FFI-rapport 2014/01072

Testoppsett for sikkerhetsmekanismer i VoIP

Vinh Pham og Lasse Øverlier

FFI Forsvarets forskningsinstitutt

Testoppsett for sikkerhetsmekanismer i VoIP

Vinh Pham og Lasse Øverlier

Forsvarets forskningsinstitutt (FFI)

13. august 2014

FFI-rapport 2014/01072

1242

P: ISBN 978-82-464-2408-8
E: ISBN 978-82-464-2409-5

Emneord

Voice over IP

Kryptering

SIPS

SRTP

ZRTP

Godkjent av

Ronny Windvik	Prosjektleder
Anders Eggen	Avdelingssjef

Sammendrag

Denne rapporten utforsker muligheten for sikring av Voice-over-IP (VoIP)-trafikk, og demonstrerer hvordan man kan sette opp server og klienter for eksperimentering med VoIP-sikkerhet. Dette innebærer bruk av ulike sikkerhetsmekanismer for beskyttelse av både signalerings- og mediatrafikken. Vi har utført en rekke eksperimenter for å teste ut sikkerhetsmekanismene. Hensikten har vært å få erfaring med og undersøke hvordan disse mekanismene fungerer i praksis, og om mulig avdekke feil eller svakheter og de utfordringer tjenesteleverandører vil kunne erfare ved innføring av slike mekanismer. Vi demonstrerer også hvordan en kan anvende programvarebasert verktøy i analysen av testresultatene.

English summary

This report explores the possibility to secure Voice-over-IP (VoIP)-traffic, and demonstrate how to set-up a test bed for experimenting with VoIP-security. This implies applying different security mechanisms for protection of both signaling and media traffics. A number of experiments have been conducted to test out the security mechanisms. The intention has been to gain experience and determine how good these mechanisms are when used in real life experiments, and potentially discover any failure or weakness as well as challenges a service provider may encounter when applying these mechanisms. We also demonstrate how to use a software-based tool to analyze and visualize results from the experiments.

Innhold

1 Innledning

I de siste tiårene har det har vært en rivende utvikling innen telekommunikasjon. Ny teknologi har gitt oss nye muligheter og måter vi benytter teknologien på. Internett og mobiltelefoni er typiske eksempler på teknologi som siden sin ankomst har endret vår hverdag og måten vi kommuniserer på. En annen større endring som er i ferd med å skje er utfasingen av linjesvitsjet fasttelefoni. I Norge har Telenor uttalt at fasttelefoni nettet, Public Switched Telephone Network (PSTN) og Integrated Services Digital Network (ISDN) nettet vil bli faset ut innen 2017, som følge av nedgang i etterspørsel og mangel på kompetanse og reserveutstyr[1]. Det er forventet at mobiltelefoni og Voice-over-IP (VoIP) vil overta dagens løsninger for fasttelefoni.

VoIP er en teknologi for tale- og multimediakommunikasjon over Internet Protocol (IP). Anvendelsen av VoIP innebærer en migrering fra et lukket linjesvitsjet PSTN-nett til ett mer åpent og pakkesvitsjet IP-nett. Dette betyr igjen at det vil være et større behov for sikkerhet for å kunne beskytte trafikken mot ulike trusler som avlytting, manipulering osv. Motivasjonen i denne rapporten har derfor vært å utforske de muligheter som finnes for sikring av VoIP-trafikk, som inkluderer sikkerhetsmekanismer for beskyttelse av både signalerings- og mediatrafikk (det vil si tale-/videostrømmen). Hensikten har vært å få innsikt i hvordan disse mekanismene virker både teoretisk og i praksis, samt evaluere effekten og eventuelt avdekke svakheter/styrker i de løsningene som finnes pr i dag.

Fremgangsmåten i dette arbeidet har vært tredelt, og består av litteraturstudie, praktiske lab-forsøk og analyse av resultater fra forsøkene. Litteraturstudien har vært viktig for å få oversikt over tilgjengelige sikkerhetsmekanismer og dypere teoretisk innsikt i hvordan disse mekanismene virker. Det er gjort lab-forsøk for å teste ut og verifisere at mekanismene virker i henhold til spesifikasjonen. En viktig del av lab-forsøket har vært relatert til selve prosessen med installering og konfigurering av laboppsettet. Dette er en prosess som til dels har vært utfordrende på grunn av mangel på dokumentasjon. Av den grunn har hensikten med denne rapporten delvis vært å dokumentere denne prosessen slik at leseren i ettertid kan enkelt gjenskape et tilsvarende lab-oppsett. I analysen har vi benyttet Wireshark som verktøy for både logging av trafikken, avspilling, og analyse av pakkeinnhold.

Rapporten er delt inn i 5 kapitler, hvor vi i kapittel 2 gir en kort gjennomgang av de sikkerhetsmekanismene som er mest aktuelle og relevante i denne rapporten. Kapittel 3 gir en detaljert beskrivelse for hvordan server/klient-programvare kan installeres og konfigureres. I kapittel 4 presenterer vi de eksperimentene som er utført for uttesting av sikkerhetsmekanismene, samt resultater fra analysen. Rapporten avslutter med oppsummering og konklusjon i kapittel 5.

Vi har i denne rapporten utelatt en gjennomgang av VoIP-protokollen ettersom dette kan leses i en tidligere FFI-rapport [2]. Videre, vi henviser til [3] for en mer generell gjennomgang av VoIP-sikkerhetsmekanismer.

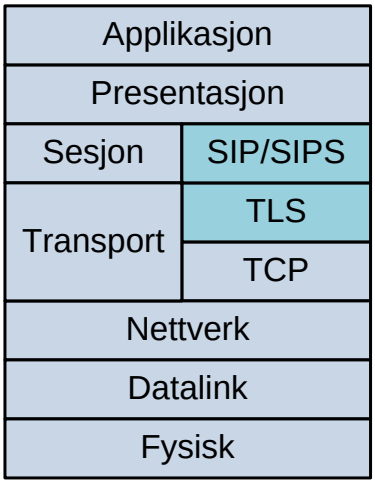
2 Sikkerhetsmekanismer i VoIP

En VoIP-sesjon består av to trafikktyper, Session Initiation Protocol (SIP) og Real-Time Protocol (RTP). SIP er signaleringstrafikken mens RTP er selve mediatrafikken (audio/video). På grunn av denne dualiteten er det derfor nødvendig å beskytte både SIP og RTP for å oppnå god sikkerhet. I avsnittene nedenfor vil vi gi en gjennomgang av de sikkerhetsmekanismer som er relevant for denne rapporten.

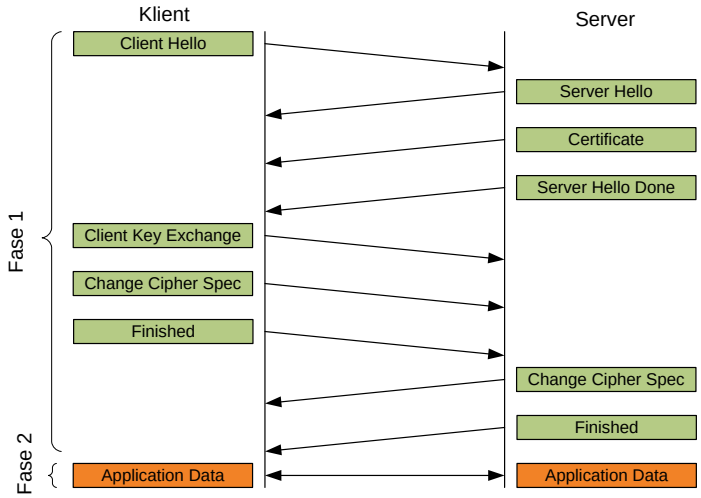
2.1 SIPS

Secure SIP (SIPS) er en sikkerhetsmekanisme for beskyttelse av SIP-signaleringstrafikk mot sikkerhetstrusler[4]. Den gir både autentisering, konfidensialitets- og integritetsbeskyttelse. SIPS bygger på Transport Layer Security (TLS) som er en sikkerhetsprotokoll for TCP-transportprotokollen. Som illustrert i Figur 2.1 er TLS en del av transportlaget i Open Systems Interconnection (OSI) modellen, mens SIPS befinner seg i sesjonslaget. TLS var opprinnelig utviklet av Netscape under navnet SSL og tiltenkt HTTP trafikk. Senere er SSL (versjon 3) modifisert og tilpasset av Internet Engineering Task Force (IETF), og omdøpt til TLS. TLS er nå blitt en akseptert Internet standard og anvendes i mange type applikasjoner. I litteraturen blir akronymet TLS ofte benyttet til å omtale SIPS fordi SIPS er bygget på TLS, og av den grunn, vil dette også forekomme i denne rapporten.

En SIPS-sesjon består av to faser som illustrert i Figur 2.2. Den første fasen består av autentisering, forhandling av krypto-algoritme, og generering av en felles sesjonsnøkkel. For autentisering, må serveren sende over sitt sertifikat til klienten. Ved toveis autentisering vil også klienten sende over sitt sertifikat til serveren. SIPS anvender X.509 sertifikat for autentisering. Sertifikatet er gjerne validert av en anerkjent tredjeparts sertifikat myndighet (CA) som for eksempel Symantec (VeriSign) og DigiCert.



Figur 2.1 SIPS i OSI-modellen

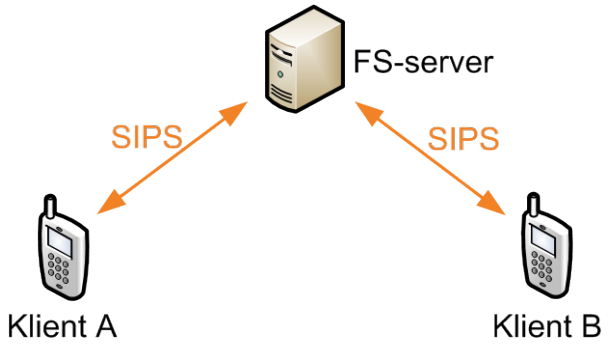


Figur 2.2 Signaleringssekvens i SIPS

I den andre fasen, overføres SIP-trafikk kryptert over TLS forbindelsen. Her blir sesjonsnøkkelen generert i første fase benyttet sammen med en symmetrisk krypto-algoritme for kryptering/dekryptering av meldinger. Standard krypto-algoritmer brukt i denne fasen er AES.

Et viktig moment er at SIPS gir kun stegvis kryptering, det vil si mellom en klient og server. Dette betyr igjen at ved initiering av en samtale mellom to klienter A og B, må det etableres to separate SIPS sesjoner. Først mellom klient A og server, og i neste steg mellom server og klient B, som illustrert i Figur 2.3. Her er det viktig å være klar over at signaleringen er kun beskyttet mellom klient og server, og ikke ende-til-ende mellom klientene. Serveren vil med andre ord opptre som en betrodd mellommann (Trusted-Man-in-the-Middle (MiTM)) og har dermed full tilgang til innholdet i signaleringstrafikken. Dette er nødvendig ettersom serveren er avhengig av innholdet i signaleringen for etablering av samtaleforbindelse og eventuell avregning/fakturering.

En begrensning med SIPS er at den, på grunn av TLS, kun støtter Transmission Control Protocol (TCP) som transportprotokoll, mens SIP vanligvis bruker User Datagram Protocol (UDP).



Figur 2.3 Stegvis beskyttelse av signaleringstrafikk i SIPS

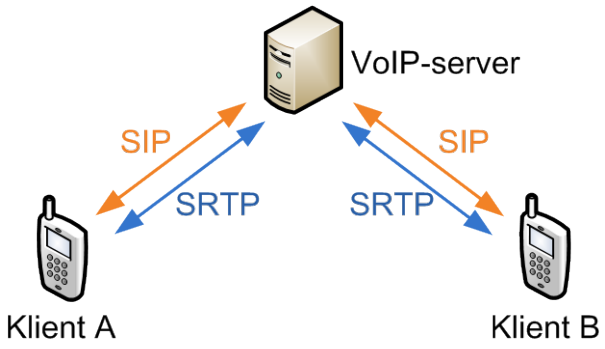
2.2 SRTP

Secure Real-time Transport Protocol (SRTP) er en sikkerhetsmekanisme for sanntid multimediatrafikk [5]. Den gir konfidensialitets-, integritets- og replay-beskyttelse, samt meldingsautentisering både for unikast[1] og multikast[2] RTP mediatrafikk. Samme beskyttelse gis også til RTP's kontroll trafikk RTCP, gjennom SRTCP.

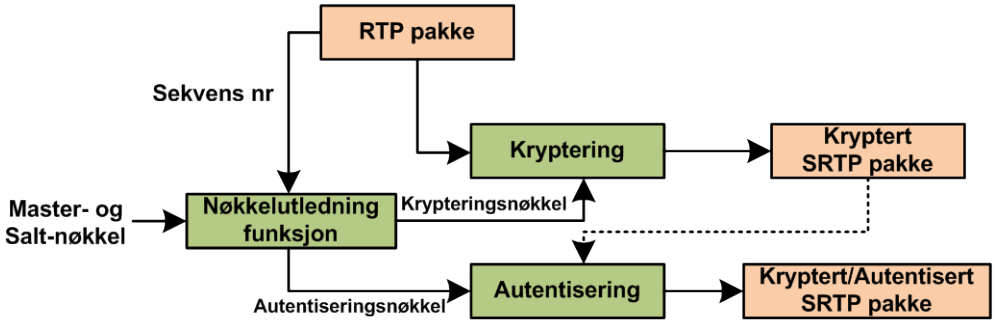
Ved etablering av en kryptert mediaforbindelse mellom to klienter A og B, som vist i Figur 2.4, settes det først opp en SRTP forbindelse mellom A og server, og deretter en ny SRTP forbindelse mellom server og B. Resultatet er stegvis kryptering på samme måte som i SIPS. Serveren vil også her opptre som en betrodd mellommann og har full tilgang til mediatrafikken.

Figur 2.5 viser blokkskjema for krypterings- og autentiseringsprosessen i SRTP. En RTP-pakke som inneholder tale- eller video-data blir før avsending konvertert til en kryptert SRTP-pakke.

[1] Unikast er en-til-en trafikk, hvor man har en sender og en mottaker
[2] Multikast er en-til-mange trafikk, hvor man har en sender og en eller flere mottakere



Figur 2.4 Stegvis kryptering i SRTP, hvor server fungerer som betrodd mellommann.

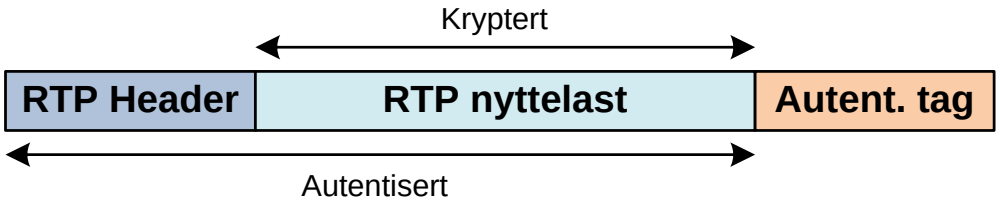


Figur 2.5 Blokkskjema over krypterings og autentiseringsprosessen i SRTP

Dette foregår i to steg, hvor RTP-pakkens nyttelast først krypteres, og i neste steg legges det til autentiserings- og integritetsbeskyttelse. Resultatet er en kryptert og autentisert SRTP-pakke, klar for transmisjon til mottaker. På mottakersiden vil en tilsvarende reverseringsprosess foregå, hvor den mottatte SRTP-pakken blir autentisert/dekryptert og transformert tilbake til sin opprinnelige form som er RTP.

Som standard benyttes Advanced Encryption Standard (AES) [6] med 128 bits nøkkellengde for kryptering av nyttelasten. Fordelen med AES er toleranse for pakketap og pakker som ankommer i feil rekkefølge til mottaker, hvilke er viktige egenskaper spesielt for sanntidsapplikasjoner. I tillegg gir AES beskyttelse mot tjenestenektangrep (DoS) [7] ettersom hver datablokk kan dekrypteres uavhengig av foregående datablokker. Blokkering av en pakke vil dermed ikke påvirke muligheten til å dekryptere etterfølgende pakker.

For meldingsautentisering beregnes det en MAC (Message Authentication Code) kode for hver pakke ved å lage en hash-verdi av hele RTP-pakken, inkludert meldingsheader og den krypterte nyttelasten. Resultatet legges til i pakken i feltet Authentication Tag som vist i Figur 2.6. MAC-koden ivaretar autentisiteten og integriteten til en SRTP-pakke, og er en viktig sikkerhets-mekanisme mot angrep, som for eksempel ”meldingsavspilling” (message replay) og modifikasjon av meldingsheader eller meldingsinnhold uten at det blir oppdaget. Standard-algoritme for MAC-kode utregning er HMAC-SHA1 med 160 bits nøkkellengde.



Figur 2.6 Kryptert og autentisert del av RTP-pakke.

Legg merke til at det kun er RTP nyttelast som blir kryptert, mens meldingsheader kun blir integritetsbeskyttet men ikke kryptert. Dette er et bevisst valg for blant annet å muliggjøre avregning/fakturering og header-kompresjon. På en annen side, kan dette medføre en viss risiko for trafikkanalyse, der en angriper kan samle informasjon fra RTP-header[7].

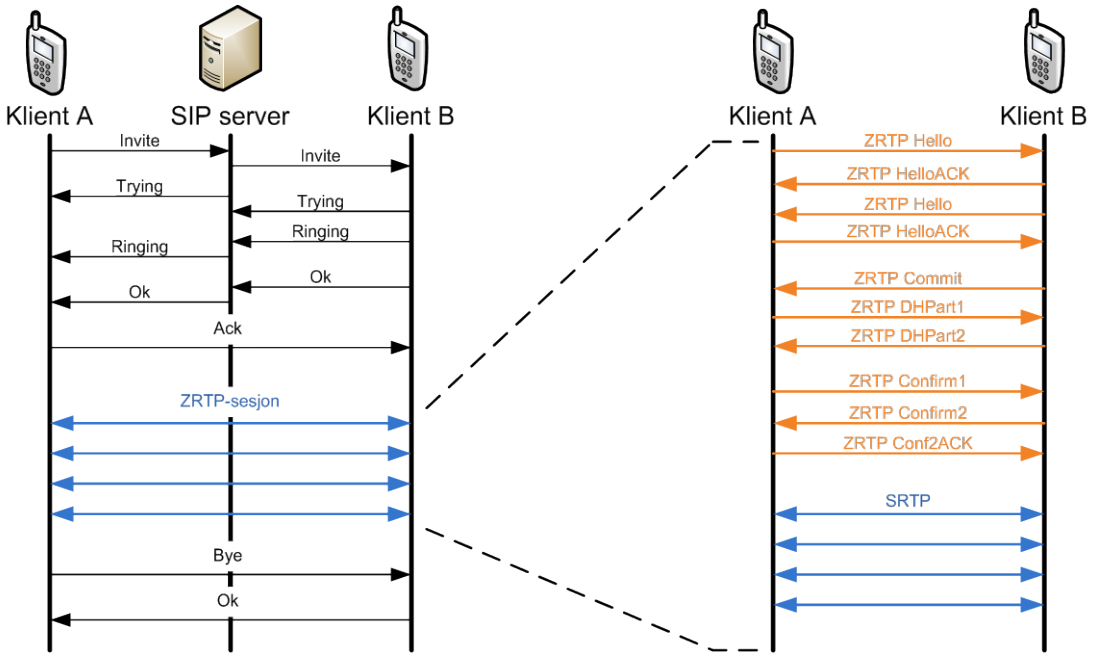
Innen kryptografi er etablering av felles hemmelig nøkkel mellom partene i en kommunikasjons-sesjon kritisk for sikker kommunikasjon. Kryptografiske nøkler er sentrale komponenter og brukes både under kryptering/dekryptering og autentisering av meldinger. Kun parter som besitter nøklene vil være i stand til å ta del i den krypterte kommunikasjonen. SRTP har ingen egen mekanisme for etablering eller utveksling av nøkler, men overlater dette ansvaret til eksterne eller tredjeparts nøkkelhåndterings-protokoller som for eksempel MIKEY[7]. En alternativ metode som ofte anvendes, er nøkkelutveksling gjennom SIP signaleringsprotokollen. Det sendes da en masternøkkel og en master Salt-nøkkel i klar tekst (se Figur 4.8), gjennom ”SIP Invite” og ”SIP Ok” meldinger. Som illustrert i Figur 2.5, anvendes ikke disse nøklene direkte i krypterings- og autentiseringsprosessen, men blir i stedet brukt til utledning av egne krypterings- og autentiseringsnøkler. Det er disse sistnevnte nøklene som anvendes i selve kryptering og autentiseringsprosessen.

For økt sikkerhet, har SRTP i tillegg støtte for regenerering av krypterings- og autentiserings-nøkler, hvilke betyr at nøklene kan fornyes med jevne mellomrom. Pakke-sekvensnummeret brukes da som input slik at nye nøkler genereres etter en bestemt pakkesyklus.

Som beskrevet ovenfor, blir kryptografiske nøkler sendt som klar tekst ved nøkkelutveksling via SIP-meldinger. Det er derfor svært viktig at SIP trafikken også beskyttes. Uten forsvarlig beskyttelse kan uvedkommende få tilgang på nøklene og vil dermed være i stand til å kompromittere kommunikasjonen, det vil si både lese og manipulere, innholdet til selve RTP-trafikken.

2.3 ZRTP

ZRTP er en sikkerhetsmekanisme basert på SRTP, og har mulighet for ende-til-ende kryptering av multimediatrafikk. ZRTP skiller seg fra SRTP ved at den har en integrert mekanisme for nøkkelutveksling basert på Diffie-Hellman (DH) algoritmen som utveksles over RTP-kanalen. Ettersom ZRTP er basert på SRTP, har den de samme sikkerhetsmekanismene som i SRTP. Dette inkluderer konfidensialitets-, integritets- og replay-beskyttelse, samt meldingsautentisering.



Figur 2.7 Signaleringssekvens i ZRTP. Signaleringen gjennomføres over mediakanalen.

Figur 2.7 illustrerer signaleringen i en ZRTP-sesjon. En VoIP-samtale mellom klient A og B initieres med SIP signalering for å etablere en RTP-forbindelse. Når RTP-forbindelsen er etablert starter selve ZRTP-signaleringen med utsending av en ZRTP Hello melding. Her sender klient A en Hello for å sjekke om klient B støtter ZRTP og for å finne ut hvilke algoritmer de to endeklientene har til felles. En Hello-melding inneholder en unik ZID (en tilfeldig verdi på 96-bit som genereres for eksempel under installasjon av programvaren som implementerer ZRTP), og informasjon om versjonsnummer og algoritmer som støttes for ZRTP og SRTP. Dette inkluderer algoritmer for blant annet hashing, kryptering, nøkkelutveksling og Short Authentication String (SAS). Klient B vil ved mottak av Hello respondere med en HelloACK for å bekrefte mottakelsen meldingen. Ettersom ZRTP har innebygget mekanisme for retransmisjon ved pakketap, vil derfor en HelloACK også forhindre retransmisjon av flere Hello-meldinger fra klient A. Tilsvarende, vil også klient B sende en Hello, hvor klient A responderer med en HelloACK.

Etter at Hello- og HelloACK-sekvensen er gjennomført, sendes det ut en Commit-melding for å initiere ZRTP-nøkkelutveksling. Her kan både A og B stå for utsendingen av Commit-meldingen. I vårt eksempel er det klient B som sender Commit-meldingen. Den parten som sender ut Commit-meldingen betraktes som *initiator*, mens mottaker vil være *responder*. Commit-meldingen inneholder en liste over algoritmer valgt ut av initiator, og som benyttes i den etterfølgende nøkkelderivasjonen og utveksling.

I denne fasen er partene blitt enige om hvilke algoritmer som skal anvendes, og både initiator og responder vil nå generere hver sin hemmelige verdi *sv*. Den hemmelige verdien *sv* benyttes så til generering av en tilsvarende offentlig verdi *pv*. Disse verdiene er analogt med et privat og offentlig nøkkelpar i asymmetriske kryptoalgoritmer.

I neste steg utveksles Diffie-Hellman (DH) parametrer, blant annet *pv* verdien, gjennom DHPart1 og DHPart2 meldingene. Legg merke til at det er *pv* verdien som blir sendt til motpart, mens *sv* verdien derimot ikke blir sendt ut da dette er en hemmelig verdi som må beskyttes.

Etter at utvekslingen av DH-parametrene er fullført, har partene tilstrekkelig informasjon til å utlede en felles hemmelighet *s0*. I utledningen av *s0* benyttes det blant annet den hemmelige *sv* verdien og DH-parametrene, inkludert *pv* verdien til initiator og responder. Den felles hemmelige verdien *s0* brukes videre for utledning av øvrige nøkler og verdier, inkludert SAS, ZRTP-nøkler og MAC-nøkler, og SRTP masternøkkel og Salt-verdi.

For å verifisere at utveksling og utledning av nøkler har vært vellykket sender responder og initiator Confirm1 og Confirm2 meldinger til hverandre. Confirm meldingene inneholder kryptert data for å verifisere den krypterte forbindelsen og eventuelt detektere uoverensstemmelser i nøkkelutvekslingen. Dersom partene er i stand til å dekryptere hverandres meldinger er dette en indikasjon på at etableringen av den krypterte forbindelse har vært vellykket.

ZRTP-signaleringen avsluttes med at responder sender en Conf2ACK-melding, og RTP-trafikk kan heretter krypteres med SRTP.

ZRTP har sin fordel overfor SRTP ved at den har integrert mekanisme for nøkkelutveksling basert på DH. En annen fordel ved bruk av DH er uavhengigheten til Public Key Infrastructure (PKI), som kan være både kostbart og samtidig ressurskrevende med tanke på administrasjon av sertifikater. På en annen side, er DH sårbar for MiTM-angrep ettersom sertifikater utstedt av en anerkjent CA ikke benyttes, og det er derfor ingen måte å autentisere motpart. For å motvirke denne type sårbarhet anvender ZRTP en mekanisme kalt Short Authentication String (SAS) for verifikasjon om en ZRTP-forbindelse er sikret ende-til-ende. Ideen er at en (eller flere) SAS-streng blir vist på skjermen hos brukerne, og brukerne verifiserer strengen verbalt med hverandre. Dersom SAS-strengen i begge ender er identiske, er dette en indikasjon på at forbindelsen er sikret ende-til-ende, hvilket betyr at det ikke finner sted et MiTM-angrep.

3 Installering og konfigurering av programvare

Det er i denne rapporten anvendt varierende programvare for å gjennomføre eksperimentene. Tabell 3.1 gir en oversikt over benyttet programvare, som inkluderer operativ systemer, server/klient programvare, og verktøy. Årsaken til at disse programmene har blitt valgt ut er fordi de oppfyller en eller flere av kriterier som høy popularitet, åpen kilde kode, støtte for sikkerhetsmekanismene omtalt i Kapittel 2. Installering og konfigurering av enkelte av disse programvarene kan være kompliserte og krevende. Av den grunn vil vi i avsnittene nedenfor gi en gjennomgang på hvordan dette kan gjøres for disse programvarene. Hensikten med dette er at eksperimentene som er utført i denne rapporten lettere skal kunne gjenskapes av leseren.

Programvare	Versjon	Beskrivelse
Ubuntu	12.04 LTS	Linux operativ system
Windows	XP SP3	Windows operativ system
Asterisk	1.8.20.1	VoIP server
Freeswitch	1.5.1b	VoIP server
CSipsimple	1.02.00 r2330	VoIP klient for smarttelefon
Bria	2.0.0.49465	VoIP klient for smarttelefon
Blink	0.2.10	VoIP klient for PC
XLite	4.5.2 70142	VoIP klient for PC
ZFone	0.92 build 218	VoIP klient for PC
Wireshark	1.8.6	Verktøy for logging og analyse av nettverkstrafikk

Tabell 3.1 Oversikt over programvare anvendt i rapporten

3.1 Asterisk – VoIP-server

Asterisk er server-programvare for VoIP Private Branch Exchange (PBX), det vil si privat telefonsentral. Den tilbyr ulike telefonitjenester som intern- og eksterntelefoni, telefonkonferanse, og mulighet for oppkobling mot det offentlige fasttelefonisystemet Public Switched Telephone Network (PSTN). Asterisk er en av de mest populære VoIP-serverene, og på verdens basis er den anvendt i over en million systemer og i 170 land. Den er gratis tilgjengelig og har åpen kildekode. Asterisk støtter Linux, og vi har i denne rapporten brukt Ubuntu som operativsystem.

3.1.1 Installering av Asterisk

1. Last ned kilde koden fra http://www.asterisk.org/downloads (*asterisk-1.8-current.tar.gz*). I dette eksemplet brukes versjon 1.8.20.1, men fremgangsmåten bør være lik for andre versjoner.
2. Flytt *asterisk-1.8-current.tar.gz* til */usr/src* katalogen:
   ```
   sudo mv asterisk-1.8-current.tar.gz /usr/src
   ```
3. Gå til katalogen */usr/src*, og unzip filen:
   ```
   sudo tar -xzvf asterisk-1.8-current.tar.gz
   ```
4. Gå til katalogen */usr/src/asterisk-1.8.20.1*, og installer nødvendige pakker:
   ```
   sudo ./contrib/scripts/install_prereq install
   ```
5. Kjør deretter følgende kommando for å konfigurere:
   ```
   sudo ./configure
   ```
6. For å aktivere støtte for SRTP kryptering, kjør følgende kommandoen og påse at *res_srtp* under *'Resource Modules'* er sjekket:
   ```
   sudo make menuselect
   ```
7. Kjør følgende kommando for å kompilere og installere:
   ```
   sudo make
   sudo make install
   ```
8. For å generere standard konfigurasjonsfiler *i /etc/asterisk/* katalogen:
   ```
   sudo make samples
   ```
9. For å verifisere at Asterisk støtter TLS:
   ```
   ldd /usr/sbin/asterisk | egrep 'ssl|srtp'
   ```

Responsen bør være: *libssl.so.1.0.0 => /usr/lib/i386-linux-gnu/i686/cmov/libssl.so.1.0.0*

10. For å verifisere at Asterisk støtter SRTP:

```
ls -l /usr/lib/asterisk/modules/res_srtp.so
```

Responsen bør være: *-rw-r--r-- 1 root root 14276 Aug 5 20:29 /usr/lib/asterisk/modules/res_srtp.so*

3.1.2 Generering av sikkerhetssertifikat for SIPS/TLS-kryptering

For å anvende SIPS/TLS kryptering av SIP trafikk trenger vi sikkerhetssertifikater. Stegene nedenfor viser hvordan sikkerhetssertifikater kan genereres:

1. Opprett en katalog for lagring av nøkler:

```
sudo mkdir /etc/asterisk/keys
```

2. For generering av sikkerhetssertifikater trenger vi *ast_tls_cert* skriptet som er tilgjengelig under katalogen */usr/src/asterisk-1.8.20.1/contrib/scripts*/. Kopier dette skriptet til */etc/asterisk/keys katalogen.*
3. Et selvsignert sertifikat myndighet (CA) og Asterisk serversertifikat kan genereres ved å kjøre kommandoen:

```
./ast_tls_cert -C 128.139.1.1 -O "FFI" -d
/etc/asterisk/keys
```

"-C" opsjon for å spesifisere vertsadressen, enten DNS navn eller IP adressen til serveren. I dette eksempelet bruker vi IP-adressen 128.139.1.1.
"-O" opsjon for å spesifisere navnet på organisasjonen.
"-d" opsjon for å spesifisere hvilke katalog nøkkelfiler lagres i.

Du vil få beskjed om å taste inn et passord (4 ganger). Når dette er gjort, vil nøkler og sertifikater som genereres ligge i */etc/asterisk/keys/.*

4. For å verifisere om sertifikatene fungerer kan en for eksempel kjører følgende kommando:

```
openssl s_client –connet 128.139.1.1:5061 \
–CApath /etc/asterisk/keys
```

hvor *128.139.1.1:5061* er IP-adressen og port til serveren, og */etc/asterisk/keys* er lokasjonen til de genererte nøklene.

3.1.3 Konfigurasjon

Konfigurasjonsfiler til Asterisk ligger under katalogen */etc/asterisk*. Den eneste filen som vi trenger å se på i denne konteksten er *sip.conf*. Dette er en konfigurasjonsfil for hvordan Asterisk håndterer inngående og utgående samtaler basert på SIP signalering. Nedenfor er et eksempel på hvordan *sip.conf* fila kan se ut. Den består av to seksjoner, en generell- og en brukerseksjon. Den generelle seksjonen er generelle konfigurasjonsparametre for VoIP/SIP-serveren, mens

brukerseksjonen er konfigurasjonsparametre for hver enkelt bruker; i dette tilfellet har vi to brukere, det vil si user201 og user202. Tabell 3.2 gir en beskrivelse av de viktigste parametrene relatert til sikkerhetsmekanismene SIPS/TLS og SRTP. For en mer utfyllende beskrivelse på de øvrige parametrene henvises det til en tidligere FFI-rapport [8].

```
[general]
port=5060
bindaddr=0.0.0.0
context=domenenavn
tos=0x18
nat=yes
externip=128.139.1.1
disallow=all
allow=all

;Enable SIP encryption (TLS)
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlscafile=/etc/asterisk/keys/ca.crt
tlscipher=ALL
tlsclientmethod=tlsv1 ;none of the others seem to work with Blink
as the client

[user201]
regexten=201
type=friend
host=dynamic
context= domenenavn
secret=passwrd1
callerid="User 201"
dtmfmode=rfc2833
nat=yes
disallow=all
allow=all
registertrying=yes
transport=tls ;Enable SIP encryption (TLS)
encryption=yes ;Enable SRTP

[user202]
regexten=202
type=friend
host=dynamic
```

```
context=domenenavn
secret= passwrd2
callerid="User 202"
dtmfmode=rfc2833
nat=yes
disallow=all
allow=all
registertrying=yes
transport=tls ;Enable SIP encryption (TLS)
encryption=yes ;Enable SRTP
```

Parameter	Beskrivelse
port	Port forteller hvilke port som gjelder for SIP trafikk. Standard er port 5060
externip	Dette er den eksterne IP-adressen til VoIP-serveren
tlsenable	Aktivering av TLS server
tlsbindaddr	Spesifiserer hvilken IP-adresse TLS-serveren bindes til, dvs. adressen til TCP-socketen. 0.0.0.0 betyr at serveren skal lytte på alle interfacer.
tlscertfile	Lokasjonen på filen for server-sertifikat, som inneholder både nøkkel og sertifikat.
tlscafile	Dette er CA-sertifikat, dvs. sertifikat til en CA-myndighet, og brukes for å validere autentisiteten til en annen parts sertifikat, for eksempel en klient eller en annen VoIP-server.
tlscipher	Spesifiserer hvilken SSL-cipher som skal brukes
tlsclientmethod	Spesifiserer hvilken versjon av krypteringsprotokollen som skal brukes (feks. tlsv1, ssl3 og ssl2)
transport	Spesifiserer hvilken transportprotokoll som skal brukes for SIP-trafikk (mulig argumenter er: udp, tcp og tls)
encryption	Spesifiserer om RTP-trafikken skal beskyttes med SRTP-kryptering

Tabell 3.2 Konfigurasjonsparametre i sip.conf-fila.

3.1.4 Viktige Asterisk-kommandoer

I bash terminal:

For å starte Asterisk i debug modus: `sudo asterisk –vvvvvvvvvvvvvgdc`

I kommandoen ovenfor angir `v`'ene at debugging informasjon skal vises på skjermvinduet. Flere `v`'er resulterer i at mer debugging informasjon vises.

For å se på alternative opsjoner til asterisk kommandoen: `asterisk –help`

Kommandoer i Asterisk CLI:

For å stoppe asterisk umiddelbart: `core stop now`

For å stoppe asterisk på en grasiøs måte: `core stop gracefully`

For å restarte asterisk umiddelbart: `core restart now`

For å vise sip-peers: `sip show peers`
For å vise sip-brukere: `sip show users`
For å restarte sip modulen: `sip reload`
For å restarte en modul ved endringer i konfigurasjonsfil eksempelvis *sip.conf*:

```
config reload /etc/asterisk/sip.conf
```

For å vise oversikt over kommandoer: `core show help`

3.2 Freeswitch – VoIP-server

Freeswitch er en alternativ og populær VoIP-server. Den støtter flere plattformer deriblant Windows og Linux. I likhet med Asterisk, har vi installert Freeswitch på samme PC med Ubuntu (Linux) operativsystem. Vi har i denne rapporten brukt Freeswitch til å teste ut krypterings-mekanismen i ZRTP. Dette er en sikkerhetsmekanisme som ikke støttes av Asterisk.

3.2.1 Installering av FreeSwitch

1. Installer nødvendige pakker:

```
sudo apt-get install libcurl4-openssl-dev \
libexpat1-dev libssl-dev libtiff4-dev libx11-dev \
unixodbc-dev python2.6-dev zlib1g-dev libzrtpcpp-dev \
libasound2-dev libogg-dev libvorbis-dev libperl-dev \
libgdbm-dev libdb-dev python-dev uuid-dev bison \
autoconf g++ libncurses-dev
```

1. Gå til ønsket installasjonskatalog for eksempel:

```
cd /usr/src
```

2. Last ned freeswitch:

```
sudo git clone git://git.freeswitch.org/freeswitch.git
```

3. Endre eier av katalogen til *nyeier*:

```
sudo chown -R nyeier: nyeier freeswitch/
```

4. Gå deretter til freeswitch katalogen og kjør bootstrap:

```
cd freeswitch
./bootstrap.sh
```

5. Kompilering:

```
./configure --enable-zrtp
make
```

6. Installering:

```
sudo make all install cd-sounds-install cd-moh-install
```

3.2.2 Konfigurering av FreeSwitch

Konfigurering av FreeSwitch kan være omfattende. Heldigvis er den på forhånd ferdig konfigurert og inneholder ett sett med standard konfigurasjonsfiler. Disse ligger under katalogen */usr/src/freeswitch/conf/*. På forhånd er det definert ett sett med brukerprofiler med nummer fra 1000-1019. Disse brukerprofilene ligger under katalogen */usr/src/freeswitch/conf/insideout/directory/*.

3.2.3 FreeSwitch kommandoer

Her er en liste over noen nyttige FS-kommandoer:

Kommandoer i bash terminal
For å starte FS fra bash terminal:

```
cd /usr/local/freeswitch/bin/
sudo ./freeswitch
```

Kommandoer i FreeSwitch CLI
For å stoppe FS: `fsctl shutdown now`
For å restarte FS: `fsctl shutdown restart`

En mer detaljert oversikt over FS-kommandoer er tilgjengelig på Freeswitch sin Wiki-side[3].

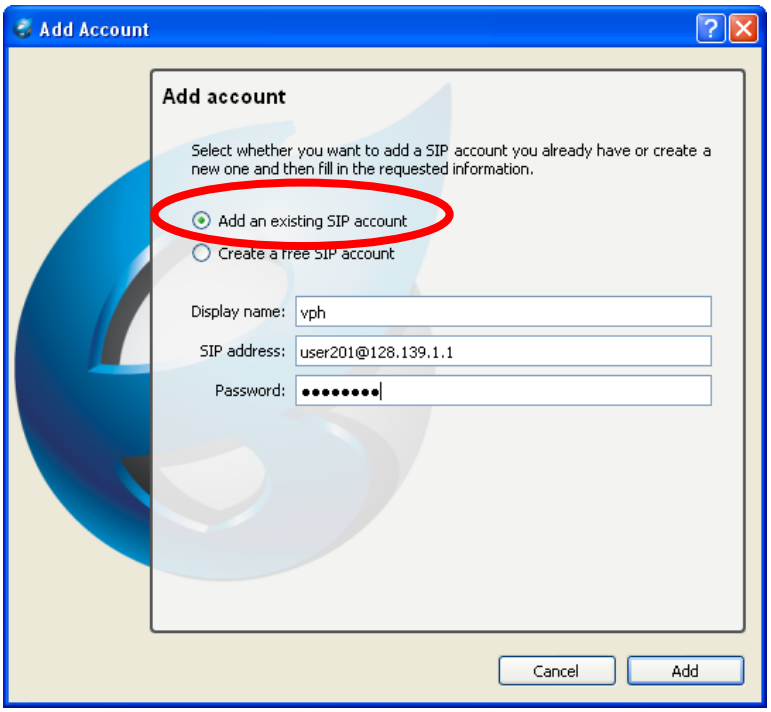
3.3 Blink Softphone VoIP-klient

Blink er en programvarebasert VoIP-klient for PC'er. Denne beskrivelsen viser hvordan Blink konfigureres for oppkobling mot Asterisk-serveren, med og uten støtte for kryptering av SIP/RTP-trafikk. Vi har installert Blink på Windows bærbare maskiner. Prosedyren for hvordan Blink-klienten installeres er utelatt da dette er en relativ enkel oppgave. Klienten kan lastes ned fra Blink sin hjemme-side[4].

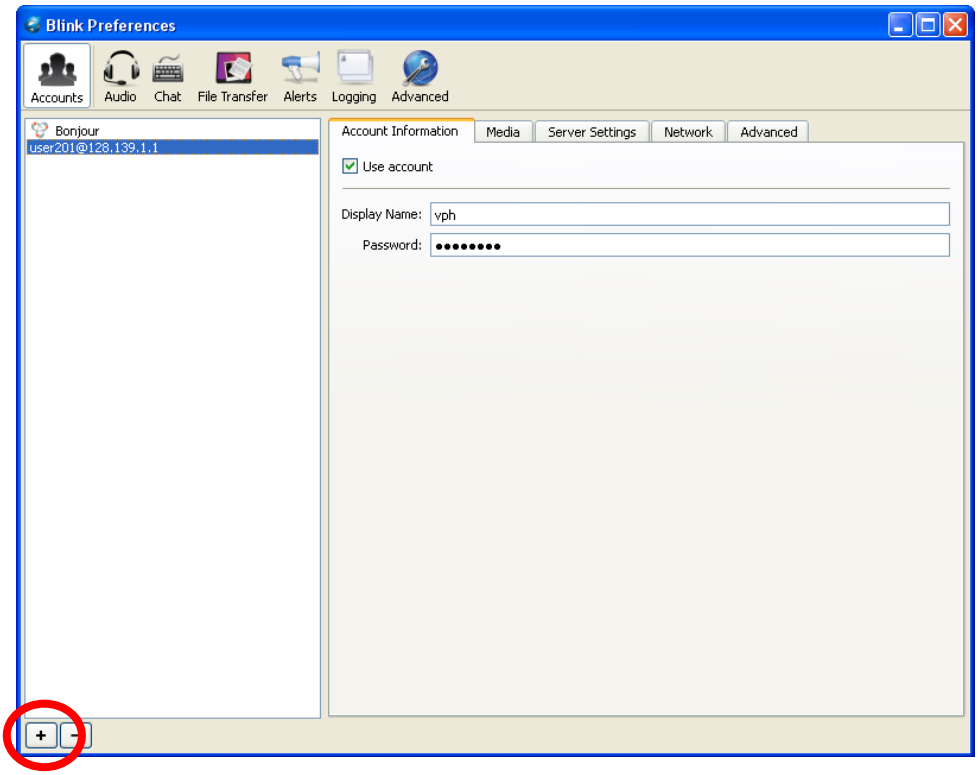
3.3.1 Standard oppsett

1. For å lage en bruker velg Blink->Preferences
2. Klikk på + knappen nederst til venstre for å lage en ny bruker (se Figur 3.2)
3. Velg *Add an existing SIP account* og legg inn ett valgfri ID i *Display name* feltet (Figur 3.1).
4. Spesifiser så URI'en til brukerkontoen, som består av brukerID og IP-adressen til VoIP-serveren. BrukerID og IP-adresse må være identisk ID'en og parameteren `externip` som er definert tidligere i avsnitt 3.1.3. Eksempelvis i vårt tilfelle, *user201@128.139.1.1*.
5. Legg inn passordet for brukeren. Dette er definert i feltet `secret` i avsnitt 3.1.3
6. Velg så den nye brukeren som aktiv brukerkonto på vestre vindu og påse at *Use account* er merket av på høyre vindu (Figur 3.2)
7. Velg *Server Settings* og legg inn IP-adressen til Asterisk-serveren (Figur 3.3). *Port* skal være 5060 som standardverdi og *Transport protocol* skal være UDP. Dersom en ønsker å kryptere SIP-trafikken skal *Port* være 5061 og *Transport protocol* være TLS.
8. Nå skal Blink være ferdig konfigurert for standard VoIP/SIP

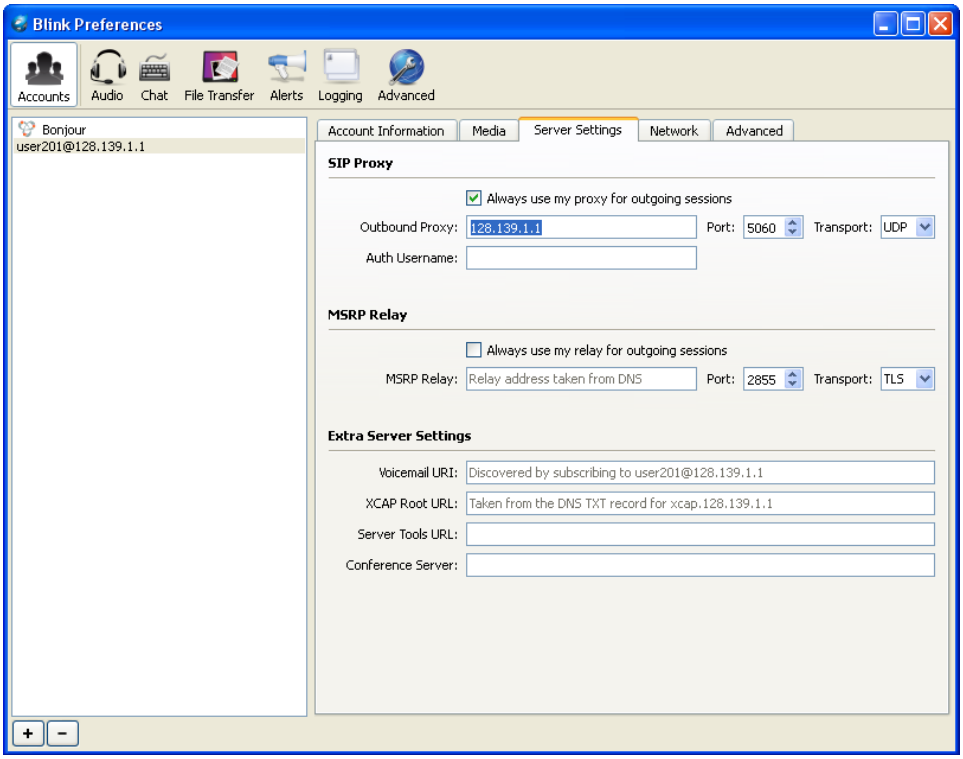
[3] http://wiki.freeswitch.org/wiki/Mod_commands#CLI
[4] http://icanblink.com/



Figur 3.1 Legge inn ny bruker i Blink



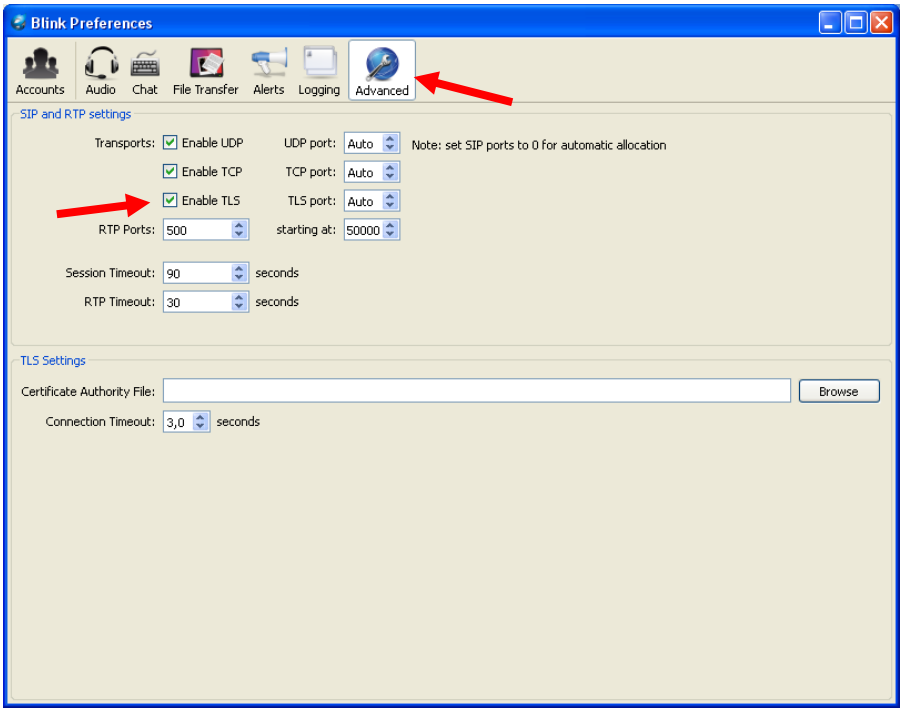
Figur 3.2 Valg av aktiv brukerkonto



Figur 3.3 Serverinnstilling

3.3.2 Oppsett med SIPS/TLS-kryptering

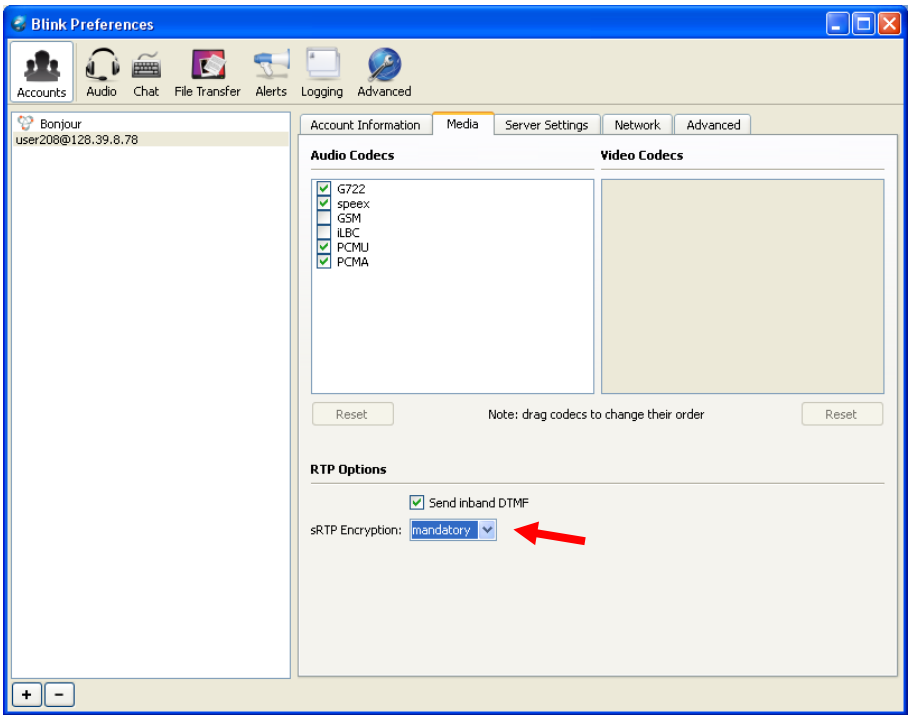
Dersom man ønsker å aktivere SIPS/TLS kryptering, velg *Advanced* knappen helt øverst og huk av på *Enable TLS* (Figur 3.4)



Figur 3.4 Aktivering av SIPS/TLS for kryptering av SIP-trafikk

3.3.3 Oppsett med SRTP-kryptering

Dersom man ønsker å aktivere kryptering, velg *Media*, og velg *SRTP mandatory* (Figur 3.5)



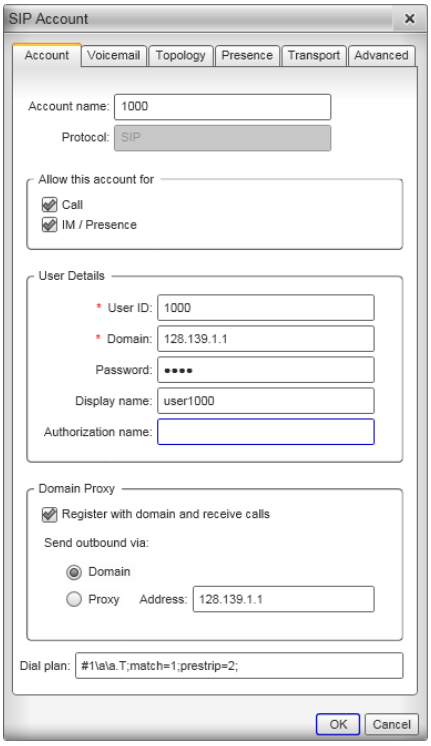
Figur 3.5 Aktivering av SRTP (Kryptering av RTP-trafikk)

3.4 X-lite VoIP klient

X-lite er en alternativ VoIP-klient for PC, og vi har benyttet denne klienten til uttesting av ZRTP-kryptering. Prosedyren for installering av X-lite er enkel og vil derfor ikke bli beskrevet her. I stedet vil vi fokusere på hvordan X-lite kan konfigureres for å kjøre sammen med FreeSwitch. I denne beskrivelsen har vi tatt utgangspunkt i X-lite versjon 4.5.2 som kan lastes ned fra leverandørens hjemmeside[5].

1. For å legge til en ny brukerkonto i X-lite, velges *Softphone* fra menyen. Et nytt vindu som vist i Figur 3.6 vil vises.
2. Velg ett navn for *Account name*. I eksempelet nedenfor har vi brukt brukerprofil 1000.
3. Tast inn bruker-ID 1000 i feltet User ID. Her er det viktig at ID stemmer overens med brukerprofilene definert i */usr/src/freeswitch/conf/insideout/directory/* diskutert i avsnitt 3.2.2
4. Legg inn IP-addressen til VoIP-serveren i *Domain*-feltet.
5. Legg inn passordet for brukeren i *Password*-feltet.
6. Helt til slutt legges IP-addressen til VoIP-serveren på nytt inn i *Domain proxy*-feltet, og trykk Ok.

[5] http://www.counterpath.com/x-lite.html



Figur 3.6 Brukerinnstilling i X-lite VoIP-klient

3.5 CSipSimple – VoIP-klient

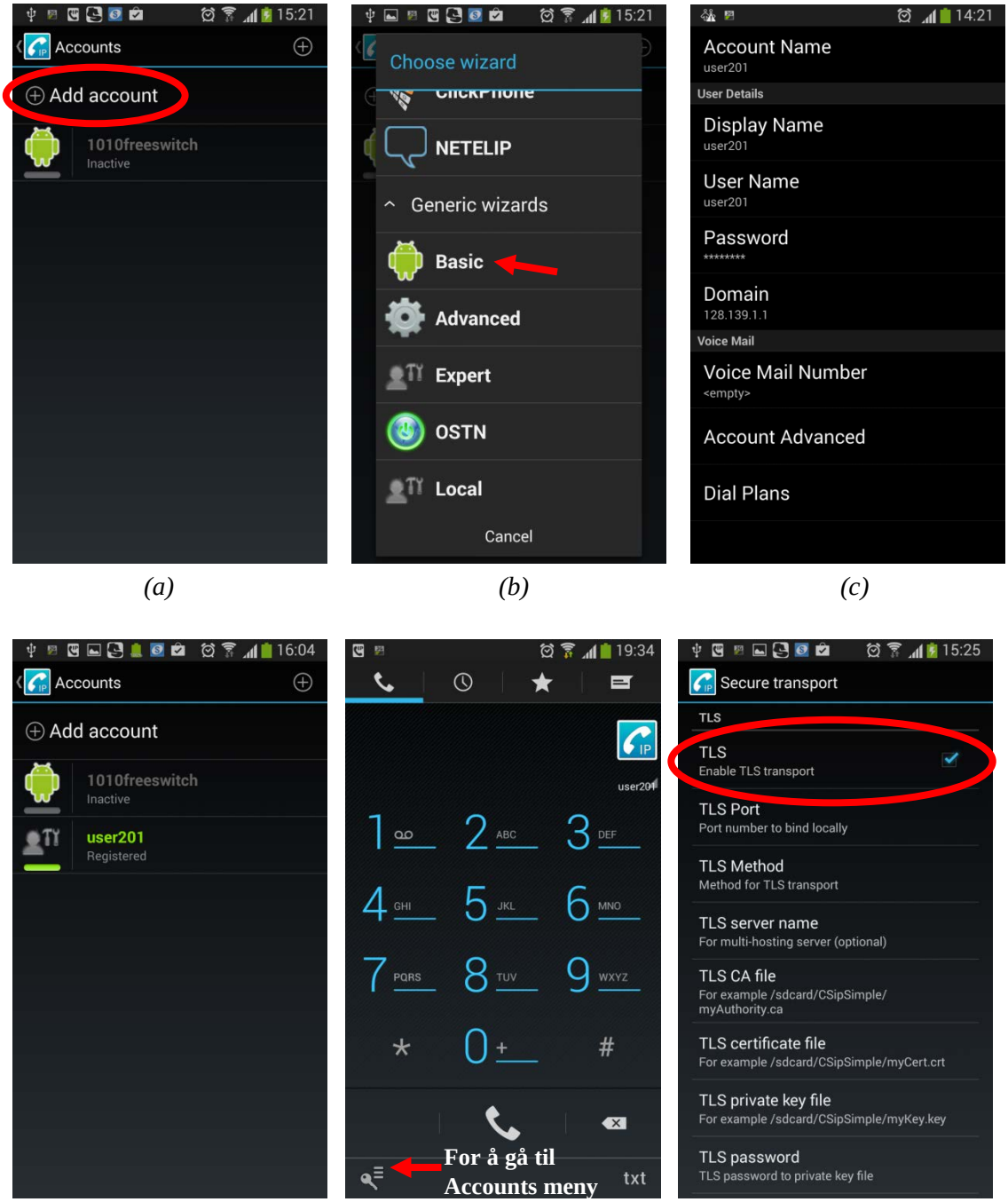
CSipSimple er en populær og gratis VoIP-klient app for Android smarttelefoner. Den kan enkelt lastes ned og installeres fra *Google Play*. Vi viser her hvordan appen skal konfigureres både som standard oppsett, og med støtte for kryptering.

Standard oppsett

1. Velg *Add account* for å opprette en ny konto (Figur 3.7a)
2. Bla nedover og velg *Basic* som vist i Figur 3.7b.
3. Fyll ut feltene som vist i Figur 3.7c og lagre innstillingen. Verdiene skal stemme overens med brukerprofilen i Asterisk (vist i figuren) eller Freeswitch.
4. Nå har vi som vist i Figur 3.7d opprettet en bruker med ID user201. Det er mulig å opprette flere brukerkontoer. Aktivering av gjeldende konto gjøres ved å trykke på den ønskede kontoen. I figuren har vi opprettet to kontoer, en for Freeswitch og en for Asterisk, der sist nevnte er gjeldende konto.

Oppsett med støtte for SIPS/TLS-kryptering

1. For å aktivere SIPS/TLS-kryptering på SIP-trafikken for user201 gjøres følgende: trykk en stund på *user201* kontoen til det dukker opp en liten meny og velg *Choose wizard* (Figur 3.7b), og velg *Expert*. Trykk en gang til på user201 kontoen for å konfigurere. Man vil nå få opp en meny med avanserte innstillinger. Bla nedover og velg *Transport* og velg så *TLS* for å aktivere SIPS/TLS. Lagre innstillingen. Husk at dette er kun brukerinnstillinger for user201.



Figur 3.7 Konfigurering av CSIPSimple

2. Vi trenger også å aktivere SIPS/TLS i globale innstillinger. Trykk på tilbake-knappen på telefonen for å komme til hovedsiden som vist i Figur 3.7e. Trykk på meny-knappen på telefonen og velg *Settings*. Velg *Network*, bla nedover og velg *Secure transport*. En får da opp menyen som vist i Figur 3.7f. Huk av på *TLS*. Under *TLS Method*, velg *TLSv1*.
3. Dersom en ønsker verifikasjon av server-sertifikatet, må CA-sertifikatet generert i avsnitt 3.1.2 kopieres inn på telefonen. Dette er kun nødvendig dersom sertifikatet er selvsignert. Spesifiser lokasjonen på CA-sertifikatet i feltet *TLS CA file*. Huk så av på *Check server* under *Secure transport*-menyen.

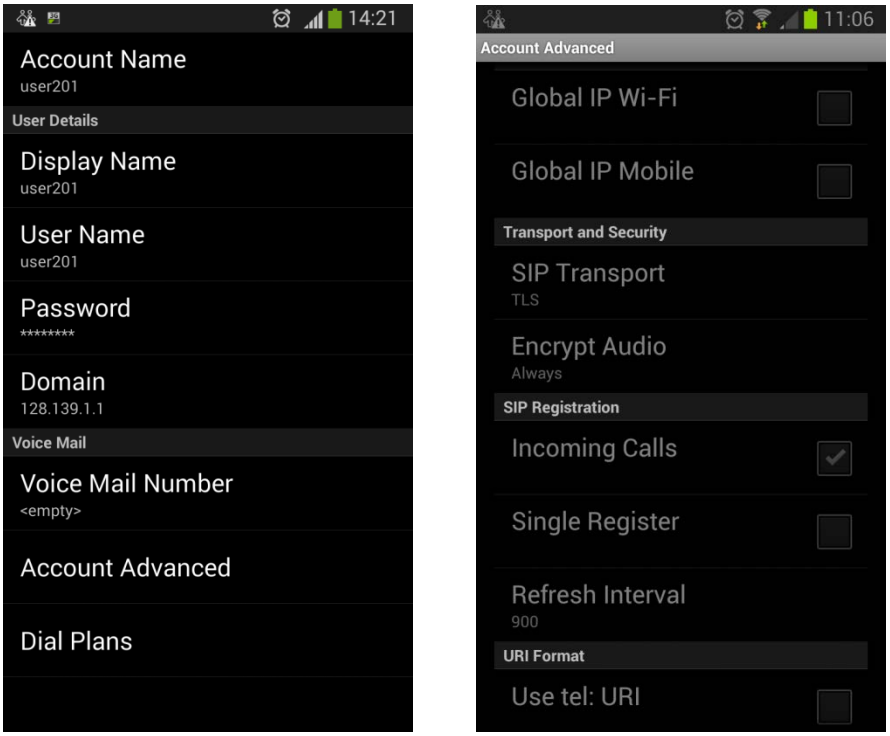
4. Nå er konfigureringen ferdig. Trykk på tilbake-knappen på telefonen for å komme tilbake til hovedmeny.

Innstillinger for SRTP/ZRTP-kryptering

1. For aktivering av SRTP, gå til *Secure transport* i globale innstillinger (Figur 3.7f), bla nedover og velg *SRTP mode*. Velg så *Mandatory*. Om man isteden vil aktivere ZRPT, kan dette gjøres på samme måte ved å velge *ZRTP mode*.
2. Gå tilbake til *Accounts* (Figur 3.7a) ved å trykke på nøkkelikonet i Figur 3.7e. Velg gjeldende bruker. Dersom *Expert modus* ikke er aktivert kan dette gjøres ved å følge punkt 1 under *Oppsett med støtte for SIPS/TLS-kryptering.*
3. I brukerinnstillinger, bla ned til *SRTP mode* og velg *Mandatory*. Dersom man isteden vil aktivere ZRTP, kan dette gjøres på samme måte ved å velge *ZRTP mode*.
4. Lagre innstillingene.

3.6 Bria

Bria er en VoIP-klient for smarttelefoner tilsvarende CSipSimple. Vi har i denne rapporten anvendt Bria versjon 2.0.0.49465, som støtter både kryptering av signalerings- og mediatrafikk ved bruk av SIPS/TLS og SRTP, men mangler støtte for ZRTP. Figur 3.8 viser hvordan en kan konfigurere Bria, både standard oppsett uten kryptering (a), og oppsett med SIPS/TLS og SRTP-kryptering (b).



(a) (b)

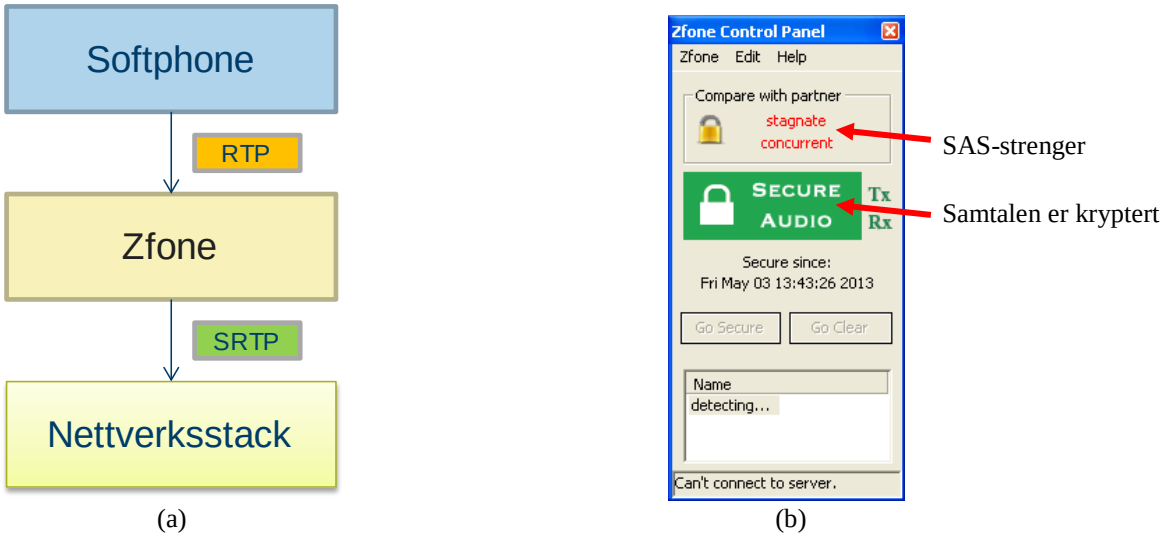
Figur 3.8 Standard innstilling i Bria uten kryptering (a). Innstilling med støtte for SIPS/TLS og SRTP-kryptering (b).

3.7 Zfone

I motsetning til de programvarene som vi har beskrevet tidligere, er ZFone ikke en VoIP klient. Derimot er ZFone en tredjepartsprogramvare som implementerer ZRTP sikkerhetsmekanismen. Den er derfor beregnet for VoIP klienter som ikke har ZRTP sikkerhetsmekanismen innebygget.

Zfone kjører i Internet protokollstacken, og støtter mange ulike OS-plattformer som Windows, Mac OS eller Linux. For VoIP-klienter vil eksistensen av Zfone være helt transparent, ettersom programmet kun lytter på protokollstacken for innkommende og utgående VoIP-pakker for deretter å dektryptere/kryptere pakkene som illustrert i Figur 3.9 (a). Legg merke til at ZFone krypterer en utgående RTP-pakke og gjør om pakken til en SRTP-pakke og ikke en ZRTP-pakke, før den blir sendt ut. Dette er fordi ZRTP bruker SRTP for kryptering, som tidligere forklart i avsnitt 2.3.

For å kunne verifisere om en VoIP-samtale er kryptert, har Zfone et eget grafisk grensesnitt som forteller brukeren om dette, som vist i Figur 3.9 (b). I tillegg har den ett felt med to SAS-strenger for verifikasjon av ende-til-ende kryptering og deteksjon av mulige MiTM angrep.



Figur 3.9 (a) Zfone's krypterings/dekrypteringsprosess. (b) Zfones grafiske grensesnitt

Zfone har vært tilgjengelig for nedlasting fra den offisielle hjemmesiden: http://zfone.com/getstarted.html. Dessverre er dette ikke lenger mulig ettersom serveren er blitt lagt ned. I stedet, kan man laste ned programvaren fra alternative uoffisielle sider som for eksempel: http://www.softpedia.com/get/Internet/Telephony-SMS-GSM/Zfone.shtml

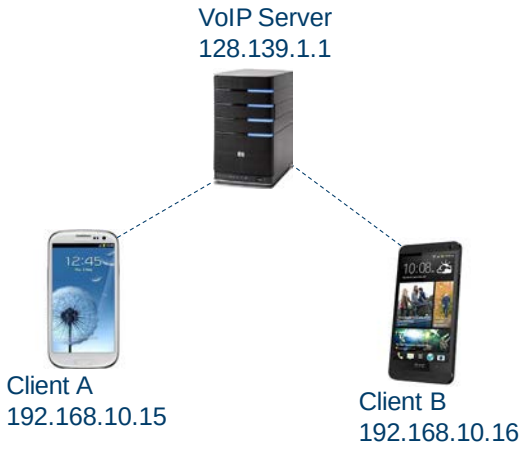
Når programmet er nedlastet er det en enkel sak å installere programmet. Ingen spesielle forandringer i konfigurasjonen av programmet er nødvendig.

4 Eksperimenter

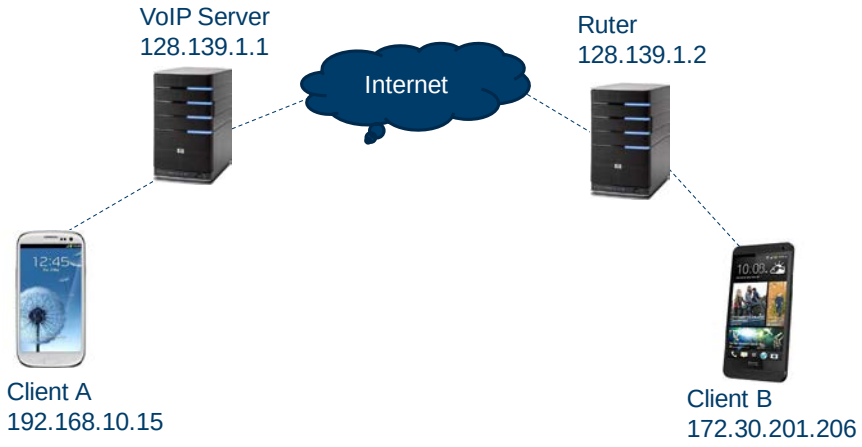
4.1 Testoppsett

For å kjøre eksperimenter med VoIP-krypteringsmekanismer har vi satt opp to testoppsett. Figur 4.1 viser testoppsett 1, som er et enkelt oppsett der to klienter A og B er tilkoblet samme server. I dette oppsettet kommuniserer klientene med hverandre ved bruk av VoIP. All trafikk vil forflytte seg innenfor samme domene eller LAN. Fordelen med et slikt enkelt oppsett er at det er lettere å teste ut funksjonalitet, færre feilkilder, og det er enklere å foreta logging og feilsøking. Dette oppsettet er benyttet til å teste ut krypteringsmekanismene både på Asterisk og Freeswitch VoIP-server.

Figur 4.2 viser testoppsett 2, som er et mer realistisk scenario, bestående av to server og to klienter. Den ene serveren fungerer som VoIP-server, mens den andre fungerer kun som en ruter. Dette testoppsettet gir oss mulighet til å teste ut hvordan VoIP og krypteringsmekanismer fungerer ved bruk av NAT. Vi har benyttet dette testoppsettet til å teste ut ZRTP sammen med Freeswitch.



Figur 4.1 Testoppsett 1



Figur 4.2 Testoppsett 2

4.2 Resultater fra eksperimenter

Det er utført eksperimenter med forskjellige kombinasjoner av signalerings- og mediatrafikk, med og uten kryptering, server/klient-programvare, og testoppsett. Tabell 4.1 viser en oversikt over de eksperimenter som er utført og tilhørende resultater. Eksperimentene er delt inn i fem testsett (TS) som representert i første kolonne. *Oppsett* angir hvilke testoppsett som er benyttet i eksperimentet. *Trafikk* angir trafikktypen, det vil si om det er med eller uten kryptering. Eksempelvis betyr SIP+SRTP at det er ingen kryptering på SIP trafikken, mens mediatrafikken er kryptert med SRTP. *Server* angir hvilken VoIP-server som er anvendt, enten Asterisk (A) eller Freeswitch (FS), mens *klient* angir VoIP-klienten. De siste to kolonnene beskriver testresultater (Res) og eventuelle kommentarer.

TS	Oppsett	Trafikk	Server	Klient	Res	Kommentar
1	1 og 2	SIP+RTP	A/FS	Alle	OK	
2	1	SIPS+RTP	A	Blink	OK	Med verifikasjon av sertifikat
				Bria	OK	
				CSipSimple	OK	Med verifikasjon av sertifikat
3	1	SIP+SRTP	A	Blink	OK	
				Bria	OK	
				CSipSimple	OK	
4	1	SIPS+SRTP	A	Blink	OK	Med verifikasjon av sertifikat
				Bria	OK	
				CSipSimple	OK	Med verifikasjon av sertifikat
5	1	SIP+ZRTP	FS (D)	X-lite+Zfone	OK	Kryptert, ulike SAS
			FS (P)	X-lite+Zfone	Feil	Ukryptert
			FS (B)	X-lite+Zfone	Feil	Ukryptert
			FS (D)	CSipSimple	OK	Kryptert, ulike SAS
			FS (P)	CSipSimple	OK	Kryptert, like SAS
			FS (B)	CSipSimple	OK	Kryptert, like SAS
	2		FS (D)	X-lite+Zfone	Ok	Kryptert, like SAS
			FS (P)	X-lite+Zfone	Feil	Ukryptert
			FS (B)	X-lite+Zfone	NAP	Oppkobling av samtale feiler
			FS (D)	CSipSimple	OK	Kryptert, ulike SAS
			FS (P)	CSipSimple	OK	Kryptert, like SAS
			FS (B)	CSipSimple	NAP	Oppkobling av samtale feiler

Tabell 4.1 Oversikt over eksperimenter og resultater

4.2.1 Eksperimenter med SIP og RTP

I testsett 1 har vi testet VoIP uten kryptering på verken signalerings- eller mediatrafikken, både med Asterisk og Freeswitch, testoppsett 1 og 2, og med alle klienter omtalt i Kapittel 3. Resultater viser at dette fungerer bra.

4.2.2 Eksperimenter med SIPS og SRTP

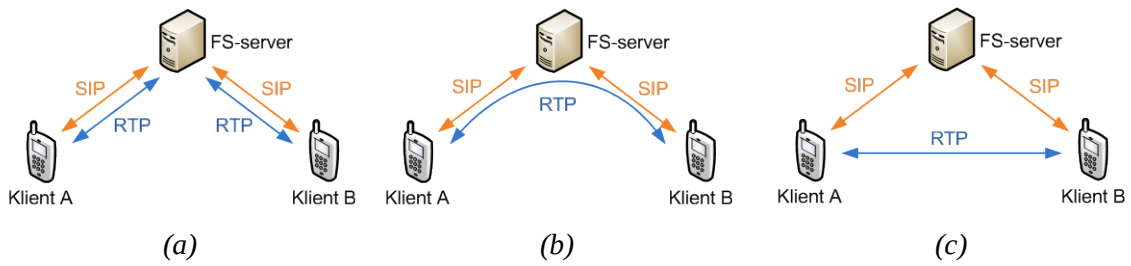
Videre er det testet ut kombinasjonene SIPS+RTP, SIP+SRTP og SIPS+SRTP (testsett 2-4), med testoppsett 1 og Asterisk som server. Som klient er det benyttet både Blink, Bria og CSipSimple. Også resultatet i disse testene var tilfredsstillende.

Ved bruk av SIPS, sender serveren over sitt server-sertifikat (fra *asterisk.crt* eller *asterisk.pem*) for å autentisere seg selv (se avsnitt 2.1). For at klienten skal kunne validere sertifikatet, må sertifikatet enten sjekkes opp mot en PKI-server eller mot et CA-sertifikat installert lokalt på klienten. I vårt tilfelle hvor sertifikatet er selvgenerert vil derfor kun sistnevnte metode være aktuell. Blant de VoIP-klientene vi har testet ut er det kun CSipSimple og Blink som støtter validering av server-sertifikat. Vi har testet og verifisert at funksjonaliteten fungerer på begge applikasjonene. Prosessen med installering av CA-sertifikatet lokalt på klienten kan være plundrete for en vanlig bruker, og denne prosessen må derfor automatiseres for å gjøre det mer brukervennlig.

4.2.3 Eksperimeter med ZRTP

I testsett 5 har vi testet ut ZRTP kryptering på mediatrafikken. Testene er utført med testoppsett 1 og 2 og Freeswitch(FS) som VoIP-server, ettersom Asterisk ikke støtter ZRTP. I Freeswitch er det tre forskjellige servermodus: *default*, *proxy-media*, og *by-pass*.

I default modus (betegnes med D i Tabell 4.1), fungerer FS som en proxy eller betrodd mellommann. I en talesesjon mellom en klient A og B, vil all trafikk mellom disse gå via FS-serveren, det vil si både signalerings- og mediatrafikk (se Figur 4.3 a). Her blir innkommende datapakker fra A, både signalerings og mediatrafikk, analysert og prosessert før de blir videresendt til B. Samme prosess gjelder også for trafikk fra B til A. Hensikten med denne modusen er at kontrollen av en talesesjon er fullstendig overlatt til FS-serveren. FS vil dermed ha ansvaret for blant annet forhandling om codec (algoritmer for koding/dekoding av digitale data signaler), krypteringsalgoritmer etc. Dessuten vil FS kunne tilby mer avanserte funksjoner som taleopptak og IVR[6] (talemenyer) når trafikken går via den. Et annet viktig poeng ved bruk av denne modusen er at klienter skal ikke kunne kommunisere direkte med hverandre, men indirekte via serveren. Det betyr med andre ord at endeklientenes IP-adresse, og dermed lokasjon, vil være skjult for hverandre (topology hiding).



Figur 4.3 Server-modus i Freeswitch. (a) Default. (b) Proxy-media. (c) By-pass

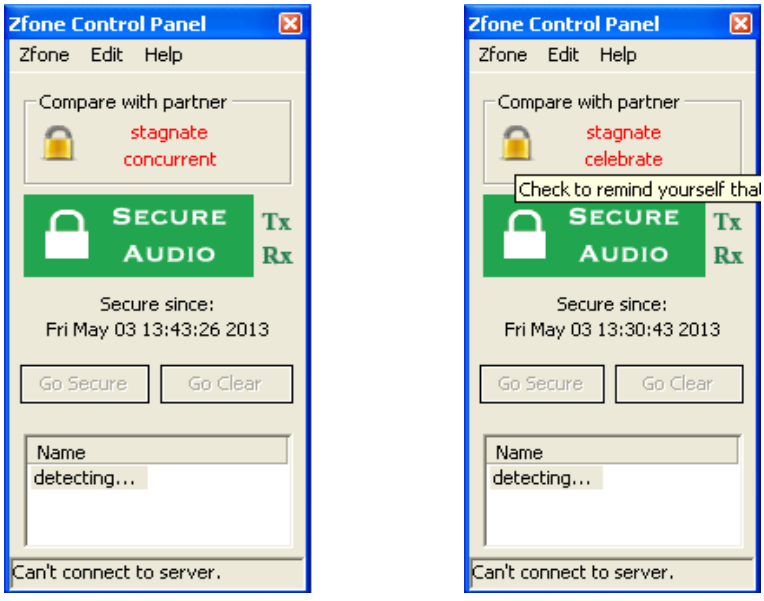
[6] Interactive Voice Response

I proxy-media modus (betegnes med P i Tabell 4.1), vil også all trafikk mellom A og B gå via FS-serveren (se Figur 4.3 b). Forskjellen her er at kun signaleringstrafikken blir prosessert i FS-serveren. Taletrafikken derimot (RTP pakker), vil passere gjennom FS-serveren uprosessert. Ved bruk av denne modusen må blant annet ende-klientene selv håndtere codec-forhandlingen. Sammenlignet med default modus, vil proxy-media modusen være betydelig mindre ressurs-krevende for serveren siden prosessering av RTP-pakker er utelatt.

I by-pass modus (betegnes med B i Tabell 4.1), vil kun signaleringstrafikken gå via FS-serveren (se Figur 4.3 c). Derimot vil RTP-trafikken kunne gå direkte mellom A og B. Denne modusen er nyttig dersom endeklientene har behov for bruk av codec som ikke er støttet av FS, eller når en ønsker høy ytelse for eksempel ved videotelefoni og minimal belastning på FS-serveren.

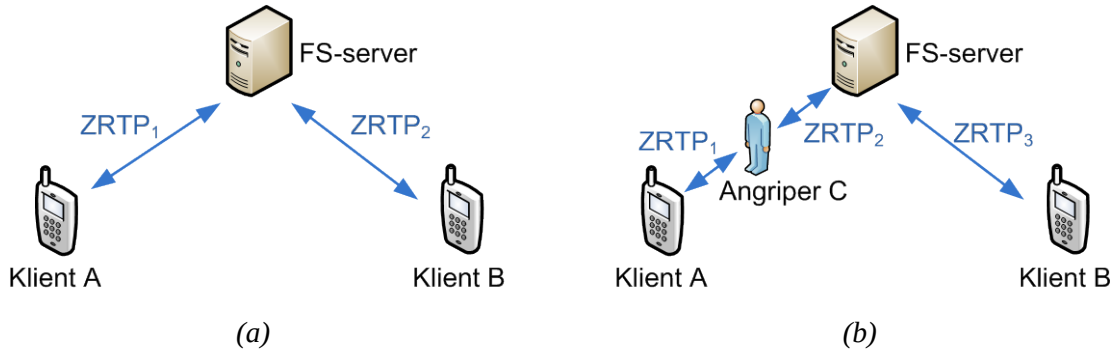
Videre, i dette testsettet har vi anvendt klientene X-lite og CSipSimple. CSipSimple kjører på Android smarttelefoner mens X-lite kjører på bærbare datamaskiner. CSipSimple har integrert støtte for ZRTP-kryptering, mens for X-lite er det behov for bruk av tredjepartsprogramvaren ZFone for håndtering av krypteringen.

Ved bruk av default modus ser vi for begge tilfeller, CSipSimple og X-lite, at mediatrafikken blir kryptert med ZRTP. Krypteringen er dessverre ikke ende-til-ende ettersom vi observerer at SAS-verdiene hos klientene A og B er ulike som vist i Figur 4.4. Dette er som forventet, ettersom i default-modus vil FS-serveren fungere som en betrodd mellommann, hvor det da etableres to separate ZRTP-forbindelser, det vil si $ZRTP_1$ mellom A og FS og $ZRTP_2$ mellom FS og B (se Figur 4.5 a). All trafikk mellom A og B er dermed kryptert og beskyttet over ZRTP-forbindelsene, men vil fortsatt være tilgjengelig for FS.



(a) *(b)*

Figur 4.4 *Figuren viser at RTP-trafikken er kryptert, men vi har ikke ende-til-ende kryptering siden SAS-verdiene på klient A (a) ikke samsvarer med verdiene på klient B (b). VoIP-serveren fungerer som betrodd mellommann (trusted MiTM).*



Figur 4.5 (a) ZRTP-forbindelser i default-modus. (b) MiTM-angrep i default-modus.

Det er imidlertid et problem ved bruk av ZRTP i default-modus, det vil si risiko for Man-in-the-Middle (MiTM) angrep uten at klientene A og B er i stand til å detektere det. Figur 4.5 (b) illustrerer dette problemet. En angriper C befinner seg mellom A og FS-server og når A prøver å etablere en kryptert forbindelse til B via FS, blir det etablert tre ZRTP-forbindelser, det vil si $ZRTP_1$ mellom A og C, $ZRTP_2$ mellom C og FS, og $ZRTP_3$ mellom FS og B, i stedet for at det etableres to ZRTP-forbindelser som i Figur 4.5 (a). Angriper C vil i likehet med FS-server har full tilgang til all trafikk mellom A og B. Som diskutert tidligere, er ZRTP avhengig av SAS-mekanismen for å kunne detektere et MiTM-angrep. Fordi SAS-verdiene uansett vil være ulike i default modus, kan dermed ikke klientene A og B kunne detektere at deres dataforbindelse er kompromittert av C.

Ved bruk av proxy-media og by-pass modus er resultatet ulikt for CSIPSimple og X-lite. I CSIPSimple sitt tilfelle, fungerer ZRTP-krypteringen fra ende-til-ende ved at SAS-verdien hos begge klientene A og B er lik. Derimot observerer vi at krypteringene ikke har fungert for X-lite/ZFone, hvilket medfører at mediatrafikken overføres ukryptert.

Vi har også utført samme forsøk men med testoppsett 2. Resultatet fra testen er den samme som i forrige test for default og proxy-media modus. Dette indikerer at NAT ikke er en hindring for å oppnå kryptert kommunikasjon. For by-pass modus derimot, viser det seg at A og B ikke er i stand til å kommunisere direkte med hverandre, ettersom de begge har privat IP-adresser som ikke er tilgjengelige fra Internett.

For begge testoppsettene viser resultatet at X-lite/ZFone ikke har fungert som forventet, med unntak i default modus. Vi har derfor undersøkt dette nærmere ved å studere trafikklogger. Forskjellen ved bruk av default-modus kontra andre server-modi, er at signaleringen for å etablere en kryptert forbindelse foregår mellom klient A/B og FS-server. Som vi har vist, har dette fungert som forventet. Derimot i proxy-media og by-pass modus, vil signaleringen i stedet være direkte mellom klientene A og B. FS-serveren vil ikke være involvert i denne prosessen. Vi observerer her at kun ”ZRTP Hello”-meldinger blir sendt mellom A og B. Til tross for at disse meldingene når frem kan vi ikke observere noen videre progress i signaleringen. Det mangler med andre ord en rekke meldinger som Hello Ack, Commit, DH utveksling og Confirm for å fullbyrde signaleringen.

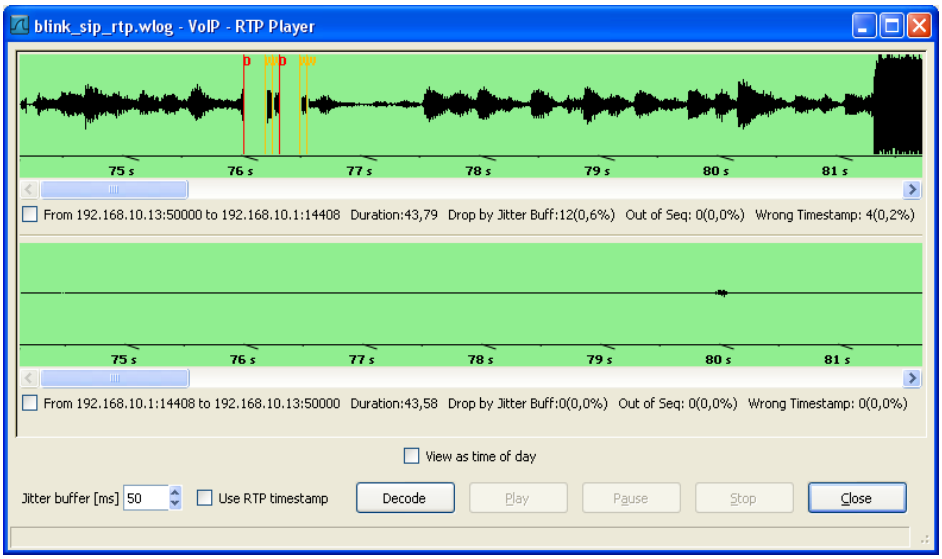
Det er nærliggende å tro at dette kan skyldes en implementasjonsfeil i ZFone, og denne feilen oppstår når signaleringen foregår direkte mellom endeklientene. For å kunne verifisere dette med større sikkerhet må det foretas feilsøking i kildekoden fremfor en analyse basert på nettverkstrafikken. Vi anser dette som en oppgave utenfor omfanget til denne rapporten. Et annet moment som kan være med å styrke vår mistanke er at ZFone versjonen som er tilgjengelig fortsatt er en beta-versjon. Det er ikke annonsert noen planer om videreutvikling av ZFone på deres hjemmeside.

4.3 Analyser med Wireshark

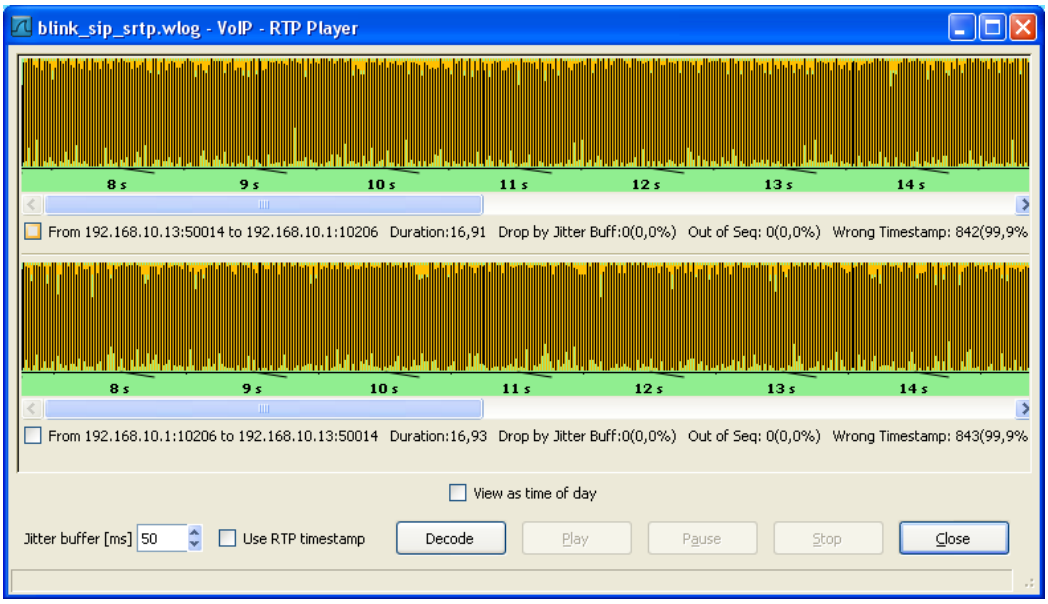
4.3.1 Avspilling av mediatrafikk

Wireshark er et nyttig verktøy for analyse av nettverkstrafikk. Vi har under arbeidet i denne rapporten benyttet verktøyet til både logging og analysering av nettverkstrafikken mellom klient og server. Wireshark har funksjonalitet for avspilling av mediatrafikk, og denne funksjonaliteten er benyttet for å verifisere at kryptering av RTP-trafikken virker som den skal. Figur 4.6 viser lydbildet av ukryptert RTP-trafikk, mens tilsvarende lydbilde av kryptert trafikk er vist i Figur 4.7. For ukryptert trafikk er det fult mulig å avspille samtalen, mens for kryptert trafikk kan vi kun registrere støy i avspillingen.

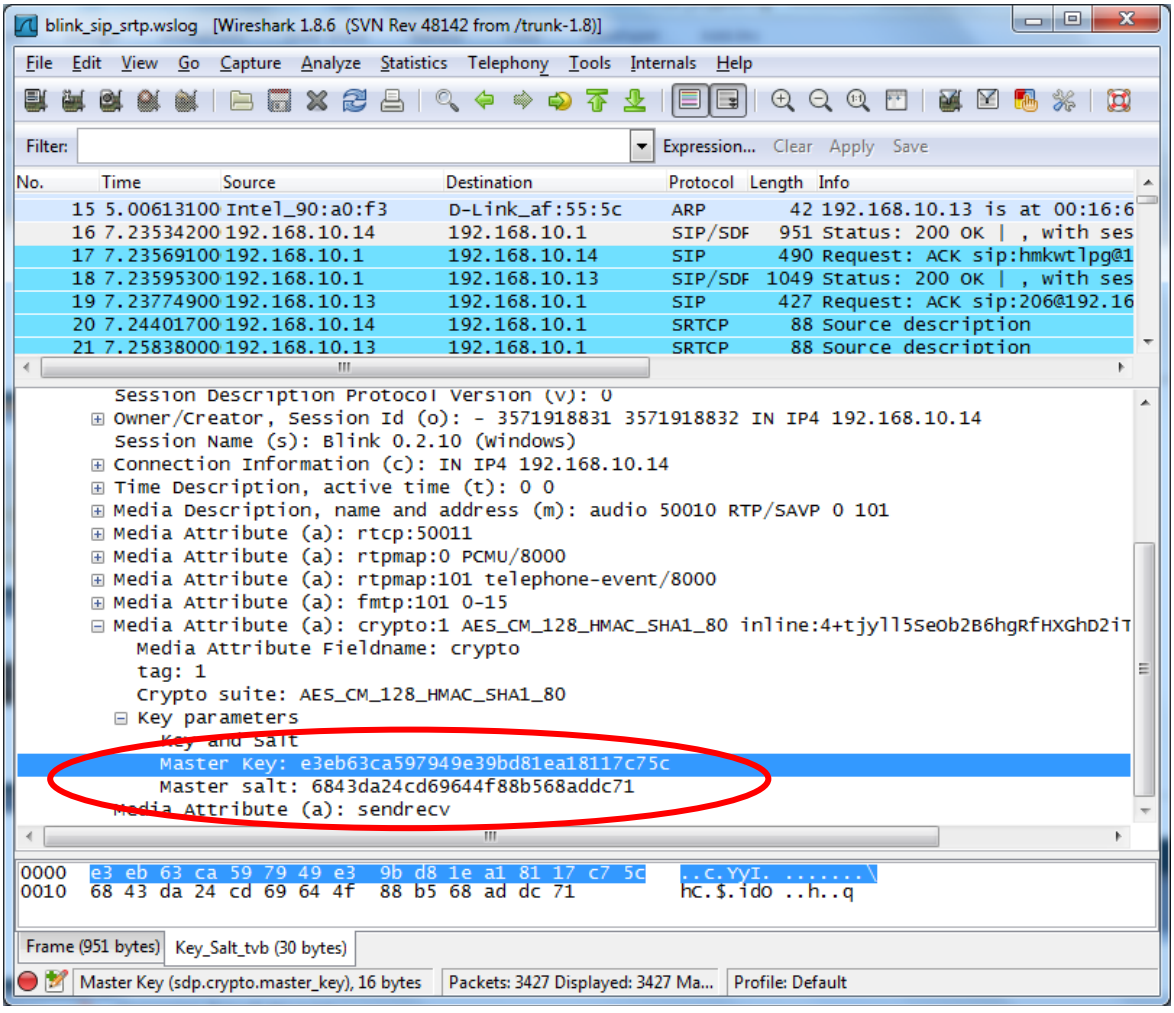
Som tidligere diskutert i avsnitt 2.2, ved bruk av SRTP-kryptering blir kryptografiske nøkler utvekslet gjennom SIP-signaleringstrafikk. Figur 4.8 viser loggen fra en VoIP-samtale hvor signaleringstrafikken er ukryptert, mens mediatrafikken er kryptert med SRTP. Ettersom signaleringstrafikken er ukryptert kan en potensiell angriper lett lese av masternøkkel og Salt-verdien direkte fra loggen da disse sendes som klartekst i en SIP-melding. Med disse nøklene for hånden vil det være mulig å dekryptere SRTP-trafikken. Dette eksempelet viser at uten tilstrekkelig beskyttelse av signaleringstrafikken har krypteringen av mediatrafikken med andre ord liten verdi. God innsikt i virkemåte og riktig anvendelse av en sikkerhetsmekanisme er avgjørende faktorer for at sikkerheten blir ivaretatt.



Figur 4.6 Avspilling av ukryptert RTP-trafikk



Figur 4.7 Avspilling av kryptert RTP-trafikk.



Figur 4.8 Kryptografiske nøkler i SRTP sendes som klartekst i SIP-meldinger. Det er derfor særdeles viktig at SIP-trafikk også beskyttes!

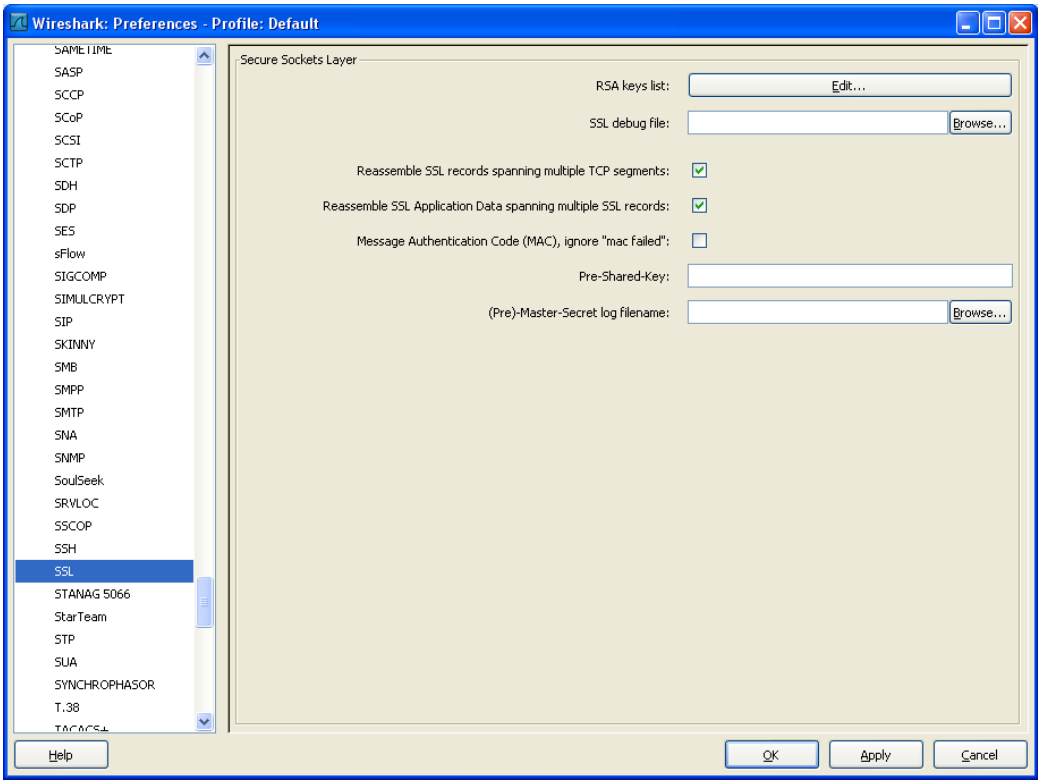
4.3.2 Dekryptering av SIPS/TLS-trafikk

Som diskutert tidligere i avsnitt 2.1 brukes det sertifikat i en SIPS sesjon for autentisering. Dersom den private nøkkelen av sertifikatet er tilgjengelig vil en kunne dekryptere SIPS-trafikken ved hjelp av Wireshark. Prosedyren for dette er beskrevet nedenfor:

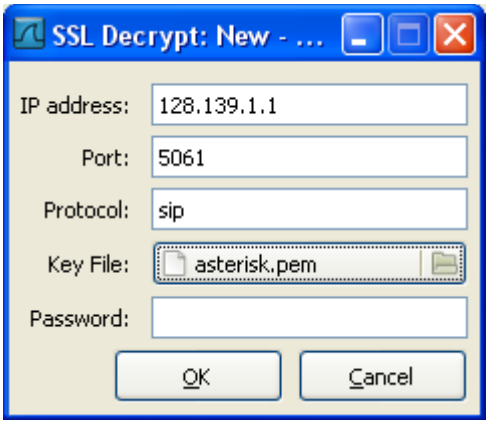
1. I Wireshark, velg *Edit->Preferences*
2. I venstre del av vinduet, velg *Protocol->SSL* (Figur 4.9)
3. Velg *Edit* knappen ved siden av "*RSA keys list*", og velg *New* for å lage ny dekrypteringsprofil
4. Fyll inn verdiene som vist i Figur 4.10, der *IP address* er adressen til Asterisk-server, *Port* er TLS port, og protokoll er type trafikk som transporteres over SIPS, det vil si SIP. Legg til slutt inn filen som inneholder den private nøkkelen (*asterisk.pem*), og klikk på Ok
5. Dersom man senere ikke ønsker at Wireshark skal dekryptere SSL-trafikk, kan dette gjøres ved å slette dekrypteringsprofilen. (Velg *Edit*, velg aktuell profil og klikk på *Delete*)
6. NB! For at dekryptering av TLS skal være mulig, må logg-filen være komplett. Dette innebærer at SIPS-signaleringen i den initielle fasen (Client Hello, Server Hello, Client Key Exchange osv.) må være med i logg-filen. Uten tilgang på den fullstendige signaleringssekvensen vil ikke Wireshark være i stand til å finne sesjonsnøkkelen, og dermed kan den heller ikke dekryptere SIPS.

Figur 4.11 og Figur 4.12 viser samme nettverkstrafikk før og etter dekryptering av SIPS-trafikken. Ettersom SIPS er basert på TLS versjon 1 vises SIPS-trafikken som TLSv1 i Wireshark. Vi ser at etter dekryptering er det full mulig å lese innholdet i SIP-signaleringen i klar tekst. Dette demonstrerer hvor kritisk det er å beskytte den private nøkkelen fra å komme på avveie.

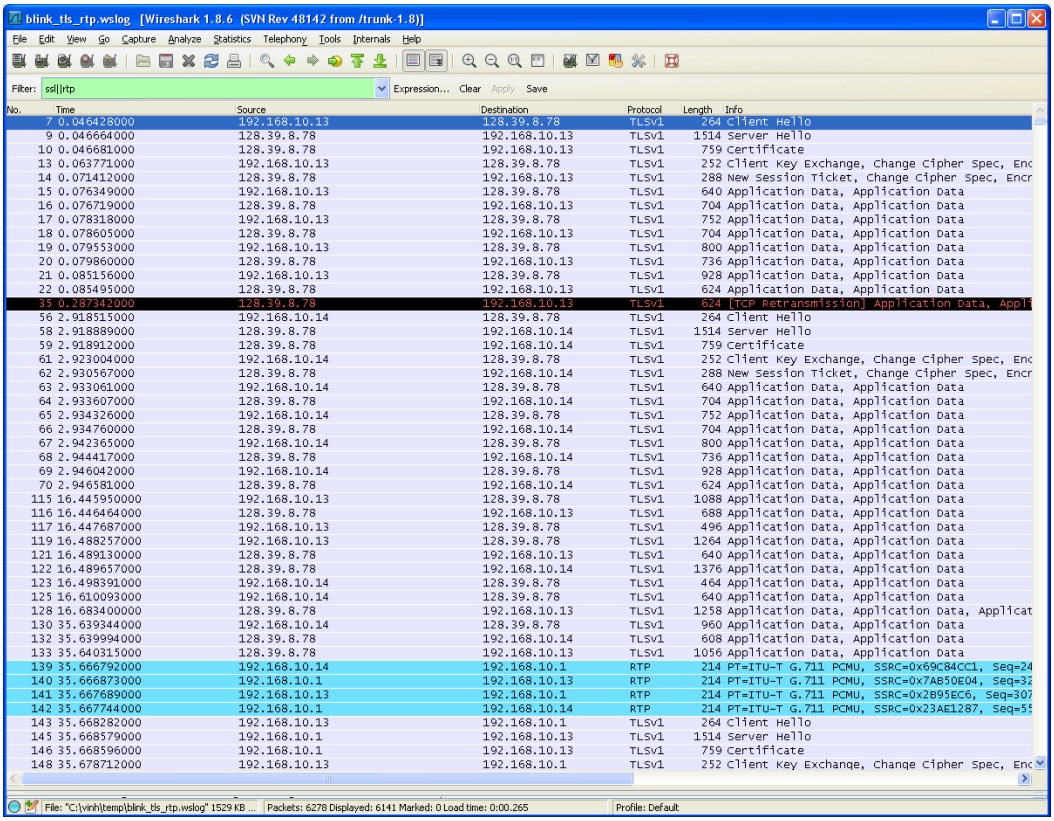
I teorien skal det også være mulig å dekryptere SRTP-mediatrafikk når man først har lykkes med å dekryptere signaleringstrafikken. Dette er fordi nøkkelutvekslingen for SRTP går over SIP-signaleringstrafikken, som diskutert tidligere. Dessverre er denne funksjonalitet pr i dag ikke implementert i Wireshark.



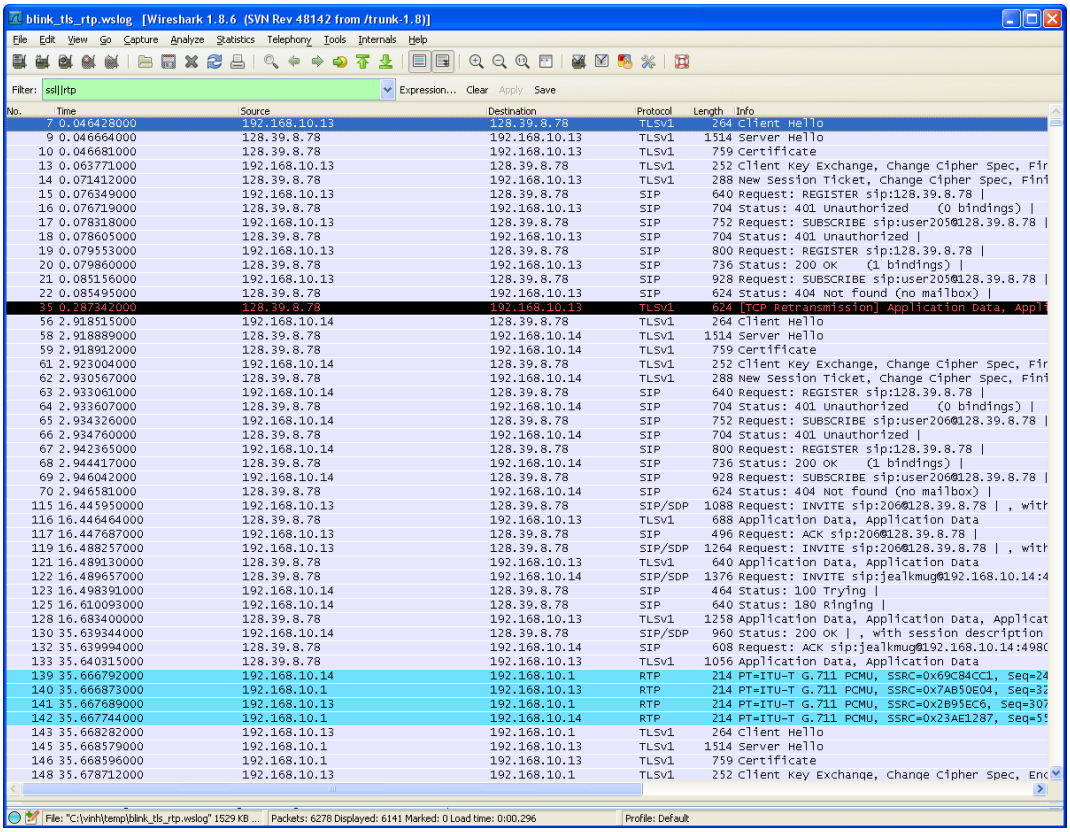
Figur 4.9 Innstilling for dekryptering av TLS/SSL-trafikk (1)



Figur 4.10 Innstillinger for dekryptering av TLS/SSL-trafikk (2)



Figur 4.11 Udekryptert SIPS/TLS-trafikk



Figur 4.12 Dekryptert SIPS/TLS-trafikk

5 Oppsummering og konklusjon

Vi har her demonstrert hvordan en kan sette opp server og klient programvare for eksperimentering med sikkerhetsmekanismer i VoIP. Hensikten er å teste ut hvordan slike mekanismer fungerer i praksis, og om mulig avdekke potensielle svakheter eller feil og mangler.

For kryptering av SIP signaleringstrafikk har vi testet ut SIPS som er en sikkerhetsmekanisme basert på TLS. SIPS tilbyr tosidig autentisering, integritets- og konfidensialitetsbeskyttelse. Forsøk viser at SIPS er kapabel til å sikre SIP-trafikken. En ulempe er at TLS-forbindelsen som settes opp er kun stegvis, det vil si mellom klient og server. Det er ingen garanti for at kryptering av SIP blir benyttet hele veien mellom endeklientene. En annen utfordring er bruken av sikkerhetssertifikater for autentisering og etablering av felles hemmelig nøkkel. Dette medfører ekstra arbeid og kostnad i håndtering av sertifikatene. Et annet problem er at få VoIP-klienter har støtte for verifisering av server-sertifikat. Av de klientene vi har sett på er det kun CSipSimple og Blink som har denne funksjonaliteten. Det er en sårbarhet som kan unyttes av en angriper dersom det ikke utføres verifikasjon av server-sertifikat.

For kryptering av RTP mediatrafikk har vi testet ut sikkerhetsmekanismene SRTP og ZRTP. Vi har observert at SRTP nøkkelutveksling blir signalert over SIP-protokollen. Uten forsvarlig sikring av SIP vil en som lytter på linjen kunne lese av krypteringsnøklene i klartekst, noe som igjen gir mulighet for dekryptering av mediatrafikken. Det er derfor en nødvendighet at SIP krypteres for at SRTP kryptering av mediatrafikk skal ha noen verdi.

ZRTP er en videreutvikling av SRTP, der SRTP anvendes for kryptering av mediatrafikk. I tillegg har ZRTP en integrert nøkkelutvekslingsmekanisme hvor signalering går over RTP istedenfor SIP-kanalen. ZRTP anvender Diffie-Hellman for etablering av felles hemmelig nøkkel. Til tross for at dette er en asymmetrisk protokoll, er den hverken avhengig av PKI eller tredjeparts CA sertifikat myndighet. På en annen side, uten bruk av PKI-sertifikater medfører det en potensiell fare for MiTM-angrep. ZRTP løser dette ved å benytte SAS-mekanismen, hvor endeklientene kan verifisere om en forbindelse er kryptert ende-til-ende. Det er imidlertid et problem når VoIP-serveren opererer som en betrodd mellommann (default-modus). SAS-mekanismen vil i dette tilfellet ikke kunne detektere et MiTM-angrep, ettersom SAS-verdiene på endeklientene uansett vil være ulike.

SIPS, SRTP og ZRTP synes å være de mest aktuelle VoIP-sikkerhetsmekanismene pr i dag, ettersom det er kun disse løsningene vi kjenner til som er blitt implementert og tilgjengelig for eksperimentering. I sin nåværende form er disse sikkerhetsmekanismene etter vår oppfatning fortsatt på et eksperimentelt stadium. Det er generelt høy brukerterskel relatert til installering og konfigurering av nødvendig programvare, og brukerdokumentasjon er mangelfull. I forbindelse med installering/konfigurering og i testfasen har vi ofte vært nødt til å prøve og feile, samt foreta søk på ulike diskusjonsforum for å finne frem til riktig fremgangsmåte.

At kun et fåtall av eksisterende VoIP programvarer, både på server og klient-siden, støtter de sikkerhetsmekanismene som vi har testet, er en indikasjon på at VoIP-sikkerhet i liten grad er utbredt på nåværende tidspunkt. Særlig med tanke på det store utvalget av VoIP-klientprogramvarer som eksisterer er det påfallende at få har støtte for kryptering. I tillegg er kompleksitet under installering og konfigurering medvirkende faktorer for den lave utbredelsen.

Generelt synes det at ende-til-ende kryptering av mediatrafikk kan være en utfordring å få til i praksis. Som vi har diskutert tidligere, gir SRTP kun stegvis kryptering. Derimot, har ZRTP støtte for ende-til-ende kryptering, men det forutsetter at VoIP-serveren er konfigurert riktig (proxy-media eller bypass server-modus) og at endeklientene er tilknyttet samme VoIP-tilbyder. Det er uvisst om det lar seg gjøre å få til ende-til-ende kryptering dersom endeklientene er tilknyttet forskjellige tilbydere. Hvis det skulle være mulig, forutsetter det samarbeidsavtaler tilbyderne seg imellom for at sikkerheten i forbindelsen mellom tilbydernes VoIP-servere også blir ivaretatt. Med tanke på det store antallet VoIP-tilbydere på verdensbasis, kan dette være en utfordring. Ende-til-ende kryptering på tvers av kommunikasjonsplattformer, for eksempel mellom VoIP og PSTN, er en enda større utfordring, ettersom sistnevnte ikke støtter kryptering.

Det er forventet at VoIP-teknologien innen få år vil overta dagens telefoniløsninger basert på PSTN. En slik migrering fra det lukkede PSTN-nettet til et mer åpent og IP-basert nett vil medføre større risiko med tanke på datasikkerheten. Sikkerhetsmekanismer for beskyttelse av VoIP-trafikk vil derfor spille en viktig rolle i møte med den økte trusselen. Mangel på standardiserte og brukervennlige sikkerhetsmekanismer, både på server- og klient-siden har vært en viktig faktor for at de i liten grad er blitt tatt i bruk. Andre medvirkende årsaker ved innføring av slike sikkerhetsmekanismer er høyere krav til ressurser med tanke på prosesseringskraft/minne, og merkostnader relatert til administrering og håndtering av sertifikater.

På bakgrunn av det økende behovet, har sikkerhet i VoIP i den senere tid fått større fokus for å bringe frem gode løsninger, spesielt innen forskningssektoren [9]. De sikkerhetsmekanismene som vi har sett på i denne rapporten er eksempler på resultatet av denne innsatsen, og det gjenstår å se hvilke løsninger som etter hvert blir akseptert som sikkerhetsstandard for VoIP.

Referanser

[1] Direktoratet for samfunnssikkerhet og beredskap (DSB), "Teknologiskiftet i Telenors infrastruktur," 2013.

[2] Bodil Hvesser Farsund og Anne Pernille Hveem, "SIP realisert i Asterisk - teori og eksperiment," FFI-rapport 2013/03081, 2013.

[3] Anne Pernille Hveem, "Sikkerhet i Voice over IP og andre multimediasesjoner basert på SIP og RTP," FFI-rapport 2012/00521, 2012.

[4] IETF RFC 3261, "SIP: Session Initiation Protocol," 2002.

[5] IETF RFC 3711, "The Secure Real-time Transport Protocol (SRTP)," 2004.

[6] IETF RFC 3268., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)," 2002.

[7] Peter Thermos and Ari Takanen, *Securing VoIP Networks. Threats, Vulnerabilities, and Countermeasures.* Addison Wesley, 2008.

[8] Lasse Øverlier, "Egen VoIP-server på 1-2-3," FFI-rapport 2012/01106, 2012.

[9] Angelos D.Keromytis, "A Comprehensive Survey of Voice over IP Security Research,", VOL. 14, NO. 2 ed 2012.

Akronymer

AES	Advanced Encryption Standard
CA	Certificate Authority
DES	Data Encryption Standard
DH	Diffie-Hellman
HMAC	Keyed-hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
IVR	Interactive Voice Response
LAN	Local Area Network
MAC	Message Authentication Code
MIKEY	Multimedia Internet Keying
MiTM	Man-in-the-Middle
NAT	Network Address Translation
OSI	Open Systems Interconnection
PBX	Private Branch Exchange
PKI	Public Key Infrastructure
PSTN	Public Switched Telephone Network
RTP	Real-time Transport Protocol
SAS	Short Authentication String
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SIPS	Secure Session Initiation Protocol
SRTP	Secure Real-time Transport Protocol
SSL	Secure Sockets Layer
TCP	Transport Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VoIP	Voice over IP
ZRTP	Zimmermann Secure Real-time Transport Protocol