulazni pretinac prikazan u kreiranom klijentu e-pošte, autorski rad 32
Slika 27: Ne kriptirana poruka otvorena u vlastitom klijentu e-pošte, autorski rad 32
Slika 28: Kriptirana poruka otvorena u vlastitom klijentu e-pošte, autorski rad 33
Slika 29: Ulazni pretinac otvoren u SquirrelMail klijentu, autorski rad 33
Slika 30: Ne kriptirana poruka otvorena u SquirrelMail klijentu, autorski rad 34
Slika 31: Kriptirana poruka otvorena u SquirrelMail klijentu, autorski rad 34