УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Филип Дутина

Једно решење програмске подршке за безбедан пренос података путем *BroadR-Reach* спреге

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2018

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Завршни (Bachelor) рад
Аутор, **АУ**:		Филип Дутина
Ментор, **МН**:		Доц. др Богдан Павковић
Наслов рада, **НР**:		Једно решење програмске подршке за безбедан пренос података путем *BroadR-Reach* спреге.
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2018
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		Безбедан пренос података, енкрипција/декрипција, *BroadR-Reach* спрега, *IPv6* протокол
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Задатак рада је дизајнирање и имплементација безбедног преноса података са наменске платформе на рачунар. Енкрипција података који се преносе је неопходна како би се осигурала безбедна и сигурна комуникација. Пренос података се врши путем *BroadR-Reach* спреге и *IPv6* протокола.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан:	Потпис ментора
	Члан, ментор:	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monographic publication
Type of record, **TR**:	Textual printed material
Contents code, **CC**:	Bachelor Thesis
Author, **AU**:	Filip Dutina
Mentor, **MN**:	Bogdan Pavković, Phd
Title, **TI**:	One software solution for safe data transmision via *BroadR-Reach* interface.
Language of text, **LT**:	Serbian
Language of abstract, **LA**:	Serbian
Country of publication, **CP**:	Republic of Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2018
Publisher, **PB**:	Author's reprint
Publication place, **PP**:	Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)	
Scientific field, **SF**:	Electrical Engineering
Scientific discipline, **SD**:	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S/KW**:	Safe data transmission, encryption/decription, *BroadR-Reach* interface, *IPv6* protocol
UC	
Holding data, **HD**:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:	
Abstract, **AB**:	This paper presents design and implementation of safe data transmission from embedded board to the PC. Data encryption is necessary in order to secure safe communication. Data transmission is done via *BroadR-Reach* interface and *IPv6* protocol.
Accepted by the Scientific Board on, **ASB**:	
Defended on, **DE**:	
Defended Board, **DB**: President:	
Member:	
Member, Mentor:	

Menthor's sign

Zahvalnost

Srdačno se zahvaljujem mentoru Doc. dr Bogdanu Pavkoviću na stručnoj pomoći tokom izrade ovog rada.

Posebno se zahvaljujem Draganu Radanoviću na stručnoj pomoći, stalnoj podršci, strpljenju, utrošenom vremenu, razumevanju i savetovanju tokom izrade ovog diplomskog rada.

Takođe se zahvaljujem Institutu *RT-RK* na ukazanoj prilici da se bolje upoznam sa načinom rada u inženjerskom okruženju i na omogućenom usavršavanja znanja iz date oblasti.

Na kraju, najveće hvala Tatjani, Milošu, Vjeri, Mirku, Jeleni i svima onima koji su mi bili podrška, potpora i motivacija tokom školovanja.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

SoC - *System on a Chip*, integisano kolo koje sadrži sve elektronske komponente nekog elektronskog sistema na jednom čipu

TCP - *Transmission Control Protocol*, mrežni protokol

IPv6 - *Internet Protocol Version 6,* šesta revizija Internet Protokola

RSA - *Rivest-Shamir-Adleman algorithm*, algoritam asimetrične enkripcije

MIT - *The Massachusetts Institute of Technology*, Tehnološki institut Masačusetsa

RTOS - *Real-time operating system*, operativni sistem koji radi u realnom vremenu

ARM - *Advanced RISC machine/Acorn RISC machine*, familija *RISC* čipova

ACK - *Acknowledgement*, paket koji potvrđuje neku tvrdnju

GUI - *Graphical user interface*, grafička korisnička sprega

ADAS - *Advanced driver-assistance systems*, sistem za asistenciju vozaču

MSC - *Message sequence chart*, dijagram interakcije

1. Uvod

Da bi autonomni automobili preuzeli primat na putevima i postali deo svakodnevice, jedan od najvećih izazova sa inženjerske tačke gledišta svakako jeste prenos, skladištenje i manipulacija podacima. Navodno, devedeset minuta vožnje jednog autonomnog automobila predstavlja ekvivalent korišćenju milion pametnih telefona u rasponu od dvadeset i četiri časa po obimu obrađenih podataka. U savremenim automobilima može se nalaziti više desetina, sve češće i više stotina elektronskih sistema, koji međusobno komuniciraju preko automobilskih mreža (*CAN, CAN HS, Ethernet, LIN* itd.). Sa stanovišta bezbednosti od neželjenog pristupa, sigurnost automobilskih mreža nije na nivou kao što je u potrošačkoj industriji, već mora biti na daleko višem nivou. Ovde je potrebno obezbediti dodatnu zaštitu, kao i šifrovanje podataka, kako bi se onemogućila, ili makar znatno otežala manipulacija podacima koji se čuvaju u automobilu, kao što su dijagnostički podaci, rezultati očitavanja senzora, podaci o izvršenim popravkama i servisima, ukupna pređena kilometraža vozila i slično.

U ovom radu biće opisano jedno rešenje programske podrške za bezbedan prenos podataka sa namenske ploče na računar, njihova enkripcija pre slanja i dekripcija pri primanju, putem *BroadR-Reach* sprege i *IPv6* protokola.

Rad je sačinjen od 7 poglavlja. Prvo poglavlje sadrži kraći uvod u rad, motivaciju i opis samog zadatka.

U drugom poglavlju se nalaze teorijske osnove koje su neophodne za shvatanje rada - opis *BroadR-Reach* sprege, pojašnjenje *TCP* i *IPv6* protokola kao i nekoliko reči o asimetričnoj *RSA* enkripciji.

Treće poglavlje sadrži koncept rešenja.

Četvrto poglavlje daje detaljan opis programskog rešenja sa dubljim analizama napisanih funkcija i kritičnih sekcija koda.

U petom poglavlju su predstavljeni rezultati testnih slučajeva i njihovo tumačenje.

Šesto poglavlje sadrži kratak pregled onoga što je urađeno u ovom radu i ideje na koji način bi programska podrška mogla dalje da se razvija.

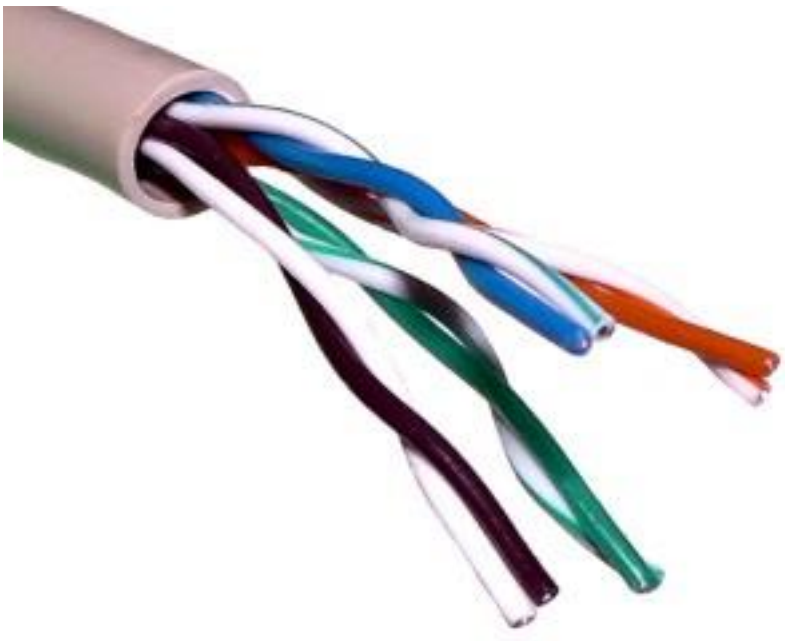
U sedmom poglavlju je dat spisak korišćene literature tokom izrade ovog rada.

2. Teorijske osnove

2.1 *BroadR-Reach* sprega

Slika 2.1 *BroadR-Reach* sprega [1]

BroadR-Reach tehnologija predstavlja standard fizičkog sloja koji je osmišljen za umrežavanje komponenti u automobilskoj industriji. *BroadR-Reach* pruža mogućnost istovremenog pristupa informacijama preko neoklopljenih upredenih parica.



Slika 2.2 Neoklopljene uprede parice [2]

Jedna od najvećih prednosti koje proizvođačima pruža ova tehnologija je mnogostruko smanjena cena i težina kablova u vozilu.

Korišćenje *BroadR-Reach* tehnologije omogućava prelazak sa više zatvorenih mreža u automobilu na jednu otvorenu koja je bazirana na *Eternet* (engl. *Ethernet*) protokolu. Ovo dozvoljava proizvođačima da ugrade veliki broj elektronskih sistema i uređaja u vozilo koji se tiču bezbednosti, udobnosti i razonode. Brzina prenosa podataka može da dostigne i 100Mbit/s, što prevazilazi standardnu *Eternet* vezu. Takođe, *BroadR-Reach* standard omogućava istovremeno slanje i primanje podataka preko iste neobložene upredene parice (engl. *full-duplex*). Propusni opseg *BroadR-Reach* tehnologije iznosi 33.3 *MHz*, što je veliki napredak u odnosu na prethodno korišćene tehnologije. [3]

Jedan on najvažnijih alata koji se ovde koristi je *ADAS*, i on može biti pasivan i aktivan. Pasivan *ADAS* pruža vozačima samo upozorenja ukoliko, na primer, izađu van svoje trake. Sa druge strane, aktivan *ADAS* ne samo da upozorava vozača, već i preuzima kontrolu nad vozilom u slučaju potencijalne nezgode (vraćanje vozila u svoju traku, automatsko kočenje itd.). Aktivni *ADAS* takođe može samostalno da parkira vozilo.

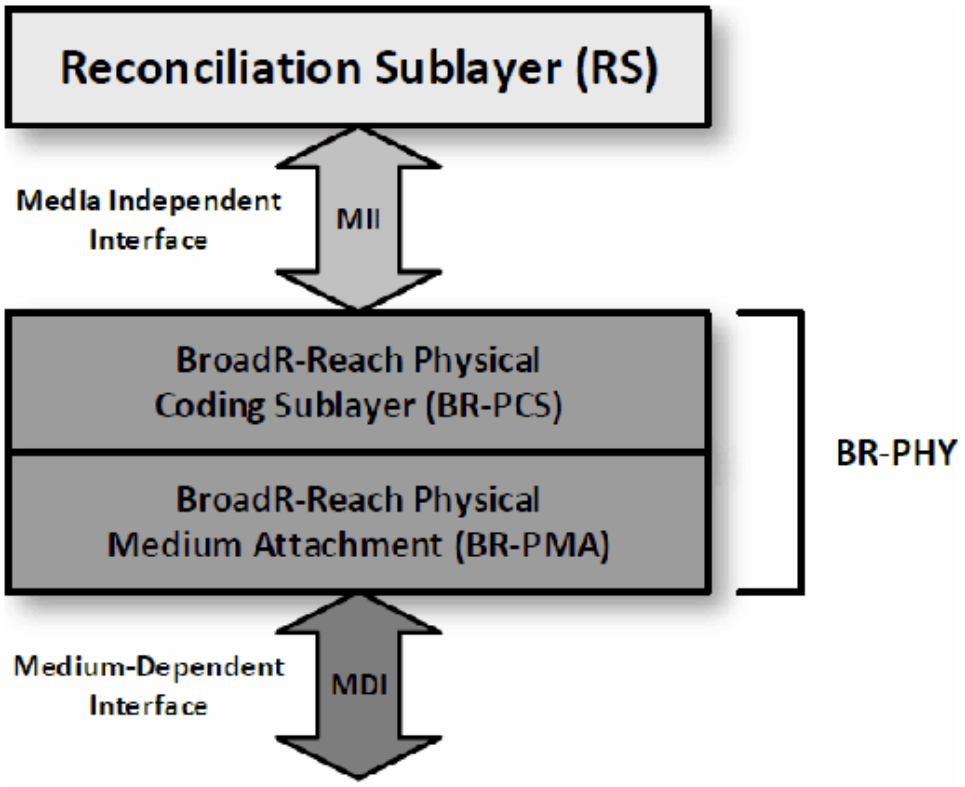
Sa brzinom od 100Mbit/s, *BroadR-Reach* tehnologija je najpogodnija za pasivne *ADAS* primene, jer aktivni *ADAS* zahteva video bez gubitaka kako bi algoritmi za prepoznavanje oblika mogli neometano funkcionisati. [4]

2.1.1 Arhitektura fizičkog sloja *BroadR-Reach*-a

Novije tehnologije *Eternet* fizičkog sloja delom su dizajnirane pozajmljivanjem koncepata iz prethodnih verzija, dok u nekim slučajevima to pozajmljivanje uključuje mrežne elemente koji čak ni ne pripadaju *Eternet* tehnologiji. Korišćenjem rešenja koja već postoje i za koja se zna da pravilno funkcionišu, inženjeri mogu da se fokusiraju na oblasti koje su nove, ili na one koje zahtevaju promene ili poboljšanja. Ovo znači da se do novih rešenja može doći brže, efikasnije i po nižoj ceni.

Ovaj pristup je korišćen pri razvijanju *BroadR-Reach*-a, koji je u velikoj meri baziran na upredenim paricama gigabit *Eternet* (prenos od 1 Gbit/s) fizičkog sloja. Ne samo da su mnoge tehničke karakteristike u *BroadR-Reach*-u pozajmljene od gigabit *Eterneta*, već se u dokumentaciji standarda *BroadR-Reach* ne navodi kao nova tehnologija, već gde se i na koji način razlikuje od gigabit *Eterneta*. Ovo samo pokazuje koliko su *BroadR-Reach* i gigabit *Eternet* slični.

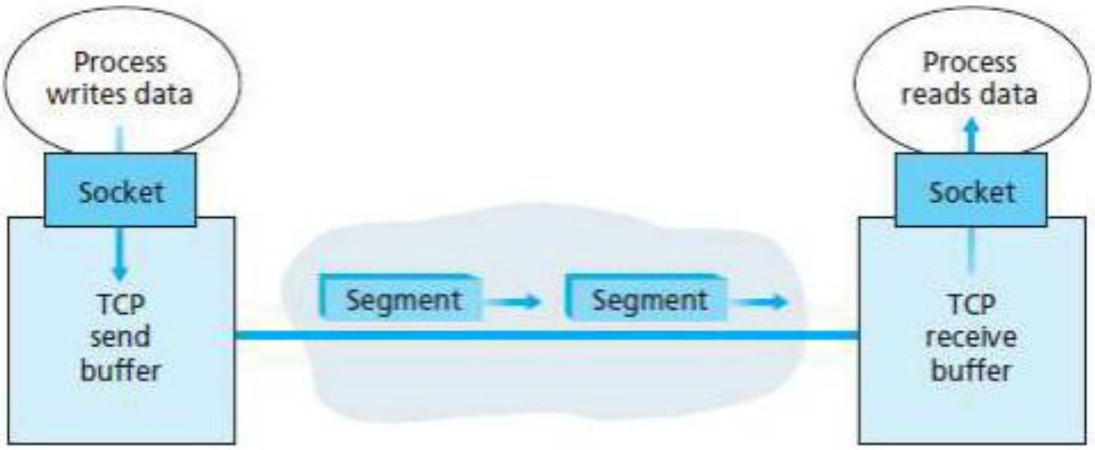
Kao što je već ranije navedeno, *BroadR-Reach* i gigabit *Eternet* praktično koriste isti fizički sloj, međutim sa jednom bitnom razlikom, a to je da sa nivoom veze komuniciraju preko drugačije sprege, a to je zbog razlike u brzinama (gigabit *Eternet* radi na 1000 Mbit/s, a *BroadR-Reach* radi na 100 Mbit/s). [5]



Slika 2.3 Arhitektura fizičkog sloja *BroadR-Reach*-a [6]

2.2 *TCP* protokol

TCP predstavlja protokol transportnog nivoa. Ovaj protokol omogućava istovremeno dvosmernu pouzdanu komunikaciju između klijenta i servera. Komunikacija je realizovana u vidu veze koja se uspostavlja pomoću metode rukovanja (engl. *handshaking*).

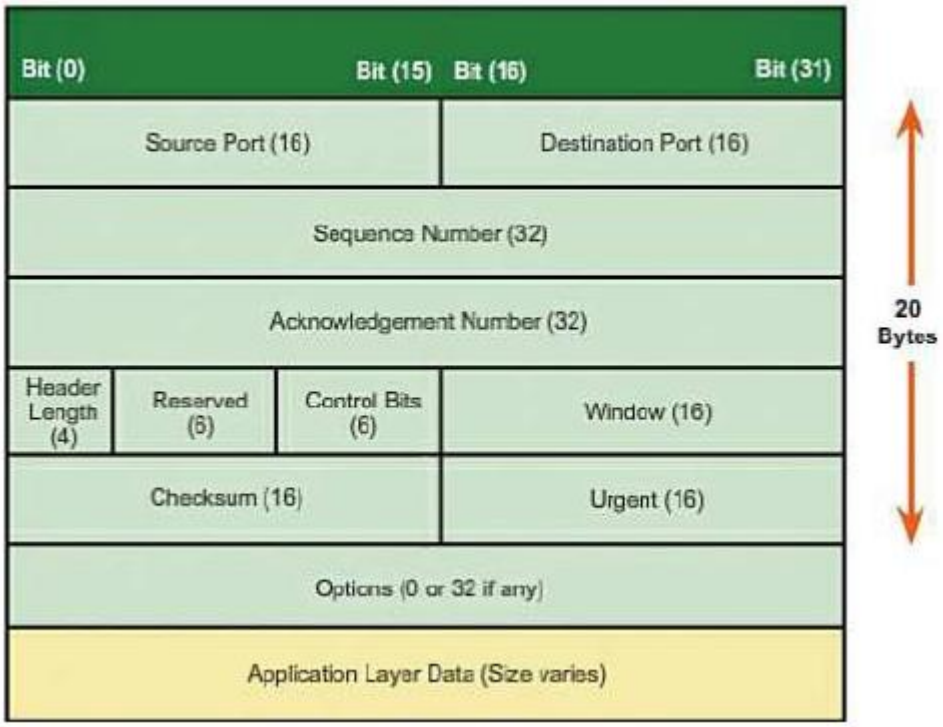


Slika 2.4 Segmentacija [7]

TCP koristi segment za jedinicu prenosa (jedinice podataka transportnog sloja). Proces podele originalne poruke aplikativnog nivoa na segmente naziva se segmentacija. *TCP* omogućava praćenje poslatih paketa, kontrolu toka komunikacije, kao i redosleda pristiglih segmenata.

TCP definiše uslugu pouzdane isporuke toka (engl. *stream*) korisničkih podataka. Osobine *TCP*-a su:

- Obavlja kontrolu toka podataka, *TCP* obezbeđuje komunikaciju sistema različitih brzina.
- Osnovna jedinica prenosa *TCP*-a je segment podataka. Segmenti se koriste za prenos upravljačke informacije (npr. poruke za uspostavu i raskid veze), ili za prenos podataka.
- Format segmenta je izabran tako da je moguće potvrđivanje podataka iz jednog smera, njihovim uključivanjem u zaglavlje segmenata koji se šalju u drugom smeru.
- Kontrola toka je realizovana tako što prijemnik oglašava količinu podataka koju je spreman da primi.
- *TCP* takođe podržava poruke van opsega (engl. *out of band*), koje služe za slanje urgentnih podataka i za forsiranje isporuke korišćenjem gurajućih (engl. *push*) podataka.



Slika 2.5 Izgled *TCP* segmenta [8]

Na slici 2.5 je dat prikaz *TCP* segmenta, koji se sastoji iz zaglavlja i dela u kome se nalaze podaci. Zaglavlje se sastoji iz:

• *Source port* - izvorišni port (port pošiljaoca). Port ima ulogu u identifikaciji aplikacije.

• *Destination port* - odredišni port (port primaoca).

• *Sequence number* - broj prvog bajta segmenta u okviru toka podataka. Broj sekvence omogućava praćenje toka podataka. Inicijalni broj sekvence se bira nasumično, kako bi se otklonila mogućnost interferencije između različitih veza.

  o *syn = 1* - Inicijalni broj sekvence.

  o *syn = 0* - Akumulirani broj sekvence.

• *Acknowledgement number* - broj sekvence narednog segmenta koji se očekuje. Segmenti koji pristignu van redosleda, u zavisnosti od implementacije, mogu se odbaciti ili čuvati.

• *Header Length* - dužina zaglavlja.

• *Reserved* - rezervisano za buduću upotrebu.

• *Control bits* - ukazuje na funkciju i namenu segmenta.

  o *ack* - ukazuje na validnost vrednosti potvrde (engl. *acknowledgment*).

  o *syn*, *rst* i *fin* - omogućavaju uspostavljanje i prekid veze.

  o *psh* - ukazuje prijemnoj strani da se momentalno pošalju podaci višem sloju.

  o *urg* - ukazuje na postojanje urgentnih podataka u segmentu.

• *Window* - maksimalan broj bajtova koje je moguće poslati, a da prethodno nije potvrđen njihov prijem.

• *Checksum* - koristi se za proveru da li se desila greška poruke nad zaglavljem i podacima prilikom prenosa.

• *Urgent* - lokacija poslednjeg bajta koji je markiran kao urgentan.

• *Options* - dodatne opcije.

• *Application data* - podaci viših slojeva.

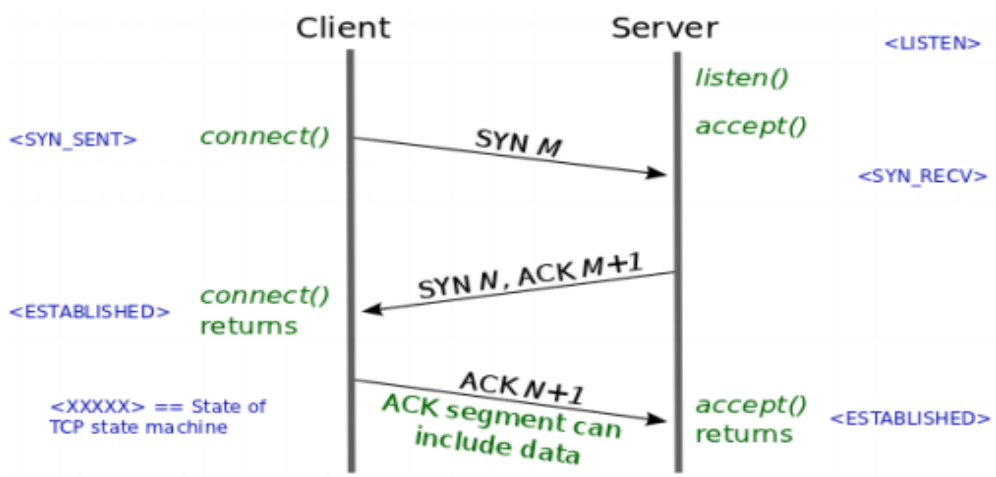
2.2.1 Uspostavljanje veze

Komunikacija klijenta i servera preko *TCP* protokola zahteva uspostavu veze koja se ostvaruje tako što se između predajne i prijemne strane iz tri puta razmene poruke sa podešenih odgovarajućim kontrolnim bitima (*TCP three-way handshake*):

• Predajna strana *A* šalje poruku sa podešenim kontrolnim bitom *SYN = 1*, ostali su podešeni na 0, pri čemu nasumično odabere redni broj segmenta (*SEQa*).

• Prijemna strana *B* odgovara porukom sa kontrolnim bitima *SYN* i *ACK = 1*, takođe nasumično bira broj segmenta (*SEQb*), a za *ACK* uzima broj *ACK = (SEQa) + 1*. Na ovaj način je uspostavljena veza na liniji od predajne ka prijemnoj strani.

• Slanjem poruke sa podešenim kontrolnim bitom *SYN = 1* od prijemne strane, ona zahteva da predajna strana potvrdi uspostavljanje veze od prijemne ka predajnoj strani. Predajna strana to čini slanjem poruke sa podešenim kontrolnim bitom *ACK = 1* i uzima vrednost *ACK* broja *ACK = (SEQb) + 1*. Na ovaj način je uspostavljena veza između klijenta i servera.

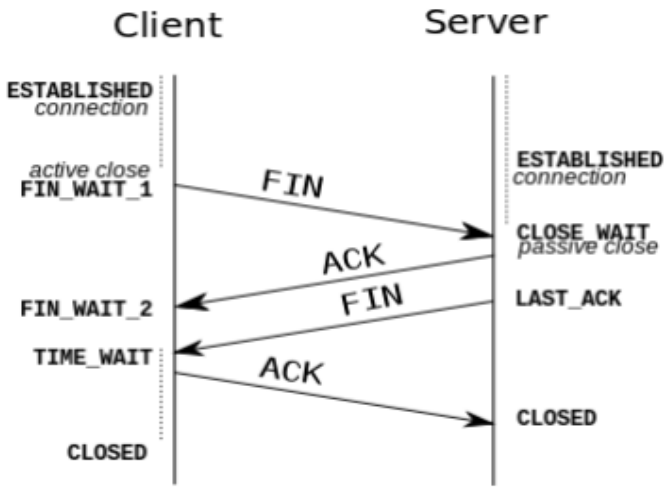


Slika 2.6 *TCP* uspostava veze [9]

2.2.2 Prekid veze

Prekid veze se ostvaruje u četiri koraka sledećim redosledom:

1. Klijent inicira prekid klijent-server veze.
2. Server potvrđuje zahtev za prekid klijent-server veze.
3. Server šalje zahtev za prekid server-klijent veze.
4. Klijent odgovara na zahtev za prekid server-klijent veze. [10]



Slika 2.7 *TCP* prekid veze [11]

2.3 *IPv6* protokol

Internet protokol verzija 6 (*IPv6*) je protokol sloja mreže, naslednik internet protokola verzije 4 (*IPv4*), osmišljen od strane *IETF*-a (engl. *Internet Engineering Task Force*).

2.3.1 *IPv6* adresa

IPv6 je novi (ali ne još široko korišten) standardni internet protokol, gde su adrese 128 bita široke, što bi, čak i sa velikim dodelama netblokova, trebalo da zadovolji blisku budućnost. Teoretski, postojalo bi tačno 2^{128}, ili 3.403×10^{38} unikatnih adresa domaćinskih sprega. Kada bi zemlja bila kompletno sačinjena od blokova veličine $1cm^3$, onda bi mogla da se dodeli jedinstvena

adresa svakom bloku u 300 miliona planeta dimenzija zemlje. Ovaj veliki prostor za adrese teško da će ikada biti potpuno popunjen. Adresa verzije 6 se piše kao osam četvorocifrenih heksadecimalnih brojeva (8 puta po 16 bitova) odvojenih dvotačkama. Jedan niz nula po adresi može da se izostavi, pa je 1080::800:0:417A isto što i 1080:0:0:0:0:800:0:417A. Globalne adrese koje se šalju ka jednom odredištu se sastoje iz dva dela: 64-bitni deo za rutiranje i 64-bitni identifikator domaćina.

2.3.2 Struktura *IPv6* paketa

Jedan *IPv6* paket ima sledeću opštu formu:

Slika 2.8 Izgled *IPv6* paketa [12]

Jedino potrebno zaglavlje se odnosi na *IPv6* zaglavlje. Ovo je fiksna veličina sa dužinom od 40 bajtova, u poređenju sa 20 bajtova pomoćnog dela *IPv4*. Sledeća produžena zaglavlja su definisana kao:

• Zaglavlje *Hop-by-Hop* opcija: definišu specijalne opcije koje zahtevaju *hop-by-hop* procesiranje.

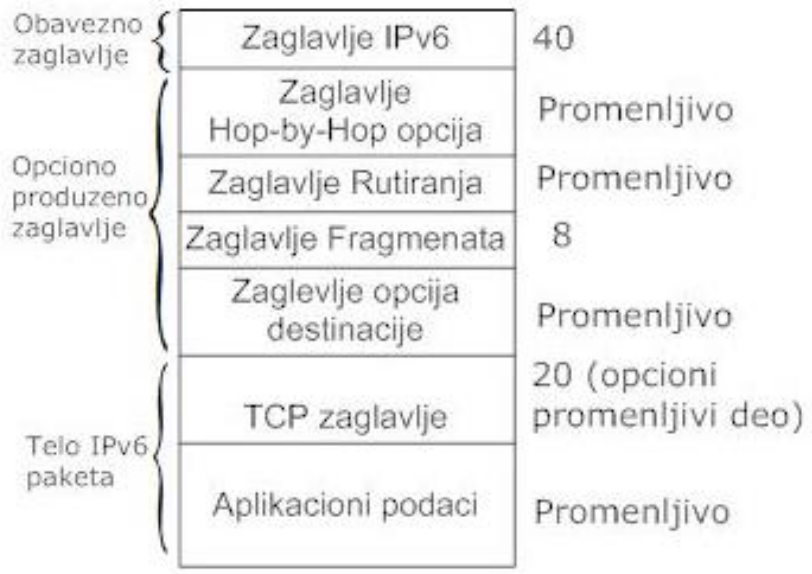
• Zaglavlje rutiranja: obezbeđuje prošireno rutiranje, slično izvorišnom rutiranju u *IPv4*.

• Zaglavlje fragmenata: sadrži informacije o fragmentaciji i ponovnom sklapanju.

• Zaglavlje autentičnosti: obezbeđuje integritet i autentičnost svakog paketa.

• Zaglavlje enkapsulacije sigurnosti podatka: obezbeđuje privatnost.

• Zaglavlje odredišnih opcija: sadrži fakultativne informacije koje će ispitati odredišni čvor.



Slika 2.9 Primer *IPv6* paketa sa uključenim zaglavljima [13]

IPv6 zaglavlje i svako produženo zaglavlje sadrže polje *sledeće zaglavlje*. Ovo polje identifikuje tip sledećeg zaglavlja. Ako je sledeće zaglavlje jedno od produženih zaglavlja, onda ovo polje sadrži identifikator tipa tog zaglavlja. U protivnom, ovo polje sadrži identifikator protokola za protokol višeg sloja koji koristi *IPv6* (obično protokol transportnog sloja), koriste se iste veličine kao i polja *IPv4* protokola. Na slici 2.9, protokol višeg sloja je *TCP*, tako da se podaci viših slojeva koje nosi *IPv6* paket sadrže od zaglavlja *TCP*-a koje prati blok aplikacionih podataka. [14]

2.4 Asimetrična RSA enkripcija, osobine i primena

Enkripcija (engl. *encryption*) ili šifrovanje je proces u kriptografiji kojim se vrši izmena podataka tako da se podaci, ili poruke, učine nečitljivim za osobe koje ne poseduju određeno znanje (ključ). Na taj način se dobija šifrovana informacija. Da bi ovi podaci postali razumljivi i upotrebljivi, potrebno je da se dešifruju. Dešifrovanje se vrši procesom suprotnim od enkripcije koji se naziva dekripcija (engl. *decryption*).

2.4.1 Elementi enkripcije

Svi sistemi enkripcije imaju u svojoj osnovi sledeće zajedničke elemente:

• *Algoritam*: funkcija, obično sa jakom matematičkom osnovom, koja obavlja zadatak enkripcije podataka.

• *Ključevi*: koriste se zajedno sa algoritmima enkripcije i određuju način na koji su podaci šifrovani.

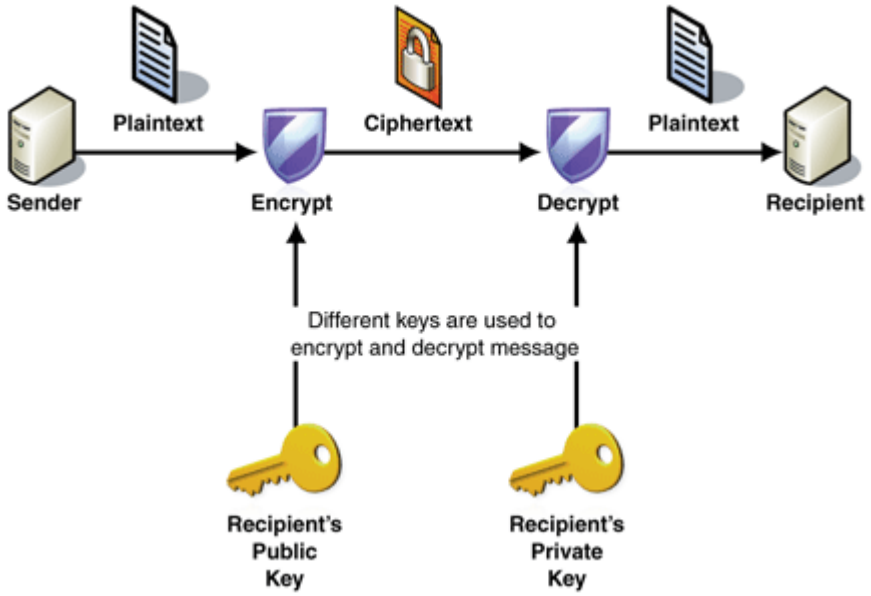
• *Dužina ključa*: enkripcioni ključevi imaju određenu dužinu u zavisnosti od toga koji enkripcioni sistemi se koriste. Dužina se meri brojem bitova, a što su duži ključevi, teži su za oštećenje sistema enkripcije.

• *Otvoren tekst* (engl. *plaintext*): informacije koje želimo da šifrujemo.

• *Šifrovan tekst* (engl. *ciphertext*): informacije nakon šifrovanja. [15]

2.4.2 Osobine RSA enkripcije

Postoje dve osnovne vrste enkripcije, simetrična i asimetrična enkripcija. Kod simetrične enkripcije se i za šifrovanje i za dešifrovanje koristi ista šifra (ključ), i samim tim što obe strane koriste isti ključ i za enkripciju i za dekripciju proboj sistema je znatno olakšan. Kod asimetrične enkripcije postoji poseban ključ samo za šifrovanje i drugi koji služi samo za dešifrovanje. Ova dva ključa nazivaju se još **tajni** i **javni** ključevi. Tajni ključ se dodeljuje onda kada se vrši enkripcija i na osnovu njega se generiše javni ključ, koji koristi strana koja treba da pročita podatke. *RSA* enkripcija (akronim od Rivest - Shamir - Adleman) predstavlja jedan od najčešće korištenih algoritama za asimetričnu enkripciju, razvijen od strane trojice stručnjaka sa *MIT*-a 1977. godine i u sledećem poglavlju će detaljno biti opisan način na koji funkcioniše.



Slika 2.10 Prikaz *RSA* enkripcije i dekripcije [16]

2.4.3 Algoritam RSA enkripcije

Kao što je već rečeno, kod *RSA* enkripcije se koriste dva ključa, tajni i javni. Javni ključ je dostupan svima i on se koristi za enkriptovanje poruka. Poruke koje su enkriptovane javnim ključem mogu se dekriptovati jedino tajnim ključem. Ovaj par ključeva se generiše na sledeći način:

1) Biraju se dva nasumična velika prosta broja $\boldsymbol{p}$ i $\boldsymbol{q}$.
2) Računa se njihov proizvod $\boldsymbol{n} = \boldsymbol{p} * \boldsymbol{q}$.
3) Nakon toga se računa vrednost *Ojlerove fi funkcije* $\boldsymbol{\phi(n)} = (\boldsymbol{p} - \mathbf{1}) * (\boldsymbol{q} - \mathbf{1})$. U teoriji brojeva, *Ojlerova fi funkcija* $\boldsymbol{\phi(n)}$, za pozitivne cele brojeve *n*, je definisana kao broj pozitivnih celih brojeva manjih ili jednakih sa *n*, koji su uzajamno prosti sa *n*. [17]
4) Bira se ceo broj $\boldsymbol{e}$ takav da zadovoljava uslov $\mathbf{1} < \boldsymbol{e} < \boldsymbol{\phi(n)}$ i $\boldsymbol{e}$ i $\boldsymbol{\phi(n)}$ ne smeju imati zajednički faktor osim broja **1.** Par $\boldsymbol{e}$ i $\boldsymbol{n}$ predstavljaju javni ključ.
5) Na kraju se računa $\boldsymbol{d}$ tako da zadovoljava uslov $\boldsymbol{d} * \boldsymbol{e} = \mathbf{1} + \boldsymbol{k} * \boldsymbol{\phi(n)}$ za neki ceo broj $\boldsymbol{k}$. Par $\boldsymbol{d}$ i $\boldsymbol{n}$ se čuvaju kao tajni ključ.

Formula koja se koristi za enkripciju poruke $\boldsymbol{m}$ je $\boldsymbol{c} = \boldsymbol{m^e} * (\boldsymbol{mod(n)})$, a da bi se iz enkriptovane poruke $\boldsymbol{c}$ došlo do originalne poruke $\boldsymbol{m}$, neophodno je uraditi sledeće $\boldsymbol{m} = \boldsymbol{c^d} * (\boldsymbol{mod(n)})$. [18]

Kod *RSA* enkripcije se postavlja pitanje zašto se koriste baš prosti brojevi? Zašto ne bilo koja dva velika broja? Odgovor leži u tome da je današnjim računarima veoma lako pomnožiti dva

velika prosta broja i naći broj $\boldsymbol{n}$, ali ne postoji način na koji bi se efikasno odradila inverzna operacija kako bi se došlo do faktora $\boldsymbol{p}$ i $\boldsymbol{q}$ jer to direktno proizilazi iz osnovne teoreme algebre. Ona kaže da se svaki složeni broj, koji je veći od 1, može napisati na tačno jedan način kao proizvod prostih brojeva. Sa malim brojevima to je lako, npr. $\mathbf{15 = 3 * 5}$ ili $\mathbf{255 = 3 * 5 * 17}$, ali kada je dat problem koji izgleda kao $\boldsymbol{p * q = 6700283}$, onda je izuzetno teško zaključiti da je $\boldsymbol{p = 1889}$ i da je $\boldsymbol{q = 3547}$. Do današnjeg dana nije osmišljen algoritam koji ovo omogućava i ne preostaje ništa drugo no pokušati sa *brute-force* pretragom (sistematično nabrajanje svih mogućih kandidata za rešavanje problema i proveravanje da li svaki kandidat zadovoljava problem). Ako su brojevi $\boldsymbol{p}$ i $\boldsymbol{q}$ dovoljno veliki, srednje vreme *brute-force* pretrage može da iznosi nekoliko desetina, pa čak i nekoliko stotina godina, što svakako ide u prilog robusnosti i sigurnosti *RSA* algoritma.

3. Koncept rešenja

U ovom poglavlju biće pojašnjen koncept rešenja. Rešenje prikazano u ovom radu se sastoji iz dva dela - prvi deo rešenja predstavlja klijentsku stranu i ona se izvršava na računaru, dok je drugi deo rešenja serverska strana koja se izvršava na namenskoj platformi. Obe strane su realizovane u programskom jeziku *C*. Radno okruženje koje je korišćeno prilikom izrade klijentske strane je *Visual Studio 2017 Professional*. Samo okruženje poseduje sve neophodne biblioteke i podešavanja kako bi se uspostavila komunikacija sa namenskom platformom. Operativni sistem koji se nalazi na računaru je *Windows 10*. Na namenskoj platformi se nalazi *RTOS VxWorks Wind River* operativni sistem sa svim neophodnim bibliotekama za razmenu podataka sa klijentskom stranom. Serverska strana je napisana u *Notepad++* alatu. Dodatak rešenju jeste grafička korisnička sprega na klijentskoj strani koje je napisao u *Python* jeziku, koristeći *TkInter* paket za izradu *GUI* elemenata.

SoC koji se nalazi na namenskoj platformi i koji omogućava realizaciju rešenja sa serverske strane je *Altera Cyclone V Soc* sa integrisanim *ARM* procesorom.

Slika 3.1 *Altera Cyclone* pete generacije [19]

Ideja zadatka je da se podaci sa namenske platforme izvuku bezbedno, i da samo vlasnici tajnog *RSA* ključa mogu da ih pročitaju u njihovom originalnom obliku. Kada bi neko sa malicioznim tendencijama pokušao da izvuče podatke sa platforme oni bi bili nečitljivi i njihovo vraćanje u originalni oblik bi bilo praktično nemoguće kao što je i objašnjeno u poglavlju *RSA* enkripcije.

Na samom početku rešenja neophodno je izabrati tajni i javni ključ. Oni se biraju na slučajan način na klijentskoj strani što svakako uvećava robusnost.

Komunikacija između računara i namenske platforme započinje tako što se najpre pošalje zahtev sa klijentske strane za početak komunikacije. To je realizovano tako što obe strane unapred znaju odgovarajuću reč (npr. *Start*) i proverom primljene i unapred poznate reči server lako utvrdi ko pokušava da pristupi namenskoj platformi. Ako je primljena reč potvrđena kao validna server šalje klijentu *ACK* paket i rešenje je spremno za bezbedan prenos informacija preko mreže.

Sledeći korak je slanje javnog ključa serveru kako bi se mogla izvršiti enkripcija podataka, dok tajni ključ ostaje u posedu klijenta radi dekripcije. Poslati javni ključ je prethodno simetrično enkriptovan radi uvećanja bezbednosti sistema. Nakon toga sledi prikupljanje podataka iz željenog direktorijuma na namenskoj platformi, njihova enkripcija i slanje ka klijentu. Klijentska strana vrši dekripciju svakog paketa odmah nakon primanja i upisuje dekriptovani paket u datoteku. Rešenje je realizovano dinamički, što znači da broj datoteka koje se šalju ne mora biti unapred poznat već se vrši njihovo prebrojavanje nakon otvaranja željenog direktorijuma. Jedino što unapred mora biti poznato jeste ime željenog direktorijuma i putanja do njega.

Slika 3.2 Izgled klijentskog terminala

Na slici 3.2 su prikazani ispisi u terminalu na klijentskoj strani koji demonstriraju sve što je prethodno objašnjeno, uz dodatak da klijentska strana vodi računa o tome na koji način se prima svaka datoteka, koliko celih paketa treba da dobije i koje je veličine poslednji paket, o čemu će biti reči u idućem poglavlju.

4. Programsko rešenje

Kao što je naglašeno ranije, programsko rešenje je podeljeno u dva modula - na serversku stranu koja se nalazi na namenskoj platformi i na klijentsku stranu koja se nalazi na računaru. U ovom poglavlju biće opisana oba modula detaljno, ali sa akcentom na funkcionisanje jednog modula kao celine, nezavisno od drugog (iako svakako zavise jedan od drugog, međutim ovaj pristup omogućuje dublji uvid u ideju i realizaciju iste zarad uspešnijeg rešenja problema).

4.1 Klijentska strana

int main(void)	*Main* funkcija
void receiveFile(void)	Primanje datoteke
void decrypt(void)	Dekripcija
uint32_t prime(uint32_t pr)	Funkcija u kojoj se proverava da li je broj prost
void ce(void)	Funkcija u kojoj se računaju tajni i javni ključevi
uint32_t cd(uint32_t x)	Pomoćna funkcija za računanje tajnih ključeva

Tabela 4.1 Funkcije na klijentskoj strani

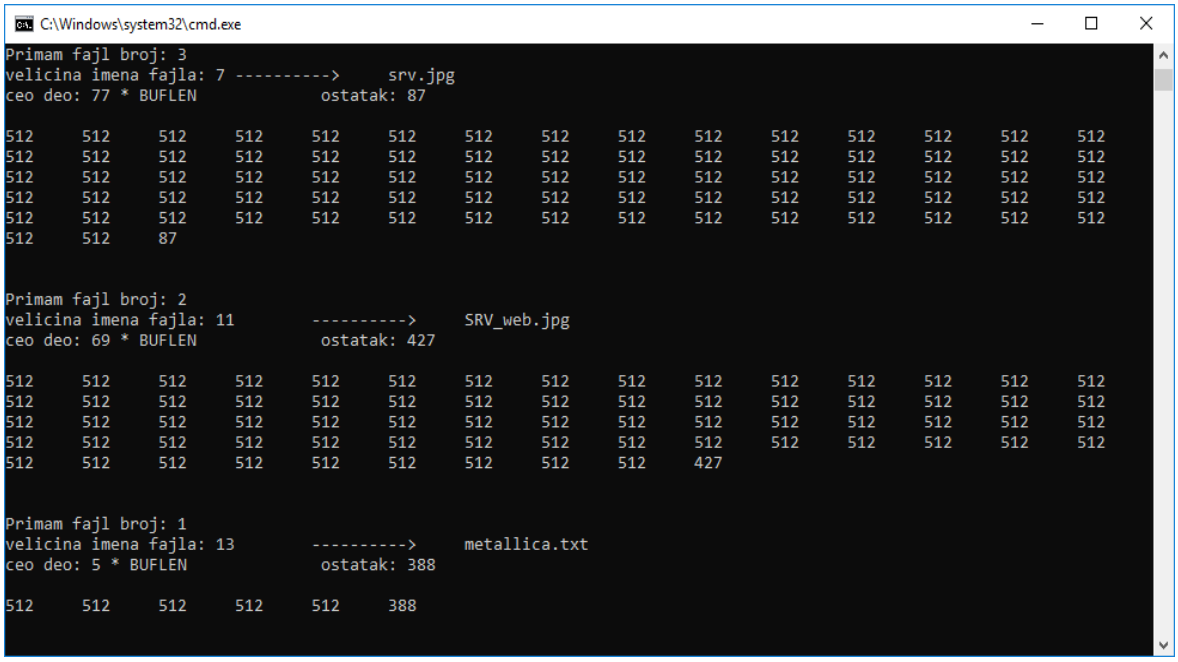
U narednim poglavljima biće detaljno pojašnjena svaka napisana funkcija na klijentskoj strani rešenja.

4.1.1 *Main* funkcija

Izvršavanje klijentske strane počinje u *int main(void)* funkciji u kojoj se na početku na slučajan način biraju dva prosta broja iz niza prostih brojeva. Nakon toga se pozivaju funkcije *void ce(void)* i *uint32_t cd(uint_t x)* zahvaljujući kojima se računaju tajni i javni ključevi pomoću *RSA* algoritma (poglavlje *2.5.3*). Kada su ključevi izračunati sledeći korak je pravljenje *socket-a* za komunikaciju i popunjavanje strukture *sockaddr_in6* odgovarajućim parametrima kako bi se uspešno uspostavila veza sa namenskom platformom. Sledi povezivanje sa serverskom stranom na platformi. Nakon uspešnog povezivanja sledi slanje inicijalne poruke, odnosno unapred dogovorene reči kako bi se potvrdila veza i kako bi usledilo slanje enkriptovanih podataka. Kada klijentska strana primi odobrenje od servera, sledeći korak je slanje javnog ključa (koji je prethodno simetrično enkriptovan, radi robusnosti rešenja) kako bi serverska strana mogla da enkriptuje željene datoteke. Zatim se na klijentskoj strani bira iz kog direktorijuma namenska platforma treba da pročita podatke i dobija se odgovor od platforme koliko će datoteka biti poslato. Poslednje što treba uraditi u *int main(void)* funkciji je proći kroz petlju onoliko puta koliko je platforma rekla da ima datoteka u željenom direktorijumu i za svaki prolaz pozvati funkciju *void receiveFile(void)*. Nakon izlaska iz petlje treba još zatvoriti *socket* i izvršavanje funkcije je gotovo.

4.1.2 Primanje datoteka

U funkciji *void receiveFile(void)*odvija se primanje jedne datoteke. Server najpre šalje ime i ekstenziju datoteke, pa zatim i veličinu. Na klijentskoj strani se otvara datoteka sa identičnim imenom i ekstenzijom. Nakon toga sledi računanje broja paketa koji će biti veličine *512* bajta (kao što je definisano u pretprocesorskoj direktivi) i računa se koliki će biti poslednji paket (najčešće je manji od *512* bajta). Sledi petlja u kojoj se vrši primanje paketa. Nakon primljenog paketa poziva se funkcija *void decrypt(void)* u kojoj se primljeni paket dekriptuje i odmah nakon toga upisuje u otvorenu datoteku. Sledi čišćenje svih bafera, provera da li je primljeni paket poslednji i ako jeste izlazi se iz *while* petlje. Nakon toga se zatvara otvorena datoteka i izlazi se iz *void receiveFile(void)* funkcije.



Slika 4.1 Izgled klijentskog terminala pri primanju datoteka

4.1.3 Dekripcija paketa

U funkciji *void decrypt(void)* vrši se dekripcija pristiglog paketa na osnovu algoritma koji je izložen u poglavlju *2.5.3*. Ova funkcija će biti pozvana onoliko puta koliko paketa stigne na klijentsku stranu.

4.1.4 Provera da li je prosleđeni broj prost

Zadatak funkcije *uint32_t prime(uint32_t pr)* je da proveri da li je broj koji je prosleđen kao argument funkcije prost ili ne. Iako jednostavna po svojoj kompleksnosti, ima ključnu ulogu, jer kao što je već rečeno prosti brojevi su od suštinskog značaja za *RSA* algoritam.

4.1.5 Generisanje parova tajnih i javnih ključeva

Uloga funkcija *void ce(void)* i *uint32_t cd(uint32_t x)* jeste generisanje parova tajnih i javnih ključeva. U funkciji *void ce(void)* se bira vrednost ključa za enkriptovanje podataka, a pomoću funkcije *uint32_t cd(uint32_t x)* se bira vrednost tajnog ključa, odnosno ključa za dekripciju. Brojevi $\boldsymbol{e}$ i $\boldsymbol{d}$ koji se koriste za enkripciju i dekripciju podataka moraju biti prosti, pa se u rešenju

za gornju granicu provere postavlja prvih *100* prostih brojeva (ovo ne mora da bude slučaj, tako je izabrano radi preglednosti rešenja).

4.2 Serverska strana

FUNC(void, RTE_CTCDETHCOM_APPL_CODE) REthComInit(void)	Funkcija koja će se pozvati samo jednom pri pokretanju serverske strane
FUNC(void, RTE_CTCDETHCOM_APPL_CODE) REthComCyclic(void)	Funkcija koja će se pozivati u pozadini nakon određenog vremena
static void backgroundTask(void)	*Callback* funkcija u kojoj se nalazi veći deo logike serverske strane rešenja
static void receivePublicKeys(void)	Primanje ključeva za enkripciju
static void sendFile(const char fs_name[])	Slanje datoteke
static int32_t numOfFiles(void)	Broj datoteka u direktorijumu
static void encrypt(void)	Enkripcija

Tabela 4.2 Funkcije na serverskoj strani

Serverska strana nema *main* funkciju, pa će taj deo rešenja izgledati malo drugačije. Sledi detaljan opis svake funkcije ponaosob na serverskoj strani.

4.2.1 Inicijalizacija namenske platforme

Funkcija *FUNC(void, RTE_CTCDETHCOM_APPL_CODE) REthComInit(void)* je funkcija koja se poziva samo jednom kada se namenska platforma upali. U njoj se nalazi poziv *callback* funkcije *static void backgroundTask()*. U funkciji je napravljen red poruka iz biblioteke *msgQLib.h* koji služi za kontrolu promene stanja. Tu se takođe nalazi funkcija pomoću koje se kreira *task* koji će pozvati *callback* funkciju.

4.2.2 Promena stanja sistema

Ovaj deo koda se poziva svaki put kada istekne predefinisano vreme (ciklična funkcija). U njoj će se poslati poruka koja će ući u prethodno napravljen red poruka, i na osnovu njene vrednosti se prelazi u odgovarajuće stanje, ili ako je vrednost poruke nepromenjena, stanje se ne menja.

4.2.3 Glavna logika serverske strane

U funkciji *static void backgroundTask(void)* je realizovana glavna logika serverske strane. Na početku je neophodno osposobiti ovu stranu rešenja za komunikaciju sa računarom. Kao i na klijentskoj strani, prva stvar koju treba uraditi jeste napraviti *socket*. Nakon toga se popunjava *sockaddr_in6* struktura, poziva se funkcija koja povezuje napravljeni *socket* sa željenom adresom. Sledeći je poziv funkcije koja čeka veze i na kraju sledi poziv funkcije koja prihvata komunikaciju sa uređajem koji pokušava da pristupi namenskoj platformi. Nakon što je komunikacija uspešno uspostavljena sledi utvrđivanje identiteta onoga ko pokušava da pristupi namenskoj platformi. Kao što je već rečeno, klijentska strana će poslati unapred dogovorenu reč kojom će se utvrditi da je reč o pouzdanom uređaju koji pokušava da izvuče resurse sa platforme. Ukoliko su reči identične komunikacija se nastavlja, u suprotnom dolazi do prekida. Sledi primanje javnog ključa za enkripciju, prebrojavanje datoteka u direktorijumu i slanje prebrojanog stanja. Zatim se poziva funkcija *static void sendFile(const char fs_name[])* za svaku prebrojanu datoteku u kojoj se ista i šalje. Poslednje što treba uraditi jeste poslati poruku u red poruka kako se više ne bi ulazilo u ovaj deo koda, odnosno kako bi došlo do promene stanja.

4.2.4 Primanje javnih ključeva

Ova funkcija služi za prijem javnog ključa koji se koristi za enkripciju željenih podataka. Primljeni ključ je simetrično enkriptovan radi podizanja bezbednosti sistema na viši nivo, stoga je neophodno odraditi dekripciju primljenih podataka. Ključ se prima iz dva dela jer kao što je već rečeno, par $\boldsymbol{e}$ i $\boldsymbol{n}$ predstavljaju javni ključ.

4.2.5 Slanje datoteka

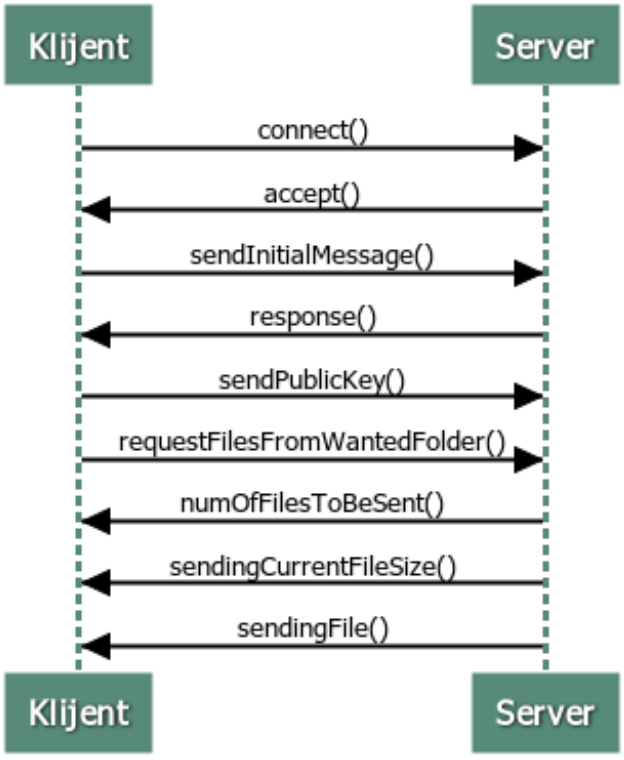
U *static void sendFile(const char fs_name[])* funkciji nalazi se glavna logika za enkripciju i slanje datoteka. Najpre se klijentu šalje ime i veličina imena datoteke koja će biti poslata. Zatim se na serverskoj strani čita datoteka radi utvrđivanja njene veličine, pa se i veličina šalje klijentu. Nakon toga se datoteka raspoređuje u pakete, vrši se enkripcija svakog paketa i tako enkriptovani paketi se šalju klijentu. Veličina paketa je *512* bajta, ali u suštini može biti bilo koja vrednost.

4.2.6 Prebrojavanje datoteka u željenom direktorijumu

U funkciji static int32_t numOfFiles(void) se vrši prebrojavanje datoteka u željenom direktorijumu. Povratna vrednost funkcije je broj datoteka koje će biti poslate klijentu.

4.2.7 Enkripcija paketa

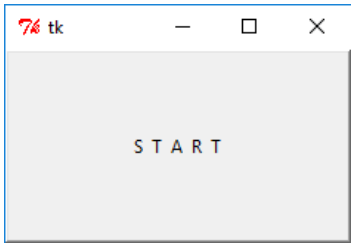
U funkciji *static void encrypt(void)* će biti izvršena enkripcija paketa po algoritmu iz poglavlja *2.5.3*.



Slika 4.2 *MSC* dijagram komunikacije između klijentske i serverske strane

4.3 Grafička korisnička sprega

Mala dopuna u radu jeste izrada jednostavne grafičke korisničke sprege koja olakšava korišćenje programske podrške na klijentskoj strani. Sastoji se od jednog dugmeta, i pritiskom na to dugme otvara se *.exe* datoteka koja pokreće klijentsku stranu programske podrške. Grafička korisnička sprega je napisano u jeziku *Python* koristeći *TkInter* paket za izradu *GUI* elemenata.

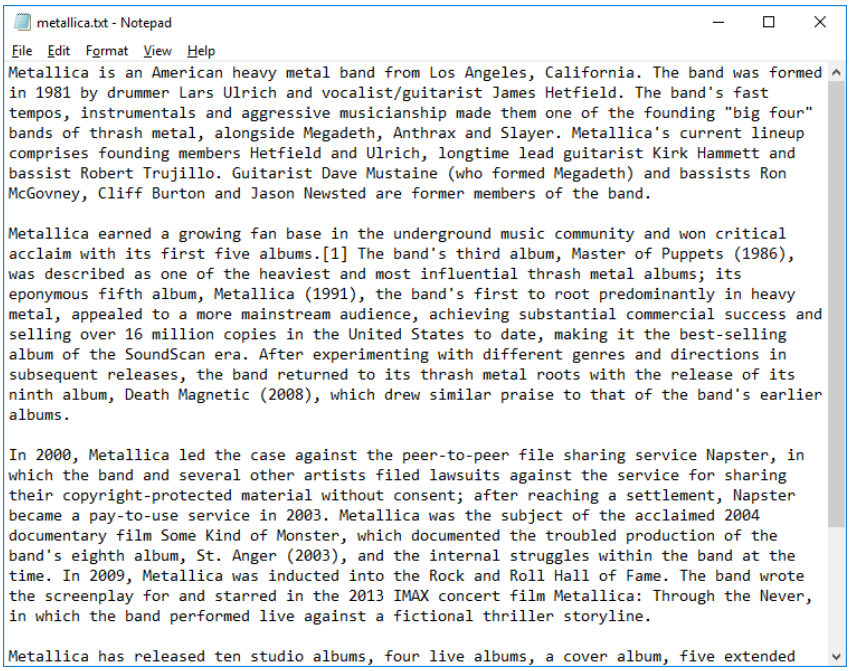


Slika 4.3 *GUI* za pokretanje *.exe* datoteke klijentske strane

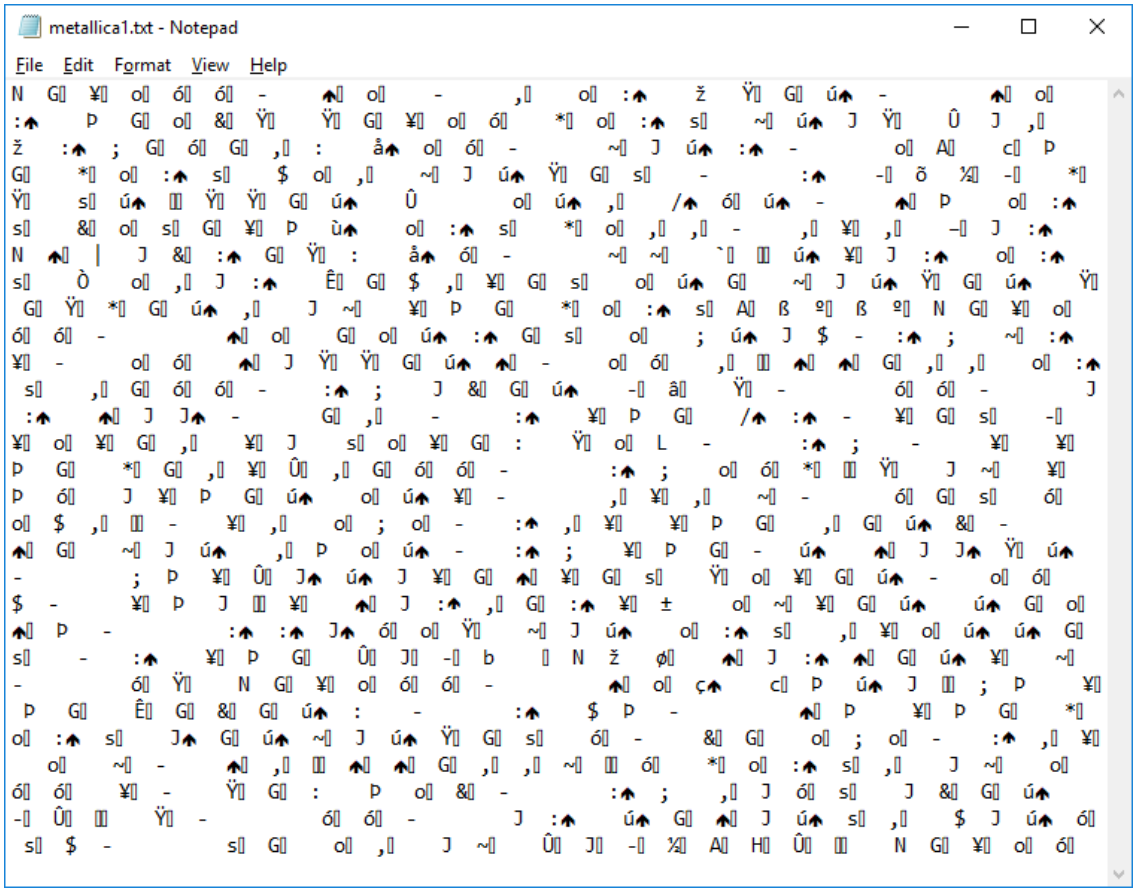
5. Testiranje i verifikacija

U okviru ovog poglavlja biće opisano testiranje i verifikacija rešenja. S obzirom da je akcenat u zadatku bio na enkripciji i dekripciji podataka, može se reći da je provera ispravnosti rešenja poprilično jednostavna, odnosno potrebno je utvrditi da li se enkriptovani podaci mogu vratiti u originalni oblik bez poznavanja tajnog ključa. Odgovor je da ne mogu. Jedini način za vraćanje originalnih podataka je *brute-force* pretraga tajnog ključa, ali kao što je već rečeno u poglavlju *2.5.3*, *brute-force* pretraga može da potraje i po nekoliko desetina godina, pa čak i više.

Pri testiranju rešenja isceniran je slučaj presretanja komunikacije, gde ona strana koja presreće može samo da vidi enkriptovane podatke.

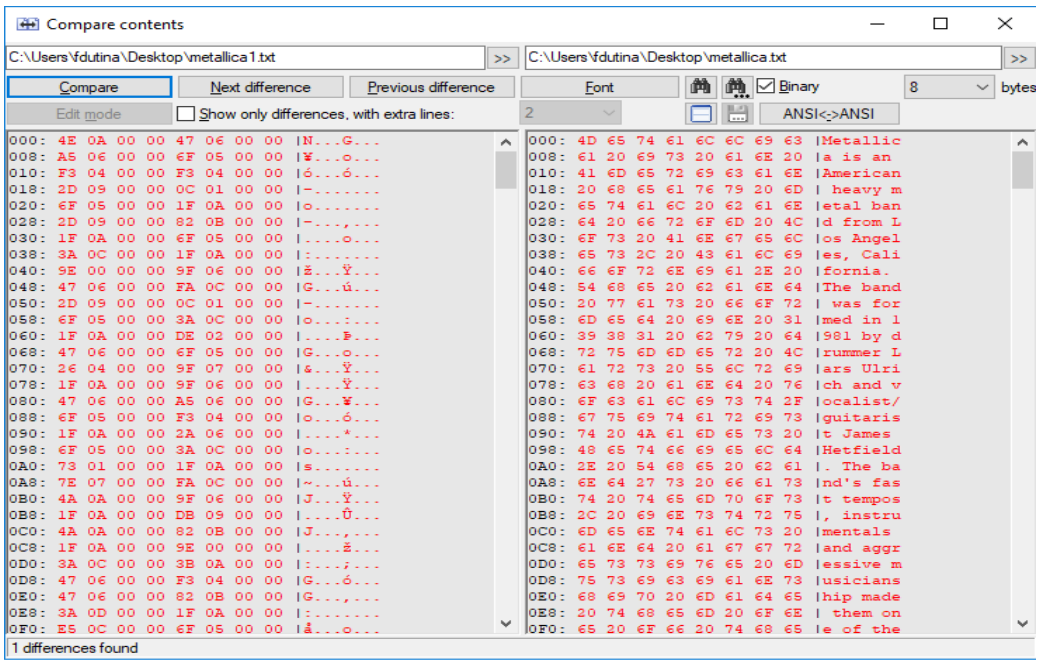


Slika 5.1 Prikaz originalnog teksta



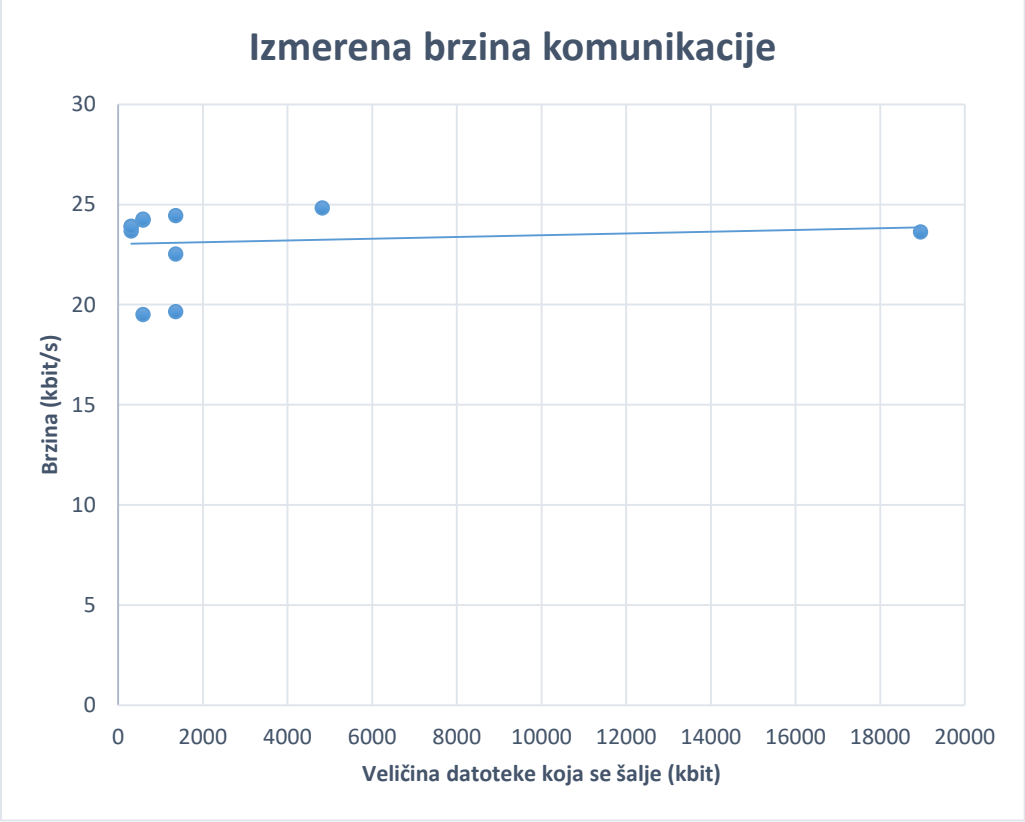
Slika 5.2 Prikaz enkriptovanog teksta

Kao što se vidi na slici 5.2, podaci koji su izvučeni sa namenske platforme bez prethodnog poznavanja tajnog ključa su bezvredni. Slika 5.3 pokazuje da razlika između enkriptovanog i originalnog teksta nije konzistentna, odnosno da je razlika između enkriptovanih i originalnih bajta na istim pozicijama u tekstu potpuno stohastička (što na primer nije slučaj kod algoritma enkripcije koji se zove *Cezarova šifra*, gde se svaki bajt enkriptovanog podatka pomera levo ili desno za unapred definisan broj mesta). Dakle kao što se i na datom primeru vidi, *RSA* algoritam enkripcije je praktično neprobojan.



Slika 5.3 Razlika između enkriptovanog i originalnog teksta

Rešenje je, pored tekstualnih datoteka testirano i na slikama različitih formata. Kao što je i očekivano, enkriptovana slika ne može biti reprodukovana, dok se dekriptovana reprodukuje bez ikakvih problema.



Slika 5.4 Izmerena brzina komunikacije

Testiranjem je utvrđeno da prosečna izmerena brzina komunikacije između namenske platforme i računara iznosi 23.48 kbit/s. Kao što se vidi na slici 5.4, brzina slanja podataka je konstantna, sa zanemarljivim fluktuacijama. Ova brzina nije na nivou brzina koje se koriste u svakodnevnoj upotrebi, međutim kao što je već poznato, sa namenske platforme će se izvlačiti dijagnostički podaci čija veličina se meri u kilobajtima, tako da je brzina komunikacije između računara i namenske platforme sasvim zadovoljavajuća za ovaj vid primene.

6. Zaključak

Zadatak ovog rada bio je dizajniranje i implementacija jednog rešenja programske podrške za bezbedan prenos podataka sa namenske platforme na računar. Akcenat je stavljen na enkripciju i dekripciju kako bi se očuvao integritet podataka koji se šalju preko mreže. Pri postupku testiranja programske podrške iscenirano je zlonamerno presretanje paketa radi čitanja podataka koji nisu javno dostupni i utvrđeno je da su presretnuti podaci bezvredni bez prethodnog poznavanja tajnog ključa *RSA* enkripcije.

Bezbednost rešenja može se podići na viši nivo upotrebom većih prostih brojeva pri izradi *RSA* tajnih i javnih ključeva, a pošto se u rešenju brojevi $\boldsymbol{e}$ i $\boldsymbol{d}$ biraju na slučajan način iz niza ključeva, prostim proširenjem niza iz kog se biraju ti ključevi može dovesti do veće sigurnosti. Takođe, ovo rešenje moguće je poboljšati dodatnim optimizacijama napisanog koda kako bi se povećala brzina obrade paketa, a samim tim i brzina razmene podataka između namenske platforme i računara. Još jedan od načina unapređenja programske podrške jeste korišćenje neke od komercijalnih biblioteka za enkripciju/dekripciju podataka.

Stalni napredak u automobilskoj industriji dovodi do toga da se u modernom vozilu svake sekunde stvara i obrađuje velika količina podataka, pa je neminovno da je potreba za čuvanjem i obezbeđivanjem podataka jedna od primarnih zadataka inženjera, kao što je i demonstrirano u ovom rešenju.

7. Reference

[1] Slika 2.1, http://eetimes.jp/ee/articles/1707/24/news069.html, jun 2018

[2] Slika 2.2, https://en.wikipedia.org/wiki/Twisted_pair, jun 2018

[3] *BroadR-Reach*, https://en.wikipedia.org/wiki/BroadR-Reach, jun 2018

[4] *BroadR-Reach*, http://www.electronicdesign.com/automotive/what-s-difference-between-broadr-reach-and-100base-t1, jul 2018

[5] *BroadR-Reach*, https://dl.rt-rk.com/?t=9de0775856cbcaff241d9cf388ca826b, avgust 2018

[6] Slika 2.3, https://dl.rt-rk.com/?t=9de0775856cbcaff241d9cf388ca826b, avgust 2018

[7] Slika 2.4, http://www.rt-rk.uns.ac.rs/predmeti/e2/orm-1-osnovi-računarskih-mreža-1, jun 2018

[8] Slika 2.5, http://www.rt-rk.uns.ac.rs/predmeti/e2/orm-1-osnovi-računarskih-mreža-1, jun 2018

[9] Slika 2.6, http://www.rt-rk.uns.ac.rs/predmeti/e2/orm-1-osnovi-računarskih-mreža-1, jun 2018

[10] *TCP*, http://www.rt-rk.uns.ac.rs/predmeti/e2/orm-1-osnovi-računarskih-mreža-1, jun 2018

[11] Slika 2.7, http://www.rt-rk.uns.ac.rs/predmeti/e2/orm-1-osnovi-računarskih-mreža-1, jun 2018

[12] Slika 2.8, https://sr.wikipedia.org/sr-el/IPv6, jun 2018

[13] Slika 2.9, https://sr.wikipedia.org/sr-el/IPv6, jun 2018

[14] *IPv6*, https://sr.wikipedia.org/sr-el/IPv6, jun 2018

[15] *RSA*, https://sr.wikipedia.org/wiki/Enkripcija, jun 2018

[16] Slika 2.10, https://www.safaribooksonline.com/library/view/openstack-cloud-security/9781782170983/ch04s03.html, jun 2018

[17] Ojlerova fi funkcija, https://sr.wikipedia.org/wiki/%D0%9E%D1%98%D0%BB%D0%B5%D1%80%D0%BE%D0%B2%D0%B0_%D1%84%D0%B8_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%98%D0%B0, jul 2018

[18] *RSA*, https://simple.wikipedia.org/wiki/RSA_algorithm, jun 2018

[19] *Altera Cyclone V*, https://www.altera.com/products/soc/ecosystem/system-on-modules.html, jun 2018