SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 246

RADNI OKVIR ZA SIGURNU KOMUNIKACIJU U POTROŠAČKOM RAČUNALNOM OBLAKU

Ružica Zec

Zagreb, lipanj 2011.

Zahvaljujem prof. dr. sc. Siniši Srbljiću i asistentu Marinu Šiliću, dipl. ing. na vodstvu, pomoći i savjetima tijekom izrade diplomskog rada.

Sadržaj

1. Uvod

S ubrzanim razvojem računalnih i telekomunikacijskih sustava pojavljuju se sve veći zahtjevi za uspostavom i održavanjem sigurnosti u tim sustavima na svim razinama, sklopovlju, operacijskim i programskim sustavima te računalnim oblacima. Potrebno je zaštititi osjetljive i vrijedne informacije i usluge tako da budu dostupne samo onim korisnicima kojima su namijenjene. To se uvelike odnosi na korisnike koji različitim programskim sustavima za komunikaciju razmjenjuju poruke preko Interneta.

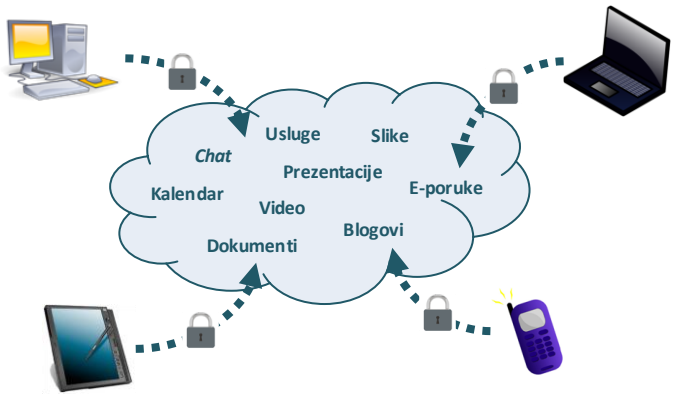
Također, rastuća popularnost različitih društvenih mreža donijela je sa sobom povećan razvoj programskih sustava različitih funkcionalnosti namijenjenih izvođenju na tim platformama. Zbog toga se uvodi potreba za ostvarenjem sigurnosnih mehanizama za zaštitu podataka koji se kroz ove programske sustave šalju ili primaju s mreže. Osim toga, računarstvo u oblacima, novi pristup u računarstvu i upotrebi računalne moći koji se sve više koristi, donijelo je dodatne zahtjeve za sigurnošću koji trebaju biti zadovoljeni. Metode za zaštitu informacija koje se koriste u različitim programskim sustavima zasnivaju se na kriptografskim algoritmima.

Programiranje prilagođeno potrošaču omogućava potrošačima, korisnicima računala koji ne posjeduju znanje o programiranju, da izgrađuju primjenske sustave. Geppeto je programski alat koji potrošačima omogućava izgradnju udomljenika, jednostavnih primjenskih sustava koji se mogu ugraditi u web stranice i druge programske sustave. Potrošači kroz grafičko sučelje izgrađuju udomljenike iz elemenata od kojih su sačinjeni postojeći udomljenici. Udomljenici mogu razmjenjivati podatke s drugim udomljenicima i programskim sustavima. Potrebno je potrošačima omogućiti da zaštite informacije koje izgrađeni udomljenici prenose preko mreže. U tu svrhu izgrađen je radni okvir koji pruža mehanizme za izgradnju sigurnih primjenskih sustava. Radni okvir sačinjavaju udomljenici koji pružaju elemente za sigurnu komunikaciju, kriptografska usluga koju ti udomljenici koriste te posrednički sustav preko kojeg se razmjenjuju poruke.

Sljedeća tri poglavlja opisuju osnovne karakteristike računarstva u oblacima, programiranja prilagođenog potrošaču i društvenih mreža. Peto poglavlje opisuje osnovne komunikacijske modele koji se koriste u raspodijeljenim sustavima, njihova obilježja i zahtjeve koje zadovoljavaju. Poseban naglasak je na komunikaciji porukama jer je to model koji je programski ostvaren. U šestom poglavlju opisani su kriptografski mehanizmi koji se koriste za uspostavu sigurnosti pri komunikaciji. Opisana su osnovna obilježja kriptografskih algoritama te algoritmi korišteni u programskom.ostvarenju U sedmom poglavlju opisan je ostvareni sustav za sigurnu komunikaciju u potrošačkom računalnom oblaku.

2. Računarstvo u oblacima

Računarstvo u oblacima (engl. cloud computing) je pristup u računarstvu u kojem se računalni resursi korisnicima pružaju preko mreže, a ne sa lokalnog računala. Lokalno računalo sadrži minimalne funkcionalnosti, na primjer operacijski sustav i internetski preglednik, koje mu omogućuju pristup udaljenim uslugama i programskim sustavima preko lokalne mreže. Osnovna karakteristika računarstva u oblacima jest da nije specificirano niti poznato na kojem računalu ili skupu računala se izvodi procesiranje. Kaže se da se usluge i programski sustavi nalaze na računalima koja su „negdje u oblacima" i otuda naziv *cloud computing*. Drugi nazivi koji se ponekad koriste jesu računarstvo na zahtjev (engl. on-demand computing), programska potpora kao usluga (engl. software as a service) te Internet[1] kao platforma (engl. the Internet as a platform). (Hayes, 2008) Slikoviti prikaz računarstva u oblacima prikazan je na slici 2.1.



Slika 2.1. Računarstvo u oblacima

Računarstvo u oblacima je relativno nov pojam u računarstvu, pojavio se 2007. godine, ali ideja koju predstavlja izgrađena je nad dobro poznatim, starijim konceptima. Predstavlja evolucijsku promjenu u pristupu i upotrebi računalne moći. (Sosinsky, 2011.) Karakteristike koje pronalazimo u autonomnom računarstvu (engl. autonomic computing), modelu klijent-

[1] Internet (pisano velikim početnim slovom) označava jedinstvenu globalnu mrežu, a internet (pisano malim početnim slovom) označava općenitu mrežu računala.

poslužitelj, spletu računala (engl. grid), *peer-to-peer* sustavima, računarstvu zasnovanom na uslugama (engl. service oriented computing) čine ujedno osnovne karakteristike računarstva u oblacima.

Računalni oblak je skup sklopovske opreme, mrežnih resursa, spremničkih prostora, usluga i sučelja koje isporučuju računalnu moć kao uslugu. Računalni oblak na korisnikov zahtjev pruža usluge isporuke programske opreme, infrastrukture i spremnika preko Interneta, ili kao čitave platforme ili kao pojedine komponente.

Računalni oblaci trebaju imati standardizirana programska sučelja (engl. application programming interface, API). Ova sučelja opisuju komunikaciju između programskih sustava i podatkovnih resursa koji međusobno komuniciraju. Standardizirana sučelja olakšavaju povezivanje potrošača s uslugama računalnih oblaka, smanjuje se potreba za dodatnim programiranjem. Računalni oblaci obično koriste API baziran na REST-u (engl. Representational State Transfer). REST opisuje programsku arhitekturu u raspodijeljenoj okolini. Ključni koncept REST-a predstavljaju sredstava, entiteti koje pronalazimo u mreži, kojima su pridijeljeni globalni identifikatori preko kojih im se pristupa korištenjem odgovarajućih metoda. (Fielding, 2000.)

Osnovna svojstva koja predstavljaju bit modela računalnih oblaka jesu samoposluga (engl. self-service), mjerenje i naplata prema upotrebi, elastičnost i prilagodljivost.

Svojstvo samoposluge računalnog oblaka označava da potrošači mogu zahtijevati, prilagođavati, plaćati i koristiti uslugu računalnog oblaka bez intervencije ljudskih operatora.

Potrošači ne moraju unaprijed reći u kojoj će mjeri koristiti uslugu, oni uslugu koriste u onoj mjeri koliko im treba. Trajanje korištenja usluge mjeri se u kratkim intervalima i ona se naplaćuje samo za ono vrijeme u kojem je korištena. Zbog toga u oblacima trebaju biti implementirane efikasne trgovinske usluge poput cjenika, računovodstva i naplate. Mjerenje upotrebe usluge se vrši na različite načine ovisno o pruženoj usluzi; pohrana podataka, procesiranje, propusnost se mjere različitim načinima.

Računalni oblaci naizgled pružaju neograničene računalne resurse. Potrošači zbog toga očekuju da u svakom trenutku mogu dobiti bilo koju količinu računalnih resursa. Oni očekuju

da će dodatni resursi biti pribavljeni (po mogućnosti automatski, bez slanja zahtjeva) kada se poveća opterećenje, odnosno otpušteni kada se opterećenje umanji. Resursi unajmljeni na računalnom oblaku moraju biti prilagodljivi u skladu s potrebama različitih potrošača. (Buyya, 2011.)

Računalni oblaci usluge pružaju na pet slojeva: klijentskom, aplikacijskom, platforma, infrastruktura i poslužiteljski sloj.

Klijentski sloj računalnog oblaka se sastoji od sklopovske opreme i/ili programske potpore koji omogućava isporuku aplikacija ili je posebno izgrađen u svrhu pružanja usluga računalnog oblaka. Primjeri klijentskog sloja su računala, telefoni i drugi uređaji, operacijski sustavi i internetski pretraživači.

Aplikacijski sloj, naziva se još i programska potpora kao usluga (engl. Software as a Service, SaaS), isporučuje programsku potporu preko Interneta. Uklanja potrebu instaliranja ili pokretanja programskoh sustava na potrošačevim računalima, pojednostavljuje održavanje i tehničku podršku.

Platforma računalnog oblaka, drugog naziva platforma kao usluga (engl. Platform as a Service, PaaS), isporučuje računalnu platformu i/ili *solution stack*, skup podsustava ili komponenti programske potpore potrebnih za isporuku potpune funkcionalnosti rješenja, proizvoda ili usluge. Pritom se često koristi infrastruktura računalnog oblaka i održavaju programski sustavi na oblaku. Olakšava postavljanje (engl. deployment) aplikacija te uklanja potrebu za upravljanjem nižim programskim slojevima i sklopovljem.

Infrastruktura računalnog oblaka ili infrastruktura kao usluga (engl. Infrastructure as a Service, IaaS) isporučuje računalnu infrastrukturu, obično virtualizaciju platforme, kao uslugu. Umjesto kupnje poslužitelja, programske opreme, podatkovnih centara ili mrežne opreme, klijenti kupuju ove resurse kao vanjsku uslugu.

Poslužiteljski sloj čini sklopovlje i/ili programska oprema koja je posebno dizajnirana u svrhe isporuke usluga računalnog oblaka. Uključuje višejezgrene procesore i posebne operacijske sustave namijenjene računalnim oblacima.

Tri su implementacije modela računalnih oblaka (engl. cloud deployment model): javni, privatni i hibridni računalni oblaci. Javnim oblacima upravljaju pružatelji koji usluge oblaka

nude svima zainteresiranima. Potrošači imaju slabu kontrolu i uvid u fizičku organizaciju i sigurnosne aspekte oblaka. Privatni oblaci nalaze se u privatnim mrežama. Svi resursi, mrežni, računalni i spremnički, se iznajmljuju samo jednoj organizaciji. Kod javnih oblaka pružatelj je odgovoran za upravljanje oblakom dok su kod privatnih oblaka za to odgovorne organizacije koje zakupljuju te oblake. Hibridni oblaci kombiniraju karakteristike javnih i privatnih oblaka.

Sigurnost u računalnom oblaku (engl. cloud computing security, cloud security) odnosi se na skup politika, tehnologija i regulacija razvijenih radi zaštite podataka, aplikacija i infrastrukture računalnog oblaka. Ključni problem vezan za sigurnost u javnim računalnim oblacima predstavlja činjenica da se aplikacije i podaci nalaze u javno dostupnom okruženju koje dijele s drugim aplikacijama i podacima. Sigurnost se može promatrati na više razina: mrežnoj, aplikacijskoj i *host* razini.

Na mrežnoj razini sigurnost u privatnim računalnim oblacima osigurava se uobičajenim sigurnosnim tehnikama koje se koriste za zaštitu privatnih mreža kao što su vatrozidi (engl. firewall), demilitarizirane zone, segmentacija mreže, sustavi za detekciju uljeza i sprječavanje napada, alati za nadgledanje mreže. U javnim računalnim oblacima potrebne su dodatne tehnike za zaštitu. Potrebno je na mrežnoj razini osigurati tajnost i integritet podataka koji se prenose prema oblaku i sa oblaka, uspostaviti pravila pristupa za sva sredstva koja se nalaze na računalnom oblaku.

Zaštita na *host* razini usko je vezana uz sloj na kojem se nude usluge (SaaS, PaaS, IaaS) i vrstu računalnog oblaka. Prijetnje na *host* razini vezane su uz virtualne mašine na kojima se izvode aplikacije i konfiguraciju sustava. (Furht, 2011)

Dodatna pažnja treba biti usmjerena na izgradnju programskih sustava koji će se izvoditi na računalnom oblaku. Potrebno je u proces razvoja programskih sustava uključiti sigurnosne mehanizme kako bi se eliminirale njihove ranjivosti. (Mather , 2010)

3. Programiranje prilagođeno potrošaču

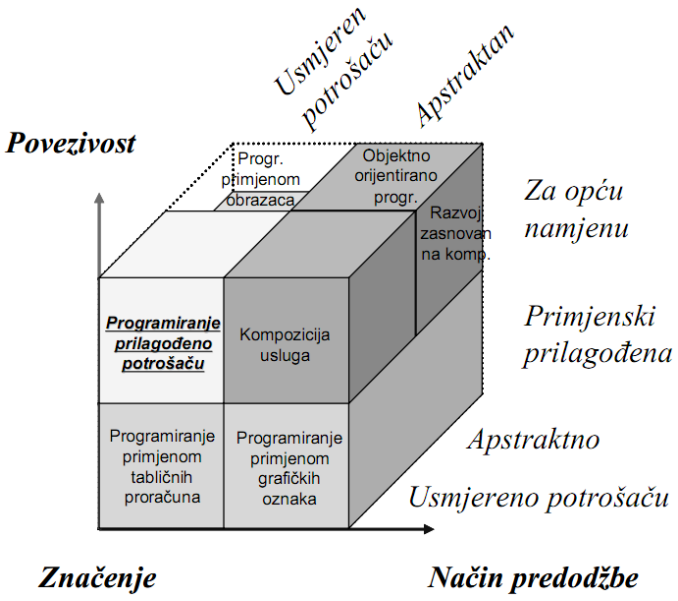
Programiranje prilagođeno potrošaču (engl. consumer programming) proučava tehnologije, postupke i alate koji potrošačima omogućavaju da samostalno izgrađuju računalne primjenske programe bez potrebe za posebnim oblicima obrazovanja.

Današnje programske paradigme i jezici se prema ciljanoj korisničkoj skupini mogu podijeliti na dvije grane: granu profesionalnog razvoja programske potpore i granu razvoja prilagođenu krajnjem korisniku (engl. end-user development, EUD). Grana profesionalnog razvoja zahtijeva obrazovane stručnjake za razvoj programske potpore. U ovoj grani koriste se programske paradigme i jezici kojima je moguće zadovoljiti većinu zahtjeva naručitelja. U grani razvoja prilagođenog krajnjem korisniku oblikuju se programske paradigme i jezici posebne namjene koje koriste stručnjaci u pojedinim granama djelatnosti za izgradnju primjenskih programa za potporu obavljanja vlastite djelatnosti. Programiranje prilagođeno potrošaču nova je grana u današnjoj programskoj paradigmi i jezicima koja programiranje pokušava približiti svim korisnicima računala bez obzira na obrazovanje i djelatnost kojom se bave. Programska paradigma prilagođena potrošaču zasniva se na izboru kognitivno prihvatljivih elemenata, tehnika i pravila za izgradnju primjenskih programa, općim znanjima stečenim u redovnom školovanju te vještinama stečenim upotrebom računala.

Značajke programiranja prilagođenog potrošaču opisuju radni okvir za vrednovanje značajki programskih elemenata od kojih se grade primjenski programi i radni okvir za vrednovanje postupaka programiranja kojima se osnovni elementi povezuju u primjenski program. Unutar radnog okvira za vrednovanje značajki programski elementi koji se koriste za izgradnju mogu se ocijeniti s obzirom na tri dimenzije:

- značenje koje elementi imaju za potrošača
- način predodžbe elemenata potrošaču
- izražajnost povezivanja programskih elemenata u složene primjenske programe.

Kod programiranja prilagođenog potrošaču značenje i način predodžbe programskih elemenata su usmjereni potrošaču (za razliku od apstraktnog), a njihovim povezivanjem moguće je graditi primjenske programe opće namjene. Radni okvir za vrednovanje značajki prikazan je na slici 3.1. Na slici su vidljive i ocjene značajki nekoliko najzastupljenijih programskih paradigmi.

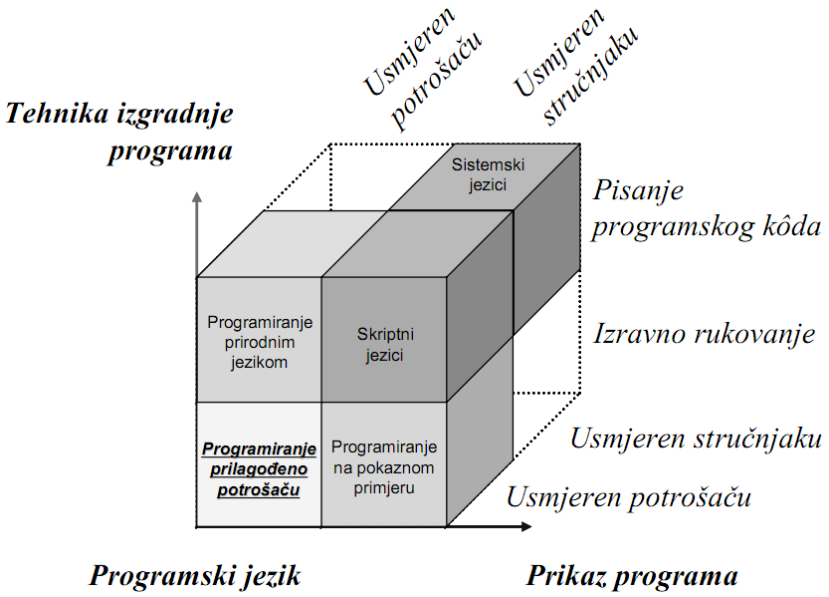


Slika 3.1. Radni okvir za vrednovanje značajki programskih elemenata (Škvorc, 2010.)

Radni okvir za ocjenu značajki postupaka programiranja ocjenjuje postupke programiranja po tri dimenzije

- svojstva programskog jezika koji definira međudjelovanje programskih elemenata kojim se ostvaruje željena funkcionalnost primjenskog programa koji se gradi
- prikaz programa
- tehnika izgradnje programa.

Kod programiranja prilagođenog potrošaču, svojstva i prikaz programa su potrošaču usmjereni (za razliku od usmjerenih stručnjacima), a tehnika izgradnje programa je zasnovana na izravnom rukovanju programskim elementima. Radni okvir za vrednovanje značajki postupaka programiranja prikazan je na slici 3.2. Na slici su vidljive i ocjene značajki nekoliko najzastupljenijih programskih jezika i tehnika programiranja.



Slika 3.2. Radni okvir za vrednnovanje značajki postupaka programiranja (Škvorc, 2010)

Model za ostvarenje programiranja prilagođenog potrošaču treba zadovoljavati zahtjeve opisane radnim okvirima za vrednovanje značajki programskih elemenata i postupaka programiranja kojima se osnovni elementi povezuju u program. Ključni čimbenici koji se koriste pri oblikovanju modela su psihologija programiranja, mjere umnog napora tijekom primjene sustava znakovlja te odnos potrošača prema tehnologiji.

Svojstva modela za ostvarenje programiranja prilagođenog potrošaču jesu:

- blokovska izgradivost primjenskog programa
- opažajni doživljaj značenja blokova
- samodostatnost programskih blokova
- višerazinsko usložnjavanje blokova
- jednakost korištenja i povezivanja blokova
- nezavisnost povezujućih elemenata i značenja blokova
- izgradivost primjenskog programa putem grafičkog korisničkog sučelja
- upravljivost i uočljivost vremenske relacije
- potpuna izgradivost logike za usložnjavanje blokova od strane potrošača
- opažajni doživljaj prikaza programa.

Blokovska izgradivost primjenskog programa označava svojstvo izgradnje primjenskih programa iz gotovih programskih blokova. Ovakva izgradnja potrošačima omogućava da djeluju na razini međusobnog odnosa pojedinih komponenti umjesto na razini njihove unutarnje građe. Novi primjenski programi izgrađuju se povezivanjem gotovih blokova iz nezavisnih primjenskih programa kako bi zadovoljili pojedinačne potrebe potrošača. Blokovi obavljaju jezgrene funkcije novog primjenskog programa, a povezujućom programskom logikom se skup blokova povezuje u radni tijek kojim se definira tok podataka i redoslijed izvođenja blokova.

Opažajni doživljaj značenja blokova vezan je uz način na koji se ti blokovi predstavljaju potrošaču. U programiranju prilagođenom potrošaču programski blokovi su samoopisivi i za razumijevanje njihove uloge u procesu izgradnje primjenskih programa nije potreban dodatni napor, školovanje ili proučavanje priručnika za upotrebu. Koristi se predodžbeni model za prikaz blokova - programski su blokovi predstavljeni kroz grafičko korisničko sučelje unutar internetskog preglednika. Na taj način potrošač može vidjeti izgled i funkcije programskih blokova za vrijeme oblikovanja novog primjenskog programa.

Samodostatnost programskih blokova označava svojstvo da oni postoje i funkcioniraju i izvan konteksta programske paradigme u kojoj se koriste za izgradnju primjenskih programa. Osim za izgradnju primjenskih programa mogu se koristiti i kao samostalni primjenski programi. Blokovi koji se koriste za izgradnju nisu oblikovani isključivo za potrebe izgradnje novih primjenskih programa

Višerazinsko usložnjavanje blokova označava svojstvo da se izgrađeni primjenski programi mogu koristiti kao osnovni primjenski programi za izgradnju novih.

Svojstvom jednakosti korištenja i povezivanja blokova zahtijeva se da se povezivanje nezavisnih blokova u novi primjenski program izvodi istim tehnikama koje se koriste pri uporabi tih blokova unutar osnovnog samostalnog primjenskog programa.

Svojstvo nezavisnosti povezujućih elemenata i značenja blokova označava da se povezivanje blokova obavlja konačnim skupom povezujućih elemenata koji su nezavisni od

primjenskih funkcija pojedinih blokova primjenskih programa koji se koriste za izgradnju. Time je omogućeno da se primjenom konačnog skupa povezujućih elemenata oblikuje beskonačno mnogo različitih uzoraka povezivanja blokova.

Izgradivost primjenskog programa putem grafičkog korisničkog sučelja opisuje svojstvo izgradnje programske logike za povezivanje blokova u primjenski program putem grafičkog korisničkog sučelja.

Upravljivost i uočljivost vremenske relacije označava da je redoslijed izvođenja programskih blokova povezanih u radni tijek pod nadzorom i utjecajem potrošača. Time je potrošaču omogućeno da definira različite vremenske relacije nad istim skupom programskih blokova i oblikuje složene uzorke izvođenja programa.

Potpuna izgradivost logike za usložnjavanje blokova od strane potrošača mjera je izražajnosti potrošaču prilagođene primjenske logike. Govori o mogućnostima povezivanja blokova u različite radne tijekove. Programska paradigma koja zadovoljava ovo svojstvo omogućava povezivanje blokova u bilo koji radni tijek. Potrošaču prilagođena programska paradigma zadovoljava ovo svojstvo.

Opažajni doživljaj prikaza programa je svojstvo koje zahtijeva da se programska logika prikazuje primjenom znakovlja i nazivlja koji su dio općeg znanja čovjeka ili se koriste u uobičajenoj primjeni računalnih primjenskih programa. Nije potrebno dodatno posebno znanje da bi se razumjelo značenje izgrađenog primjenskog programa. (Škvorc, 2010.)

4. Društvene mreže

Društvena mreža (engl. social network) je računalna usluga, platforma ili stranica koja pruža mogućnost međusobnog povezivanja ljudi, najčešće prema njihovim interesima, aktivnostima i poznanstvima u stvarnom životu. U društvenoj mreži svaki njen član posjeduje profil preko kojeg se predstavlja i povezuje s drugim članovima društvene mreže. Osim pojedinačnih korisnika društvene mreže uvelike koriste i različite organizacije koje na taj način reklamiraju svoje proizvode i usluge.

Društvene mreže trenutno nisu standardizirane. Zajedničko svim društvenim mrežama jesu profili koje kreiraju njeni korisnici. Korisnici tada mogu postavljati slike, pisati komentare koje mogu čitati drugi korisnici mreže, čitati komentare ostalih korisnika, povezivati se s ostalim korisnicima i tako stvarati listu kontakata (prijatelja), koristiti ostale usluge koje društvena mreža nudi. Svoju privatnost čuvaju tako da samo svojim kontaktima pruže mogućnost uvida u podatke koje imaju na svom profilu.

Velik problem u društvenim mrežama predstavlja privatnost, odnosno odavanje previše osobnih informacija od strane korisnika čime se mogu dovesti u opasnost. Moguće su krađe podataka i korisničkih računa čime se može izravno naštetiti korisnicima. Postoji i opravdan strah da informacije iznesene na društvenim mrežama različite korporacije i vladina tijela mogu iskoristiti protiv korisnika društvenih mreža. Velik problem predstavlja i činjenica da izmijenjeni ili obrisani podaci i dalje ostaju na poslužiteljima društvenih mreža te se potencijalno mogu proslijediti trećim osobama. Osim toga, upitno je imaju li znanstvena istraživanja koja se provode prikupljanjem podataka na društvenim mrežama pristanak korisnika da se njihovi podaci iskoriste u istraživanju. Čak i u slučaju javne dostupnosti tih podataka na društvenoj mreži ovakav način se može smatrati napadom na privatnost. Uz problem privatnosti povezana je i dubinska analiza podataka (engl. data mining) dobivenih s društvenih mreža. Analiziraju se korisničke akcije te izrađuju potrošački profili koji sadrže demografska obilježja korisnika i njihovo ponašanje na internetu. Kompanije dobivene

rezultate analize koriste za poboljšanje prodaje svojih usluga i proizvoda i time si povećavaju profit.

5. Komunikacijski modeli u raspodijeljenim sustavima

5.1. Komunikacija

Raspodijeljeni sustav (engl. distributed system) je sustav u kojem programske i sklopovske komponente sustava umreženih računala komuniciraju i usklađuju svoje aktivnosti isključivo razmjenom poruka. (Coulouris, 2005.) Raspodijeljeni sustavi su heterogeni, mogu se sastojati od različitih vrsta računala i mrežnih poveznica između računala. Pojedina računala koja sačinjavaju raspodijeljeni sustav moraju biti otporna na pogreške. Ona imaju ograničeno i nepotpuno znanje o raspodijeljenom sustavu koji sačinjavaju. Osnovu svakog raspodijeljenog sustava čini komunikacija autonomnih udaljenih procesa.

Komunikacija u raspodijeljenim sustavima jest razmjena podataka između dvaju ili više procesa. Podaci se u određenom formatu i prema određenim pravilima prenose mrežom između računala. Na mrežnoj razini komunikacija se izvodi na različitim slojevima mreže. OSI model (engl. Open Systems Interconnection model) je teorijski standard koji opisuje slojevitu mrežnu arhitekturu. Definira sedam mrežnih slojeva na kojima se izvodi komunikacija. Njegova implementacija jest TCP/IP model. U TCP/IP modelu svakom sloju pridruženi su određeni komunikacijski protokoli koji definiraju pravila komunikacije i format podataka koji se prenose na tome sloju između procesa. Svaki viši mrežni sloj koristi funkcionalnosti nižih slojeva za ostvarenje komunikacije. (Ciccarelli, 2004) Tablica 5.1. prikazuje nazive mrežnih slojeva u OSI i TCP/IP modelima.

Tablica 5.1. Slojevi OSI i TCP/IP modela

OSI model	TCP/IP model
Aplikacijski sloj	Aplikacijski sloj
Prezentacijski sloj	
Sloj sesije	
Transportni sloj	Transportni sloj
Mrežni sloj	Mrežni sloj

Podatkovni sloj	Sloj podatkovne veze
Fizički sloj	

Programski sustavi u raspodijeljenim sustavima koriste mrežne usluge aplikacijskog[2] sloja (a s time i ostalih nižih slojeva) za ostvarenje komunikacije.[3] Osnovni komunikacijski modeli za razmjenu podataka između programskih sustava (objekata, procesa) u raspodijeljenim sustavima su model klijent-poslužitelj (engl. client-server model), model objavi-pretplati (engl. publish subscribe model), dijeljeni podatkovni prostor (engl. shared dana space) te komunikacija porukama (engl. message-queuing systems, message-oriented middleware, MOM). Grupnu komunikaciju, komunikaciju između većeg broja procesa, opisuje P2P (engl. peer-to-peer) model. Ovi komunikacijski modeli imaju različita komunikacijska obilježja kojima su definirana.

5.2. Obilježja komunikacijskih modela

Komunikacija između objekata u raspodijeljenom računalnom sustavu može biti konekcijska i bezkonekcijska. U konekcijskoj se komunikaciji prije slanja podataka razmjenjuju informacije među procesima koje služe za uspostavu konekcije između procesa. U bezkonekcijskoj komunikaciji nema uspostave konekcije, odmah se šalju poruke s podacima.

Drugo obilježje komunikacije odnosi se na isporuku poruke primatelju. Ukoliko je zajamčena isporuka poruke primatelju čak i onda kada on nije aktivan prilikom slanja poruke od strane pošiljatelja, radi se o perzistentnoj komunikaciji. U perzistentnoj komunikaciji nužno je postojanje posrednika koji pohranjuje poruku u sustavu do njene isporuke primatelju. Kada je isporuka poruke primatelju zajamčena samo onda kada je primatelj aktivan prilikom slanja poruke radi se o tranzijentnoj komunikaciji. Tranzijentna komunikacija ne jamči isporuku poruke u slučaju da primatelj nije dostupan prilikom slanja.

[2] U daljnjem tekstu podrazumijeva se da se slojevi odnose na TCP/IP model.
[3] Aplikacije ponekad umjesto funkcionalnosti aplikacijskog sloja izravno koriste funkcionalnosti transportnog sloja za ostvarenje komunikacije.

Sinkronost i asinkronost predstavljaju još jedno obilježje komunikacije. U sinkronoj komunikaciji pošiljatelj nakon slanja poruke čeka potvrdu vezanu za slanje poruke. Pošiljatelj je blokiran dok ne dobije informaciju da je ili poruka poslana ili da je poruka isporučena ili da je primatelj obradio poruku i poslao odgovor. U asinkronoj komunikaciji pošiljatelj odmah nakon slanja poruke nastavlja sa svojim radom. Ne čeka nikakvu potvrdu vezanu za stanje slanja poruke. Poruka se predaje operacijskom sustavu koji se brine za njenu isporuku primatelju.

Komunikacija se može zasnivati na načelu *pull* ili *push*. U *push* komunikaciji pošiljatelj nakon slanja poruke registrira poseban proces koji osluškuje odgovor primatelja te obavještava pošiljatelja o pristiglom odgovoru. U *pull* komunikaciji nema registracije zasebnog procesa za osluškivanje odgovora.

Vremenska neovisnost procesa označava da procesi ne moraju biti istovremeno aktivni za uspostavu komunikacije. Vremenski ovisni procesi moraju biti istovremeno aktivni kako bi procesi mogli komunicirati.

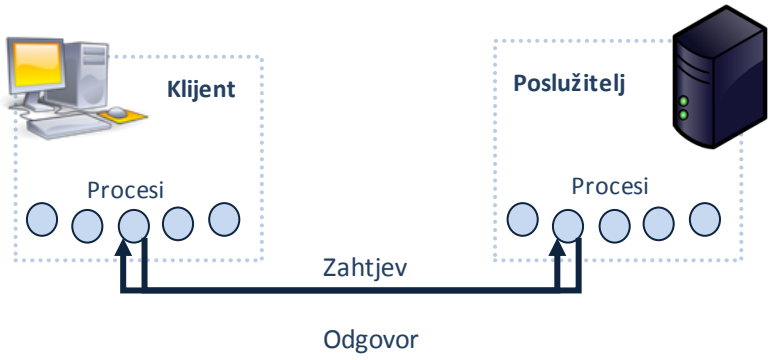
Ukoliko pošiljatelj mora znati identifikator (adresu) primatelja s kojim želi komunicirati, tada je pošiljatelj ovisan o referenci sugovornika. U suprotnom, pošiljatelj nije ovisan o referenci sugovornika.

5.3. Komunikacijski modeli

5.3.1. Komunikacijski model klijent–poslužitelj

Model klijent-poslužitelj prevladavajući je komunikacijski model u raspodijeljenim sustavima. Komunikacija se izvodi između klijenta i poslužitelja. Klijent šalje zahtjeve poslužitelju za određenom uslugom koju poslužitelj nudi. Poslužitelj osluškuje zahtjeve klijenata, prima i obrađuje njihove zahtjeve te po potrebi šalje odgovor. Ovaj model se često naziva modelom zahtjev-odgovor. Slika 5.1 prikazuje komunikaciju u modelu klijent-poslužitelj. Neka od postojećih rješenja za komunikaciju modelom klijent-poslužitelj su

komunikacija korištenjem priključnica (engl. socket communication), poziv udaljene procedure (engl. remote procedure call, RPC) te poziv udaljene metode (engl. remote method invocation).



5.1. Komunikacijski model klijent-poslužitelj

Komunikacija korištenjem priključnica na transportnom sloju može se odvijati preko komunikacijskih protokola TCP ili UDP. Komunikacija preko TCP protokola je konekcijska i osigurava pouzdan prijenos podataka, dok se preko UDP protokola prenose nezavisni paketi (datagrami) i nije osiguran pouzdan prijenos paketa. Priključnica označava komunikacijsku točku preko koje aplikacija šalje podatke mrežom i preko koje čita primljene podatke. Predstavlja višu razinu apstrakcije od komunikacijske točke koju operacijski sustav koristi za pristup transportnom sloju. Priključnica se veže uz broj vrata (engl. port) koja jednoznačno određuju aplikaciju koja izvodi komunikaciju, šalje i prima poruke. Kod konkurentnih korisničkih zahtjeva poslužitelju za svaki se novi pristigli zahtjev stvara nova priključnica s novim brojem vrata preko koje se nastavlja komunikacija korisničkog procesa i poslužitelja. Stvorene priključnice kopije su originalne koja služi osluškivanju zahtjeva. Procesi koji komuniciraju preko priključnica su vremenski ovisni, i poslužitelj i klijent moraju biti istovremeno aktivni kako bi se komunikacija ostvarila. Komunikacija je tranzijentna, isporuka se garantira samo ako su procesi istovremeno aktivni. Komunikacija koja se izvodi preko UDP-a je asinkrona, klijent nastavlja s obradom nakon slanja poruke, dok je komunikacija preko TCP-a sinkrona, klijent šalje zahtjev za stvaranje konekcije te je blokiran do uspostave

konekcije. Komunikacija preko UDP-a se može se implementirati i na načelu *pull* i *push*, dok se kod TCP-a pokreće na načelu *pull*.

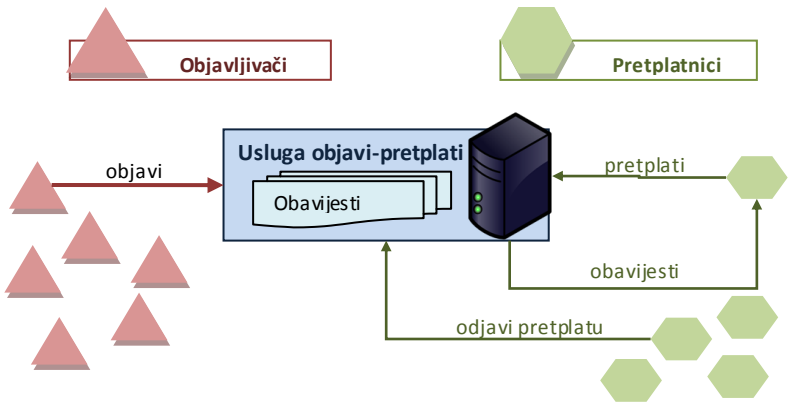
Poziv udaljene procedure je način komunikacije u kojoj proces na klijentu poziva izvršavanje procedure s određenim parametrima koja se nalazi u drugom adresnom prostoru, obično na udaljenom računalu poslužitelju, bez potrebe za dodatnim kodiranjem za uspostavu udaljene komunikacije. Nakon izvršavanja procedure poslužitelj šalje odgovor. Klijent je blokiran dok ne primi potvrdu da je poslužitelj zaprimio zahtjev ili dok ne primi odgovor. Udaljene procedure trebaju biti idempotentne, ne bi smjelo biti popratnih efekata u slučaju da je procedura zbog nepredvidljivih mrežnih problema pozvana više puta umjesto jednom. Kada su programi koji komuniciraju izgrađeni na objektno orijentiranoj paradigmi, poziv udaljene procedure se naziva poziv udaljene metode, proces umjesto procedure poziva izvođenje metode udaljenog objekta.

Slijed događaja koji se izvode pri pozivu udaljene procedure:

1. Klijentski proces poziva klijentski stub. Ovo je lokalni poziv u kojem se parametri procedure (odnosno metode kod RMI-ja) stavljaju na stog (navode se tip i vrijednost parametra).
2. Klijentski *stub* pakira parametre procedure (metode) u poruku (engl. marshalling) te izvodi sistemski poziv za slanje poruke.
3. Jezgra klijentskog sustava šalje poruku od klijenta do poslužitelja.
4. Jezgra poslužiteljskog sustava prosljeđuje poruku poslužiteljskom *stubu* (odnosno *skeletonu* u slučaju RMI-ja) koji čita parametre iz pristigle poruke (engl. unmarshaling).
5. Poslužiteljski *stub* (*skeleton*) poziva proceduru (metodu) s određenim parametrima i šalje odgovor. Odgovor procedure (metode) se šalje obrnutim slijedom od poziva procedure (metode).

5.3.2. Komunikacijski model objavi-pretplati

Komunikacijski model objavi-pretplati se sastoji od objavljivača (engl. publisher), pretplatnika (engl. subscriber) i usluge objavi-pretplati. Objavljivači generiraju obavijesti (engl. notifications), poruke određenog formata. Pretplatnici se pretplaćuju ili odjavljuju pretplate na obavijesti. Oni pretplatama definiraju svojstva obavijesti koje žele primati. Usluga objavi-pretplati brine se za obradu i pohranu obavijesti, obrađuje pristigle pretplate i odjave pretplata te isporučuje obavijesti njenim pretplatnicima. Obavijesti mogu biti definirane kao nestrukturirani tekst ili strukturirani skup parova. U slučaju obavijesti definiranih nestrukturiranim tekstom, pretplata se definira skupom ključnih riječi. Kada je obavijest strukturirani skup atributa i vrijednosti, pretplata je definirana skupom uvjeta nad atributima. Postoje dvije vrste pretplata: pretplata na kanal i pretplata na sadržaj. Kod pretplate na kanal, obavijesti se grupiraju u kanale, pretplatnici se pretplaćuju na kanale te primaju sve obavijesti s onih kanala na koje su pretplaćeni. Kod pretplate na sadržaj pretplatnici izravno definiraju svojstva obavijesti koje žele primati. Slika X.X prikazuje komunikacijski model objavi-pretplati.



Slika 5.2. Komunikacijski model objavi pretplati

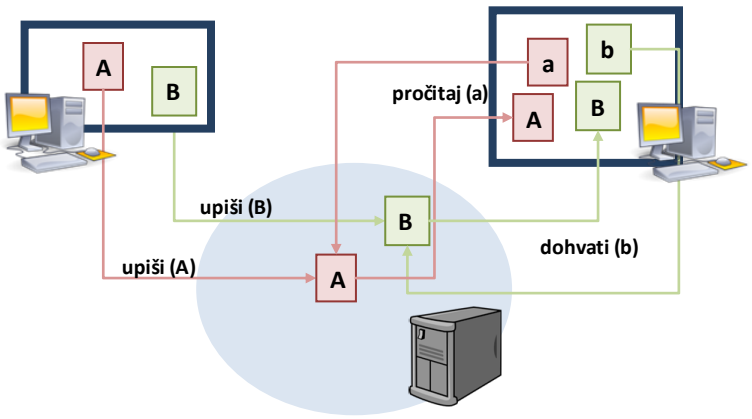
Usluga objavi-pretplati može biti centralizirane ili raspodijeljene arhitekture. Kod centralizirane arhitekture postoji jedan posrednički poslužitelj na kojem se definiraju pretplate i razmjenjuju obavijesti. Glavni nedostatci centralizirane arhitekture jesu

neskalabilnost i nepouzdanost sustava. Povećanjem broja korisnika u sustavu se smanjuju performanse, a središnji poslužitelj na kojem se nalazi usluga objavi-pretplati zbog opterećenosti svim korisničkim zahtjevima predstavlja jedinstvenu točku ispada (engl. single point of failure). U raspodijeljenoj arhitekturi usluga objavi-pretplati se nalazi na skupu posredničkih poslužitelja (engl. event brokers). Raspodijeljena arhitektura je skalabilna i nema jedinstvene točke ispada. Svaki poslužitelj je zadužen za dio objavljivača i pretplatnika. Potrebno je ostvariti algoritme za usmjeravanje obavijesti i informacija o pretplatama između posrednika prema zainteresiranosti pretplatnika. Posrednici stoga prema pretplatama i odjavama pretplata održavaju tablice usmjeravanja prema kojima odlučuju kojim posrednicima trebaju proslijediti neku obavijest. Dva su načela usmjeravanja: preplavljivanje i filtriranje poruka. Kod preplavljivanja tablica usmjeravanja sadrži informacije o svim susjednim posrednicima i pretplatnicima te sve obavijesti, pretplate i odjave pretplata prosljeđuje svim susjedima osim onome od koga je primio poruku. Kod filtriranja poruka uspoređuju se obavijesti s pretplatama te se prosljeđuju samo zainteresiranim posrednicima s ciljem smanjenja prometa u mreži posrednika.

Procesi koji komuniciraju modelom objavi-pretplati su vremenski neovisni, objavljivači i pretplatnici ne moraju biti istovremeno aktivni radi ostvarenja komunikacije. U modelu objavi-pretplati komunikacija je anonimna - objavljivači ne moraju znati identitet pretplatnika, perzistentna – obavijesti se pohranjuju na posrednicima i garantira se njihova isporuka pretplatnicima te asinkrona - objavljivači nakon slanja obavijesti nastavljaju s obradom ne čekajući odgovor. Pokretanje komunikacije je na načelu *push*, objavljivači šalju obavijesti posrednicima koji ih onda prosljeđuju pretplatnicima bez prethodnog eksplicitnog zahtjeva.

5.3.3. Komunikacijski model dijeljeni podatkovni prostor

Dijeljeni podatkovni prostor je komunikacijski model koji se temelji na zajedničkoj memoriji preko koje procesi komuniciraju. Zajedničkoj memoriji pristupaju preko mreže. Procesi mogu upisivati poruke u zajedničku memoriju te čitati ili dohvaćati poruke iz zajedničke memorije prema određenim predlošcima. Nakon dohvaćanja dohvaćena poruka se briše iz zajedničke memorije, nakon čitanja ona ostaje u zajedničkoj memoriji. Komunikacija između procesa je anonimna - temelji se na sadržaju podataka, asinkrona – proces stavlja podatak u dijeljeni podatkovni prostor i nastavlja s obradom, perzistentna – podaci su pohranjeni dok ih neki proces ne dohvati. Komunikacija se pokreće na načelu *pull*, šalje se eksplicitni zahtjev za čitanjem ili dohvatom podataka. Slika 5.3 prikazuje dijeljeni podatkovni prostor.

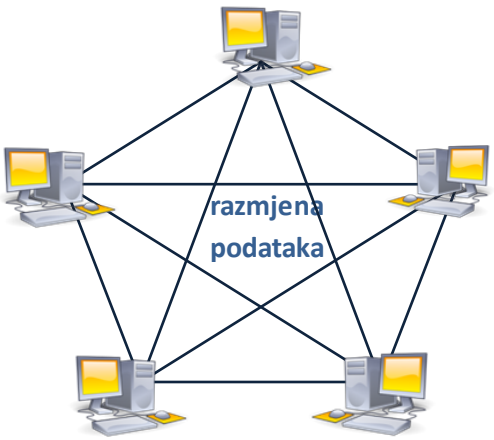


Slika 5.3. Komunikacijski model dijeljeni podatkovni prostor

5.3.4. Komunikacijski model P2P

Komunikacijski model P2P opisuje grupnu komunikaciju između većeg broja korisnika. Korisnici koji međusobno komuniciraju nazivaju se *peerovi*. Oni šalju zahtjeve drugim korisnicima i odgovaraju na zahtjeve koje primaju od njih. Zbog toga što istovremeno šalju zahtjeve i odgovaraju na primljene zahtjeve, ovaj je model suprotnost klijent-poslužitelj modela u kojem klijenti isključivo šalju zahtjeve, a poslužitelji odgovaraju na zahtjeve. Korisnici u ovim sustavima komuniciraju kako bi raspodijeljeno pohranjivali podatke te pronalazili informacije i resurse.

P2P sustavi mogu biti strukturirani i nestrukturirani. U strukturiranim sustavima povezanost korisnika izvedena je prema posebnim algoritmima kako bi se informacije koje putuju između njih efikasno usmjeravale. Najčešće se koriste raspodijeljene tablice sažetka (engl. distributed hash tables) kojima se pospješuje razmjena informacija. U nestrukturiranim sustavima nisu definirani algoritmi za organizaciju i povezanost korisnika. Slika 5.4. prikazuje decentralizirani P2P sustav.

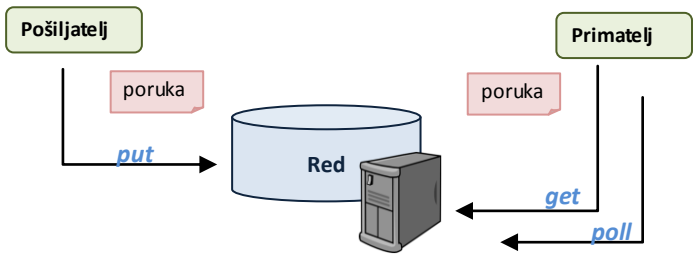


Slika 5.4. *peer-to-peer* sustav

5.4. Komunikacija porukama

Komunikacija porukama jedan je od osnovnih načina komunikacije između objekata u raspodijeljenim sustavima. Komunikacija porukama se učestalo koristi za komunikaciju među procesima unutar operacijskog sustava i unutar aplikacija.

U raspodijeljenim sustavima objekti koji komuniciraju poprimaju uloge pošiljatelja (engl. sender) i primatelja (engl. receiver) poruke. Komunikacija se odvija preko spremnika poruka, obično reda poruka (engl. message queue). Pošiljatelj stavlja poruku u red poruka, a primatelj dohvaća poruke iz reda (Slika 5.5). Spremnik poruka ima jedinstven identifikator u sustavu te je pridijeljen primatelju. Redovi poruka obično imaju ograničenu veličinu poruke koju mogu pohraniti kao i broj poruka koje se mogu staviti u red poruku. Kao spremnici poruka mogu se koristiti i poštanski sandučići (engl. mailbox). Razlika između redova poruka i poštanskih sandučića nije jasno i strogo definirana, oni svoju definiciju poprimaju u implementacijama. Ponekad označavaju istu vrstu spremnika, a u slučaju različitih implementacija osnovna se razlika najčešće odnosi na veličinu poruka koje se pohranjuju, kod redova je veličina ograničena, kod sandučića nije.



Slika 5.5. Komunikacija porukama

Komunikacija porukama je vremenski neovisna, objekti koji komuniciraju ne moraju biti istovremeno aktivni prilikom komunikacije. Pošiljatelj može staviti poruku u red poruka i kada je primatelj neaktivan. Vrijedi i obratno, primatelj može dohvatiti poruku i kada je pošiljatelj

neaktivan. Nužno je da postoji red poruka pridijeljen primatelju koji je jednoznačno identificiran u sustavu, a kojem pošiljatelj i primatelj mogu pristupiti.

Komunikacija porukama je perzistentna, garantira se isporuka poruke primatelju i kada je on neaktivan. Poruka se pohranjuje u redu poruka sve dok primatelj ne postane aktivan i ne dohvati poruku iz reda poruka.

Komunikacija porukama je asinkrona komunikacija. Pošiljatelj poruke nakon slanja poruke nastavlja sa radom ne čekajući odgovor od primatelja.

Pokretanje komunikacije je na načelu *pull*. Pošiljatelj šalje poruku, a primatelj dohvaća poruku iz svoga reda poruka. Pošiljatelj ne registrira posebne procese za osluškivanje odgovora od primatelja.

6. Uspostava sigurne komunikacije kriptografijom

6.1. Uspostava sigurne komunikacije

Sigurna komunikacija je komunikacija u kojoj dva objekta (računala, procesi, osobe) razmjenjuju informacije uz određeni stupanj sigurnosti da informacije koje se prenose nisu dostupne trećim objektima. Osnovni zahtjevi za sigurnu komunikaciju su:

- povjerljivost (tajnost, engl. confidentiality) - informacije su pristupačne samo ovlaštenim korisnicima
- raspoloživost (engl. availiability) – informacije moraju uvijek biti dostupne korisnicima, unatoč neočekivanim događajima poput nestanka struje, prirodnih nepogoda, nesreća ili zlonamjernih napada
- besprijekornost (engl. integrity) – informacije mogu mijenjati samo ovlašteni korisnici, prenose se u izvornom i nepromijenjenom obliku
- autentičnost (engl. authenticity) – moguće je jednoznačno prepoznavanje korisnika
- autorizacija (engl. authorisation) – pravima pristupa korisnicima je dozvoljen pristup samo do određenog sadržaja
- neporecivost (engl. non-repudiation) – korisnik ne može opovrgnuti slanje podataka koje jest poslao.

Postoje različiti mehanizmi za uspostavu sigurne komunikacije. Ti sigurnosni mehanizmi nužno ne zadovoljavaju sve sigurnosne zahtjeve. Tri su osnovne kategorije u koje se mogu svrstati načini uspostave sigurne komunikacije: prikrivanje sadržaja koji se prenosi među objektima, prikrivanje objekata koji komuniciraju te prikrivanje postojanja komunikacije među objektima. Prikrivanje sadržaja se može obaviti kriptografijom, kodiranjem, steganografijom. Kodiranjem se informacije pretvaraju u drugi prikaz prema određenom skupu pravila. Steganografskim tehnikama se tajna poruka skriva unutar neke druge bezazlene poruke. Prikrivanje objekata koji komuniciraju se ostvaruje preko anonimnih

posredničkih poslužitelja, metodama usmjeravanja koje otežavaju praćenje puta poruke, anonimnim komunikacijskim uređajima, anonimnim grupama u kojima je teško odrediti tko je proizveo koju poruku. Prikrivanje postojanja komunikacije smože se ostvariti generiranjem dodatnog slučajnog prometa unutar kojeg se šalje poruka.

6.2. Kriptografski algoritmi

Kriptografija je jedan od načina uspostave sigurne komunikacije u raspodijeljenim sustavima koji se zasniva na prikrivanju informacija koji se prenose. Sigurnost postiže pretvaranjem informacija u oblik nečitljiv objektima koji ne sudjeluju u komunikaciji (Slika 6.1).

Slika 6.1. Uspostava sigurnosti kriptografijom

U kriptografiji se izvorni oblik informacija koje želimo prenijeti do odredišta naziva razgovijetnim ili jasnim tekstom (engl. plaintext, cleartext). Kriptiranjem (engl. encryption, enciphering) se jasni tekst prevodi u kriptirani tekst (engl. ciphertext). Dekriptiranje (engl. decryption, deciphering) je obrnuti postupak kriptiranja kojim se kriptirani tekst prevodi u jasni tekst. Parametar koji omogućava kriptiranje odnosno dekriptiranje naziva se ključ kriptiranja (engl. encryption key) odnosno ključ dekriptiranja (engl. decryption key).

Komunikacijskim kanalima prenosi se kriptirani tekst čime su zadovoljeni svi sigurnosni zahtjevi osim raspoloživosti.

Uz oznake:

$P - jasni\ tekst$

$C - kriptirani\ tekst$

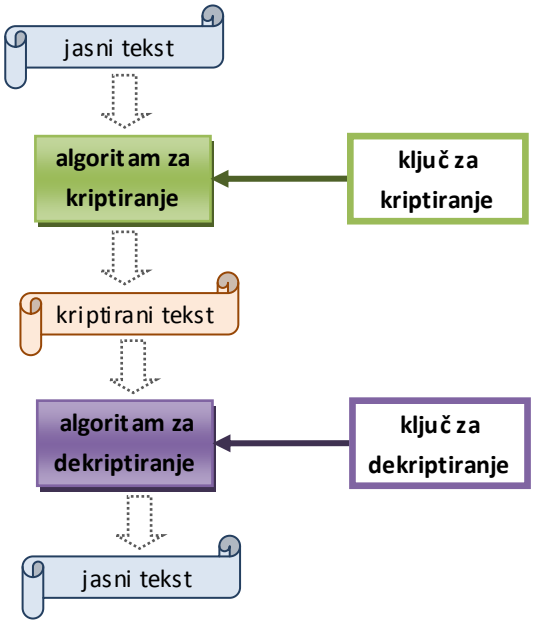
$E - funkcija\ kriptiranja$

$D - funkcija\ dekriptiranja$

$K_E -$ $parametar\ ili\ ključ\ kriptiranja$

$K_D -$ $parametar\ ili\ ključ\ dekriptiranja$

funkcija kriptiranja se može zapisati kao $C = E(P, K_E)$, a funkcija dekriptiranja kao $P = D(C, K_D)$. Funkcija dekriptiranja je inverzna funkciji kriptiranja pa vrijedi da dekriptiranjem kriptiranog teksta dobivamo početni jasni tekst $P = D(E(P, K_E), K_D)$. Funkcije kriptiranja i dekriptiranja zajedno s algoritmima za generiranje ključeva za kriptiranje i dekriptiranje čine kriptosustav (engl. cryptosystem). Slika 6.2 prikazuje arhitekturu općenitog kriptosustava.



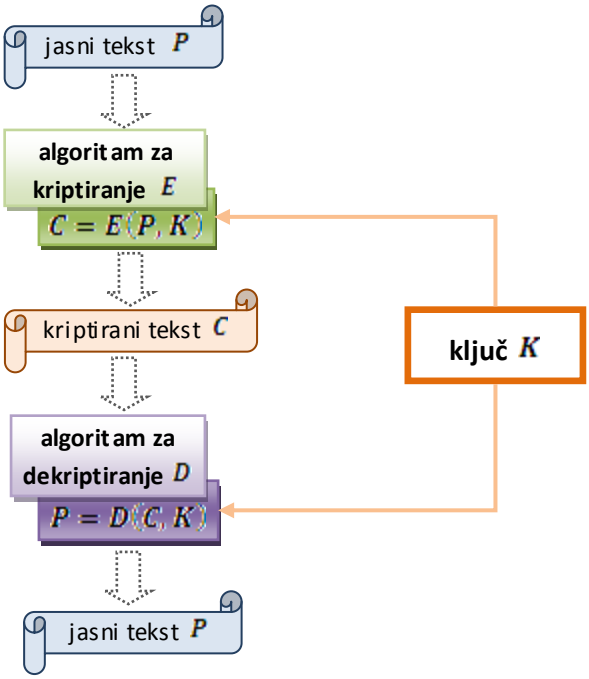
Slika 6.2. Arhitektura općenitog kriptosustava

Kriptosustav pretvara nesigurni komunikacijski kanal (engl. unsecure channel) kojim se prenose informacije u sigurni komunikacijski kanal (engl. trusted channel). Informacije koje se prenose moći će pročitati samo odredište koje će kriptirani tekst dekriptirati ključem za dekriptiranje prevodeći ga time u jasni tekst. Uljez može presresti kriptirani tekst koji se prenosi, ali zato što ne posjeduje ključ dekriptiranja neće moći dekriptirati kriptirani tekst i time dobiti uvid u informacije koje se prenose. Danas postoje dva osnovna oblika kriptosustava, simetrični i asimetrični. Kod simetričnih kriptosustava ključ za kriptiranje jednak je ključu za dekriptiranje. Kod asimetričnih kriptosustava ti su ključevi različiti.

Kriptoanaliza (engl. cryptoanalysis) je znanstvena disciplina koja se bavi proučavanjem postupaka otkrivanja otvorenog teksta bez poznavanja ključa za dekriptiranje. Kriptoanaliza se najčešće svodi na pronalaženje ključa za dekriptiranje. Težina pronalaženja toga ključa određuje dobrotu kriptosustava. Što je veći broj mogućih ključeva to ga je teže otkriti te je tada i dobrota sustava veća.

6.2.1. Simetrični kriptosustavi

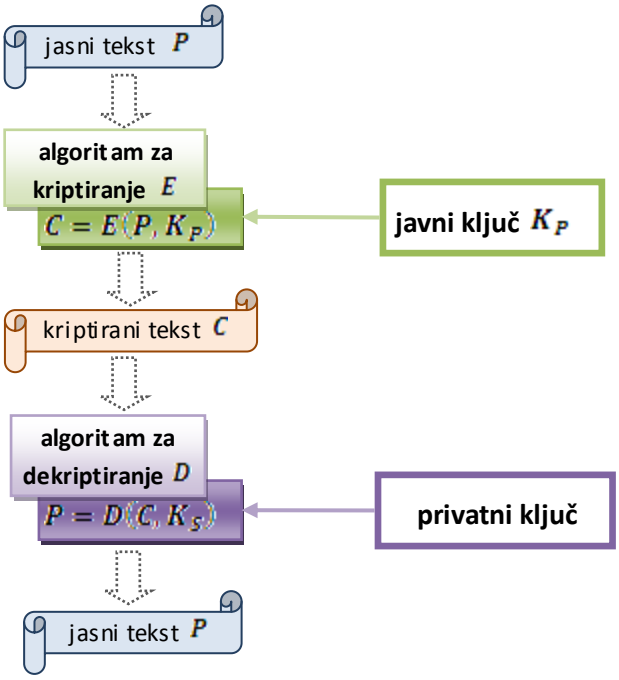
Simetrični kriptosustav koristi isti ključ za kriptiranje i dekriptiranje poruke. Kriptiranje i dekriptiranje se izvode na razini bitova, najčešće korištenjem logičke operacije *isključivo ili* (engl. exclusive or, XOR). Dvije su osnovne vrste algoritama za kriptiranje u simetričnim kriptosustavima: kriptiranje toka podataka (engl. stream cipher) i kriptiranje bloka podataka (engl. block cipher). Algoritmi za kriptiranje toka podataka kriptiraju bit po bit poruke. Algoritmi za kriptiranje bloka podataka uzimaju veći broj bitova i kriptiraju ih kao jednu cjelinu. Slika 6.3 prikazuje postupak kriptiranja u simetričnom kriptosustavu.



Slika 6.3. Simetrični kriptosustav

6.2.2. Asimetrični kriptosustavi

Asimetrična kriptografija još se naziva i kriptografijom javnog ključa (engl. public-key cryptography). Asimetrični kriptosustavi imaju različite ključeve za kriptiranje i dekriptiranje. Temelje se na određenim svojstvima brojeva koja proučava teorija brojeva (engl. number theory). Kriptiranje i dekriptiranje se izvode nad prirodnim brojevima: bitovi koji sačinjavaju poruku se kodiraju kao niz prirodnih brojeva. Kriptiranjem se niz prirodnih brojeva jasnog teksta prevodi u niz prirodnih brojeva kriptiranog teksta korištenjem određenog asimetričnog algoritma i ključa za kriptiranje. Dekriptiranjem se provodi obrnuti postupak kriptiranja. Iz niza prirodnih brojeva kriptiranog teksta se odgovarajućim algoritmom i odgovarajućim ključem za dekriptiranje određuje izvorni niz prirodnih brojeva jasnog teksta. Slika 6.4 prikazuje asimetrični kriptosustav.



Slika 6.4. Asimetrični kriptosustav

Neki osnovni pojmovi i činjenice iz teorije brojeva[4] na kojima se zasniva kriptografija javnog ključa.

- Djeljivost brojeva. Broj a djeljiv je brojem d kada je a višekratnik od d. Označava se kao $d|a$ ili $a = k * d$ gdje je k cijeli broj. Najmanji djelitelj od a je $d = 1$, a najveći je $d = a$. To su trivijalni djelitelji. Netrivijalni djelitelji zovu se faktori.
- Prosti (prim) brojevi. Broj koji je veći od jedan, a koji nema faktore (ima samo djelitelje 1 i a) zove se prosti ili prim broj.
- Teorem o dijeljenju s ostatkom. Za cijeli broj a i proizvoljan prirodan broj n postoje jedinstveni cijeli brojevi q i r takvi da je $a = q * n + r$ gdje je $0 \leq r < a$. q se naziva kvocijent (količnik), a r ostatak ili reziduum.
- Ekvivalentnost po modulu (kongruentnost). Broj a je ekvivalentan broju b po modulu n ako vrijedi $a \bmod n = b \bmod n$. Još se kaže da se a i b kongruentni po modulu n i piše se $a \equiv b \pmod n$.
- Relativno prosti brojevi. Cijeli brojevi a i b su relativno prosti ako je najveći zajednički djelitelj (nzd) brojeva a i b jednak 1. Cijeli brojevi $a_1, a_2, \ldots, a_n$ su relativno prosti ako je najveći zajednički djelitelj brojeva $a_1, a_2, \ldots, a_n$ jednak 1.
- Eulerova phi funkcija. Neka je $Z_n = \{0, 1, 2, \ldots, n-1\}$ prsten u kojem su definiranje operacije zbrajanja, oduzimanja i množenja po modulu n. Neka je Z_n^* podskup elemenata skupa Z_n koji su relativno prosti u odnosu na n: $Z_n^* = \{a \in Z_n, nzd(a,n) = 1\}$. Broje elemenata skupa Z_n^* jednak je Eulerovoj phi ili $totient$ funkciji $\varphi(n)$.

  Ako je $n = p$ prosti broj, onda je $\varphi(n) = p - 1$.

  Ako je $n = p * q$, gdje su p i q prosti brojevi, onda je $\varphi(n) = (p-1)(q-1)$.

[4] Ovdje su sažeto navedene činjenice iz teorije brojeva koje se koriste u kriptografiji javnog ključa. Za dodatna pojašnjenja pogledati (Dujella, 2009.).

Ako n ima rastav na proste faktore $n = p_1^{e_1} * p_2^{e_2} * \dots * p_k^{e_k}$, onda je $(n) = n * \left(1 - \frac{1}{p_1}\right) * \left(1 - \frac{1}{p_2}\right) * \dots * \left(1 - \frac{1}{p_n}\right)$.

- Modularno potenciranje. Potenciranje s velikim eksponentima efikasno se može obaviti uzastopnim kvadriranjem. Izračun $d = b^a \ mod \ n$ može se obaviti sljedećim algoritmom. Pri tome je $a_m, a_{m-1}, a_{m-2}, \dots, a_1, a_0$ binarni prikaz broja a.

```
d = 1;
i = m;
dok je (i >= 0) {
  d = (d*d) mod n;
  ako je (a[i] == 1) {
    d = (d*b) mod n;
  }
  i--;
}
```

6.2.2.1.RSA

RSA je najraširenији asimetrični kriptosustav. Sustav su razradili Ron Rivest, Adi Shamir i Len Adleman i po početnim slovima njihovih prezimena sustav je dobio ime. RSA je prvi algoritam koji je bio jednako pogodan za digitalno potpisivanje kao i za kriptiranje. RSA se učestalo koristi u protokolima za elektroničko poslovanje. Vjeruje se da je uz dovoljno dugačak ključ RSA kriptosustav dovoljno siguran.

RSA algoritam se sastoji od tri koraka: generiranje para privatnog i javnog ključa, kriptiranje i dekriptiranje.

Generiranje ključeva.

1. Odabiru se dva velika prosta broja p i q ($p > 10^{100}, q > 10^{100}$).

   Zbog sigurnosnih razloga ovi bi brojevi trebali biti odabrani slučajno te je poželjno da budu jednake duljine u bitovima.
2. Računa se umnožak $n = p * q$.

   n se pri kriptiranju i dekriptiranju koristi kao modul javnog odnosno privatnog ključa.

3. Računa se umnožak $\varphi(n) = (p-1) * (q-1)$.
4. Odabire se broj e za koji vrijedi $1 < e < \varphi(n)$ i koji je relativno prost u odnosu na $\varphi(n)$ (nema zajedničkih faktora sa $\varphi(n)$).

   e se koristi kao eksponent javnog ključa.
5. Računa se broj $d < \varphi(n)$ tako da vrijedi $e * d \equiv 1 \ (mod\ \varphi(n))$ što se može zapisati kao $e * d = k * \varphi(n) + 1$.

   d se koristi kao eksponent privatnog ključa.

Par $K_E = (e, n)$ je javni ključ korisnika koji se obznanjuje.

Par $K_D = (d, n)$ je privatni ključ korisnika koji se taji.

Kriptiranje se obavlja funkcijom kriptiranja na sljedeći način:

$$C = E\ (P, K_E) = RSA(P, K_E) = P^e\ (mod\ n).$$

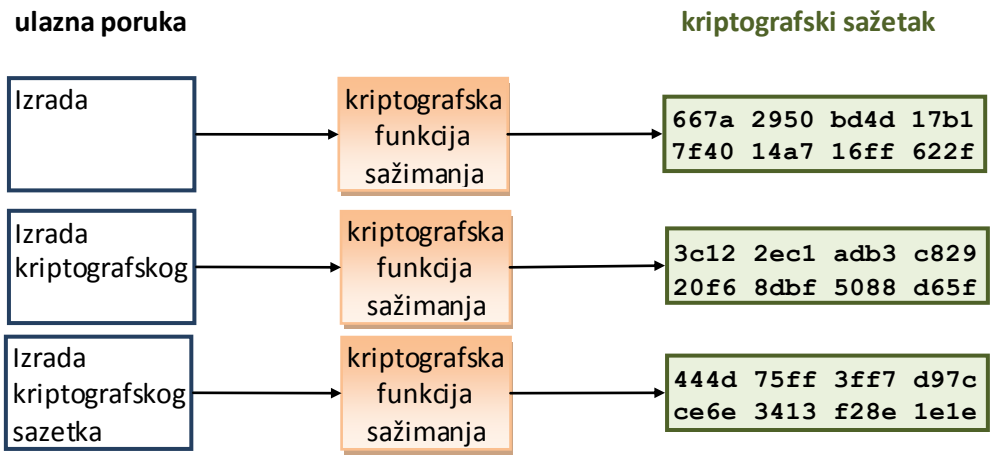
Dekriptiranje se izvodi funkcijom dekriptiranja:

$$P = D\ (P, K_D) = RSA^{-1}(C, K_D) = C^d\ (mod\ n).$$

Kriptiranje i dekriptiranje se mogu obaviti algoritmom modularnog potenciranja.

6.3. Kriptografski sažetak

Kriptografski sažetak (engl. message digest) je blok bitova fiksne duljine koji se dobiva kriptografskim funkcijama sažimanja (engl. cryptographic hash function). Funkcije sažimanja uzimaju poruke proizvoljne duljine, vrše skup određenih operacija nad bitovima poruke te proizvode sažetak. Promjena samo jednog bita ulazne poruke uzrokuje stvaranje drukčijeg sažetka. Funkcije sažimanja su jednosmjerne, jednostavno je odrediti sažetak poruke, ali je praktički nemoguće iz sažetka odrediti izvornu poruku. Slika 6.5 prikazuje stvaranje kriptografskog sažetka različitih poruka.



Slika 6.5. Stvaranje kriptografskog sažetka poruke

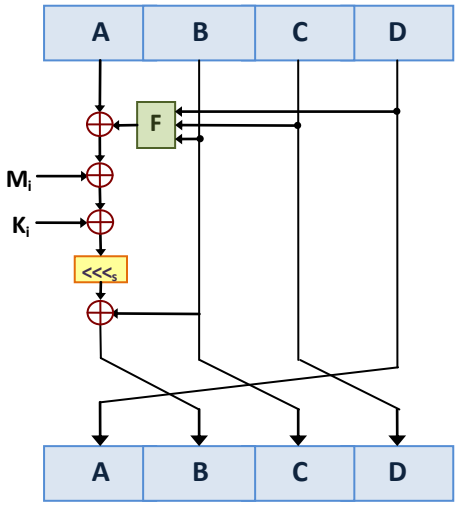
Osim što funkcije sažimanja moraju biti otporne na izračunavanje izvorne poruke iz sažetka (engl. preimage resistance), one trebaju biti otporne na izračunavanje poruke koja daje isti sažetak kao neka druga poruka (engl. second preimage resistance) te ne smije biti moguće pronaći dvije različite poruke koje daju isti sažetak (engl. collision resistance). Ova svojstva moraju biti zadovoljena unatoč tome što zbog ograničene duljine sažetka te neograničenog broja poruka znamo da postoji neograničen broj poruka s istim sažetkom.

Kriptografski sažeci imaju široku primjenu u održavanju sigurnosti računalnih sustava, ponajviše u autentifikaciji. Koriste se u izradi digitalne omotnice, potpisa i pečata.

6.3.1. MD5

MD5 (engl. message digest) je kriptografska funkcija sažimanja čiji je sažetak duljine 128 bita. Koristi se u svrhu očuvanja integriteta podataka koji se šalju.

Izvorna poruka dijeli se na blokove duljine 512 bita. Zadnji blok se nadopunjuje na sljedeći način: doda se jedinica, iza nje se stave nule osim na zadnja 64 bita - na zadnja 64 bita upisuje se duljina izvorne poruke u bitovima (bez nadopunjujućih bitova). Svaki blok se dijeli na 16 podblokova duljine 32 bita: $M_0, M_1, M_2, \dots, M_{15}$. Izračunavanje sažetka se obavlja u 64 koraka podijeljena u četiri kruga za svaki blok. Jedan korak izračunavanja sažetka sa MD5 funkcijom prikazan je na slici 6.6 ($\oplus$ označava xor funkciju, a $<<<_s$ rotaciju ulijevo za s bitova).



Slika 6.6. Jedan korak u izračunavanju MD5 sažetka

MD5 algoritam.

- sažetak se sastoji od četiri nadovezane 32-bitne konstante A, B, C, D koje se početno inicijaliziraju konstantama: $A_0 = 01234567_{16}$, $B_0 = 89ABCDEF_{16}$, $C_0 = FEDCBA98_{16}$, $D_0 = 76543210_{16}$
- u svakom $koraku\ i\ (1 \leq i \leq 16)$ se pri izračunavanju koristi 32-bitna $konstanta\ K_i$ koja se računa kao $K_i = 2^{32} * abs(sin(i))$

- rezultat prethodne iteracije je ulaz u sljedeću iteraciju

1. krug $(1 \leq i \leq 16)$
    - koristi se nelinearna funkcija: $F_i(X,\ Y,\ Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$
    - redoslijed dovođenja blokova: $M_0,\ M_1,\ M_2, \dots,\ M_{15}$
    - rotacije ulijevo za $s = 7, s = 12, s = 17, s = 22$ u četiri uzastopna koraka, četiri puta se ponavlja
2. krug $(17 \leq i \leq 32)$:
    - koristi se nelinearna funkcija: $F_i(X,\ Y,\ Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$
    - redoslijed dovođenja blokova: $M_1,\ M_6,\ M_{11},\ M_0, M_5,\ M_{10},\ M_{15},\ M_4, M_9,\ M_{14},\ M_3,\ M_8, M_{13},\ M_2,\ M_7,\ M_{12}$
    - rotacije ulijevo za $s = 5, s = 9, s = 14, s = 20$ u četiri uzastopna koraka, četiri puta se ponavlja
3. krug $(33 \leq i \leq 48)$:
    - koristi se nelinearna funkcija: $F_i(X,\ Y,\ Z) = X \oplus Y \oplus Z$
    - redoslijed dovođenja blokova: $M_5,\ M_8,\ M_{11},\ M_{14}, M_1,\ M_4,\ M_7,\ M_{10}, M_{13},\ M_0,\ M_3,\ M_6, M_9,\ M_{12},\ M_{15},\ M_2$
    - rotacije ulijevo za $s = 4, s = 11, s = 16, s = 23$ u četiri uzastopna koraka, četiri puta se ponavlja
4. krug $(49 \leq i \leq 64)$:
    - koristi se nelinearna funkcija: $F_i(X,\ Y,\ Z) = Y \oplus (X \vee \neg Z)$
    - redoslijed dovođenja blokova: $M_0,\ M_7,\ M_{14},\ M_5, M_{12},\ M_3,\ M_{10},\ M_1, M_8,\ M_{15},\ M_6,\ M_{13}, M_4,\ M_{11},\ M_2,\ M_9$

- rotacije ulijevo za $s = 6, s = 10, s = 15, s = 21$ u četiri uzastopna koraka, četiri puta se ponavlja

Konačnim vrijednostima varijabli A, B, C, D se pribrajaju konstante A_0, B_0, C_0, D_0: $A = A_{64} + A_0$, $B = B_{64} + B_0$, $C = C_{64} + C_0$, $D = D_{64} + D_0$. Uzastopni slijed rezultata zbroja $A\ B\ C\ D$ čini 128-bitni MD5 sažetak.

6.4. Digitalna omotnica, potpis i pečat

Prilikom razmjene poruke između korisnika postoji više načina na koji se ta poruka može zaštiti upotrebom kriptografskih algoritama. Ovisno o odabranom načinu zadovoljeni su različiti skupovi sigurnosnih zahtjeva.

Prvi način razmjene poruka zove se digitalna omotnica (engl. digital envelope). Slanje poruke se izvodi na sljedeći način:

- simetričnim algoritmom SA i proizvoljnim tajnim ključem K pošiljatelj kriptira izvornu poruku P koju želi poslati: $C_1 = SA\ (P, K)$.
- asimetričnim algoritmom AA i javnim ključem primatelja K_{EB} kriptira se tajni ključ K: $C_2 = AA(K, K_{EB})$.
- šalje se poruka $M = (C_1, C_2)$.

Nakon primitka poruke, primatelj izvodi sljedeće operacije:

- primatelj dekriptira poruku C_2 svojim privatnim ključem K_{DB} koristeći odgovarajući asimetrični algoritam i time otkriva tajni simetrični ključ K: $K = AA^{-1}(C_2, K_{DB}) = AA^{-1}(\ AA(K, K_{EB}), K_{DB})$
- tajnim ključem K koristeći odgovarajući simetrični algoritam dekriptira C_1 i otkriva izvornu poruku: $P = SA^{-1}(C_1, K) = SA^{-1}(SA\ (P, K), K)$.

Poruka M koja se prenosi naziva se digitalna omotnica. Digitalnom omotnicom zadovoljen je zahtjev tajnosti poruke.

Digitalni potpis (engl. digital signature) zadovoljava sigurnosni zahtjev očuvanja besprijekornosti poruke. Slanje poruke se izvodi na sljedeći način:

- funkcijom sažimanja H računa se sažetak poruke P
- asimetričnim algoritmom AA i privatnim ključem pošiljatelja K_{DA} kriptira se sažetak poruke $H(P)$
- šalje se poruka $M = (P, AA(H(P), K_{DA}))$.

Po primitku poruke primatelj izvodi sljedeće operacije:

- asimetričnim algoritmom AA i javnim ključem pošiljatelja K_{EA} dekriptira kriptirani sažetak: $H(P) = AA^{-1}(AA(H(P), K_{DA}),\ K_{EA})$
- računa sažetak primljene poruke P , uspoređuje ga s dekriptiranim sažetkom.

Ukoliko su izračunati sažetak i dekriptirani sažetak jednaki primatelj je siguran da poruka nije mijenjana na putu te da ju je poslala upravo osoba koja tvrdi da ju je poslala. Uz besprijekornost su dakle zadovoljeni zahtjevi autentičnosti i neporecivosti.

Digitalni pečat (engl. digital stamp) je digitalno potpisana digitalna omotnica. Slanje digitalno zapečaćene poruke izvodi se na sljedeći način:

- izrađuje se digitalna omotnica $(C_1,\ C_2)$
- sažetak digitalne omotnice $S = H(C_1, C_2)$ se kriptira privatnim ključem pošiljatelja K_{DA}: $C_3 = AA(S,\ K_{DA})$
- šalje se poruka $M = (C_1,\ C_2,\ C_3)$

Po primitku poruke izvode se sljedeće operacije:

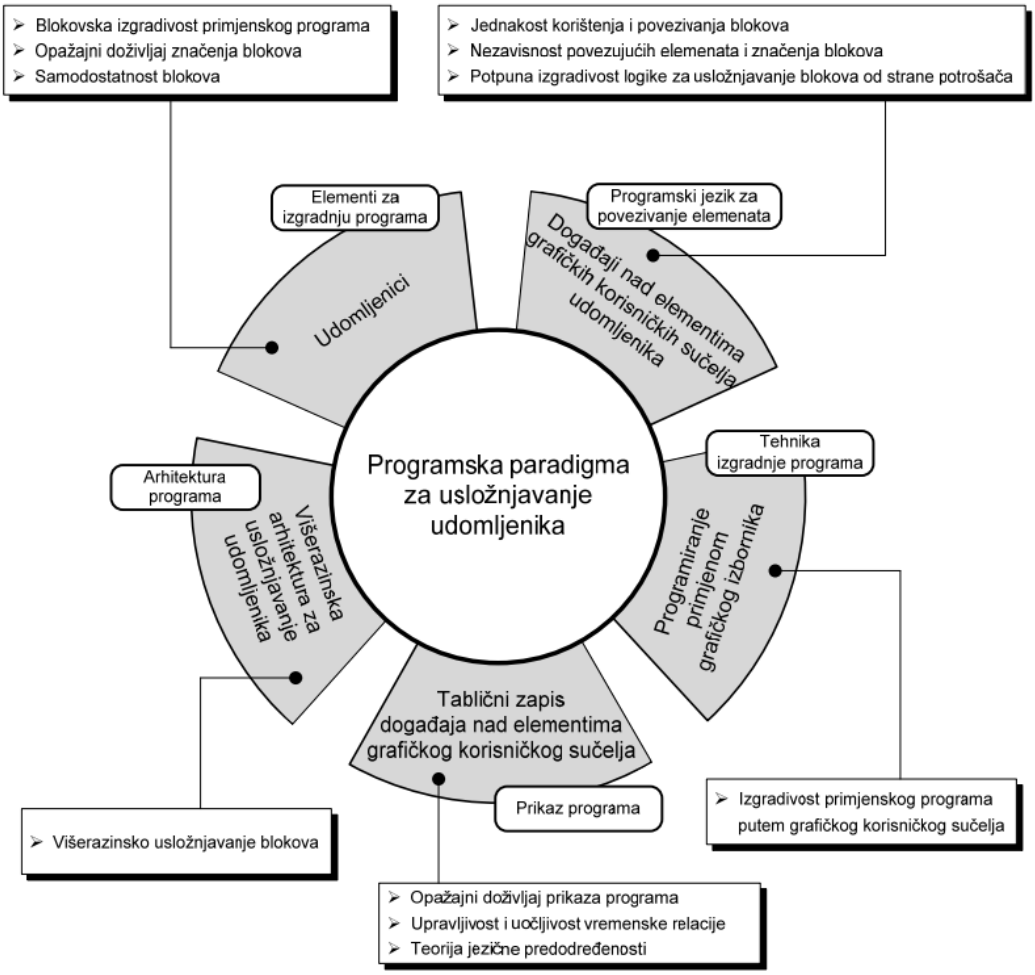
- otvara se digitalna omotnica i saznaje izvorna poruka P
- dekriptira se kriptirani sažetak digitalne omotnice javnim ključem pošiljatelja K_{EA}: $S = AA^{-1}(C_3,\ K_{DA}) = AA^{-1}(AA(S,\ K_{DA}),\ K_{DA})$

- računa se sažetak primljene digitalne omotnice i uspoređuje sa dekriptiranim sažetkom

Ako su izračunati sažeci jednaki primatelj je siguran da su zadovoljeni svi sigurnosni zahtjevi osim raspoloživosti. (Budin, 2010.)

7. Sustav za sigurnu komunikaciju u potrošačkom računalnom oblaku

Sustav za sigurnu komunikaciju u potrošačkom računalnom oblaku izgrađen je u svrhu ostvarenja sigurnosnih mehanizama unutar potrošačkog programskog alata Geppeto (Gadget Parallel Programming Tool). Geppeto je potrošačka programska okolina za izgradnju usloženih primjenskih programa, udomljenika (engl. widget). Kroz grafičko sučelje omogućava izgradnju novih udomljenika iz osnovnih gradbenih elemenata od kojih su sačinjeni postojeći udomljenici. Potrošači ne moraju posjedovati opsežna znanja iz programiranja kako bi ovim alatom stvarali nove udomljenike. Slika 7.1. prikazuje programsku paradigmu za usložnjavanje udomljenika.

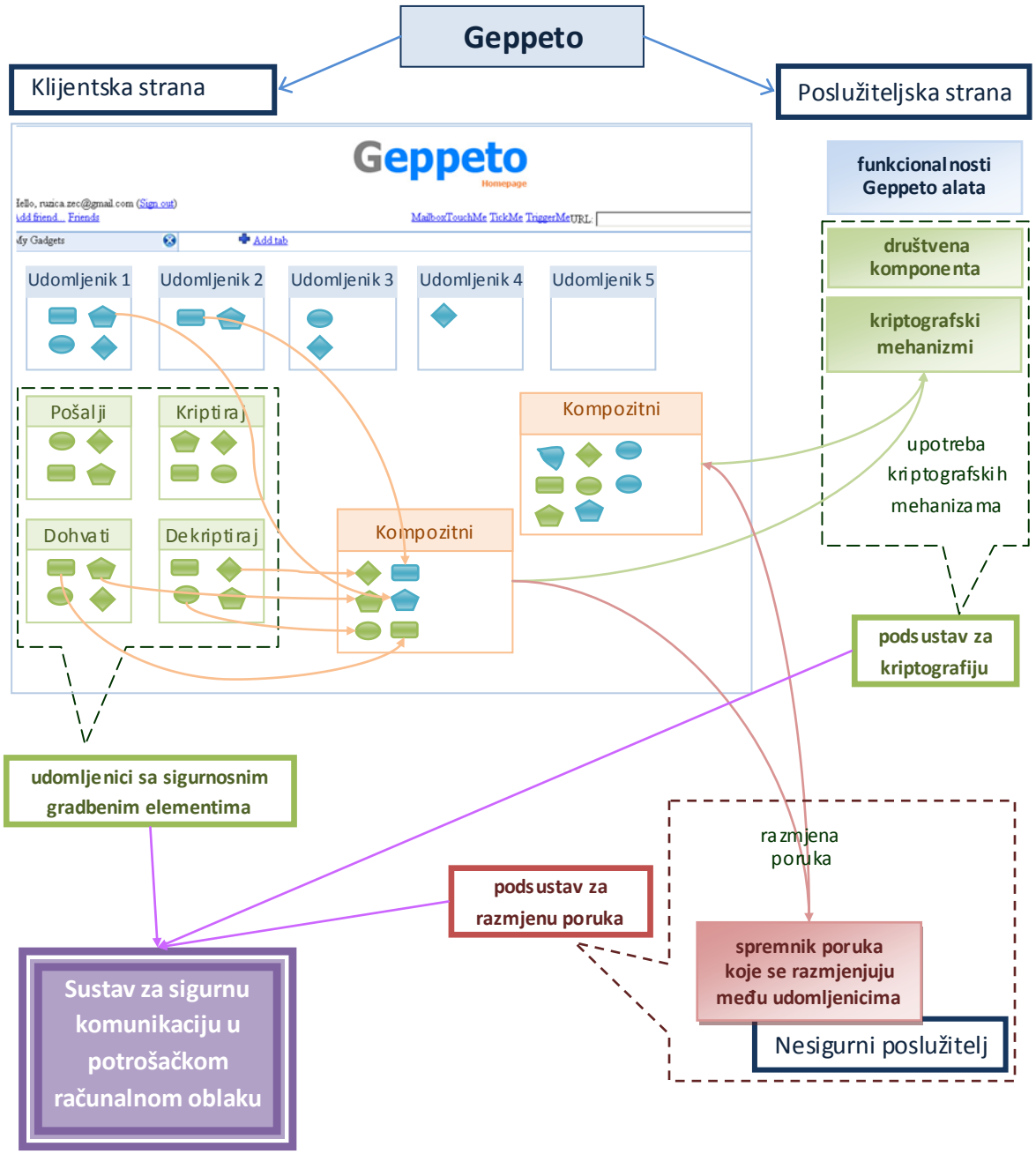


Slika 7.1. Programska paradigma za usložnjavanje udomljenika (Škvorc, 2010.)

Sustav za sigurnu komunikaciju kroz četiri udomljenika pruža sigurnosne gradbene elemente za izgradnju novih udomljenika. Uz udomljenike, potporu za sigurnu komunikaciju pružaju usluge poslužiteljske strane na kojoj su implementirani zaštitni kriptografski mehanizmi i koje su dio poslužiteljske strane Geppeto alata. Te usluge udomljenici koriste za kriptiranje i dekriptiranje poruka. Sustav za sigurnu komunikaciju još čini i podsustav za razmjenu poruka koji nije dio Geppeto alata, a preko kojeg udomljenici komuniciraju razmjenom poruka. Slika 7.2 prikazuje integraciju sustava za sigurnu komunikaciju u Geppeto alat. Na slici je vidljivo kako se pri izgradnji novih udomljenika između ostalih koriste i gradbeni elementi sigurnosnih udomljenika. Novi udomljenici tada sigurnim kanalima komuniciraju s poslužiteljskom stranom Geppeto alata na kojoj vrše kriptiranje i dekriptiranje

poruke te preko nesigurnih kanala kriptirane poruke stavljaju u spremnik poruka koji se nalazi na nesigurnom poslužitelju.

Uz implementaciju sigurnosnih mehanizama ostvarena je i društvena komponenta. Društvena komponenta vezana je uz podsustav za kriptografiju. Omogućava korisnicima povezivanje sa ostalim korisnicima sustava – korisnici mogu jedni druge dodavati na svoju listu prijatelja. Sigurna komunikacija je moguća samo između prijatelja.



Slika 7.2. Integracija sustava za sigurnu komunikaciju u Geppeto

7.1. Arhitektura sustava za sigurnu komunikaciju

Sustav za sigurnu komunikaciju čine četiri osnovna dijela:

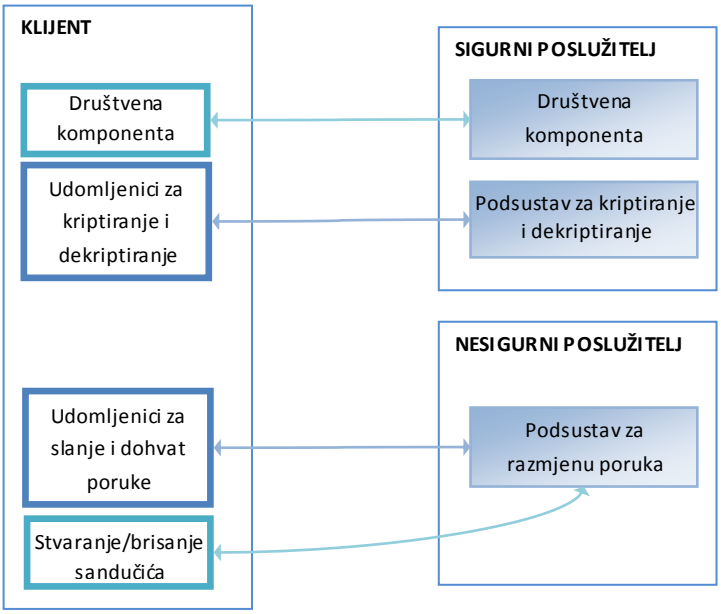
- podsustav za kriptografiju
- podsustav za razmjenu poruka
- društvena komponenta
- četiri udomljenika.

Slika 7.3 prikazuje općenitu arhitekturu sustava za sigurnu komunikaciju.

Kako bi jedan korisnik mogao sigurno razmjenjivati poruke s nekim drugim korisnikom potrebno je da on tog korisnika ima na listi svojih prijatelja. Za slanje zahtjeva za prijateljstvom i održavanje liste prijatelja brine se društvena komponenta sustava. Prije nego što korisnik pošalje poruku prijatelju potrebno je izvršiti kriptiranje poruke. Udomljenik za kriptiranje sigurnim komunikacijskim kanalom poruku šalje podsustavu za kriptiranje kako bi se poruka kriptirala. Podsustav izvršava kriptiranje i korisniku kao odgovor šalje kriptiranu poruku. Udomljenik za slanje kriptiranu poruku nesigurnim komunikacijskim kanalom šalje nesigurnom poslužitelju. Udomljenik prilikom slanja definira poštanski sandučić u koji poslužitelj treba pohraniti poruku. Podsustav za kriptiranje i slanje se mogu nalaziti na različitim poslužiteljima. Bitno je da se podsustav za kriptiranje nalazi na sigurnom poslužitelju i da je kanal kojim udomljenik komunicira s njime siguran.

Dohvaćanje poruke se odvija obrnutim redoslijedom od primanja. Udomljenik definira naziv poštanskog sandučića te dohvaća poruku iz tog sandučića. Nakon dohvata poruke drugim se udomljenikom kriptirana poruka šalje podsustavu za dekriptiranje koji poruku dekriptira te je vraća udomljeniku.

Razmjena poruka se odvija preko poštanskih sandučića. Oni se nalaze na nesigurnom poslužitelju te je potrebno da poruke koje se tamo pohranjuju budu zaštićene odnosno kriptirane. Korisnici sami stvaraju poštanske sandučiće pri čemu definiraju njihove nazive. Korisnici mogu obrisati stvorene poštanske sandučiće.



Slika 7.3. Arhitektura sustava za sigurnu komunikaciju

7.1.1. Društvena komponenta

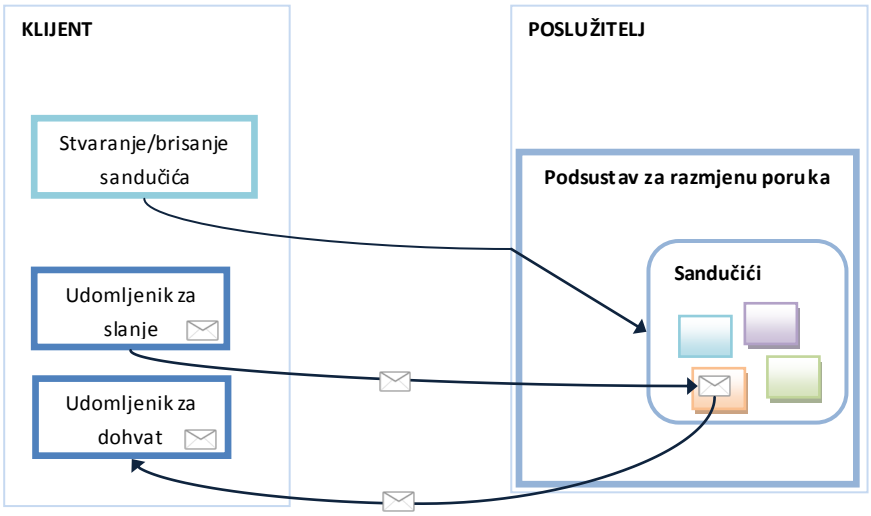
Društvena komponenta sustava sastoji se od klijentske i poslužiteljske strane. Na klijentskoj strani korisnik unosi ime korisnika kojeg želi dodati na svoju listu prijatelja. Poslužiteljska strana obrađuje pristigle zahtjeve za prijateljstvom. Korisnik koji primi zahtjev za prijateljstvom može prihvatiti (engl. accept) ili odbiti (engl. decline) primljeni zahtjev za prijateljstvom. U slučaju da prihvati zahtjev dva korisnika postaju prijatelji te mogu razmjenjivati kriptirane poruke. U slučaju da odbije zahtjev, oni ne mogu razmjenjivati poruke na siguran način. Ako su dva korisnika prijatelji, oni to prijateljstvo mogu raskinuti u svakom trenutku – maknuti korisnika sa svoje liste prijatelja (engl. remove). Također, nakon što jedan korisnik pošalje zahtjev, on u bilo kojem trenutku dok taj zahtjev nije prihvaćen može odustati (engl. cancel) od poslanog zahtjeva. Akcije koje korisnici mogu obavljati nakon što jedan korisnik pošalje zahtjev za prijateljstvo drugom korisniku prikazane su u tablici 7.1. Drugi i treći stupac prikazuje akcije koje prvi i drugi korisnik mogu obaviti nakon što je jedan od korisnika obavio akciju prikazanu u prvom stupcu (početna akcija je slanje zahtjeva za prijateljstvo od prvoga korisnika drugome). *init* označava inicijalno stanje u kojem se nalaze korisnici prije slanja zahtjeva za prijateljstvo, odnosno stanje u koje se oni prelaze nakon što maknu prijatelja iz liste prijatelja ili odustanu od zahtjeva za prijateljstvom.

Tablica 7.1. Akcije vezane za upravljanje listom prijatelja

Akcija koju izvršava korisnik 1	Korisnik 1	Korisnik 2
init	**send** (prva akcija koja se obavlja)	send
send	Cancel	accept, decline
remove	*Init*	*init*
Akcija koju izvršava korisnik 2	**Korisnik 1**	**Korisnik 2**
cancel	*Init*	*init*
accept	Remove	remove
decline	Init	init
remove	*Init*	*init*

7.1.2. Podsustav za razmjenu poruka

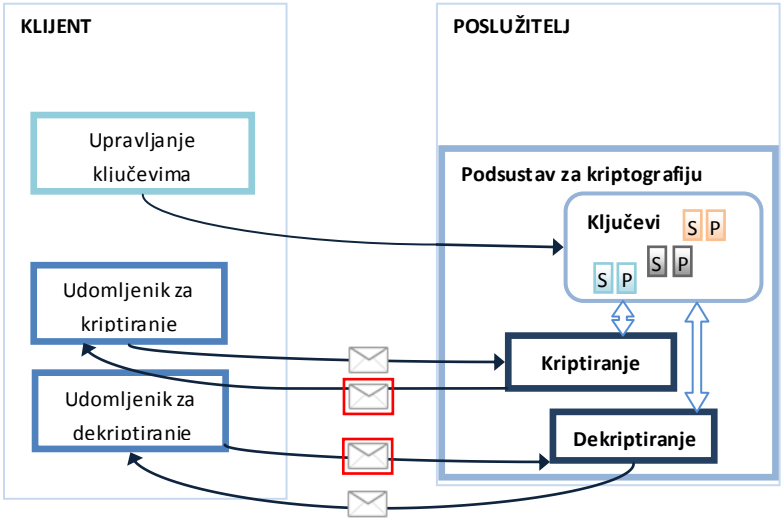
Podsustav za razmjenu poruka omogućava razmjenu poruka između dva korisnika. Poruka se šalje sa udomljenika za slanje poruka i nesigurnim komunikacijskim kanalom pristiže do poslužitelja na kojem se nalazi podsustav za razmjenu poruka. Podsustav sadrži imenovane poštanske sandučiće u koje korisnici mogu pohranjivati poruke i iz kojeg dohvaćaju poruke. Veličina poruke koja se može pohraniti u sandučić nije ograničena, ali broj poruka koje se mogu pohraniti jest. Sandučić se ponaša kao cirkularni FIFO (engl. first-in-first-out) red, poruka koja je prva stavljena bit će najprije dohvaćena. Arhitektura podsustava za razmjenu poruka prikazana je na slici 7.4. Korisnici mogu sami stvarati i brisati sandučiće određenog naziva.



Slika 7.4. Arhitektura podsustava za razmjenu poruka

7.1.3. Podsustav za kriptografiju

Podsustav za kriptografiju pruža kriptografske mehanizme za zaštitu poruka. Udomitelj za kriptiranje sigurnim kanalom podsustavu za kriptografiju šalje poruku koja se treba kriptirati. Uz poruku se šalje ime korisnika kojem je poruka namijenjena te naziv algoritma kojim se želi kriptirati poruka. Na poslužitelju se izvodi kriptiranje. Poruka se kriptira javnim ključem primatelja. Izrađuje se sažetak poruke i on se kriptira privatnim ključem primatelja. Kao odgovor udomljeniku za kriptiranje se šalju kriptirana poruka i kriptirani sažetak (odnosno digitalno potpisana digitalna omotnica). Dekriptiranje započinje slanjem kriptirane poruke podsustavu za kriptografiju. Kriptirana poruka je sačinjena od same kriptirane poruke te kriptiranog sažetka poruke. Poruka se dekriptira privatnim ključem primatelja, a sažetak javnim ključem pošiljatelja. Računa se sažetak poruke te uspoređuje s dekriptiranim sažetkom. Ukoliko su sažeci jednaki primatelj je siguran da je poruka sigurno stigla do njega. Na kraju se dekriptirana poruka kao odgovor šalje udomljeniku za dekriptiranje. Uz kriptiranje i dekriptiranje poruka podsustav za kriptografiju izvršava i generiranje parova javnog i privatnog ključa. Ključeve pohranjuje na (sigurnom) poslužitelju. Arhitektura podsustava za kriptografiju je prikazana na slici 7.5.



Slika 7.5. Arhitektura podsustava za kriptografiju

7.1.4. Udomljenici

Udomljenici su jednostavne aplikacije koje se mogu ugraditi u web stranice i druge aplikacije. U sustavu za sigurnu komunikaciju postoje četiri udomljenika:

- udomljenik za slanje poruke
- udomljenik za dohvat poruke
- udomljenik za kriptiranje poruke
- udomljenik za dekriptiranje poruke.

Udomljenici za slanje i dohvat poruke usko su povezani uz podsustav za razmjenu poruka. Predstavljaju klijentsko sučelje kroz koje se poruke šalju i dohvaćaju. Ekvivalentno vrijedi i za udomljenike za kriptiranje i dekriptiranje koji su korisničko sučelje preko kojeg se šalju poruke na kriptiranje odnosno dekriptiranje podsustavu za kriptografiju.

7.2. Programsko ostvarenje sustava za sigurnu komunikaciju

Sustav za sigurnu komunikaciju izgrađen je programskim jezicima Java i HTML/JavaScript. Korišteno je razvojno okruženje *Eclipse Galileo* i *jetty* poslužitelj. Dodane tehnologije koje su korištene u izgradnji sustava jesu GXP (Google XML Pages), CSS (Cascading Style Sheets), XML (Extensible Markup Language), JSON (JavaScript Object Notation) i AJAX (Asynchronous JavaScript).

7.2.1. Društvena komponenta

Društvena komponenta sastoji se od klijentske i poslužiteljske strane. Klijentska strana izrađena je u *JavaScriptu.* Pruža grafičko sučelje preko kojeg jedan korisnik može poslati zahtjev za prijateljstvom drugom korisniku. Ime korisnika kojeg se želi dodati na listu prijatelja se HTTP POST metodom kroz AJAX zahtjev šalje poslužiteljskoj strani društvene komponente. Poslužiteljska strana izgrađena je u *Javi*. Poslužitelj obrađuje primljeni zahtjev i u XML datoteke obaju korisnika, onog koji je poslao zahtjev i onog kojemu je zahtjev poslan, dodaje zapise o poslanom odnosno primljenom zahtjevu za prijateljstvo. Struktura korisnikove XML datoteke i zapisi o zahtjevima za prijateljstvo prikazani su na slici 7.6.

```
▼<Friends>
   <Friend friendId="pahulja@gmail.com" friendStatus="friend"/>
   <Friend friendId="oblak@gmail.com" friendStatus="friend"/>
   <Friend friendId="donald.duck@gmail.com" friendStatus="pending"/>
   <Friend friendId="zekoslav@gmail.com" friendStatus="sent"/>
 </Friends>
```

Slika 7.6. Zapisi o zahtjevima za prijateljstvo u korisnikovoj XML datoteci

Na poslužiteljskoj strani društvene komponente bitno je održati konzistentnost XML datoteka svih korisnika. Kad se zaprimi zahtjev za prijateljstvom ili pristigne neka druga akcija vezana za održavanje liste prijatelja (prihvaćanje zahtjeva, odbijanje zahtjeva, uklanjanje prijatelja s liste, odustajanje od zahtjeva) taj zahtjev vezan je uz dva korisnika i sukladno tome utječe na stanja dvaju korisnika koje je zapisano u dvije XML datoteke. Drugim riječima, ako u XML datoteci korisnika A postoji zapis o statusu prijateljstva s korisnikom B, tada u XML

datoteci korisnika B postoji zapis o statusu prijateljstva s korisnikom A; kada dođe akcija koja mijenja status prijateljstva tih dvaju korisnika, tada se promjene moraju izvršiti ili u obje XML datoteke ili se ne izvršiti ni u jednoj – treba biti zadovoljeno svojstvo atomarnosti. Također, ne smije se dogoditi niti sljedeća situacija jer i ona dovodi do nekonzistentnosti na poslužitelju:

- korisnik A pošalje zahtjev za promjenom statusa prijateljstva s korisnikom B
- poslužitelj zapisuje tu promjenu u XML datoteku korisnika A
- korisnik B pošalje zahtjev za promjenom statusa prijateljstva s korisnikom A (drukčiji nego je korisnik A poslao)
- poslužitelj zapisuje promjenu uzrokovanu zahtjevom korisnika B u XML datoteku korisnika B
- poslužitelj zapisuje promjenu uzrokovanu zahtjevom korisnika A u XML datoteku korisnika B.

Na primjer, nakon što korisnik A pošalje zahtjev za prijateljstvom korisniku B, korisnik A može odustati od zahtjeva, a korisnik B može prihvatiti ili odbiti zahtjev za prijateljstvom. Neka istovremeno korisnik B prihvati zahtjev, a korisnik A odustane od zahtjeva. Odustajanjem od zahtjeva se briše informacija o poslanom zahtjevu za prijateljstvo iz odgovarajućih XML datoteka (datoteka korisnika A i korisnika B). Prihvaćanjem se mijenjaju odgovarajući zapisi u XML datotekama korisnika A i B. Zbog višedretvenosti u izvođenju moguć je sljedeći slijed događaja koji se ne smije dogoditi:

- brisanje zapisa o prijateljstvu korisnika A i B iz XML datoteke korisnika A
- u XML datoteci korisnika A se mijenja zapis o prijateljstvu korisnika A i B, korisnici postaju prijatelji
- brisanje zapisa o prijateljstvu korisnika A i B iz XML datoteke korisnika B.

Potrebno je još u XML datoteci korisnika B promijeniti zapis o prijateljstvu s korisnikom A. To neće biti moguće jer je zbog isprepletanja operacija prouzrokovanih dvjema akcijama nad istim objektom (zapisom u XML datoteci) došlo do nekonzistentne situacije - zapis se ne može

izmijeniti jer on ne postoji. Očuvanje konzistentnosti riješeno je pomoću Javine ključne riječi *synchronized* kojom je onemogućeno dvjema dretvama da istovremeno izvode akcije. Kada se krene izvršavati jedna akcije, druga akcija će moći biti izvršena tek kada završi izvođenje prve akcije.

7.2.2. Podsustav za komunikaciju porukama

Podsustav za komunikaciju porukama ostvaren je kroz četiri klase:

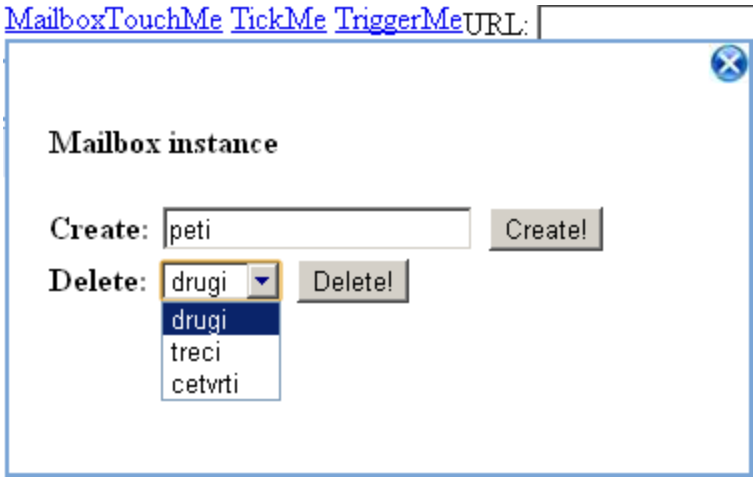
- MailboxServlet.java
- MailboxManager.java
- OneMailbox.java
- MailboxXMLFUnctions.java

MailboxServlet.java

MailboxServlet.java je Javin servlet koji prima svi zahtjeve koji se obrađuju na podsustavu za komunikaciju porukama. Prima sve HTTP POST zahtjeve vezane za razmjenu poruka: kreiranje instance sandučića, brisanje sandučića, pohrana poruke u sandučić te dohvat poruke iz sandučića. Poziva odgovarajuće metode MailboxManager-a za obradu ovih zahtjeva. Nakon obrade zahtjeva šalje odgovor korisniku.

MailboxManager.java

MailboxManager.java sadrži metode za kreiranje i brisanje poštanskih sandučića. Klijentsko sučelje za kreiranje i brisanje instanci sandučića prikazano je na slici 7.7. Korisnik prilikom stvaranja sandučića definira njihov naziv te šalje HTTP zahtjev za stvaranje sandučića. Na klijentskoj strani popis svih instanciranih sandučića i njihovih URL-ova se pohranjuje u dvodimenzionalno polje.



Slika 7.7. Sučelje za stvaranje i brisanje instance sandučića

Na poslužiteljskoj strani podaci o stvorenim instancama se pohranjuju u HashMap-u; ključ je naziv sandučića, a vrijednost referenca na instancu sandučića. Osim u HashMap-u podaci se pohranjuju i u Mailboxes.xml datoteku kako bi bili sačuvani i nakon prestanka rada sustava. Pohranjuje se naziv sandučića i njegova URL (engl. Uniform Resource Locator) adresa koji ga jednoznačno određuju. Ova klasa također sadrži metode koje pozivaju metode instanci sandučića za pohranu i dohvat poruka.

OneMailbox.java

OneMailbox.java predstavlja programsko ostvarenje poštanskog sandučića u kojeg se stavljaju poruke. Poruke su niz znakova (engl. character) te se pohranjuju u polje *stringova*. Metodama `putMessage(String message)` i `getMessage(String message)` poruke se stavljaju odnosno dohvaćaju iz polja *stringova*. Ostvareno rješenje sandučića se ponaša kao FIFO red poruka, poruke koje su prije stavljene se prije dohvaćaju. Unutar klase definiran je veličina sandučića, broj poruka koje se mogu pohraniti. Osim u polje *stringova* poruke se pohranjuju i u XML datoteke kako bi poruke bile sačuvane i u slučaju prestanka rada sustava. Prilikom stvaranja instanci sandučića za svaki se sandučić stvara XML datoteka *naziv_sandučića.xml* u koju se pohranjuju poruke. Struktura ovih XML datoteka je prikazana na slici 7.8.

Slika 7.8. Struktura XML datoteke koja opisuje sandučić

MailboxXMLFunctions.java

MailboxXMLFunctions.java sadrži metode za pohranu i dohvat podataka iz XML datoteka.

7.2.3. Podsustav za kriptografiju

Podsustav za kriptografiju ostvaren je kroz sljedeće klase:

- CryptographyServlet.java
- CryptographyManager.java
- CryptographyXMLFunctions.java
- RSA.java
- MD5.java

CryptographyServlet.java

CryptographyServlet.java zaprima HTTP POST zahtjeve za kriptiranje i dekriptiranje poruka te generiranje ključeva. Servlet poziva odgovarajuće metode CryptographyManagera-a te mu kao parametre šalje naziv odabranog algoritma za kriptiranje i ime korisnika kojemu

je namijenjena poruka koja se kriptira kako bi se mogli odrediti ključevi za kriptiranje, dekriptiranje ili izradu sažetka. U slučaju generiranja ključeva kao parametar se prenosi ime korisnika kojemu se generiraju ključevi.

CryptographyManager.java

CryptographyManager.java dohvaća kriptografske ključeve određenih korisnika iz XML datoteke te poziva metode konkretnih algoritama za generiranje ključeva, kriptiranje, dekriptiranje ili izradu sažetka. Implementiran je RSA algoritam za kriptiranje i dekriptiranje poruka te MD5 za izradu sažetka.

CryptographyXMLFunctions.java

CryptographyXMLFunctions.java sadrži funkcije za pohranu i dohvat ključeva iz XML datoteke koja služi za spremanje ključeva svih korisnika. Format ove XML datoteke je prikazan na slici 7.9.

Slika 7.9. Struktura XML datoteke u koju se pohranjuju ključevi za kriptiranje i dekriptiranje

RSA.java i **MD5.java** sadrže implementacije ovih algoritama koji su opisani u 6. poglavlju. RSA.java sadrži funkcije `encrypt(String userId, String message, String`

`algorithm, String friendId)`, `decrypt(String userId, String message, String algorithm, String friendId)` i `generatekeys (String userId)`. MD5.java sadrži funkciju `createMessageDigest (String message)`.

Kriptiranje, dekriptiranje i izrada sažetka izvode se nad nizovima bitova. Rezultat ovih operacija je drukčiji niz bitova. Rezultatni niz bitova kodira se *base64* formatom. Ovaj format se koristi za pohranu složenih podataka. Pogodan je za pohranu kriptografskih ključeva u XML datoteku i prijenos mrežom. Kodiranjem se po 24 bita dijele na četiri dijela po 6 bitova i oni se zamjenjuju jednim znakom. Znakovi kojima se zamjenjuju su velika i mala slova engleske abecede (A-Z, a-z), brojevi (0-9) te znakovi plus (+), *slash* (/) i jednakost (=). Ako zadnji blok ima manje od 24 bita izvodi nadopuna. Ako ima 8 bitova, dodaju se četiri bita vrijednosti 0, 12 bitova se dijeli u dva dijela po šest bitova koji se zamjenjuju odgovarajućim znakovima te se dodaju dva znaka jednakosti. Ako ima 16 bitova, dodaju se dva bita vrijednosti 0, niz se dijeli na tri dijela koji se zamjenjuju odgovarajućim znakovima i na kraj se dodaje jedan znak jednakosti.

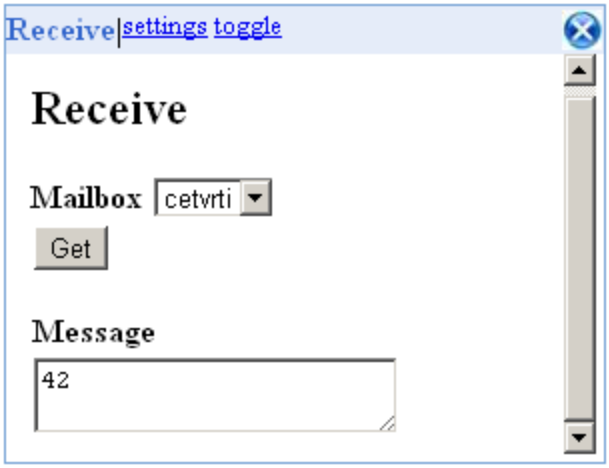
7.2.4. Udomljenici

Udomljenici su izrađeni u HTML-u i *JavaScriptu*. Dodatne je korišten CSS za definiciju stila udomljenika te AJAX za slanje HTTP zahtjeva poslužitelju. Udomljenici su prikazani na slici 7.9.

Kod udomljenika za slanje (Slika 7.10) u *textbox* se upisuje poruka koja se želi poslati te odabire instanca sandučića u koju se poruka želi staviti. Popis instanci udomljenika sadrži udomitelj. Udomljenik kroz RPC poziv udomitelju dohvaća listu svih instanci sandučića. Za svaku instancu sandučića definirana je i URL adresa te instance. Klikom na gumb *Send* poruka se šalje u sandučić određen njegovim URL-om te se unutar udomljenika ispisuje poruka o uspješnosti slanja.

Slika 7.10. Udomljenik za slanje poruka

Udomljenik za primanje poruke (Slika 7.11) sadrži *select menu* iz kojeg se odabire instanca sandučića iz kojeg se dohvaća poruka. Popis instanci i pripadnih URL adresa se RPC pozivom dohvaća sa udomitelja. Poruka se zatim dohvaća iz odabranog sandučića klikom na gumb *Get* te se upisuje u *textbox*.



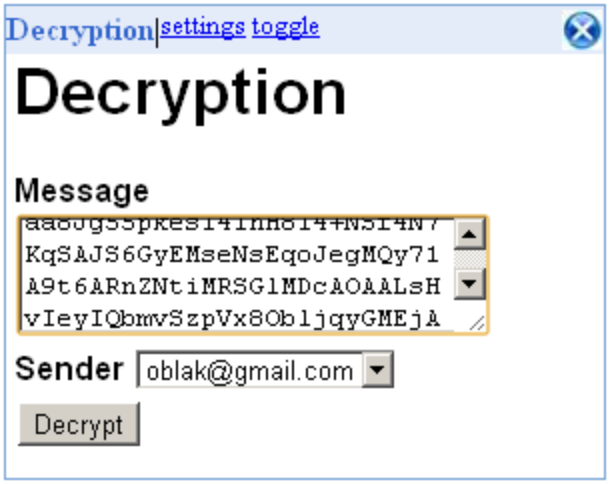
Slika 7.11. Udomljenik za dohvat poruke

Udomljenik za kriptiranje (Slika 7.12) sadrži *textbox* u koji se upisuje poruka koja se želi poslati. Odabire se algoritam kriptiranja i osoba kojoj je ta poruka namijenjena. Pošiljatelj može poruke slati samo svojim prijateljima. Popis prijatelja se saznaje preko RPC poziva

udomitelju. Poruka, naziv algoritma i ime primatelja šalju se na poslužitelj klikom na gumb *Encrypt.* Kao odgovor će se dobiti digitalno potpisana digitalna omotnica koja se upisuje u *textbox*. Odgovor će biti XML zapis koji unutar korijenskog *encryptionResult* sadrži dva čvora, *encryptedMsg* unutar kojeg je zapisana kriptirana poruka i *encryptedMsgHash* unutar kojeg je zapisan kriptirani sažetak poruke.

Slika 7.12. Udomljenik za kriptiranje poruke

Udomljenik za dekriptiranje (Slika 7.13) sadrži *textbox* u koji se upisuje digitalno potpisana digitalna omotnica. Odabire se algoritam dekriptiranja i ime osobe od koje smo dobili poruku te se klikom na gumb *Decrypt* poruka šalje na dekriptiranje. Poruke se mogu primiti samo od prijatelja te se taj popis dohvaća sa udomljenika preko RPC poziva. Odgovor poslužitelja će biti dekriptirana poruka.



Slika 7.13. Udomljenik za dekriptiranje poruke

8. Zaključak

Sve složeniji razvoj programskih i komunikacijskih sustava donosi potrebu za uspostavom sigurnosnih mehanizama u svim programskim sustavima koji razmjenjuju poruke preko Interneta. Zbog toga je potrebno i unutar Geppeta, potrošačkog programskog alata za izgradnju složenih udomljenika, ostvariti zaštitne kriptografske mehanizme koje će izgrađeni udomljenici koji razmjenjuju poruke preko Interneta koristiti za sigurnu komunikaciju.

Ostvareni radni okvir za sigurnu komunikaciju sastoji se od udomljenika koji pružaju sigurnosne i komunikacijske gradbene elemente za izgradnju novih udomljenika, podsustava za kriptografiju koji ostvaruje kriptografske algoritme za zaštitu podataka, posredničkog sustava preko kojeg se razmjenjuju poruke te društvene komponente koja omogućuje povezivanje korisnika. Ostvareni udomljenici se koriste kao pružatelji gradbenih elemenata koji služe za izgradnju složenih udomljenika koji mogu sigurno komunicirati preko mreže. Ti udomljenici koriste ostvarene kriptografske usluge poslužitelje strane Geppeta za kriptiranje poruka RSA algoritmom i izradu kriptiranog sažetka MD5 algoritmom kako bi bili zadovoljeni osnovni kriptografski zahtjevi: povjerljivost, besprijekornost, autentičnost i neporecivost. Ostvarena je i posrednička usluga za razmjenu poruka preko koje udomljenici mogu razmjenjivati poruke. Poruke se pohranjuju u poštanskim sandučićima na posredničkom poslužitelju dok ih primatelj ne dohvati.

Potpis:

_____

9. Literatura

[1] Buyya R., Broberg J., Goscinski A. M. Cloud Computing: Principles and Paradigms. Prvo izdanje. Australija: John Wiley and Sons, 2011.

[2] Fielding R. T. Architectural Styles and the Design of Network based Software Architecture. Doktorska disertacija. Sveučilište u Kaliforniji, 2000.

[3] Mather T. Kumaraswamy S., Latif S. Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. Prvo izdanje. SAD. O'Reilly Media, 2009.

[4] Furht B., Escalante A. Handbook of Cloud Computing. Prvo izdanje. SAD: Springer, 2010.

[5] Hayes B. Cloud computing. Communications of ACM. Vol.51/No.7., 2008, 9-11.

[6] Sosinsky B. Cloud computing: Bible. Prvo izdanje. SAD: Wiley and sons, 2011.

[7] Škvorc D. Programiranje prilagođeno potrošaču. Doktorska disertacija. Sveučilište u Zagrebu, 2010.

[8] Ciccarelli P., Faulkner C. Networking Foundations. SAD: Sybex, 2004.

[9] Coulouris G., Dollimore J., Kindberg T. Distributed systems: Concepts and design. Četvrto izdanje. Engleska: Addison Wesley, 2005.

[10] Lovrek I., Vrsalović D., Kušek M., Podnar Žarko I. Raspodijeljeni sustavi: predavanja za ak. god 2010./2011.

[11] Budin L., Golub M., Jakobović D., Jelenković L. Operacijski sustavi. Prvo izdanje. Zagreb: Element, 2010.

[12] Yan S. Y. Number Theory for Computing. Second edition. Birmingham: UK. Springer, 2002.

[13] Dujella A. Diskretna matematika: Matematičke osnove kriptografije javnog ključa. Bilješke s predavanja za ak. god. 2009./2010. http://web.math.hr/~duje/diskretna/diskretna.pdf. 6.5.2011.

[14] Wikipedia: Computer security. http://en.wikipedia.org/wiki/Computer_security 24.5.2011.

[15] Rivest R. The MD5 Message-Digest Algorithm: MD5 Algorithm description. http://tools.ietf.org/html/rfc1321 8.5.2011.

Sažetak

Radni okvir za sigurnu komunikaciju u potrošačkom računalnom oblaku

Radni okvir za sigurnu komunikaciju u potrošačkom računalnom oblaku izgrađen je u svrhu pružanja sigurnosnih mehanizama unutar potrošačkog programskog alata Geppeto. Radni okvir za sigurnu komunikaciju sastoji se od četiri udomljenika koji pružaju sigurnosne i komunikacijske gradbene elemente za izgradnju novih udomljenika, podsustava za kriptografiju koji ostvaruje kriptografske algoritme za zaštitu podataka, posredničkog sustava preko kojeg se razmjenjuju poruke te društvene komponente koja omogućava povezivanje korisnika. Ostvareni su mehanizmi za sigurnu komunikaciju porukama preko posredničkog poslužitelja između udomljenika. Komunikacija zadovoljava osnovne sigurnosne zahtjeve: povjerljivost, besprijekornost, autentičnost, neporecivost. Zaštita poruka koje se razmjenjuju preko poštanskih sandučića ostvarena je korištenjem kriptografskog algoritma RSA i kriptografske funkcije za sažimanje MD5 kojima se izrađuje digitalno potpisana digitalna omotnica.

Ključne riječi:

Geppeto
udomljenik
sigurna komunikacija
komunikacija porukama
kriptografija
RSA
MD5
poštanski sandučić

Abstract

Framework for secure communication in the consumer computing cloud

Framework for secure communication in the consumer computing cloud is built in order to provide mechanisms for secure communication in the consumer-level programming environment Geppeto. Framework for secure communication consists of four widgets which provide security and communication building blocks for widget composition, cryptography subsystem that implements the cryptographic algorithms for securing the data, middleware through which messages are exchanged and the social component that offers users to connect to each other Mechanisms for secure message communication between widgets through middleware are implemented. Communication meets the basic security requirements: confidentiality, integrity, authenticity, non-repudiation. The protection of messages exchanged over the mailbox was achieved using cryptographic algorithm RSA and cryptographic hash function MD5 which produces a digitally signed digital envelope.

Keywords:

Geppeto

widget

secure communication

message-queuing

cryptography

RSA

MD5

mailbox