Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Pinterič

Sistemi za takojšnje sporočanje

diplomsko delo

univerzitetni študij
računalništvo in informatika

Mentor: dr. Andrej Brodnik

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:
Sistemi za takojšnje sporočanje

Tematika naloge:

Ko so očetje ARPANET-a in kasneje interneta prvič pognali svoje novo okolje, je bila ena prvih storitev, ki so jih pripravili (tedaj so dejali, da so implementirali aplikacijo), *talk*. Storitev *talk* je bila prvi primer nečesa, čemur danes rečemo takojšnje sporočanje. Kasneje so ji sledili zapleteni sistemi, kot so IRC, vse do danes, ko jih na nek način nadgrajuje *twitter*. V diplomski nalogi preglejte, kateri sistemi za takojšnje sporočanje so na trgu na voljo. Po njihovi primerjavi namestite enega od njih ter ga ovrednotite. Pomembne lastnosti, ki naj jih ima vaš sistem, so: možnost uporabe na širokem naboru operacijskih sistemov, možnost javnega in zasebnega pogovora, varnostni vidiki, video pogovor, usklajevanje ter možnost pregledovanja zgodovine pogovora.

Izjava o avtorstvu zaključnega dela

Spodaj podpisani Miha Pinterič, vpisna številka 63010109, avtor zaključnega dela z naslovom:

Sistemi za takojšnje sporočanje (angl. *Instant Messaging Systems*)

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom dr. Andreja Brodnika;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 12. septembra 2016 Podpis študenta/-ke:

Zahvaljujem se mentorju dr. Andreju Brodniku za vodenje in strokovne nasvete pri izdelavi diplomske naloge. Hvala družini in Soizic za vso podporo in spodbudo v letih študija. Hvala Mitju za posojen telefon in Piji za pomoč pri testiranju.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Struktura naloge 2

2 Storitve takojšnjega sporočanja **3**

2.1 Osnovni namen in arhitektura takojšnjega sporočanja 3

2.1.1 Osnovni namen 3

2.1.2 Arhitektura 3

2.2 Dodatne storitve 4

2.3 Varnostne zahteve pri takojšnjem sporočanju 6

2.4 Odprtokodna rešitev TS 7

2.5 Komercialni ponudniki TS 8

3 Odprtokodne rešitve **11**

3.1 Jedro protokola XMPP 12

3.1.1 Arhitektura 13

3.1.2 Povezovanje s strežnikom 14

3.1.3 Komunikacijski elementi protokola XMPP 20

3.1.4 Upravljanje kontaktov in prisotnost v omrežju 21

3.2 Razširitve XEP (XMPP Extension Protocols) 21

3.2.1 Protokol Jingle 22

3.2.2 Drugi načini prenosa datotek 25

3.2.3 Usklajevanje sporočil na napravah 26

3.2.4 Arhiv sporočil na strežniku 26
3.2.5 Klepetalnice 27

4 Orodja za zagotavljanje varnosti **29**
4.1 Avtentikacija sogovornikov 30
4.2 Varna izmenjava sporočil 32

5 Ovrednotenje in analiza **37**
5.1 Postavitev strežnika 37
5.1.1 Konfiguracija 40
5.1.2 Moduli strežnika Prosody 42
5.1.3 Preverjanje varnostnih nastavitev 44
5.2 Ovrednotenje rešitve 45
5.2.1 Izbira odjemalcev 45
5.2.2 Analiza delovanja 46

6 Sklepne ugotovitve **49**
6.1 Tehnična primerjava sistemov 49
6.1.1 Arhitektura 49
6.1.2 Odpiranje uporabniškega računa in dodajanje kontaktov . . 50
6.1.3 Odprtost programske kode 51
6.2 Primerjava uporabniške izkušnje 51
6.2.1 Bodoče izboljšave 51
6.3 Zaključek 53

Seznam uporabljenih kratic

kratica	angleško	slovensko
AES	Advanced Encryption Standard	napredni šifrirni standard
D-H	Diffie-Hellman key exchange	izmenjava ključev Diffie-Hellman
HTTP	Hypertext Transfer Protocol	protokol za prenos hiperbesedila
IETF	Internet Engineering Task Force	delovna skupina za internetno tehniko
IM	Instant Messaging	takojšnje sporočanje
IP	Internet Protocol	internetni protokol
IRC	Internet Relay Chat	klepet preko interneta
JID	Jabber ID	identifikacijski naslov Jabber
LAN	Local Area Network	lokalno omrežje
MAC	Message Authentication Code	overitvena sporočila
MB	Megabyte	megabajt
OMEMO	Multi-End Message and Object Encryption	več-končno šifriranje sporočil in objektov
OTR	Off-the-Record Messaging Protocol	,,Off-the-record“ varnostni protokol za takojšnje sporočanje
PGP	Pretty Good Privacy	precej dobra zasebnost
RFC	Request for Comments	zahtevek za komentarje
RTP	Real Time Protocol	protokol za prenos v stvarnem času

kratica	**angleško**	**slovensko**
SIP	Session Initiation Protocol	protokol za vzpostavitev seje
TCP	Transmission Control Protocol	protokol z nadzorom prenosa
TLS	Transport Layer Security	varnost prenosnega sloja
UDP	User Datagram Protocol	protokol uporabnikovih datagramov
URL	Uniform Resource Locator	enotni naslov vira
XEP	XMPP Extension Protocols	razširitve protokola XMPP
XML	Extensible Markup Language	razširljivi označevalni jezik
XMPP	Extensible Messaging and Presence Protocol	razširljiv protokol za sporočanje in prisotnost

Povzetek

Naslov: Sistemi za takojšnje sporočanje

Število uporabnikov takojšnjega sporočanja (TS) se je v zadnjih letih zelo povečalo. Komercialni ponudniki uporabljajo zaprta centralizirana omrežja in ne omogočajo komunikacije z ostalimi omrežji, čeprav standardizirani protokoli za TS obstajajo. V diplomski nalogi raziščemo alternativno možnost TS z uporabo protokola XMPP in razpršene mreže strežnikov. Opišemo osnovni namen, dodatne storitve in varnostne zahteve TS. Opišemo delovanje in možnosti protokola XMPP ter odprtokodnih orodij za varno komunikacijo. Glavni cilj diplomske naloge je postavitev strežnika Prosody, ki uporablja tehnologijo XMPP, in analiza delovanja z različnimi odjemalci XMPP na različnih operacijskih sistemih. V zaključku ovrednotimo prednosti in slabosti odprtokodne rešitve v primerjavi s komercialnimi ponudniki.

Ključne besede: takojšnje sporočanje, XMPP, Prosody, XEP, OMEMO .

Abstract

Title: Instant Messaging Systems

The number of users of instant messaging (IM) has significantly increased in the last few years. Commercial providers are using closed centralized networks and do not enable users to communicate with other networks, although standardized protocols for IM do exist. The present thesis explores an alternative option of IM with the use of the XMPP protocol and a federated network of servers. Furthermore, it includes a description of the basic purpose, additional services and security requirements of IM. In addition, there is a description of the architecture and properties of the XMPP protocol and security tools for end-to-end security. The main goal of the present thesis is to set up an IM open-source Prosody server, and to analyse the possibilities and the use of open-source clients. The conclusion of the thesis presents the advantages and disadvantages of open-source solution versus commercial providers.

Keywords: instant messaging, XMPP, Prosody, XEP, OMEMO.

Poglavje 1

Uvod

Verjetno si težko predstavljamo, da bi za pošiljanje elektronske pošte prijatelju bili primorani uporabljati istega ponudnika in enak program kot on. V primeru, da tega ne bi hoteli, pošiljanje sporočila prijatelju sploh ne bi bilo možno. Razen seveda, če bi se on odločil, da mu je komunikacija z nami preveč pomembna, in bi odprl račun tudi pri našem ponudniku. Če bi se tako odločil še za nekaj drugih prijateljev, bi imel kmalu deset različnih naslovov za elektronsko pošto in vsak dan pregledoval deset različnih programov.

Opisana situacija se zdi nemogoča, ampak je v svetu takojšnjega sporočanja (TS) trenutno prav taka. Deset najbolj uspešnih ponudnikov TS je zbralo že več kot pet milijard uporabnikov [75], povprečno število računov enega uporabnika družabnih omrežjih (katerih del je tudi TS) pa je 5,54 računa [64].

Pet milijard uporabnikov je ogromna številka in ogromno informacij, ki jih ponudnikom omrežij nudimo brezplačno (in nevede?) v zamenjavo za uporabo njihovih sistemov.

Wikileaks je z objavo dokumentov o množičnem nadzoru [2] zavedanje javnosti o problematiki zasebnosti na internetu precej izboljšal. Posledica tega je, da smo v zadnjih letih priča spreminjajočim se trendom na področju varnosti in zasebnosti komunikacije TS. Nekaj glavnih ponudnikov je v letošnjem letu implementiralo šifriranje od konca do konca [24, 33, 74], ki obljublja popolnoma zasebno in varno komunikacijo med uporabniki.

V diplomski nalogi nas zanima, kakšne so alternativne, nekomercialne možnosti TS in kako se obnesejo v primerjavi s komercialnimi. Kakšne so trenutne prednosti

in slabosti in ali nam lahko ponudijo celo kaj več?

1.1 Struktura naloge

Najprej definiramo osnovni namen TS in podamo dodatne zahteve, ki naj bi jih izpolnjeval odprtokodni sistem TS.

V 3. poglavju opišemo delovanje in značilnosti protokola XMPP. Pregledamo razširitve protokola za uporabo s TS, ki izpolnjujejo predhodno podane zahteve.

V 4. poglavju opišemo značilnosti in delovanje orodij za zagotavljanje varnosti v odprtokodnih sistemih TS.

V praktičnem delu diplomske naloge postavimo enega izmed odprtokodnih strežnikov XMPP in analiziramo uporabo sistema z odprtokodnimi odjemalci za XMPP na različnih operacijskih sistemih.

V sklepnem delu primerjamo komercialne sisteme z odprtokodnimi sistemi za TS. Podamo svoje ugotovitve o uporabnosti, prednostih in slabostih odprtokodnega sistema TS.

Poglavje 2

Storitve takojšnjega sporočanja

2.1 Osnovni namen in arhitektura takojšnjega sporočanja

2.1.1 Osnovni namen

Takojšnje sporočanje (TS) je tehnologija, ki uporabnikom v omrežju (internet, LAN) omogoča izmenjavo besedilnih sporočil v stvarnem času. Poglavitni lastnosti, ki jo razlikujeta od drugih podobnih tehnologij sporočanja (IRC, *twitter*,...) sta:

1. Komunikacija je mogoča le z osebami, ki so predhodno dodane na uporabnikov seznam kontaktov.

2. Informacija o prisotnosti kontaktov v omrežju je uporabniku vidna.

Osnovni namen TS je hitra izmenjava krajših sporočil med uporabniki, ki se med seboj poznajo.

2.1.2 Arhitektura

Nadzorna in podatkovna raven komunikacije

Nadzorna raven komunikacije skrbi za vzpostavljanje in zaključevanje seje, dogovarjanje o različnih parametrih seje, spreminjanje parametrov in podobno. Podatkovna raven komunikacije se uporablja za dejanski prenos podatkov.

Zaradi majhnosti sporočil sta pri TS možna dva načina nadzora in prenosa:

1. Nadzorna in podatkovna raven sta združeni.
   Sporočila in signali se prenašajo po istem komunikacijsko - signalizacijskem kanalu. V primeru prenosa večje količine podatkov (datoteke, zvok, video) se mora ustvariti dodaten komunikacijski kanal, po katerem se prenašajo podatki, obstoječi kanal pa se uporablja za signalizacijo.

2. Nadzorna in podatkovna raven sta ločeni.
   Sporočila in signali se prenašajo ločeno po komunikacijskem in signalizacijskem kanalu. V primeru prenosa večje količine podatkov se uporabi obstoječ komunikacijski kanal, spremeni se le tip podatkov.

Način delovanja

Sinhron način delovanja od sogovornikov zahteva, da so sočasno prijavljeni v omrežje - le tako je sporočanje možno. Sistem z arhitekturo vsak z vsakim („*peer to peer*“), kjer se pošiljatelj poveže neposredno s prejemnikom, je primer sinhronega sporočanja. Zaradi tehničnih omejitev (npr. napake v omrežju, izguba mobilnega omrežja, izpraznitev baterije naprave,...) sinhroni protokoli sporočanja za uporabnike niso tako praktični, kot asinhroni, in jih v diplomski nalogi ne obravnavamo.

Asinhron način delovanja omogoča pošiljanje sporočila prejemniku, ki ni prijavljen v omrežje. To je mogoče z arhitekturo odjemalec - strežnik. Tak sistem omogoča pomnenje sporočil na strežniku in posredovanje prejemniku ob prijavi v omrežje.

2.2 Dodatne storitve

Dandanes večina uporabnikov TS zahteva več kot le izmenjavo besedilnih sporočil. Nekatere pogostejše zahteve so:

- prenos datotek med uporabniki;
- možnost klica ali videoklica;
- uporaba zasebne klepetalnice, katero ustvari skupina znancev, ki si želijo skupinskega pogovora; ter

- sodelovanje v javni klepetalnici (ustvari jo moderator), ki obravnava neko določeno tematiko - sodelujoči so anonimni.

Za prenos govora in videa v stvarnem času je potrebno vzpostaviti neposredno povezavo za prenos podatkov med dvema entitetama. Sistem TS mora poleg protokola za sporočanje vključevati tudi protokol za vzpostavitev in nadzor seje, npr. RTP [26].

Prenos datotek se lahko vrši preko vmesnega strežnika neposredno med dvema entitetama (npr. z uporabo protokola SOCKS5 [27]) ali s prenašanjem datoteke na strežnik (npr. z uporabo protokola HTTP [6]).

Uporabnik lahko uporablja več naprav in/ali operacijskih sistemov. V tem primeru si želi imeti pogovore v vseh odjemalcih na različnih napravah in/ali sistemih usklajene. Prihaja do problema identifikacije uporabnika, ki je prijavljen z večih naprav. V sistemu TS mora biti zagotovljen način naslavljanja uporabnika glede na napravo, s katere dostopa. Podpora usklajevanju mora omogočati razpošiljanje na vse naprave, v primeru katerekoli kombinacije naslova uporabnik/naprava.

Pri usklajevanju prihaja do različnih situacij glede na prijavljenost v omrežje:

1. uporabnik dostopa z več odjemalcev, prijavljenih v omrežje,

2. uporabnik je z nekaterimi odjemalci prijavljen, z drugimi ne,

3. uporabnik se prvič prijavi z nekim novim odjemalcem.

Način usklajevanja pod točkama 1 in 2 je podoben, s to razliko, da se v točki 2 sporočila najprej pomnijo na strežniku, ob prijavi s prej neprijavljenim odjemalcem pa se posredujejo naprej in na strežniku izbrišejo.

Usklajevanje v primeru točke 3 zahteva shranjevanje vseh v preteklosti poslanih sporočil na strežniku. Ob prijavi z novim odjemalcem se sporočila prenesejo s strežnika.

2.3 Varnostne zahteve pri takojšnjem sporočanju

Leta 2004 objavljeno delo *,,Off-the-Record Communication, or, Why Not To Use PGP“* [9] primerja pogovor preko omrežja s pogovorom v živo, ki se ne snema in mu ne prisluškuje nobena tretja oseba. Avtorji predlagajo varnostne zahteve, ki bi se morale upoštevati pri komunikaciji dveh oseb (Ana in Borut) preko omrežja:

1. Samo Ana in Borut lahko bereta sporočila.

2. Borut mora biti prepričan, da je avtor sporočil resnično Ana, in obratno.

3. Sporočila med komunikacijsko potjo ne smejo biti kakorkoli spremenjena.

4. Po izmenjavi sporočil nihče (niti Ana niti Borut) ne more dešifrirati morebitnih shranjenih (šifriranih) sporočil.

5. Po izmenjavi sporočil nihče (niti Ana niti Borut) ne more preveriti in dokazati avtorstva sporočil.

Predlagane zahteve so osnova za varnostni protokol *,,Off-the-Record Messaging Protocol“* [10], ki pa je le eden izmed mnogih varnostnih protokolov za sporočanje.

Avtorji dela *,,SoK: Secure Messaging“* [66] iz leta 2015 ugotavljajo, da enotna vizija o varnostnem protokolu za sporočanje ne obstaja. Poleg tega nekateri komercialni ponudniki (zaradi zahtev v javnosti) obljubljajo varnostne rešitve, ki naj bi zagotavljale ,,vojaško raven varnosti“. Po neodvisno narejenih testih se izkaže, da niso ustrezne.

Avtorji predstavijo sistematsko metodologijo za ovrednotenje varnostnih orodij. Vrednotenje razdelijo v tri sklope:

1. Vzpostavitev zaupanja med sodelujočimi v pogovoru.
   Izmenjava ključev je proces, s katerim se izmenja kriptografski material za nadaljnje šifriranje. Ključi morajo biti ustrezno avtenticirani, s čimer se zagotovi, da ključ res pripada pravi osebi.

2. Varna izmenjava sporočil.
   Po uspešni vzpostavitvi zaupanja mora biti zagotovljena varnost in zasebnost izmenjave sporočil. V tem sklopu se vrednoti način šifriranja sporočil, izbor in ustrezno rabo kriptografskih protokolov.

3. Zasebnost pri prenosu sporočil.
   V tem sklopu se vrednoti način prenosa sporočil glede na informacije, ki jih razkrivajo. Tem podatkom pravimo metapodatki in lahko vsebujejo informacije o pošiljatelju, prejemniku in pogovoru, kateremu sporočila pripadajo.

V poglavju 4 bolj podrobno opišemo vsak sklop zahtev in ovrednotimo varnostna orodja OpenPGP, OTR in OMEMO.

2.4 Odprtokodna rešitev TS

V tem podpoglavju postavimo zahteve, ki jih bi radi izpolnili z odprtokodno rešitvijo TS. Naša rešitev bi morala (poleg osnovnega namena TS) omogočati:

- uporabo odjemalcev na različnih operacijskih sistemih in napravah;
- usklajevanje pogovorov v odjemalcih na različnih sistemih in napravah;
- arhiv sporočil na strežniku;
- zasebne in javne klepetalnice;
- prenos datotek;
- možnost klica in videoklica; ter
- varno in zasebno komunikacijo.

V naslednjem podpoglavju pregledamo, kako uspešni pri izpolnjevanju teh zahtev so nekateri komercialni ponudniki.

2.5 Komercialni ponudniki TS

Po podatkih z interneta [75] imajo trenutno največ uporabnikov ponudniki WhatsApp (pribl. 1 milijardo), Skype (pribl. 300 milijonov) in Viber (pribl. 249 milijonov). Pri tem nismo upoštevali kitajskega trga. Pregledali smo za katere operacijske sisteme so na voljo (tabela 2.1), katere dodatne storitve nam nudijo (tabela 2.2) in kakšen je nivo varnosti (tabela 2.3).

sistem / ponudnik	WhatsApp	Skype	Viber
Windows	✓	✓	✓
Linux	X	✓	✓
MAC OS X	✓	✓	✓
Android	✓	✓	✓
iOS	✓	✓	✓
v brskalniku	✓	(beta)	X

Tabela 2.1: Podprti operacijski sistemi.

storitev / ponudnik	WhatsApp	Skype	Viber
zasebne klepetalnice	✓	✓	✓
javne klepetalnice	X	X	✓
prenos datotek	✓	✓	✓
tel. preko IP	✓	✓	✓
video pogovor	X	✓	X
usklajevanje med napravami	✓	✓	✓
arhiv na strežniku	X	X	X

Tabela 2.2: Dodatne storitve.

Prenosi datotek imajo določene omejitve glede velikosti in tipov datotek (podatki z uradnih spletnih strani).

WhatsApp: max. 100 MB, ne podpira vseh tipov datotek.

Skype: max. 300 MB, podprti so vsi tipi datotek.

Viber: max. 200 MB, podprti so vsi tipi datotek.

šifriranje / ponudnik	WhatsApp	Skype	Viber
sporočila	✓	le med prenosom	✓
datoteke	✓	le med prenosom	✓
tel. prek IP	✓	le med prenosom	✓
video pogovor	ni na voljo	le med prenosom	ni na voljo

Tabela 2.3: Vrsta šifriranja: znak ✓pomeni šifriranje od konca do konca.

WhatsApp [24] in Viber [74] za šifriranje uporabljata prirejene različice protokola Signal [70], ki je odprtokoden. Skype [63] ponuja zelo skope informacije o načinu šifriranja, ki ne izpolnjujejo zahtev šifriranja od konca do konca.

Brez pretočitve telefonskega imenika na strežnik ponudnika uporaba odjemalcev WhatsApp in Viber ni mogoča. Skype poleg vpisa s telefonsko številko omogoča tudi vpis z uporabniškim imenom in geslom.

Poglavje 3

Odprtokodne rešitve

Protokola SIP (Session Initiation Protocol) [50] in XMPP (Extensible Messaging and Presence Protocol) [4] sta odprtokodna protokola za komunikacijo v stvarnem času. Oba omogočata takojšnje sporočanje (SIP z razširitvijo SIMPLE [49]) in prenos zvoka in videa v stvarnem času (XMPP z razširitvijo Jingle [22]). Največje razlike med njima so zbrane v tabeli 3.1.

	XMPP	SIP
arhitektura	odjemalec - strežnik	vsak z vsakim
format	jezik XML	besedilno orientiran
komunikacija	vedno preko strežnika	ne potrebuje strežnika
kaj prenaša	dejanske podatke	samo signale za sejo
vrste prenosa	TCP /TLS	UDP, TCP /TLS

Tabela 3.1: Razlike med protokoloma XMPP in SIP.

Osnovna naloga protokola SIP je vzpostavitev in nadzor seje. Protokol SIP je le nadzorni protokol, podatki se prenašajo po ločenem kanalu z uporabo ustreznega protokola. Povezava med odjemalcema je ločena na nadzorno in podatkovno raven. Ustvari se neposredna povezava med odjemalcema, pri čemer nastane problem prehajanja skozi požarne zidove oz. naprave za prevajanje omrežnih naslovov. Bolj problematično je tudi doseči asinhronost oz. shranjevanje sporočil na strežniku,

ker zasnova protokola SIP tega ni predvidevala.

Protokol XMPP je bil zasnovan kot protokol za TS. Arhitektura odjemalec - strežnik omogoča prehajanje skozi požarne zidove, ker je odjemalec vedno pobudnik komunikacije. Protokol ne ločuje nadzorne in podatkovne ravni pri pošiljanju sporočil, vsa komunikacija gre preko strežnika - možno je shranjevanje in kasnejše pošiljanje sporočil. Format XML omogoča razširitve osnovnih komunikacijskih elementov in s tem dodatne funkcionalnosti sporočanja. Z razširitvijo Jingle je protokol pridobil tudi možnost vzpostavitve seje za prenos zvoka in videa v stvarnem času.

Za TS je torej bolj primeren protokol XMPP. Poleg tega uživa veliko podporo razvijalcev odprtokodnih odjemalcev in strežnikov za TS. V nadaljevanju poglavja ga opisujemo podrobneje.

3.1 Jedro protokola XMPP

Protokol XMPP je prosto dostopna tehnologija za komunikacijo v stvarnem času, ki uporablja jezik XML kot osnovni format za izmenjavo informacij. Jedro protokola omogoča pošiljanje manjših podatkovnih enot v jeziku XML med dvema entitetama v stvarnem času.

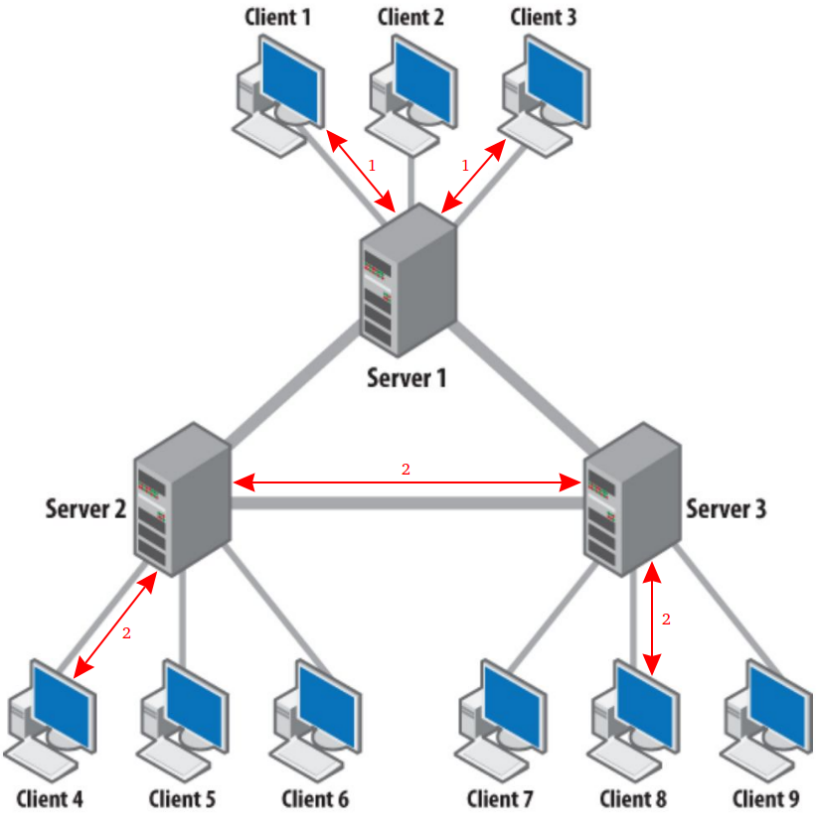
Začetnik razvoja tehnologije je Jeremie Miller [35], ki je leta 1999 objavil prvo verzijo strežnika s tehnologijo XMPP, imenovanega *„jabberd“*. Skupnost razvijalcev odprtokodnih programov je kmalu zatem razvila odjemalce za različne sisteme (Linux, MAC OS X, Windows) in poskrbela za nadaljnji razvoj. Zaradi velikega zanimanja in naraščanja uporabe je organizacija IETF leta 2004 izdala glavne specifikacije XMPP protokola v RFC 3920 [40] in RFC 3921 [41] dokumentih. Objava specifikacij je imela za posledico množično uporabo tehnologije XMPP. Med večjimi družbami, ki so jo uporabile, najdemo Google z Google Talk in Facebook s Facebook Chat sistemom TS [72]. Kasneje sta tako Google kot Facebook podporo XMPP ukinila [15, 17].

Protokol je trenutno definiran v dokumentih RFC 6120 - jedro [55], RFC 6121 - TS in prisotnost [56] ter RFC 7622 - naslovni format [54]. Poleg omenjenih definicij obstaja za protokol veliko razširitev (XEP), ki jih sprejema, razvija in/ali preverja Združenje za standard XMPP (*„XMPP Standard Foundation“*), in so sprejete v

različne razvojne faze [4].

3.1.1 Arhitektura

Tehnologija XMPP deluje preko razpršene mreže odjemalcev in strežnikov, ki lahko medsebojno komunicirajo na način odjemalec - strežnik in strežnik - strežnik, kot je razvidno iz slike 3.1.



Slika 3.1: Arhitektura tehnologije XMPP. Vir: [72].

Fizična pot med dvema odjemalcema je lahko odjemalec - strežnik - odjemalec (pot št. 1 na sliki 3.1) ali odjemalec - strežnik - strežnik - odjemalec (pot št. 2 na sliki 3.1). Fizična pot nikoli ne vsebuje več kot dva strežnika.

Odjemalci in strežniki so v mreži dosegljivi preko globalno enoličnih naslovov enake oblike kot so naslovi za elektronsko pošto, npr. `ana@xmppchat.eu`. Naslov je sestavljen iz lokalnega dela naslova (`ana`) in domenskega dela naslova (`xmppchat.eu`), ki je domena strežnika.

Tehnologija XMPP se je na začetku razvoja imenovala Jabber. Zaradi tega zgodovinskega dejstva se za naslove uporablja oznaka JID (Jabber ID). Obstajata dve vrsti naslovov JID, in sicer:

- osnovni naslovi JID (bare JID) npr. `ana@xmppchat.eu`; in
- polni naslovi JID (full JID) npr. `ana@xmppchat.eu/prenosnik`.

Polni naslovi JID imajo poleg lokalnega in domenskega dela še del z imenom naprave („*resource*“). To je potrebno, ker strežnik XMPP uporabniku omogoča več prijav z istim naslovom, ampak z različnih naprav.

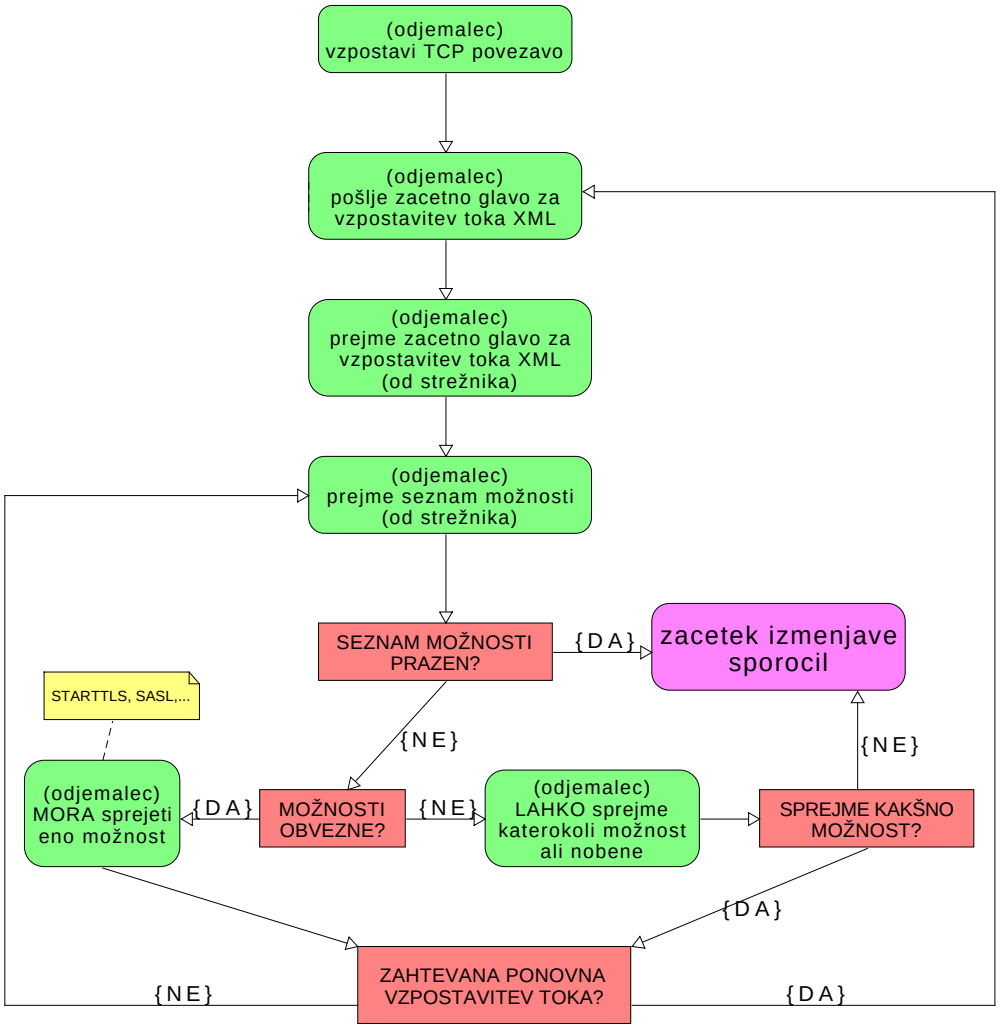
3.1.2 Povezovanje s strežnikom

Cilj uspešne povezave odjemalca s strežnikom je vzpostavitev toka XML preko varne povezave TCP [46]. Tok se začne z XML oznako `<stream>`. Dokler se vzpostavljen tok ne prekine (`</stream>`), je mogoče asinhrono pošiljanje neomejenega števila elementov XML, ki jih opisujemo v poglavju 3.1.3.

Preden se lahko pošiljanje teh elementov začne, se morata strežnik in odjemalec dogovoriti o treh (obveznih) zahtevah:

1. Vzpostavitev varne povezave s protokolom TLS [47].
2. Avtentikacija entitete z mehanizmom SASL [78].
3. Ime naprave („*resource*“), s katere bo odjemalec pošiljal sporočila.

Potek povezovanja odjemalca s strežnikom je prikazan na sliki 3.2.



Slika 3.2: Povezovanje odjemalca in strežnika.

Odjemalec najprej vzpostavi nezavarovano povezavo TCP, vendar takoj zatem strežnik pošlje zahtevo po vzpostavitvi varne povezave s protokolom TLS. Potrebna je ponovna vzpostavitev toka.

Naslednja zahteva je avtentikacija uporabnika z mehanizmom SASL. Strežnik pošlje odjemalcu seznam podprtih načinov avtentikacije. Primer takega seznama je na sliki 3.3. Seznam je tudi prednostna lista - prednost imajo načini avtentikacije

z vrha seznama.

```
<stream:features>
   <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>PLAIN</mechanism>
      <mechanism>SCRAM-SHA-1</mechanism>
      <mechanism>SCRAM-SHA-1-PLUS</mechanism>
   </mechanisms>
</stream:features>
```

Slika 3.3: Načini avtentikacije s SASL mehanizmom.

Seznama strežnika in odjemalca se primerjata. Izbere se način, ki ga podpirata oba in je najvišje na prednostni listi. Način *,,PLAIN“* pomeni, da odjemalec strežniku pošlje uporabniško ime in geslo zakodirano v *,,Base64“* kodo. Primer: uporabnik ,,`ana`“ ima geslo ,,`password`“. Odjemalec zakodira niz ,,`\0ana\0password`“ v *,,Base64“* kodo in strežniku pošlje dobljen niz ,,`XDBhbmFcMHBhc3N3b3Jk`“. Strežnik na svoji strani stori isto in niza primerja - če sta enaka, potrdi avtentikacijo, v nasprotnem primeru pošlje sporočilo o napaki.

Glede na to, da je povezava TCP že šifrirana, je tak način avtentikacije dopusten. V nasprotnem primeru bi bilo geslo med prenosom nezavarovano.

Protokol XMPP privzeto ne dopušča avtentikacije (in sploh kakršnekoli komunikacije) preko nešifrirane povezave TCP. V kolikor je to iz nekih razlogov zaželjeno, se za avtentikacijo lahko uporabi način, ki ne pošilja gesla v čistopisu npr. *,,SCRAM-SHA-1“*. Način avtentikacije je bolj zapleten in ga na tem mestu ne bomo razlagali, podrobnosti lahko najdemo v [77].

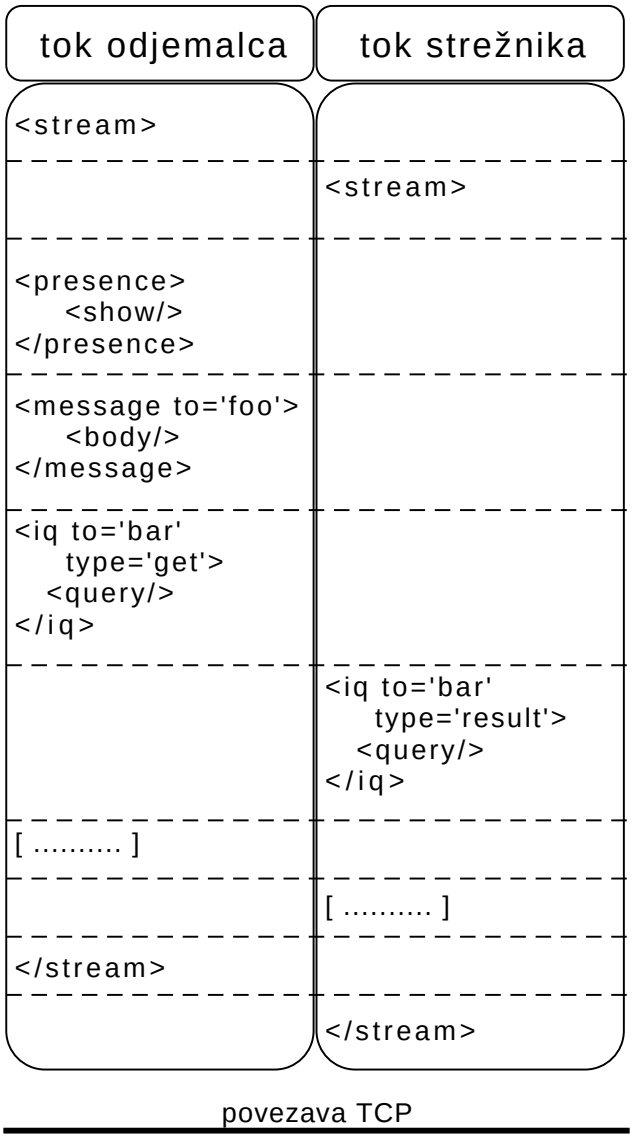
Po uspešni avtentikaciji je potrebna ponovna vzpostavitev toka. Nadaljne možnosti o katerih se pogajata odjemalec in strežnik ne zahtevajo ponovne vzpostavitve toka.

Strežnik mora (po uspešni avtentikaciji) za vzpostavljeno sejo določiti še napravo, s katere bo odjemalec pošiljal sporočila. To je potrebno za pravilno naslavljanje sporočil. Uporabnik se namreč lahko prijavi z različnih naprav in/ali odjemalcev. Uporabnikov osnovni naslov JID se tako razširi v polni.

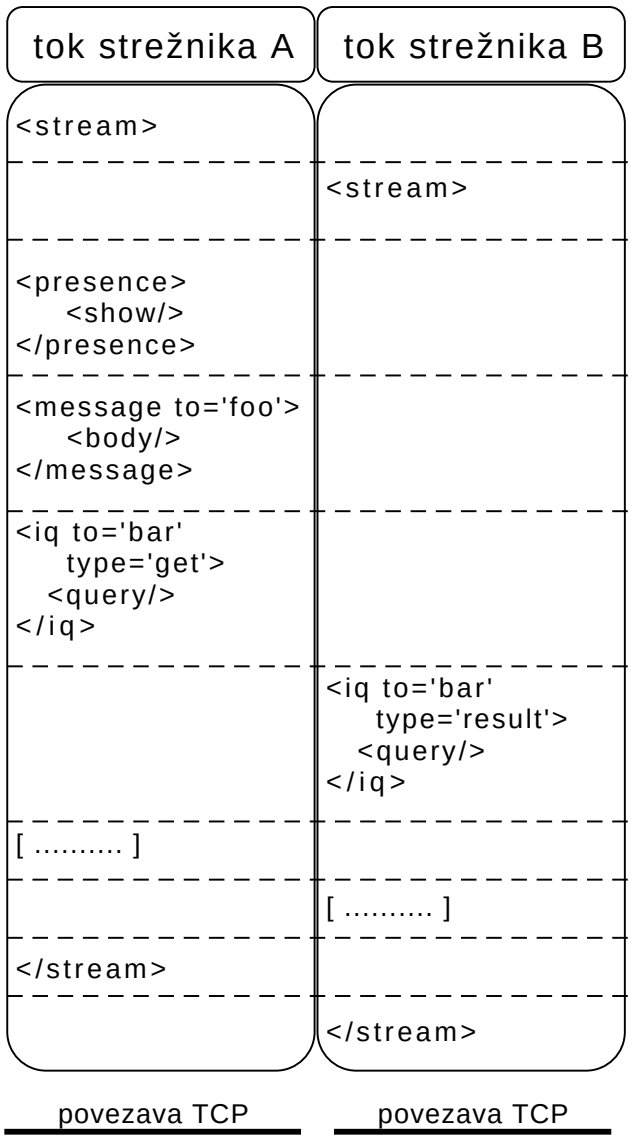
Odjemalec s strežnikom komunicira skozi vrata 5222 preko ene povezave TCP. Dva strežnika med seboj komunicirata skozi vrata 5269 preko dveh povezav TCP. Dve povezavi TCP med strežnikoma sta potrebni zaradi večjega prometa in boljše koherentnosti poslanih zahtev.

V primeru povezave odjemalec - strežnik se ustvarita dva enosmerna toka XML (preko ene povezave TCP). Podatki XML se prenašajo po enem toku v smeri odjemalec → strežnik in po drugem toku v smeri strežnik → odjemalec. Primer je na sliki 3.4.

V primeru povezave strežnik - strežnik se ustvari en tok XML na vsaki povezavi TCP. Primer je na sliki 3.5.



Slika 3.4: Povezava med odjemalcem in strežnikom.



Slika 3.5: Povezava med dvema strežnikoma.

3.1.3 Komunikacijski elementi protokola XMPP

Po uspešni vzpostavitvi toka XML odjemalec in strežnik lahko pričneta z izmenjevanjem komunikacijskih elementov XML (*„XML stanzas“*). Obstajajo trije osnovni elementi: `<message/>`, `<presence/>` in `<iq/>`.

Element `<message/>` se uporablja za prenos informacij (npr. sporočila med uporabniki, alarmi, obvestila). Sporočil te vrste se ne potrjuje oz. ne zahtevajo odgovora.

Element `<presence/>` naznanja prisotnost uporabnika v omrežju uporabnikom, ki so na njegovo prisotnost „naročeni“. Poleg prisotnosti v omrežju lahko naznanja tudi status uporabnika (npr. odsoten, na kosilu,...).

Element `<iq/>`, kar pomeni *„Info/Query“*, lahko sprašuje po določenih informacijah ali informacije oz. zahteve podaja. Vedno mora dobiti odgovor oz. povratno informacijo o (ne)uspešnosti izvedbe. Zahteve in odgovori nanje imajo enako identifikacijsko označbo za sledenje, kar omogoča asinhrono pošiljanje zahtev. Odjemalec tako lahko pošlje neomejeno število zahtev, ne da bi moral čakati na odgovor strežnika. Strežnik bo poslal odgovor dinamično - takoj, ko bo na voljo.

Obstaja tudi pet atributov, ki so skupni trem osnovnim elementom. To so atributi `to`, `from`, `type`, `id` in `xml:lang`.

Atributa `to` in `from` vsebujeta prejemnikov oz. pošiljateljev naslov JID.

Atribut `type` specificira namen in vsebino elementov `<message/>`, `<presence/>` oz. `<iq/>`. Element `<iq/>` npr. ima štiri različne atribute vrste `type`: `type = get` (zahteva informacijo), `type = put` (podaja informacijo), `type = result` (odgovor na uspešno izvedbo zahteve) in `type = error` (odgovor na neuspešno izvedbo zahteve).

Atribut `id` se uporablja za sledenje določenemu elementu. V primeru napake prejemnik lahko pošiljatelju pošlje sporočilo o napaki, ki vsebuje atribut `id` poslanega elementa, ki je povzročil napako. Za element `<iq/>` je atribut `id` obvezen zaradi sledenja odgovora na zahtevo. Za elementa `<message/>` in `<presence/>` je priporočljiv.

Atribut `xml:lang` definira jezik vsebine (sporočila), ki je namenjena uporabnikom. Atribut postavi pobudnik komunikacije pri vzpostavitvi toka in s tem definira privzeti jezik. V kolikor ga ne postavi na začetku komunikacije, mora vsak element,

ki vsebuje za uporabnika namenjeno vsebino, vsebovati tudi element `xml:lang`.

Strežnik elemente usmerja različno glede na vrsto elementa, prav tako jih različno obdeluje tudi odjemalec. Osnovni trije elementi imajo podrejene elemente, ki določajo vsebino. Vsebina je bodisi prikazana uporabniku bodisi procesirana samodejno, kot je določeno z opisom specifikacije vsebine v protokolu ali njegovi razširitvi.

Protokol XMPP ohlapno definira največjo dovoljeno velikost komunikacijskega elementa, ki ne sme biti nastavljena na manj kot 10 kB (10.000 bajtov) [55]. Ne definira pa največje možne velikosti elementa, ki iz varnostnih razlogov (napad z ohromitvijo storitve) tudi potrebuje omejitev. Omejitev je prepuščena skrbniku strežnika, privzeta je ponavadi 64 kB. Skrbnik strežnika lahko omeji tudi število elementov v določeni časovni enoti. Več o varnostnih problemih in omejitvah najdemo v razširitvi XEP-0205 [60].

3.1.4 Upravljanje kontaktov in prisotnost v omrežju

Kontakti, ki so uporabniku vidni v omrežju in s katerimi lahko komunicira, so shranjeni na posebnem seznamu na strežniku XMPP. Za dodajanje kontakta na seznam je potrebno poznati njegov naslov JID, mu poslati prošnjo za odobritev poznanstva in počakati na njeno potrditev. Po uspešni obojestranski potrditvi se kontakt doda na seznama kontaktov obeh vpletenih.

Informacija o prisotnosti v omrežju se po uspešnem povezovanju s strežnikom XMPP pošlje vsem kontaktom z uporabnikovega seznama. Prav tako strežnik naredi poizvedbo o prisotnosti kontaktov, ki so na uporabnikovem seznamu, in vrne stanje prisotnosti.

3.2 Razširitve XEP (XMPP Extension Protocols)

Protokol XMPP je razširljiv - vsakdo lahko doda funkcionalnost, ki jo potrebuje, in jo vključi v delovanje protokola. Nekatere funkcionalnosti so splošno zahtevane

oz. rešujejo problem mnogih uporabnikov, zato se s postopkom, opisanem v XEP-0001 [57, Pogl. 8], dodajo kot uradne razširitve protokola XMPP.

Razširitev protokola je zelo veliko (trenutno čez tristo). Združenje za standard XMPP je zato vzpostavilo register imen (XEP-0053 [59]), v katerega se vsaka razširitev vpiše s svojim unikatnim imenom. Trenutna oblika tega imena je `urn:xmpp:KratkoIme[:PodIme]`, v preteklosti so se sicer uporabljale tudi drugačne oblike. *„KratkoIme“* mora biti unikatno za neko razširitev, *„PodIme“* pa mora biti unikatno v tej razširitvi.

Mehanizem, definiran z razširitvijo XEP-0030 [51], skrbi za samodejno odkrivanje podprtih storitev. Odjemalec po uspešni povezavi s strežnikom pošlje poizvedbo o podprtih razširitvah XEP. Strežnik vrne seznam, v katerem so unikatna imena vseh razširitev, ki jih podpira. Odjemalec izbere razširitve, ki jih podpira tudi sam, in pošlje strežniku obvestilo, da je storitev omogočena. Na tak način se vključi dodatne storitve v protokol XMPP.

Nekatere razširitve so implementirane le s strani strežnika ali odjemalca, nekatere pa potrebujejo implementacijo na obeh straneh.

V nadaljevanju opisujemo delovanje razširitev, ki izpolnjujejo zahteve (v domeni protokola XMPP) za odprtokodno rešitev TS iz poglavja 2.4.

3.2.1 Protokol Jingle

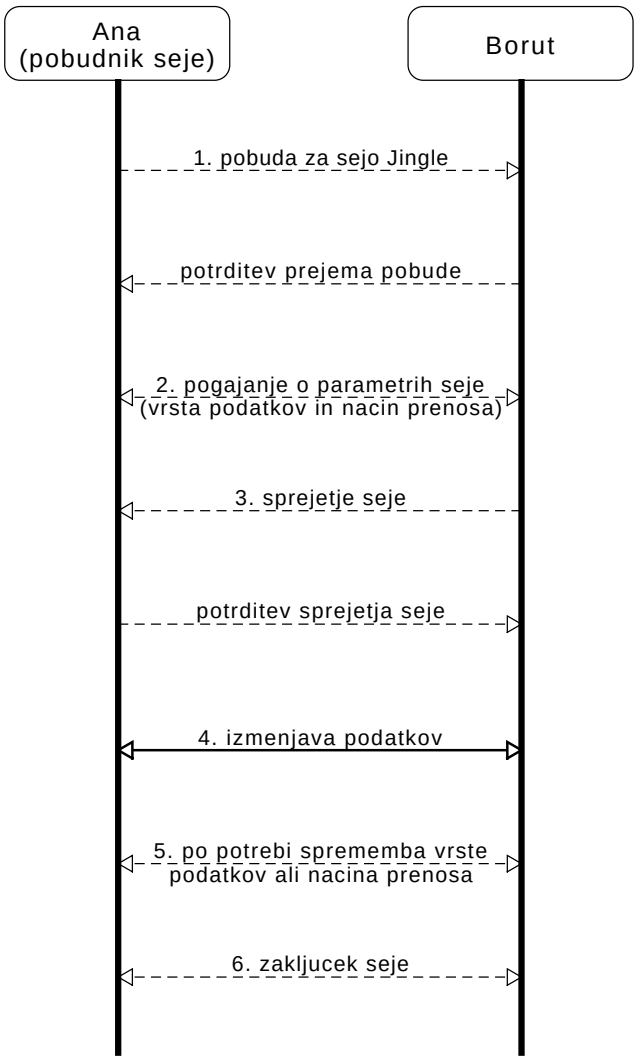
Jingle je protokol za prenos podatkov, ki uporablja že vzpostavljen kanal XMPP za signalizacijo, za dejanski prenos pa vzpostavi nov komunikacijski kanal med uporabnikoma, neodvisno od strežnika XMPP. Odvisno od vrste vsebine, se za prenos uporabi ustrezen protokol. Jingle se uporablja za prenos zvoka in videa v stvarnem času in kot novejši, izboljšan način prenosa datotek.

Pri ustvarjanju neposredne povezave med dvema uporabnikoma prihaja do problemov zaradi prevajanja omrežnih naslovov (NAT) oz. prehoda skozi požarne zidove. Glede na vrsto podatkov, ki se prenašajo, protokol Jingle uporablja različne tehnološke rešitve za vzpostavitev povezave:

- protokol ICE (*„Interactive Connectivity Establishment“*) [48] za prenos podatkov z UDP [45] načinom prenosa (zvok, video); in

- protokol SOCKS5 [27] za prenos podatkov s TCP načinom prenosa (za prenos datotek).

Osnovno ogrodje protokola je definirano z razširitvijo XEP-0166 [22] in prikazano na sliki 3.6.



Slika 3.6: Ogrodje protokola Jingle.

Potek seje:

1. Pobudnik pošlje sporočilo o začetku seje, ki vsebuje informacije o vrsti podatkov (zvok, video, datotečni prenos) in načinu prenosa (UDP, ICE, TCP). Naslovnik potrdi prejem pobude.

2. Strani se dogovorita o parametrih (različni kodeki za zvok in video, ime in velikost datoteke, način prenosa)

3. Naslovnik glede na uspešnost dogovora o parametrih sprejme oz. zavrne sejo (nepodprti kodeki,...). Pobudnik potrdi sprejetje/zavrnitev seje.

4. V primeru sprejetja seje se začne prenos podatkov po ločeno vzpostavljenem podatkovnem kanalu.

5. Protokol dovoljuje dinamično spreminjanje dogovorjenih parametrov (npr. izbere boljšo povezavo za prenos, zvoku doda še video,...).

6. Strani zaključita sejo.

V XEP-0166 je definirano samo osnovno ogrodje protokola Jingle, ki je v bistvu skupek različnih razširitev. V točki 2 poteka seje se, glede na vrsto podatkov, določijo parametri in izbira prenosa, ki so definirani v

- XEP-0167 [12] za prenos zvoka in videa s protokolom RTP [26]; in

- XEP-0234 [67] za prenos datotek.

Podrobnosti o samem prenosu podatkov in izbiri ustreznih kandidatov (naslov IP in vrata) pa so definirane v:

- za povezave UDP:
  XEP-0176 [36], ki uporablja tehnologijo ICE [48] za prehod skozi naprave za prevajanje omrežnih naslovov oz. požarne zidove, in
  XEP-0177 [13], ki je uporaben le v primeru, če se strani v omrežju ,,vidita“ in lahko sami določita ustrezen naslov IP in vrata za vzpostavitev neposredne povezave; ter

- za povezave TCP:
  XEP-0260 [20], ki uporablja protokol SOCKS5 [27] z vmesnim strežnikom

preko katerega se vrši prenos datotek, in
XEP-0261 [61], ki uporablja prenos znotraj že vzpostavljenega toka XML (po protokolu XMPP).

3.2.2 Drugi načini prenosa datotek

Pred razvojem protokola Jingle je že obstajalo ogrodje za prenos datotek, ki definira parametre datoteke za prenos (ime, velikost) in način prenosa (SOCKS5 ali znotraj XML toka). Definicijo ogrodja najdemo v XEP-0096 [39].

Ogrodje omogoča samodejno ponovno vzpostavitev seje v primeru neuspešnega pogajanja o prenosu preko vmesnega strežnika po protokolu SOCKS5 (XEP-0065 [30]). V tem primeru se v naslednji seji prenos izvrši znotraj toka XML (XEP-0047 [52]). Tak način prenosa datoteko zakodira v *„Base64“* kodo in to kodo zaradi omejitve velikosti komunikacijskih elementov razdeli na ustrezno majhne dele. Dele pošilja znotraj toka XML, vsak del ima svojo zaporedno številko.

Starejše ogrodje ni združljivo z ogrodjem Jingle. Glavna razlika je v načinu dogovarjanja o prenosu. V primeru neuspeha, protokol Jingle ne zahteva ponovne vzpostavitve seje in pošiljanja parametrov, ampak se dogovarjanje o parametrih dogaja dinamično. Starejše ogrodje vsakič vzpostavi novo sejo.

Obstaja še en način prenosa, pri katerem se datoteka naloži na strežnik HTTP, prejemnik pa v svojem odjemalcu dobi naslov URL, s katerega lahko prenese datoteko (XEP-0363 [19]). Ta način prenosa je uporaben v situacijah, ko:

- prejemnik datoteke ni prijavljen v omrežju (asinhronost);
- prejemnik uporablja različne naprave; ali
- hočemo deliti datoteko z večimi uporabniki v skupinskem pogovoru.

Strežnik XMPP mora v tem primeru imeti nastavljeno omejitev velikosti datoteke, ki se lahko prenese in shrani na tak način.

3.2.3 Usklajevanje sporočil na napravah

Razširitev XEP-0280 [37] definira način usklajevanja sporočil na napravah. Tip elementa `<message/>`, ki se lahko usklajuje, mora biti `type = chat` ali `type = normal`. Gre za sporočila, ki so poslana med dvema uporabnikoma (od katerih ima lahko en ali oba več naprav). Sporočil poslanih v klepetalnicah, alarmov, napak itd. se ne usklajuje.

Ko strežnik dobi sporočilo, ki je upravičeno za uskladitev, ga ovije v novo sporočilo z atributoma `from` in `to`, ki vsebujeta uporabnikov osnovni JID naslov (`from`) in razširjen(e) JID naslov(e) (`to`) z imenom naprave. To stori za vsako napravo posebej, razen za napravo, na kateri je prejel originalno sporočilo.

Razširitev lahko podpirajo tako odjemalci kot strežniki. Strežnik, ki razširitev podpira, bo v vsakem primeru, tudi če dobi sporočilo od odjemalca, ki razširitve ne podpira, razposlal sporočilo upravičenim odjemalcem. Odjemalec, ki razširitve ne podpira, pa kopij sporočil ne more prejemati.

3.2.4 Arhiv sporočil na strežniku

Razširitev XEP-0313 [65] omogoča:

- samodejno usklajevanje zgodovine sporočil med večimi odjemalci;
- prikazovanje zgodovine sporočil glede na čas; ter
- samodejno pretakanje in prikazovanje sporočil iz arhiva v primeru, ko uporabnik išče po preteklih sporočilih.

Arhiv lahko vsebuje sporočila uporabnikov, klepetalnic, strežnikov in tudi shranjene elemente drugih razširitev. Vsako shranjeno sporočilo mora vsebovati:

- časovni žig časa pošiljanja ali prejema;
- naslov prejemnika (poslana sporočila) ali pošiljatelja (prejeta sporočila); ter
- unikatno identifikacijsko oznako.

Odjemalec naredi poizvedbo z elementom `<iq/>`, ki vsebuje njegov naslov JID in ostale parametre, kot so npr.: zahteva po vseh sporočilih, čas po katerem zahteva vsa sporočila, naslov JID uporabnika ali sobe od katere zahteva arhiv, itd.

Uporabnik lahko nastavi (v kolikor je to implementirano v odjemalcu) arhiviranje sporočil samo za določene naslove JID iz seznama kontaktov. Če je arhiviranje omogočeno, se shranjujejo sporočila tipa `type = chat` ali `type = normal`, za klepetalnice `type = groupchat`.

V kolikor uporabnik uporablja šifriranje od konca do konca, je v določenih primerih arhiviranje nesmiselno, ker se shranjenih šifriranih sporočil v prihodnosti ne da več dešifrirati. V tem primeru mora implementacija šifrirnega protokola vsakemu sporočilu dodati oznako, ki preprečuje arhiviranje.

3.2.5 Klepetalnice

Za razliko od drugih tehnologij, kot je npr. IRC [29], osnovna uporaba TS predvideva pogovor med dvema uporabnikoma. Razširitev XEP-0045 [58] omogoča ustvarjanje klepetalnic tudi s protokolom XMPP. Razširitev je precej obsežna in omogoča veliko različnih načinov klepetalnic:

- javne ali zasebne;
- dolgoročne ali začasne;
- samo za člane ali odprte;
- z ali brez moderatorja; ter
- ne-anonimne (uporabnik lahko vidi prave naslove JID sodelujočih),
  pol-anonimne (le moderator lahko vidi prave naslove JID sodelujočih) in
  anonimne (noben uporabnik ne vidi pravega naslova JID sodelujočih).

Klepetalnica je identificirana z naslovom JID oblike `ime_klepetalnice@storitev`. Vsak uporabnik, ki se prijavi v klepetalnico, pa dobi (nov) naslov JID oblike `ime_klepetalnice@storitev/vzdevek`.

Sporočila, poslana v klepetalnici, se pošljejo na naslov klepetalnice in razpošljejo na vse naslove sodelujočih. Element `<message/>`, ki se uporablja za pošiljanje sporočil, je v tem primeru posebnega tipa `type = groupchat`.

Ob prijavi uporabnika v klepetalnico se pošlje element `<presence/>` na naslov klepetalnice. Uporabnik si mora izbrati svoj vzdevek, s katerim se bo predstavljal v klepetalnici. Informacija o njegovi prisotnosti se z elementom `<presence/>`, ki

zdaj vsebuje njegov vzdevek, razpošlje vsem sodelujočim. Prav tako uporabnik preko elementa `<presence/>` dobi informacijo o prisotnosti vseh sodelujočih (z vzdevki) v klepetalnici.

Poglavje 4

Orodja za zagotavljanje varnosti

Za zagotavljanje varnosti komunikacije se uporablja kombinacija simetrične in asimetrične kriptografije [43].

Simetrična kriptografija uporablja simetrične ključe - en ključ šifrira in dešifrira podatke. Simetrične ključe za šifriranje sporočil lahko dobimo z izmenjavo *„Diffie - Hellman“* (D-H) [21].

Asimetrična kriptografija uporablja koncept javnega in zasebnega ključa, ki sta medsebojno odvisna - kar je šifrirano z javnim ključem lahko dešifrira le zasebni ključ in obratno, primer asimetrične kriptografije je npr. RSA [28].

Kot smo že omenili v poglavju 2.3, enotna vizija o varnostnem protokolu TS ne obstaja. Obstaja pa skupek varnostnih zahtev, ki preprečujejo različne vrste napadov in jih lahko razdelimo v tri sklope:

1. Vzpostavitev zaupanja med sodelujočimi v pogovoru.
   Avtentikacija sogovornikov - napadalcu prepreči predstavljanje z lažno identiteto.

2. Varna izmenjava sporočil.
   Zaupnost sporočil - napadalcu prepreči branje sporočil.
   Integriteta sporočil - napadalcu prepreči spreminjanje sporočil.
   Avtentikacija sporočil - napadalcu prepreči pošiljanje lažnih sporočil.

Prihodnja varnost - v primeru, da napadalec prestreza šifrirano komunikacijo in nekoč v prihodnosti pridobi trenutne šifrirne ključe, z njimi ne more dešifrirati pretekle komunikacije.
Vzvratna varnost - v primeru, da je napadalec v preteklosti pridobil šifrirni ključ, z njim ne more dešifrirati prihodnje komunikacije.
Nedokazljivost avtorja sporočil.

3. Zasebnost pri prenosu sporočil.
   Preprečuje razkrivanje metapodatkov (npr. IP naslov pošiljatelja in prejemnika).
   Nedokazljivost obstoja pogovora.

Za doseganje varnosti od konca do konca mora varnostni protokol izpolnjevati zahtevo pod točko 1 in vsaj zaupnost, integriteto in avtentikacijo sporočil pod točko 2. Z ostalimi zahtevami pod točko 2 protokol preprečuje dodatne možnosti napadov.

Izpolnitev zahtev pod točko 3 je v odjemalcih XMPP mogoča z uporabo mreže *,,Tor“* [7]. V mreži *,,Tor“* se sporočilo pošlje po vnaprej naključno izbrani poti preko večih vmesnih strežnikov. Sporočilo je šifrirano v več plasteh, v vsakem vozlišču se dešifrira ena plast, do zadnjega strežnika, kjer se razkrije originalno sporočilo in pošlje na končni cilj. Vmesni strežniki poznajo le naslov predhodnega in naslednjega vmesnega strežnika in so naključno izbrani.

Uporaba mreže *,,Tor“* zagotavlja anonimnost pošiljatelja, anonimnost sodelujočih v pogovoru in nedokazljivost obstoja pogovora [66].

Za izpolnjevanje zahtev pod točkama 1 in 2 je v odjemalcih XMPP mogoča izbira med tremi varnostnimi protokoli: OpenPGP [71], OTR [9] in OMEMO [68]. Katere zahteve izpolnjujejo in na kakšen način to dosežejo opisujemo v nadaljevanju poglavja.

4.1 Avtentikacija sogovornikov

OpenPGP Ana ima javni in zasebni ključ. Javni ključ deli (preko elektronske pošte, sistema javnih strežnikov, USB ključka,...) z Borutom. Borut mora preveriti

istovetnost ključa s primerjanjem zgoščene vrednosti prejetega ključa in ključa, ki ga ima Ana. To mora storiti v sodelovanju z Ano preko nekega drugega kanala (telefonsko, v živo,...). Ko se prepriča, da ima pravi ključ, s svojim zasebnim ključem podpiše informacijo o identiteti lastnika prejetega ključa - s tem se gradi mreža zaupanja, v kateri oseba A, ki zaupa osebi B, lahko zaupa tudi osebi C, če ji zaupa oseba B. Anin javni ključ je tako potrjen in Borut ga lahko uporablja za potrjevanje Aninega digitalnega podpisa.

V odjemalcih XMPP se avtentikacija uporabnika zgodi na začetku komunikacije, pri prijavi v mrežo XMPP, v razdelku `<presence/>`. Ana podpiše nek (lahko tudi prazen) XML niz, ki ga Borut preveri z Aninim javnim ključem. Uporaba protokola OpenPGP je opisana v razširitvi XEP-0027 [38].

OTR Protokol OTR ob prvi avtentikaciji uporabnika uporablja protokol *„Socialist Millionaire Protocol“* - SMP [16] ali ročno potrjevanje zgoščene vrednosti javnega ključa.

Protokol SMP je primer avtentikacije z izzivom. Ana zastavi vprašanje Borutu, za katerega je prepričana, da bo vedel odgovor. Ana in Borut vsak na svoji strani napišeta odgovor na vprašanje, ki se nato primerja. V kolikor sta odgovora enaka je avtentikacija oseb uspešna.

Za ročno potrjevanje morata Ana in Borut primerjati zgoščeni vrednosti njunih javnih ključev preko nekega drugega kanala (npr. telefonski klic).

Po uspešni prvi potrditvi javnega ključa se ta vpiše v uporabnikov seznam javnih ključev v odjemalcu. Prihodnja avtentikacija poteka samodejno.

OMEMO Zaradi krajših ključev se ne primerja zgoščenih vrednosti, ampak kar ključe. Protokol OMEMO ob prvi avtentikaciji uporabnika uporablja ročno potrjevanje. Ana in Borut morata primerjati njuna javna ključa preko nekega drugega kanala.

Po uspešni potrditvi javnega ključa se ta vpiše v uporabnikov seznam javnih ključev v odjemalcu. Prihodnja avtentikacija poteka samodejno.

4.2 Varna izmenjava sporočil

OpenPGP Trenutna uporaba protokola OpenPGP v odjemalcih XMPP (XEP-0027 [38]) ni standardizirana in tudi ni najbolj ustrezna. Izmenjana sporočila niso digitalno podpisana, kar odpira možnosti napadov s ponovitvijo (,,*replay attack*“). V razvoju je boljša različica uporabe XEP-0374 [8], ki bi uvedla digitalno podpisovanje s časovnim žigom (,,*timestamp*“) za vsako poslano sporočilo. Avtorstvo vsakega sporočila bi bilo dokazljivo, kar je lahko uporabno v določenih poslovnih okoljih. Razširitev še ni implementirana v nobenem odjemalcu.

Šifriranje sporočil poteka z uporabo simetrične in asimetrične kriptografije. Sporočilo se najprej šifrira z naključno ustvarjenim simetričnim ključem. Simetrični ključ se šifrira z javnim ključem prejemnika. Sporočilo in šifriran simetrični ključ se pošljeta prejemniku, ki najprej dešifrira simetrični ključ s svojim zasebnim ključem, nato pa še samo sporočilo z dobljenim simetričnim ključem.

S takim načinom šifriranja protokol OpenPGP izpolnjuje zaupnost, integriteto in nedokazljivost avtorja sporočil (trenutna uporaba brez digitalnega podpisovanja sporočil). Kot že povedano sporočila niso avtenticirana, ne obstojata tudi prihodnja in vzvratna varnost.

OTR Ana in Borut pred začetkom izmenjave sporočil z D-H izmenjavo vzpostavita skupno skrivnost. Skupna skrivnost se uporabi za izpeljavo simetričnega šifrirnega ključa, s katerim se šifrira sporočilo - tako je zgotovljena zaupnost sporočil. Postopek se imenuje avtenticirana izmenjava ključev in se izvaja po protokolu SIGMA [31].

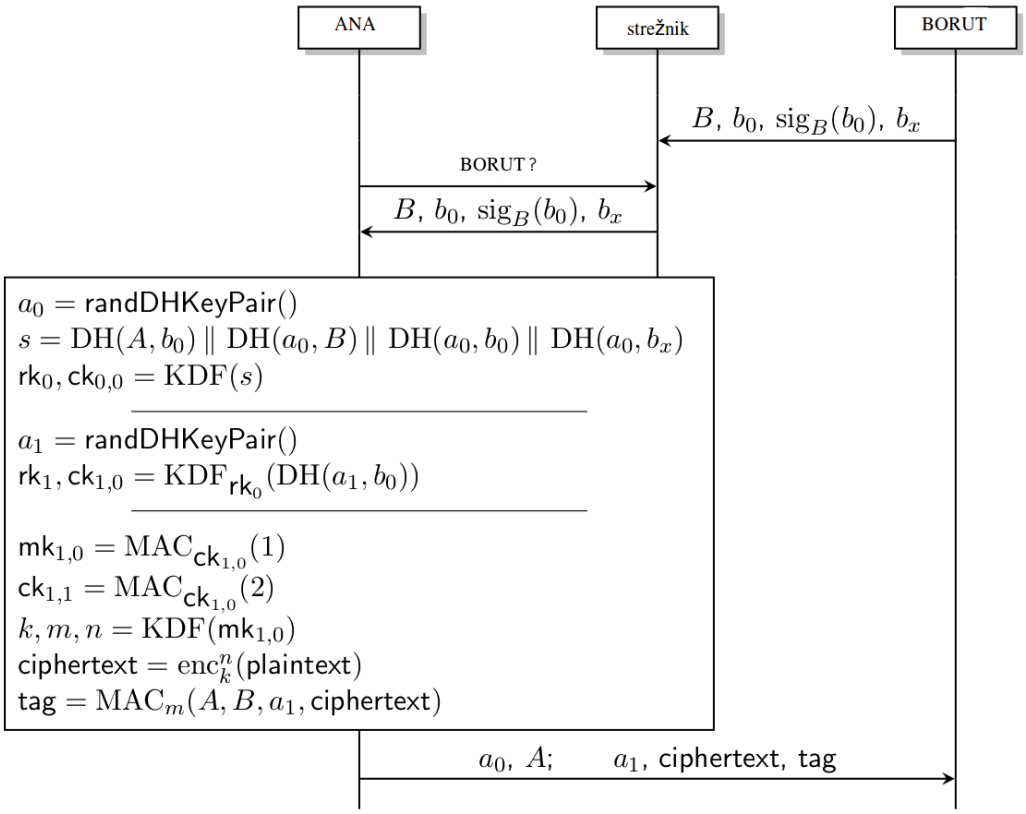
Pri vsaki izmenjavi sporočil Ana in Borut vzpostavita novo skupno skrivnost iz katere izpeljeta nove šifrirne ključe - s tem je zagotovljena prihodnja varnost.

Sporočila so avtenticirana z uporabo postopka SHA1-HMAC [11]. Ključ MAC, uporabljen v postopku avtentikacije sporočil, je zgoščena vrednost simetričnega ključa, ki je izpeljan iz skupne skrivnosti - s tem je zagotovljena avtentikacija in integriteta sporočil.

Za izpolnitev zahteve o nedokazljivosti avtorstva sporočil se (po uporabi za določeno sporočilo) ključ MAC objavi javno. To pomeni, da ga lahko uporabi vsakdo in s tem postane ,,avtor“ sporočila. Tudi če se ključi MAC ne bi objavili javno, sta možna avtorja sporočil tako Ana kot Borut, ker ključi izhajajo iz skupne

skrivnosti.

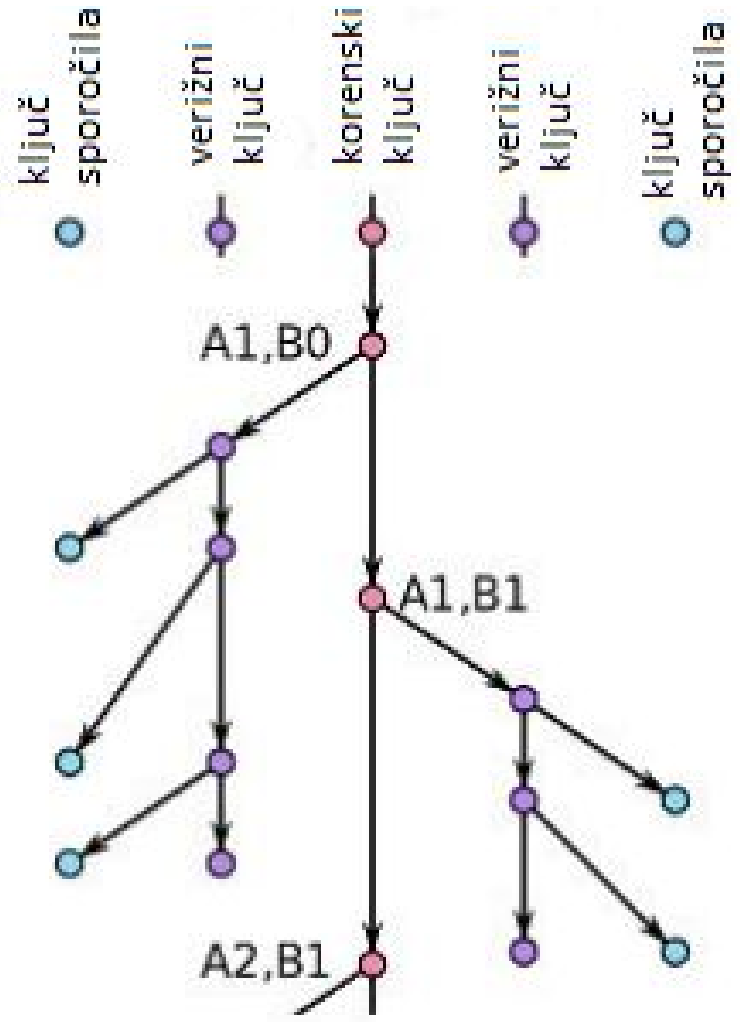
OMEMO Protokol OMEMO je izpeljanka protokola Signal [70] za uporabo z odjemalci XMPP. Potek protokola je razviden iz slike 4.1.



Slika 4.1: Potek protokola OMEMO. Vir: [73]

Borut na strežnik prenese svoj kriptografski material. Ko hoče Ana prvič komunicirati z Borutom, najprej zahteva prednaložen kriptografski material s strežnika. S trojno D-H izmenjavo [34] pride do skupne skrivnosti s, iz katere izpelje prvi korenski ključ rk_0 („root key“). Od tu naprej protokol uporablja algoritem *„Double Ratchet“* [69], ki ga opisujemo v nadaljevanju. Ogrodje je prikazano na sliki 4.2.

Z vsakim prejetim sporočilom Ana posodobi korenski ključ. Za izpeljavo novega korenskega ključa rk_1 in novega verižnega ključa $ck_{0,1}$ („chain key“) uporabi funkcijo „HKDF“ [14] s prejšnjim korenskim ključem rk_0 - s tem je zagotovljena



Slika 4.2: Algoritem *„Double Ratchet“*. Vir: [69]

vzvratna varnost.

Za vsako poslano sporočilo Ana ustvari ključ sporočila $mk_{1,0}$ („message key“) in nov verižni ključ $ck_{1,1}$. Pridobi ju iz starega verižnega ključa $ck_{0,1}$ z uporabo postopka „SHA256-HMAC“ [11] - s tem je zagotovljena prihodnja varnost.

Ana nato z uporabo „HKDF“ in ključa sporočila *mk* ustvari tri ključe: šifrirni ključ k in inicializacijski vektor n (s katerima šifrira sporočilo) in avtentikacijski ključ m. S ključem m in postopkom „SHA256-HMAC“ izračuna avtentikacijsko označbo nad šifriranim sporočilom - s tem je zagotovljena avtentikacija in integri-

teta sporočil.

Borutu pošlje šifrirano sporočilo, avtentikacijsko označbo in svoj kriptografski material.

Poglavje 5

Ovrednotenje in analiza

5.1 Postavitev strežnika

Obširen seznam strežnikov XMPP najdemo na uradni strani standarda XMPP [4]. Za potrebe diplomske naloge smo izbrali strežnik Prosody [76], ki podpira operacijske sisteme Linux, MAC OS X in BSD. Strežnik Prosody je uradna izbira Fundacije za standard XMPP [3]. Priporočen je v vodiču za postavitev komunikacij v stvarnem času kot primer dobre prakse [44]. Napisan je v programskem jeziku Lua, odlikuje ga majhna poraba sistemskih virov, preprostost konfiguracije in razširljivost.

Strežnik Prosody smo namestili na Linux distribuciji Lubuntu, ki je okleščena verzija distribucije Ubuntu, in je primerna tudi za manj zmogljive računalnike. Potek namestitve je opisan v nadaljevanju.

Umestitev strežnika Dodajanje lokacije za prenos najnovejše verzije:

```
# echo deb http://packages.prosody.im/debian $(lsb_release -sc) main |
sudo tee -a /etc/apt/sources.list
# wget https://prosody.im/files/prosody-debian-packages.key -O- |
sudo apt-key add -
```

Namestitev strežnika:

```
# sudo apt-get update
```

```
# sudo apt-get install prosody
```

Strežnik želimo povezati v mrežo obstoječih strežnikov XMPP in ga uporabljati od kjerkoli. Za dosego tega cilja smo rezervirali domeno `xmppchat.eu` in ji dodelili statični IP naslov. Na našem usmerjevalniku smo odprli vrata številka 5222 (za povezave z odjemalci) in 5269 (za povezave s strežniki), ki so privzeta vrata, na katerih posluša strežnik Prosody.

V „Manifestu o vseprisotni varnosti in šifriranju povezav“ [53] v mreži XMPP najdemo smernice za varnostne nastavitve strežnika:

- poskrbeti moramo za šifrirano povezavo TLS z avtentikacijo na relacijah odjemalec-strežnik in strežnik-strežnik;
- zaželjena oz. zahtevana je uporaba šifrirnih algoritmov, ki omogočajo prihodnjo tajnost; ter
- zaželjena oz. zahtevana je namestitev certifikatov dobro znanih in zanesljivih izdajateljev.

Namestitev certifikata Za namestitev certifikata smo izbrali *Let's Encrypt* izdajatelja. *Let's Encrypt* je znan in zanesljiv izdajatelj, ki ponuja brezplačne certifikate, katerih pridobitev je preprosta in varna. Namestitev certifikata je opisana v nadaljevanju.
Namestimo program git, s katerim prenesemo agenta Certbot za pridobitev certifikata:

```
# sudo apt-get install git
# git clone https://github.com/certbot/certbot
```

Postavimo se v mapo, kamor smo prenesli agenta Certbot ter ustvarimo certifikat in pripadajoč zasebni ključ za domeno `xmppchat.eu`:

```
# ./certbot-auto certonly --standalone -d xmppchat.eu
```

Certifikat in ključ se ustvarita v mapi `/etc/letsencrypt/live/xmppchat.eu`.
V izogib problemom s pravicami za dostop do datotek prekopiramo datoteki `fullchain.pem` in `privkey.pem` v mapo `/etc/prosody/certs`. Zaradi pravic moramo delati v administratorskem načinu:

```
# sudo -i
# cd /etc/letsencrypt/live/xmppchat.eu
# cp fullchain.pem privkey.pem /etc/prosody/certs
```

Dodatni moduli Eleganten način za kasnejšo lažjo namestitev in posodobitev modulov je prenos vseh obstoječih modulov s pomočjo orodja Mercurial. Namestitev orodja Mercurial:

```
# sudo apt-get install mercurial
```

Nato se postavimo v mapo `/usr/lib/prosody` in prenesemo module:

```
# cd /usr/lib/prosody
# sudo hg clone https://hg.prosody.im/prosody-modules/prosody-modules
```

Ukaz ustvari novo mapo `prosody-modules` z vsemi razpoložljivimi dodatnimi moduli. Kadarkoli želimo posodobiti vse module, se postavimo v omenjeno mapo in zaženemo ukaz:

```
# hg pull -u
```

Po posodobitvi moramo ponovno zagnati strežnik Prosody.

Upravljanje shranjenih sporočil Strežnik bo omogočal shranjevanje sporočil in njihovo sinhronizacijo na različnih napravah. V ta namen namestimo sistem za upravljanje zbirk podatkov MySQL, ustvarimo bazo in uporabnika:

```
# apt-get install mysql-server
# apt-get install mysql-client
# mysql -u root -p
# CREATE DATABASE prosody;
# CREATE USER prosody@localhost;
# SET PASSWORD FOR prosody@localhost= PASSWORD(‘password’);
# GRANT ALL PRIVILEGES ON prosody.* TO prosody@localhost
IDENTIFIED BY ‘password’;
# FLUSH PRIVILEGES;
# exit;
```

5.1.1 Konfiguracija

Vsa konfiguracija strežnika Prosody je shranjena v datoteki `/etc/prosody/prosody.cfg.lua`. Po namestitvi strežnika se v datoteki nahajajo osnovne nastavitve s komentarji.

Konfiguracijska datoteka je razdeljena na dva dela. V prvem delu so globalne nastavitve, ki veljajo za celoten strežnik in vse gostitelje. Gostitelj je domena, ki jo uporabimo za uporabniške račune (npr. `xmppchat.eu`). Ena instanca strežnika lahko streže večim domenam. V drugem delu datoteke nastavljamo posamezne domene, ki se začnejo z vnosom `VirtualHost` ali `Component`. Nastavitve med dvema `VirtualHost` oz. `Component` vnosoma veljajo le za posamezno domeno.

```
---------- 1. DEL KONFIGURACIJSKE DATOTEKE ----------

plugin_paths = { "/usr/lib/prosody/prosody-modules" }; -- pot do
    dodatnih modulov

modules_enabled = {

-- obvezni moduli --
"roster"; -- seznam kontaktov
"saslauth"; -- nacini avtentikacije
"tls"; -- podpira povezave TLS
"dialback"; -- starejsi nacin avtentikacije med strezniki
"disco"; -- odkriva podprte razsiritve
"posix"; -- POSIX

-- dodatni moduli --
"carbons"; -- usklajevanje sporocil na napravah
"smacks"; -- vzpostavljanje toka XML ob krajsih izgubah omrezja
"mam"; -- arhiv sporocil
"csi"; -- prikazovanje stanja uporabnika
"throttle_presence"; -- filtrira poslana sporocila o statusu
"filter_chatstates"; -- filtrira poslana sporocila o aktivnostih
"http"; -- mini streznik HTTP
"http_upload"; -- prenos datotek na streznik
"proxy65"; -- vmesni streznik
```

```
-- neobvezni moduli --
"version"; -- vrne verzijo streznika
"uptime"; -- cas delovanja
"time"; --trenutni cas
"ping"; -- ping-pong
"pep"; -- oglasevanje stanja
"register"; -- moznost registracije iz odjemalca
};

allow_registration = true; -- omogoca ustvarjanje racunov v
    odjemalcih

ssl = {
     key = "/etc/prosody/certs/privkey.pem";
     certificate = "/etc/prosody/certs/fullchain.pem";
     dhparam = "/etc/prosody/certs/dh_2048.pem";
}
-- kljuc, certifikat in D-H parametri za streznik
c2s_require_encryption = true; -- TLS med odjemalcem in
    streznikom
allow_unencrypted_plain_auth = false; -- prepreci nesifrirano
    avtentikacijo
s2s_secure_auth = true; -- zahteva avtentikacijo s certifikatom
    med strezniki


interfaces = { "192.168.1.74" } -- IP naslov na katerem je
    streznik

proxy65_ports = { 5555 } -- vrata za vmesni streznik
proxy65_interfaces = { "192.168.1.74" } -- IP naslov na katerem
    je vmesni streznik

http_ports = { 5280 } -- vrata za streznik HTTP
http_interfaces = { "192.168.1.74" } -- IP naslov na katerem je
    streznik HTTP
https_ports = { 5281 } -- vrata za streznik HTTPS
https_interfaces = { "192.168.1.74" } -- IP naslov na katerem je
```

```
    streznik HTTPS
http_upload_file_size_limit = 100*1024*1024 -- omejitev
    velikosti dat. na HTTP strezniku (100 MB)

pidfile = "/var/run/prosody/prosody.pid"

authentication = "internal_plain" -- gesla v cistopisu
--(internal_hashed -> zgoscene vrednosti)

storage = "sql" -- nastavitve arhiva sporocil
sql = { driver = "MySQL", database = "prosody",username ="
    prosody", password = "prosodypass", host = "localhost" }

log = {
    info = "/var/log/prosody/prosody.log";
    error = "/var/log/prosody/prosody.err"; -- belezenje
    "*syslog";
}



---------------------- 2. DEL KONFIGURACIJSKE DATOTEKE ----------------------

VirtualHost "xmppchat.eu" -- domena streznika

Component "proxy.xmppchat.eu" "proxy65" -- domena vmesnega
    streznika

Component "klepetalnica.xmppchat.eu" -- domena klepetalnice
```

5.1.2 Moduli strežnika Prosody

Moduli potrebni za osnovno delovanje strežnika so naloženi samodejno oz. so privzeto dodani v razdelku `modules_enabled` v prvem delu konfiguracijske datoteke. Module za nekatere dodatne opcije, ki jih omogoča strežnik, dodajamo v ta

razdelek sami. Seznam dodanih modulov z opisi nalog vsebuje:

- `carbons`
  omogoča sinhroniziran potek pogovora na vseh uporabnikovih odjemalcih in napravah, ki so priključeni v omrežje - razširitev ,,XEP-0280: Message Carbons“;

- `smacks`
  v primeru krajših prekinitev povezave v omrežje omogoča odjemalcu obnovitev prekinjene seje in prepreči izgubljanje sporočil - razširitev ,,XEP-0198: Stream Management“;

- `mam`
  shrani sporočila na strežnik in omogoča uporabniku dostop do arhiva - razširitev ,,XEP-0313: Message Archive Management“;

- `csi`
  v kolikor uporabnik odjemalca ne uporablja aktivno, strežnik lahko ustrezno optimizira promet do odjemalca, s tem prihrani na porabi sistemskih sredstev in manj obremenjuje mrežno povezavo - razširitev ,,XEP-0352: Client State Indication“;

- `throttle_presence`
  glede na informacije o aktivnosti uporabnika (modul `csi`) onemogoča ali omogoča obvestila o statusu drugih uporabnikov;

- `filter_chatstates`
  glede na informacije o aktivnosti uporabnika (modul `csi`) onemogoča ali omogoča obvestila o aktivnostih drugih uporabnikov - razširitev ,,XEP-0085: Chat State Notifications“;

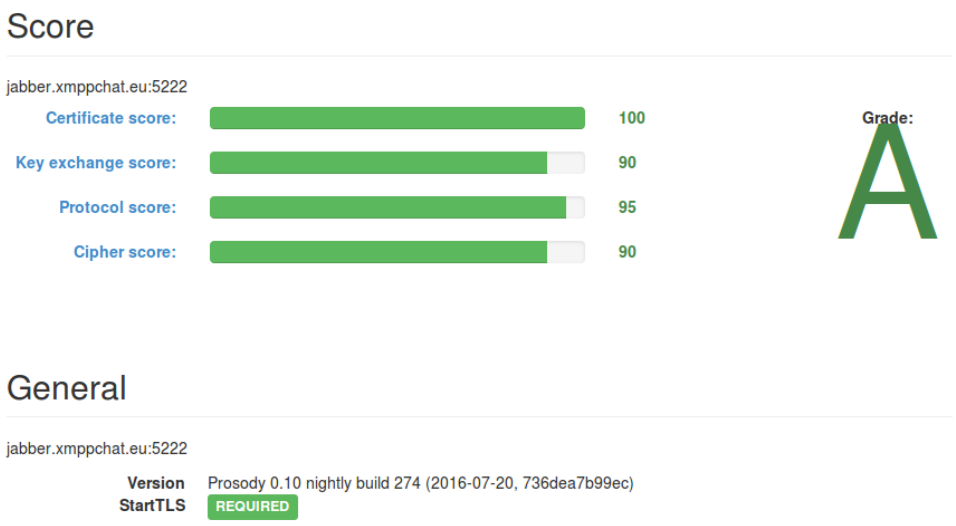
- `http`
  omogoči v Prosody vgrajeni mini strežnik HTTP;

- `http_upload`
  odjemalcem omogoča prenos datotek na strežnik - razširitev ,,XEP-0363: HTTP File Upload“; ter

- `proxy65`
  omogoča prenos datotek med dvema odjemalcema, ki sta povezana preko usmerjevalnika in/ali zaščitena s požarnim zidom in ne bi mogla prenašati datotek drugače.

5.1.3 Preverjanje varnostnih nastavitev

Internetna stran `https://xmpp.net` omogoča varnostni test strežnikov XMPP. Testirali smo strežnik in dobili oceno A, ki je najvišja možna ocena. Nekaj izboljšav bi lahko naredili glede dolžine ključev (4096 namesto 2048 bitni ključi) in izbire algoritmov šifriranja, vendar menimo, da za domačo uporabo to ni potrebno. Rezultati testa so na sliki 5.1.



Slika 5.1: Varnostni test strežnika.

5.2 Ovrednotenje rešitve

5.2.1 Izbira odjemalcev

Izbira odjemalcev za takojšnje sporočanje (TS) s tehnologijo XMPP je zelo obširna, seznam lahko najdemo na uradni strani standarda XMPP [4].

Informacije o podprtih razširitvah XEP so pri večini odjemalcev nepopolne ali celo niso podane, zato je bila potrebna namestitev in pregled podprtih možnosti. Po pregledu smo izbrali odjemalce Pidgin [1], Gajim [32], Jitsi [25], Conversations [18] in ChatSecure [5], ki podpirajo največ različnih možnosti.

V nadaljevanju podajamo informacije o podprtih operacijskih sistemih v tabeli 5.1, iz katere je razvidno, da obstaja vsaj en odjemalec na izbranem operacijskem sistemu, kar omogoča (vsaj) osnovni namen TS (poglavje 2.1) preko vseh sistemov.

V tabeli 5.2 so zbrane informacije o dodatnih storitvah, ki jih podpirajo odjemalci. Pri uspešnosti delovanja dodatnih storitev med odjemalci prihaja do različnih problemov, ki jih opisujemo v poglavju 5.2.2.

V tabeli 5.3 so zbrane informacije o nivoju varnosti pri omenjenih odjemalcih.

sistem / odjemalec	Pidgin	Gajim	Jitsi	Conv.	ChatSecure
Windows	✓	✓	✓	X	X
Linux	✓	✓	✓	X	X
MAC OS X	(Adium [62])	X	✓	X	X
Android	X	X	(beta)	✓	✓
iOS	X	X	X	X	✓

Tabela 5.1: Podprti operacijski sistemi.

storitev / odjemalec	Pidgin	Gajim	Jitsi	Convers.	ChatSecure
zasebne klepetalnice	✓	✓	✓	✓	✓
javne klepetalnice	✓	✓	✓	✓	✓
prenos datotek	✓	✓	✓	✓	✓*
tel. preko IP	✓	✓	✓	X	X
video pogovor	✓	✓	✓	X	X
sinhronizacija	X	✓	✓	✓	X
arhiv na strežniku	X	✓	X	✓	X

Tabela 5.2: Dodatne storitve.

* - prenos datotek je uspešen samo med uporabniki enakega odjemalca

šifriranje / odjemalec	Pidgin	Gajim	Jitsi	Convers.	ChatSecure
sporočila	✓	✓	✓	✓	✓*
datoteke	X	X	X	✓*	✓*
tel. prek IP	X	X	✓*	ni na voljo	ni na voljo
video pogovor	X	X	✓*	ni na voljo	ni na voljo

Tabela 5.3: Vrsta šifriranja: znak ✓pomeni šifriranje od konca do konca.

* - (de)šifriranje je uspešno samo med uporabniki enakega odjemalca

5.2.2 Analiza delovanja

Prenosi datotek

V protokolu XMPP obstaja veliko načinov prenosa datotek, zato prihaja do problemov z združljivostjo med odjemalci. Trenutno stanje združljivosti glede na implementirane razširitve XEP v odjemalcih je podano v tabeli 5.4. (Odjemalec ChatSecure je izvzet, ker omogoča pošiljanje datotek samo med uporabniki ChatSecure.)

	Gajim	Pidgin	Jitsi	Conver.
Gajim	Jingle	SOCKS5	In-band	Jingle
Pidgin	SOCKS5	SOCKS5	In-band	ni podprto
Jitsi	In-band	In-band	In-band	ni podprto
Conver.	Jingle	HTTP*	HTTP*	Jingle

Tabela 5.4: Združljivost načinov prenosov datotek med odjemalci.

* - v kolikor strežnik XMPP to omogoča

Pri testiranju prenosov smo ugotovili, da Pidgin, Jitsi in Conversations implementirane razširitve tudi v praksi pravilno izvajajo.

Pri testiranju odjemalca Gajim smo našli napake zbrane v tabeli 5.5. Pri drugih odjemalcih napak pri (implementiranih) prenosih ni bilo.

način prenosa / OS	Windows	Linux
In-band	ne deluje	deluje, razen s Pidgin
SOCKS5	deluje z napakami	deluje
Jingle In-band	ne deluje	ne deluje
Jingle SOCKS5	deluje	deluje

Tabela 5.5: Napake pri prenosih z odjemalcem Gajim

Orodja za varnost

Odjemalci omogočajo uporabo različnih orodij za varnost. Podatki so zbrani v tabeli 5.6, iz katere lahko razberemo, da je trenutno najbolj podprto šifriranje z uporabo protokola OTR. Odjemalec ChatSecure to vrsto šifriranja sicer omogoča, vendar uporaba z drugimi odjemalci ni možna.

orodje / odjemalec	Pidgin	Gajim	Jitsi	Convers.	ChatSecure
OpenPGP	✓	✓	X	✓	X
OTR	✓	✓	✓	✓	✓*
OMEMO	X	✓	X	✓	X

Tabela 5.6: Orodja za varnost v različnih odjemalcih

* - (de)šifriranje je uspešno samo med uporabniki enakega odjemalca

Zvok in video v stvarnem času

Zvok in video v stvarnem času omogočajo odjemalci Pidgin, Gajim in Jitsi. Najbolj zanesljiv pri delovanju je Jitsi, ki deluje na vseh sistemih (Windows, Linux, MAC OS X) in komunikacijo tudi šifrira od konca do konca. Gajim in Pidgin delujeta samo na sistemu Linux, komunikacije pa ne šifrirata.

Poglavje 6

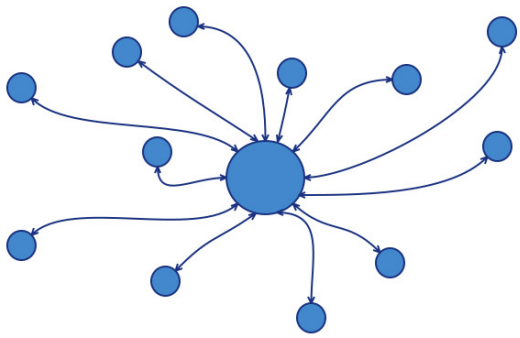
Sklepne ugotovitve

V sklepnih ugotovitvah naredimo primerjavo tehničnih značilnosti in uporabniške izkušnje med odprtokodnimi in komercialnimi rešitvami sistema za takojšnje sporočanje (TS). Poudarimo prednosti in slabosti tehnologije XMPP.

6.1 Tehnična primerjava sistemov

6.1.1 Arhitektura

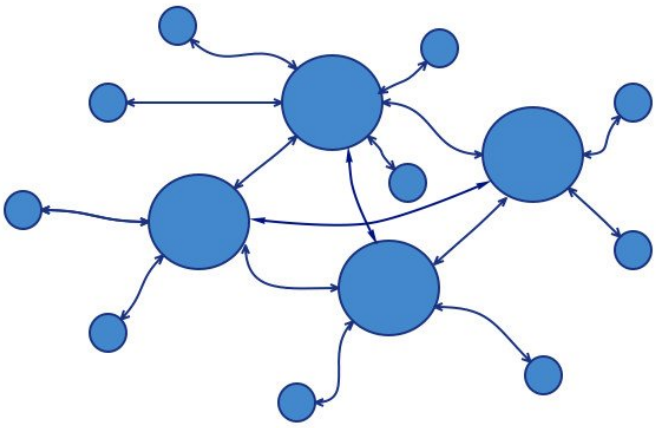
Komercialni ponudniki uporabljajo centralizirano mrežno arhitekturo, ki je prikazana na sliki 6.1.



Slika 6.1: Centralizirana mrežna arhitektura komercialnih ponudnikov.

Komunikacija je možna samo med uporabniki istega ponudnika. Arhitektura omogoča sledenje in hrambo metapodatkov ter njihovo obdelavo, uporabo v komercialne namene, prodajo, predajo na zahtevo varnostnih agencij in podobno. Možno je tudi blokiranje ponudnika s strani vlade, kar se je zgodilo npr. v primeru ponudnika WhatsApp v Braziliji [42].

Tehnologija XMPP uporablja razpršeno mrežno arhitekturo, ki je prikazana na sliki 6.2.



Slika 6.2: Razpršena mrežna arhitektura tehnologije XMPP.

Metapodatki so porazdeljeni v ogromni mreži strežnikov, ker uporabniki uporabljajo različne strežnike ali si postavijo svoje. Mreža strežnikov XMPP je razpršena in zelo razvejana, blokiranje vseh strežnikov XMPP je nemogoče.

6.1.2 Odpiranje uporabniškega računa in dodajanje kontaktov

Komercialni ponudniki za odpiranje računa večinoma uporabljajo telefonsko številko uporabnika in za odkrivanje kontaktov njegov telefonski imenik. Uporabnik ima takoj na voljo veliko mrežo kontaktov, ki se samo izberejo s seznama - ročno vpisovanje ni potrebno. Nekateri ponudniki ne ponujajo drugačnih možnosti dodajanja kontaktov kot le preko telefonske številke. Na strežnike ponudnika se shranijo tudi

telefonske številke kontaktov, ki niso prijavljeni v sistem (za kasnejše odkrivanje). Uporabniška izkušnja je sicer zelo dobra, vendar se pojavi vprašanje zasebnosti.

Odpiranje računa XMPP je za nevešče uporabnike bolj zapleteno. Uporabnik si mora sam izbrati (ali postaviti) strežnik in odjemalca XMPP in registrirati uporabniški račun, za katerega si izbere lokalni del naslova JID. Vsak kontakt je potrebno dodati posebej, kar ni tako praktično in hitro, vendar je boljše z vidika zasebnosti in ponuja več svobode pri ustvarjanju seznama kontaktov.

6.1.3 Odprtost programske kode

Programska koda odjemalcev in strežnikov komercialnih ponudnikov ni na vpogled. Pravilnim implementacijam varnostnih protokolov lahko verjamemo samo na osnovi zagotovil.

Tehnologija XMPP je odprtokodna, prav tako je na javni vpogled dana koda odjemalcev in strežnikov. Kodo lahko pregledujejo neodvisni programerji, varnostni strokovnjaki itd., ki odkrivajo varnostne luknje in napake v delovanju.

6.2 Primerjava uporabniške izkušnje

V primeru komercialnih ponudnikov stvari delujejo samodejno, brez nastavitev in/ali nameščanja dodatnih vtičnikov, knjižnic, verzij programa,... Uporabniški vmesniki ponudnika so na različnih sistemih podobni in intuitivni.

Glavni razlog za nepopularnost TS z uporabo tehnologije XMPP tako po našem mnenju tiči v uporabniški izkušnji.

6.2.1 Bodoče izboljšave

V nadaljevanju poudarjamo nekaj zahtev in opisujemo možne rešitve, o katerih v skupnosti razvijalcev odjemalcev in strežnikov XMPP že potekajo pogovori.

1. Ustvarjanje uporabniškega računa.
   Ustvarjanje računa na javnem strežniku poteka preko internetne strani strežnika.

Poleg tega je v nekaterih odjemalcih potrebno vpisati naslov JID v celoti, v nekaterih pa posebej lokalni del naslova (npr. `ana`) in domenski del naslova (npr. `xmppchat.eu`). To lahko pripomore k dodatni zmedi neveščih uporabnikov.

Bolj transparentna bi bila možnost ustvarjanja računa v odjemalcih. Problem je že rešen v odjemalcu Conversations tako, da uporabnik vpiše samo svoj (lokalni) del naslova, domena strežnika (v lasti avtorja) pa se doda samodejno. Taka rešitev je transparentna in omogoča hiter začetek.

2. Velika izbira javnih strežnikov XMPP.
   Prevelika izbira je včasih slabša kot odsotnost izbire. Vsaj za nevešče uporabnike, ki jih obširna ponudba različnih strežnikov s čudnimi domenami lahko zmede. Po drugi strani so informacije o podprtih razširitvah strežnika največkrat zelo skope, kar lahko privede do nezadovoljstva ob uporabi z bolj tehnološko naprednim odjemalcem.

   Rešitev bi lahko bila v dogovoru razvijalcev odjemalcev in lastnikov javnih strežnikov. Izbrane javne strežnike bi se lahko vključilo v odjemalce kot možne izbire. Podprtost razširitvam bi seveda morala biti enaka s strani odjemalcev in strežnikov - tako ne bi prihajalo do nepričakovanih zapletov pri delovanju. Uporabniki bi se tudi lažje odločili kateri strežnik izbrati in bi mu bolj zaupali. Uporabniško ime bi se ustvarilo v odjemalcu.

3. Velika izbira odjemalcev XMPP.
   Uporabnik je prepuščen samostojnemu raziskovanju in izbiri odjemalca, ki mu ustreza glede na operacijski sistem, uporabniški vmesnik in podprte razširitve. Kot v primeru strežnikov tudi tu informacij o podprtih razširitvah največkrat ni. Večina odjemalcev tudi (še) ne podpira razširitev, ki so v današnjem času skorajda nujne.

   Trenutno imamo poplavo najrazličnejših odjemalcev za vse sisteme, zelo redki pa svojo stvar opravljajo zanesljivo in času primerno. Potrebna bi bila boljša komunikacija in sodelovanje med različnimi razvijalci na različnih platformah. Napredek v tej smeri so pokazali razvijalci odjemalcev Conversations, ChatSecure in Gajim.

4. Nastavitve šifriranja v odjemalcih
   Odjemalci ponujajo različne vrste šifriranja od konca do konca, kar lahko uporabnika zelo zmede. Največkrat je potrebno nameščanje vtičnikov in v večini primerov še naknadni ročni vklop šifriranja. Večji komercialni ponudniki so v svoje sisteme že vgradili varnostne protokole, ki so skoraj (avtentikacija) popolnoma transparentni za uporabnika.

   Vsi odjemalci bi morali čimprej preiti na implementacijo oz. vgradnjo varnostnega protokola kot privzeto vključeno možnost, brez potrebe po dodatnih vtičnikih ali knjižicah.

6.3 Zaključek

Glede na zahteve, ki smo jih podali v poglavju 2.4, smo ugotovili, da odprtokodna rešitev s protokolom XMPP zadovoljivo nadomešča komercialno. Trenutno največja pomanjkljivost je podprtost sistema iOS, za katerega trenutno še ne obstaja odjemalec, ki bi (v povezavi z drugimi operacijskimi sistemi) omogočal kaj več, kot le osnovni namen TS. Na ostalih sistemih pa so odprtokodne rešitve s tehničnega in varnostnega vidika ravno tako dobre kot komercialne oz. jih v določenih primerih prekašajo:

1. Kombinacija mreže *„Tor“* in šifriranja od konca do konca s protokolom OMEMO izpolnjuje vse tri varnostne sklope zahtev, podanih v poglavju 4.

2. Odjemalec Jitsi zagotavlja zanesljiv in od konca do konca šifriran prenos videoklica, kar zaenkrat ne zagotavlja noben komercialni ponudnik.

3. Možna je postavitev lastnega strežnika z arhivom sporočil, ki je tako v rokah uporabnika in ne na strežnikih komercialnih ponudnikov.

4. Izbira odjemalcev je velika, uporabnik lahko izbere tistega, ki mu najbolj ustreza.

5. Ni omejitve velikosti in tipa datotek, ki se lahko prenašajo.

Z vidika uporabniške izkušnje je potrebno še precej dela in izboljšav, da bo sistem TS s protokolom XMPP lahko konkuriral komercialnim sistemom. Vendar gre razvoj v pravo smer - odjemalec Conversations je lep primer, kako se tudi uporabniška izkušnja lahko približa komercialni.

Razvoj odjemalcev XMPP bi bil zagotovo hitrejši, če bi obstajala večja baza uporabnikov tehnologije XMPP. Tu se pojavi problem prehoda uporabnikov, ki imajo svoje socialne mreže že trdno vzpostavljene pri komercialnih ponudnikih. Mogoče pa se bo z nedavno združitvijo baz podatkov sistemov Facebook in WhatsApp [23], spremenilo tudi to?

Literatura

[1] Pidgin. IM all your friends in one place. `http://pidgin.im/`. [Dostopano: 18. 6. 2016].

[2] WikiLeaks. `https://wikileaks.org/the-spyfiles.html`. [Dostopano: 18. 6. 2016].

[3] XMPP Standards Foundation (XSF). `http://xmpp.org/about/xmpp-standards-foundation`. [Dostopano: 4. 6. 2016].

[4] XMPP. The most secure messaging standard. `http://xmpp.org/`. [Dostopano: 4. 6. 2016].

[5] C. Ballinger. *ChatSecure Encrypted Messenger for iOS and Android*. `https://chatsecure.org`. [Dostopano: 18. 6. 2016].

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. `https://tools.ietf.org/html/rfc2616`. [Dostopano: 3. 9. 2016].

[7] I. Bagueros, G. Boyce, K. Brade, A. Breault. *Tor, Anonymity Online*. `https://www.torproject.org/`. [Dostopano: 4. 6. 2016].

[8] F. Schmaus, D. Schürmann, V. Breitmoser. *XEP-0374: OpenPGP for XMPP Instant Messaging*. `http://xmpp.org/extensions/xep-0374.html`. [Dostopano: 3. 9. 2016].

[9] N. Borisov, I. Goldberg, E. Brewer. *Off-the-Record Communication, or, Why Not To Use PGP*. `https://otr.cypherpunks.ca/otr-wpes.pdf`. [Dostopano: 3. 9. 2016].

[10] N. Borisov, I. Goldberg, E. Brewer. *Off-the-Record Messaging Protocol version 3.* https://otr.cypherpunks.ca/Protocol-v3-4.1.1.html. [Dostopano: 3. 9. 2016].

[11] H. Krawczyk, M. Bellare, R. Canetti. *Keyed-Hashing for Message Authentication.* http://www.ietf.org/rfc/rfc2104.txt. [Dostopano: 3. 9. 2016].

[12] S. Ludwig, P. Saint-Andre, S. Egan, R. McQueen, D. Cionoiu. *XEP-0167: Jingle RTP Sessions.* http://xmpp.org/extensions/xep-0167.html. [Dostopano: 3. 9. 2016].

[13] J. Beda, P. Saint-Andre, S. Ludwig, J. Hildebrand, S. Egan. *XEP-0177: Jingle Raw UDP Transport Method.* http://xmpp.org/extensions/xep-0177.html. [Dostopano: 3. 9. 2016].

[14] H. Krawczyk, P. Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF).* https://www.rfc-editor.org/rfc/rfc5869.txt. [Dostopano: 3. 9. 2016].

[15] Facebook. *Facebook for developers.* https://developers.facebook.com/docs/chat/. [Dostopano: 18. 6. 2016].

[16] C. Alexander, I. Goldberg. *Improved User Authentication in Off-The-Record Messaging.* https://cypherpunks.ca/~iang/pubs/impauth.pdf. [Dostopano: 3. 9. 2016].

[17] Google. *Google Talk for Developers.* https://developers.google.com/talk/open_communications. [Dostopano: 18. 6. 2016].

[18] D. Gultsch. *Conversations. The very last word in instant messaging.* https://conversations.im. [Dostopano: 18. 6. 2016].

[19] D. Gultsch. *XEP-0363: HTTP File Upload.* http://xmpp.org/extensions/xep-0363.html. [Dostopano: 3. 9. 2016].

[20] P. Saint-Andre, D. Meyer, J. Karneges, M. Lundblad, T. Markmann, K. Hartke. *XEP-0260: Jingle SOCKS5 Bytestreams Transport Method.* http://xmpp.org/extensions/xep-0260.html. [Dostopano: 3. 9. 2016].

[21] W. Diffie, M. Hellman. *New Directions in Cryptography.* `http://www-ee.stanford.edu/~Ehellman/publications/24.pdf`. [Dostopano: 3. 9. 2016].

[22] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, J. Hildebrand. *XEP-0166: Jingle.* `http://xmpp.org/extensions/xep-0166.html`. [Dostopano: 3. 9. 2016].

[23] WhatsApp Inc. *Looking ahead for WhatsApp.* `https://blog.whatsapp.com/10000627/Looking-ahead-for-WhatsApp`. [Dostopano: 3. 9. 2016].

[24] WhatsApp Inc. *WhatsApp Security.* `https://www.whatsapp.com/security/`. [Dostopano: 18. 6. 2016].

[25] E. Ivov. *Jitsi - Open Source Video Calls and Chat.* `https://jitsi.org/Main/Features`. [Dostopano: 18. 6. 2016].

[26] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications.* `https://tools.ietf.org/html/rfc3550`. [Dostopano: 3. 9. 2016].

[27] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones. *SOCKS Protocol Version 5.* `https://tools.ietf.org/html/rfc1928`. [Dostopano: 3. 9. 2016].

[28] J. Jonsson, B. Kaliski. *Public-Key Cryptography Standards.* `https://tools.ietf.org/html/rfc3447`. [Dostopano: 3. 9. 2016].

[29] C. Kalt. *Internet Relay Chat: Server Protocol.* `https://tools.ietf.org/html/rfc2813`. [Dostopano: 3. 9. 2016].

[30] D. Smith, M. Miller, P. Saint-Andre, J. Karneges. *XEP-0065: SOCKS5 Bytestreams.* `http://xmpp.org/extensions/xep-0065.html`. [Dostopano: 3. 9. 2016].

[31] H. Krawczyk. *The SIGMA Family of Key-Exchange Protocols.* `http://webee.technion.ac.il/~hugo/sigma.html`. [Dostopano: 3. 9. 2016].

[32] D. Fomin, Y. Leboulanger. *Gajim!* `https://gajim.org/`. [Dostopano: 18. 6. 2016].

[33] M. Marlinspike. *Open Whisper Systems partners with Google on end-to-end encryption for Allo* . https://whispersystems.org/blog/allo/. [Dostopano: 18. 6. 2016].

[34] M. Marlinspike. *Simplifying OTR deniability*. https://whispersystems.org/blog/simplifying-otr-deniability/. [Dostopano: 3. 9. 2016].

[35] R. McMillan. *The Messenger: An Interview with Jabber's Creator, Jeremie Miller*. http://www.linux-mag.com/id/902/. [Dostopano: 18. 6. 2016].

[36] J. Beda, S. Ludwig, P. Saint-Andre, J. Hildebrand, S. Egan, R. McQueen. *XEP-0176: Jingle ICE-UDP Transport Method*. http://xmpp.org/extensions/xep-0176.html. [Dostopano: 3. 9. 2016].

[37] J. Hildebrand, M. Miller. *XEP-0280: Message Carbons*. http://xmpp.org/extensions/xep-0280.html. [Dostopano: 3. 9. 2016].

[38] T. Muldowney. *XEP-0027: Current Jabber OpenPGP Usage*. http://xmpp.org/extensions/xep-0027.html. [Dostopano: 3. 9. 2016].

[39] T. Muldowney. *XEP-0096: SI File Transfer*. http://xmpp.org/extensions/xep-0096.html. [Dostopano: 3. 9. 2016].

[40] Ed. P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Core*. https://www.ietf.org/rfc/rfc3920.txt. [Dostopano: 3. 9. 2016].

[41] Ed. P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. https://tools.ietf.org/html/rfc3921. [Dostopano: 3. 9. 2016].

[42] A. Alerigi Jr., G. Parra-Bernal. *Brazil judge orders WhatsApp blocked, affecting 100 million users*. http://www.reuters.com/article/us-facebook-brazil-whatsapp-idUSKCN0XT1KB. [Dostopano: 18. 6. 2016].

[43] C. Paar, J. Pelzl. *Online Cryptography Course*. http://wiki.crypto.rub.de/Buch/movies.php. [Dostopano: 4. 6. 2016].

[44] D. Pocock. *Real-Time Communications Quick Start Guide*. http://rtcquickstart.org/guide/multi/index.html. [Dostopano: 4. 6. 2016].

[45] J. Postel. *User Datagram Protocol*. `https://www.ietf.org/rfc/rfc768.txt`. [Dostopano: 3. 9. 2016].

[46] Darpa Internet Program. *Transmission Control Protocol*. `https://tools.ietf.org/html/rfc793`. [Dostopano: 3. 9. 2016].

[47] T. Dierks, E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. `https://tools.ietf.org/html/rfc5246`. [Dostopano: 3. 9. 2016].

[48] J. Rosenberg. *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*. `https://tools.ietf.org/html/rfc5245`. [Dostopano: 3. 9. 2016].

[49] J. Rosenberg. *SIMPLE Made Simple: An Overview*. https://tools.ietf.org/html/rfc6914. [Dostopano: 3. 9. 2016].

[50] J. Rosenberg. *SIP: Session Initiation Protocol*. `https://datatracker.ietf.org/doc/rfc3261/`. [Dostopano: 18. 6. 2016].

[51] J. Hildebrand, P. Millard, R. Eatmon, P. Saint-Andre. *XEP-0030: Service Discovery*. `http://xmpp.org/extensions/xep-0030.html`. [Dostopano: 3. 9. 2016].

[52] J. Karneges, P. Saint-Andre. *XEP-0047: In-Band Bytestreams*. `http://xmpp.org/extensions/xep-0047.html`. [Dostopano: 3. 9. 2016].

[53] P. Saint-Andre. *A Public Statement Regarding Ubiquitous Encryption on the XMPP Network* . `https://github.com/stpeter/manifesto/blob/master/manifesto.txt`. [Dostopano: 4. 6. 2016].

[54] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Address Format*. `https://tools.ietf.org/html/rfc7622`. [Dostopano: 3. 9. 2016].

[55] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Core*. `https://tools.ietf.org/html/rfc6120`. [Dostopano: 3. 9. 2016].

[56] P. Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. `https://tools.ietf.org/html/rfc6121`. [Dostopano: 3. 9. 2016].

[57] P. Saint-Andre. *XEP-0001: XMPP Extension Protocols*. http://xmpp.org/extensions/xep-0001.html. [Dostopano: 3. 9. 2016].

[58] P. Saint-Andre. *XEP-0045: Multi-User Chat*. http://xmpp.org/extensions/xep-0045.html. [Dostopano: 3. 9. 2016].

[59] P. Saint-Andre. *XEP-0053: XMPP Registrar Function*. http://xmpp.org/extensions/xep-0053.html. [Dostopano: 3. 9. 2016].

[60] P. Saint-Andre. *XEP-0205: Best Practices to Discourage Denial of Service Attacks*. http://xmpp.org/extensions/xep-0205.html. [Dostopano: 3. 9. 2016].

[61] P. Saint-Andre. *XEP-0261: Jingle In-Band Bytestreams Transport Method*. http://xmpp.org/extensions/xep-0261.html. [Dostopano: 3. 9. 2016].

[62] T. Alkemade, E. Richie, F. Dowsett, A. Godoroja, M. Needham, E. Schoenberg. *Adium is a free instant messaging application for Mac OS X*. https://adium.im/. [Dostopano: 18. 6. 2016].

[63] Skype and/or Microsoft . *Does Skype use encryption?* https://support.skype.com/en/faq/FA31/does-skype-use-encryption?q=security. [Dostopano: 18. 6. 2016].

[64] K. Smith. *Marketing: 96 Amazing Social Media Statistics and Facts for 2016*. https://www.brandwatch.com/2016/03/96-amazing-social-media-statistics-and-facts-for-2016/. [Dostopano: 18. 6. 2016].

[65] M. Wild, K. Smith. *XEP-0313: Message Archive Management*. http://xmpp.org/extensions/xep-0313.html. [Dostopano: 3. 9. 2016].

[66] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, M. Smith. *SoK: Secure Messaging*. http://cacr.uwaterloo.ca/techreports/2015/cacr2015-02.pdf. [Dostopano: 3. 9. 2016].

[67] P. Saint-Andre, L. Stout. *XEP-0234: Jingle File Transfer*. http://xmpp.org/extensions/xep-0234.html. [Dostopano: 3. 9. 2016].

[68] A. Straub. *XEP-xxxx: OMEMO Encryption.* `https://xmpp.org/extensions/inbox/omemo.html`. [Dostopano: 3. 9. 2016].

[69] Open Whisper Systems. *Advanced cryptographic ratcheting.* `https://whispersystems.org/blog/advanced-ratcheting/`. [Dostopano: 3. 9. 2016].

[70] Open Whisper Systems. *Signal Protocol library for Java/Android.* `https://github.com/WhisperSystems/libsignal-protocol-java`. [Dostopano: 3. 9. 2016].

[71] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer. *OpenPGP Message Format.* `https://tools.ietf.org/html/rfc4880`. [Dostopano: 3. 9. 2016].

[72] P. Saint-Andre, K. Smith, R. Troncon. *XMPP: The Definitive Guide.* Sebastopol, O'Reilly Media, Inc., 2009.

[73] S. Verschoor. *OMEMO: Cryptographic Analysis Report.* `https://conversations.im/omemo/audit.pdf`. [Dostopano: 3. 9. 2016].

[74] Viber Media S.à r.l. *Viber Encryption Overview.* `http://www.viber.com/en/security-overview`. [Dostopano: 18. 6. 2016].

[75] Wikipedia. *List of virtual communities with more than 100 million active users.* `https://en.wikipedia.org/wiki/List_of_virtual_communities_with_more_than_100_million_active_users`. [Dostopano: 18. 6. 2016].

[76] M. Wild. *Prosody. A study in simplicity.* `http://prosody.im/`. [Dostopano: 18. 6. 2016].

[77] C. Newman, A. Menon-Sen, A. Melnikov, N. Williams. *Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms.* `https://tools.ietf.org/html/rfc5802`. [Dostopano: 3. 9. 2016].

[78] A. Melnikov, K. Zeilenga. *Simple Authentication and Security Layer (SASL).* `https://tools.ietf.org/html/rfc4422`. [Dostopano: 3. 9. 2016].