SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1727

Razmjena ključeva za protokol SRTP temeljena na protokolu DTLS

Martin Vladić

Zagreb, 2008.

Zahvaljujem mr. sc. Stjepanu Grošu
na stručnom vodstvu.

Sažetak

U ovom radu opisuje se problem uspostave ključeva za protokol SRTP. Dan je uvod u područje IP multimedijalne komunikacije. Definirani su zahtjevi na konačno rješenje i navedeni su predloženi načini uspostave ključeva. Detaljno je opisan protokol DTLS-SRTP. U praktičnom dijelu ostvarena je biblioteka za protokol DTLS-SRTP i integrirana u Ekiga programski telefon.

Abstract

This thesis describes the problem of establishing keys for SRTP protocol. Introduction to the field of IP multimedia communication is given. Requirements for final solution are defined and recommended methods for establishing keys are evaluated. In practical part, library for DTLS-SRTP protocol is implemented and integrated inside EKIGA softphone.

Sadržaj

1. Uvod

RTP protokol je temeljni protokol za sustave koji isporučuju audio i video podatke u realnom vremenu preko IP mreža, kao što je Internet. Primjeri takvih sustava su video strujanje (npr. IP televizija) i IP telefonija. Kod ovog potonjeg, sasvim je razumljiv zahtjev da je glasovna komunikacija između dva sudionika telefonskog poziva zaštićena, primjerice od prisluškivanja. RTP protokol u svom osnovnom profilu ne štiti, tj. kriptira promet; stoga svatko tko ima mogućnost prisluškivanja IP prometa na nekom od poslužitelja koji se nalazi na putu između dva sudionika, može bez problema prisluškivati razgovor koristeći neki od javno dostupnih programa (npr. VoIPong i Wireshark). Stoga je osmišljen i standardiziran sigurnosni profil RTP protokola koji se zove SRTP.

SRTP omogućuje za RTP protokol neke sigurnosne fukcije kao što su enkripcija podataka, autentifikacija i integritet poruka te zaštita od ponavljanja. Kao što je poznato, svaki mehanizam za enkripciju ili autentifikaciju zahtjeva korištenje ključeva, bilo simetričnih ili asimetričnih, koji su poznati svakoj od strana koje razmjenjuju povjerljive informacije. SRTP sve svoje ključeve izvodi iz jednog glavnog ključa na kriptografski siguran način, ali ne definira postupak za razmjenu tog glavnog ključa. Upravo razmjena tog glavnog ključa i još nekih sigurnosnih parametara koji čine SRTP-ov kriptografski kontekst je tema ovog rada.

U ovom radu se u drugom poglavlju opisuju osnovi pojmovi i protokoli (SIP, SDP i RTP) bitni za razumjevanje IP zasnovane multimedijalne komunikacije. U trećem poglavlju obrađeno je područje razmjene ključeva za protokol SRTP, objašnjeni su osnovni pojmovi bitni za razumjevanje zahtjeva na metode za razmjenu ključeva, pobrojani su zahtjevi i navedene sve predložene metode. U četvrtom poglavlju detaljno se opisuje DTLS-SRTP metoda za razmjenu SRTP ključeva. Ostatak rada posvećen je praktičnom dijelu. U petom poglavlju obrađuje se DSRTP biblioteka za DTLS-SRTP protokol: njeno sučelje za aplikacije, dizajn i impementacija. U šestom poglavlju obrađuje se integracija DSRTP biblioteke unutar Ekige i OpalVoip biblioteke. Nakon toga slijedi zaključak u sedmom poglavlju, popis korištene literature u osmom poglavlju te dodatak u kojem je dan detaljan opis svih funkcija koje čine sučelje DSRTP biblioteke.

2. IP multimedijalna komunikacija

U ovom poglavlju opisani su oni dijelovi područja IP multimedijalne komunikacije koji su nužni za razumijevanje ostatka rada. Dan je pregled slijedećih protokola:

- Protokol SIP za uspostavu multimedijalne sjednice,
- Protokol SDP za opis sjednice, tj. podataka koji se razmjenjuju u takvoj sjednici,
- Protokol RTP za prijenos samih multimedijalnih podataka,
- Sigurnosni profil SRTP za RTP protokol.

2.1. SIP

SIP (engl. Session Initiation Protocol) [4] je protokol koji pripada aplikacijskom sloju, a služi za uspostavu, modificiranje i poništavanje multimedijalne sjednice između dvaju ili više sudionika. Tipičan primjer takve sjednice je telefonski poziv preko Interneta.

SIP obavlja sljedeće funkcije:

- Određivanje lokacije korisnika kojeg se poziva da sudjeluje u sjednici.
- Određivanje spremnosti korisnika da sudjeluje u sjednici.
- Određivanje formata zapisa sadržaja (audio i video) koji će se razmjenjivati u sjednici.
- Uspostava sjednice: “zvonjenje”, određivanje parametara sjednice na strani pozivatelja i na strani pozivanog.
- Upravljanje sjednicom, uključujući: transfer, poništavanje sjednice, modifikacija parametara sjednice te pozivanje usluge.

SIP sustav se sastoji od sljedećih entiteta: *krajnje točke* (engl. end points), *posredničkog poslužitelja* (engl. proxy server), *poslužitelja za preusmjeravanje* (engl. redirect server), *registracijskog poslužitelja* (engl. registrar) i *usluge za određivanje položaja* (engl. location service).

SIP identifikacija korisnika je zasnovana na specijalnom tipu *jedinstvenog identifikatora resursa* (engl. Uniform Resource Identifikator, URI) koji se naziva SIP URI. On ima oblik sličan adresi elektroničke pošte i sastoji se od imena korisnika i imena računala, npr. “sip:alice@atlanta.com”.

Krajnja točka ili *korisnički agent* (engl. user agent, UA) je entitet koji predstavlja telefonski uređaj ili programski telefon. Korisnički agent se dijeli na *klijenta korisničkog agenta* (engl. user agent client, UAC) koji kreira zahtjeve i na *poslužitelja korisničkog agenta* (engl. user agent server, UAS) koji generira odgovore na zahtjeve. Zahtjev za uspostavu poziva započinje u izvorišoj krajnjoj točki i preko određenog broja posredničkih poslužitelja završava u odredišnoj krajnjoj točki.

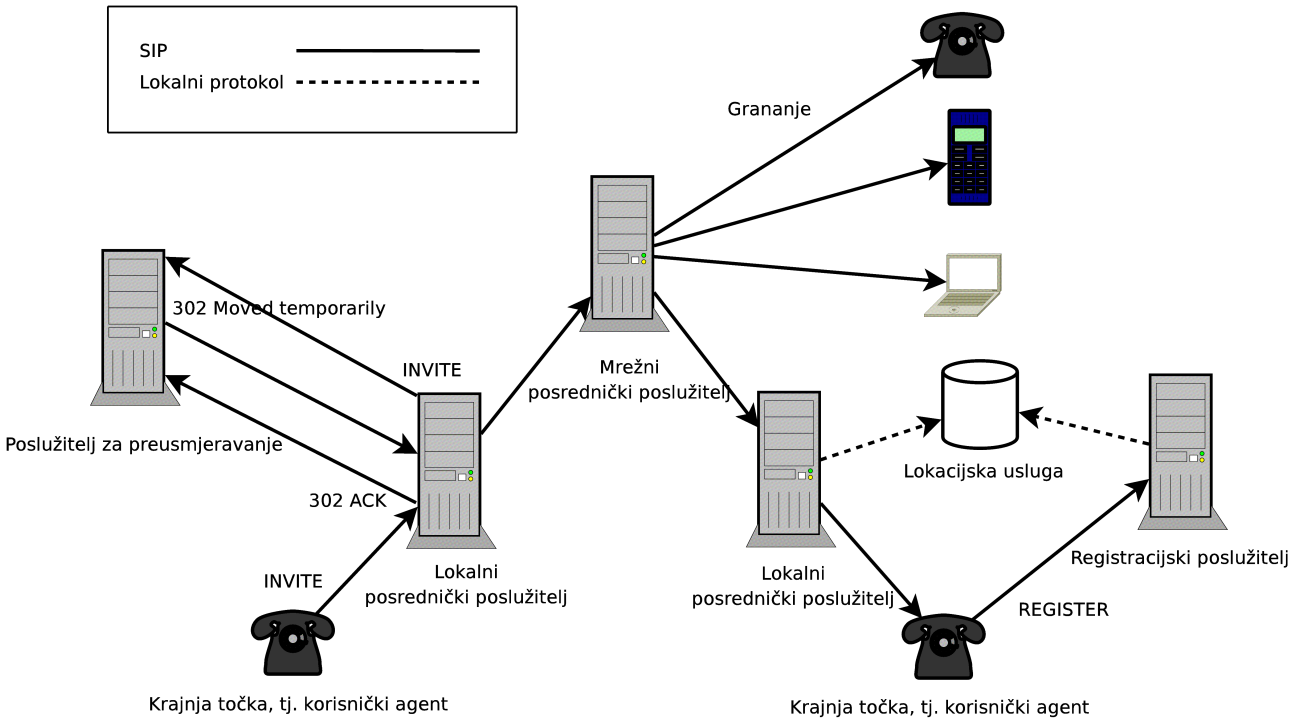
Posrednički poslužitelj služi za usmjeravanje poziva bliže trenutnom položaju odredišne krajnje točke. On obavlja autentifikaciju korisnika i autorizaciju korisnika na određene usluge (npr. da li je korisniku dopušteno obavljati pozive).

Poslužitelj za preusmjeravanje, umjesto da proslijedi klijentov zahtjev duž staze, generira neki od 3xx odgovora na zatjev, te na taj način predaje zamjenski skup URI-ja.

SIP također podržava specijalan tip posredničkog poslužitelja koji primljeni zahtjev proslijedi prema više odredišta. Takav poslužitelj se naziva *poslužitelj za grananje* (engl. forking server).

SIP korisnik nije vezan isključivo za jedan uređaj već može koristiti više uređaja istovremeno. Korisnik ima jednu javnu adresu koja se naziva AOR (engl. Adress of Record). AOR je SIP URI koji pokazuje na računalo s podržanom SIP lokacijskom uslugom koja mapira AOR URI u URI uređaja gdje bi korisnik mogao biti dostupan. Korisnik registrira uređaj na lokacijskoj usluzi koristeći usluge registracijskog poslužitelja. Registracijski poslužitelj prima REGISTER zahtjeve i primljenu informaciju sprema na poslužitelju koji sadrži lokacijsku uslugu. Lokacijska usluga se koristi od strane posredničkog poslužitelja i poslužitelja za preusmjeravanje kako bi se došlo do informacije o mogućim lokacijama korisnika.

Na slici 2.1 prikazan je smještaj i interakcija različitih entiteta u SIP sustavu. Važno je napomenuti kako je ova podjela na entitete isključivo logička, a ne fizička. To znači da neki poslužitelj može obavljati funkcije primjerice posredničkog poslužitelja i poslužitelja za preusmjeravanje.



Slika 2.1: SIP sustav

SIP je zasnovan na transakcijskom modelu zahtjev/odgovor koji nalikuje onome u HTTP protokolu. Svaka transakcija se sastoji od zahtjeva koji traži da se izvrši određena operacija na poslužitelju i najmanje jednog odgovora na taj zahtjev.

U protokolu SIP postoji šest tipova zahtjeva koji se razlikuju po vrsti operacije, tj. metode kako se naziva u SIP specifikaciji:

1. *REGISTER.* Koristi se za registriranje lokacije AOR-a, a implementirana je na registracijskom poslužitelju.
2. *INVITE. K*oristi se za pozivanje korisnika da sudjeluje u pozivu, tj. sjednici.
3. *ACK*. Koristi se isključivo uz INVITE zahtjeve i s ovom metodom klijent potvrđuje da je primio konačni odgovor na INVITE zahtjev.
4. *CANCEL.* Koristi se za poništavanja zahtjeva prethodno poslanog od strane klijenta.
5. *BYE.* Koristi se za završetak tekućeg poziva.
6. *OPTIONS.* Koristi se za otkrivanje mogućnosti poslužitelja.
7. UPDATE. Koristi se za obnovu parametara sjednice.

Odgovor na zahtjev sadrži statusni kod i frazu koja opisuje trenutno stanje zahtjeva. Statusni kodovi su podjeljeni u šest generalnih kategorija:

1. *1xx: Privremeni odgovori*. Zahtjev je prihvaćen i nastavlja se s obradom zatjeva. Klijent može primiti više ovakvih odgovora prije nego što primi konačni odgovor - odgovori koji pripadaju ostalim kategorijama smatraju se konačnim odgovorima na zahtjev.
2. *2xx: Uspjeh.* Zahtjev je uspješno primljen, shvaćen i prihvaćen.
3. *3xx: preusmjeravanje.* Potrebno je poduzeti dodatnu akciju kako bi se zahtjev mogao izvršiti.
4. *4xx: Greška kod klijenta.* Zahtjev sadrži sintaksne pogreške ili se ne može izvršiti na poslužitelju.
5. *5xx: Greška na poslužitelju.* Poslužitelj ne može izvršiti sasvim ispravan zahtjev.
6. *6xx: Globalna greška.* Zahtjev se ne može izvršiti niti na jednom poslužitelju.

Primjer 1 prikazuje kako izgleda tipičan INVITE zahtjev s kojim pozivatelj inicira uspostavu sjednice, a primjer 2 prikazuje kako izgleda 200 OK odgovor na taj zahtjev s kojim pozivani odgovara potvrdno na zahtjev za uspostavu sjednice. U primjerima 1 i 2 linije koje ne stanu u jedan redak razlomljene su na više redaka. Kao oznaka da je linija razlomljena koristi se znak \ na kraju retka; u tom slučaju slijedeći je redak uvučen sa znakom TAB. Ti znakovi nisu dio poruke.

Kao potvrdu da je primio 200 OK odgovor, pozivatelj šalje ACK zahtjev na koji nije potrebno slati odgovor.

```
INVITE sip:192.168.1.118:15060 SIP/2.0
Date: Tue, 18 Mar 2008 18:32:47 GMT
CSeq: 1 INVITE
Via: SIP/2.0/UDP 192.168.1.104:25060;\
     branch=z9hG4bK840a0d60-87f3-dc11-8b69-000e353de31b;rport
User-Agent: Ekiga/2.9.0
From: "Alice" <sip:alice@192.168.1.104:25060;transport=udp>;\
     tag=4a6d0c60-87f3-dc11-8b69-000e353de31b
Call-ID: b4c09b5e-87f3-dc11-8b69-000e353de31b@mvladic-fujitsu
To: <sip:192.168.1.118:15060>
Contact: <sip:alice@192.168.1.104:25060;transport=udp>
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE,\
     INFO,PING,PUBLISH
Content-Type: application/sdp
Content-Length: 571
Max-Forwards: 70

(SDP izostavljen)
```

Primjer 1: SIP INVITE zahtjev s kojim se inicira uspostava sjednice

```
SIP/2.0 200 OK
CSeq: 1 INVITE
Via: SIP/2.0/UDP 192.168.1.104:25060;\
     branch=z9hG4bK840a0d60-87f3-dc11-8b69-000e353de31b;\
     rport=25060;received=192.168.1.104
User-Agent: Ekiga/2.9.0
From: "Alice" <sip:alice@192.168.1.104:25060;transport=udp>;\
     tag=4a6d0c60-87f3-dc11-8b69-000e353de31b
Call-ID: b4c09b5e-87f3-dc11-8b69-000e353de31b@mvladic-fujitsu
To: <sip:192.168.1.118:15060>;tag=34902350-87f3-dc11-9994-001b770d2582
Contact: <sip:bob@192.168.1.118:15060;transport=udp>
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE,\
     INFO,PING,PUBLISH
Content-Type: application/sdp
Content-Length: 248

(SDP izostavljen)
```

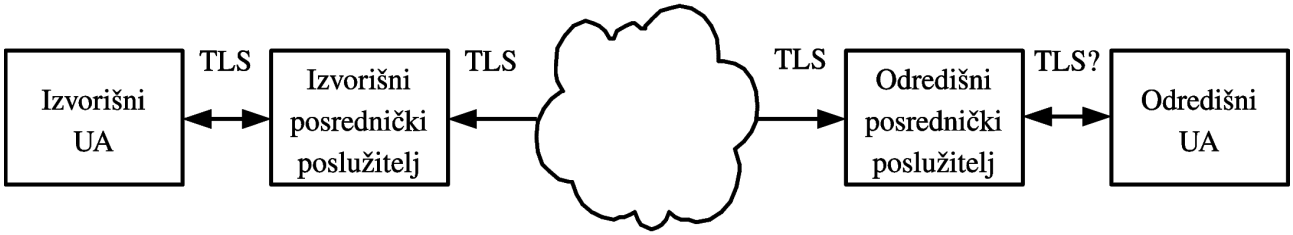
Primjer 2: SIP 200 OK potvrdan odgovor na INVITE zahtjev

SIP klijenti obično koriste pristup 5060 protokola UDP kako bi komunicirali sa SIP poslužiteljima.

2.1.1. SIPS

Ukoliko korisnik želi izvršiti sigurnu transakciju on može koristiti SIPS URI (npr. sips:alice@atlanta.com). Ovakva vrsta URI-ja govori korisničkom agentu i posredničkim poslužiteljima duž čitavog puta, da se SIP poruka mora razmjenjivati na siguran način *korak po korak* (engl. hop-by-hop). To znači da se mora uspostaviti TLS veze između svaka dva susjeda u prijenosu (slika 2.2) osim možda između odredišnog posredničkog poslužitelja i odredišnog

korisničkog agenta – što je prepušteno politici odredišne domene. Ako ovi uvjeti nisu ispunjeni u potpunosti, transakcija treba završiti s greškom.



Slika 2.2: Korištenje SIPS URI-ja zahtjeva upotrebu TLS protokola između susjednih entiteta

2.1.2. S/MIME

SIP poruke nose u svom tijelu MIME poruke; u većini slučajeva su to SDP poruke. Kao što je poznato, MIME norma sadrži mehanizme pod nazivom S/MIME za zaštitu integriteta i povjerljivosti MIME poruka. Ovakva zaštita radi *s kraja na kraj* (engl. end-to-end), što znači da se poruka zaštićuje na jednom kraju i kao takva prenosi na drugi kraj komunikacijskog lanca, bez mogućnosti promatranja ili mijenjanja od strane poslužitelja koji se nalaze na putu. Ovo može predstavljati problem određenim poslužiteljima koji svoj rad temelje na promatranju i mijenjanju sadržaja koji se prenose u tijelu SIP poruka. Drugi problem sa S/MIME zaštitom je što zahtjeva certifikate u krajnjim točkama, tj. za krajnje korisnike.

2.2. SDP

SIP je signalni protokol i služi za uspostavu i poništavanje multimedijalne sjednice. Razmjena podataka (audio i video) u realnom vremenu unutar takve sjednice obavlja se npr. pomoću RTP protokola. Ono što nedostaje u čitavoj priči jest poveznica između signalnog protokola i protokola za razmjenu podataka, tj. na koji način se protokol SIP definira buduća multimedijalna komunikacija unutar sjednice. U primjerima 1 i 2 izostavljeno je tijelo INVITE zahtjeva i 200 OK odgovora na taj zahtjev. Upravo se unutar tog tijela nalaze informacije bitne za uspostavu multimedijalne sjednice između sudionika. Format zapisa tog tijela zadan je *protokolom za opis sjednice* (engl. Session Description Protokol, SDP) [5].

U SIP poruci, upotreba SDP-a u tijelu poruke naznačuje se u zaglavlju poruke pomoću `Content-Type` polja tako što se navede `application/sdp` vrijednost.

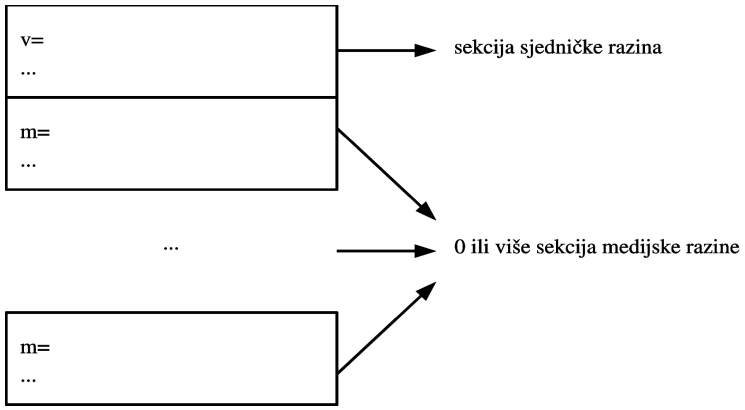
SDP opis sjednice zadan je u tekstualnom obliku i struktura tog opisa je strogo definirana. Čitav opis sastoji se od niza linija slijedećeg oblika:

```
<vrsta linije>=<vrijednost linije>
```

Vrsta linije označena je s malim slovom abecede, a vrijednost linije je tekst čiji format ovisi o vrsti linije.

SDP opis sjednice sastoji se od više sekcija (slika 2.3). Na početku se nalazi sekcija *sjedničke razine* (engl. session-level), nakon koje slijedi nula ili više sekcija *medijske razine* (engl. media-level). Sekcija sjedničke razine započinje s `v=` linijom i traje do prve sekcije medijske razine. Svaka

sekcija medijske razine započinje s `m=` linijom i traje do slijedeće sekcije medijske razine ili do kraja cijelokupnog SDP opisa. Svaka sekcija medijske razine opisuje jedan medij koji sudjeluje u sjednici. Vrijednosti koje se nalaze unutar sekcije sjedničke razine odnose se na sve medije opisane SDP-om. U slučaju kada se ista vrsta linije nalazi na sjedničkoj i na medijskoj razini, uzima se vrijednost iz medijske razine koja nadjačava vrijednost iz sjedničke razine.



Slika 2.3: Razine SDP opisa sjednice

Sekcija sjedničke razine sadrži slijedeće linije:

- `v=` verzija protokola
- `o=` pokretač i identifikator sjednice
- `s=` ime sjednice
- `i=*` informacija o sjednici
- `u=*` URI opisa
- `e=*` adresa elektroničke pošte
- `p=*` broj telefona
- `c=*` informacije o konekciji – može se izostaviti ukoliko je zastupljeno u svim sekcijama medijske razine
- `b=*` nula ili više linija s informacijama o širini propusnog opsega

Nula ili više vremenskih opisa (`t=` i `r=*` linije)

- `z=*` korekcija vremenske zone
- `k=*` enkripcijski ključ
- `a=*` nula ili više linija s atributima sjednice

Vremenski opis, koji se nalazi nula ili više puta unutar sekcije sjedničke razine, sadrži slijedeće linije:

- `t=` vrijeme u kojem je sjednica aktivna
- `r=*` nula ili više vremena ponavljanja

Sekcije medijske razine sadrže slijedeće linije:

- `m=` vrsta medija, komunikacijski pristup, načini kodiranja
- `i=*` naslov medija
- `c=*` informacije o konekciji – može se izostaviti ako je uključeno na sjedničkoj razini
- `b=*` nula ili više linija s informacijama o širina propusnog opsega
- `k=*` enkripcijski ključ
- `a=*` nula ili više linija s atributima medija

U gornjim sekcijama poredak linija je bitan, tako u sekciji sjedničke razine prva linija mora biti `v=`, nakon nje mora ići `o=` linija, itd. Linije označene sa znakom * su opcionalne linije i mogu se izostaviti.

Skup slova koje označavaju vrstu linija je namjerno mali i nije predviđeno da bude proširiv. Jedini način proširivanja SDP protokola je preko `a=` linija.

2.3. SDP model ponuda/odgovor

Protokol SIP koristi SDP *model ponuda/odgovor (*engl. Offer/Answer Model*)* [6] za dogovaranje parametara komunikacije unutar multimedijalne sjednice.

U ovom modelu jedan od sudionika multimedijalne sjednice, tzv. ponuditelj, generira ponudu, tj. SDP opis željene sjednice. Taj opis, za svaki medij koji čini ponudu, definira tip medija i skup kodiranja koji se žele koristiti te IP adresu i pristup na kojem se žele primati podaci. Ta ponuda se, zatim, šalje, pomoću protokol višeg nivoa (SIP), drugoj strani, tzv. odgovaratelju, koji generira SDP opis odgovora na ponudu. Taj opis, za svaki medij sadržan u ponudi, daje odgovor da li je prihvaćen ili ne, koji način kodiranja je odabran te na kojoj IP adresi i pristupu se žele primati podaci. Kasnije, jedna ili druga strana može poslati novu ponudu, koja se nadovezuje na ovu prethodnu ponudu (preko o= linije), u svrhu modificiranje uspostavljene sjednice.

Pokažimo na primjeru kako izgledaju SDP ponuda i SDP odgovor. SDP ponuda je prikazana u tablici 2.1, a šalje se unutar SIP INVITE zahtjeva (primjer 1). SDP odgovor je prikazan u tablici 2.2, a šalje se unutar SIP 200 OK odgovora (primjer 2). Alternativno, moguće je poslati prazan INVITE koji ne sadrži SDP ponudu; u tom slučaju SDP ponudu može poslati pozivani unutar 200 OK odgovora, a SDP odgovor se može nalaziti u ACK-u.

Linija	Opis
v=0	Verzija SDP protokola (0).
o=usera 1205953025 1205953025 IN IP4 192.168.1.104	Pokretač i identifikator sjednice (SIP).
s=Opal SIP Session	Ime sjednice.
c=IN IP4 192.168.1.104	Postavke konekcije, sastoji se od: vrsta mreže (IN for Internet), vrsta adrese (IP4 za IP verzija 4) i od same adrese (192.168.1.104).
t=0 0	Vrijeme kada sjednica započinje i završava – ne koristi se unutar protokola SIP.
m=audio 32504 RTP/AVP 8 0	Započinje sekcija s opisom zvukovnog (audio) medija. Ova linija sadrži pristup na kojem se sluša (32504), vrsta prijenosnog protokola (RTP/AVP) te brojeve zvukovnih profila (8 i 0). Ti profili se dodatno opisuju preko linija s rtpmap atributom.
a=rtpmap:8 PCMA/8000	Atributi RTP/AVP zvukovnog profila broj 8: način kodiranja (PCMA – PCM a-Law) i učestalost sempliranja (8000).
a=rtpmap:0 PCMU/8000	Atributi RTP/AVP zvukovnog profila broj 0: način kodiranja (PCMU – PCM μ-Law) i učestalost sempliranja (8000).
m=video 32506 RTP/AVP 122 31	Započinje sekcija s opisom video (video) medija. Ova linija sadrži pristup na kojem se sluša (32506), vrsta prijenosnog protokola (RTP/AVP) te brojeve video profila (122 i 31).
a=rtpmap:122 h264/90000	Atributi RTP/AVP video profila 122: način kodiranja (h264 – H.264) i učestalost uzorkovanja (90000).
a=rtpmap:31 h261/90000	Atributi RTP/AVP video profila 31: način kodiranja (h261 – H.261) i učestalost uzorkovanja (90000).

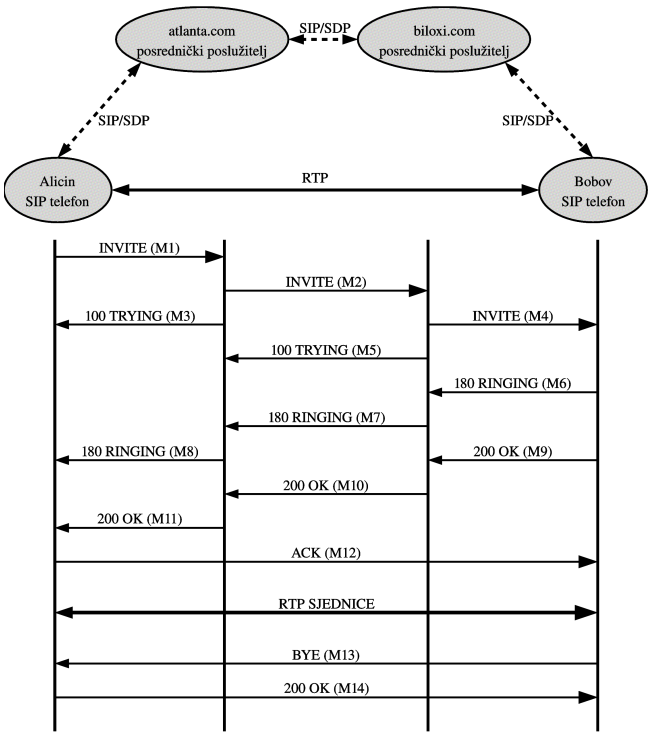
Tablica 2.1: Primjer SDP ponude koja se šalje unutar SIP INVITE zahtjeva

Linija	Opis
`v=0`	Verzija SDP protokola (`0`).
`o=userb 1205574890 1205574890 IN IP4 192.168.1.118`	Pokretač i identifikator sjednice.
`s=Opal SIP Session`	Ime sjednice.
`c=IN IP4 192.168.1.118`	Postavke konekcije, sastoji se od: vrsta mreže (`IN` for Internet), vrsta adrese (`IP4` za IP verzija 4) i od same adrese (`192.168.1.118`).
`t=0 0`	Vrijeme kada sjednica započinje i završava – ne koristi se unutar SIP-a.
`m=audio 31000 RTP/AVP 8`	Započinje sekcija s opisom zvukovnog (`audio`) medija. Budući da je odabran pristup koji je različit od 0 (`31000`), to znači da je zvukovna sjednica prihvaćena. Odabran je zvukovni profil `8`.
`a=rtpmap:8 PCMA/8000`	Ovaj rtpmap atribut navodi atribute RTP/AVP zvukovnog profila `8`, a to su način kodiranja (`PCMA` – PCM a-Law) i učestalost uzorkovanja (`8000`).
`m=video 0 RTP/AVP 122 31`	Započinje sekcija s opisom video (`video`) medija. Budući da je odabran pristup `0` to znači da je video sjednica odbijena.

Tablica 2.2: Primjer SDP odgovora koji se šalje unutar SIP 200 OK odgovora

2.4. SIP trapezoid

Na slici 2.4 prikazan je primjer uspostavu multimedijalne sjednice između dvije krajnje točke korištenjem protokola SIP. Ovakva konfiguracija se naziva *SIP trapezoid*. Alice koristi svoj SIP telefon kako bi pozvala Boba na njegov SIP telefon koristeći Internet. Također su prikazana i dva SIP posrednička poslužitelja koja djeluju u korist Alice i Boba za pomoć pri uspustavi sjednice.



Slika 2.4: SIP uspostave sjednice na primjeru SIP trapezoida

Ukoliko želi pozvati Boba, Alice mora znati njegov SIP URI. U našem primjeru to je sip:bob@biloxi.com, gdje je biloxi.com domena Bobovog pružatelja SIP usluge. Alicin SIP URI je sip:alice@atlanta.com.

INVITE zahtjev (poruka M1 na slici 2.4) za uspostavom sjednice koji šalje Alice može izgledati ovako:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP alicepc.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@anapc.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@anapc.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(SDP nije prikazan)
```

Prva linija ove tekstualne poruke sadrži ime metode (INVITE). Linije koje slijede su polja zaglavlja; ovaj primjer sadrži minimalan nužan skup takvih polja. Polja imaju sljedeće značenje:

- **Via** sadrži adresu (anapc.atlanta.com) na kojoj Alice treba primiti odgovor na zahtjev. Također sadrži i *branch* parametar koji identificira ovu transakciju.
- **To** sadrži *ime za prikaz* (engl. display name) (Bob) i SIP URI (sip:bob@biloxi.com) prema kome je zahtjev izvorno upućen.

- **From** sadrži ime za prikaz (Alice) i SIP URI (sip:alice@atlanta.com) koji označavaju pošiljaoca zahtjeva. Ovo polje također sadrži i parametar *tag* koji sadrži slučajan niz karaktera koji je nadodan na URI od strane Alicinog SIP telefona, a koristi se za identikacijske potrebe.
- **Call-ID** sadrži globalno jedinstven identifikator za ovaj poziv koji je generiran kombiniranjem slučajnog niza karaktera s imenom računala na kojem se nalazi SIP programski telefon ili s IP adresom tog računala. Kombinacija parametra *tag* polja To, parametra *tag* polja From i Call-ID polja u potpunosti definira SIP sjednicu između Alice i Boba (u SIP terminologiji sjednica se još naziva i *Dialog*).
- **CSeq** sadrži broj i ime metode. Ovaj broj se povećava za jedan kod svakog novog zahtjeva unutar iste sjednice, dakle radi se o tipičnom sekvencijskom broju.
- **Contact** sadrži SIP URI koji predstavlja direktnu stazu prema Alici. Via polje je adresa na koju treba stići odgovor na ovaj zahtjev, a Contact polje je adresa na koju trebaju biti upućeni budući zahtjevi unutar iste SIP sjednice.
- **Max-Forwards** sadrži maksimalan broj koraka (engl. hop) koje zahtjev može napraviti prije nego se odbaci.
- **Content-Type** sadrži vrstu MIME sadržaja koji se prenosi u poruci.
- **Content-Length** sadrži veličinu u oktetima MIME sadržaja koji se prenosi.

Budući da Alicin SIP telefon ne zna Bobovu lokaciju ili lokaciju SIP posredničkog poslužitelja u biloxi.com domeni, INVITE zahtjev se šalje prema posredničkom poslužitelju koji opslužuje Alicinu domenu atlanta.com. Adresa ovog poslužitelja može biti unešena kao konfiguracijski parametar u Alicinom SIP telefonu, ili može biti dohvaćena korištenjem npr. DHCP protokola.

U ovom primjeru, posrednički poslužitelj prima INVITE zahtjev i šalje 100 Trying odgovor natrag prema Alicinom telefonu. Taj odgovor označava da je zahtjev primljen i da posrednički poslužitelj radi na tome da INVITE usmjeri prema njegovom odredištu. Ddgovor sadrži isto To polje, From polje, Call-ID polje, CSeq polje i branch parametar u Via polju kao i originalan INVITE zahtjev tako da Alicin telefon može napraviti korelaciju između originalnog zahtjeva i ovog odgovora.

Posrednički poslužitelj atlanta.com locira biloxi.com posrednički poslužitelj, u većini slučajeva izvođenjem određene vrste DNS upita kako bi pronašao SIP poslužitelj koji opslužuje biloxi.com domenu. Kada je locirao biloxi.com posrednički poslužitelj, atlanta.com prosljeđuje INVITE zahtjev prema njemu. Prije prosljeđivanja zahtjeva, atlanta.com posrednički poslužitelj dodaje dodatno Via polje u zaglavlje INVITE poruke koje sadrži njegovu vlastitu adresu (INVITE zahtjev već sadrži Alicinu adresu u prvom Via polju).

Posrednički poslužitelj biloxi.com prima INVITE od atlanta.com poslužitelja i šalje odgovar 100 Trying prema atlanta.com poslužitelju kao potvrdu da je zahtjev primljen i da radi na daljnjem procesiranju zahtjeva. Poslužitelj biloxi.com konzultira lokacijski uslugu za biloxi.com domenu kako bi pronašao trenutnu IP adresu na koju je prijavljen Bob. Poslužitelj zatim dodaje novo Via

polje koje sadrži njegovu vlastitu adresu i prosljeđuje INVITE zahtjev prema Bobovom SIP telefonu.

Bobov telefon zvonjenjem upozorava Boba da je stigao dolazni poziv od Alice tako da Bob može odlučiti da li će odgovoriti na poziv. Bobov telefon naznačuje ovo sa 180 Ringing odgovorom koji se usmjeruje kroz dva posrednička poslužitelja sve do Alicinog telefona. Posrednički poslužitelji koriste Via polje koje se nalazi na vrhu kako bi odredili kome poslati odgovor i odstranjuju svoju adresu s vrha. Dakle, iako su različiti DNS upiti i upiti prema lokacijskom serveru bili potrebni za slanje inicijalnog INVITE zahtjeva od Alice prema Bobu, korištenjem Via polja, 180 Ringing odgovor se može poslati od Boba prema Alici bez određenih DNS upita i bez zahtjeva da povratne adrese budu spremljene na strani posredničkog poslužitelja. Kad Alicin telefon primi 180 Ringing on o tome informira Alicu npr. zvonjenjem ili prikazom poruke na ekranu Alicinog telefona.

U našem primjeru, Bob odluči odgovoriti na Alicin poziv tako što podigne slušalicu ili u slučaju programskog telefona tako što klikne na Odgovori tipku. U tom trenutku Bobov telefon pošalje 200 OK SIP odgovor zajedno sa SDP porukom koja je Bobov odgovor na SDP ponudu koju je poslala Alice u originalnom INVITE zahtjevu. Dakle, radi se o SDP modelu ponuda/odgovor koji je opisan u prethodnom poglavlju. Pogledajmo kako bi mogla izgledati 200 OK poruka u trenutku kad je šalje Bobov SIP telefon:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com;\
     branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;\
     branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP alicapc.atlanta.com;\
     branch=z9hG4bK776asdhds;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@anapc.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(SDP nije prikazan)
```

Prva linija sadrži statusni kod (200) i frazu koja opisuje razlog (OK) odgovora. Ostale linije sadrže polja zaglavlja. Polja Via, To, From, Call-ID i CSeq su iskopirana iz originalnog INVITE zahtjeva. Osim originalnog Via polja kojeg je dodao Alicin SIP telefon postoje i još dva dodatna Via polja koje su dodali atlanta.com i biloxi.com posrednički poslužitelji. Bobov SIP telefon je dodao tag parametar u To polje i dodao je Contact polje u kojem se nalazi URI preko kojeg se može Bobovom telefon direktno pristupiti (npr. ACK poruka se šalje direktno, a ne preko posredničkog poslužitelja).

Konačno, Alicin SIP telefon šalje ACK poruku (direktno, preskačući posredničke poslužitelje) prema Bobovom SIP telefonu kako bi potvrdio primitak 200 OK poruke. RTP sjednice (npr. audio i video) sada mogu početi između Alice i Boba.

U našem primjeru, poziv završava kada Bob zaklopi slušalicu i njegov SIP telefon pošalje BYE poruku (opet direktno, preskačući posredničke poslužitelje). Alice potvrđuje primitak BYE poruke odgovarajući s 200 OK porukom, koja terminira sjednicu i BYE transakciju.

2.5. RTP

RTP (engl. Real-time Transport Protocol) [7] je protokol za prijenos podataka u realnom vremenu, primjerice audio i video podataka. RTP je vrlo kompleksan protokol, pa ćemo ovdje pokušati objasniti samo neke pojmove koji su bitni za razumjevanje ostatka teksta.

Uz RTP protokol koji služi za prijenos podataka, postoji i protokol za kontrolu prijenosa koji se zove RTCP (engl. Real-time Control Protocol).

RTP sjednica se sastoji od grupe sudionika koji komunciraju koristeći RTP protokol. Jedna RTP sjednica je namjenjena za prijenos jedne vrste podataka pa u multimedijalnoj komunikaciji gdje se razmjenjuje više vrsta podataka za svaku vrstu treba biti uspostavljena zasebna sjednica. Jedan sudionik može biti aktivan u više sjednica istovremeno, primjerice u jednoj sjednici za razmjenu audio podataka i u još jednoj sjednici za razmjenu video podataka. Iz dosadašnjeg teksta možemo zaključiti kako jedna SIP sjednica uspostavlja jednu ili više RTP sjednica!

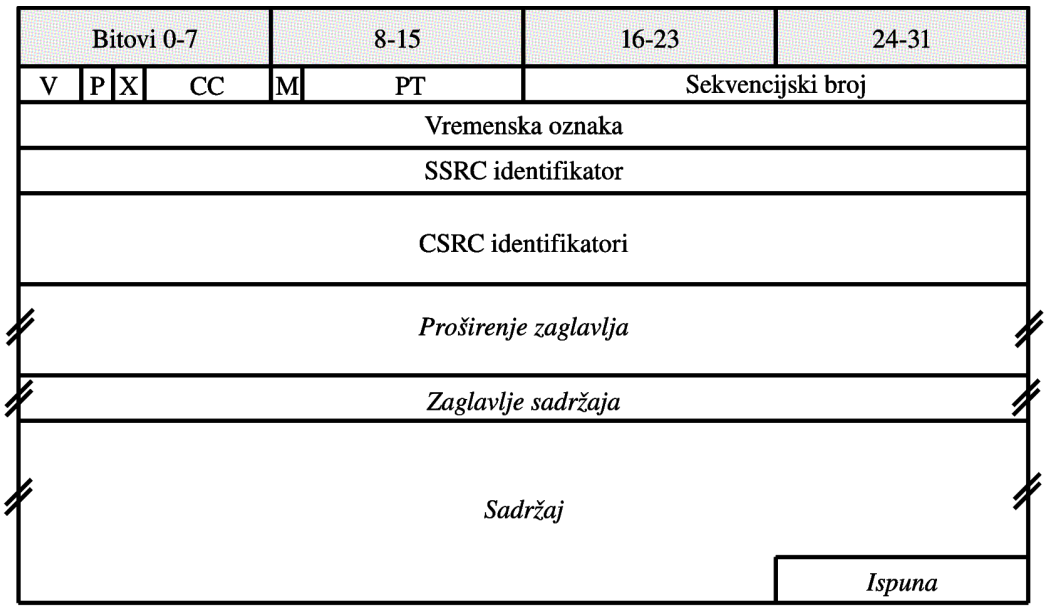
Sa stanovišta svakog sudionika, sjednica je definirana s mrežnom adresom i parom pristupa na koje treba slati podatke te parom pristupa na kojima se primaju podaci. Parovi pristupa su susjedni pristupi, gdje manji, paran pristup predstavlja RTP pristup, a veći, neparan pristup predstavlja RTCP pristup.

U svrhu identifikacije sudionika RTP sjednice, umjesto korištenja mrežne adrese i pristupa, bolje je koristiti mehanizam RTP protokola koji se zove *SSRC* (engl. Synchronization Source). Svaki sudionik sjednice odabire, kao svoj identifikator, slučajan broj duljine 32 bita koji se naziva SSRC i on se zapisuje u zaglavlje svih RTP paketa koje šalje taj sudionik. Budući da se slučajni brojevi generiraju lokalno, moguća je situacija kada dva sudionika odabiru isti SSRC broj. Takva kolizija se može detektirati kad neki sudionik primi RTP paket sa SSRC brojem koji je jednak njegovom. Za pomoć pri razrješavanju kolizije koristi se RTCP protokol sa svojom RTCP BYE porukom koju sudionik koji je detektirao koliziju mora poslati s originalnim SSRC-om te nakon toga odabrati novi SSRC za sebe.

Format RTP paketa za prijenos podataka prikazan je na slici 2.5. Paket možemo podijeliti na četiri dijela:

1. RTP zaglavlje,
2. proširenje zaglavlja,
3. zaglavlje sadržaja i
4. sadržaj.

RTP zaglavlje se sastoji od slijedećih polja:



Slika 2.5: Format RTP paketa

- Verzija (V): 2 bita
  Trenutna aktivna verzija RTP protokola je 1.
- Zastavica ispune (P): 1 bit
  Ako je ovaj bit postavljen onda se na kraju sadržaja nalazi ispuna koja nije dio korisnog sadržaja. Posljednji oktet ispune govori koliko je ispuna velika u oktetima.
- Zastavica proširenja (X): 1 bit
  Ako je ovaj bit postavljen onda se nakon zaglavlja nalazi proširenje zaglavlja. Proširenje zaglavlja je predviđeno kako bi se pojedinim aplikacijama omogućilo eksperimentirati s novim mogućnostima koje ne ovise o vrsti sadržaja, a koje zahtjevaju prijenos dodatnih informacija unutar RTP zaglavlja.
- CSRC brojač (CC): 4 bita
  Ovaj brojač označava koliko CSRC identifikatora se nalazi u zaglavlju.
- Oznaka (M): 1 bit
  Namjena ove oznake definirana je od strane RTP profila.
- Vrsta sadržaja (PT): 7 bitova
  Ovo polje određuje format RTP sadržaja. RTP profil definira kako se ovo polje mapira u specifikaciju formata.
- Sekvencijski broj: 16 bitova
  Ovaj broj se uvećava za jedan kod svakog slanja paketa i primatelj ga koristi za detekciju izgubljenih paketa.

- Vremenska oznaka: 32 bita
  Ova polje služi kao oznaka vremenskog trenutka kada je uzorkovan prvi oktet sadržaja. Tumačenje tog polja, primjerice vremenska jedinica koja se koristi, definirana je od strane RTP profila.
- SSRC: 32 bita
  Identifikator izvora podataka
- CSRC: 32 x CC bitova
  Ovo polje sadrži listu SSRC identifikatora od izvora koji doprinose sadržaju, a ubačena je u zaglavlje od strane RTP miksera. RTP mikser je sustav posrednik koji prima RTP pakete od grupe izvora, kombinira ih u jedan izlaz i eventualno promijeni način kodiranja prije nego što proslijedi rezultat.

2.6. SRTP

SRTP, tj. sigurni RTP, je profil RTP protokola namjenjen za pružanje enkripcije, autentifikacije i integriteta poruka te zaštite od ponavljanja za RTP i RTCP protokole [8].

SRTP ne pruža uslugu razmjene ključeva, prema tome potreban je vanjski protokol za razmjenu ključeva. Budući da SRTP koristi jedan glavni ključ iz kojeg onda generira ostale ključeve pomoću kriptografski sigurne funkcije sažimanja (engl. hash function), protokol za razmjenu ključeva treba razmjeniti samo jedan glavni ključ.

SRTP kriptografski kontekst se sastoji od: glavnog ključa, sjedničkog ključa, identifikatora enkripcijskog algoritma (tj. vrsta algoritma i operacijski mod), identifikatora algoritma za autentifikaciju poruke, vremena života sjedničkih ključeva i *brojača prekoračenja* (engl. Rolover Counter, ROC). Kriptografski kontekst treba biti održavan i od strane pošiljaoca i od strane primaoca za vrijeme trajanja SRTP sjednice i to za svaki SSRC po jedan.

ROC se održava na strani primaoca, a uvećava se za 1 svaki put kada SEQ broj prekorači svoju maksimalnu vrijednost. SEQ je 16 bitni broj koji je dio zaglavlja RTP paketa i monotono raste - svaki novi paket dobije SEQ za 1 veći od prethodnog.

Za enkripciju podataka, SRTP koristi AES enkripcijski algoritam u dva operacijska moda: mod brojača (engl. counter mode) i f-8 mod. AES na ulazu prima trojku (ključ, SSRC, SEQ), gdje je "ključ" enkripcijski ključ. Umjesto da koristi AES kao enkripcijski algoritam nad blokom podataka (engl. block cipher), SRTP ga koristi kao enkripcijski algoritam nad tokom podataka (engl. stream cipher) tako da datagram kriptira XOR-ujući ga s AES(ključ, SSRC, SEQ).

SRTP koristi kriptografski sigurnu pseudo-slučajnu funkciju (engl. Pseudo Random Function, PRF) kako bi se generirali sjednički ključevi za enkripciju i autentifikaciju. Ključevi se generiraju iz glavnog ključa, slučajne vrijednosti i SEQ broja (inicijalni SEQ broj je odabran od strane pošiljaoca). Glavni ključ i slučajna vrijednost se deterministički izvode upotrebom HMAC algoritma nad tekstom koji je definiranim od strane vanjskog protokola za razmjenu ključeva. Ključ koji koristi HMAC algoritam je razmjenjen vanjskim protokolom.

Način kako radi SRTP kriptografski algoritam u kriptografiji se zove *"one-time pad"*. Taj algoritam zahtjeva da se isti ključ može koristiti samo jednom. Problem koji se zove *"two-time pad"* nastaje kada se isti ključ koristi za kriptiranje različitih podataka, a može se pojaviti u SRTP-u ako pošiljaoci koriste isti ključ i istu RTP SSRC vrijednost.

3. Razmjena ključeva za SRTP

U ovom poglavlju obrađeni su neki osnovni pojmovi i scenariji korištenja protokola SIP koji imaju poseban utjecaj na protokol za razmjenu ključeva; dan je pregled zahtjeva koje treba zadovoljiti protokol za razmjenu ključeva za SRTP; nabrojani su i ukratko objašnjeni svi bitni protokoli koji su danas u upotrebi.

3.1. Pojmovi bitni za rezumijevanje zahtjeva

Prije nego što počnemo konkretno nabrajati specifične zahtjeve objasniti ćemo slijedeće pojmove koji su nužni za razumjevanje tih zahtjeva:

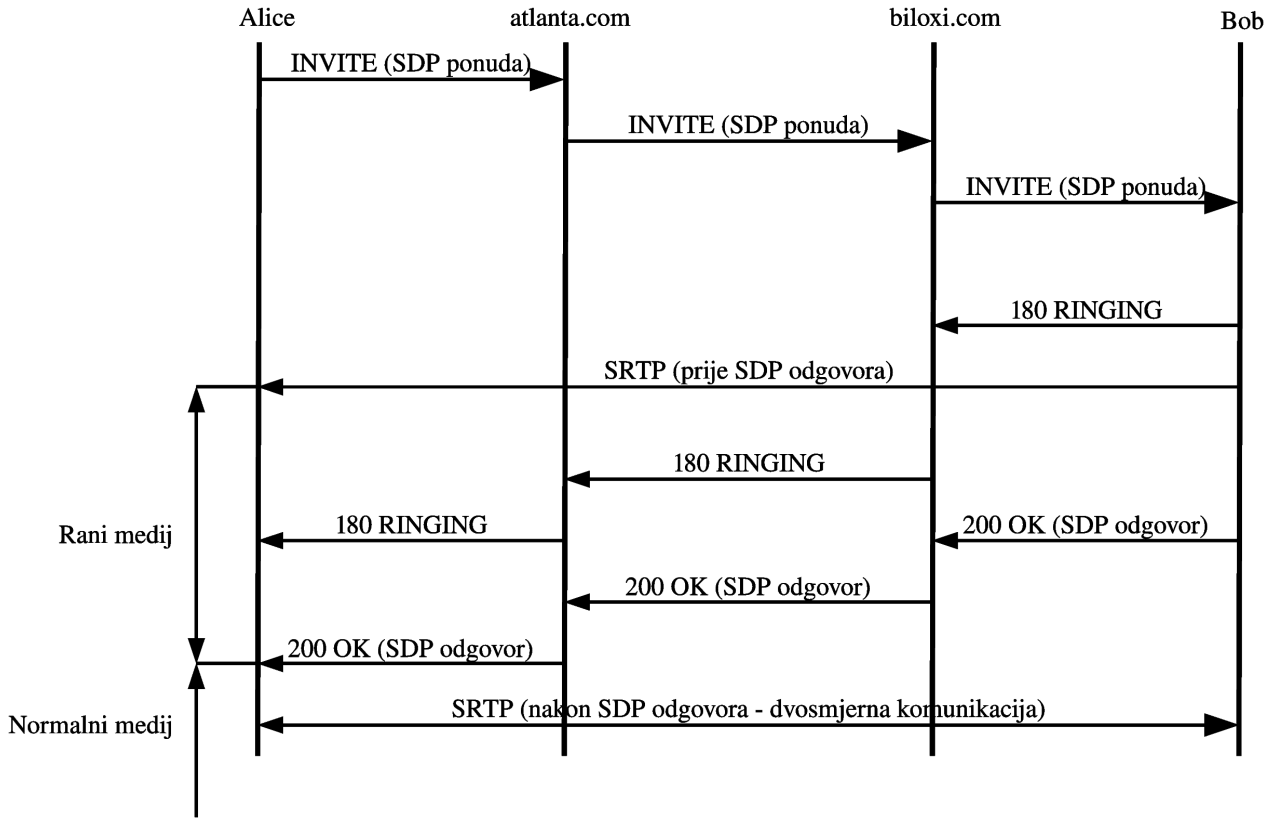
- Problem odsjecanja medija;
- SIP preusmjeravanje i grananje;
- Savršena tajnost prema unaprijed.

3.1.1. Problem odsjecanje medija

Prema modelu ponuda/odgovor SDP protokola, čim je ponuditelj poslao ponudu, on mora biti spreman primiti i reproducirati multimedijalne podatke za svaki medij koji se nalazi u ponudi - dakle, prije nego što je stigao SDP odgovor na ponudu.

Kako bi se zadovoljio ovaj zahtjev unutar SRTP protokola, ponuditelj mora znati ključeva za dolazeći medij. U suprotnome, medij se ne može dekriptirati, tj. dolazi do *odsjecanja medija* (engl. media clipping) što je prikazano na slici 3.1.

Ispunjavanje ovog zahtjeva je problem onim protkolima koji dovršavaju razmjenu ključeva tek kad stigne SDP odgovor (unutar 200 OK SIP odgovora). Tim protokolima može koristiti privremeni odgovor 183 koji služi za obavijesti o napredovanju sjednice. Taj privremeni odgovor bi sadržavao SDP odgovor. Ali privremeni odgovori (1xx) su nepouzdani jer se za privremene odgovore ne šalje ACK. Iznimka je slučaj kada obje strane podržavaju PRACK [20] ili koriste TCP između svih SIP posredničkih poslužitelja ili implementiraju sigurnosne preduvjete [21] ili ako obje strane implementiraju ICE [27] i odgovaratelj implementira pouzdan mehanizam za privremene odgovore koji je opisan u ICE-u. Protokol ICE koriste multimedijalne sjednice, uspostavljene modelom ponuda/odgovor kakav koristi SIP, za prolazak kroz NAT. Nažalost, niti jedan od ovih mehanizama nije u široj upotrebi i protokol za razmjenu ključeva ne smije zahtjevati njihovu upotrebu u svrhu izbjegavanja odsjecanja medija.



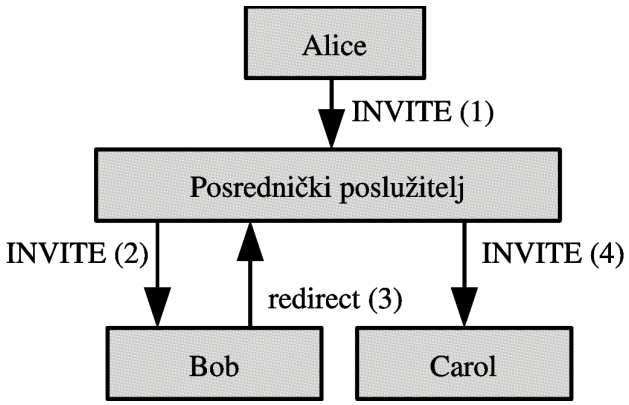
Slika 3.1: Situacija u kojoj može doći do odsjecanja medija; Rani medij i normalni medij

Ovakvo ponašanje je uvedeno u SIP da bi se podržao tzv. *rani medij* (engl. early media). Pojam rani medij unutar protokola SIP odnosi se na medij koji se razmjenjuje unutar medijskog kanala prije nego što se sjednica prihvati od strane pozivanog korisnika. Znači, rani medij se pojavljuje od trenutka kada se pošalje INVITE (koji sadrži SDP ponudu) pa sve dok poslužitelj korisničkog agenta ne generira konačni odgovor (npr. 200 OK koji sadrži SDP odgovor). Rani medij može biti jednosmjeran ili dvosmjernan i može ga generirati pozivatelj, pozivani ili oboje. Tipičan primjer ranog medija koji generira pozivani je ton zvona ili neka objava (npr. stanje čekanja), a tipičan primjer ranog medija koji generira pozivatelj je glasovna komanda ili DTMF u svrhu upravljanja sustavom za interaktivni glasovni odziv (engl. interactive voice response, IVR). U trenutku kada se pošalje 200 OK odgovor (slika 3.1) rani medij prelazi u normalni medij (engl. regular media).

3.1.2. SIP preusmjeravanje i grananje

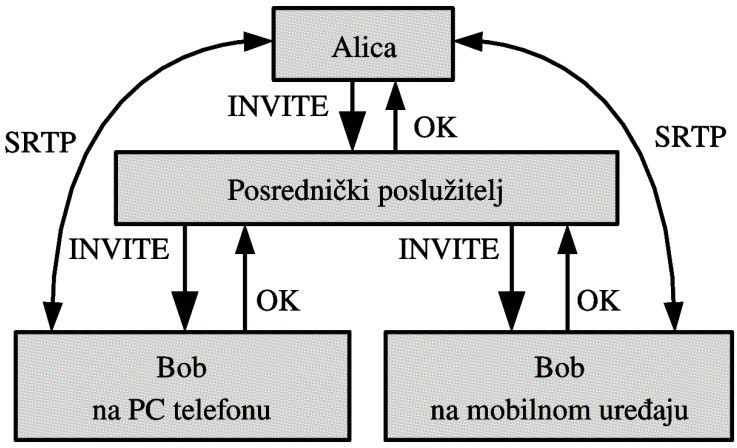
U protokolu SIP, zahtjev poslan prema nekom određenom AOR-u, a isporučen na neki drugi AOR zove se preusmjeravanje. Tipičan primjer je usluga *proslijeđivanja poziva* (engl. call forwarding) koja je prikazana na slici 3.2. Zbog mogućnosti preusmjeravanja pozivatelj ne može znati unaprijed tko će prihvatiti poziv, Bob ili Carol.

Grananje je isporučivanje jednog SIP zahtjeva na više lokacija (slika 3.3). To se dešava kada je AOR registriran na više lokacija, npr. na programskom PC telefonu i na mobilnom telefonskom uređaju.



Slika 3.2: SIP preusmjeravanje

U slučaju grananja, posrednički poslužitelj prima odgovore od više krajnjih točaka. Ako su to privremeni odgovori (1xx) ili konačni odgovori uspjeha (2xx) on te odgovore prosljeđuje krajnjoj točki koja je poslala zahtjev. Ako je riječ o konačnim odgovorima greške, posrednički poslužitelj prosljeđuje samo jedan, po njemu najbolji odgovor. To znači ako je zahtjev odbijen u nekoj krajnjoj točki u većini slučajeva taj odgovor nikad neće stići do krajnje točke koja je poslala zahtjev. Zahtjev može biti odbijen npr. zbog toga što su informacije o ključevima odbijene. To znači da povratne informacije o toj krajnjoj točki, koje su poslane prema pošiljatelju u svrhu ponavljanja zahtjeva, nikad neće stići do pošiljatelja. Ovaj problem se naziva **HERFP** (engl. Heterogeneous Error Response Forking Problem) i jako teško ga je riješiti unutar protokola SIP i vjerojatno će predstavljati problem i ubuduće, a to znači da protokol za razmjenu ključeva mora funkcionirati i u njegovom prisustvu.



Slika 3.3: SIP grananje

3.1.3. Savršena tajnost prema unaprijed

Savršena tajnost prema unaprijed (engl. Perfect Forward Secrecy) je svojstvo protokola za razmjenu ključeva da razotkrivanje trajnih ključeva neće kompromitirati ključeve koji su prije toga izvedeni iz tih trajnih ključeva [22].

Za podržavanje ovog svojstva protokoli obično koriste Diffie-Hellman razmjenu ključeva.

3.2. Zahtjevi

Zahtjevi su podijeljeni u tri skupine [33]. U prvoj skupini su zahtjevi koji se odnose na sam protokol za razmjenu ključeva i prikazani su u tablici 3.1. U drugoj skupini su sigurnosni zahtjevi i zahtjevi koji se odnose na različite scenarije napada i prikazani su u tablici 3.2. I konačno, u trećoj skupini se nalaze zahtjevi koji se mogu podržati unutar ili izvan protokola za razmjenu ključeva i prikazani su u tablici 3.3.

SIP grananje i preusmjeravanje	
R-FORK-RETARGET	Protokol mora, na siguran način, podržati grananje i preusmjeravanje. Rješenje ne smije biti takvo da u slučaju grananja i preusmjeravanje poziv ne uspije. Ovo mora vrijediti i u slučaju kada sve krajnje točke žele koristiti SRTP i u slučaju kada postoji mješavina onih koji žele SRTP i onih koji žele RTP. Ovaj zahtjev podrazumjeva da krajnje točke koje nisu odgovorile na poziv ne smiju saznati SRTP ključeve koji će se koristiti u uspostavljenom pozivu.
R-DISTINCT	Protokol mora uspostaviti različite ključeve za svaku krajnju točku u pozivu u kojem je prisutno grananje.
R-HERFP	Protokol mora sigurno funkcionirati čak i u slučaju HERFP ponašanja.
Performanse	
R-REUSE	Protokol može podržati ponovno korištenje prethodno uspostavljenog sigurnosnog konteksta. Ovo svojstvo se koristi kako bi se smanjila količina kriptografskog računanja kada dvije strane ponovo uspostavljaju komunikaciju.
Medij	
R-AVOID-CLIPPING	Protokol treba izbjegavati prekide u reprodukciji prije nego što stigne SDP odgovor bez da zahtjeva sigurnosne preduvjete za SDP [21].
R-RTP-VALID	Ukoliko protokol koristi medijski kanal za razmjenu ključeva (koristeći iste UDP/TCP pristupe kao i medijski paketi), paketi za razmjenu ključeva ne smiju proći RTP test valjanosti kako je to definirano u dodatku A.1 od RTP RFC-a [7].
R-ASSOC	Protokol bi trebao uključiti poveznicu između samih poruka protokola za razmjenu ključeva i poruka signalnog i medijskog kanala. Ako je takva poveznica omogućena, moguće je izbjeći razmjenu ključeva (što može biti vremenski intezivna operacija) s napadačima koji nisu vidjeli signalne poruke.
R-NEGOTIATE	Protokol mora dopustiti SIP korisničkom agentu da dogovara sigurnosne parametre za svaku pojedinačnu sjednicu.
R-PSTN	Protokol mora podržati prestanak sigurnosti medijskog kanala na PSTN povezniku.

Tablica 3.1: Zahtjevi koji se odnose na protokol za razmjenu ključeva

R-PFS	Protokol mora podržavati PFS, tj. savršenu sigurnost prema unaprijed.
R-COMPUTE	Protokol mora podržati ponudu dodatnih SRTP profila bez potrebe za značajnim računskim resursima.
R-CERTS	Ukoliko protokol koristi certifikate, tada treba omogućiti korištenje i samo-potpisanih i CA-potpisanih certifikata.
R-FIPS	Protokol treba koristiti algoritme koji dopuštaju upotrebu FIPS 140-2 certifikata.
R-DOS	Protokol ne bi trebao omogućiti nove vrste DOS (engl. Denial of Service) napada, tj. protokol ne bi trebao zahtjevati izvođenje značajnih računskih operacija ako zahtjevi nisu validni ili ovjereni.
R-EXISTING	Protokol treba dopustiti krajnjim točkama autentificiranje korištenjem već postojećih kriptografskih ključeva, tj. certifikata ili unaprijed podijeljenih zajedničkih ključeva.
R-AGILITY	Protokol treba omogućiti nadogradnju, tj. prilagodbu prema novim kriptografskim ili sigurnosnim zahtjevima. To npr. znači da se za već postavljene implementacije treba omogućiti nadogradnja na nove vrste kriptografskih algoritama bez potrebe za značajnijim izmjenama.
R-DOWNGRADE	Protokol mora zaštiti postupak dogovaranja kriptografskih parametara od *napada degradiranja* (engl. downgrading attack).
R-PASS-MEDIA	Protokol mora podržavati način rada koji onemogućuje pasivnog napadača koji ima pristup medijskom kanalu da dođe u posjed ključevima koji se koriste za zaštitu SRTP paketa.
R-PASS-SIG	Protokol mora podržavati način rada koji onemogućuje pasivnog napadača koji ima pristup signalnom kanalu da dođe u posjed ključevima koji se koriste za zaštitu SRTP paketa.
R-ID-BINDING	Protokol mora podržati kriptografsko povezivanje ključeva i identiteta krajnje točke. (Ovaj zahtjev omogućuje upotrebu SIP Identity protokola [9])
R-ACT-ACT	Protokol mora podržavati način rada koji detektira napadača koji ima aktivni pristup i na signalni i na medijski kanal. Takoder, protokol može podržavati i načine rada koji pružaju slabije garancije.

Tablica 3.2: Sigurnosni zahtjevi i zahtjevi specifični za različite scenarije napada

R-BEST-SECURE	U slučaju kada neke krajnje točke od odgranatog ili preusmjerenog poziva ne podržavaju SRTP, mora biti opisano rješenje koje dopušta uspostavljanje SRTP sjednice s onim krajnjim točkama koje podržavaju SRTP, a s onim krajnjim točkama koje ne podržavaju SRTP dopušta uspostavljanje RTP sjednice.
R-OTHER-SIGNALING	Rješenje treba omogućiti razmjenu ključeva za SRTP sjednice kreirane od strane različitih signalnih protokola za uspostavu poziva (tj. SIP, Jabber, H.323, MGCP).
R-RECORDING	Treba biti opisano rješenje koje podržava snimanje dekriptiranog RTP prometa od strane posrednika za snimanje.
R-TRANSCODER	Treba biti opisano rješenje koje podržave poslužitelje posrednike koji se nalaze na medijskog kanalu između dviju kranjih točaka (npr. transkoderi), a koji terminiraju ili obrađuju medij.
R-ALLOW-RTP	Treba biti opisano rješenje koje dopušta primanje RTP prometa prije nego što je dogovorena SRTP veza, nakon čega se SRTP preferira u odnosu na RTP.

Tablica 3.3: Zahtjevi koji se mogu podržati unutar ili izvan protokola za razmjenu ključe

3.3. Podjela metoda s obzirom na vrste napada

S obzirom na vrstu pristupa koje napadači imaju napade dijelimo na one koji imaju pristup samo signalnom kanal, samo medijskom kanalu ili na oba kanala.

S obzirom na mogućnosti koje napadači imaju napade dijelimo na pasivne u kojima napadači podatke mogu samo čitati i na aktivne u kojima napadači podatke mogu i mijenjati.

U ovom radu ne analiziramo napade u kojima napadači imaju pasivni ili aktivni pristup samo na signalni kanal.

S obzirom na vrste napada metode za razmjenu ključeva dijelimo na ove grupe:

1. Dovoljan je pasivni napad na medijski kanal da bi se otkrio sadržaj podataka koji se razmjenjuju.
2. Potreban je aktivni napad na medijski kanal da bi se otkrio sadržaj podataka koji se razmjenjuju.
3. Potreban je pasivni napad i na signalni i na medijski kanal da bi se otkrio sadržaj podataka koji se razmjenjuju.
4. Potreban je pasivni napad na signalni kanal i aktivni napad na medijski kanal da bi se otkrio sadržaj podataka koji se razmjenjuju.

5. Potreban je aktivni napad na na signalni kanal i pasivni napad na medijski kanal da bi se otkrio sadržaj podataka koji se razmjenjuju.
6. Potreban je aktivni napad i na signalni i na medijski kanal da bi se otkrio sadržaj podataka koji se razmjenjuju.
7. Iako je napad aktivan na oba kanal, napad je otkriven.

Nevedimo neke protokole i kojoj grupi pripadaju:

- RTP pripada u grupu 1.
- SDES pripada u grupi 3 jer napadač saznaje ključ gledanjem SIP poruke na strani SIP posredničkog poslužitelja.
- DTLS-SRTP bez implementiranog protokola za ovjeru potpisa certifikata pripada u grupu 6.
- DTLS-SRTP s implementiranim protokolom za ovjeru potpisa certifikata pripada u grupu 7.

3.4. Metode za razmjenu SRTP ključeva

S obzirom na način kako razmjenjuju SRTP ključeve, metode za razmjenu SRTP ključeva dijelimo na slijedeće grupe:

1. Koristi se samo signalni kanal za razmjenu ključeva.
2. Koristi se samo medijski kanal za razmjenu ključeva.
3. Koristi se i signalni i medijski kanal za razmjenu ključeva.

U nastavku se nalazi pregled metoda za razmjenu ključeva. Pregled je dan u sažetom obliku, osim za DTLS-SRTP metodu koja je opisana detaljno u poglavlju 4.

3.4.1. MIKEY

MIKEY (engl. Multimedia Key Exchange) [11] radi na način da se razmjena ključeva obavlja unutar protokola za dogovaranje parametara komunikacije u multimedijalnoj sjednici, tj. najčešće unutar modela ponuda/odgovor SDP protokola. Stoga ova metoda za razmjenu ključeva spada u grupu 1, tj. koristi se samo signalni kanal za razmjenu ključeva.

Budući da model ponuda/odgovor radi na način da se šalju dvije poruke - prvo od ponuditelja prema odgovaratelju, a onda od odgovaratelja prema ponuditelju - to znači da i sam MIKEY protokol treba obaviti razmjenu ključeva i međusobnu autentifikaciju unutar te dvije poruke.

U svrhu zaštite od ponavljanja poruke, MIKEY ne koristi *izazov-odziv* (engl. challange-response) mehanizam, već koristi vremensko označavanje poruka zajedno s privremenom pohranom poruka. Zbog toga MIKEY zahtjeva relativno dobro sinkronizirane satove između dvije krajnje točke (koliko dobro ovisi o veličini spremnika za privremenu pohranu poruka koji postoji na obje strane).

MIKEY podržave više načina razmjene ključeva:

- MIKEY-NULL [11]. U ovom načinu ponuditelj šalje SRTP ključeva za oba smjera. Ključevi se šalju nezaštićeni unutar SDP-a, što znači da se SDP mora zaštititi ili korištenjem SIPS TLS zaštite korak po korak ili korištenjem S/MIME zaštite s kraja na kraj.
- MIKEY-PSK [11]. U ovom načinu ponuditelj šalje SRTP ključeve za oba smjera. Ključevi se šalju kriptirani pomoću zajedničkog ključa koji je unaprijed poznat svim krajnjim točkama koje sudjeluju u pozivu.
- MIKEY-RSA [11]. U ovom načinu ponuditelj šalje SRTP ključeva za oba smjera. Ključevi se šalju kriptirani pomoću javnog ključa odgovaratelja. Način dohvaćanja javnog ključa odgovaratelja nije definiran od strane MIKEY protokola.
- MIKEY-RSA-R [12]. U ovom načinu odogovarač šalje SRTP ključeve za oba smjera. Ključevi se šalju kriptirani pomoću javnog ključa ponuditelja. I ponuditelj i odgovaratelj ovjeravaju jedan drugome ključeve koristeći standardne X.509 tehnike za ovjeravanje.
- MIKEY-DHSIGN [11]. U ovom načinu ponuditelj i odgovaratelj izvode ključeve pomoću Diffie-Hellman razmjene. Kako bi se spriječio aktivni napad tipa *"čovjek u sredini"* (engl. man-in-the-middle), DH razmjena je potpisana s privatnim ključevima svake krajnje točke, dok se pridruženi javni ključevi ovjeravaju koristeći standardne X.509 tehnike za ovjeravanje.
- MIKEY-DHHMAC [13]. Isto kao MIKEY-DHSIGN, ali umjesto autentifikacije certifikatima, koristi se unaprijed poznati zajednički ključ (kao u MIKEY-PSK) kako bi se HMAC autentificirala DH razmjena.
- MIKEY-ECIES i MIKEY-ECMQV (MIKEY-ECC). Ovo su novi načini MIKEY razmjene ključeva opisani u [ietf-msec-mikey-ecc]. Opisano je kako ECC (engl. Elliptic-Curve Cryptography, odnosno kriptografija zasnovana na eliptičkim krivuljama) može biti upotrebljena s MIKEY-RSA i MIKEY-DHSIGN načinima razmjene ključeva, a opisana su i dva nova načina razmjene ključeva zasnovana na ECC-u: ECIES (engl. Elliptic Curve Integrated Encryption Scheme) i ECMQV (engl. Elliptic Curve Menezes-Qu-Vanstone). Ovi načini su po svojim karkteristikama ekvivalentni MIKEY-RSA, tj. MIKEY-DHSIGN načinima pa se u daljnjem tekstu ne spominju.

Što se tiče SIP grananja i preusmjeravanja, podržavaju ih na siguran način MIKEY-RSA-R, MIKEY-DHSIGN i MIKEY-DHHMAC. Dok ih drugi načini ne podržavaju jer u njima ponuditelj šalje SRTP ključeve za sve odgovaratelje, pa je na taj način prekršeno R-FORK-RETARGET i R-DISTINCT.

Što se tiče prekidanja u reprodukciji (R-AVOID-CLIPPING) prije nego što stigne SDP odgovor situacija je drukčija, do prekida dolazi kod MIKEY-RSA-R, MIKEY-DHSIGN i MIKEY-DHHMAC, zato jer je za konačnu razmjenu ključeva podreban SDP odgovor. Kod MIKEY-NULL, MIKEY-PSK i MIKEY-RSA ne dolazi do prekida jer ponuditelj šalje SRTP ključeve za oba smjera.

Savršenu sigurnost prema unaprijed (PFS) podržavaju MIKEY-DHSIGN, MIKEY-DHHMAC.

3.4.2. SDES

SDES (engl. Security Descriptions) [14] radi na način da svaka strana šalje svoj SRTP ključ. Ključevi se šalju nezaštićeni unutar SDP-a, što znači da se SDP mora zaštititi ili korištenjem SIPS TLS zaštite korak po korak ili korištenjem S/MIME zaštite s kraja na kraj. Ova metoda spada u grupu 1, tj. koristi se samo signalni kanal za razmjenu ključeva.
SDES ne podržava R-FORK-RETARGET, R-DISTINCT i R-AVOID-CLIPPING zahtjeve.

3.4.3. ZRTP

ZRTP [32] metoda ne koristi signalni kanal za razmjenu ključeva, iako ponuditelj može indicirati da podržava ZRTP dodavanje "a=zrtp" atributa unutar SDP ponude, a novije verzije drafta uvode još neke atribute koji se mogu slati kroz SDP. ZRTP razmjenjuje ključeve isključivo kroz medijski kanalkoristeći Diffie-Hellman razmjenu ključeva. ZRTP poruke se multipleksiraju zajedno sa SRTP-om na istom pristupu. ZRTP ne koristi PKI (engl. Public Key Infrastructure) i ne zahtjeva certifikate u krajnjim točkama. ZRTP koristi, u svrhu detekcije napada tipa "čovjek u sredini", glasovnu autentifikaciju na način da svaka osoba koja sudjeluje u pozivu mora pročitati na glas *kratki niz karaktera za autentifikaciju* (engl. Short Authentication String, SAS) drugoj osobi i usporediti ga da vidi da li se poklapa. Slično kao i kod DTLS-SRTP metode, autentifikaciju je moguće obaviti i u suradnji sa signalnim kanalom. To se radi tako da se pošalje "a=zrtp-hash" atribut kroz SDP, a integritet tog atributa se štiti "SIP Idenitiy" mehanizmom [9] ili bolje "SIP Identity using Media Path" mehanizmom [29].

ZRTP protokol je predložio Phil Zimmerman, autor programa PGP koji se koristi za potpisivanje i kriptiranje elektroničke pošte.

ZRTP zadovoljava sve zahtjeve koji su postavljeni na protokol za razmjenu SRTP ključeva.

4. DTLS-SRTP

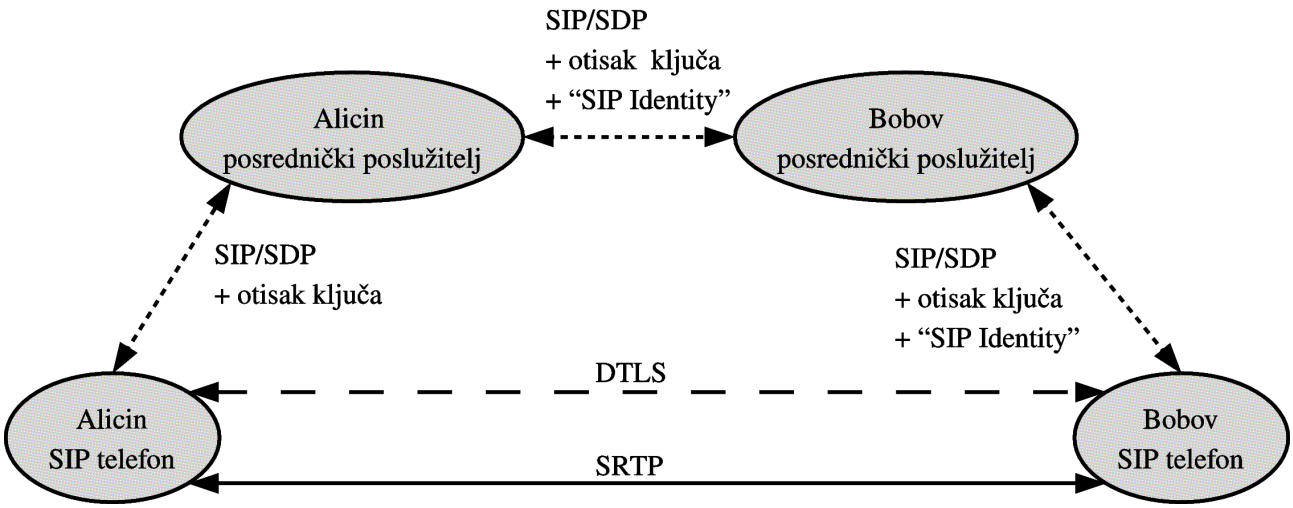
DTLS-SRTP protokol [23] je predložio Eric Rescorla, koautor TLS 1.1 i DTLS protokola, kao alternativu ZRTP protokolu. Za razliku od ZRTP protokola koji definira novi mehanizam za razmjenu ključeva, protokol DTLS-SRTP je zasnovan na postojećem DTLS protokolu. DTLS-SRTP je u konkurenciji ZRTP i MIKEY protokola na IETF BOF sastanku u Pragu iz svibnja 2007. odabran kao protokol koji će biti osnova za budući rad u ovom području.

DTLS-SRTP protokol za razmjenu SRTP ključeva koristi i signalni i medijski kanal. Signalni kanal se koristi kako bi se razmjenili, pomoću SDP protokola, *otisci certifikata* (engl. certificate fingerprint). Ti certifikati se koriste tijekom DTLS *dogovaranja* (engl. handshake) koje se obavlja unutar medijskog kanala. Kao i kod ZRTP protokola, DTLS poruke se multipleksiraju zajedno s porukama SRTP protokola na istom pristupu.

Tijekom DTLS dogovaranja, klijent koji je obično SDP odgovaratelj i poslužitelj koji je obično SDP ponuditelj, koristeći Diffie-Hellman razmjenu dogovore glavni ključ iz kojeg se onda pomoću funkcije za izvođenje ključeva izvode SRTP glavni ključevi i slučajne vrijednosti za oba smjera: od klijenta prema poslužitelju i od poslužitelja prema klijentu. Također, tijekom dogovaranja, koristeći "use-srtp" ekstenziju DTLS protokola, klijent i poslužitelj dogovaraju i vrste algoritama te njihove parametre koji će se koristit tijekom SRTP komunikacije.

DTLS-SRTP protokol koristi samo-potpisane certifikate čija sigurnost je osigurana prenošenjem sigurnosnog potpisa tih certifikata unutar SDP poruka kojima je integritet osiguran.

Na slici 4.1 skiciran je SIP trapezoid u slučaju kada se koristi DTLS-SRTP.



Slika 4.1: SIP trapezoid kada se DTLS-SRTP koristi za zaštitu RTP prometa

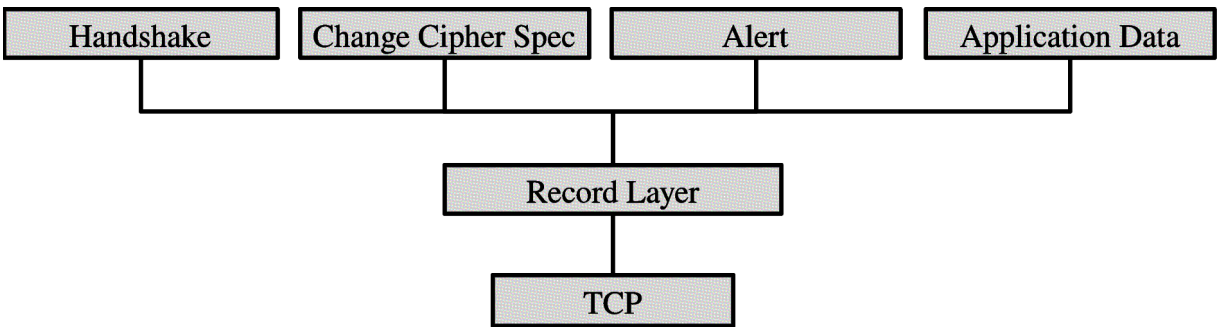
4.1. Osnove protokola DTLS

DTLS (engl. Datagram Transport Security Layer) protokol [16] omogućuje sigurnu mrežnu komunikaciju između aplikacija koje koriste bespojni protokol kao što je UDP. Pod sigurnom komunikacijom podrazumjevamo privatnost i integritet podataka.

DTLS koristi sve elemente TLS protokola [15] ali uz određene preinake koje su nužne zbog činjenice da TLS radi samo preko pouzdanog prijenosnog protokola kao što je TCP, dok DTLS mora raditi preko nepouzdanog prijenosnog protokola kao što je UDP.

Budući da je DTLS zasnovan na TLS protokolu u daljnjem tekstu opisujemo TLS protokol te navodimo razlike koje uvodi DTLS.

TLS je višeslojni protokol (slika 4.2). Na nižem sloju je *rekord protokol* koji preuzima od višeg sloja poruke koje se žele prenijeti pomoću TCP protokola koji pripada prijenosnom sloju. Rekord protokol dijeli te poruke u veličine pogodne za prijenos, opcionalno ih kompresira, aplicira MAC (engl. Message Authentication Code, odnosno kôd za autentifikaciju poruka), kriptira te predaje rezultat TCP prijenosnom protokolu za daljnji prijenos.



Slika 4.2: Višeslojna struktura TLS protokola

Klijenti rekord protokola su *protkol dogovaranja* (engl. Handshake Protocol), *protokol promjene specifikacije algoritama* (engl. Change Cipher Spec Protocol), *protokol upozorenja* (engl. Alert Protocol) i *protokol razmjene aplikacijskih podataka* (engl. Application Data Protocol).

Protokol dogovaranja služi za dogovoranje parametara sjednice između klijenta i poslužitelja. Parametri sjednice su: identifikator sjednice, certifikati, vrsta algoritma za kompresiju, vrsta kripto i MAC algoritma, parametri tih algoritama i glavna tajna. Iz tih parametara izvode se sigurnosni parametri koje onda koristi rekord protokol.

U svakom trenutku, postoji dva skupa sigurnosnih parametara za TLS rekord protokol: tekući parametri koji se trenutno koriste od strane rekord protokola i parametri u pregovaranju koji se upravo dogovaraju pomoću protokola za dogovaranje. U jednom trenutku parametri u pregovaranju postaju tekući parametri, a to se signalizira pomoću protokola promjene specifikacije algoritama. Taj protokol definira samo jednu poruku: ChangeCipherSpec.

Protokol upozorenja koristi se za slanje upozorenja, koja mogu biti obična upozorenja ili fatalna upozorenja nakon kojih je potrebno ugasiti konekciju.

Protokol razmjene aplikacijskih podataka služi za prenošenje aplikacijskih podataka između klijenta i poslužitelja.

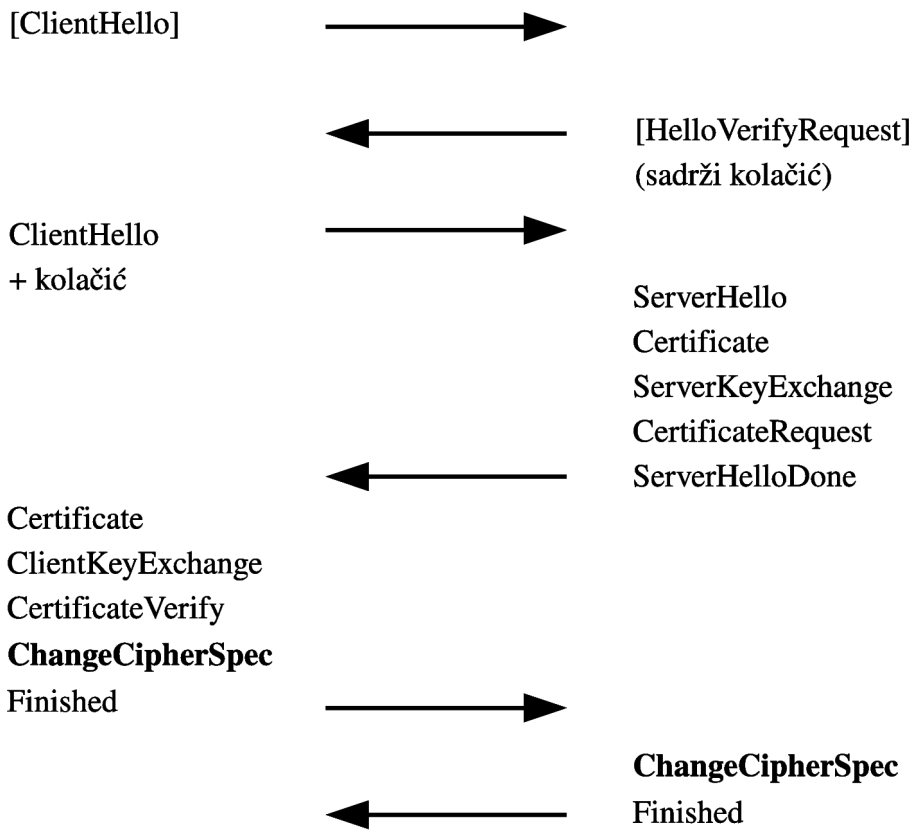
Pogledajmo kako izgleda osnovni okvir DTLS rekord protokola (opisni jezik definiran je u [15]):

```
enum {
    change_cipher_spec(20), alert(21), handshake(22),
    application_data(23), (255)
} ContentType;

struct {
    uint8 major, minor;
} ProtocolVersion;

struct {
    ContentType type;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    uint16 length;
    opaque fragment[DTLSPlaintext.length];
} DTLSPlaintext;
```

U odnosu na TLS, protokol DTLS uvodi dva nova polja: `epoch` i `sequence_number`. Polje `epoch` povećava se za 1 prilikom slanja ChangeCipherSpec poruke, te se na taj način spriječava da dođe do zabune oko toga koji skup sigurnosnih parametara je tekući; do zabune može doći zbog gubitaka podataka jer se koristi nepouzdan UDP protokol. Budući da UDP paketi ne moraju dolaziti redom kojim su poslani, ili čak isti paket može doći dva ili više puta, potrebno je u svakom okviru rekord protokola poslati `sequence_number`, tj. redni broj poruke.

Razmjena poruka unutar DTLS protokola za dogovaranje prikazana je na slici 4.3, a prikazan je primjer s *kratkotrajnom Diffie-Hellman* (engl. Ephermal Diffie-Hellman) razmjenom ključeva gdje su DH parametri potpisani koristeći javni RSA ključ koji se nalazi u X.509 certifikatima.

DTLS dogovaranje je gotovo identično TLS dogovaranju, ali postoje dvije veće razlike: 1) dodana je razmjena *kolačića* (engl. cookie) radi sprečavanje DoS napada i 2) dodana je fragmentacija i ponovno slaganje poruka. Ova potonja izmjena je potrebna zbog toga što DTLS rekord mora stati unutar jednog UDP datagrama, a maksimalna veličina datagrama između dva računala PMTU (engl. Path Maximum Transmission Unit) je manja od maksimalne veličine DTLS rekorda.



Slika 4.3: DTLS dogovaranje

Opišimo pojedine faze dogovaranja:

- **ClientHello:** klijent šalje verziju DTLS protokola koju podržava, slučajan broj, identifikator sjednice, kolačić, listu kripto algoritama i listu kompresijskih metoda koje klijent podržava. Slučajan broj se koristi za zaštitu od ponovljanja poruke.
- **HelloVerifyRequest:** poslužitelj šalje kolačić radi sprečavanja DOS napada. Klijent je dužan ponoviti kolačić u ClientHello poruci koja slijedi. Poslužitelj šalje ovu poruku kada ne može provjeriti valjanost kolačića kojeg je poslao klijent u prvoj ClientHello poruci. Ova poruka je opcionalna, pa ako se ne koristi onda je tijek dogovaranja identičan TLS-u.
- **ServerHello:** poslužitelj šalje izbor verzije, algoritama i svoj slučajan broj.

- **Certificate:** poslužitelj šalje X.509 certifikat koji sadrži RSA javni ključ koji poslužitelj koristi za potpisivanje DH parametera.
- **ServerKeyExchange:** poslužitelj šalje Diffie-Hellman parametre pomoću kojeg klijent može završiti razmjenu ključeva.
- **CertificateRequest:** poslužitelj šalje zahtjev da klijent pošalje svoj certifikat koji klijent koristi za potpisivanje DH parametara.
- **ServerHelloDone:** ova poruka označava da je to zadnja poruka od poslužitelja u ovom letu.
- **Certificate:** klijent šalje svoj X.509 certifikat.
- **ClientKeyExchange:** klijent šalje Diffie-Hellman parametre pomoću kojeg će obje strane moći završiti razmjenu ključeva.
- **CertificateVerify:** klijent šalje potpis, koristeći svoj certifikat, od svih prethodnih primljenih i poslanih poruka.
- **ChangeCipherSpec:** klijent šalje poruku koja označava da je prešao na upravo dogovorene sigurnosne parametre.
- **Finished:** klijent šalje poruku koja sadrži MAC svih prethodnih poruka. Poruka je zaštićena s upravo dogovorenim sigurnosnim parametrima.
- **ChangeCipherSpec:** poslužitelj šalje poruku koja označava da je prešao na upravo dogovorene sigurnosne parametre.
- **Finished:** poslužitelj šalje poruku koja sadrži MAC svih prethodnih poruka. Poruka je zaštićena s upravo dogovorenim sigurnosnim parametrima.

Pogledajmo kako izgleda okvir protokola za dogovaranje, koji se prenosi unutar rekord okvira:

```
enum {
    hello_request(0), client_hello(1), server_hello(2),
    hello_verify_request(3),
    certificate(11), server_key_exchange (12),
    certificate_request(13), server_hello_done(14),
    certificate_verify(15), client_key_exchange(16),
    finished(20), (255)
} HandshakeType;

struct {
   HandshakeType msg_type;
   uint24 length;
   uint16 message_seq;
   uint24 fragment_offset;
   uint24 fragment_length;
   opaque msg_fragment[Handshake.fragment_length];
} Handshake;
```

U odnosu na TLS, u DTLS okvirima protokola dogovaranja nalaze se tri nova polja: `message_seq`, `fragment_offset` i `fragment_length`. Polje `message_seq` se se koristi za detekciju dupliciranih

poruka, a polja `fragment_offset` i `fragment_length` su potrebna radi rekonstrukcije poruka koje su poslane fragmentirano.

Budući da se DTLS poruke mogu izgubiti zbog nepouzdanosti UDP protokola, DTLS sadrži mehanizam za retransmisiju poruka.

4.2. Korištenje DTLS protokola unutar DTLS-SRTP protokola

DTLS-SRTP sjednica se sastoji od jedne DTLS konekcije i jednog SRTP kriptografskog konteksta. Jedna DTLS-SRTP sjednica smije štititi podatke koji se prenose točno jednim parom UDP pristupa.

RTP promet između dvije krajnje točke A i B može ići u smjeru od A prema B i od B prema A. Ako se u oba slučaja koriste isti parovi pristupa samo obrnuto onda je riječ simetričnom RTP-u [17]. Taj način korištenja pristupa je sličan onome u TCP protokolu. Kada se koristi simetričan RTP jedna DTLS-SRTP sjedica je dovoljna za promet u oba smjera. Isto vrijedi i za RTCP protokol ako je simetričan.

RTP i RTCP promet obično se šalje preko dva različita UDP pristupa, pa ako je riječ o simetričnom RTP-u i RTCP-u onda su potrebne dvije DTLS-SRTP sjednice, jedna za RTP promet, a druga za RTCP promet; a ako RTP i RTCP nisu simetrični onda je potrebno četiri DTLS-SRTP sjednice.

RTP i RTCP mogu biti multipleksirani na jednom UDP pristupu [25], a ako se uz to koriste i simetrični RTP i RTCP tada je za zaštitu prometa dovoljna jedna DTLS-SRTP sjednica.

Uspostava DTLS-SRTP sjednice započinje uspostavom DTLS konekcije tako što se napravi DTLS dogovaranje s "use-srtp" ekstenzijom. Nakon toga, iz ključeva koji se razmjene s DTLS protokolom, generiraju se glavni ključ i slučajna vrijednost za oba SRTP smjera. Korištenjem "use-srtp" ekstenzije razmjenjuju se i neki drugi parameteri koji čine SRTP kriptografski kontekst.

4.2.1. "use-srtp" ekstenzija za DTLS

TLS protokol podržava ekstenzije koje se koriste za dodavanje novih funkcionalnosti u TLS protokol [18]. Isto vrijedi i za DTLS protokol.

ClientHello i ServerHello poruke koriste se za objavu korištenja ekstenzije i za dogovaranje parametara tih ekstenzija.

Pogledajmo kako izgleda ClientHello poruka koja je sadržana unutar poruke protokola za dogovaranje, a koja je pak sadržana unutar poruke rekord protokola:

```
struct {
      ProtocolVersion client_version;
      Random random;
      SessionID session_id;
      CipherSuite cipher_suites<2..2^16-1>;
      CompressionMethod compression_methods<1..2^8-1>;
      Extension client_hello_extension_list<0..2^16-1>;
} ClientHello;
```

Dakle, na kraju ClientHello poruke nalazi se `client_hello_extension_list` polje koje

sadrži nula ili više ekstenzija. Format svake od tih ekstenzija izgleda ovako:

```
struct {
      ExtensionType extension_type;
      opaque extension_data<0..2^16-1>;
} Extension;
```

Prvo polje je `extension_type` koji sadrži broj ekstenzije; naime, svaka ekstenzija ima pridružen broj. Unutar TLS-a definirani su ovi tipovi ekstenzija i njima pridruženi brojevi [18]:

```
enum {
      server_name(0), max_fragment_length(1),
      client_certificate_url(2), trusted_ca_keys(3),
      truncated_hmac(4), status_request(5), (65535)
} ExtensionType;
```

Drugo polje je `extension_data` koje sadrži podatke čiji format ovisi o vrsti ekstenzije. Napomenimo da definicija `opaque extension_data<0..2^16-1>` označava da je `extension_data` polje čija veličina je zapisana u prva dva okteta ali ne uključujući ta dva okteta, a može biti veliko od 0 do 2^16-1 (65535) okteta - zbog toga su nužna dva okteta na početku kako bi se mogao zapisati broj veličine 65535.

DTLS-SRTP protokol definira novu vrstu DTLS ekstenzije koja se zove use-srtp ekstenzija [24]. S ovom ekstenzijom se dogovaraju parametri buduće SRTP sjednice.

Pogledajmo kako izgleda format use-srtp ekstenzije:

```
uint8 SRTPProtectionProfile[2];
struct {
      SRTPProtectionProfiles SRTPProtectionProfiles;
      uint8 srtp_mki<0..255>;
} UseSRTPExtension;
SRTPProtectionProfile SRTPProtectionProfiles<2^16-1>;
```

Parameteri SRTP sjednice, koji se dogovaraju s "use-srtp" ekstenzijom su: SRTP profil (`SRTPProtectionProfiles`) i MKI vrijednost (`srtp_mki`).

Mogući SRTP profili su:

- SRTP_AES128_CM_SHA1_80 `{0x00, 0x01}`
- SRTP_AES128_CM_SHA1_32 `{0x00, 0x02}`
- SRTP_AES256_CM_SHA1_80 `{0x00, 0x03}`
- SRTP_AES256_CM_SHA1_32 `{0x00, 0x04}`
- SRTP_NULL_SHA1_80 `{0x00, 0x05}`
- SRTP_NULL_SHA1_32 `{0x00, 0x06}`

Npr. SRTP_AES128_CM_SHA1_80 označava da će SRTP sjednica koristiti AES algoritam veličine ključa 128 bita u modu brojača za kriptiranje i SHA1 algoritam veličine ključa od 80 bita za autentifikaciju.

SRTP profil se dogovara između klijenta i poslužitelja na način da klijent prvo pošalje use-srtp ekstenziju u ClientHello poruci u kojoj se nalazi lista svih SRTP profila koje klijent podržava, a

poslužitelj odgovara sa svojom use-srtp ekstenzijom unutar ServerHello poruke, u kojoj odabire točno jedan od ponuđenih SRTP profila.

MKI vrijednost sadrži SRTP MasterKeyIdentifier vrijednost koju će klijent koristiti za SRTP poruke koje šalje. Ako je ova vrijednost jednaka 0 onda se MKI ne koristi.

4.2.2. Generiranje SRTP ključeva

Za generiranje SRTP ključeva i slučajnih vrijednosti za klijenta i poslužitelja koristi se TLS ekstraktor definiran u [26].

TLS ekstraktor na ulazu prima tri vrijednosti od protokola iz višeg sloja, tj. u našem slučaju DTLS-SRTP protokola:

- `label`
- `context_value`
- `length`

TLS ekstraktor zatim računa slijedeće:

```
PRF(master_secret, labela, client_random + server_random +
    context_value_length + context_value)[length]
```

gdje je `PRF` je pseudo-slučajna funkcija koja je definirana u [15], `master_secret` je glavna tajna razmijenjena tijekom DTLS dogovaranja, `client_random` i `server_random` se također razmijene tijekom DTLS dogovaranja. Izlaz je pseudo-slučajan niz bitova veličine `length` okteta.

DTLS-SRTP protokol za `label` definira vrijednost `EXTRACTOR-dtls_srtp`.

Vrijednost `length` je jednaka `2*(srtp_master_key_len+srtp_master_salt_len)`, gdje je `srtp_master_key_len` veličina SRTP glavnog ključa, a `srtp_master_salt_len` veličina slučajne vrijednosti, a te veličine ovise o izabranom SRTP profilu.

Vrijednost `context_value` se u protokolu DTLS-SRTP ne koristi.

Generiranih `length` pseudo-slučajnih okteta se preraspodjele na slijedeći način:

- prvih `master_key_len` okteta postaju glavni ključevi za SRTP promet koji šalje klijent
- slijedećih `master_key_len` okteta postaju glavni ključevi za SRTP promet koji šalje poslužitelj
- slijedećih `master_salt_len` okteta postaju slučajna vrijednost za SRTP promet koji šalje klijent
- slijedećih `master_salt_len` okteta postaju slučajna vrijednost za SRTP promet koji šalje poslužitelj

4.3. Razmjena certifikata

U DTLS-SRTP protokolu dvije krajnje točke prezentiraju svoj identitet kao dio DTLS dogovaranja koristeći certifikate. Ovaj protokol koristi certifikate na isti način kako je to opisano u [19].

Korištenje certifikata u krajnjim točkama potpisanih od strane poznatih *autoriteta za certifikate* (engl. Certificate Authority, skraćeno CA) je u praksi nepraktično, zbog toga DTLS-SRTP protokol definira način rada koji dopušta korištenje samo-potpisanih certifikata (engl. self-signed certificates) na siguran način ukoliko je integritet SDP protokola osiguran. To je postignuto na način da se kroz SDP protokol prenosi sigurnosni sažetak certifikata, tzv. *potpis certifikata* (engl. certificate fingerprint).

Pogledajmo kako izgleda SDP model ponuda/odgovor u slučaju kada se koristi DTLS-SRTP protokol unutar medijskog kanala (slika 4.1).

Krajnje točke moraju koristit `setup` attribut unutar SDP protokola. Krajnja točka koja je SDP ponuditelj mora koristiti `setup:actpass` vrijednost tog atributa, a krajnja točka koja je SDP odgovaratelj treba koristiti vrijednost `setup:active`. To znači da odgovaratelj treba poslati ClientHello poruku prema ponuditelju, što znači da je odgovaratelj DTLS klijent, a ponuditelj DTLS poslužitelj.

Krajnje točke ne smiju koristiti `connection` atribut koji je definiran u [19].

Krajnje točke moraju koristiti `fingerprint` atribut kako je to definirano u [19] za prenošenje sigurnosnog potpisa certfikat. Pogledajmo na primjeru kako izgleda SDP linija s `fingerprint` atributom kada se koristi SHA-1 funkcija sažimanja:

```
a=fingerprint:SHA-1 4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Tako preneseni sigurnosni potipis certifikata mora odgovorati certifikatu koji se kasnije prezentira unutar DTLS dogovaranja.

Pogledajmo na primjeru kako izgleda tipična SDP ponuda u slučaju korištenja DTLS-SRTP protokola (navedene su samo linije specifične za DTLS-SRTP):

```
m=audio 32504 UDP/TLS/RTP/SAVP 8 0
a=setup:passive
a=fingerprint:SHA-1 4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Primjetite da se koristi `UDP/TLS/RTP/SAVP` prijenosni protokol, što je upravo oznaka za korištenje protokola DTLS-SRTP preko UDP prijenosnog protokola [24].

4.4. Integritet otiska certifikata koji se prenosi unutar SDP-a

Integritet sigurnosnog otiska certifikata koji se prenosi unutar SDP-a tijekom uspostave poziva može biti osiguran koristeći neki od ovih mehanizam:

- "SIP Identity" [9] i "Connected SIP Identity" [10]
- "SIP Identity using Media Path" [29],
- "End-to-End Security for DTLS-SRTP" [30].

SIP Identity mehanizam služi za autentifikaciju identiteta krajnjeg korisnika koji inicira SIP poziv i zaštitu integriteta SIP poruke i SDP tijela koje je sadržano u toj poruci. Ovaj mehanizam pruža dvije vrste usluga: autentifikacijsku uslugu i verifikacijsku uslugu.

Autentifikacijska usluga se obično nalazi na strani SIP posredničkog poslužitelja i posjeduje privatni ključ za certifikat domene na kojoj se nalazi te ima pristup mogućnostima SIP poslužitelja za registriranje koje mu omogučavaju da autentificira korisnike u toj domeni. Autentifikacijska usluga autentificira krajnjeg korisnika, potipisuje neka zaglavlja SIP poruke (From, To, Call-Id, Cseq, Date i Contact) i čitavo SDP tijelo koje se nalazi u SIP poruci, sprema potpis u novo SIP zaglavlja pod imenom Identity te dodaje još jedno zaglavlje pod imenom Identity-Info koje sadrži URI lokaciju gdje se može dohvatiti certifikat poslužitelja koji implementira autentifikacijsku uslugu.

Verifikacijska usluga obavlja verifikaciju potpisa koji je stigao unutar Identity zaglavlja SIP poruke. Verifikacijska usluga prvo dobavlja certifikat poslužitelja koji je potpisao SIP poruku. Zatim određuje da li je taj poslužitelj autoritativan za SIP URI koji se nalazi u From zaglavlju tako što usporedi domenu From zaglavlja i domenu koja je zapisana u certifikatu te naposljetku verificira Identity zaglavlje. Verifikacijska usluga dobavlja certifikat autentifikacijskog poslužitelja koristeći URI lokaciju koja je zapisana u Identity-Info zaglavlju.

Uz "SIP Identity" mehanizam, postoji i njemu sličan mehanizam pod nazivom "Connected SIP Identity", koji služi za autentifikaciju već spojenog korisnika, npr. za autentifikaciju korisnika koji je odgovorio na poziv ili za autentifikaciju korisnika koji zamjenjuje neku od strana koje sudjeluju u pozivu.

Gore opisani mehanizmi ne rade u slučaju kada se SIP poruke modificiraju od strane nekih poslužitelja koji se nalaze na putu između dvije krajnje točke, tj. između točke u kojoj se obavlja autentifikacija i točke u kojoj se obavlja verifikacija. Ti poslužitelji mogu biti *kontrolori granice sjednice* (engl. Session Border Controller, skraćeno SBC) ili *dvostrani korisnički agenti* (engl. Back-to-Back User Agents, skraćeno B2BUA). Modificiranjem SIP poruke, odnosno SDP tijela te poruke ti poslužitelji razbijaju potpis koji se nalazi unutar Identity zaglavlja, upravo kako bi to napravio neki "čovjek u sredini" napadač.

Mehanizam "SIP Identity using Media Path" [29] pokušava riješiti ovaj problem na način da se potpisuju samo neke linije, kao npr. `a=fingerprint`, a ne čitavo SDP tijelo. Ali ni ovaj mehanizam nije rješenje budući da postoje poslužitelji koji modificiraju From polje koje se nalazi među podacima koji se potpisuju.

Konačno, mehanizam "End-to-End Security for DTLS-SRTP" [30] rješava gore navedene problem na način da se potpisuju samo originalno From zaglavlje kroz novo uvedeno zaglavlje SIP poruke `Fingerprint-Identity` i samo SDP linija `a=fingerprint` koja sadrži potpis certifikata koji se koristi unutar DTLS dogovaranja.

4.5. Analiza zahtjeva

Pogledajmo kako se DTLS-SRTP nosi sa zahtjevima definiranim u poglavlju 3.2 [23]:

- SDP ponuda koju šalje DTLS-SRTP ne ovisi o identitetu druge strane, pa prema tome grananje i preusmjeravanje rade dok god su sve strana spremne koristiti SRTP (R-FORK-

RETARGET). Ukoliko to nije tako, tj. ako su neke strane spremne koristiti SRTP a neke nisu, u tom slučaju DTLS-SRTP protokol koristi SDP-ov mehanizam za pregovaranje mogućnosti definiran u [28] kako bi transparentno dogovorio sigurnu komunikaciju tamo gdje je to moguće (R-BEST-SECURE). Budući da DTLS uspostavlja nove ključeve za svaku sjednicu, samo strana s kojom je poziv u konačnici uspostavljen dobija ključeve za buduću SRTP komunikaciju. (R-DISTICT)

- DTLS dopušta da sjedica bude nastavljena ako se koristi TLS-ova funkcionalnost za nastavljanje sjednice (R-REUSE).
- Budući da DTLS-SRTP uspostavlja ključeve unutar medijskog kanala to znači da ne dolazi do prekidanja medija prije nego što stigne SDP odgovor (R-AVOID-CLIPPING).
- Algoritmi s javnim ključevima koji se koriste u DTLS protokolu otporni su na pasivne napade unutar medijskog kanala (R-PASS-MEDIA). DTLS također pruža zaštitu i od pasivnih napada unutar signalnog kanala, budući da se samo potpis certifikata prenosi tim kanalom (R-PASS-SIG). Napadač koji kontrolira medijski kanal može izvesti napad tipa "čovjek u sredini" nad DTLS dogovaranjem; taj će napad promijeniti certifikate, što će uzrokovati da provjera potpisa certifikate ne uspije pa prema tome svaki uspješan napad zahtjeva i promjenu signalnih poruka kako bi se zamijenili potipisi certifikata (R-SIG-MEDIA). Napadač koji kontrolira signalni kanal u bilo kojoj točki na putu od posredničkog poslužitelja koji obavlja autentifikaciju identiteta do posredničkog poslužitelja u kojem se obavlja verifikacija *Identity* potpisa u zaglavlju ne može zamijeniti potpis certifikata a da ne poništi Identity potpis; prema tome čak i napadač koji može aktivno utjecati i na signalni i na medijski kanal ne može uspješno izvesti napad a da ne bude detektiran (R-ACT-ACT).
- DTLS-SRTP koristi mehanizme "SIP Identity" [9] i "Connected SIP Identity" [10] kako bi povezao potpise certifikata s *From* adresom u zaglavlju protokola SIP; potpis certifikata je uključen u *Identity* potpis (R-ID-BINDING).
- DTLS podržava kratkotrajnu Diffie-Hellman razmjenu ključeva koja nudi savršenu sigurnost prema unaprijed (R-FPS).
- DTLS dogovara SRTP profil prije nego što počne obavljati značajna kriptografska računanja, pa prema tome dopušta ponudu dodatnih SRTP profila bez potrebe za dodatnim računskim resursima (R-COMPUTE).
- DTLS paketi ne prolaze RTP test valjanosti jer prvi oktet DTLS paketa `ContentType` ima prva dva bita jednaka nuli za sve do sada definirane vrste sadržaja, pa prema tome pada na prvom testu valjanosti (R-RTP-VALID).
- DTLS-SRTP ne zahtjeva ali dopušta korištenje CA-certifikata ili dijeljenih ključeva ukoliko su oni kompatibilni s TLS-om (R-CERTS, R-EXISTING).
- DTLS-SRTP razmjena ključeva povezana je sa signalnim porukama pomoću potpisa certifikata koji se nalazi u SDP-u i certifikata koji se razmjene u protokolu DTLS (R-ASSOC).

- DTLS-SRTP metoda može se primjeniti na bilo koji signalni protokol koji je sposoban razmijeniti potpis certifikata i opis medija (R-OTHER-SIGNALING).
- Ekstenzija je predložena [31] koja omogućuje snimanje i transkodiranje na način da se posrednici ne ponašaju kao "čovjek u sredini" napadači (R-RECORDING, R-TRANSCODER).
- DTLS-SRTP dopušta da se sigurnost medija prekine na PSTN povezniku. U tom slučaju DTLS-SRTP ne pruža sigurnost s kraja na kraj, ali s obzirom da je poveznik autoriziran da "govori" u ime PSTN-a to je u skladu sa sigurnosnim ciljevima DTLS-SRTP protokola. (R-PSTN).
- Ovi zathjevi su podržani zbog opće poznatih svojstava DTLS protokola: R-FIPS, R-DOS, R-AGILITY, R-DOWNGRADE, R-NEGOTIATE.

5. Praktični rad

Cilj praktičnog rada je ostvariti DSRTP biblioteku funkcija za implementaciju DTLS-SRTP zaštite RTP prometa u već postojeće ili buduće aplikacije. Kao primjer integracije biblioteke u postojeću aplikaciju, ugraditi ćemo biblioteku u Ekiga aplikaciju, tj. OpalVoip biblioteka koju ona koristi.

5.1. DSRTP biblioteka

DSRTP biblioteka namijenjena je aplikacijama koje žele implementirati DTLS-SRTP metodu za zaštitu RTP prometa. Biblioteka implementira razmjenu ključeva pomoću DTLS protokola i implementira SRTP sigurnosni profil RTP protokola.

Pretpostavka je da aplikacija već ima implementiranu SIP/SDP signalizaciju i RTP prijenos. Moguće je koristiti i bilo koju drugu signalizaciju jer biblioteka ni na jedan način ne pretpostavlja korištenje SIP/SDP-a.

Biblioteka na jednostavan način omogućuje aplikacijama da postojeći RTP prijenos podataka nadograde korištenjem DTLS-SRTP protokola za zaštitu tog prijenosa. Biblioteka koristi metodu koja se na engl. naziva *"bump in the stack"* (BITS), tj. biblioteka se koristit kao podsloj između aplikacije i RTP prijenosa.

S obzirom da biblioteka ne radi sa signalizacijskim protokolima, tj. SIP/SDP-om, taj dio DTLS-SRTP metode prepušten je aplikaciji. Što npr. znači da aplikacija mora sama autentificirati potpise certifikata koji se razmijenjene unutar DTLS dogovaranja; naravno, aplikacija dolazi do potpisa pomoću funkcija biblioteke.

5.2. Korištenje DSRTP biblioteke (API)

Na početku izvršavanja, aplikacija prvo mora inicijalizirati biblioteku na način da se pozove funkcija `dsrtp_init` koja će kreirati tzv. globalni kontekst u kojem se spremaju sve globalne varijable koje su potrebne biblioteci u toku izvršavanja aplikacije. Pogledajmo na primjeru kako izgleda ta inicijalizacija:

```
1: #include <dsrtp.h>
   ...
2: dsrtp_global_ctx_handle ctx;
3: dsrtp_result_t res;
4: res = dsrtp_init(&ctx, "/home/mvladic/rtpsec/libdsrtp/test/data/");
5: if (res != DSRTP_RESULT_OK)
      exit(-1);
```

U gornjem primjeru možemo uočiti nekoliko detalja koji su zajednički cijelokupnom API-ju:

- API je definiran u zaglavlju `dsrtp.h`, stoga je na početku potrebno uključiti to zaglavlje.
- Imena funkcija i tipova započinju sa `dsrtp_`.

- Većina funkcija vraća rezultat tipa `dsrtp_result_t`. Taj rezultat nam govori da li je funkcija uspješno obavila zadatak ili nije. Ako je funkcija uspješno obavila zadatak vraća se `DSRTP_RESULT_OK`, a ako je došlo do greške vraća se jedna od `DSRTP_RESULT_ERR...` vrijednosti.
- Ako funkcija treba vratiti još neku vrijednost, onda se ta vrijednost vraća kroz neki od parametara tipa pointer. Npr. u gornjem primjeru funkcije `dsrtp_init` vraća *rukovatelj* (engl. handle) globalnog konteksta kroz prvi parameter koji je pointer na `dsrtp_global_ctx_handle`. Taj rukovatelj je potrebno proslijediti kao parametar nekim drugim funkcijama iz API-ja.

Biblioteka se u svom radu oslanja na neke funkcije koje moraju biti implementirane unutar aplikacije. Pogledajmo koje su to funkcije:

- `dsrtp_log`
  Biblioteka poziva ovu funkciju kada treba logirati neku informaciju. Datoteka `rtpsec/libdsrtp/test_common.c` sadrži primjer implementacije te funkcije.
- `dsrtp_session_send_dgram`
  Biblioteka poziva ovu funkciju kada treba poslati datagram koji sadrži poruke DTLS-ovog protokola dogovaranja. Datagram se šalje u kontekstu DTLS-SRTP sjednice i kao prvi parametar ova funkcija prima rukovatelj sjednice unutar koje se treba poslati datagram.
- `dsrtp_session_event_callback`
  Biblioteka poziva ovu funkciju kada se desi neki važan događaj unutar DTLS-SRTP sjednice kao što je primjerice trenutak kada je konekcija postala sigurna - tada je stanje sjednice jednako `DSRTP_SESSION_STATE_SECURE`.

Slijedeći korak je kreiranje DTLS-SRTP sjednice, a to se radi pozivom funkcije `dsrtp_session_create`, kojoj je na ulazu potrebno predati informaciju o tome da li se radi o klijentskoj strani (`DSRTP_CONNECTION_END_CLIENT`) ili o poslužiteljskoj strani DTLS konekcije (`DSRTP_CONNECTION_END_SERVER`). Funkcija `dsrtp_session_create` na ulazu prima i ime datoteke koja sadrži certifikat koji će se koristiti u DTLS dogovaranju. Potpis tog certifikata, kako to zahtjeva DTLS-SRTP protokol, potrebno je poslati drugoj strani pomoću signalnog protokol (SIP/SDP). Za dohvaćanje potpisa aplikacija poziva funkciju `dsrtp_get_my_cert_fingerprint`, a za samo slanje potpisa signalnim protokolom zadužena je aplikacija.

DTLS dogovaranje pokreće se pozivom funkcije `dsrtp_session_start`. U tom trenutku aplikacija mora imati spreman mehanizam za slanje datagrama mrežom, jer će biblioteka početi pozivati funkciju `dsrtp_session_send_dgram`.

Kada je dogovaranje gotovo aplikacija može početi slati i primati SRTP pakete, ali prije toga je potrebno provjeriti potpis certifikata koji je razmijenjen DTLS dogovaranjem s onim koji je razmijenjen pomoću signalnog protokola. Potpis koji je razmijenjen DTLS dogovaranjem dohvaća se pomoću funkcije `dsrtp_get_peer_cert_fingerprint`. Za dohvaćanje potpisa koji je razmijenjen pomoću signalnog protokola zadužena je aplikacija.

Kada želi poslati SRTP paket, aplikacija prvo mora pripremljeni RTP paket pozivom funkcije `dsrtp_session_process_rtp` pretvoriti u SRTP paket koji je onda spreman za slanje mrežom. Za slanje RTCP paketa koristi se funkcija `dsrtp_session_process_rtcp`.

Primljeni SRTP paket aplikacija pretvara u RTP paket pozivom funkcije `dsrtp_session_process_srtp`. Za SRTCP pakete aplikacija koristi funkciju `dsrtp_session_process_srtcp`.

U bilo kojem trenutku aplikacija može doznati u kojem je stanju sjednica pozivom funkcije `dsrtp_session_get_state`.

Aplikacija uništava DTLS-SRTP sjednicu pozivom funkcije `dsrtp_session_destroy`.

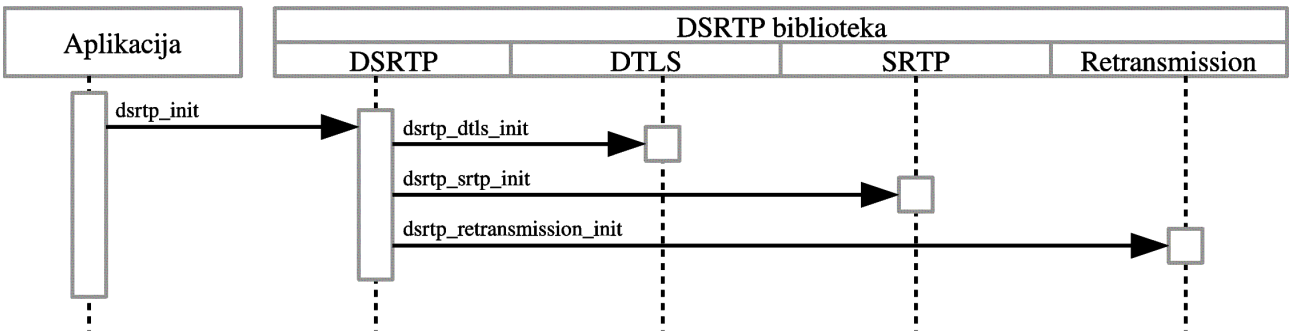
Na kraju, aplikacija mora osloboditi globalni kontekst koji je kreiran pozivom funkcije `dsrtp_init`, a to se obavlja pozivom funkcije `dsrtp_down`.

5.3. Dizajn DSRTP biblioteke

DSRTP biblioteka je podijeljena na slijedeće komponente:

- DSRTP je glavna komponenta koja implementira cjelokupni javni API na način da zadatke delegira prema ostalim komponentama.
- DTLS komponenta je zadužena za DTLS protokol.
- SRTP komponenta je zadužena za SRTP protokol.
- Retransmission komponenta je zadužena za retransmisiju paketa za DTLS protokol.
- System komponenta je apstrakcijski sloj prema uslugama operativnog sustava. Postojanje takvog sloja olakšava prebacivanja biblioteke na različite operativne sustave.
- Util komponenta sadrži pomoćne funkcije za rad s listama, logiranje, itd.

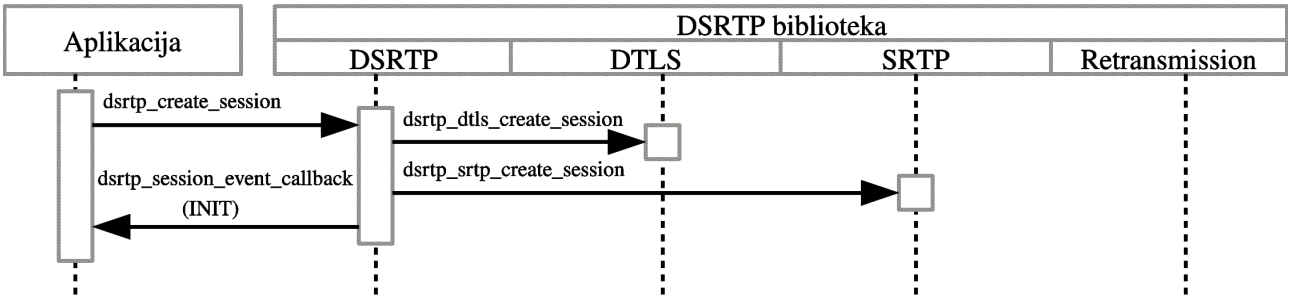
Inicijalizacija biblioteke i globalnog konteksta (slika 5.1) obavlja se u funkciji `dsrtp_init` koja je dio DSRTP komponetne i javnog API-ja. Globalni kontekst sadrži sve globalne varijable bitne za rad biblioteke, a koje ne ovise o pojedinoj DTLS-SRTP sjednici. Globalni kontekst se drži u `dsrtp_global_ctx_t` strukturi.



Slika 5.1: Inicijalizacija DSRTP biblioteke – sekvencijski dijagram

DSRTP biblioteka omogućuje aplikacijama da kreiraju proizvoljan broj DTLS-SRTP sjednica, i to po jednu za svaku RTP i RTCP sjednicu koja se otvori unutar aplikacije (vidi poglavlje 4.2 za detalje kada treba kreirati novu DTLS-SRTP sjednicu).

DTLS-SRTP sjednica kreira se pozivom funkcije `dsrtp_create_session` (slika 5.2). Budući da se postupak DTLS dogovaranja razlikuje ovisno o tome da li je riječ o klijentskoj ili poslužiteljskoj strani, prilikom kreiranja DTLS-SRTP sjednice aplikacija mora naznačiti o kojoj se strani radi. Svi podaci o DTLS-SRTP sjednici čuvaju se u `dsrtp_session_t` strukturi.

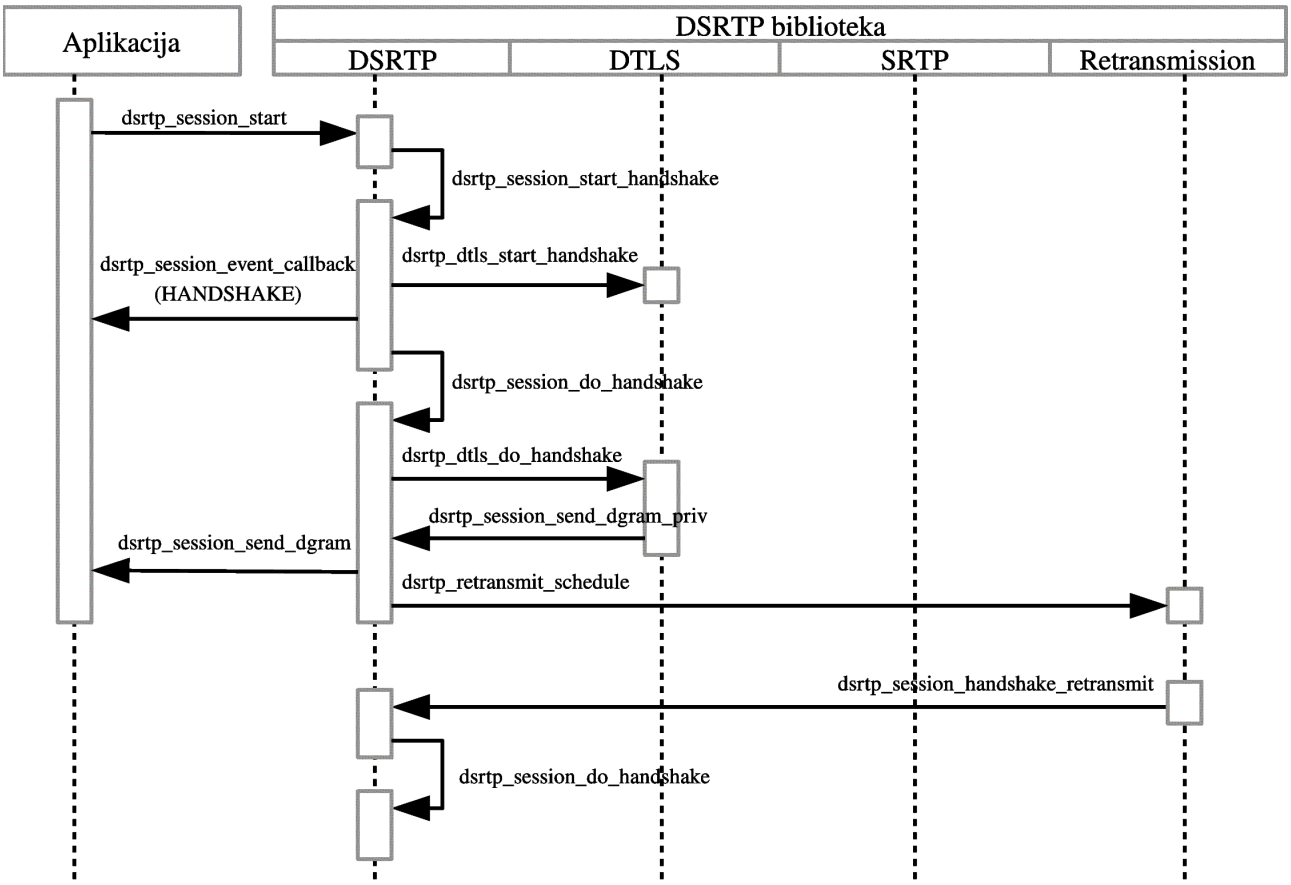


Slika 5.2: Kreiranje DTLS-SRTP sjednice - sekvencijski dijagram

DTLS dogovaranje, unutar DTLS-SRTP sjednice, pokreće se pozivom funkcije `dsrtp_session_start` (slika 5.3). Zadaća DTLS dogovaranja delegira se prema DTLS komponenti, koja za postupak dogovaranja definira dvije funkcije: `dsrtp_dtls_start_handshake` i `dsrtp_dtls_do_handshake`. Postupak dogovaranja u DTLS komponenti započinje pozivom funkcije `dsrtp_dtls_start_handshake`.

Funkciju `dsrtp_dtls_do_handshake` potrebno je pozivati jednom na početku, što radi funkcija `dsrtp_dtls_start_handshake`, i svaki put kad imamo novi datagram na ulazu (slika 5.4), a to se desi kada aplikacija pozove funkciju `dsrtp_session_process_srtp` kada stigne novi datagram s DTLS porukom dogovaranja. Podsjetimo se da su poruke DTLS-a multipleksirane zajedno sa SRTP porukama na istom UDP pristupu.

Funkcija `dsrtp_dtls_do_handshake` poziva funkciju `dsrtp_session_send_dgram` svaki put kada želi poslati DTLS datagram na izlaz koristeći UDP ili neki drugi prijenosni protokol. Funkcija `dsrtp_session_send_dgram` je implementirana u aplikaciji.

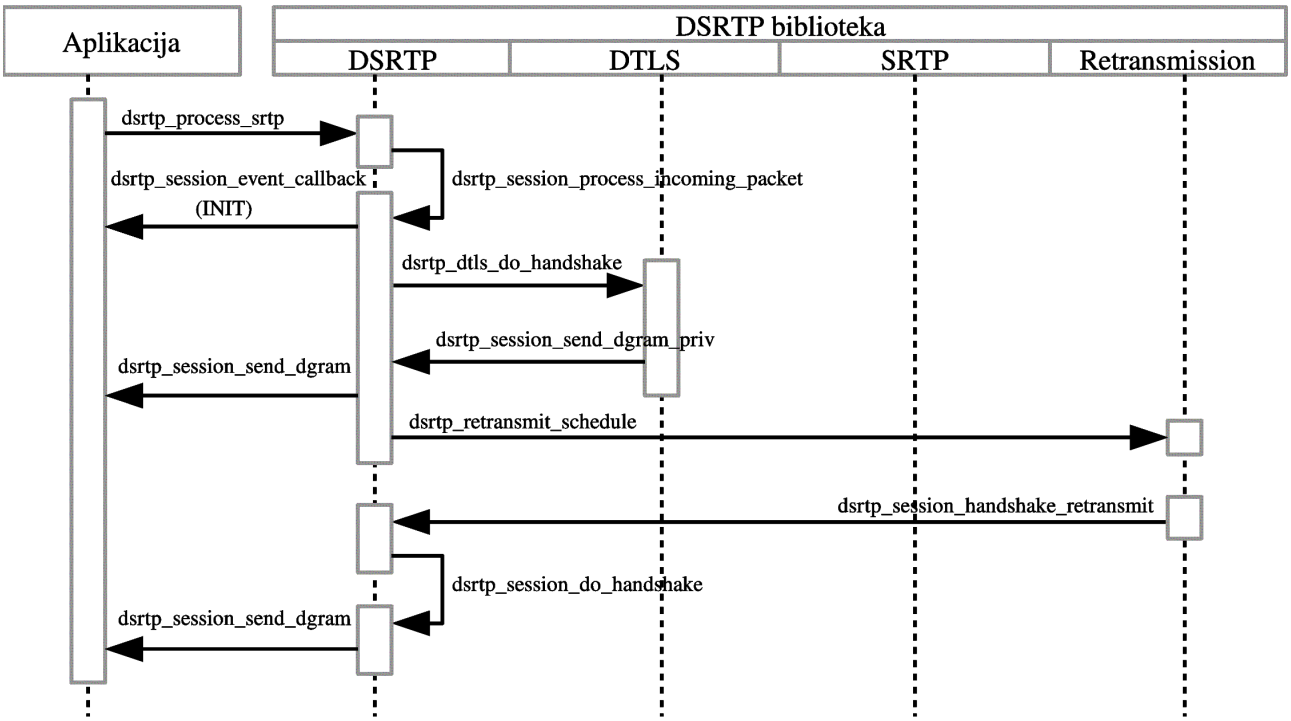


Slika 5.3: Pokretanje DTLS dogovaranje unutar DTLS-SRTP sjednice - sekvencijski dijagram

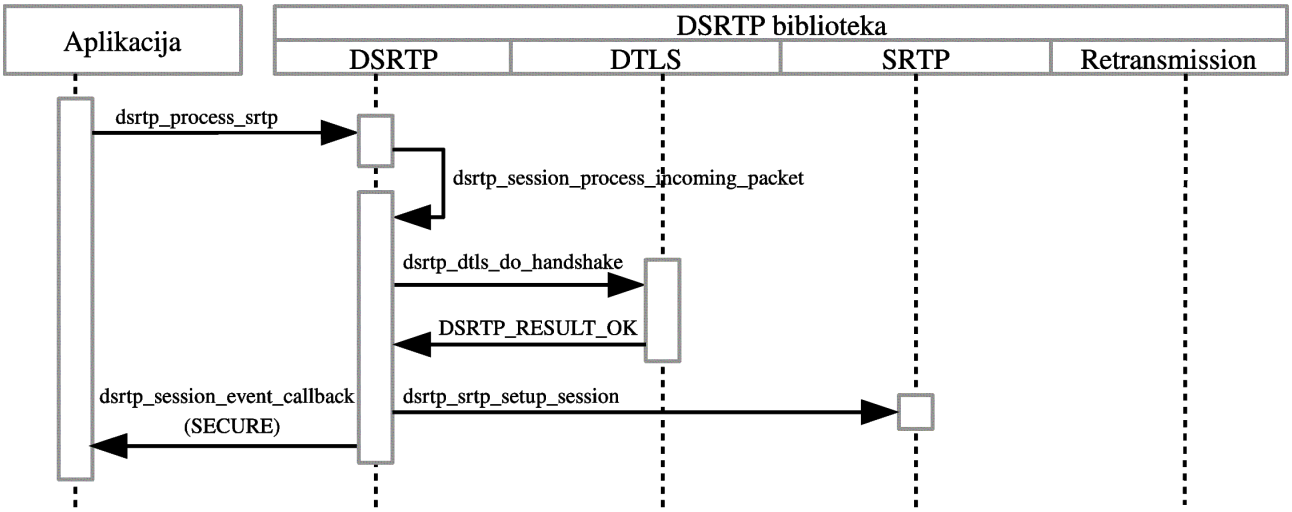
Funkcija `dsrtp_dtls_do_handshake` vraća rezultat `DSRTP_RESULT_ERR_WANT_READ` kada je za nastavak dogovaranja potrebno da stigne novi paket od druge strane.

DSRTP komponenta pozivom funkcije `dsrtp_retransmit_schedule` nalaže komponenti za retransmisiju da periodično poziva funkciju `dsrtp_session_handshake_retransmit`. Ta funkcija poziva `dsrtp_dtls_do_handshake`, koja budući da nije stigao novi paket od druge strane ponovo šalje sve datagrame iz prethodnog leta, kako to nalaže DTLS protokol, pretpostavljajući da se neki od prethodnih datagrama izgubio.

DTLS dogovaranje je gotovo kada funkcija `dsrtp_dtls_do_handshake` vrati rezultat `DSRTP_RESULT_OK` (slika 5.5). Nakon toga se uspostavlja SRTP sjednica pozivom funkcije `dtls_srtp_setup_session` koja je implementirana unutar SRTP komponente. Biblioteka obavještava aplikaciju da je dogovaranje završilo i da je spremna procesirati SRTP pakete tako što pozove funkciju `dsrtp_session_event_callback` koja je implementirana na strani aplikacije.



Slika 5.4: Nastavak DTLS dogovaranja kada stigne novi DTLS paket na ulazu - sekvencijski dijagram



Slika 5.5: Završetak DTLS dogovaranja i uspostavljanje SRTP sjednice - sekvencijski dijagram

5.4. Implementacija DSRTP biblioteke

DSRTP biblioteka je ostvarena korištenjem programskog jezika C na Linux operativnom sustavu. Zbog postojanja sistemskog sloja prebacivanje biblioteke na druge operativne sustave bi trebalo biti relativno jednostavno.

DTLS komponenta je ostvarena korištenjem biblioteke otvorenog kôda **OpenSSL**, a SRTP komponenta je ostvarena korištenjem biblioteke otvorenog kôda **libSRTP**.

5.5. OpenSSL biblioteka

OpenSSL biblioteka sadrži podršku za SSL, TLS i DTLS protokole. Osim toga, unutar biblioteke su ostvarene osnovne kriptografske funkcije i različite pomoćne funkcije koje se mogu samostalno koristiti.

Osim same biblioteke kao dio iste distribucije postoji i openssl komandno-linijski program za izvršavanje velikog broja kriptografskih operacija, kao npr. generiranje certfikata i slično.

DTLS protokol je relativno nedavno ostvaren unutar OpenSSL biblioteke, pa je dostupna dokumentacija vrlo ograničena što je predstavljalo određeni problem prilikom korištenja unutar DSRTP biblioteke.

5.5.1. Modifikacija OpenSSL-a

Prvi korak je bio ostvariti "use-srtp" DTLS ekstenziju za dogovaranje SRTP sigurnosnih parametera (vidi poglavlje 4.2.1) budući da OpenSSL biblioteka ne podržava tu ekstenziju. To je napravljeno tako što je modificiran sam DTLS kôd OpenSSL biblioteke i dodane su određene funkcije u javni API OpenSSL biblioteke.

Modifikacija OpenSSL biblioteke u svrhu ostvarenja "use-srtp" ekstenzije sastoji se od ovih dijelova:

1. Generiranje okvira "use-srtp" ekstenzije koji sadrži listu SRTP profila koje podržava klijent i slanje generiranog okvira na kraju ClientHello poruke. Lista SRTP profila zadaje se pomoću funkcije `SSL_CTX_set_srtp_protection_profiles_data` koja je dodana u javni API OpenSSL biblioteke. Ovo je ostvareno unutar funkcije `dtls1_client_hello`.
2. Parsiranje na strani poslužitelja ClientHello poruke koja sadrži "use-srtp" ekstenziju i odabir SRTP profila koji je ponudio klijent a koji se nalazi u listi SRTP profila koje podržava poslužitelj. Lista SRTP profila zadaje se pomoću funkcije `SSL_CTX_set_srtp_protection_profiles_data` koja je dodana u javni API OpenSSL biblioteke. Ovo je ostvareno unutar funkcije `ssl_parse_clienthello_tlsext`.
3. Generiranje okvira "use-srtp" ekstenzije koji sadrži odabrani SRTP profil i slanje tog okvira na kraju ServerHello poruke. Ovo je ostvareno unutar funkcije `dtls1_send_server_hello`.
4. Parsiranje na strani klijenta ServerHello poruke koja sadrži "use-srtp" ekstenziju s odabranim SRTP profilom. Ostvareno unutar funkcije `ssl_parse_clienthello_tlsext`.

Odabrani SRTP profil može se dohvatiti pomoću funkcije `SSL_get_srtp_protection_profile` koja je dodana u javni API OpenSSL biblioteke.

Budući da libsrtp biblioteka ne podržava MKI unutar "use-srtp" ekstenzije ostvareno je samo dogovoranje SRTP sigurnosnog profila.

Osim ove "use-srtp" modifikacije, napravljena je još jedna manja modifikacija OpenSSL biblioteke. Podsjetimo se, za generiranje SRTP ključeva iz glavne tajne koristi se pseudo-slučajna funkcija (poglavlje 4.2.2). Implementacija te funkcije nalazi se unutar OpenSSL-a i zove se `tls1_PRF`, ali nažalost ta funkcija nije dio javnog API-ja. Tako da je dodana funkciju `TLS_PRF` u javni API (datoteka `ssl.h`), a implementacija te funkcije poziva funkciju `tls1_PRF`:

```
void TLS_PRF(const EVP_MD *md5, const EVP_MD *sha1,
             unsigned char *label, int label_len,
             const unsigned char *sec, int slen, unsigned char *out1,
             unsigned char *out2, int olen)
{
    tls1_PRF(md5, sha1, label, label_len, sec, slen, out1, out2, olen);
}
```

5.5.2. OpenSSL dogovaranje

Opišimo kako izgleda korištenje OpenSSL-a za DTLS dogovaranje.

OpenSSL koristi isti API bez obzira da li je u pitanju SSL, TLS ili DTLS protokol, samo je razlika u parameterima koje se predaju funkcijama.

Na početku moramo kreirati DTLS kontekst koristeći funkciju SSL_CTX_new. DTLS kontekst služi za definiranje svih parametara koje ću buduća DTLS konekcija koristiti.

Kod kreiranja DTLS konteksta potrebno je odabrati verziju protokola. Verzija protokola odabire se kreiranjem SSL_METHOD objekta i predajom tog objekta funkciji za kreiranje SSL konteksta. U našem slučaju kreira se SSL_METHOD objekt koristeći funkciju `DTLSv1_client_method` kada je u pitanju DTLS klijent i funkciju `DTLSv1_setver_method` kada je u pitanju DTLS poslužitelj.

Nakon što smo kreirali i postavili SSL kontekst, slijedeći korak je kreiranje DTLS konekcije s funkcijom `SSL_new`, zatim zadavanje BIO objekta koji će ta DTLS konekcija koristiti za sve ulazno/izlazne operacije, tj. za primanje i slanje datagrama. I na kraju moramo odrediti da li je riječ o klijentskoj ili poslužiteljskoj strani DTLS konekciju; to se radi pozivom funkcije `SSL_set_connect_state` ako je u pitanju DTLS klijent, tj. pozivom funkcije `SSL_set_accept_state` ako je u pitanje DTLS poslužitelj.

Sada smo spremni pokrenuti DTLS dogovaranje. Za to služi funkcija `SSL_do_handshake` koja može raditi u dva načina: blokirajući i neblokirajući.

U blokirajućem načinu radu, funkcija blokira pozivni thread sve dok dogovaranje ne završi, bilo uspješno ili neuspješno. Za retransmisiju poruka, tj. odabiranje trenutka kada se ponovo šalju datagrami iz prethodnog leta, brine se OpenSSL.

U neblokirajućem načinu rada, funkcija obavlja DTLS dogovaranje sve dok se desi da nema više ulaznih datagrama. Kada se to desi, funkcija prekida izvršavanje i obavještava pozivni program da ne može nastaviti dogovaranje sve dok se ne pojavi novi datagram na ulazu. Ukoliko se ponovo pozove `SSL_do_handshake`, a još uvijek nema ulaznog datagrama, funkcija će obaviti retransmisiju, tj. ponovo će poslati sve datagrame iz prethodnog leta. U ovom načinu rada, pozivni program je zadužen za odabir trenutka kada se obavlja retransmisija.

Da li će dogovaranje biti blokirajuće ili neblokirajuće ovisi o BIO objektu. BIO objekt koji koristi DSRTP biblioteka, a koji je ostvaren unutar DSRTP biblioteke, je takav da je odabrano neblokirajuće dogovaranje. Za čitanje ulaznih datagrama, `SSL_do_handshake` poziva funkciju `bio_read`. Ako nema novog paketa funkcija `bio_read` vrati `-1`. Tada se pozove funkcija `bio_ctrl(BIO_CTRL_DGRAM_GET_RECV_TIMER_EXP)` i ako je povratna vrijednosti 1 radi se retransmisija, a ako je povratna vrijednost `0` izlazi se iz `SSL_do_handshake` funkcije te se vraća kontrola pozivnom programu do pojave novog paketa. Za slanje izlaznih datagrama `SSL_do_handshake` poziva funkciju `bio_write`. U DSRTP biblioteci, `bio_read` poziva funkciju `dsrtp_session_get_dgram` koja provjeri da li postoji novi paket za pripadnu DTLS-SRTP konekciju - podsjetimo se da se novi ulazni DTLS paket aplikacija predaje DSRTP biblioteci pozivom funkcije `dsrtp_session_process_srtp`. U DSRTP biblioteci, `bio_write` poziva funkciju `dsrtp_session_send_dgram_priv` koja pak pozove funkciju `dsrtp_session_send_dgram` koju implementira aplikacija, te se tako slanje izlaznih DTLS paketa delegira prema aplikaciji.

5.6. LibSRTP biblioteka

Za ostvarenje potrebne funkcionalnosti SRTP komponenta koristi libSRTP biblioteku.

Nakon što je DTLS dogovaranje završilo, DSRTP komponenta uspostavlja SRTP sjednicu pozivom funkcije `dsrtp_srtp_setup_session`. SRTP sjednica se sastoji od dvije libSRTP sjednice, jedna za odlazne pakete i jedna za dolazne pakete; tj. klijent ima dvije libSRTP sjednice i poslužitelj ima dvije libSRTP sjednice. Odlazna sjednica klijenta koristi isti ključ kako dolazna sjednica poslužitelja; dolazna sjednica klijenta koristi isti ključ kao odlazna sjednica poslužitelja.

Za kreiranje libSRTP sjednice koristi se funkciju srtp_create. Ali prije nego što pozovemo tu funkciju potrebno je definirati SRTP sigurnosni profil, glavni ključ, slučajnu vrijednost i SSRC vrijednost.

Sigurnosni profil definiran je tijekom DTLS dogovaranja koristeći "use-srtp" ekstenziju. SRTP komponenta ga dohvaća pozivom funkcije `dsrtp_dtls_get_srtp_protection_profile` koja je ostvatena unutar DTLS komponente. Obje libSRTP sjednice koriste isti SRTP profil.

Glavni ključevi i slučajne vrijednosti za obje sjednice dobivaju se pozivom funkcije `dsrtp_dtls_generate_keying_material`. Ta funkcija je ostvarena unutar DTLS komponente, a koristi se metoda koja je opisana u poglavlju 4.2.2. Ovdje se koristi funkcija `TLS_PRF` koja je dodana u javni API od OpenSSL biblioteke.

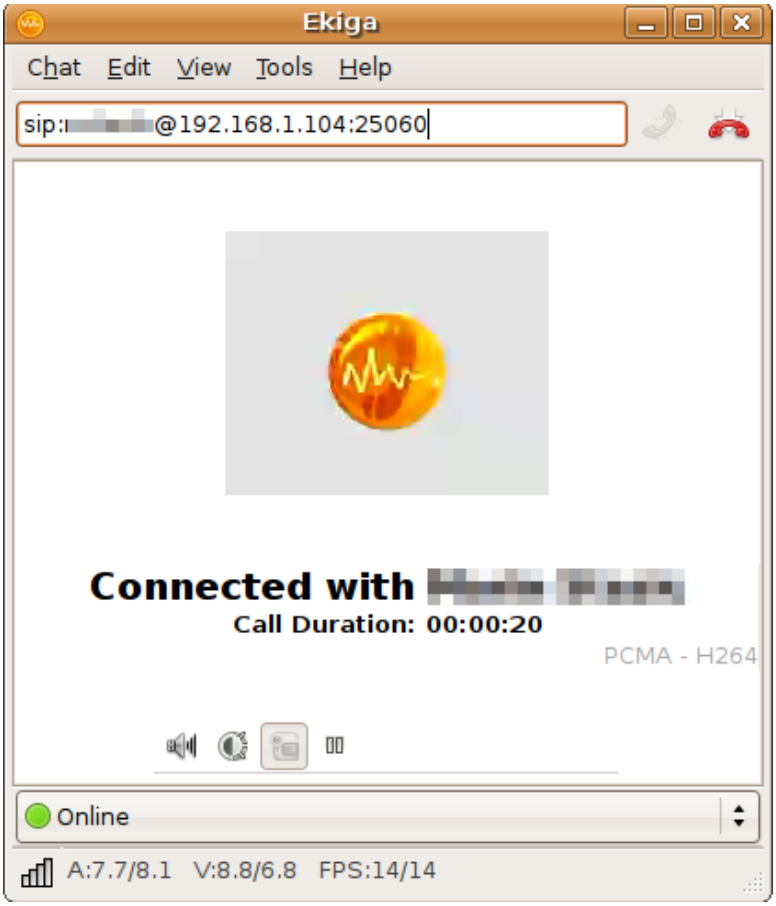
SSRC vrijednost može biti postavljena na neki konkretan broj ukoliko je on poznat, ili može biti naznačeno da se prihvaća bilo koji SSRC tako što se SSCR vrijednost postavi na `ssrc_any_inbound` za dolaznu sjednicu, tj. `ssrc_any_outbound` za odlaznu sjednicu.

Kada smo kreirali libSRTP sjednice možemo početi štititi RTP/RTCP promet. Za kriptiranje i autentifikaciju odlaznih RTP/RTCP paketa, tj. pretvaranje u SRTP/SRTCP pakete spremne za slanje mrežom, koristi se funkcija srtp_protect, odnosno srtp_protect_rtcp. Za verifikaciju i dekriptiranje

dolaznih SRTP/SRTCP paketa, tj. pretvaranje u RTP/RTCP pakete, koristi se funkcija srtp_unprotect, odnosno srtp_unprotect_rtcp.

6. Integracija DSRTP biblioteke u Ekigu i OpalVoip

Ekiga (bivši GnomeMeeting) je SIP i H.323 sukladna, korisnička aplikacija za VoIP, IP telefoniju i video konferencije [http://www.gnomemeeting.org/]. Ekiga je besplatna aplikacija otvorenog kôda i radi pod GNOME grafičkim okružjem (slika 6.1).



Slika 6.1: Ekiga kada je SIP poziv u tijeku

Ekiga je u osnovi GUI ljuska oko OpalVoip sustava. OpalVoip je C++ biblioteka klasa za razvoj aplikacija koje koristite SIP i H.323 protokole za multimedijalnu komunikaciju preko paketno zasnovanih mreža [http://www.opalvoip.org].

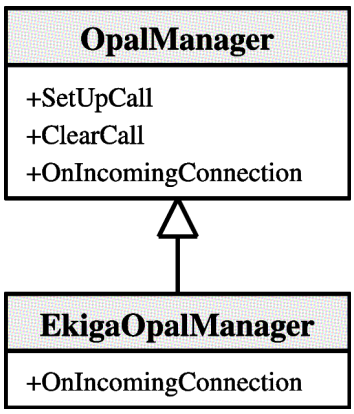
Budući da Ekiga koristi OpalVoip za SIP, integracija DSRTP biblioteke u Ekigu smještena je upravo unutar OpalVoip sustava.

Integracija je izvedena nad Ekigom 2.9.0 i Opalom 3.3, što su u trenutku izrade ovog rada najnovije razvojne inačice tih programskih sustava.

6.1. Osnove OpalVoip biblioteke

OpalVoip je napisan u programskom jeziku C++ i koristi objektno-orijentiranu programsku paradigmu.

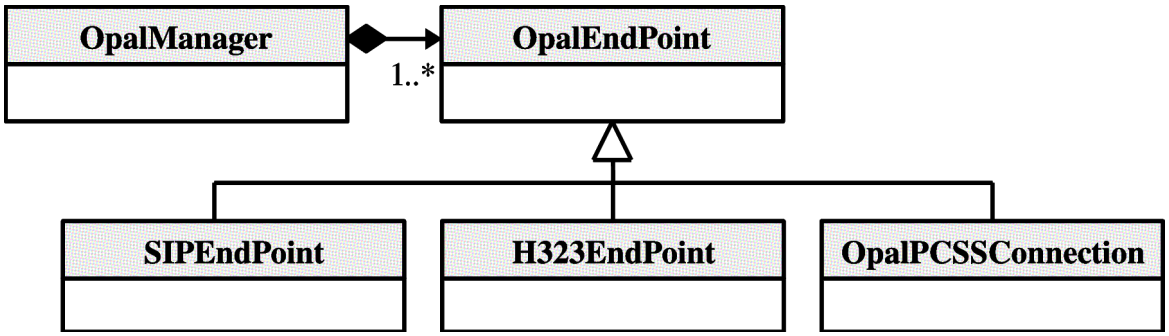
Centralna klasa OpalVoip biblioteke je `OpalManager`. Očekuje se da aplikacija definira svoju klasu koja je izvedena iz klase `OpalManager` (slika 6.2) i kreira jednu instancu (objekt) iz te klase. Također, aplikacija treba preusmjeriti određene funkcije iz klase OpalManager koje OpalVoip poziva kada se dogodi određeni događaj na koji bi aplikacija trebala reagirati. Primjerice, kada se pojavi novi dolazni poziv, OpalVoip poziva funkciju `OnIncomingConnection`, a aplikacija u toj funkciji mora odlučiti što želi, prihvatiti poziv ili ne i o tome obavjestiti OpalVoip.



Slika 6.2: Aplikacija (Ekiga) definira klasu EkigaOpalManager koja je izvedena iz klase OpalManager i preusmjerava funkciju OnIncomingConnection

Aplikacija poziva funkcije `OpalManager` klase primjerice kada želi uspostaviti novi odlazni poziv (`SetUpCall`) ili kada želi poništiti neki aktivni poziv (`ClearCall`).

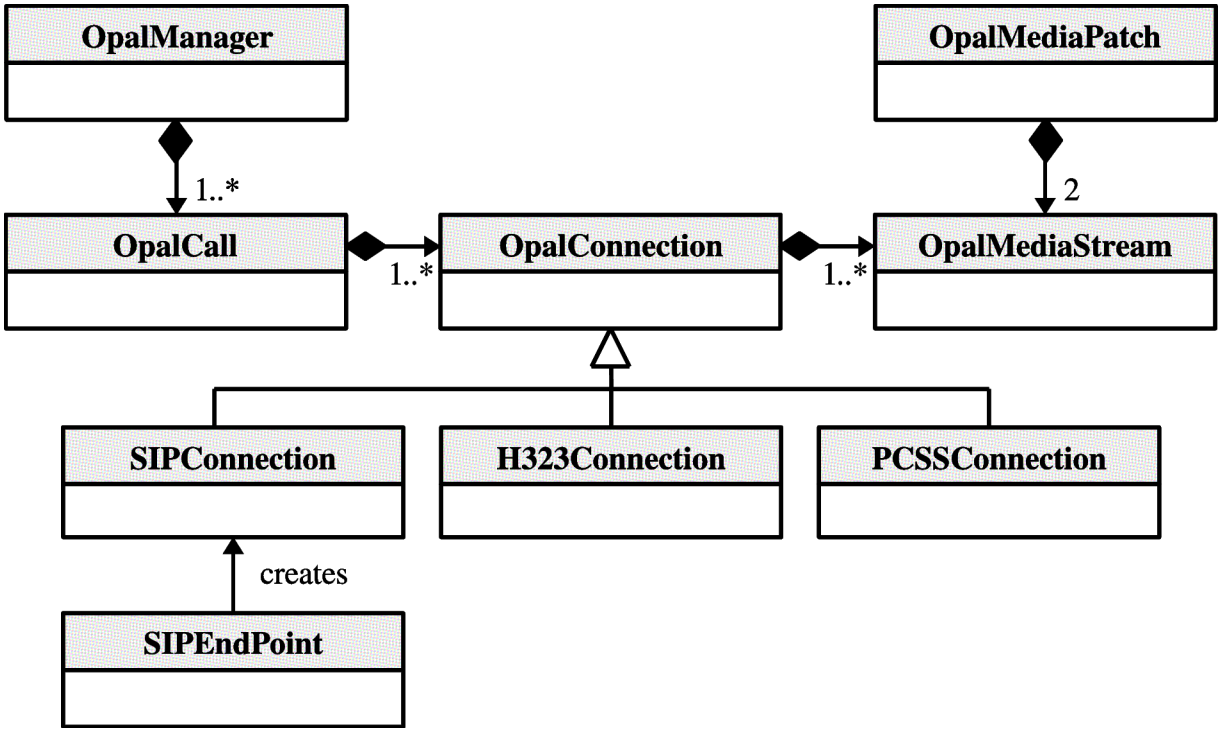
Kako bi mogla primati dolazne i uspostavljati odlazne pozive za neki od standardnih protokola (SIP, H.323), aplikacija prvo mora kreirati upravljačke objekte za te protokole i dodati ih u svoj `OpalManager` objekt. Upravljački objekti su instance klase `OpalEndPoint` i to za svaki protokol postoji specifična klasa koja je izvedena iz klase `OpalEndPoint` (slika 6.3). Primjerice, za protokol SIP definirana je klasa `SIPEndPoint`.



Slika 6.3: Svaki protokol definira svoju upravljačku klasu koja je izvedena iz `OpalEndPoint`

Klasa `OpalConnection` je osnovna klasa za konekcije u krajnjim točkama (slika 6.4). Svaki protokol definira svoju klasu izvedenu iz `OpalConnection`, koja pokriva specifičnu semantiku

konekcije za taj protokol. Tako postoji SIPConnection klasa za konekciju u SIP ili prema SIP krajnjoj točki, zatim H323Connection za H323 krajnje točke, ili PCSSConnection klasa koja predstavlja konekciju prema zvučnom sustavu osobnog računala (engl. PC Sound System, PCSS skraćeno).



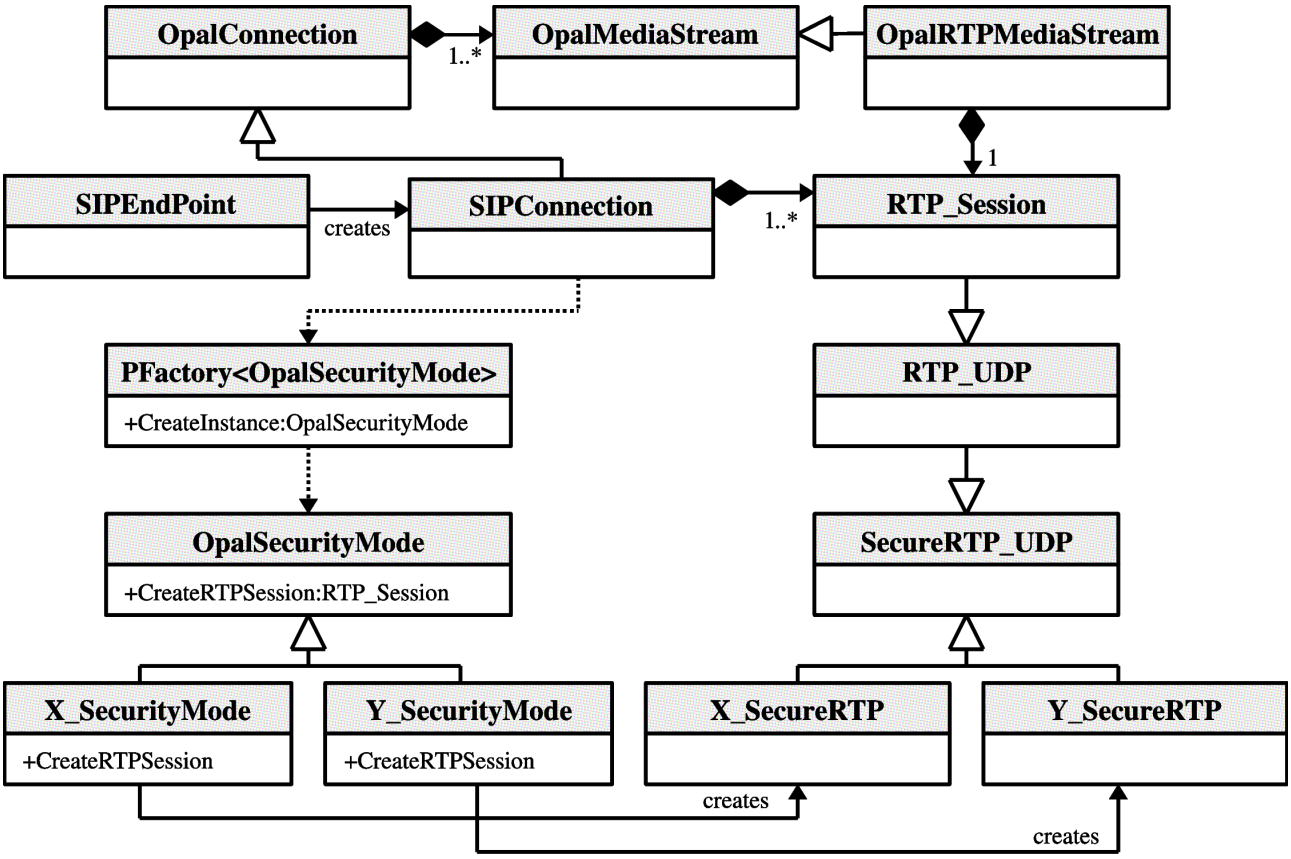
Slika 6.4: Dijagram klasa OpalVoip biblioteke

Klasa `OpalCall` upravlja pozivom. Poziv se sastoji od jednog ili više `OpalConnection` objekta. Primjerice, u slučaju tipičnog SIP poziva OpalCall upravlja s jednim `SIPConnection` objektom i s jednim `PCSSConnection` objektom. Ako `OpalCall` upravlja s jednom SIP konekcijom i jednom H. 323 konekcijom, tada on radi kao *poveznik* (engl. gateway) između ta dva protokola. U slučaju konferencije, jedan `OpalCall` objekt može upravljati s puno `OpalConnection` objekata.

`OpalConnection` objekti nastaju u krajnjim točkama, tj. u `OpalEndPoint` objektima, ali su "vlasništvo" `OpalCall` objekta.

Konekcija je zadužena za kreiranje `OpalMediaStream` objekata koji predstavljaju kanale za razmjenu podataka između dva OpalVoip entiteta. Svaki `OpalMediaStream` objekt povezan je s drugim `OpalMediaStream` objektom preko `OpalMediaPatch` objekta.

Pogledajmo sada kako stvari izgledaju unutar SIP podsustava (slika 6.5). Već smo spomenuli da postoji SIPEndPoint koji kreira SIPConnection. SIPConnection kreira jednu ili više RTP sjednicu za svaki medij (audio i video) prisutan u konekciji. RTP sjednica je predstavljena s `RTP_Session` klasom, koja je pridružena `OpalRTPMediaStream` klasi koja je izvedena iz `OpealMediaStream` klase – tako da su podaci koji se razmijenjuju unutar RTP sjednice dostupni ostatku OpalVoip sustava.



Slika 6.5: Dijagram klasa SIP podsustava OpalVoip biblioteke

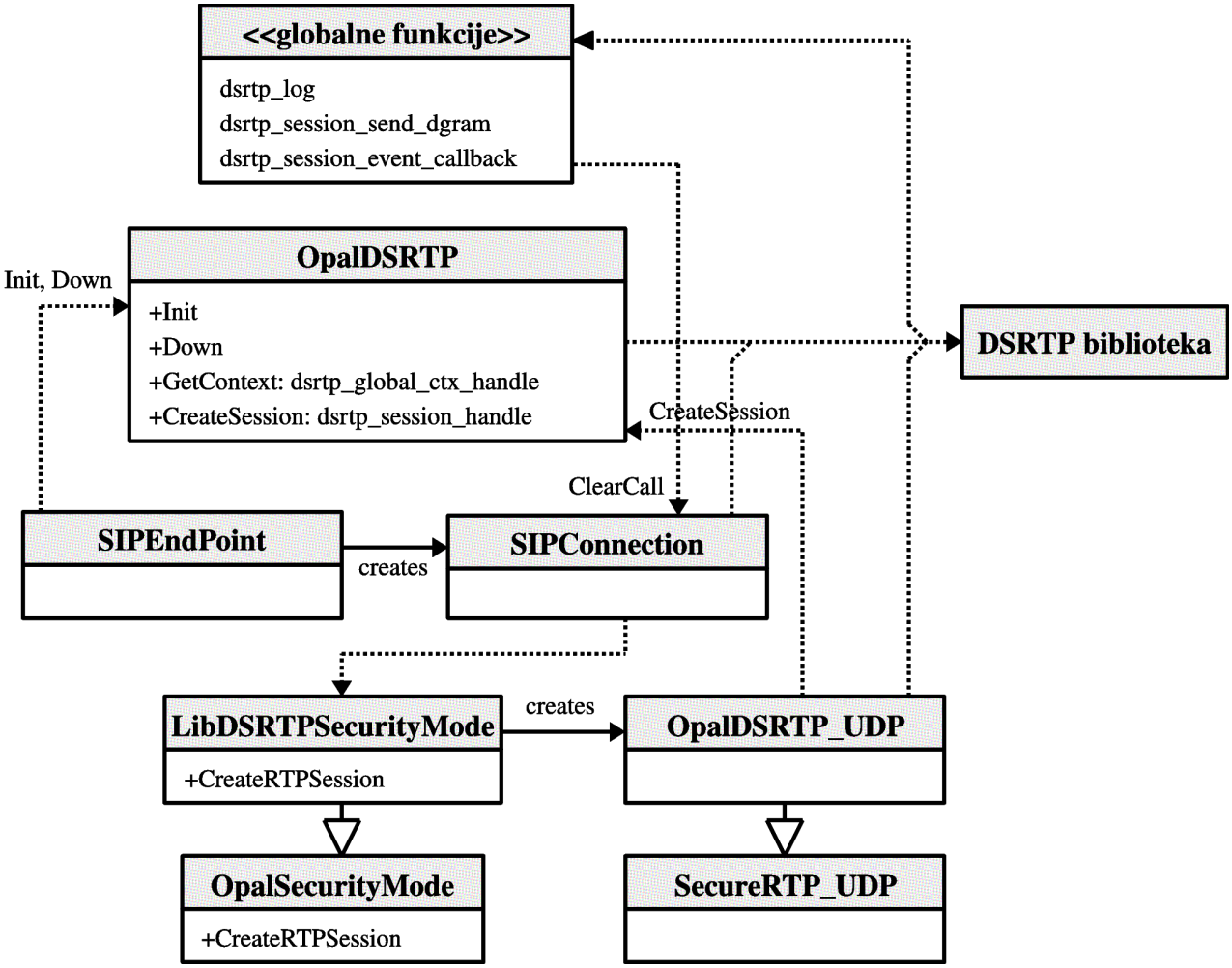
`SIPConnection` nikad ne kreira RTP sjednički objekt direktno iz `RTP_Session` klase, nego kreira ili iz `RTP_UDP` klase koja je izvedena iz RTP_Session klase ili iz neke od klasa koje su izvedene iz `SecureRTP_UDP` klase koja je pak izvedena iz `RTP_UDP` klase. U ovom potonjem slučaju, `SIPConnection` za kreiranje `SecureRTP_UDP` objekta koristi usluge pripadnog `OpalSecurityMode` objekta, kojeg pronalazi iz danog imena sigurnosnog moda preko `Pfactory<OpalSecurityMode>` klase. Pogledajmo kako to izgleda u kôdu:

```
RTP_UDP *rtpSession = NULL;
if (!securityMode.IsEmpty())
{
      OpalSecurityMode *sm =
            Pfactory<OpalSecurityMode>::CreateInstance(securityMode);
      rtpSession = sm->CreateRTPSession(...);
}
else
{
      rtpSession = new RTP_UDP(...);
}
```

SIPConnection koristi usluge klase `SDPSessionDescription` klase za generiranje i parsiranje SDP poruka. `SDPSessionDescription` za svaki medij prisutan u konekciji ima pridružen `SDPMediaDescription` objekt.

6.2. Integracija DSRTP biblioteke u OpalVoip

Pogledajmo kako je DSRTP biblioteke integrirana u OpalVoip i koje nove klase i globalne funkcije su dodane (slika 6.6).



Slika 6.6: Integracija DSRTP biblioteke u OpalVoip

Dodana je nova klasa `OpalDSRTP` koja služi kao C++ sučelje prema globalnom kontekstu DSRTP biblioteke (`dsrtp_global_ctx_handle`). Ova klasa sadrži statičke funkcije za kreiranje i uništavanje konteksta (`Init` i `DeInit`), dohvaćanje kreiranog konteksta (`GetContext`) i pomoćnu funkciju za kreiranje DSRTP sjednice (`CreateSession`).

Klasa `OpalDSRTP` dodana je u OPAL kroz dvije nove datoteke: `include/dsrtp/opaldsrtp.h` u kojoj se nalazi deklaracija klase i `src/dsrtp/opaldsrtp.cxx` u kojoj se nalazi definicija klase.

U istim datotekama ostvarena je funkcija `dsrtp_log` za DSRTP logiranje, zatim funkcija `dsrtp_session_send_dgram` za slanje DTLS datagrama i funkcija `dsrtp_session_event_callback` koja prima informacije od strane DSRTP biblioteke o važnim promjenama unutar DSRTP sjednice. Funkcija `dsrtp_session_event_callback` prekida poziv ako je došlo do greške prilikom dogovaranja tako da pozove `SIPConnection::ClearCall`.

DSRTP biblioteka se inicijalizira i uništava unutar `SIPEndPoint` klase.

Dodana je klasa `OpalSecurityMode` i šest klasa izvedenih iz te klase, jedna za svaki SRTP profil koji je podržan od strane DSRTP biblioteke. Tih šest klasa, u funkciji `CreateRTPSession`, kreiraju objekt koji je instanca klase `OpalDSRTP_UDP` tako da u konstruktoru predaju odabrani SRTP profil. Klase su dodana u OPAL kroz dvije nove datoteke: `include/rtp/dsrtpudp.h` u kojoj se nalaze deklaracije klasa i `src/rtp/dsrtpudp.cxx` u kojoj se nalaze definicije klasa.

Klasa `OpalDSRTP_UDP`, u svom konstruktoru, poziva funkciju `OpalDSRTP::CreateSession` (koja pak poziva funkciju `dsrtp_create_session` iz DSRTP biblioteke) dva puta – jednom za RTP sjednicu, a drugi put za RTCP sjednicu; u destruktoru se zove `dsrtp_destroy_session` direktno. Unutar iste klase, direktno se pozivaju još neke funkcije iz DSRTP biblioteke, primjerice funkcija `dsrtp_session_process_srtp` se poziva iz funkcije `OpalDSRTP_UDP::OnReceiveData` kada treba primljeni SRTP paket proslijediti prema DSRTP biblioteki na daljnju obradu.

Napravljene su i određene izmjene unutar klase `SIPConnection`. Primjerice, u funkciji `OnSendSDP`, ubačeni su `"a:setup"` i `"a:fingerprint"` atributi u SDP ponudu i SDP odgovor korištenjem `SDPSessionDescription` objekta, a u funkciji `OfferSDPMediaDescription`, gdje se kreira `SDPMediaDescription`, dodana je podrška za DTLS-SRTP biranjem `"UDP/TLS/RTP/SAVP"` prijenosa unutar `"m="` linije SDP ponude. Zbog ovih izmjena bilo je potrebno izmjeniti i klase `SDPSessionDescription` i `SDPMediaDescription` kako bi mogle prihvatiti ove izmjene unutar SDP-a.

6.3. Ispitivanje

Obavljena je uspostava zaštićenog poziva između dvije Ekiga na lokalnoj mreži, gdje je SIP sjednica uspostavljena direktno bez korištenja posredničkog poslužitelja. Ispitivanje je obavljeno uspješno i poziv je bio zaštićen. Korišten je program Wireshark za praćenje mrežnog prometa, u našem slučaju to su DTLS, SIP, SDP i RTP poruke. Taj program vrlo uspješno može snimiti nezaštićeni RTP promet, ali u našem slučaju, kada je poziv bio zaštićen, to naravno nije bilo moguće.

7. Zaključak

Osnovni cilj ovog rada, a to je izrada biblioteke u programskom jeziku C za DTLS-SRTP metodu je uspješno ostvaren. Takva biblioteka se može relativno jednostavno integrirati u već postojeće aplikacija, kao što ja to na primjeru Ekige u ovom radu i pokazano. Upotrebom biblioteke implementirane u ovom radu ostvaren je zaštićen, direktan SIP poziv između dvije Ekige na lokalnoj mreži.

Pokazano je da implementacija DTLS-SRTP protokola nije složena zbog toga što se za razmjenu ključeva koristi postojeći protokol DTLS za koji postoji implementacija otvorenog kôda, a koja se u ovom radu koristi.

Ono što nedostaje protokolu DTLS-SRTP, a nalazi se primjerice u protokolu ZRTP, je SAS autentifikacija. Takva autentifikacija bi predstavljala dodatnu sigurnost za one korisnike koji ne vjeruju davateljima SIP usluge koji implementiraju autentifikacijsku uslugu, a koja je nužna u protokolu DTLS-SRTP kako bi se detektirao napad tipa "čovjek u sredini".

Slijedeći korak bi svakako bio daljnje ispitivanje na složenijim scenarijima. Primjerice, uspostava poziva preko jednog ili više SIP posredničkih poslužitelja, uspostava poziva kada se Ekiga nalazi iza NAT poslužitelja, uspostava poziva koji će prouzročiti grananje ili preusmjeravanje, uspostava poziva sa PSTN uređajem. Također, trebalo bi ispitati rad u prisustvu autentifikacijskog SIP posredničkog poslužitelja. Vjerojatno bi se nakon ovakvih proširenih ispitivanja otkrili određeni nedostaci u ostvarenoj biblioteci i Ekiga integraciji.

8. Literatura

[1] Henry Sinnreich and Alan B. Johnston, *"Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol"*, Wiley Computer Publishing, 2001.

[2] Colin Perkins, *"RTP: Audio and Video for the Internet"*, Addison-Wesley, 2003.

[3] John Viega, Matt Messier and Pravir Chandra, *"Network Security with OpenSSL"*, O'Reilly, 2002.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, *"SIP: Session Initiation Protocol"*, IETF, June 2002, http://www.ietf.org/rfc/rfc3261.txt

[5] M. Handley, V. Jacobson and C. Perkins, *"SDP: Session Description Protocol"*, IETF, July 2006, http://www.faqs.org/rfcs/rfc4566.html

[6] J. Rosenberg, H. Schulzrinne, *"An Offer/Answer Model with the Session Description Protocol (SDP)"*, IETF, June 2002, http://www.ietf.org/rfc/rfc3264.txt

[7] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, *"RTP: A Transport Protocol for Real-Time Applications"*, IETF, July 2003, http://www.ietf.org/rfc/rfc3550.txt

[8] M. Baugher, D. McGrew, M. Naslund, E. Carrara and K. Norrman: *"The Secure Real-time Transport Protocol (SRTP)"*, IETF, March 2004, http://www.ietf.org/rfc/rfc3711.txt

[9] J. Peterson and C. Jennings, *"Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)"*, IETF, August 2006, http://www.ietf.org/rfc/rfc4474.txt

[10] J. Elwell, *"Connected Identity in the Session Initiation Protocol (SIP)"*, IETF, June 2007, http://www.ietf.org/rfc/rfc4916.txt

[11] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, *"MIKEY: Multimedia Internet KEYing"*, IETF, August 2004, http://www.ietf.org/rfc/rfc3830.txt

[12] D. Ignjatic, L. Dondeti, F. Audet, P. Lin, *"MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)"*, IETF, November 2006, http://www.ietf.org/rfc/rfc4738.txt

[13] M. Euchner, *"HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY)"*, IETF, September 2006, http://www.ietf.org/rfc/rfc4650.txt

[14] F. Andreasen, M. Baugher and D. Wing, *"Session Description Protocol (SDP) Security Descriptions for Media Streams"*, IETF, July 2006, http://www.ietf.org/rfc/rfc4568.txt

[15] T. Dierks and E. Rescorla, *"The Transport Layer Security (TLS) Protocol Version 1.1"*, IETF, April 2006, http://www.ietf.org/rfc/rfc4346.txt

[16] E. Rescorla and N. Modadugu, *"Datagram Transport Layer Security"*, IETF, April 2006, http://www.ietf.org/rfc/rfc4347.txt

[17] D. Wing, *"Symmetric RTP / RTP Control Protocol (RTCP)"*, IETF, July 2007, http://www.ietf.org/rfc/rfc4961.txt

[18] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen and T. Wright, *"Transport Layer Security (TLS) Extensions"*, IETF, June 2003, http://www.ietf.org/rfc/rfc3546.txt

[19] J. Lennox, *"Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)"*, IETF, July 2006, http://www.ietf.org/rfc/rfc4572.txt

[20] J. Rosenberg and H. Schulzrinne, *"Reliability of Provisional Responses in the Session Initiation Protocol (SIP)"*, IETF, June 2002, http://www.ietf.org/rfc/rfc3262.txt

[21] F. Andreasen and D. Wing, *"Security Preconditions for Session Description Protocol (SDP) Media Streams"*, IETF, October 2007, http://www.ietf.org/rfc/rfc5027.txt

[22] R. Shirey, *"Internet Security Glossary, Version 2"*, IETF, August 2007, http://www.ietf.org/rfc/rfc4949.txt

[23] J. Fischl, H. Tschofenig and E. Rescorla, *"Framework for Establishing an SRTP Security Context using DTLS"*, February 23, 2008, http://www.ietf.org/internet-drafts/draft-ietf-sip-dtls-srtp-framework-01.txt

[24] D. McGrew and E. Rescorla, *"Datagram Transport Layer Security (DTLS) Extension to Establish Keys for Secure Real-time Transport Protocol (SRTP)"*, February 25, 2008, http://www.ietf.org/internet-drafts/draft-ietf-avt-dtls-srtp-02.txt

[25] C. Perkins and M. Westerlund, *"Multiplexing RTP Data and Control Packets on a Single Port"*, August 1, 2007, http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-and-rtcp-mux-07.txt

[26] E. Rescorla, *"Keying Material Extractors for Transport Layer Security (TLS)"*, February 20, 2008, http://www.ietf.org/internet-drafts/draft-ietf-tls-extractor-01.txt

[27] J. Rosenberg, *"Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols"*, October 29, 2007, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-19.txt

[28] F. Andreasen, *"SDP Capability Negotiation"*, December 11, 2007, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdp-capability-negotiation-08.txt

[29] D. Wing and H. Kaplan, *"SIP Identity using Media Path"*, February 23, 2008, http://www.ietf.org/internet-drafts/draft-wing-sip-identity-media-02.txt

[30] K. Fischer, *"End-to-End Security for DTLS-SRTP"*, January 22, 2008, http://www.ietf.org/internet-drafts/draft-fischer-sip-e2e-sec-media-00.txt

[31] D. Wing, F. Audet, S. Fries, H. Tschofenig and A. Johnston, *"Secure Media Recording and Transcoding with the Session Initiation Protocol"*, February 24, 2008, http://www.ietf.org/internet-drafts/draft-wing-sipping-srtp-key-03.txt

[32] P. Zimmermann, A. Johnston, Ed. and J. Callas, *"ZRTP: Media Path Key Agreement for Secure RTP"*, March 10, 2008, http://www.ietf.org/internet-drafts/draft-zimmermann-avt-zrtp-06.txt

[33] D. Wing, S. Fries, H. Tschofenig and F. Audet, *"Requirements and Analysis of Media Security Management Protocols"*, March 20, 2008, http://www.ietf.org/internet-drafts/draft-ietf-sip-media-security-requirements-04.txt

Dodatak A – Funkcije DSRTP biblioteke

dsrtp_init

Ova funkcija obavlja inicijalizaciju DSRTP biblioteke i potrebno ju je pozvati barem jednom na početku izvršavanja programa, tj. prije pozivanja bilo koje druge funkcije iz API-ja. Fukcija kreira globalni kontekst u kojem se spremaju sve globalne varijable koje su potrebne biblioteci u toku izvršavanja programa, i vraća rukovatelj (engl. handle) globalnog konteksta kroz parametar `p_hctx`. Rukovatelj globalnog konteksta predaje se drugim funkcijama iz API-ja (npr. `dsrtp_create_session`). Funkciju `dsrtp_init` moguće je pozvati više puta i pritom se svaki put kreira potpuno novi globalni kontekst koji nema nikakve veze s nekim prethodno kreiranim globalnim kontekstom. Višestruko pozivanje ove funkcije nalazi svoju primjenu u programima koji imaju dvije ili više nezavisnih komponenti koje koriste dsrtp biblioteku kao dijeljenu biblioteka (engl. shared library).

Deklaracija

```
dsrtp_result_t dsrtp_init(dsrtp_global_ctx_handle *p_hctx,
      const char* data_dir);
```

Parametri

p_hctx

Pointer na `dsrtp_global_ctx_handle` u koji se sprema tzv. rukovatelj globalnim kontekstom. Ovaj rukovatelj se predaju drugim funkcijama iz API-ja kojima je potreban globalni kontekst.

data_dir

Ime direktorija u kojem se nalaze datoteke s podacima koje su potrebne biblioteci u toku rada. U ovom direktoriju moraju se nalaziti slijedeće datoteke u kojim su spremljeni Diffie-Hellman parametri odgovarajuće veličine (u budućim verzijama biblioteke možda će se ovdje trebati nalaziti i neke druge datoteke):

- dh512.pem
- dh1024.pem
- dh2048.pem
- dh4096.pem

Obično se ove datoteke kreiraju u toku instalacije programa koristeći `openssl` komandno-linijski alat. Primjer pozivanja tog alata izgleda ovako:

```
% openssl dhparam -check -text -5 512 -out dh512.pem
```

Povratna vrijednost

Ova funkcija u slučaju uspjeha vraća `DSRTP_RESULT_OK`, a u slučaju greške jednu od slijedećih vrijednosti: `DSRTP_RESULT_ERR_ALLOC`, `DSRTP_RESULT_ERR_LIBSRTP_INIT`, `DSRTP_RESULT_ERR_INIT_OPENSSL` ili `DSRTP_RESULT_ERR`.

dsrtp_down

Ova funkcije obavlja uništavanje globalnog konteksta dsrtp biblioteke. Potrebno ju je pozvati točno jednom za svaki kreirani globalni kontekst kada se on više ne namjerava koristiti.

Deklaracija

```
void dsrtp_down(dsrtp_global_ctx_handle hctx);
```

Parametri

`hctx`
: Rukovatelj globalnog konteksta koji se želi uništititi.

Povratna vrijednost

Ova funkcija nema povratnu vrijednost.

dsrtp_session_create

Ova funkcija služi za kreiranje DSRTP sjednice.

Deklaracija

```
dsrtp_result_t dsrtp_session_create(dsrtp_global_ctx_handle hctx,
    dsrtp_session_params_t* params, dsrtp_session_handle *p_hsession);
```

Parametri

`hctx`
: Globalni kontekst DSRTP biblioteke unutar kojeg se DSRTP sjednica kreira.

`params`
: Svi parameteri kreiranja sjedice predaju se kroz ovaj jedan parametera koji je strukturnog tipa, a riječ je o strukturi `dsrtp_session_params_t` koja se sastoji od ovih polja:

- `dsrtp_connection_end_t connection_end`: određuje na kojoj je strani sjednica, na strani klijenta (vrijednost `DSRTP_CONNECTION_END_CLIENT`) ili na strani poslužitelja (vrijednost `DSRTP_CONNECTION_END_SERVER`).
- `void* user_data`: korisnički parametar definiran od strane aplikacije koji aplikacija pridružuje ovoj sjednici.
- `const char* cert_file`: sadrži ime datoteke u kojoj se nalazi certifikat.
- `dsrtp_uint64_t retransmission_timer`: vrijeme nakon kojeg se obavlja retransmisija unutar DTLS dogovaranja, ako je nula, onda je aplikacija zadužena za retransmisiju, pa mora pozivati funkciju `dsrtp_session_handshake_retransmit` eksplicitno.
- `int mtu`: iznos veličine MTU za DTLS pakete.

- `int *srtp_protection_profiles_data`: polje SRTP profila koje će ova DSRTP sjednica ponuditi unutar DTLS dogovaranja ako se radi o klijent strani. A ako se radi o poslužiteljskoj strani, onda su to SRTP profili koje poslužitelj odabire iz ponuđenih profila.
- `unsigned int srtp_protection_profiles_data_length`: veličina polja SRTP profila.

p_hsession

Pointer na `dsrtp_session_handle` u koji se sprema rukovatelj DSRTP sjednicom. Ovaj rukovatelj se predaju drugim sjedničkim funkcijama iz API-ja.

Povratna vrijednost

Funkcija vraća `DSRTP_RESULT_OK` ako je sjednica uspješno kreirana, a u slučaju greške jednu od slijedećih vrijednosti: `DSRTP_RESULT_ERR_ALLOC`, `DSRTP_RESULT_ERR_LOADING_CERT_FILE`, `DSRTP_RESULT_ERR_LOADING_PRIVATE_KEY`, `DSRTP_RESULT_ERR_CHECK_PRIVATE_KEY` ili `DSRTP_RESULT_ERR`.

dsrtp_session_destroy

Ova funkcija uništava DSRTP sjednicu.

Deklaracija

```
void dsrtp_session_destroy(dsrtp_session_handle hsession);
```

Parametri

hsession

Rukovatelj DSRTP sjednice koja se želi uništititi.

Povratna vrijednost

Ova funkcija nema povratnu vrijednost.

dsrtp_session_get_state

Ova funkcija vraća stanje u kojem se DSRTP sjednica nalazi.

Deklaracija

```
dsrtp_session_state_t dsrtp_session_get_state(dsrtp_session_handle hsession);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

Povratna vrijednost

Jedno od stanja definiranih u `dsrtp_session_state_t` enumeracijskom tipu:

- *DSRTP_SESSION_STATE_INIT*: sjednica je kreirana ali još nije pokrenuto DTLS dogovaranje.

- *DSRTP_SESSION_STATE_HANDSHAKE*: DTLS dogovaranje je pokrenuto i još traje.
- *DSRTP_SESSION_STATE_HANDSHAKE_ERR*: došlo je do greške prilikom DTLS dogovaranja i sjednica se treba uništiti.
- *DSRTP_SESSION_STATE_SECURE*: DTLS dogovaranje je završilo i sjednica je spremna za korištenje, tj. obradu dolaznih i odlaznih paketa.

dsrtp_session_get_connection_end

Ova funkcija vraća stranu konekcije na kojoj se DSRTP sjednica nalazi (klijent ili poslužitelj).

Deklaracija

```
dsrtp_connection_end_t dsrtp_session_get_connection_end(
      dsrtp_session_handle hsession);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

Povratna vrijednost

Jedna od dvije strane koje su definirane u `dsrtp_connection_end_t` enumeracijskom tipu:

- *DSRTP_CONNECTION_END_CLIENT*: sjednica je na strani klijenta.
- *DSRTP_CONNECTION_END_SERVER*: sjednica je na strani poslužitelja.

dsrtp_get_user_data

Ova funkcija vraća korisnički podatak, koji je pridružen DSRTP sjednici tijekom njenog kreiranja.

Deklaracija

```
void* dsrtp_session_get_user_data(dsrtp_session_handle hsession);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

Povratna vrijednost

Korisnički parametar.

dsrtp_session_start

Ova funkcija služi za pokretanje dogovaranja unutar DSRTP sjednice.

Deklaracija

```
dsrtp_result_t dsrtp_session_start(dsrtp_session_handle hsession);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

Povratna vrijednost

Funkcija vraća `DSRTP_RESULT_OK` ako je DTLS dogovaranje uspješno pokrenuto ili `DSRTP_RESULT_ERR` ako je došlo do greške.

dsrtp_session_handshake_retransmit

Ovu funkciju poziva aplikacija kada želi obaviti retransmisija zadnjeg leta poruka DTLS dogovaranja. Retransmisija se inače obavlja unutar same biblioteke, ali ako aplikacija tako želi može prilikom kreiranja DSRTP sjednice odlučiti da sama obavlja retransmisiju tako da postavi retransmisijsko vrijeme na 0.

Deklaracija

```
dsrtp_result_t dsrtp_session_handshake_retransmit(
      dsrtp_session_handle hsession);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

Povratna vrijednost

Funkcija vraća `DSRTP_RESULT_OK` ako je retransmisija uspješno obavljena ili `DSRTP_RESULT_ERR` ako je došlo do greške.

dsrtp_session_process_srtp

Aplikacija poziva ovu funkciju kako bi dolazni SRTP paket pretvorila u dekriptirani RTP. Dok traje DTLS dogovaranje, dolazni paket je vjerojatno DTLS paket. U tom slučaju funkcija vraća `DSRTP_RESULT_DROP`, pa aplikacija može ignorirati taj paket i prestati s daljnjom obradom.

Deklaracija

```
dsrtp_result_t dsrtp_session_process_srtp(dsrtp_session_handle hsession,
    char* data, int* size);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

data

Međuspremnik u kojem se nalazi ulazni SRTP paket. U isti međuspremnik sprema se dekriptirani RTP paket.

size

Veličina ulaznog međuspremnika. Nakon obrade, sadrži veličinu dekriptiranog RTP paketa.

Povratna vrijednost

Funkcija vraća `DSRTP_RESULT_OK` ako je paket uspješno dekriptiran i spreman za daljnju obradu od strane aplikacije, `DSRTP_RESULT_DROP` ako nije riječ o SRTP paketu pa aplikacija može prestati s daljnjom obradom ili `DSRTP_RESULT_ERR_DECRYPT` ako je došlo do greške prilikom dekriptiranja.

dsrtp_session_process_rtp

Aplikacija poziva ovu funkciju kako bi odlazni RTP paket zaštitila i pretvorila u SRTP. DSRTP sjednica mora biti u stanje `DSRTP_SESSION_STATE_SECURE` kako bi se obrada mogla obaviti .

Deklaracija

```
dsrtp_result_t dsrtp_session_process_rtp(dsrtp_session_handle hsession,
      char* data, int* size);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

data

Međuspremnik u kojem se nalazi ulazni RTP paket. U isti međuspremnik sprema se zaštićeni SRTP paket.

size

Veličina ulaznog međuspremnika. Nakon obrade, sadrži veličinu SRTP paketa.

Povratna vrijednost

Funkcija vraća `DSRTP_RESULT_OK` ako je paket uspješno zaštićen i spreman za daljnju obradu od strane aplikacije, `DSRTP_RESULT_ERR_SESSION_NOT_SECURE` ako sjednica nije u *`DSRTP_SESSION_STATE_SECURE`* stanju ili `DSRTP_RESULT_ERR_ENCRYPT` ako je došlo do greške prilikom kriptiranja.

dsrtp_session_process_srtcp

Ova funkcija je identična funkciji dsrtp_process_srtp, samo što ju aplikacija poziva kada treba dekriptirati ulazni SRTCP paket.

dsrtp_session_process_rtcp

Ova funkcija je identična funkciji dsrtp_process_rtp, samo što ju aplikacija poziva kada treba kriptirati odlazni RTCP paket.

dsrtp_get_my_cert_fingerprint

Ova funkcija vraća potpis certifikata koji je predan prilikom kreiranja DSRTP sjednice.

Deklaracija

```
dsrtp_result_t dsrtp_get_my_cert_fingerprint(dsrtp_session_handle hsession,
      const char *digest, unsigned char *fingerprint,
      unsigned int *fingerprint_size);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

digest

Naziv algoritma za izračun sažetka koji se koristi prilikom izrade potpisa, npr. `"SHA1"` ili `"MD5"`.

fingerprint

Spremnik u koji se sprema potpis po izlazu iz funkcije. Vrijednost ovog parametra može biti `NULL` i u tom slučaju se samo izračuna veličina potrebnog spremnika.

fingerprint_size

Na ulazu ovaj parametar sadrži veličinu razpoloživog spremnika za potpis, a na izlazu se u ovaj parametar sprema potrebna, izračunata veličina spremnika za potpis.

Povratna vrijednost

Ova funkcija vraća `DSRTP_RESULT_OK` u slučaju uspješno izračunatog i vraćenog potpisa, `DSRTP_RESULT_ERR_INVALID_PARAM` ako spremnik nije dovoljne veličine, `DSRTP_RESULT_ERR_GET_CERT` ako je došlo do greške prilikom dohvaćanja certifikata pridruženog ovoj DSRTP sjednici ili `DSRTP_RESULT_ERR_X509_DIGEST` ako je došlo do greške prilikom izračuna potpisa.

dsrtp_get_peer_cert_fingerprint

Ova funkcija vraća potpis certifikata koji je razmijenjen s drugom stranom tijekom DTLS dogovaranja i može se pozvati samo ako je DSRTP sjednica u stanju `DSRTP_SESSION_STATE_SECURE`.

Deklaracija

```
dsrtp_result_t dsrtp_get_peer_cert_fingerprint(
      dsrtp_session_handle hsession, const char *digest,
      unsigned char *fingerprint, unsigned int *fingerprint_size);
```

Parametri

hsession

Rukovatelj DSRTP sjednice.

digest

Naziv algoritma za izradu sažetka koji se koristi prilikom izrade potpisa, npr. `"SHA1"` ili `"MD5"`.

fingerprint

Spremnik u koji se sprema potpis po izlazu iz funkcije. Vrijednost ovog parametra može biti `NULL` i u tom slučaju se samo izračuna veličina potrebnog spremnika.

`fingerprint_size`

Na ulazu ovaj parametar sadrži veličinu razpoloživog spremnika za potpis, a na izlazu se u ovaj parametar sprema potrebna, izračunata veličina spremnika za potpis.

Povratna vrijednost

Ova funkcija vraća `DSRTP_RESULT_OK` u slučaju uspješno izračunatog i vraćenog potpisa, `DSRTP_RESULT_ERR_SESSION_WRONG_STATE` ako sjednica nije u stanju `DSRTP_SESSION_STATE_SECURE`, `DSRTP_RESULT_ERR_INVALID_PARAM` ako spremnik nije dovoljne veličine, `DSRTP_RESULT_ERR_GET_CERT` ako je došlo do greške prilikom dohvaćanja certifikata ili `DSRTP_RESULT_ERR_X509_DIGEST` ako je došlo do greške prilikom izračuna potpisa.

dsrtp_session_send_dgram

Ovo je funkcija za povratni poziv koja je definirana unutar aplikacije, a poziva je DSRTP biblioteka kada želi poslati datagram tijekom DTLS dogovaranja.

Deklaracija

```
int dsrtp_session_send_dgram(dsrtp_session_handle hsession,
      const char *data, int size);
```

Parametri

`hsession`

Rukovatelj DSRTP sjednice.

`data`

Spremnik u kojem se nalaze okteti koji čine datagram.

`size`

Veličina spremnika

Povratna vrijednost

Broj poslanih okteta ili -1 ako je došlo do greške.

dsrtp_session_event_callback

Ovo je povratna funkcija, koja je definirana unutar aplikacije, a poziva je DSRTP biblioteka kada želi obavjestiti aplikaciju o nekom važnom događaju koji se desio unutar DSRTP sjednice.

Deklaracija

```
void dsrtp_session_event_callback(dsrtp_session_handle hsession,
      dsrtp_session_event_t event);
```

Parametri

`hsession`

Rukovatelj DSRTP sjednice.

event

Jedan od događaja definiranih u *dsrtp_session_event_t* enumeracijskom tipu. Za sada je samo jedan događaj definiran, a to je *DSRTP_SESSION_EVENT_STATE_CHANGED*. Taj događaj označava da se desila promjena u stanju sjednice.

Povratna vrijednost

Ova funkcija nema povratnu vrijednost.

dsrtp_log

Ovo je povratna funkcija, koja je definirana unutar aplikacije, a poziva je DSRTP biblioteka kada želi logirati neku poruku - što je jako korisno kod ispravljanja pogrešaka kako u samoj biblioteci, tako i u aplikaciji koja ju koristi. Na aplikaciji je da odluči da li će i kako logirati tu poruku, primjerice na zaslonu ili u datoteku.

Deklaracija

```
void dsrtp_log(dsrtp_log_level_t level, const char* format, ...);
```

Parametri

level

Vrsta logirane poruke (definirano kroz `dsrtp_log_level_t` enumeracijski tip):

- *DSRTP_LOG_DEBUG*: poruka korisna za ispravljanje pogrešaka.
- *DSRTP_LOG_INFO*: informacijska poruka.
- *DSRTP_LOG_WARNING*: poruka upozorenja.
- *DSRTP_LOG_ERROR*: poruka pogreške.
- *DSRTP_LOG_FATAL*: poruka kritične pogreške u logici programa.

format

Format poruke koji ujedno definira i tipove parametara koji slijede budući da ova funkcija prima promijenjivi broj parametara (slično kao `printf` funkcija).

Povratna vrijednost

Ova funkcija nema povratnu vrijednost.