SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2208

Radni okvir za sigurnu komunikaciju unutar potrošačkog oblaka primjenom simetričnih algoritama kriptiranja

Alen Agić

Zagreb, 6.2011.

Veliko hvala upućujem prof. dr. sc. Siniši Srbljiću
na ukazanom povjerenju i prilici
za rad na vrlo zanimljivom projektu.
Posebno se zahvaljujem dipl. ing. Marin Šilić na
izuzetnom strpljenju, potpori i korisnim sugestijama
pri pisanju završnog rada.

Sadržaj

1. Uvod

Razvojem tehnologije u posljednjem stoljeću došlo je do primjene informacijskih tehnologija kao sredstva prijenosa podataka. Mnogi od tih podataka namijenjeni su samo određenoj zajednici ili određenom pojedincu. Kako bi se osigurao prijenos podataka između dvije točke, razvila se posebna znanstvena disciplina koja se bavi zaštitom podataka, kriptografija.

Još u doba Cezara, prije više od 2000 godina, ljudi su se bavili problemom zaštite poruka i raznih podataka od čitatelja kojima oni nisu bili namijenjeni.
Osnovni zadatak kriptografije je omogućiti dvjema osobama sigurnu komunikaciju putem nesigurnog komunikacijskog kanala, tako da treća osoba koja može nadzirati kanal ne može razumjeti poruke. Danas je primjena kriptografije široka, prvenstveno zbog mnogo nesigurnih komunikacijskih kanala poput telefonske linije, računalne mreže, te ostalih.

Usluge za društveno povezivanje primarno su fokusirane na stvaranje zajednice istomišljenika ili povezivanje određene skupine ljudi, prvenstveno putem interneta. Ove usluge također se mogu oblikovati kao poslovni sustavi, stvarajući raznorazne mogućnosti prilagodbe među korisnicima tog sustava. Društveno povezivanje i slične usluge danas su jedan od najpopularnijih načina komunikacije. Poslovne primjene takvih usluga su danas vrlo raširene, stoga se poslovni sustavi oblikuju sa posebnim oprezom. Primjena kriptografije je ovdje ključna za stvaranje zaštićenih prijenosa putem nesigurnih komunikacijskih kanala, te omogućavanje izvršavanja poslovnih transakcija bez ometanja vanjskih sudionika.

U ovom radu nastoji se pokazati jedan jednostavni sustav prilagođen korisniku, namijenjen za omogućavanje sigurne komunikacije između njih putem nesigurnog komunikacijskog kanala. U okviru ovog rada programski je ostvareno jednostavno grafičko korisničko sučelje za provjeru valjanosti korisnika, slanje poruka i njihovu enkripciju, te dekripciju.

2. Kriptiranje i kriptologija

Kriptografija je znanstvena disciplina koja se bavi proučavanjem metoda za zaštitu poruka tako da ih samo onaj kome su namijenjene može pročitati. Sama riječ kriptografija je grčkog podrijetla i mogla bi se doslovno prevesti kao tajnopis.

Osnovni zadatak kriptografije je omogućiti dvjema osobama, primatelju i pošiljatelju, komuniciranje preko nesigurnog komunikacijskog kanala kojeg može promatrati treća osoba, tako da može pristupiti poslanim podatcima, ali ih ne može razumjeti ili mijenjati.

Kriptologija se dijeli na sljedeće dvije cjeline:

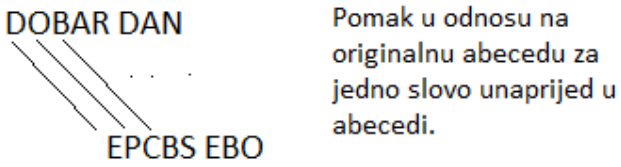
- Kriptoanalizu
- Kriptografija

Kriptoanaliza je znanstvena disciplina koja se bavi proučavanjem postupaka za čitanje skrivenih poruka bez poznavanja ključa šifriranja.
Kriptologija je grana znanosti koja obuhvaća kriptografiju i kriptoanalizu.
Kriptografski algoritam ili šifra je skup matematičkih funkcija, obično u paru, jedna za šifriranje i jedna za dešifriranje, koje u ovisnosti o ključu obavljaju preslikavanje elemenata izvornog teksta u kriptogram, te iz kriptograma u izvorni tekst. [4]

2.1. Simetrično kriptiranje

Osnovna karakteristika simetrične kriptografije je identičnost kriptografskih ključeva koji se koriste prilikom šifriranja i dešifriranja. Povijest simetrične kriptografije seže daleko u prošlost kada su se pojavili prvi načini kriptografskog prikrivanja podataka. Najpoznatije metode tada bile su Cezarova šifra, te primjena skitala.

Kod Cezarove metode tekst se šifrira na način da se svako slovo originalnog teksta zamijeni nekim drugim slovom abecede, a zamjensko šifrirano slovo određuje se tako da se abeceda pomakne za par mjesta udesno ili ulijevo, to jest za određen broj pomaka. Broj mjesta za koji se pomiče abeceda šifriranog teksta u odnosu na originalni tekst mijenjao se od poruke do poruke. Ključ kod ovog načina šifriranja predstavlja broj mjesta za koliko su abeceda originalnog i šifriranog teksta pomaknute.



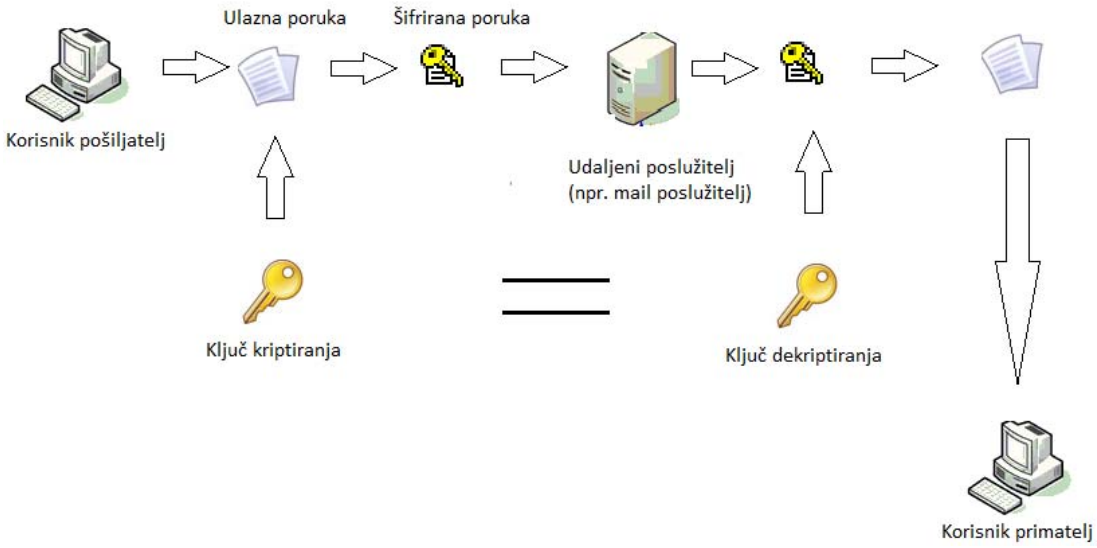
Slika 1. Primjer cezarove šifre

Današnji simetrični kriptografski algoritmi zasnovani su na informatičkoj tehnologiji i koriste kriptografske ključeve u digitalnom obliku. Njihova duljina najčešće se kreće od 64 bita do 256 bita.

Informacije o kriptografskom ključu za kriptiranje i dekriptiranje nužno je zadržati u tajnosti, odnosno unutar kruga pošiljatelja i primatelja šifrirane poruke. Komunikacijski kanal za razmjenu kriptografskih ključeva u izvornom obliku mora biti siguran, to jest mora onemogućiti trećoj strani da sazna bilo kakvu informaciju o ključevima. U suprotnom, šifriranje poruka ključevima rezultiralo bi otkrivanjem šifriranih podataka i ne bi imalo nikakvog smisla. Uobičajeno je da se kod simetričnih kriptografskih sustava potrebni kriptografski ključevi unaprijed razmjene između korisnika.

Lako je uočiti da upravljanje kriptografskim ključevima kod simetričnih kriptografskih sustava predstavlja veliki problem. S druge strane njihova prednost je brzina šifriranja koja je u odnosu na asimetrične kriptografske sustave od stotinu do tisuću puta veća. Povećanjem broja bitova ključa kod asimetričnih kriptografskih ključeva znatno se usporava vrijeme potrebno za postupak šifriranja

i dešifriranja. Iz tog razloga je s asimetričnim kriptografskim sustavima nemoguće izvesti linijsko ili *online* šifriranje, odnosno šifriranje podataka u realnom vremenu. Za to se još uvijek koriste simetrični kriptografski sustavi koji se s takvim izazovima trenutno mogu vrlo uspješno nositi. [4]



Slika 2. Primjer prijenosa podataka uz simetrično kriptiranje

2.1.1 Algoritam DES

1976. godine se prvi puta izabrao standardni kriptografski algoritam, zvan *Data encryption standard*, to jest skraćeno *DES*. Ovaj algoritam stvorio je IBM, te se u prvobitnom izdanju koristio uz primjenu ključeva duljine 56 bita.
Nekada se *DES* koristio za šifriranje *PIN*-ova, te transakcija preko bankomata. Također *DES* je do nedavno bio u širokoj upotrebi u civilnim satelitskim komunikacijama.

Danas se ovaj algoritam smatra nesigurnim zbog kratkog ključa koji se može razbiti grubom silom računanja ili metodom poput diferencijalne kriptografske analize. U današnje vrijeme *DES* je u potpunosti zamijenio *AES*, iako zbog vrlo velike brzine šifriranja, danas se primjenjuju neki od njegovih nasljednika poput *RC4, TripleDES, RC5, RC6*, itd. Jedan od popularnijih navedenih algoritama je bio *RC4*, čija se primjena pronašla u protokolima poput *SSL* i WEP,

no danas se više ne preporuča njegova primjena, zbog njegove jednostavnosti, što ga čini poprilično nesigurnim. Danas najprimjenjeniji *DES* algoritam je *TripleDES*, koji se još uvijek koristi putem elektroničkog poslovanja, ili za zaštitu lozinke u programima poput *Microsoft Outlooka 2007*.

Dužina ključa kriptiranja je 64 bita, od kojih 8 otpada na proveru pariteta, tako da je efektivna dužina ključa 56 bita. *DES* kriptiranje i dekriptiranje se provodi u nekoliko koraka. Prvo se bitovi ulaznog bloka dužine 64 bita permutiraju nekom permutacijom, te ju nazivamo početnom permutacijom (eng. *Initial permutation)*. Tada se ulazni blok podijeli na dva dijela po 32 bita, lijevi *L0* i desni dio *R0*. Nad desnim blokom se obavlja funkcija *F(Ri,Ki)*, odnosno $F(Ri,K_{16-i+1})$ kod dekriptiranja, gde je *Ri* desnih 32 bita, a *Ki* je 48 bitni ključ koji se generira iz zadanog tajnog ključa kriptiranja. Vrijednost dobivena operacijom *XOR* između vrijednosti funkcije *F* i lijevih 32 bita podataka, postaje R_{i+1}, tj. desnih 32 bita za sljedeći korak iteracije. L_{i+1} za slijedeći korak je *Ri*. Nakon 16 takvih koraka blokovi se zamijenjuju te se spajaju i obavlja se konačna permutacija koja je inverzna početnoj, tj. IP^{-1}. Dobivenih 64 bita su kriptirani blokovi. Budući da se nakon dvije uzastopne operacije *XOR* sa istim brojem dobije početna vrijednost, tj. *a = (aXORb)XOR b*, postupak dekriptiranja može se provesti tako da se operacije obavljaju obrnutim redoslijedom. Zbog simetričnosti algoritma to se postiže tako da se kriptirani blok pusti kroz isti algoritam, te se umjesto ključa *Ki* u i-tom koraku upotrijebi ključ K_{16-i+1}. Postupak generiranja šestnaest 48 bitnih ključeva od zadanog, tajnog ključa provodi se u nekoliko koraka. Prvo se pomoću zadane tablice permutacije iz ključa generiraju dva bloka po 28 bita. Zatim se svaki blok rotira u lijevo za određeni broj bita, u zavisnosti o kojem je koraku riječ, te se iz nastalih blokova pomoću tablicom zadane permutacije generira ključ *Ki*, gde je *i* broj koraka. Funkcija enkripcije *F* je zapravo najkritičniji dio algoritma, tj. upravo zbog njene kompleksnosti ne postoji način probijanja *DES*-a, osim grubom računarskom silom. Vrijednost funkcije dobiva se u nekoliko koraka.

Najprije se od ulaznih 32 bita *Ri* proširenjem zadanom tablicom dobiva 48 bita. Ta se vrijednost zbraja logičkom operacijom *XOR* sa ključem *Ki* paralelno nad svakim bitom. Dobivena se 48 bitna vrijednost dijeli na osam dijelova od po šest bita. Prvi i zadnji bit svakog dijela predstavlja adresu reda, a srednja četiri adresu kolone u tablici selekcije, odnosno, pomoću šest određena su četiri bita. Istim postupkom

nad svakom šestorkom od ulaznih 48 bita selekcijom dobivamo 32 bita. Tih se 32 bita još permutira zadanom tablicom te se dobiva konačna vrijednost funkcije *F*. [2]

2.2. Asimetrično kriptiranje

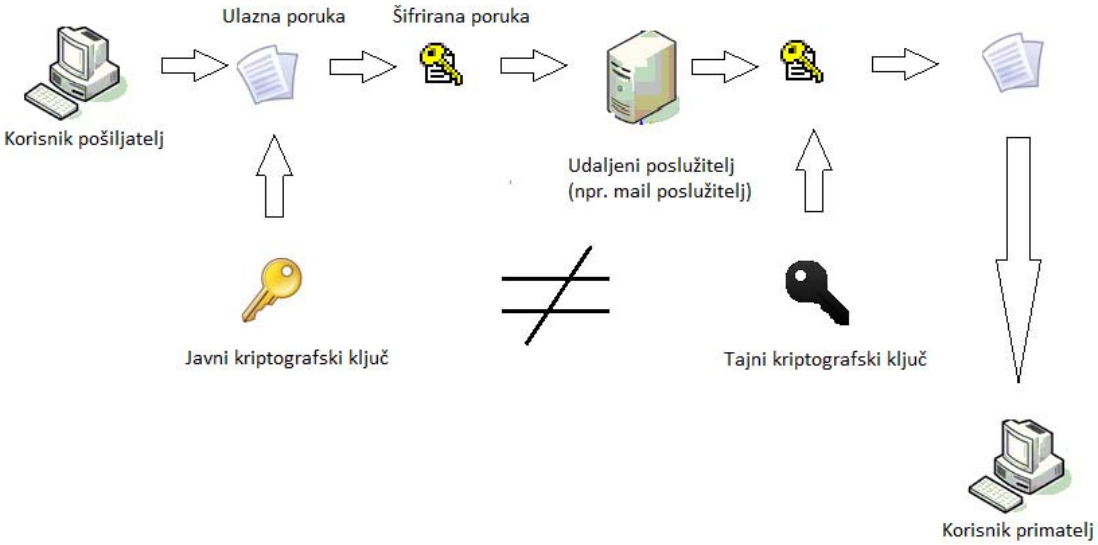
Godine 1976. Whitfield Diffie i Martin Hellman su ponudili jedno moguće rješenje problema razmjene ključeva, zasnovano na činjenici da je u nekim grupama potenciranje puno jednostavnije od logaritmiranja. Ponuđeno je rješenje razmjene simetričnih kriptografskih ključeva preko nesigurnog kanala. Dvije godine kasnije objavljen je prvi praktično iskoristiv asimetrični kriptografski sustav nazvan RSA po prvim slovima njegovih autora; Rivest, Shamir i Adleman. Nakon toga došlo je do značajnog razvoja asimetričnih kriptografskih sustava i moderna kriptografija od tada kreće u sasvim drugom smjeru. Pojavom asimetrične kriptografije otvorila su se vrata i za upotrebu digitalnog potpisa koji je zasnovan na asimetričnom algoritmu.

Osnovna razlika između simetričnih i asimetričnih kriptografskih sustava je u organizaciji kriptografskih ključeva. Kod simetričnih kriptografskih sustava pošiljatelj i primatelj šifriranih poruka su istim ključem šifrirali i dešifrirali poruke, dok kod asimetričnih kriptografskih sustava to nije slučaj. Naime, kriptografski ključ za šifriranje asimetričnim kriptografskim sustavom nije moguće koristiti i za dešifriranje prethodno šifrirane poruke te ga stoga nije potrebno štititi. Ovdje problem sigurne dostave kriptografskog ključa za šifriranje ne postoji, jer ga je moguće distribuirati i nesigurnim komunikacijskim kanalima.

Kod asimetričnih kriptografskih sustava postoje dvije vrste kriptografskih ključeva - javni i tajni kriptografski ključ. Javni kriptografski ključ koristimo isključivo za šifriranje, a tajni za dešifriranje. Javni i tajni kriptografski ključ čine jedinstveni par; svakom javnom ključu pridodan je jedinstveni tajni ključ. U praksi je vrlo teško, gotovo nemoguće, poznavanjem jednog od njih izračunati drugi.

Korisnik koji želi zaštićeno komunicirati izrađuje za sebe par asimetričnih kriptografskih ključeva. Tajni kriptografski ključ zadržava kod sebe te na taj način

osigurava da nitko drugi nema nikakve informacije o istome. Pripadajući javni kriptografski ključ korisnik javno objavljuje svima ili ga na neki drugi način dostavlja osobama kojima želi omogućiti šifriranje njemu namijenjenih poruka. Poruke šifrirane takvim javnim kriptografskim ključem može dešifrirati samo osoba koja posjeduje njegov komplementarni, odnosno tajni kriptografski ključ.
Čak niti osoba koja je šifrirala poruku javnim kriptografskim ključem nije u mogućnosti istu dešifrirati.



Slika 3. Primjer prijenosa podataka uz asimetrično kriptiranje

Povećanjem broja korisnika simetričnog kriptografskog sustava koji žele neovisno jedni o drugima zaštićeno komunicirati, potreban broj kriptografskih ključeva relativno brzo raste. Za 5 korisnika koji žele sigurno komunicirati svaki sa svakim, uz uvjet da svaku štićenu poruku preostali u grupi nisu u mogućnosti dešifrirati, treba 10 kriptografskih ključeva, a za 20 korisnika broj potrebnih kriptografskih ključeva penje se na čak 190. U slučaju asimetričnog kriptografskog sustava broj parova ključeva jednak je broju korisnika. Za 5 korisnika, potrebno je 5 parova ključeva, a za 20 korisnika jednako tako 20 parova ključeva. Iz ovoga je vidljivo da je broj kriptografskih ključeva koje je potrebno razmijeniti između korisnika koji žele zaštićeno komunicirati asimetričnim kriptografskim sustavom bitno smanjen u odnosu na simetrične kriptografske sustave. [4]

2.2.1 RSA algoritam

Duljina kriptografskih ključeva kod asimetričnih kriptografskih sustava najčešće se kreće od 256 bita, za asimetrične kriptografske sustave bazirane na eliptičkim krivuljama, do 4096 bita za asimetrične kriptografske sustave temeljene na RSA algoritmu. Sa današnjim stupnjem razvoja informatičke tehnologije i procesorskim moćima modernih računala navedene duljine kriptografskih ključeva zadovoljavaju sigurnosne potrebe korisnika, bilo da se radi o privatnim osobama ili državnim institucijama. Povećanjem broja matematičkih operacija koje procesori mogu izvršiti u jedinici vremena, kao i razvojem kvantnih računala, tražit će i povećanje duljine ključa kako bi se zadržala dosadašnja kvaliteta sigurnosti asimetričnih kriptografskih sustava.

Neka je $n = p \cdot q$, gdje su p i q prosti brojevi.
Za $K = (n, p, q, d, e)$ definiramo funkcije $e_K(x) = x^e \bmod n$ i $d_K(y) = y^d \bmod n$.
Vrijednosti n i e su javne, a p,q i d su tajne.
Nakon što smo definirali velike proste brojeve p i q, izračunamo $n = p \cdot q$ i $\varphi(n) = (1-p) \cdot (1-q)$. Izaberemo na slučajan način broj e takav da je $e < \varphi(n)$ i $(\varphi(n), e) = 1$. Nakon toga tajno izračunamo d tako da je $de \equiv 1 \pmod{\varphi(n)}$, tj. $d \equiv e^{-1} \pmod{\varphi(n)}$. To se radi pomoću Euklidovog algoritma.
Ovdje nam (n,e) predstavlja javni ključ za šifriranje.
Sada za kodiranje koristimo sljedeću formulu:
$M_{kodirano} = (M_{izvorno} \text{ ^ } 3) \bmod K_{javni}$, a za dekodiranje
$M_{izvorno} = (M_{kodirano} \text{ ^ } K_{tajni}) \bmod K_{javni.}$

Primjer:
Želimo kodirati riječ „Maja“. *U ASCII* formi ova riječ glasi *77 65 74 65*.
Kao dva prosta broja p i q uzeti ćemo $p=9839$ i $q = 22391$.
U tom slučaju izračunamo li ključeve dobiti ćemo
$K_{javni} = 220305049$, $K_{tajni} = 146848547$.
Koristeći gore navedene formule možemo izračunati:
$M_{kodirano} = (77657465 \text{ ^ } 3) \bmod 220305049 = 162621874$

Dok će primatelj iste poruke moći primljeni sadržaj dekodirati:

$M_{izvorno}$=(162621874 ^ 146848547) mod 220305049 = 77657465, što predstavlja izvornu poslanu poruku u *ASCII* kodu.

Prednost ovog algoritma je jednostavnost, ali i njegova sigurnost. [2]

U sljedećoj tablici prikazana je ovisnost izračuna tajnog ključa iz javnog ključa, uz primjenu računala brzine 1*MIPS,* u odnosu na duljinu ključa:

Dužina ključa u bitovima	Vrijeme
50	3.9h
100	74 godine
150	10^6 godina
200	$3,8 * 10^9$ godina

Tablica 1. Ovisnost probijanja tajnog ključa javnim u odnosu na veličinu ključa

3. Društveno i poslovno umrežavanje

Društveno umrežavanje je društvena struktura sastavljena od pojedinaca ili organizacije koji su povezani jednom ili više vrsta međuzavisnosti poput prijateljstva, srodstva, financijskih razmjena ili odnosa vjerovanja, znanja ili prestiža.

Analiza društvene mreže prikazuje društvene odnose u smislu mreža sastavljenih od čvorova, pojedinaca unutar jedne mreže, te veza, njihovih odnosa. Čvorovi na koje je pojedinac povezan su i društveni kontakti tog pojedinca. Istraživanja pokazuju da društvene mreže igraju važnu ulogu u stvaranju organizacija i ostvarivanju ciljeva pojedinaca. Mreže se također mogu koristiti za mjerenje socijalnog kapitala (vrijednosti koje pojedinac dobiva od društvene mreže).

Slika 4. Društvena mreža

Analiza društvene mreže danas koristi vlastite izjave, metode, programske podrške, analize društvene mreže i istraživače. Analitičari obično proučavaju ili

cijelu mrežu (sve veze s određenim odnosom u definiranoj populaciji) ili osobnu mrežu (veze koje određeni ljudi imaju, osobna zajednica).Njihov pristup temelji se na sljedećem:

- Ne postoji pretpostavka da su grupe temelji društva: pristup je otvoren za proučavanje od nelokalnih zajednica do veza između web stranica.
- Umjesto usmjeravanja na pojedince kao dijelove analize, fokusira se na način na koji struktura veze utječe na individue i njihove odnose.
- Za razliku od analize koja uzima u obzir da socijalizacija putem normi određuje ponašanje, analiza mreže proučava utjecaj normi na strukturu i kompoziciju veze.

Oblik društvene mreže također ima veliki značaj. Naime, manji oblik može biti nekorisniji iz razloga što se pojedinci uglavnom znaju te su razmjene znanja i mogućnosti ograničene i odveć poznate. Kod većeg oblika moguć je pristup širem krugu informacija.

Mrežna analitička programska podrška sadrži alate za zastupanje čvorova i veza u mreži, te za analiziranje mreže podataka. Poput ostalih alata, podaci mogu biti spremljeni u vanjskim datotekama. Mrežni alati za analizu omogućuju istraživačima da proučavaju velike mreže, poput samog interneta. Koriste se matematičkim funkcijama koje se mogu primijeniti na model mreže.

Društvene mreže su korisne za ispitivanje kako organizacije djeluju međusobno, te kako se odnose pojedini zaposlenici iz različitih organizacija jedan prema drugome. Igraju ključnu ulogu i u zapošljavanju, poslovnom uspjehu te obavljanju posla. Pomoću mreža može se prikupiti veliki broj informacija te se tako i dostići konkurencija.

Kada je riječ o društvenom umrežavanju, najčešće se koriste web stranice koje funkcioniraju kao zajednica korisnika interneta. Mnoge zajednice dijele zajedničke interese u obliku hobija, religije, politike i sl. [5]

3.1. Poslovno umrežavanje

Poslovno umrežavanje je aktivnost pomoću koje skupina istomišljenika prepoznaje, djeluje i stvara poslovne mogućnosti. Cilj takvog umreženja je poslovanje. Postoji nekoliko istaknutih organizacija koje stvaraju modele poslovnog umrežavanja, koji dopuštaju poslovnim osobama stvaranje novih poslovnih odnosa i novih mogućnosti u isto vrijeme. Također, poslovno umrežavanje ima veliki značaj i za informacijsku tehnologiju. Mnogi poslovni ljudi smatraju da je poslovno umrežavanje veoma isplativa metoda za stvaranje novih poslova u odnosu na oglašavanje ili odnose s javnošću. To je zato jer je poslovno umrežavanje jeftina aktivnost koja uključuje više osobni angažman nego tvrtkin novac. Kao primjer, poslovno umrežavanje može uzrokovati sastanke kolega u cilju razmjene poslovnih rezultata, što naravno podrazumijeva i žrtvovanje vlastitog vremena i građenje jedan-na-jedan odnosa sa kolegom.

Poslovnom umrežavanje može biti provedeno u lokalnoj poslovnoj zajednici ili putem interneta. Količina web stranica poslovnog umrežavanja povećala se tijekom posljednjih godina, što nije čudno, s obzirom na mogućnost spajanja ljudi diljem svijeta putem interneta. Poslovno umrežavanje ima značaj i u *ICT* području, odnosno pružanju operativne podrške tvrtkama i organizacijama.
To se odnosi na koordinaciju sa širim opsegom i jednostavnijom provedbom nego npr. improvizacija pretraživanja web stranica za transakcije kolegama (koja je korisna, ali vrlo komplicirana zbog obilja sučelja između različitih organizacija i čak između različitih *IT* primjenskih sustava iste organizacije).

S obzirom na razvoj umrežavanja, mnoge tvrtke ga koriste kao temeljni dio svoje strategije. One koje su razvile veliku mrežu veza dobavljača i tvrtka mogu se nazvati mrežnim tvrtkama. Mrežne tvrtke teže biti otvorene i pristupačne, dok one koje se oslanjaju na hijerarhijski, tradicionalni pristup su zatvorenije, selektivnije i kontroliranije.

Poslovno umrežavanje se može i samo podijeliti na dvije glavne cjeline:

- Licem u lice (eng. *Face to face*) poslovno umrežavanje
- Mrežno (eng. *Online*) poslovanje

Mrežno poslovanje služi kao sve češće sredstvo promoviranja tvrtki te proširenja kruga poslovnih kontakata same tvrtke. Također, moguće je na taj način izgraditi povjerenje sa svojim poslovnim partnerima, te je omogućena lakša potraga za ljudima unutar njihove poslovne mreže. Kroz upoznavanje, članovi mogu doći u kontakt s novim potencijalnim poslovnim partnerima. S obzirom da se sam posao i tvrtka mogu globalno širiti, poslovno umrežavanje čini jednostavnim očuvanje kontakata diljem svijeta. Da to ne vrijedi samo za velika poduzeća, i srednja i mala poduzeća razvila su posebne prekogranične platforme elektronske trgovine i poslovne mreže.

Tvrtke i mnoge organizacije trebaju neku vrstu informatičke podrške. Uobičajeno i tradicionalno je omogućena sustavima programske podrške ili paketima programske podrške. *ICT* pristup poslovnog umrežavanja zaista obnavlja operativnu podršku od nule, uključujući dvije ključne poslovne mogućnosti: informacijske doprinose i automatske izmjene informacija. Informacijski doprinosi i razmjene moraju biti podržani pohranom podataka te sigurnošću pristupa (potpis, enkripcija/ šifriranje, autentičnost, dekripcija/dešifriranje).

Licem u lice poslovno umrežavanje se koristi kada ljudi imaju tendenciju da znaju i sretnu se s onim s kim posluju. Preferira se više od mrežnog umrežavanja baš zbog veće kvalitete odnosa s partnerima. [6][7]

Primjena kriptografije u poslovnom umrežavanju je uvedena zbog održavanja nekih osnovnih kocepata poslovanja:

- Povjerljivost podataka (eng. *Data Confidentiality*) - zaštita podataka kriptiranjem, te zaštita od rizika pristupa podatcima od strane neovlaštenih osoba
- Netaknutost podataka (eng. *Data Integrity*) – integritet podataka se prati tako da ukoliko nastaju promjene obavijesti se ovlaštene osobe o promjeni
- Vjerodostojnost (eng. *Client Autenticity*) – definiranje ispravnog modela povjerenja, čime se neće narušiti sigurnost korisničkih transakcija
- Provjera valjanosti (eng. *Client Authentication*) - autentifikacija korisnika u sustavu također se kriptira, kako korisnici koji ne posjeduju pristup sustavu ne bi mogli ukrasti kroz sustav podatke korisnika koji tom sustavu pristupaju

4. Radni okvir za sigurnu komunikaciju

Primjenski sustav ostvaren u ovom završnom radu može poslužiti kao osnova za izradu kompleksnog sustava za komuniciranje nesigurnim komunikacijskim kanalima. Sustav je osmišljen kao jednostavni sustav za komunikaciju između osoba. Svakom korisniku sustava pridodijeljena je vlastita datoteka koja sadrži podatke o prijateljstvima sa drugim osobama, zahtjevima za prijateljstvima, te tajnim ključevima.

Korisnik nakon prijave na sustav može dobiti svoju datoteku, no ukoliko ona postoji za njegovo korisničko ime, pridodijelit će mu se već postojeća. Prilikom učitavanja podataka iz datoteke, korisniku se predočuju novi zahtjevi ukoliko postoje. Ukoliko korisnik nije dobio nove zahtjeve za prijateljstvom, oni se neće prikazati na početnoj stranici. Korisnik sa početne stranice može izabrati dvije ili tri opcije. Ukoliko ima zahtjeve za prijateljstvom može otići na posebnu stranicu za provjeru zahtjeva za prijateljstvima, te dodavati osobe ili odbijati zahtjeve za prijateljstvima. Prilikom dodavanja osobe, datoteke korisnika koji je prijavljen i datoteka osobe koja je zatražila prijateljstvo se ažuriraju. Prilikom ažuriranja datoteke prihvaćaju podatke osobe na suprotnoj strani, te generiranog tajnog ključa za zaštitu poruka.

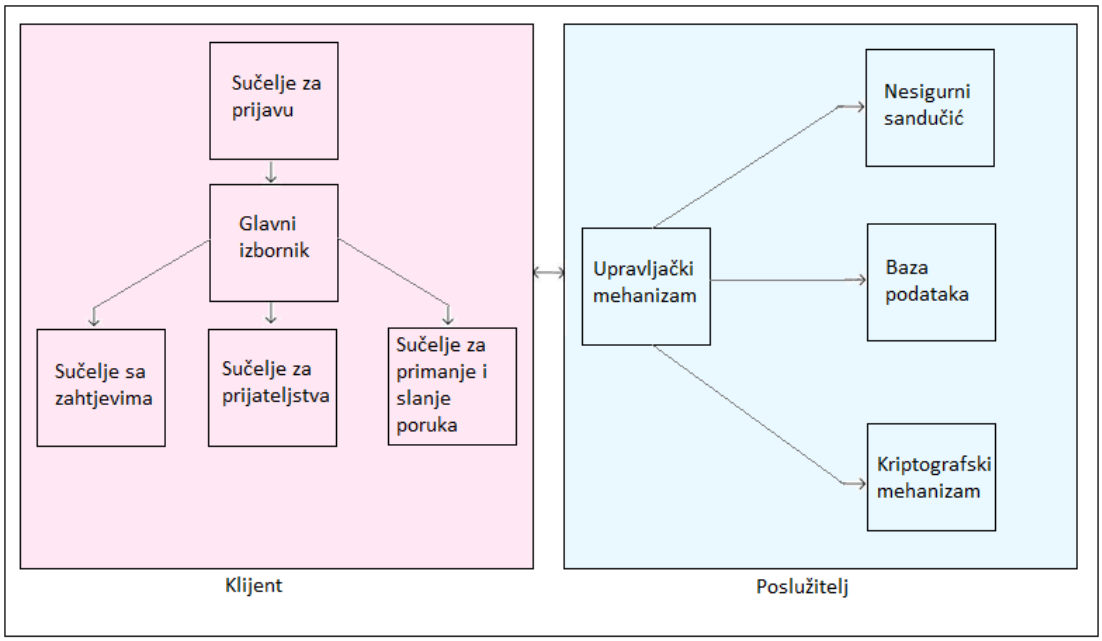
Sa glavnog izbornika korisnik može birati također odlazak na stranicu za upravljanje listom prijatelja ili na stranicu za slanje i primanje poruka. Na stranici s za upravljanje listom prijatelja možemo dodavati nepoznate osobe ili brisati postojeće prijatelje. Ukoliko dodajemo nepoznate osobe, to jest korisnike sustava koji nisu na korisničkoj listi prijatelja, njima se šalje zahtjev za prijateljstvom putem ažuriranja podataka u njihovoj korisničkoj datoteci. Ukoliko brišemo prijatelja, ažuriraju se datoteke prijavljenog korisnika i korisnika prijatelja kojega smo izbrisali.

Sa glavnog izbornika također je moguće pristupiti stranici za primanje i slanje poruka. Stranica ukoliko trenutno prijavljeni korisnik ima novih poruka omogućuje dohvat istih. Dohvaćene poruke su zaštićene, te se zatim dohvaćaju i

tajni ključevi sa vezama korisnik – pošiljatelj, te se zaštićene poruke dekriptiraju kako bi se prikazao njihov originalni sadržaj. Korisnik također može birati i opciju slanja poruke. Poruke se kriptiraju ključem koji je generiran za kominaciju korisnik – primatelj, te se spremaju u poseban direktorij na poslužitelju koji predstavlja nesigurni poštanski sandučić.

4.1 Arhitektura sustava

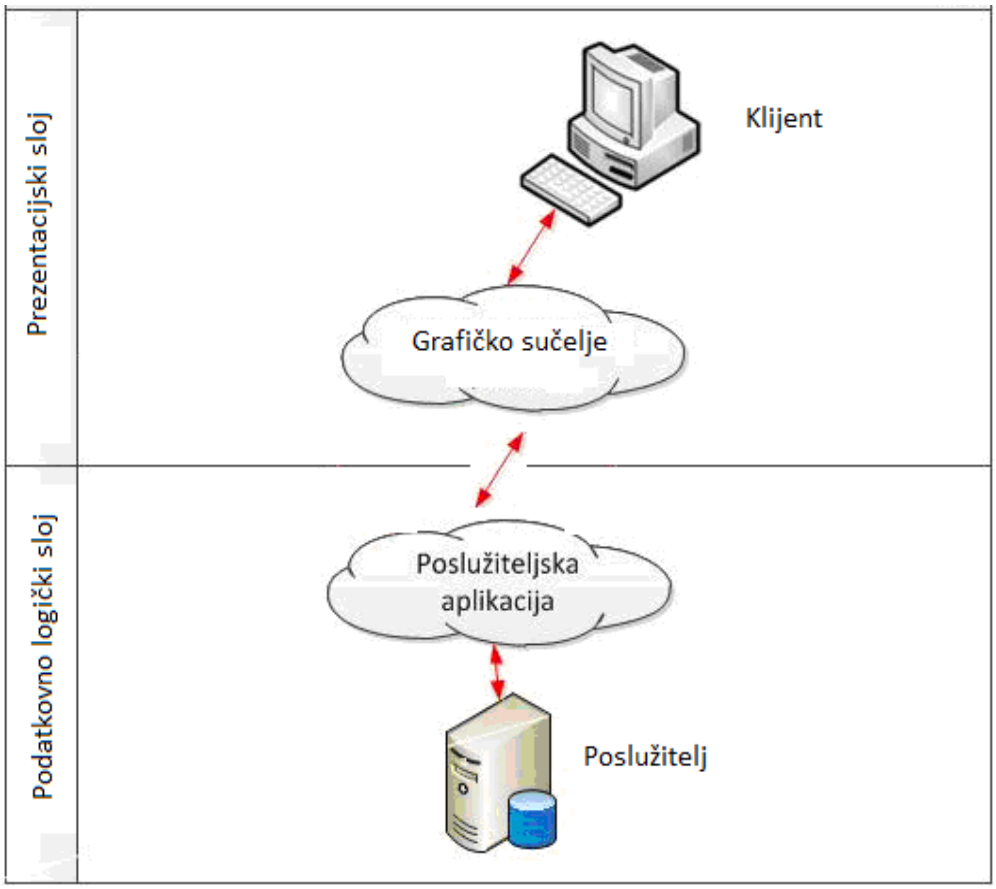
Arhitektura sustava ključna je poveznica između zahtjeva na sustav i samog ostvarenja sustava. Dobro odabrana arhitektura utječe na smanjenje cijene oblikovanja, razvoja i održavanja sustava, poboljšava razumljivost, poboljšava kvalitetu, ali i olakšava ostvarenje sustava zahvaljujući ranoj analizi i uočavanju pogrešaka u oblikovanju.



Slika 5. Arhitektura radnog okvira za sigurnu komunikaciju

Za ovakav sustav odabran je klijent-poslužitelj model dvoslojne arhitekture. Klijenti su računala koja pristupaju i šalju zahtjeve središnjem računalu - poslužitelju. Poslužitelj obrađuje zahtjeve i šalje odgovor klijentima. U našem slučaju klijenti su računala kojima pristupaju korisnici sustava koji šalju zahtjeve

poslužitelju (slanje poruka ili dodavanje/brisanje prijatelja), te nam upravo ovakav model arhitekture odgovara. Model ovog klijenta definiran je kao tanki klijent (eng. *thin client).* Kod tankog klijenta poslužitelj je zadužen za upravljanje podatcima i logiku primjenskog sustava, dok je klijentu pridružena samo prezentacijska logika.



Slika 6. Dvoslojna arhitektura klijent poslužitelj

Nedostatak ovog modela jest veliko opterećenje mreže, zbog istovremenog pristupa više korisnika odjednom, također postoji mogućnost preopterećenja poslužitelja, što može uzrokovati greške u radu, a to također znači skuplje glavno računalo. Također tanki klijent je oblikovan da bude što jednostavniji i manji. Poslužitelj prima od klijenata zahtjeve, obrađuje ih te ih vraća u obliku klijentu razumljivih informacija koje se prikazuju na grafičkom sučelju. Prednost ovog sustava leži u tome što prilikom promjene rada poslužiteljskog dijela primjenskog sustava ne mijenja se klijentski dio. U slučaju korištenja udomljenika putem internet preglednika skoro se nikada neće tražiti nadogradnja preglednika za ispravan rad udomljenika. Također prednost je u složenosti izrade

velikih primjenskih sustava, klijentska računala ne moraju imati veliku računalnu snagu, te se promjena logike primjenskog sustava može obaviti centralizirano na glavnom računalu.

Modelirani sustav je prema svojem obliku primjenski sustav za obradu, kriptiranje, slanje, primanje tekstualnih poruka, te stvaranje poveznica između više klijenata putem poslužiteljske baze podataka. Ovaj model koristi internet preglednik kao klijentski primjenski sustav, dok je poslužiteljski primjenski sustav pisan u *Java* programskom jeziku.

Arhitektura ovog sustava sadrži i dva podsustava:

- Grafička sučelja
- Bazu podataka u obliku datotečnog zapisa

4.1.1. Grafička sučelja

Predviđeno je za ovaj sustav da sadrži početno grafičko sučelje, sučelje za prijavu, dok se sva ostala sučelja generiraju nakon što su poslani zahtjevi poslužitelju. Sučelju za prijavu pristupaju svi korisnici. Na sučelju unose svoje odabrano korisničko ime, te prilikom autentifikacije od strane poslužitelja pristupaju sučelju glavnog izbornika generiranog od strane poslužitelja.
Kroz sučelje glavnog izbornika ostvareni su pozivi sučelja za prijateljstva, poruke ili provjeru pridošlih zahtjeva za prijateljstvima.
U sučelju *za* prijatelje ostvarene su funkcije za brisanje i dodavanje novih prijatelja, pri čemu se unos podataka obavlja izravno sa tipkovnice u za to predviđena polja.
Na sučelju *za* poruke ostvarene su funkcije provjere novih poruka i slanja poruke.
Sva sučelja trebala bi biti jednostavna i intuitivna za korištenje, tako da bi im mogli pristupiti i ljudi koji nemaju iskustva s radom na računalu.
Predviđeno je da sva sučelja budu izrađena u *HTML*-u, kako bi se održala konzistentnost.

4.1.2. Baza podataka

U bazu podataka pohranjuju se gotovo svi podaci bitni za rad sustava. U njoj su pohranjeni podaci o korisnicima, na temelju kojih se vrši autorizacija korisnika. U bazi se, isto tako, bilježi za svakog korisnika njegova prijateljstva, ključ kriptiranja podataka između njih, te slanje poruka između korisnika. Primjenski sustav čuva integritet i konzistentnost baze podataka, tako da bilo kakve pogreške u programu neće uzrokovati pogreške u bazi podataka, niti će istovremeni rad dva korisnika izazvati pogreške. Zbog jednostavnosti izvedbe kao baza podataka koristi se datotečni sustav, te se u datoteke sa ekstenzijom *.txt* unose svi potrebni podatci.

4.1.3. Komunikacija u sustavu

Prikazom komunikacije u sustavu definirati ćemo radnje koje sustav može obavljati. Komunikacija ovog sustava obavljati će se na načelu *request – reponse*, što znači da za svaki korisnički zahtjev (eng. *request*) prema poslužitelju, poslužitelj će obraditi podatke, te će vratiti klijentu odgovor (eng. *response*).

Sustav se sastoji od sljedećeg:

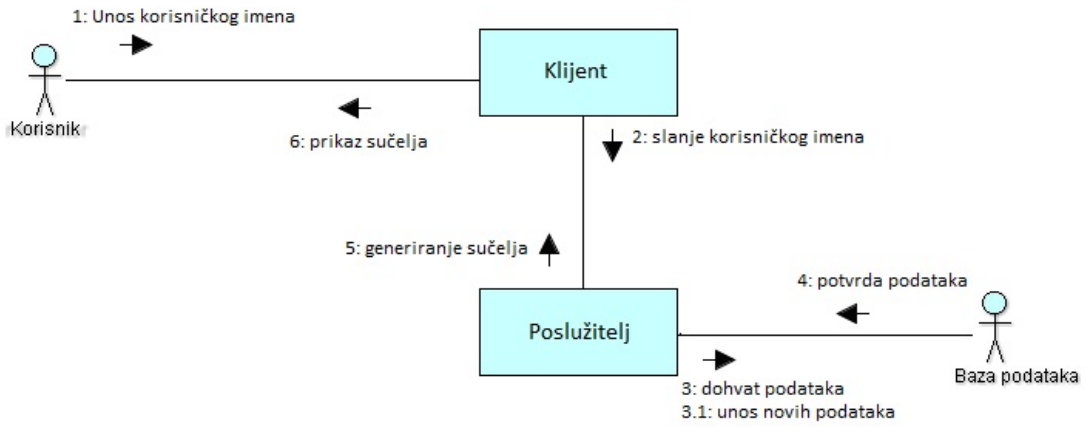
- *Sustav za prijavu*
- *Sustav za prihvaćanje prijateljstva*
- *Sustav za provjeru zahtjeva za prijateljstvo*
- *Sustav za dodavanje i brisanje prijatelja*
- *Sustav za slanje poruke*
- *Sustav za primanje poruka*

4.1.3.1. Sustav za prijavu

Sustav za prijavu definiran je kao ulazni podsustav. Korisnik unosi na glavnom sučelju svoje željeno korisničko ime te poslužitelju prenosi informaciju o upisanom korisničkom imenu. Poslužitelj tada provjerava korisničko ime u bazi podataka, ukoliko ono postoji učitavaju se podatci za danog korisnika, a u protivnom se kreira novi profil koji predstavlja tog korisnika u sustavu. Ukoliko je stvoren novi korisnik ti podatci se zapisuju u novu datoteku koja predstavlja bazu podataka. Nakon toga poslužitelj generira jedno od dva sučelja:

- Ukoliko korisnik ima zahtjeva za prijateljstvima generira se sučelje sa posebnim pristupom zahtjevima
- Ukoliko korisnik nema zahtjeva za prijateljstvima generira se sučelje bez posebnog pristupa zahtjevima

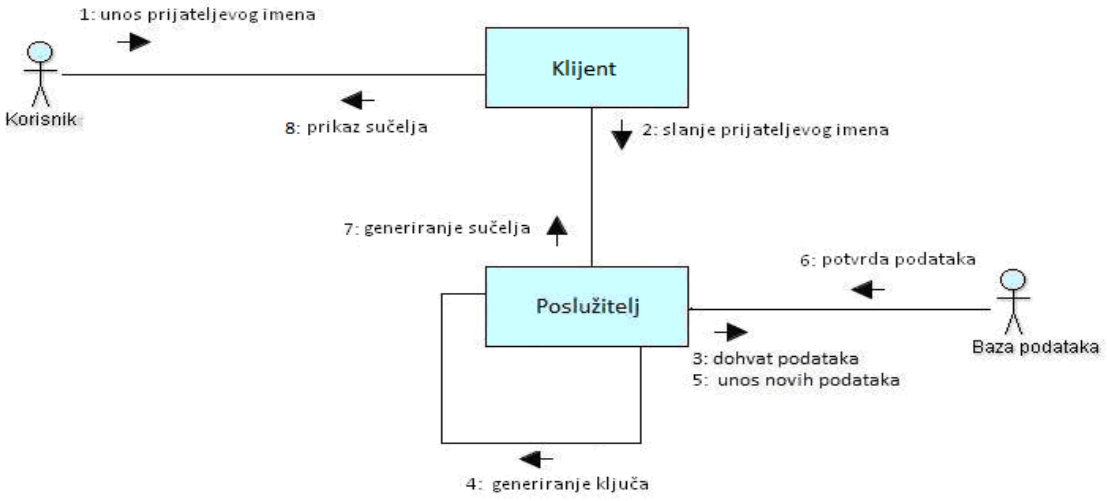
Ovo sučelje poslužitelj vraća korisniku, te putem njega korisnik nastavlja sa radom.



Slika 7. Sustav za prijavu

4.1.3.2. Sustav za prihvaćanje prijateljstva sa određenom osobom

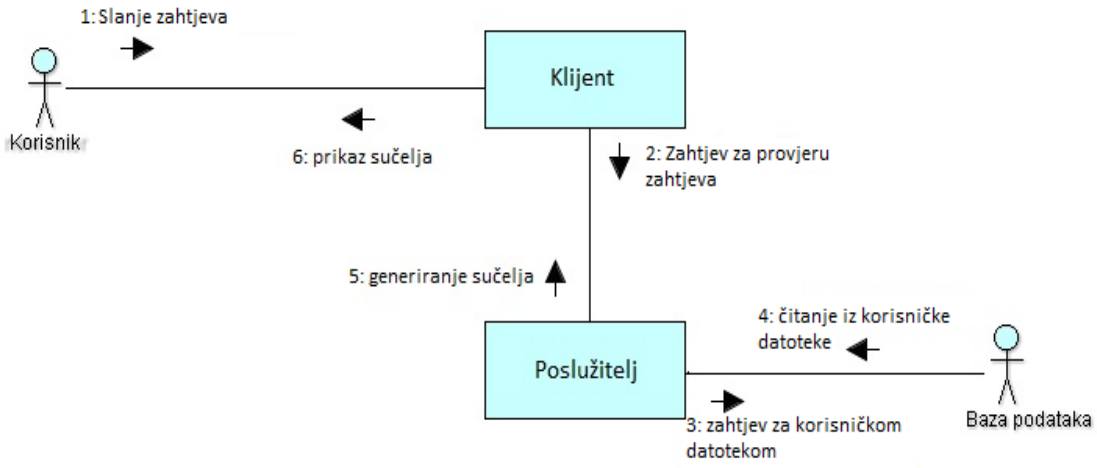
Korisnik otvaranjem sučelja za prihvaćanje prijatelja može odlučiti želi li odbiti ili dodati osobe koji su zatražili zahtjev za prijateljstvom. Korisnik unosi ime osobe sa kojim želi poslovati. Ukoliko je kliknuo prihvaćanje prijateljstva, u bazu podataka se u korisnikovu i osobinu datoteku zapisuje podatak o prijateljstvu, te poslužitelj za taj par generira ključ za enkripciju, koji se zapisuje također u bazu podataka, u datoteku obje osobe. Poslužitelj korisniku vraća podataka o uspješnosti ili neuspješnosti operacije, te na sučelju generira gumb za povratak na prethodnu stranicu.



Slika 8. Prikaz sustava za odobravanja prijatelja

4.1.3.3. Sustav za provjeru zahtjeva za prijateljstvima

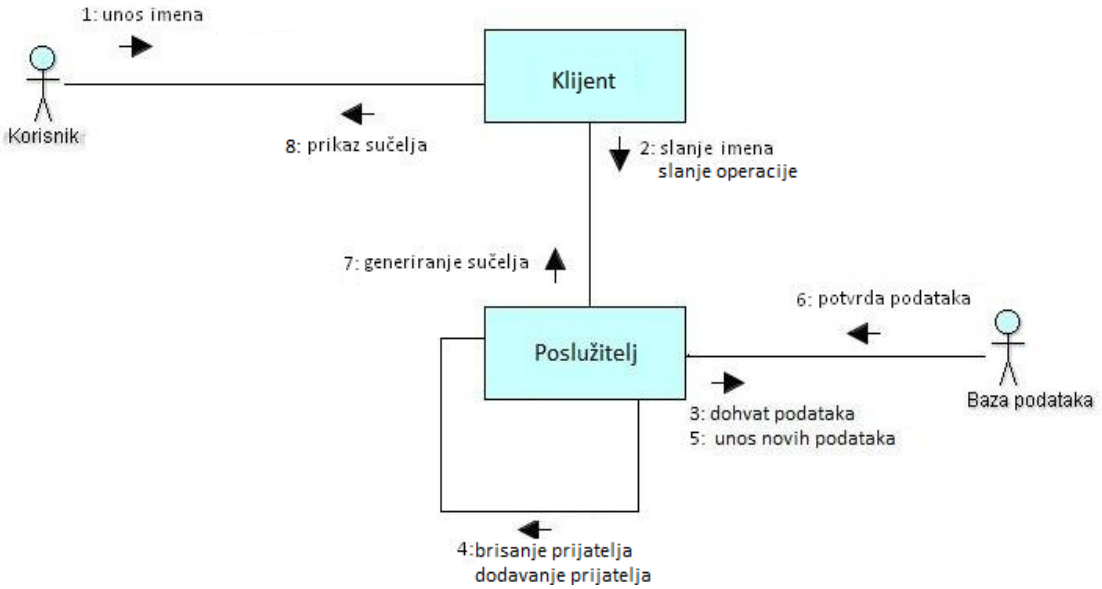
Prilikom prijave na sustav, poslužitelj učitava datoteku prijavljene osobe. Ukoliko postoje zapisi o zahtjevima korisnik klikom na gumb za zahtjeve pokreće sustav za provjeru zahtjeva za prijateljstvima. Sustav prima zahtjev od korisnika za pokretanjem funkcije, nakon čega pristupa bazi podataka, čitajući podatke iz korisničke datoteke. Nakon toga poslužitelj korisniku generira sučelje sa prikazom svih osoba koje trenutnog korisnika žele dodati kao prijatelja i mogućnostima za daljnje upravljanje.



Slika 9. Prikaz rada sustava za provjeru zahtjeva

4.1.3.4. Sustav za dodavanje i brisanje prijatelja

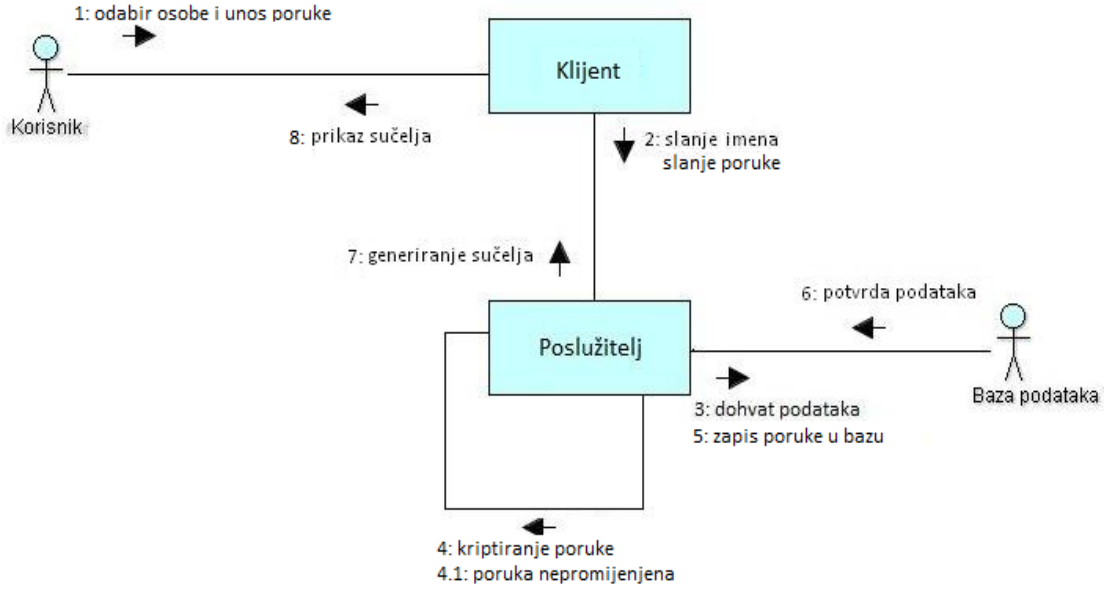
Sa korisničkog sučelja za prijateljstva korisnik može dodavati nove prijatelje ili brisati već postojeće. Korisnik unosi ime željene osobe koju želi dodati kao novog prijatelja ili unosi ime već postojećeg prijatelja ukoliko ga želi obrisati. Odabirom kontrole korisnik određuje akciju nad sustavom. Poslužitelj dohvaća uneseni podatak, te dohvaća podatke iz baze podataka. Ukoliko podatci nisu postojani javlja se greška. Nakon što su podatci učitani, u ovisnosti o odabranoj akciji poslužitelj će u bazu podataka vratiti izmijenjene podatke za korisnika i upisanu osobu. Ukoliko je traženo brisanje prijatelja, u korisničkim datotekama se brišu zapisi o prijateljstvu, a ukoliko je traženo dodavanje prijatelja, tada se u datoteku osobe s kojom želimo prijateljstvo zapisuje zahtjev za prijateljstvom. Poslužitelj korisniku vraća sučelje sa odgovorom o uspješnosti operacije i opcijama za daljnji rad.



Slika 10. Prikaz rada sustava za dodavanje i brisanje prijatelja

4.1.3.5. Sustav za slanje poruka

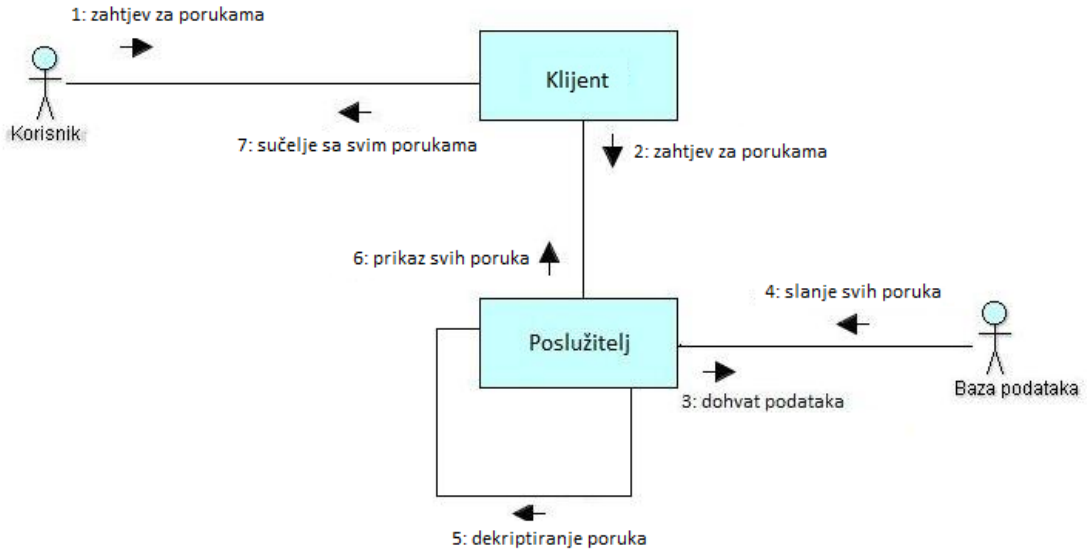
Korisnik na sučelju za slanje poruke unosi ime osobe kojoj želi slati poruku, te samu poruku koju šalje. Ukoliko se radi o osobi koja se ne nalazi na listi prijatelja, poruka se neće slati, te će se dojaviti greška. Ukoliko se osoba nalazi na listi prijatelja, poslužitelj će najprije pristupiti bazi podataka, pročitati tajni ključ kriptiranja koji je generiran samo za ovaj par osoba koje komuniciraju. Tada će primjenom tajnog ključa kriptirati korisnikovu poruku, te je zapisati u bazu podataka. Ovakva poruka je zaštićena od vanjskih utjecaja, tj. treća osoba može pristupiti sadržaju, ali ga neće razumjeti. Korisniku poslužitelj vraća sučelje sa odgovorom o pogrešci ukoliko je ista nastupila ili o uspješnosti obavljanja operacije.



Slika 11. Prikaz rada sustava za slanje poruke

4.1.3.6. Sustav za primanje poruka

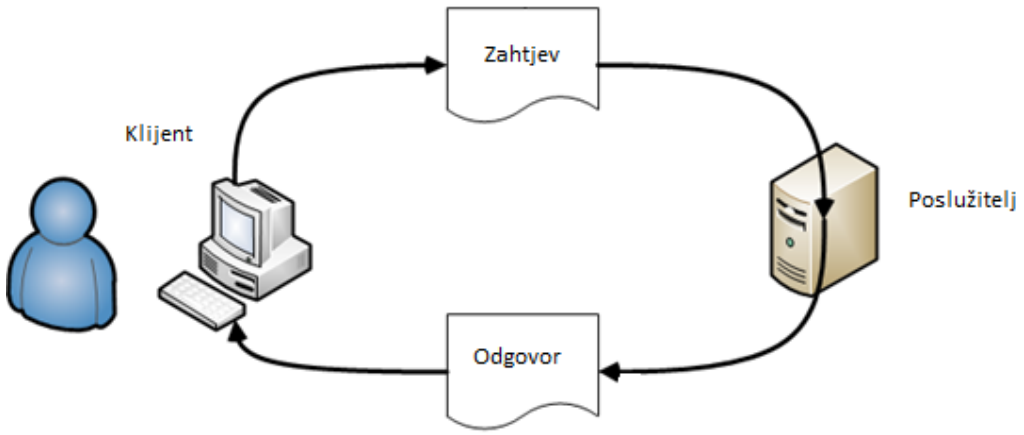
Ukoliko je sustav korisniku javio da ima poruke, korisnik može odabrati opciju prikaza svih poruka. Pozivom funkcije za prikaz poruka poslužitelj dohvaća korisničko ime, te dohvaća iz baze podataka sve datoteke koje su namijenjene za to korisničko ime. Sustav iz datoteke korisnika uzima za svakog pošiljatelja ključ enkripcije, te dekriptira poruke. Ukoliko pošiljatelj ne postoji kao prijatelj trenutnog korisnika, poruka se ne dekriptira, samo prikazuje. Nakon dekripcije poslužitelj vraća sučelje sa prikazom svih poruka, posebno odijeljenih.



Slika 12. Prikaz sustava za primanje poruka

4.2. Programsko ostvarenje

U izradi ovog završnog rada korištene su tehnologije *HTML* i *Java*. *HTML* smo pri tome koristili za generiranje korisničkih sučelja putem kojih korisnik pristupa obavljanju odabranih akcija. *Java* programski jezik korišten je kao razvojni jezik za izradu primjenskog sustava na poslužitelju. Prilikom odabrane akcije korisnik šalje zahtjeve prema poslužitelju, te nakon obavljanja određene akcije korisniku vraća odgovor. Ostvarenje komunikacije putem računalne mreže ostvareno je pomoću *Java servlet* tehnologije, koja sadrži mogućnosti primanja zahtjeva i vraćanja odgovora koje korisnik šalje putem internet preglednika. Također Java programski jezik podržava kriptografiju i ima ostvarene gotove funkcije koje barataju sa generiranjem ključeva, kodiranjem i dekodiranjem poruka. Također za prijenos podataka putem tekstualnih datoteka podatke je bilo potrebno dodatno pretvoriti u prihvatljivi oblik putem *SUN BASE64Encoder*-a.



Slika 13. Komunikacija na načelu zahtjev-odgovor

4.2.1. HTML

HTML, skraćeno od *HyperText Markup Language,* što znači prezentacijski jezik za kreiranje web stranica. Jezik nam služi za definiranje i oblikovanje internet stranica. *HTML* se sastoji od blokova kodá koje služe za definiranje izgleda stranice. Blokovi naredbi označavaju se posebnim oznakama koji dolaze u paru, jedna koja predstavlja početak definicije stranice, te jedna koja označava kraj.

HTML omogućuje umetanje slika, tablica, objekata i interaktivnih formi na našu stranicu. Primjenom interaktivnih formi možemo ostvariti funkcionalnosti unosa podataka, izmjene podataka, interakcije sa drugim korisnicima i sl.
HTML je također vrlo otvoren jezik, njega stoga obično nalazimo u interakciji sa drugim jezicima. Neki od često primjenjivanih jezika uz *HTML* su *Java*, *JavaScript*, *PHP*, itd. Uz *HTML* se još najčešće koristi i *CSS,* jezik koji dodatno opisuje prezentacijsku semantiku samog *HTML* koda. [8]

4.2.1.1. HTML primjena

U ovom radu smo koristili *HTML* kod za prikaz sučelja korisniku. *HTML* kodom je jednostavno generirati sučelje koje će korisniku biti razumljivo, jednostavno, a pokretanje zahtjevanih akcija od strane korisnika lako ostvareno. *HTML* kod se može prikazati u svakom današnjem internet pregledniku, te je stoga olakšana upotreba na korisničkoj strani primjenskog sustava. Izmjene prilikom promjena na poslužitelju neće utjecati na prikaz stranice kod korisnika.

Sučelje za prijavu
Prilikom ostvarenja sučelja za prijavu, bitno je bilo razmotriri potrebe korisnika prilikom prijave na sustav. S obzirom na jednostavnost odlučeno je ostvariti samo unos polja za korisničko ime, te gumba koji pokreće funkciju prijave na sustav. Na sljedećoj slici možemo vidjeti kod koji opisuje izgled stranice sučelja za prijavu. Najprije smo definirali da se radi o *HTML* kodu. Nakon toga je definirano tijelo stranice, u kojem stvaramo jednu interaktivnu formu, čijom se interakcijom pokreće akcija prijavljivanja na sustav putem funkcije *LoginSkripta*.
U tijelu forme sadržano je polje za unost teksta (eng. *textbox*), te jedan gumb (eng. *button*) koji pokreće akciju nakon unosa u polje. Nakon toga definirali smo kraj tijela i forme, te kraj dokumenta sa zatvarajućim *HTML* oznakama.

```
<html>
<body>
<form method="post" style="height: 171px" action="LoginSkripta">
<input name="Text1" type="text" /><br />
<br />
<input name="Login" style="height: 31px" type="submit" value="Login" />
</form>
</body>
</html>
```

Slika 14. Kod koji opisuje sučelje za prijavu

Sučelje glavnog izbornika

Nakon prijave na sustav, korisnik dolazi do glavnog izbornika programa. On je također stranica prikazana koristeći *HTML* jezik. Ovo sučelje je generirano od strane poslužitelja, te stoga postoje dvije mogućnosti. Mi ćemo prikazati sučelje sa zahtjevima za prijateljstva.

Ovo sučelje sadrži ispis broja zahtjeva prijateljstva, no s obzirom da se ovo sučelje generirano od strane poslužitelja, varijabla *brojZahtjeva* je također izračunata na poslužiteljskoj strani, te je unesena u generirani kod kao vrijednost.

Sučelje glavnog izbornika je jednostavno, te sadrži dvije forme. Prva forma sadrži gumb i akciju za provjere pridošlih zahtjeva za prijateljstvom, dok druga forma sadrži dva gumba koji omogućuju korisniku baratanje sa porukama ili baratanje sa osobama i prijateljima.

```
<html>
Imate " + brojZahtjeva + " zahtjeva za prijateljstvom.
<form action=\"provjeriPrijateljstva\" >
<INPUT TYPE=submit VALUE=\"Zahtjevi\">
<INPUT TYPE=hidden NAME=\"user2\" VALUE="+ime+ " />
</form>

<form action=\"prijateljstva\">
<INPUT TYPE=submit VALUE=\"Prijateljstva\">
</form>

<form action=\"poruke\">
<INPUT TYPE=submit VALUE=\"Poruke\">
</form>
</html>
```

Slika 15. Kod koji opisuje korisničko sučelje glavnog izbornika

Sučelje za prijateljstva

Sučelje za prijateljstva također se generira na poslužitelju, te prikazuje korisniku. Sučelje je vrlo jednostavno i intuitivno. Korisnik može unijeti ime osobe ili prijatelja te odabrati željenu akciju. Ukoliko je korisnik izabrao brisanje prijatelja, obrisati će ga sa liste prijatelja, a ukoliko je odabrao dodavanje, dodati će osobu kao prijatelja. Uvjeti za uspostavljanje ispravne komunikacije između poslužitelja i korisnika su postojanje osobe koju brišemo ili dodajemo, inače u suprotnom se vraća greška. Ovo sučelje ima dodatne dvije funkcije u odnosu na ostala sučelja, ukoliko korisnik ne zna sve svoje prijatelje, može aktivirati akciju izlistavanja prijatelja, a ukoliko ne želi ništa raditi može odabrati opciju povratka sa gumbom *Povratak*.

```
<html>
<head>
<title>Prijateljstva</title>
</head>
Ovdje unesite ime prijatelja:
<form action=\"dodavanjeBrisanjePrijatelj\" >
<input type=text name=prijatelj value=\"\">
<input type=submit name=submit value=\"Dodaj  prijatelja\" >
<input type=submit name=submit value=\"Obrisi prijatelja\" >
<input type=hidden name=\"user2\" value="+korisnik+">
</form>
<form action=\"izlistajPrijatelje\" >
<input type=submit value=\"Izlistaj prijatelje\" >
<input type=hidden name=\"user2\" value="+korisnik+">
</form>
<FORM><INPUT TYPE='button' VALUE='Povratak' onClick='history.go(-1);return true;'></FORM>
</html>
```

Slika 16. Kod za korisničko sučelje prijateljstva

Ostala sučelja

Primjenom već spomenutog, te gore opisanog koda, sva ostala sučelja su ostvarena na što jednostavniji i sličniji način. To osigurava vrlo lako baratanje programom, te korisniku olakšava izbor na samo par funkcija, te sa sigurnošću može odrediti što želi obaviti. Sva sučelja osim početnog sučelja za prijavu korisnika generirana su od strane poslužitelja, primjenjujući *Java* programski jezik. U sljedećem poglavlju biti će opisan način generiranja sučelja.

4.2.2. Java i Java servlet tehnologija

U ovom programskom ostvarenju završnog rada koristila se *Java* kao glavni programski jezik poslužiteljskog primjenskog sustava. *Java* ima vrlo široku primjenu, a s obzirom na njen široki spektar primjene predstavljala je glavni izbor prilikom izrade. Iako je isprva bila predviđena za stvaranje mobilnih primjenskih sustava, zbog svoje usavršenosti postala je raširenі programski jezik za sve vrste platformi. Prednosti *Jave* u odnosu na druge programske jezike upravo je u njenoj nezavisnosti o operacijskom sustavu i sklopovlju računala. Naime kod pisan u *Java* programskom jeziku putem prevoditelja se prevodi u bajt-kod (eng. *byte-code*), koji se može izvršavati na bilo kojem računalu koje posjeduje virtualni stroj (eng. *Java virtual machine*). Ovime smo dobili na portabilnosti koda, te je njegova primjena moguća svugdje. Također, *Javu* smo izabrali zbog njenih posebnih mogućnosti, klasa koje se bave enkripcijom i dekripcijom podataka. *Java* nam pruža vrlo visok stupanj kvalitete i organizacije kodá. [1]

Jedna od klasa *Jave* zvana *Servlet* omogućuje nam proširivanje mogućnosti poslužitelja koji poslužuju primjenske sustave putem zahtjev – odgovor modela. *Servlet* je klasa *Jave* koja se prilagođava *Java Servlet API*-u, protokolu kojim klasa *Jave* može odgovarati na zahtjeve. Najčešće se koriste za prijenos putem *HTTP* protokola, te slanje odgovora na *HTTP* zahtjev. Najčešće se uz primjenu *Java Servleta* za posluživanje primjenskih sustava koji koriste ovu klasu koristi *Apache Tomcat* poslužitelj, no o tome će biti riječi nešto kasnije. U primjeni *Servleta* najčešće se koriste metode *doGet* i *doPost*, koje imaju značajne razlike prikazane u sljedećoj tablici[9].

DoGet	DoPost
Parametrizacija ulazi u URL	Parametri se šalju posebno
Maksimalna veličina podataka 240B	Nema maksimuma za podatke
Podatci nekriptirani	Kriptirani podatci
KorisDohvat informacije sa poslužitelja	Obično za slanje podataka poslužitelju
Brža uz primjenu duljine sadržaja	Sporija jer ne koristi duljinu sadržaja

Tablica 2. Razlika između doPost i doGet metoda

4.2.2.1. Primjena Jave i Java servlet tehnologije

Svaka korisnička stranica napravljena prema modelu zahtjev-odgovor, stoga možemo odmah definirati da će sve funkcije koristiti *Java Servlet*, koji nam omogućuje upravljanje zahtjevima korisnika, te vraćanjem odgovora. Funkcijom *import* u kodu možemo označiti uključivanje određenih paketa u našu funkciju. Paketi nam služe kako bi nam osigurali dodatne funkcionalnosti, stoga će svaka naša funkcija sadržavati kod sa slike.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

Slika 17. Funkcija import

Nakon što smo definirali upotrebu *Java Servleta* u našem kodu, sada možemo primijeniti na našoj funkciji proširivanje funkcionalnosti.

```
public class LoginSkripta extends HttpServlet {
public void doPost(HttpServletRequest request,HttpServletResponse response)
 throws ServletException, IOException
```

Slika 18. Proširivanje funkcionalnosti funkcije LoginSkripta

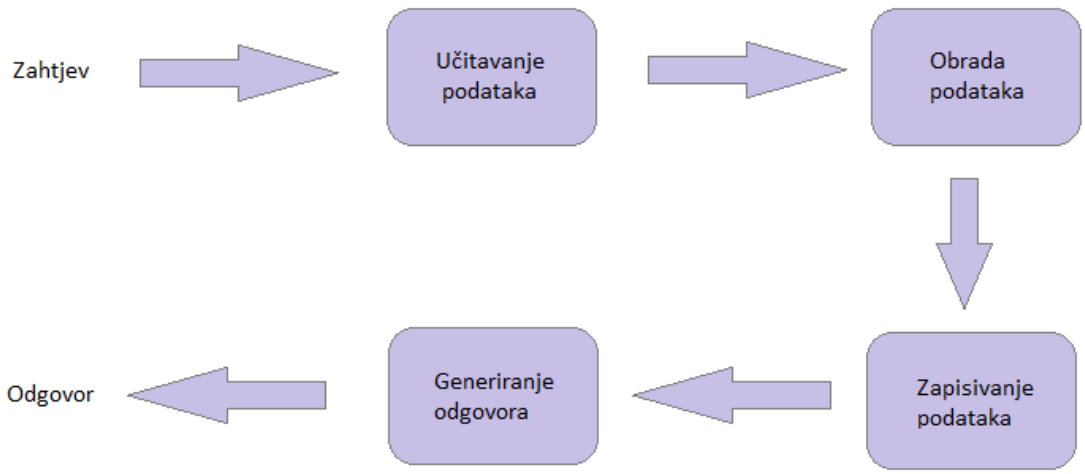
Bitno je primijetiti da se gore navedena funkcija obavlja putem *doPost* metode, no to ne znači da ćemo svaku metodu stvarati kao *doPost*, već neke i kao *doGet*.

Za sada smo definirali upotrebu *Servleta*, te *doPost* funkcije. Kako bismo definirali izlaz primjenskog sustava, odlučili smo vraćati podatke korisniku putem web stranice, no za to moramo definirati oblik ispisa. Oblik ispisa definiramo pomoću *response.setContentType* funkcije koja određuje kakvog će oblika biti ispis. U našem slučaju želimo imati *HTML* kao ispis, stoga ćemo to tako i definirati. Za konačni ispit na novostvoreni dokument, definiramo *PrintWriter*, objekt ugrađen u *Javin* paket ulaz/izlaz (eng. *I/O, input/output*). Pomoću objekta *PrintWriter* sada možemo graditi izlazne podatke, te ih slati na prazan dokument koji se zatim šalje korisniku u obliku web stranice.

Prilikom dohvaćanja podataka od strane korisnika koristi se posebna funkcija *getParameter()* koja prima argument o kojem se objektu radi. Funkcija *getParamter* je funkcija koju pruža *JavaServlet* tehnologija.

Sljedeće što smo koristili prilikom ostvarivanja ovog završnog rada su tokovi (eng. *Stream*). Dva su nam toka bila od vrlo važne koristi, *FileInputStream* i *DataInputStream*. *FileInputStream* nam služi za čitanje datoteke u njihovom najprimitivnijem obliku, dakle u zapisu jedinica i nula, kao blok bajtova. *DataInputStream* zatim preuzima te podatke te ih pretvara u nama razumljivije, podatke više razine; tekst. Također, osim za čitanje ovi tokovi nam služe i za zapisivanje podataka. Primjenu ovih tokova smo koristili za zapisivanje podataka o osobama i za zapisivanje poruka.

Nakon definiranih tokova za datoteke, potrebno je podatke iz datoteka prilagoditi programskom jeziku, te olakšati njihovo baratanje. Svi podatci su stoga pretvarani iz tekstualnog oblika iz tekstualne datoteke u objekt *String,* te su njima punjene liste. Baratanje nad listama je provedeno koristeći petlje, najviše *for* petlja. Tok podataka kroz program vrši se na sljedeći način:



Slika 19. Tok podataka kroz program

Preostalo je još pokazati definiranje kriptografskog dijela programa. Naime, *Java* već u sebi sadrži pakete koji definiraju metode i svojstva za kriptiranje raznoraznih podataka. Pri tome treba iznimno paziti na greške, jer i najmanja greška može uzrokovati krivo izračunavanje dekriptiranog teksta ili ključa kriptiranja. Kako bi izbjegli mogućnosti pogrešnog proračuna podataka u program smo definirali rukovoditelje nekih najčešćih grešaka poput rukovoditelj nepostojećeg algoritma (eng. *NoSuchAlgorithm Handler*), rukovoditelj neispravnog ključa (eng. *InvalidKey Handler*) i dr. Ovdje također vidimo definiciju kodera i dekodera, no o njima će biti riječi nešto kasnije.

```
import javax.crypto.Cipher;
import javax.crypto.BadPaddingException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import java.security.Key;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
```

Slika 20. Definicije kriptografskih paketa i rukovoditelja greškama

Korisnik prilikom autorizacije može primijetiti zahtjeve za prijateljstvom. Ukoliko želi prihvatiti neku od određenih osoba koje su zatražile prijateljstvo, između korisnika i osobe se generira tajni ključ za enkripciju podataka. Ovaj ključ vrijedi samo za ove dvije osobe, te poruke kodirane tim ključem neće moći pročitati nitko drugi osim te dvije osobe. Nakon što se generirao ključ, u datoteku obje osobe se pod *prijateljstva* upisuje zapis oblika „*prijatelj tajni_ključ*“.

Ključ je potrebno pretvoriti u zapis razumljiv za tekstualnu datoteku, te se stoga primjenjuju funkcije *getEncoded* i *encode.* Funkcija *getEncoded* dohvaća ključ u primarnom obliku, te ga funkcija *encode* koja je dio skupa funkcija *BASE64Encoder*-a, pretvara u oblik spreman za ispis u tekstualnu datoteku.

Prilikom samog procesa prate se greške, a u slučaju pojavljivanja iste, korisnika se obavijesti o grešci.

```
Key key = null;
String bytesAsString = "";
try{
KeyGenerator kg = KeyGenerator.getInstance("DES");
kg.init(56);
key = kg.generateKey();
bytesAsString=new BASE64Encoder().encode(key.getEncoded());
}

catch(NoSuchAlgorithmException e)
{
out.println("neuspjelo");
}
```

Slika 21. Kod za generiranje ključa između dvije osobe

Opisani dijelovi do sad nisu se odnosili na kriptiranje poruka. Kriptiranje poruka između dvije osobe koje žele komunicirati provodi se tako da korisnik unosi ime prijatelja s kojim želi komunicirati, te poruku koju želi poslati. Ukoliko prijatelj nije na listu prijatelja, tada se radi o nesigurnoj osobi, te joj se poruka ne šalje. Ukoliko se osoba nalazi na listi prijatelja sustav dohvaća podatak oblika „*primatelj ključ*“, te uzima posebno ključ za enkripciju podataka. Od korisnika primljeni podatak sa porukom se tada korištenjem pročitanog ključa kriptira, te se sprema poruka u obliku *primatelj_pošiljatelj.txt.* Za dekriptiranje vrijedi obratni slučaj, ukoliko korisnik je primijetio da ima novih poruka, sustav dohvaća sve poruke gdje je korisnik primatelj, te dohvaća listu svih ključeva za sve pošiljatelje. Tada se svaka poruka dekriptira sa odgovarajućim ključem te se ispisuje na zaslonu u obliku „*Pošiljatelj: poruka \n Pošiljatelj:...*“.

```
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE,key);
byte[] inputBytes = tekst.getBytes();
encryptionBytes=cipher.doFinal(inputBytes);
prijenos=new BASE64Encoder().encode(encryptionBytes);
```

Slika 22. Primjer koda za kriptiranje poruke

4.2.3. Apache tomcat servlet spremnik

Apache tomcat servlet spremnik (eng. *Apache tomcat servlet container*) je spremnik koji sadrži ostvarenje *Java Servleta* i *JavaServer* stranica (eng. *Java Server Pages*). Ovaj spremnik nam pruža mogućnosti čistog *Java* web poslužitelja (eng. *pure Java web server*), baziranog na *HTTP* protokolu. Apache tomcat sastoji se od 3 komponente:

- Catalina – Spremnik *Java Servleta*
- Coyote – *HTTP* spojnik (eng. *Connector*) za web primjenske sustave ili spremnike primjenskih sustava. Coyote je zadužen za prihvaćanje zahtjeva, te prosljeđivanje zahtjeva na obradu, a nakon toga vraćanje odgovora.
- Jasper – *JSP* stroj zadužen za prevođenje *JSP* datoteka u *JavaServlet*-e, kako bi mogli biti prihvaćeni od strane Cataline.

Neke prednosti primjene servlet spremnika su smanjeno skupljanje smeća (eng. *reduced garbage collection*), veće performanse i skalabilnost, brže parsiranje *JSP* datoteka i nativna podrška za *Windows* i *Unix* platforme. [3]

4.2.3.1. Postavke servlet spremnika

Ukoliko želimo podesiti *Apache Tomcat* za rad u poslužiteljskom načinu, moramo obaviti nekoliko postavki. Najprije je potrebno instalirati *JavaSDK,* skup razvijateljskih alata za razvoj primjenskih sustava. Nakon toga potrebno je otpaktirati sadržaj *Apache Tomcata* u željeni direktorij. Za ispravan rad potrebno je podesiti okolišne varijable (eng. *environment variables*) *JAVA_HOME* i *CATALINA_HOME*. Otvorimo kontrolnu ploču (eng. *control panel*) - > Sistem (eng. *System*) - > napredne postavke sustava (eng. *advanced system settings*) - > okolišne varijable (eng. *environment variables*). Nakon što su okolišne varijable podešene na prave vrijednosti, potrebno je pokrenuti *Apache Tomcat*. To ćemo učiniti pokretanjem datoteke *tomcat6w.exe* koja se nalazi u direktoriju *apache-tomcat-6.0.32\bin.* Ovo podešavanje je izvedeno na *Windows 7* platformi sa verzijom 6.0.32, te se ne može koristiti na drugim verzijama operacijskog sustava ili sa drugom verzijom *Apache Tomcata*.

5. Zaključak

Korisnici nerijetko imaju potrebu slati raznorazne podatke putem internetske mreže drugim korisnicima, tako da svoje podatke zaštite od vanjskih utjecaja. Najčešće se to dešava u poslovnim primjenama prilikom slanja bankovnih računa, obavljanja transakcija, kupovine na internetu i sl. Također, postoje i obični korisnici koji žele sigurno slati podatke svojim prijateljima, no ponekad su alati koji služe za to prekomplicirani kako bi ih obični korisnici shvatili ili čak nakon nekog vremena korištenja odustanu od upotrebe istih.

U ovom završnom radu izrađen je jednostavni sustav koji korisnicima omogućuje povezivanje sa ostalima, koristeći vlastitu listu prijatelja, te mogućnosti slanja poruka koje se automatski štite ukoliko ih šaljemo osobama s kojima imamo prijateljstva. Intuitinvni izbornici i lakoća upotrebe su najbitnija stvar na koju se fokusiralo, stoga korisnik na svakom izborniku ima po maksimalno dvije do tri kontrole, olakšavajući tako odabir rada. Primjenom *Jave* i *JavaServlet* tehnologija ostvarena je poslužiteljska strana primjenskog sustava, koja osim baratanjem podatcima korisniku generira sučelja pomoću kojih upravlja programom. Također primjenom alata *Apache Tomcat,* te neovisnosti *Jave* poslužitelj je moguće pokrenuti na bilo kojem računalu koje ima pristup internet mreži, pa je s time postignuta maksimalna portabilnost. Za pristup sustavu na klijentskim računalima dovoljno je imati samo odgovarajući internet preglednik, koji podržava najosnovnije *HTML* elemente.

Za ovaj rad potrebno je bilo naučiti *Javu* i primjenu *JavaServlet* tehnologija, baviti se konfiguracijom *Apache Tomcat*a, te razraditi arhitekturu programskog ostvarenja. Tijekom izrade ovog rada prikupljeno je puno znanja koje će biti korisno i u sljedećim projektima.

6. Literatura

[1] Paul Leahy: „**About.com Java Guide“,** http://java.about.com/od/gettingstarted/a/whatisjava.htm

[2] A. Dujella, M. Maretić: „**Kriptografija“**, Element, Zagreb, 2007.

[3] Chuck Caldarale: „**Apache Tomcat Wiki**“, http://wiki.apache.org/tomcat/FrontPage, 12.6.2011.

[4] „**Kriptografija**“, http://www.zsis.hr/site/Kriptografija/tabid/126/Default.aspx, 2010.

[5] „**What is social networking**“, http://www.whatissocialnetworking.com/, 10.5. 2011.

[6] „**What is business networking**“, http://www.wisegeek.com/what-is-business-networking.htm

[7] Tash Hughes: „**What is networking**“, http://www.wordconstructions.com/articles/business/networking.html

[8] PortalAlfa: „**Uvod u html**“, http://www.portalalfa.com/1/Html/uvod.htm, 7.9.2000.

[9] Oracle: „**Java servlet technology**“, http://www.oracle.com/technetwork/java/javaee/servlet/index.html