# Elektronička pošta

**Maltar, Jurica**

**Undergraduate thesis / Završni rad**

**2015**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:126:400532

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

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



*Repository / Repozitorij:*

Repository of School of Applied Mathematics and Computer Science









Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

**Jurica Maltar**

# Elektronička pošta

Završni rad

Osijek, 2015.

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

**Jurica Maltar**

# Elektronička pošta

Završni rad

Voditelj: Izv. prof. dr.sc. Domagoj Matijević

Osijek, 2015.

**Sažetak**

Ovaj završni rad iskazuje cjelokupan sustav elektroničke pošte. Koristeći literaturu i RFC (*Request for Comments*) dokumente koji propisuju standarde za internetske entitete, opisana je elektronička poruka kao medij koji sadrži informaciju, protokoli koji služe za njezin prijenos i postupci za osiguravanje sigurnosti tijekom prijenosa. Elektronička je poruka sastavljena od niza ASCII znakova, a zahvaljujući MIME standardu, uz poruku je moguće priložiti i datoteku koja nije isključivo tekstualnog tipa. Elektronička poruka, nakon što je sastavljena, pomoću SMTP protokola putuje do željene destinacije, a pri dolasku primateljev mail klijent ju dohvaća koristeći neki od protokola za dohvaćanje poruka.
Praktični dio završnog rada sastoji se od implementacije programa koji šalje elektroničku poruku.

**Ključne riječi:** elektronička pošta, elektronička poruka, klijent, server, protokol, sigurnost

**Abstract**

This bachelor's thesis presents the complete electronic mail system. By using available literature and RFC (*Request for Comments*) documents which prescribe standards for Internet related entities, the electronic mail is described as a medium that contains an information, transfer protocols and safe-keeping steps during transfer. Electronic message consists of sequences of ASCII characters, and thanks to MIME protocol, it is possible to attach a non-textual file to the message. When completed, electronic message travels to a certain destination via SMTP protocol, and then mail client access it with mail access protocols.
The practical part contains the implementation of the program which sends an electronic message.

**Key words:** electronic mail, electronic message, client, server, protocol, security

# Sadržaj

**1 Uvod u elektroničku poštu** **1**
1.1 Općenito o mrežnim aplikacijama . . . . . 2

**2 Elektronička poruka** **3**
2.1 Oblik elektroničke poruke . . . . . 3
2.2 RFC 5322 . . . . . 3
2.2.1 Zaglavlje . . . . . 3
2.2.2 Tijelo . . . . . 4
2.3 MIME . . . . . 4

**3 SMTP** **7**
3.1 SMTP prijenos . . . . . 7
3.1.1 Inicijaliziranje sjednice . . . . . 8
3.1.2 Mail transakcija . . . . . 8
3.2 ESMTP . . . . . 9

**4 Mail Access Protocols** **11**
4.1 POP3 . . . . . 11
4.2 IMAP . . . . . 11

**5 Sigurnost** **14**
5.1 Simetrični i javni ključ . . . . . 14
5.2 Sigurnost elektroničke pošte . . . . . 14

**6 Implementacija e-mail klijenta u Python programskom jeziku** **16**
6.1 smtp_client.py . . . . . 17

**7 Zaključak** **20**

# 1 Uvod u elektroničku poštu

Elektronička pošta (eng. *electronic mail*, skraćeno *e-mail*) način je razmjene elektroničkih poruka putem Interneta ili ostalih računalnih mreža, a kao takav, postoji još od početka Interneta. Počiva na principima obične, ne-elektroničke pošte, a za razliku od obične pošte, prednosti su elektroničke pošte: brzina, lakoća prijenosa i cijena (za krajnjeg je korisnika usluga elektroničke pošte besplatna).
Ključne komponente sustava elektroničke pošte (u literaturi *e-mail system*) su:

- *klijenti* (ili *mail user agents*) - preciznije, klijenti krajnjih korisnika koji su pokrenuti na njihovim računalima, u ovom slučaju - primatelj i pošiljatelj elektroničke pošte
- *mail serveri* - serveri koji omogućavaju prijenos elektroničkih poruka, a zbog toga se nazivaju i *mail transfer agents*
- SMTP - protokol kojim se vrši prijenos poruke
- *Mail Access Protocols* - protokoli za dohvaćanje poruka putem primateljeva klijenta

Klijent u kontekstu elektroničke pošte omogućuje krajnjem korisniku da čita, odgovara, prosljeđuje, dohvaća, komponira i šalje poruku. Kada pošiljatelj, koristeći svoj klijent, komponira poruku i naredi klijentu da ju pošalje (komponiranje poruke i slanje od strane klijenta naziva se *e-mail submission*), klijent šalje poruku na mail server gdje se ona sprema u odlazeći red (eng. *outgoing message queue*[1]). Također, kada primatelj poruku želi pročitati, njegov klijent dohvaća poruku sa svog mail servera, gdje se poruka nalazi u *mailboxu*, spremniku u kojem će poruke nakon prijenosa biti pohranjene, koji upravlja porukama koje trebaju biti dostavljene primatelju.
Poruka koju pošiljatelj formira mora zadovoljavati određen oblik propisan s RFC 5322, dokumentom koji opisuje standard za formatiranje poruka. Kada je poruka formirana, ona se šalje putem SMTP protokola za slanje elektroničkih poruka. Kada svi koraci SMTP transakcije završe, ona se putem *mail access protocol-a* dohvaća od strane primateljeva klijenta.
Nekoć je korisnikovo računalo bilo u isto vrijeme mail klijent i mail server. Takvu su hijerarhiju omogućili jednostavni e-mail programi (*"mail"*, *"mailx"*...) kojima se pristupalo putem naredbenog retka (eng. *command line*). Primatelj je trebao biti *online* kako bi primio poruku. Takav je pristup nepraktičan jer korisnici, nakon obavljenog posla, uglavnom gase osobna računala. Zato su u sustav prijenosa elektroničke pošte integrirani *full-time* mail serveri koji su služili za *"queing"* i daljne slanje poruke. Pojavom naprednijih operacijskih sustava, pojavljuju se i napredniji mail klijenti kompatibilni sa grafičkim sučeljem tih operacijskih sustava. Nakon instalacije programa, korisnik je trebao konfigurirati svoj e-mail račun koji mu je dodjeljen od strane ISP-a (eng. *Internet Service Provider*), ili neke organizacije, npr. sveučilišta.
U današnje vrijeme, sve popularnija je *webmail* usluga. Svi procesi vezani uz e-mail odvijaju se unutar internetskog preglednika, koji je u slučaju webmaila klijent - korisnik pomoću preglednika vrši postupak potvrde identiteta, dohvaća, kreira, šalje poruku koja je prikazana unutar pretraživača kao web stranica. Protokol za dohvaćanje i slanje poruke do mail servera je HTTP. Ipak, mail serveri komuniciraju putem SMTP protokola. Popularni pružatelji webmail usluga su: Google (Gmail), Microsoft (Hotmail), Yahoo! (Yahoo! mail).

[1] Queue - apstraktni tip podatka koji služi za pohranu niza istovrsnih elemenata.

Poglavlje **Elektronička poruka** opisuje na koji način poruka treba biti sastavljena kako bi bila kompatibilna sa ostalim protokolima koji služe za njen prijenos, te kompatibilna za prikaz.
Poglavlje **SMTP** opisuje protokol kojim se poruka prenosi od pošiljateljava računala do mail servera, te od jednog mail servera do drugoga. U poglavlju je opisan tijek prijenosa poruke u potpunosti. Također, iskazani su postupci za sigurnost veze.
Poglavlje **Mail Access Protocols** opisuje dva najpopularnija protokola za dohvaćanje poruka - IMAP i POP3. Poruku koja se nalazi u primateljevom mailboxu, primateljev klijent pomoću nekog od navedenih protokola dohvaća. IMAP je daleko napredniji protokol s više mogućnosti, no i POP3 se koristi i danas jer je jednostavan.
Poglavlje **Sigurnost** obuhvaća mehanizme kojima se poruka osigurava na putu od pošiljatelja do primatelja, a ti mehanizmi su implementirani uz SMTP protokol.
Poglavlje **Implementacija e-mail klijenta u Python programskom jeziku** sadrži implementaciju SMTP klijenta koji šalje poruku. U poglavlju je opisan tok implementacije.

## 1.1 Općenito o mrežnim aplikacijama

Kako je rečeno, unutar sustava elektroničke pošte, ključni su protokoli za prijenos i dohvaćanje poruka i procesi koji komuniciraju putem tih protokola. Prije opisa protokola za prijenos i dohvaćanje poruka i načina na koji klijenti i serveri između njih komuniciraju, važno je objasniti što općenito takva hijerarhija predstavlja.
*Mrežna aplikacija* je aplikacija koja se izvršava na različitim *hostovima*[2] i takva aplikacija komunicira s drugim hostovima putem računalne mreže koristeći neki od protokola aplikacijskog sloja. Npr. e-mail klijent za slanje poruke koristi SMTP. Mrežna aplikacija sastoji se od procesa koji putem računalne mreže izmjenjuju poruke. Klijent je proces koji inicijalizira komunikaciju, a server proces koji čeka inicijaliziranje komunikacije kako bi započeo sjednicu (eng. *session*). *Aplikacijska arhitektura* ukazuje na koji je način aplikacija struktuirana unutar različitih hostova - to može biti P2P (*peer-to-peer*) ili *client-server* arhitektura (hrv. *klijentsko-poslužnička arhitektura*). U P2P arhitekturi hostovi komuniciraju bez posredstva *stalnih servera*, a host na kojem se izvršava P2P aplikacija može biti klijent i/ili server. U klijent-server arhitekturi, klijenti ne komuniciraju direktno, već posredstvom stalnih servera. Takvi su serveri uvijek aktivni i imaju stalnu IP adresu. IP adresa reprezentirana je pomoću četiri 8-bitna broja razdvojenih točkom i služi za identificiranje hosta kojemu je poslan podatak. Budući da jedan host može izvršavati više aplikacija, uz IP adresu traženog hosta, treba biti priložen i *port* - broj koji određuje kojoj je aplikaciji podatak koji šalje drugi host namjenjen.
Protokoli aplikacijskog sloja za prijenos podataka koriste transportne protokole[3] - protokole koji aplikacijskim procesima na različitim hostovima omogućavaju komunikaciju.

[2]Host (hrv. *domaćin*, u [1] nazivan i *end system*) je svaki uređaj povezan na računalnu mrežu.
[3]Sučelje koje povezuje aplikacijski i transportni sloj naziva se *kanal* (eng. *socket*).

# 2 Elektronička poruka

## 2.1 Oblik elektroničke poruke

E-mail poruka definirana je 1982. godine s RFC 822, definicija je nadograđena prvi put 2001. s RFC 2822, a trenutna je definicija nadograđena 2008. s RFC 5322. Taj dokument opisuje da je e-mail poruka ASCII kod, dijelovi od kojih je sastavljena su zaglavlja, prazne linije i tijela, a kraj reda označen je s *CRLF-om*. Dodatne mogućnosti poruke opisane su MIME standardom.

## 2.2 RFC 5322

Poruka je niz ASCII znakova. Podjeljena je u linije, pri čemu svaka linija završava sa *CRLF-om* (*carriage-return i line-feed*) - nizom dvaju ASCII znakova (ASCII 13 i ASCII 10).
Sastoji se od zaglavlja (eng. *header*) uz koje opcionalno slijedi, nakon prazne linije, tijelo (eng. *body*). Svaka linija poruke ne smije biti dulja od 998 znakova - razlog tomu je ograničenje u mnogim implementacijama koje rukuju s porukom u korist robusnosti same implementacije.

### 2.2.1 Zaglavlje

Polja zaglavlja su linije unutar poruke koje se sastoje od imena zaglavlja, zatim dvotočke, zatim tijela zaglavlja i zaključene s *CRLF-om*. Ime zaglavlje mora biti komponirano od ASCII znakova u rangu od 33 do 126, tijelo također, uključujući i WSP (*white space characters*) ASCII znakove 32 (*space*) i 9 (*horizontal tab*) . Zbog ograničenja duljine linije na 998 znakova, tijelo zaglavlja može biti "prelomljeno" u više redova - taj se postupak naziva *folding*. Neka zaglavlja bitna su za samog primatelja, a neka za komunikaciju između mail servera tijekom SMTP transakcije. Mail serverima dopušteno je na vrh poruke dodavati nova zaglavlja u svrhu praćenja poruke i pronalaženja grešaka među serverima. Neka od zaglavlja su:

- *From* - imenuje autora poruke
- *To* - imenuje primatelja/primatelje poruke
- *Reply-to* - određuje kome će odgovor na poruku biti dostavljen, ako ne originalnom autoru
- *Cc* (*Carbon copies*) - opcionalno, imenuje jednog ili više primatelja kojima će kopija poruke biti dostavljena, ali poruka nije upućena prvenstveno njima
- *Bcc* (*Blind carbon copies*) - opcionalno, imenuje jednog ili više primatelja koji dobivaju "tajnu" kopiju poruke; niti jedanom drugom primatelju neće biti vidljiv primatelj označen unutar Bcc zaglavlja
- *Subject* - imenuje tematiku ili sadržaj same poruke
- *Message-Id* - jedinstveni string za identifikaciju poruke
- *Received* - zaglavlje dodano od strane mail servera nakon svakog prijelaza tijekom prijenosa poruke

### 2.2.2 Tijelo

Tijelo je niz linija prikazanih pomoću ASCII koda. Dvije su restrikcije: *CR* i *LF* ne smiju se unutar tijela pojaviti nezavisno jedan o drugom i duljina linije ograničena je na 998 znakova. Tijelo sadrži informaciju koja je vidljiva primatelju poruke izravno (*plaintext*) ili kao multimedijalni sadržaj. Multimedijalne sadržaje unutar poruke omogućava MIME standard.

## 2.3 MIME

MIME (*Multipurpose Internet Mail Extensions*) je standard opisan nizom RFC dokumenata kojim se nadograđuje oblik poruke kako bi se omogućilo:

- Tekstualna poruka ne mora biti isključivo u ASCII formatu
- Prošireni skup različitih formata za netekstualna tijela poruka
- *multi-part* tijela poruka
- Tijelo zaglavlja ne mora biti isključivo u ASCII formatu

MIME poruke struktuirane su kao skup dijelova gdje jedan dio može imati više poddijelova, ali one mogu biti sastavljene i od jednog dijela poput poruka definiranih standardnim RFC 822 dokumentom. Pošiljateljev "MIME-aware" klijent konvertira sve moguće dijelove unutar poruke u 7-bitni ASCII kod kako bi poruka bila kompatibilna s protokolima za prijenos poruke. Nakon što se dostavi, poruka biva konvertirana u prvobitni *encoding* od strane primateljevog "MIME-aware" klijenta - "Content-Transfer-Encoding" klijentu govori na koji način poruka treba biti dekodirana.
RFC 2045 definira i opisuje:

- *MIME-Version* zaglavlje - iskazuje MIME verziju s kojom je poruka kompatibilna i ukazuje uređajima koji rukuju s porukom da ju razlikuju od ostalih poruka generiranih pomoću starijih standarda
- *Content-type* zaglavlje - određuje tip i podtip sadržaja ( *"media-type"* i *"media-subtype"*) koji se nalazi unutar tijela poruke, pri čemu je to tijelo pridruženo upravo *Content-type* zaglavlju, te određuje izvorni prikaz tog sadržaja. Podatak koji se pojavljuje nakon "Content-type:" unutar tog polja je oblika *type/subtype* (npr. "image/gif"). Nakon tog dijela pojavljuju se opcionalni parametri koji govore o dodatnim svojstvima sadržaja.
- *Content-transfer-encoding* zaglavlje - korišteno za specifikaciju enkodiranja tijela u druge podržane formate. Npr. enkodiranje se primjenjuje na određene podatke kako bi oni mogli proći kroz transportne mehanizme koji imaju restrikcije na određen tip podataka. Zadan ( *"default"*) format je ASCII, dok su ostali *quoted-printable*, *base64*, *8bit* i obični *binary*. Quoted-printable encoding koristi se za tekst koji je uglavnom 7-bit ASCII, ali u njemu se pojavljuje pokoji 8-bitni znak (npr. *ISO-8859-n* znak) koji će biti enkodiran u tri 7-bitna znaka. Base64 korišten je za čovjeku nečitljive sadržaje.
- Dva dodatna zaglavlja *Content-Id* i *Content-description*.

RFC 2046 opisuje opću strukturu MIME media typing sustava i definira inicijalni skup tipova sadržaja. "Media-type", tj. tip sadržaja deklarira generalni tip podatka, dok "Media-subtype", tj. podtip sadržaja deklarira specifični podtip - deklaracija tipa sadržaja omogućuje da implementacija klijenta odluči treba li korisniku prikazati ili ne prikazati određeni tip sadržaja. Takav je mehanizam razuman za npr. neprepoznate tipove teksta. Definicija tipa sadržaja sastoji se od: imena tipa, opisa, načina na koji klijent ili server treba rukovati s njim, razmatranja pri slanju i određenih restrikcija.
Inicijalni *"top-level"* tipovi sadržaja (eng. *initial "top-level" media types*) su:

- *text* (tekstualna poruka) - Najjednostavniji podtip tekstualne poruke je *plain* - on ukazuje da tekst ne sadrži nikakve naredbe, tj. tekst je *cleartext* (ili *plaintext*) i prikazuje se onakvim kakav jest. Za ostale podtipove teksta, npr. *html*, potreban je odgovarajući softver koji će ga interpretirati.
- *image*
- *audio*
- *video*
- *application* - sadržaj koji treba biti obrađen od strane vanjskog programa/aplikacije

Složeni *"top-level"* tipovi sadržaja su:

- *multipart* - podatak sastavljen od više nezavisnih podtipova sadržaja. Četiri podtipa multipart sadržaja su definirana:
  - *mixed* - generički skup sadržaja različitog podtipa
  - *alternative* - jednak podatak u različitim formatima. Npr.: uz glavni *html* sadržaj dodaje se alternativni, zamjenski sadržaj tipa *plain* u slučaju da klijent koji prikazuje poruku nema sučelje za html prikaz. Današnji klijenti uglavnom su optimizirani za prikaz html koda, i većina njih sakrit će činjenicu da zamjenski *plaintext* uopće postoji
  - *parallel* - sadržaji koji se trebaju prikazati istovremeno
  - *digest* - više dijelova od koji je svaki tipa *message*, a podtipa *rfc822*, tj. svaki dio unutar poruke je poruka definirana pomoću RFC 822 standarda
- *message* - enkapsulirana poruka gdje je tijelo tog sadržaja cijela poruka, ili dio poruke.

RFC 2047 opisuje postupke kojima zaglavlja mogu sadržavati internacionalne ne-ASCII znakove.
Unutar *Python* programskog jezika, moguće je kreirati poruku koristeći *built-in* modul email. Više o modulima za kreiranje poruke bit će rečeno u poglavlju 6. Primjer stvarne poruke koja se ispisuje unutar naredbenog retka pomoću poziva sys.stdout.buffer.write(message.as_bytes()):

```
To: maltar.jurica@gmail.com
From: maltar.jurica@yahoo.com
Subject: Pozdrav
Date: Fri, 21 Aug 2015 21:47:47 +0200
Message-ID: <20150821194747.3764.85065@Jurica.lan>
```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="===============0946742580=="

--===============0946742580==
Content-Type: multipart/alternative; boundary="===============0217871652=="

--===============0217871652==
Content-Type: text/html; charset="utf-8"
Content-Transfer-Encoding: 7bit

<p> Pozdrav iz <i>Pythona</i>. Jurica </p>

--===============0217871652==
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0

Pozdrav iz Pythona. Jurica.

--===============0217871652==--

--===============0946742580==
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
 filename="C:\\Users\\Jura\\Desktop\\python_network\\attachment.txt"
MIME-Version: 1.0

Ovo je tekst iz priloga "attachment.txt"

--===============0946742580==--
```

# 3 SMTP

SMTP (*Simple Mail Transfer Protocol*) protokol je aplikacijskog sloja za prijenos e-mail poruke putem TCP (*Transmission Control Protocol*) transportnog protokola, prvi put definiran 1982. godine s RFC 821 dokumentom, a trenutni dokument koji ga definira je RFC 5321. SMTP je baziran na klijent-server arhitekturi. Svaka poruka koju koja se prenosi unutar SMTP-a mora biti u ASCII kodu, sav ostali *encoding* mora biti konvertiran u ASCII pri slanju, a nakon primanja, dekodirati se u prvobitni oblik.
SMTP zadužen je za:

- potupak klijentova prijenosa sastavljene poruke od pošiljateljeva računala do mail servera
- postupak prijenosa poruke među mail serverima dok ona ne dođe do željene destinacije

Tijekom komunikacije klijenta i servera, veza može biti autentificirana i enkriptirana (ali i ne mora). Nakon što se poruka prenese do servera, smješta se u queue. Tada završava klijentova odgovornost. Drugi postupak ne zahtjeva autentifikaciju - popularni pružatelji e-mail usluga ne znaju podatke za autentifikaciju korisnika drugog pružatelja usluga. Kada bi Googleov mail server primio poruku od Yahoo!ovog mail servera, bilo bi nemoguće da Google server zna podatke o autentifikaciji Yahoo! korisnika. Za poboljšanje sigurnosti između samih servera koriste se mehanizmi pomoću kojih jedan server može konzultirati *Sender Policy Framework* kako bi saznao ima li server koji šalje poruku autoritet za dostavljanje poruke određene kompanije ili organizacije.
Važno svojstvo SMTP-a je sposobnost prenošenja poruke kroz više različitih mreža (SMTP *mail relaying*). Poruka može proći kroz mnoštvo posredničkih *releja* ("prekidača") i *poveznika* (eng. *gateway*) na svom putu do primatelja. Pri tom prolasku, DNS MX (*Domain name service mail exchanger*) korišten je za identifikaciju odgovarajuće sljedeće destinacije tijekom transporta. Kada klijent utvrdi domenu kojoj će poruka biti proslijeđena, DNS *look-up* mora biti izveden kako bi se odredila stvarna adresa. DNS pokušava locirati MX pridružen imenu domene. Nakon pronalaženja zapisa, rezultat može biti više alternativnih adresa umjesto jedne zbog višestrukih MX zapisa, ili zbog *višepristupnosti* (eng. *multihoming*) samog hosta (host je asociran s više IP adresa). Po redoslijedu adresa unutar zapisa, klijent šalje poruke, a redoslijed je određen pomoću prioritetnih indikatora (brojeva, pri čemu manji brojevi imaju veći prioritet). Server kojem se poruka šalje može biti ili krajnja destinacija poruke, ili relej ili poveznik. Također, prijenos se može ishodovati samo jednom vezom između dva mail servera.

## 3.1 SMTP prijenos

Prijenos poruke putem SMTP-a, prema [1, p. 121], odvija se u sljedećim koracima:

- pošiljatelj komponira poruku, dodavajući joj sadržaj, adresu primatelja i ostale informacije
- pošiljateljev klijent šalje poruku na svoj SMTP server
- poruka se sprema u *message queue* pošiljateljevog servera, koji sada postaje klijent i koji otvara TCP vezu sa primateljevim SMTP serverom
- nakon rukovanja i otvaranja veze, klijent šalje pošiljateljevu poruku

- primateljev mail server prima poruku te ju pohranjuje u mailbox
- primatelj preko svog klijenta dobiva poruku

Postupak komunikacije između klijenta i SMTP mail servera započinje uspostavljanjem dvokanalne veze. Kao što je spomenuto, klijent pronalazi SMTP server pomoću MX zapisa. Nakon uspostave veze i početnog rukovanja, klijent započinje s transakcijom poruke, a server svakoj naredbi odgovara. Odgovori servera mogu ukazati jesu li klijentove naredbe važeće ili da postoji pogreška.

### 3.1.1 Inicijaliziranje sjednice

Inicijaliziranje sjednice (eng. *session initiation*) počinje kada klijent otvori TCP vezu sa serverom na portu 25 i server odgovara pozdravnom porukom. SMTP koristi tzv. "persistent" vrstu konekcije - ukoliko klijent ima nekoliko poruka koje treba poslati primajućem serveru, to može učiniti pomoću jedne TCP veze. Nakon pozdravne"220" poruke može uslijediti informacija o softverskoj implementaciji servera. SMTP omogućava serveru da "odbaci" sjednicu prilikom inicijaliziranja - umjesto "220" koda, može biti poslan "554" i tada server mora čekati klijentovu "QUIT" naredbu prije zatvaranja i trebao bi odgovoriti s "503 bed sequence of commands". Nakon pozdravne poruke, klijent serveru šalje "EHLO", ukazujući na svoj identitet - klijent je "ESMTP-aware", tj. kompatibilan je s nizom dodatnih mogućnosti definiranih u SMTP protokolu (*Extended* SMTP). Stariji klijenti umjesto "EHLO" naredbe koriste "HELO" naredbu i server tada ne smije uzvratiti odgovor kompatibilan s ESMTP-om. Ukoliko klijent nakon "EHLO" naredbe od servera primi "command not recognized", server nije kompatibilan s ESMTP-om. Slijedi mail transakcija.

### 3.1.2 Mail transakcija

Mail transakcija počinje "MAIL" naredbom od strane klijenta koja identificira pošiljatelja. Slijedi jedna ili više "RCPT" naredbi koja opskrbljava server s informacijama o primateljima. Zatim, "DATA" naredba pokreće prijenos poruke. I na kraju, s "end of mail" indikatorom, tj. točkom "." klijent potvrđuje i završava transakciju.

"MAIL" naredba pojavljuje se u obliku:

```
MAIL FROM:<reverse-path> [SP <mail-parameters>] <CRLF>
```

Ova naredba govori SMTP serveru da je započeo novi prijenos poruke. Ukoliko server prihvati naredbu, vraća odgovor "220 OK". Ukoliko naredba nije prihvaćena od strane servera zbog specifikacije mailboxa, server vraća odgovor koji ukazuje je li pogreška trajna ili privremena. Pri pogreški, server vraća "550" ili "553" kod. `<reverse-path>` sadrži izvorni mailbox, a `<mail-parameters>` asocirani su s uspostavljenim dodatnim mogućnostima SMTP-a.

"RCPT" naredba povaljuje se u obliku:

```
RCPT TO:<forward-path> [ SP <rcpt-parameters> ] <CRLF>
```

Naredba se može pojaviti jednom ili više puta. Ukoliko primatelju nije moguće dostaviti poruku, server vraća "550" odgovor, najčešće "550 no such user". `<forward-path>` imenuje **jednog** primatelja (zato se naredba šalje jednom ili više puta), i identično kao kod "MAIL" naredbe, `<rcpt-parameters>` asocirani su s uspostavljenim dodatnim mogućnostima SMTP-a.

"DATA" naredba pojavljuje se u obliku:

```
DATA <CRLF>
```
Ukoliko je naredba prihvaćena, server vraća "354" odgovor i smatra da su sve naredne linije tekst poruke, osim točke na kraju koja označava kraj poruke. Kada je tekst poruke uspješno primljen i pohranjen, server šalje "250 OK". Ukoliko se prije "DATA" naredbe ne pojavljuju "MAIL" ili "RCPT" naredba, server može vratiti "354 no valid recipients" ili "503 command out of sequence"
Zatim slijedi "QUIT" naredba od strane klijenta. Pri slanju više poruka jednom serveru, putem "persistent" veze, klijent šalje nekoliko "MAIL" naredbi, a na kraju samo jednu "QUIT" naredbu.

## 3.2 ESMTP

Dodatne mogućnosti koje pruža SMTP protokol nazivaju se *extensions*. Dvije su za sigurnost bitne dodatne mogućnosti: stvaranje TLS-a i autentifikacija (eng. *authentication*).
Budući da SMTP koristi TCP kao transportni protokol, transportni je sloj moguće osigurati nadogradnjom TCP-a zvanom SSL (*Secure Socket Layer*). [1, p. 94] opisuje redoslijed kojim SSL djeluje - nakon što je *cleartext* podatak spreman za slanje, SSL kanal enkriptira podatak i prosljeđuje ga TCP kanalu. Kad se pojavi na željenoj destinaciji, enkriptirani se podatak pomoću SSL-a dekriptira. TLS (*Transport layer security*) nadograđena je verzija SSL-a. Osigurava tajnost i integritet poruke te autentifikaciju (potpoglavlje 5, str. 14). RFC 3207 definira novu naredbu pomoću koje se uspostavlja TLS između entiteta koji sudjeluju u SMTP prijenosu - "STARTTLS" - naredba kojom klijent ukazuje serveru da želi supostaviti TLS. Pojavljuje se u obliku:

```
STARTTLS
```

- bez dodatnih parametara. Server tada odgovara s "220 ready to start TLS", "501 syntax error (no parameters allowed)" ili "454 TLS not available due to temporary reason". Ukoliko je poslan "454" odgovor, klijent odlučuje želi li nastaviti SMTP sjednicu. Ukoliko server vrati "220" odgovor, klijent mora započeti s ugovaranjem TLS-a prije nego li izda bilo koju drugu naredbu. Nakon TLS-rukovanja, obje strane odlučuju hoće li nastaviti sjednicu ili ne, u ovisnosti o tome je li postignut određen stupanj sigurnosti. Ukoliko sigurnost nije postignuta, klijent izdaje "QUIT" naredbu nakon što je TLS uspostavljen, a server na svaku klijentovu naredbu, osim "QUIT" odgovara s "554 command refused due to lack of security". Ukoliko je TLS uspostavljen, zasjedanje se vraća u prvobitno stanje - server i klijent "zaboravljaju" sve što su međusobno izmjenili, npr. dodatne argumente uz "EHLO" naredbu i listu dodatnih SMTP mogućnosti. Nakon uspješnog ugovaranja TLS-a, klijent ponovno šalje "EHLO" naredbu i počinje postupak opisan u inicijaliziranju sjednice nakon uspostavljanja veze.
"AUTH" naredbom, definiranom RFC 4954 dokumentom, klijent potvrđuje svoj identitet nekim od mehanizama koje SMTP server podržava, tako da serveru šalje korisničko ime i zaporku.
Naredba se pojavljuje u obliku:

```
AUTH mechanism
```

`mechanism` označava protokol kojim će autentifikacija biti postignuta, a dostupne protokole server izlistava klijentu kako bi on izabrao kako autetificirati vezu. Mogući protokoli navedeni su unutar RFC dokumenta. Osnovni protokol je "PLAIN" - podaci koje klijent šalje serveru nisu enkriptirani, već samo enkodirani. Npr. klijent šalje naredbu "AUTH PLAIN", a server odgovara s "334". Tada klijent šalje korisničko ime i zaporku unutar jednog *base64* stringa. Korisničko ime i zaporku moguće je dostaviti i direktno uz "AUTH PLAIN" naredbu. Uko-

liko je autentifikacija uspješna, server odgovara s "235 Authentication successful". Nakon uspješnog potvrđivanja identiteta, kao i u slučaju uspostavljanja TLS-a, počinje postupak ponovnog slanja "EHLO" naredbe i ostalih postupka opisanih u inicijaliziranju nakon uspostavljanja veze.

Primjer[4] uspostavljanja TLS-a i autentifikacije:

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250-AUTH GSSAPI DIGEST-MD5
S: 250-ENHANCEDSTATUSCODES
S: 250 STARTTLS
C: STARTTLS
S: 220 Ready to start TLS
...  TLS negotiation proceeds, further commands
protected by TLS layer ...
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250 AUTH GSSAPI DIGEST-MD5 PLAIN
C: AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=
S: 235 2.7.0 Authentication successful
```

[4]Primjer preuzet iz [8].

# 4 Mail Access Protocols

Kada se poruka od pošiljatelja do primatelja dostavi putem SMTP protokola, poruka biva smještena u primateljev mailbox. Do početka devedesetih godina prošloga stoljeća, primatelj je čitao svoju poštu tako da se ulogirao na server i pokrenuo čitač elektroničkih poruka. U današnje vrijeme korisnik čita elektroničku poštu putem klijenta, koji se odvija na korisnikovom računalu, koji pristupa mailboxu koji se nalazi na uvijek aktivnom korisnikovom mail serveru. Ukoliko je korisnikov mail server njegovo računalo, tada bi ono, ukoliko svaka poslana poruka namjenjena njemu treba biti primljena, trebalo biti uključeno konstantno.
Primateljev klijent pristupa poruci pomoću *mail access protocol-a*, kako samo ime govori. SMTP nije protokol te kategorije, jer SMTP je *push* protokol. Tri su značajna protokola za pristup elektoničkim porukama: POP3 (*Post Office Protocol 3*), IMAP (*Internet Message Access Protocol*) i HTTP (*HyperText Transfer Protocol*). HTTP nije prvenstveno stvoren za pristup porukama, između ostalog on služi i za slanje poruka, kako je navedeno. Općenito, HTTP je protokol aplikacijskog sloja baziran na klijent-server arhitekturi i TCP transportnom sloju koji služi za razmjenu *hypermedia* sadržaja.

## 4.1 POP3

POP3 je jednostavan i arhaičan mail access protokol definiran s RFC 1939 dokumentom. POP3 *sjednica* između primateljeva POP3 klijenta i njegova mail servera odvija se u 4 faze:

- *povezivanje* - klijent otvara TCP vezu na portu 110
- *autentifikacija* - klijent šalje primateljevo korisničko ime i zaporku pomoću naredba "USER" i "PASS"
- *prijenos* - klijent dohvaća poruku, u ovisnosti o konfiguraciji klijenta označava ju za brisanje i preuzima statistiku o poruci
- *nadogradnja* - klijent šalje "QUIT" naredbu, završava se POP3 sjednica i mail server briše poruku ukoliko je klijent to zatražio

POP3 klijent može biti konfiguiran na 2 načina - *"download and delete"* u kojem će poruka nakon preuzimanja, od strane klijenta biti označena za brisanje, i od strane servera obrisana te *"download and keep"* u kojem poruka ostaje pohranjena na serveru. Problem s prvim načinom je opasnost od gubitka poruke jednom zauvijek; npr. ukoliko korisnik pomoću Outlook Expressa ima konfiguiran stari "Iskon" račun za e-mail koji koristi POP3 protokol, nakon dohvaćanja poruka na svoje računalo, poruke zauvijek nestaju sa servera i ne može im se pristupiti putem drugog računala.

## 4.2 IMAP

IMAP, tj. njegova suvremena verzija IMAP4rev1 dodefinirana 2003. godine s RFC 3501 dokumentom, protokol je baziran na klijent-server arhitekturi, koji koristi TCP transportni sloj. Korisniku omogućava pristup i upravljanje elektroničkim porukama na serveru. Mnogo je napredniji protokol od POP3, ali zasniva se na njegovim načelima. Ipak, poruke se unutar mailboxa mogu organizirati - funkionalnost IMAP zasniva se na tome što se direktorijima kreiranim u mailboxu na serveru može manipulirati na jednak način na koji se manipulira lokalnim direktorijima na računalu. Svaka poruka na IMAP serveru bit će pridružena

određenom direktoriju. Kada se poruka pojavi na serveru, u početku će biti spremljena u direktorij "inbox". Tada korisnik može čitati poruku, obrisati ju svojevoljno, premjestiti poruku u drugi direktorij itd.
Prednosti IMAP-a nad POP-om, kako [2, p. 267] navodi su:

- jedna poruka može biti pridružena različitim direktorijima
- IMAP podržava *zastavice* (eng. *flags*) kojima se poruke označavaju
- poruka se može pretraživati direktno iz mailboxa, korisnik ju ne mora u potpunosti preuzeti na svoje računalo
- lokalno pohranjene poruke mogu se *uploadati* u direktorije na IMAP serveru
- mogućnost preuzimanja pojedinog dijela poruke (npr. samo tekstualni dio poruke)

Unutar modernog IMAP klijenata (npr. Thunderbird ili Outlook) moguća je sinkronizacija korisnikove aktivnosti: npr. ukoliko korisnik preuzme poruku iz mailboxa, isključi internetsku vezu, pogleda poruku i odgovori na nju, pri ponovnom povezivanju na mrežu, podaci o tome da je na poruku odgovoreno, da je poruka viđena, itd., bit će dostavljeni serveru na kojem se nalazi poruka.
RFC 3501 dokumentom definirani su postupci za: upravljanje direktorijima na serveru (kreiranje, preimenovanje, brisanje...), provjeravanje novodostiglih poruka, trajno uklanjanje poruka, postavljanje zastavica i njihovo uklanjanje, parsiranje poruka, pristup porukama unutar mailboxa, preuzimanje pojedinačnih dijelova poruke... Ti se postupci odvijaju, kao i kod SMTP protokola, izmjenjivanjem naredaba i odgovora u obliku linija (sadržaj naredbe/odgovora + CRLF) od strane klijenta i servera, i unutar [3] se nazivaju međudjelovanje (eng. *interaction*) klijenta i servera. Prije međudjelovanja, potrebno je uspostaviti, inicijaliziranu od strane klijenta, vezu i obaviti klijent/server *rukovanje* propisano dokumentom. IMAP sjednica nakon uspostavljanja veze i početka međudjelovanja može biti unutar četiri propisana stanja: *Not authenticated* stanje, *Authenticated* stanje, *Selected* stanje i *Logout* stanje, a određene klijentove naredbe dopuštene su za pojedina stanja. U prvom navedenom stanju, klijent mora dostaviti podatke za autentifikaciju i tada, pri uspjehu, prelazi u Authenticated stanje. Dopuštene naredbe od strane klijenta u non authenticated stanju su "STARTTLS", "AUTHENTICATE" i "LOGIN". Nakon uspješnog odabira mailboxa na serveru pomoću "SELECT" naredbe, IMAP sjednica je u Selected stanju. Logout stanje postiže se pri okončavanju veze, tj. kada klijent pošalje "LOGOUT" naredbu. "LOGOUT", kao i "CAPABILITY" (nakon koje server ukazuje na svoje mogućnosti) i "NOOP" dopuštene su u svakom stanju. Ukoliko je klijent neaktivan određeno vrijeme, server pomoću *autologout* brojača može samoinicijativno okončati vezu. Klijent tada (prije isteka vremena) može poslati "NOOP" naredbu kojom poništava brojač.
Uz samu poruku koja pristigne u mailbox, pridruženo je nekoliko podataka koji se nazivaju atributi poruke. Atribut koji ukazuje na vrijeme kada je poruka dostavljena u mailbox naziva se *internal date*, a *size* atribut ukazuje na veličinu poruke u bajtovima. Za dohvaćanje poruka unutar IMAP protokola, moguće je koristiti dvije vrste atributa, redni broj poruke ili UID (*unique identifier*). UID je jedinstvena 32-bitna vrijednost pridružena svakoj poruci unutar mailboxa, a svaka sljedeća poruka imat će rastući UID. UID atributi su trajni i UID koji je pridružen jednoj poruci ne smije se više mjenjati, te svaka poruka unutar pojedinog mailboxa ne smije imati UID koji je već pridružen nekoj poruci. Redni broj poruke drugi je atribut generiran slijedom kojim se poruka pojavi unutar mailboxa, tj. kad se poruka

dostavi u mailbox, pridružen joj je za jedan veći redni broj od prethodne poruke. Ipak, redni broj poruke može se mijenjati - npr. ukoliko je poruka obrisana, svim porukama koje su je slijedile, redni broj bit će dekrementiran za jedan. Također, novoj poruci može biti pridružen redni broj koji je bio pridružen poruci koja je obrisana. Atributi koji ukazuju na stanje poruke unutar mailboxa nazivaju se zastavice. Zastavice se dodavaju i brišu iz liste koja ih pohranjuje. Dva su tipa zastavica - trajne i trenutne (*session-only*). *System* zastavice počinju s "\" i one su: *"\Seen"* - ukazuje da je poruka pročitana, *"\Answered"* - korisnik je odgovorio na poruku, *"\Flagged"* - označava poruku važnom, *"\Deleted"* - poruka je označena za brisanje od strane servera nakon klijentove naredbe "EXPUNGE", *"\Draft"* - poruka nije dovršena, *"\Recent"* - poruka je upravo dostavljena u mailbox.

# 5 Sigurnost

Generalno, informacije koje se prosljeđuju putem Interneta ponekad trebaju biti zaštićene radi sprečavanja zlouporabe od strane uljeza. Četiri su sigurnosna svojstva unutar mreže: *tajnost*, *integritet poruke*, *autentifikacija* i *operacijska sigurnost*. Uljez može potajno pratiti razgovor radi dobivanja željenih informacija, ili modificirati, umetnuti te obrisati poruku.
Tajnost (eng. *confidentiality*) - informacija unutar komunikacije između pošiljatelja i planiranog primatelja treba biti razumljiva isključivo njima. Jer uljez može presresti poruku, poruka treba biti enkriptirana.
Integritet poruke - strane koje sudjeluju u konverzaciji žele biti sigurne da je njihova poruka neizmjenjena pri prijenosu, ili slučajno ili namjerno. Također, primatelj treba znati da poruka zasigurno potječe od pošiljatelja.
Autentifikacija - pošiljatelj i primatelj trebali bi potvrditi identitet međusobno.
Operacijska sigurnost - sprečava zlouporabu unutar organizacijskih mreža.
Kriptografske tehnike dopuštaju pošiljatelju da zamjeni podatak tako da uljez nema informaciju o podatku koji je presreo, dok primatelj poruke treba biti sposoban konvertirati zamjenjeni podatak u originalni oblik. Više o kriptografskim tehnikama bit će navedeno u kontekstu sigurnosti elektroničke pošte.

## 5.1 Simetrični i javni ključ

Simetrični ključ identičan je kod primatelja i pošiljatelja. Povjesni postupci za enkripciju teksta pomoću simetričnog ključa su *Cezarova šifra* i *"Monoalphabetic" šifra*. Dvije su vrste enkriptiranja pomoću simetričnog ključa - *steam* šifra (koristi se za sigurnost WLAN-a) i *block* šifra (koristi se za sigurnost TCP veza, transporta unutar mrežnog sloja, sigurnost elektroničke pošte...) Pomoću block šifre poruka se enkriptira tako da se procesuiraju blokovi od k bitova. Npr. $k = 64$ - poruka se dijeli u 64-bitne blokove i svaki od tih blokova enkriptira se nezavisno. Tada pošiljatelju i primatelju treba biti poznat *mapping* - podaci za enkripciju svake k-bitne kombinacije bitova, tj. supstitucija za svaki blok. Ali, to je složen zadatak za velike k-ove. U primjeru s $k = 64$, pošiljatelj i primatelj trebaju posjedovati podatke o $2^{64}$ vrijednosti. Zato postoje funkcije koje nasumično kombiniraju mapping. Danas popularne šifre koje koriste funkcije su DES (*Data encryption standard*), 3DES i AED (*Advanced encryption standard*).
Javni ključ vidljiv je svakome, a određen je od strane primatelja, koji posjeduje i privatni ključ. Zahvaljujući rezultatima iz teorije brojeva, pošiljatelj enkriptira poruku pomoću enkripcijskog algoritma unutar kojeg koristi javni ključ i šalje ju. Primatelj enkriptiranu poruku dekriptira pomoću dekripcijskog algoritma koristeći privatni ključ. Najpoznatiji *public key* algoritam je RSA (Rivest, Shamir i Adleman, 1977.). Javni se ključ koristi u kombinaciji s simetričnim, budući da je spor (ali pouzdan): prvo pošiljatelj odabire ključ kojim će enkodirati podatke - *session key*, koji se zatim enkriptira putem javnog ključa i šalje primatelju.

## 5.2 Sigurnost elektroničke pošte

Važna sigurnosna svojstva za elektroničku poštu su *tajnost*, *integritet poruke* i *autentifikacija*. Ona su osigurana uz navedene dodatne mogućnosti SMTP-a (potpoglavlje 3.2, str. 9). Najjednostavniji način za osiguravanje tajnosti jest da pošiljateljev klijent poruku enkriptira koristeći *symmetric key* mehanizme (DES i AES), ali problem s ovim pristupom je u nejednostavnom opskrbljivanju javnog ključa objema stranama unutar konverzacije. Promišljeniji je zato pristup putem *public key* mehanizama (npr. RSA). Primateljev *javni ključ* dostupan

je svima, i pošiljatelju i uljezu. Pomoću javnog ključa i *enkripcijskog algoritma*, pošiljateljev klijent enkriptira poruku koju šalje na primteljevu e-mail adresu. Kada primateljev klijent primi poruku, jednostavno ju dekriptira pomoću svog *privatnog ključa*. Problem s RSA pristupom je neefikasnost. Zato se koristi *session key*. Pošiljateljev klijent izabire slučajan *symmetric-session* ključ, enkriptira poruku sa simetričnim ključem, enkriptira simetrični ključ s primateljevim javnim ključem, konkatenira enkriptiranu poruku i enkriptirani simetrični ključ u tvorevinu zvanu paket (eng. *package*). Kada primateljev klijent primi paket, dekonkatenira ga, koristi privatni ključ za dekriptiranje simetričnog ključa i taj simetrični ključ koristi za dekriptiranje poruke.
Postupak za osiguravanje integriteta poruke i autentificiranje pošiljatelja počinje kada njegov klijent primjeni *hash funkciju*[5] na poruku, enkriptira hash poruke pomoću svog privatnog ključa tako da stvori *digitalni potpis*[6], konkatenira originalnu neenkriptiranu poruku zajedno s digitalnim potpisom i tako stvori paket koji šalje. Kada primateljev klijent primi poruku, primjenjuje pošiljateljev javni kluč na digitalni potpis i uspoređuje rezultat s vlastitim hashom poruke.
Sva tri sigurnosna svojstva postižu se tako da pošiljatelj prvo kreira paket u kojem se nalazi originalna poruka i digitalno postpisani hash poruke. Tada se taj podatak tretira kao početna poruka. Tu poruku pošiljatelj enkriptira pomoću simetričnog ključa, a simetrični ključ pomoću javnog ključa primatelja. Primatelj tada koristi već navedene postupke kako bi raspakirao pristiglu poruku.

[5]Hash funkcija $H()$, uzimajući "input", npr. poruku koja se treba poslati, tvori konačan niz znakova koji jedinstveno reprezentiraju poruku - uljezu je gotovo nemoguće kreirati hash neke druge poruke, tako da lažni hash i autentični budu jednaki, tj. ne postoje poruke $m$ i $m'$ tako da je $H(m) = H(m')$.

[6]Digitalni potpis (eng. *"digital signature"*) kriptografska je tehnika ekvivalentna potpisivanju fizičkih dokumenata. Digitalni se potpis stvara tako da pošiljatelj koristeći privatni ključ enkriptira poruku.

# 6 Implementacija e-mail klijenta u Python programskom jeziku

U implementaciji smtp_client.py korišteno je nekoliko standardnih modula koji služe za postupke kreiranja i slanja poruke. Sama implementacija struktuirana je u dvije funkcije - send() i create_message() - slijedom koji je prisutan u današnjim *webmail* klijentima (doduše, ovo nije webmail implementacija) - prvo se uspostavlja veza sa serverom, zatim dolazi do enkriptiranja veze i autentifikacije te tada korisnik stvara poruku koja će biti poslana. Modul smtp_client.py poziva se pomoću naredbenog retka tako da uz ime modula korisnik dostavi host i port servera s kojim želi uspostaviti komunikaciju - npr. `smtp_client.py smtp.gmail.com 587`. Pri pozivu send() funkcije prvo putem smtplib modula instanciramo vezu, tj. instanciramo klasu SMTP kojoj prosljeđujemo podatke o serveru koji su parsirani pri pozivanju modula unutar naredbenog retka. Nakon toga pojavljuje se zakomentirana metoda instance set_debuglevel(1) koja prikazuje cjelokupnu komunikaciju između klijenta i servera (zbog duljine, cjelokupni primjer komunikacije nije sadržan unutar poglavlja 3). Tada, slijedeći SMTP pravila opisana unutar RFC dokumenta, klijent serveru šalje "EHLO" naredbu putem funkcije ehlo() implementirane unutar smtplib modula, tj. metode klase SMTP i ukoliko je vraćen valjan odgovor, server podržava dodatne mogućnosti (*ekstenzije*). Ekstenzije koje smtp_client.py zahtjeva su enkriptiranje veze i autentifikacija. Ova implementacija ne koristi tzv. *opportunistic* pristup - pri takvom pristupu, ukoliko server ne nudi mogućnost enkripcije veze, klijent može pokušati nastaviti koristiti ne-enktiptiranu običnu vezu. Ovdje implementirani klijent pri nemogućnosti kreiranja TLS-a završava komunikaciju sa serverom. Unutar [2] pojavljuje se *opportunistic* implementacija. Takva implementacija ovdje ne bi niti koristila svrsi, budući da se povezujemo na poznate sustave koji zahtjevaju TLS vezu prije autentifikacije. Provjeravamo postoji li ekstenzija "STARTTLS", i tada uspostavljamo TLS vezu. Nakon uspostave TLS-a i ponovnog slanja "EHLO" naredbe (potpoglavlje 3.2, str. 9), slijedi provjera postojanja ekstenzije "AUTH" i autentifikacija - od korisnika se zahtjeva da putem naredbenog retka unese korisničko ime (putem standardne input() funkcije) i zaporuku (putem funkcije getpass()[7]). Sama autentifikacija ne bi niti imala smisla ukoliko ne bi bilo enkriptirane veze - moguće bi bilo npr. na istom *routeru* na koje je računalo povezano, preuzeti sve podatke koje korisnik šalje na server, pa tako i one za *login*. Nakon parsiranja korisničkog imena i zaporke slijedi pozivanje funkcije login() – ona šalje "AUTH PLAIN" naredbu i slijedi postupak autentifikacije, a nakon toga ponovno slanje "EHLO" naredbe.

Unutar funkcije send() poziva se funkcija create_message() koja vraća kreiranu poruku. create_message() prima prethodno parsirani parametar *username*. Tada se pomoću nekoliko input() funkcija od korisnika zahtjeva da unese podatke potrebne za kreiranje poruke - *primatelje*, *predmet*, *tekstualnu poruku* i *privitke*. Kada su podaci uneseni slijedi instanciranje poruke. Poruka naziva "message" instanca je klase EmailMessage kojoj prosljeđujemo "email.policy.SMTP" parametar kojim ona postaje kompatibilna za prijenos unutar SMTP-a. Unutar klase EmailMessage implementirana je metoda __getitem__() - objekti unutar instance klase EmailMessage dohvaćaju se na identičan način kao što se vrijednosti (eng. *values*) dohvaćaju putem metode __getitem__() u rječniku (eng. *dictionary*). Upravo na taj način promatramo poruku kao rječnik - npr. vrijednosti 'To' program pridružuje odgovarajući string: message['To'] = to, itd. Datum i vrijeme nastaju pomoću metode formatdate() unutar modula utils. Datum i vrijeme (u kodu 'Date') su string koji se sastoji od čovjeku ra-

[7] Takva se metoda pojavljuje i prilikom logina u Linux sustavu - korisnik upisuje zaporku, ali ona nije vidljiva na ekranu.

zumljivih podataka: pozvavši email.utils.formatdate(localtime=True) slijedi output: “Fri, 12 Jun 2015 22:08:41 +0200”. Sljedeći bitan podatak, također generiran unutar modula utils je message-id. Message-id je jedinstveni niz brojeva i znakova koji jedinstveno određuju svaku ikad kreiranu e-mail poruku. On se generira u ovisnosti o više faktora - vrijeme, ime računala, ime mreže, itd. Nikada se neće, kako tvrdi [2, p. 226], generirati dva identična message-id-a. Sljedeći podaci koji se pridružuju instanci “message” su tekstualna poruka i privitak. Pozvavši metodu set_content() u instanci “message” formira se novo zaglavlje čiji je sadržaj tipa “text/plain” unutar kojeg je tijelo s tekstualnom porukom. Upravo zbog MIME protokola sam tekst može biti bilo kojeg tipa - npr. dozvoljeni su znakovi “č”, “ć”, “ž”, “æ” itd. Na kraju, pomoću metode add_attachment() kojoj prosljeđujemo tip, podtip, ime privitka i sam podatak kao niz *bajtova*, dodajemo zaglavlje. Na primjer, ukoliko u naredbenom retku, nakon “Attachment:” unesemo “slika.bmp” tip sadržaja zaglavlja bit će : “image/bmp”. Sam tip podatka privitka prepoznat je putem mimetypes modula. Implementacija unutar koda za prepoznavanje tipova i podtipova sadržaja preuzeta je iz [2]. Metoda guess_type() kojoj se prosljeđuje ime privitka (npr. “piano.mp3”) prepoznat će da se radi o “audio” kao tipu i “mp3” kao podtipu sadržaja. Na kraju, zbog implementacije metode sendmail() unutar smtplib modula, instancu “message” pretvaramo u *bajtove* i šaljemo primateljima poruke putem metode sendmail() te zatvaramo vezu putem metode exit() kojom klijent šalje “QUIT” naredbu serveru.

## 6.1 smtp_client.py

```
import smtplib
import socket
import sys
import getpass
import email.policy
import email.message
import email.utils
import mimetypes
import os

#mapa u kojoj se nalaze privitci
dir = r'C:\Users\Jura\Desktop\python_network'
os.chdir(dir)

def send():

    if len(sys.argv) == 1:
        print('Unesite ime i port servera ponovno')
        sys.exit()

    server  = (sys.argv[1], sys.argv[2])

    connection = smtplib.SMTP(server[0],server[1])
    #connection.set_debuglevel(1)
    reply = connection.ehlo()[0]
    if not (200 <= reply <= 299):
        print('Server ne podrzava ESMTP')
        sys.exit()
```

```
    if connection.has_extn('starttls'):
        try:
            connection.starttls()
        except smtplib.SMTPException as e:
            print('Nije moguce uspostaviti TLS')
            print(e)
            sys.exit()
    else:
        print('Nije moguce uspostaviti TLS')
        sys.exit()

    reply = connection.ehlo()[0]
    if not (200 <= reply <= 299):
        print('Problem pri komunikaciji klijenta i servera')
        sys.exit()

    if connection.has_extn('auth'):
        print('Login:')
        username = input('Enter username: ')
        password = getpass.getpass('Enter password: ')
        try:
            connection.login(username, password)
        except smtplib.SMTPException as e:
            print('Problem pri autentifikaciji')
            print(e)
            sys.exit()
    else:
        print('Problem pri autentifikaciji')
        sys.exit()

    reply = connection.ehlo()[0]
    if not (200 <= reply <= 299):
        print('Problem pri komunikaciji klijenta i servera')
        sys.exit()

    message = create_message(username)

    connection.sendmail(message['From'], message['To'],\
                        message.as_bytes())
    print('Uspjeh')
    connection.quit()

def create_message(username):
    print('From: {}'.format(username))
    to = input('To: ')
    to = to.split(' ')
    subject = input('Subject: ')
    text = input('Text: ')
    attachments = input('Attachments: ')
    attachments = attachments.split(' ')
```

```
    message = email.message.EmailMessage(email.policy.SMTP)
    message['To'] = to
    message['From'] = username
    message['Subject'] = subject
    message['Date'] = email.utils.formatdate(localtime=True)
    message['Message-ID'] = email.utils.make_msgid()

    message.set_content(text)

    if attachments != ['']:
        for attachment in attachments:
            mime_type, encoding = mimetypes.guess_type(attachment)

            if encoding or (mime_type is None):
                mime_type = 'application/octet-stream'

            main, sub = mime_type.split('/')

            if main == 'text':
                with open(attachment, encoding = 'utf-8') as f:
                    text_attachment = f.read()
                message.add_attachment(text_attachment, sub,\
                                       filename = attachment)
            else:
                with open(attachment, 'rb') as f:
                    data = f.read()
                message.add_attachment(data, main, sub,\
                                       filename = attachment)
    return message



if __name__ == '__main__':
    send()
```

# 7 Zaključak

Sustav elektroničke pošte razvijao se od svojih početaka sukladno s razvojem informatičkih tehnologija. Tako u samom početku elektroničke pošte osamdesetih godina prošloga stoljeća, poruka je bila sastavljena od isključivo tekstualnoga dijela, da bi se definiranjem MIME standarda ona pretvorila u multimedijalni sadržaj od čak nekoliko tipova - danas je uobičajno primiti poruku koja je sadrži html stranicu i nekoliko privitaka. Povećanjem protočnosti podataka (eng. *bandwidth*) takva je poruka dostupna korisniku u trenu.
Sustavi kojima se poruka prenosi striktno su definirani pomoću RFC dokumenata, i pri najmanjoj pogreški pri komunikaciji dvaju entiteta poruku nije moguće dostaviti. Zato je pri implementaciji sučelja za prijenos poruke potrebno voditi računa o definiciji protokola. Python nudi standardne module za izgradnju takvih sučelja.

# Literatura


[1] J. F. Kurose, K. W. Ross, *Computer Networking: A Top-Down Aproach*, Addison-Wesley Publishing Company, NJ, 6th edition, 2013.

[2] B. Rhodes, J. Goerzen, *Foundations of Python Network Programming*, Apress, NY, 3rd edition, 2014.

[3] M. Crispin, *Internet Message Access Protocol - Version 4rev1*, Request for Comments: 3501, 2003.

[4] P. Resnick, *Internet Message Format*, Request for Comments: 5322, 2008.

[5] N. Freed, N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Request for Comments: 2045, 1996.

[6] N. Freed, N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, Request for Comments: 2046, 1996.

[7] J. Klensin, *Simple Mail Transfer Protocol*, Request for Comments: 5321, 2008.

[8] R. Siemborski, A. Melnikov, *SMTP Service Extension for Authentication*, Request for Comments: 4954, 2007.

[9] P. Hoffman, *SMTP Service Extension for Secure SMTP over Transport Layer Security*, Request for Comments: 3207, 2002.