VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Implementace IP telefonního SIP klienta na platformě Android s možností zabezpečené komunikace

Implementation of the SIP IP Telephony Client on the Android Platform with Using Secure Communications

2017 **Lukáš Palacký**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Lukáš Palacký**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Implementace IP telefonního SIP klienta na platformě Android s možností zabezpečené komunikace
Implementation of the SIP IP Telephony Client on the Android Platform with Using Secure Communications

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je návrh, implementace a testování IP telefonního SIP softwarového klienta s podporou šifrování signalizace i médií pro mobilní zařízení.

Zadání:

1. Detailně nastudujte možnosti SIP protokolu a problematiku bezpečnosti v IP telefonii.
2. Proveďte teoretickou analýzu stávajících SIP řešení pod OS Android se zaměřením na zabezpečení komunikace.
3. Navrhněte vhodnou knihovnu a koncept pro implementaci SIP klienta v prostředí OS Android.
4. Proveďte samotnou implementaci klienta na uvedené platformě.
5. Otestujte funkčnost vytvořeného SIP klienta na jednoduché konfiguraci s IP telefonní ústřednou.
6. Vytvořte programovou a uživatelskou dokumentaci k uvedenému klientovi.

Seznam doporučené odborné literatury:

[1] SIP Security by Dorgham Sisalem, John Floroiu, Jiri Kuthan, Ulrich Abend, Henning Schulzrinne (May 26, 2009)
[2] Hacking Exposed VoIP: Voice Over IP Security Secrets & Solutions by David Endler and Mark Collier (Nov 28, 2006).
[3] Android Forensics: Investigation, Analysis and Mobile Security for Google Android by Andrew Hoog (Jun 29, 2011)
[4] Android Apps Security by Sheran Gunasekera (Feb 24, 2012)

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 18. *dubna 2017*

……
podpis studenta

Poděkování

Rád bych poděkoval panu Ing. Filipu Řezáčovi Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce. Dále pak kolegům Ing. Jakubu Jalowiczorovi a Ing. Miroslavovi Belešovi za pomoc při testování SIP klienta a za jejich zpětnou vazbu a mé přítelkyni Ing. Kristýně Nožičkové za kontrolu pravopisu a dlouhodobou podporu při řešení této diplomové práce.

Abstrakt

Cílem této práce je teoretické zhodnocení současné nabídky SIP klientů pro mobilní platformu Android a následná implementace vlastního klienta s možností podpory zabezpečené komunikace jak signalizačních, tak hovorových informací. Klient je tedy implementován v programovacím jazyce Java a jádro celé aplikace tvoří knihovna PJSIP. Klient obsahuje funkce jako správu více uživatelských účtů a kontaktů, notifikace a hlasovou komunikaci. Současné množství SIP klientů pro platformu Android, s možností zabezpečení signalizačních a zároveň hlasových dat, je omezené. Toto již omezené množství klientů slibuje plnou podporu zabezpečení, v praxi se bohužel setkáváme jen s částečným zabezpečením komunikace nebo se žádným. Hlavním bodem této práce je tedy vytvoření plnohodnotného klienta s funkčním zabezpečením komunikace za pomocí protokolů TLS pro šifrování signalizačních dat a SRTP pro zabezpečení dat hovorových.

Klíčová slova

Android, Aplikace, Java, Klient, PJSIP, SIP, RTP, SRTP, UDP, TCP, TLS, VoIP, Šifrování, Zabezpečení

Abstract

The aim of this work is theoretical evaluation of the current market supply of SIP clients for the Android mobile platform and the following implementation of my own client with support for secure communications both signaling and speech information. The client is therefore implemented in the Java programming language and the core of the entire application consist of a library PJSIP. The client includes features such as managing multiple user accounts and contacts, notifications and voice communications. Current amount of SIP clients for the Android platform with security support at the same time signaling and voice data is limited. This limited number of existing clients promises full support security, in practice, unfortunately, we meet with only partial security of communication or none at all. The focal point of this work, is to create a full-featured client with functional communication security which will be using TLS to encrypt signaling data and SRTP for data security speech.

Key words

Android, Application, Java, Client, PJSIP, SIP, RTP, SRTP, UDP, TCP, TLS, VoIP, Encryption, Security

Obsah

Seznam použitých zkratek

Zkratka	Význam
ADT	Android Development Toolkit
API	Application Programming Interface
B2BUA	Back to Back User Agent
DoS	Denial of Service
DNS	Domain Name Systém
ENUM	E.164 NUmber Mapping
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IM	Instant Messaging
IP	Internet Protocol
IPSec	Internet Protocol Security
NAPTR	Name Authority Pointer
PKI	Public Key Infrastructure
QOS	Quality of Service
RTP	Real-Time Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SIPS	Session Initiation Protocol Security
SRTP	Secure Real-Time Protocol
SWIG	Simplified Wrapper and Interface Generator
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
RTP	Real Time Protocol

VoIP	Voice over Internet Protocol
WSS	WebSocket Secure

Seznam ilustrací

1 Úvod

Píše se rok 1995 a vedle doposud klasického telefonování vzniká technologie zvaná VoIP (Voice over Internet Protocol). V začátcích byla VoIP technologie limitována technickými možnostmi internetu což se nepříznivě projevovalo na kvalitě přeneseného zvuku. Spolu s vývojem internetových technologií a síťových infrastruktur se začalo řešení VoIP reálně implementovat do menších a středních podniků pro zajištění jejich vlastní interní hlasové komunikační sítě což pro vlastníka prakticky znamenalo kontrolu nad vlastním VoIP řešením a také razantní snížení nákladů.

S nástupem tzv. chytrých mobilních zařízení, se začalo této technologie využívat i v mobilním sektoru. Pro různé mobilní platformy tak začaly vznikat SIP (Session Initiation Protocol) klienti ve formě mobilních aplikací. Nevýhodou však byla omezená mobilita, jelikož přenosové rychlosti mobilních síti nedosahovaly v počátcích požadovaných přenosových rychlostí. Tento problém se eliminoval po zvýšení pokrytí signálu vysokorychlostním mobilním internetem. Hlavní výhodou pro uživatele, jak již bylo řečeno, je finanční úspora, ale mimo jiné také možnost realizace videohovorů, textových zpráv nebo zabezpečené hlasové komunikace.

Při implementaci softwarového SIP klienta se využívají protokoly signalizační a transportní. Pro přenos signalizačních dat se využívá protokol SIP (sestavení relací) spolu s protokolem SDP (Session Description Protocol), jenž je používán k popisu přenášeného obsahu. Přenos hlasových informací je realizován protokolem RTP (Real-Time Protocol), který neobsahuje žádné bezpečnostní mechanismy. Proto lze využít protokolu SRTP (Secure Real-Time Transport Protocol) nebo ZRTP (Zimmermann Real-Time Transport Protocol), které tyto mechanismy implementují. Podrobnému popisu principů fungování používaných technologií a mechanismů je věnována 2. kapitola této práce.

Součástí této diplomové práce je teoretická analýza současných SIP klientů pro operační systém Android. Tato analýza je rozepsána v 3. kapitole diplomové práce.

V současné době existuje několik desítek SIP klientů pro mobilní platformu Android. Z nich však většina nepodporuje možnosti zabezpečení, podporují částečně nebo v praktickém nasazení nefungují správně. Z bezpečnostního hlediska je tento stav nedostačující, jelikož jsou stále častěji zaznamenávány útoky a monitorování hovorů třetí stranou. Proto je účelem této práce navrhnout a implementovat SIP klienta pro operační systém Android s možností podpory zabezpečené komunikace jak signalizačních informací, tak hlasových dat.

V dalších kapitolách této práce je popsán kompletní koncept aplikace. Dále pak nabyté zkušenosti z tvorby aplikace, popis kompilací jednotlivých nástrojů potřebných k vývoji, konfigurace Asterisk serveru používaného při testování aplikace a nakonec samotné výsledky testování.

2 VoIP

VoIP (Voice over Internet Protocol) označuje technologii pro přenos hlasu pomocí protokolu IP (Internet Protocol), někdy také nazývanou jako IP telefonie, v reálném čase. VoIP se stalo populární alternativou tradičního telefonování díky nízkým nákladům.

Požadavky na VoIP síť, které z části plynou ze zkušeností se standardní telefonní sítí:

- Trvalá dostupnost služby.
- Jednosměrné zpoždění pod 150 ms.
- Ochrana hovorů proti odposlechu.
- Garance vysoké kvality a rychlé odezvy v reálném čase.
- Maximální kolísání zpoždění pod 30 ms.
- Maximální ztráta paketů kolem 1 %.
- Konstantní šířka pásma (12-106 kbit/s).

Hlasový přenos vyžaduje maximální spolehlivost a dostupnost sítě. Pro představu se dostupnost moderních telefonních sítí udává 99,999 %, kdy podobně kritická je jen malá část datového provozu.

Na začátku procesu zpracování hlasu je analogový hlasový signál, který je nutné digitalizovat. Toho se docílí vzorkováním kmitočtů 8 kHz a následným kvantováním, při tomto procesu se vzniklý vzorkovaný prvek přiřadí k dané kvantizační úrovni. Posledním procesem je kódování, které přiřadí kvantizační úrovni kombinaci binárních čísel. K tomu se využívají kodeky, což jsou mechanismy pro kódování/dekódování a kompresi/dekompresi hlasového signálu. Kodekům se silnější kompresí sice narůstá zpoždění a procesní náročnost, ale na druhou stranu spotřebují menší šířku pásma. Podle doporučení ITU-T se využívají následující typy kodeků:

- *G.711* - základní kódování PCM, vstupní tok s rychlostí 64 kbit/s, délka rámce je 0,125 ms, zpoždění dosahuje 0,75 ms (základní kodek využívaný VoIP zařízeními).
- *G.723.1* - kódování ACELP, přenosová rychlost je 5,3 kbit/s velikost vzorku je 30 ms a zpoždění 30 ms.
- *G.729* - kódování CS-ACELP, výstupní tok s rychlostí 8 kbit/s, délka rámce a zpoždění je 10 ms (nejpopulárnější kodek z hlediska poměru kvality, zpoždění a výkonu) [1].

2.1 Zařízení ve VoIP

Pro podporu VoIP je nutná existence IP telefonní ústředny (IP PBX) v síti, což je v podstatě komunikační server plnící funkci pobočkové ústředny. IP PBX je založena na přepínání paketů, pomocí kterých je možno přenášet jak hlas, tak i data, oproti klasické telefonní ústředně založené na přepínání okruhů.

2.2 Bezpečnost ve VoIP

Bezpečnostní hrozby ve VoIP jsou dvojího druhu:

- Bezpečnostní hrozby na telekomunikačních službách.
- Bezpečnostní hrozby na systémech vyžadujících všeobecnou dostupnost.

V prvním případě se jedná o útoky na VoIP systémy za účelem kořistění nebo podvedení provozovatele služby. Například útočník může uskutečnit placený hovor pomocí VoIP služby, ale poplatek bude účtován majiteli VoIP služby, nikoliv volajícímu. Útočník se tedy může pokusit připojit neautorizovaný telefon (účet) do VoIP sítě a bude-li mu udělen přístup, pak je útočníkovi umožněno i volání zdarma všude tam, kam má přístup i autorizovaný uživatel služby. Pouze s vhodným procesem ověřování uživatele lze omezit toto riziko.

V druhém případě je ohrožována dostupnost VoIP systémů tzv. DoS (Denial of Service) útoky. Široká škála služeb a protokolů k jejich doručení jsou dvěma hlavními důvody těchto útoku. VoIP se skládá z několika subsystémů, z nichž každý nabízí specifické funkce. Pokud útočník provede DoS útok alespoň proti jednomu nebo několika subsystémům, pak funkcionalita celého systému s velkou pravděpodobností selže [2].

2.3 Protokol SIP

Signalizační protokol SIP je jednoduchý (textový formát) aplikační protokol pro sestavení, modifikaci a ukončování interaktivních relací, které jsou nutné pro komunikování v reálném čase. SIP poskytuje služby pro lokalizaci uživatele (určení koncového systému pro danou komunikaci), navázání spojení (nastavení parametrů pro volající a volanou stranu), dostupnosti uživatele (zjištění dostupnosti a sledování přítomnosti protější strany) a uživatelské možnosti (určení typu média a jeho parametrů). Neprovádí, ale management relací po jejich navázání tzn., že neumí zajistit požadovanou QoS (Quality of Service), jelikož neumí upřednostňovat provoz ani rezervovat šířku síťových prostředků. SIP spolupracuje se dvěma nejčastěji používanými protokoly, RTP pro přenos vlastního obsahu a SDP pro popis přenášeného obsahu.[3]

Bezpečnostní metody a zásady zabezpečení používané SIP protokolem jsou následující (RFC 3261 [6]):

- Zachování důvěry a integrity zpráv.
- Zabránění replay útokům.
- Zamezení podvrhování zpráv.
- Poskytnutí autentizace a soukromí pro účastníky komunikace.
- Zabránění útokům typu DoS.
- Zajištění důvěry, integrity a autentizace zpráv účastníků.[4]

2.3.1 Adresace

Podpora telefonních čísel a webové adresace umožňuje IP komunikaci přecházet mezi telefonní sítí a Internetem bez jakýkoliv problémů. Uživatelé na libovolné síti tedy mohou komunikovat s kýmkoliv na telefonní síti nebo Internetu a nemusí měnit svá stávající zařízení. IP zařízení s podporou SIP spolu mohou komunikovat, pokud znají URI (Uniform Resource Identifier) volaného. Obecná forma zápisu SIP URI je vyobrazena níže.

```
sip:user:password@host:port;uri-parameters?headers
```

Ve formě takovéhoto zápisu se mohou vyskytovat i telefonní čísla. SIP URI se skládá z identifikace uživatele (user), domény (host), která poskytuje prostředky pro zajištění komunikace, pole hesla (password) není doporučeno používat. Standardní port pro SIP je 5060 na UDP, doplňující

parametry (uri-parameters) se od sebe oddělují středníkem a parametry hlavičky (headers) se zadávají za otazníkem. Příklady SIP URI:

- sip:uzivatel@domena.cz.
- sip:+420123456789@domena.cz.

SIP pracuje se jmennými identifikátory tzn., že je možné přeložit pomocí DNS (Domain Name System) číslo na URI. K tomu jsou určeny záznamy NAPTR (Name Authority Pointer) a pomocí techniky nazvané ENUM (E.164 NUmber Mapping) lze mapovat URI na telefonní čísla (dle ITU-T E.164). URI dle RFC 3986 je definováno následovně:

```
<scheme name> : <hierarchical part> [ ? <query> ] [ # <fragment> ]
```

URI se tedy skládá ze scheme name (např. sip, sips, h323, http, https), hierarchical part, která identifikuje zdroj, většinou začíná dvojitým lomítkem ("//") následovaným např. domain name, IP address, username:password@ nebo user@host. Je-li specifikováno číslo portu, pak se začíná dvojtečkou (":") a následuje číslo komunikačního portu (např. :5060). Query (doplňujcí informace) a Fragment (nepřímá identifikace dalšího zdroje začínající znakem "#") jsou nepovinné části.

2.3.2 **Klienti a Servery**

SIP systém se skládá z uživatelských agentů UA (User Agent) a serverů, ke kterým se UA připojují. Koncovými body jsou uživatelští agenti, uživatelské zařízení (SIP telefony, PC a Mobilní zařízení s klientským softwarem) a brány do jiných sítí (hlavně brány pro VoIP). Uživatelský agent obsahuje klienta UAC (User Agent Client), server UAS (User Agent Server):

- *UAC* odesílá požadavky a přijímá odpovědi.
- *UAS* přijímá požadavky a odesílá odpovědi.

Požadavky a odpovědi jsou dva základní typy SIP zpráv. Jelikož koncové zařízení téměř v každém případě obsahuje jak UAC tak i UAS nepoužíváme tyto označení zvlášť, ale označujeme jej UA. Volající vystupuje jako UAC, pokud odesílá zprávu INVITE (požadavek na sestavení spojení) a přijímá na tento požadavek odpověď. Volaný funguje jako UAS, pokud obdrží zprávu INVITE a odešle odpověď na tento požadavek. Tato situace se však mění, když se volaný rozhodne ukončit hovor a odesílá tak zprávu BYE, kterou ukončuje spojení. V takovém případě přebírá volající roli UAC a volaný UAS.

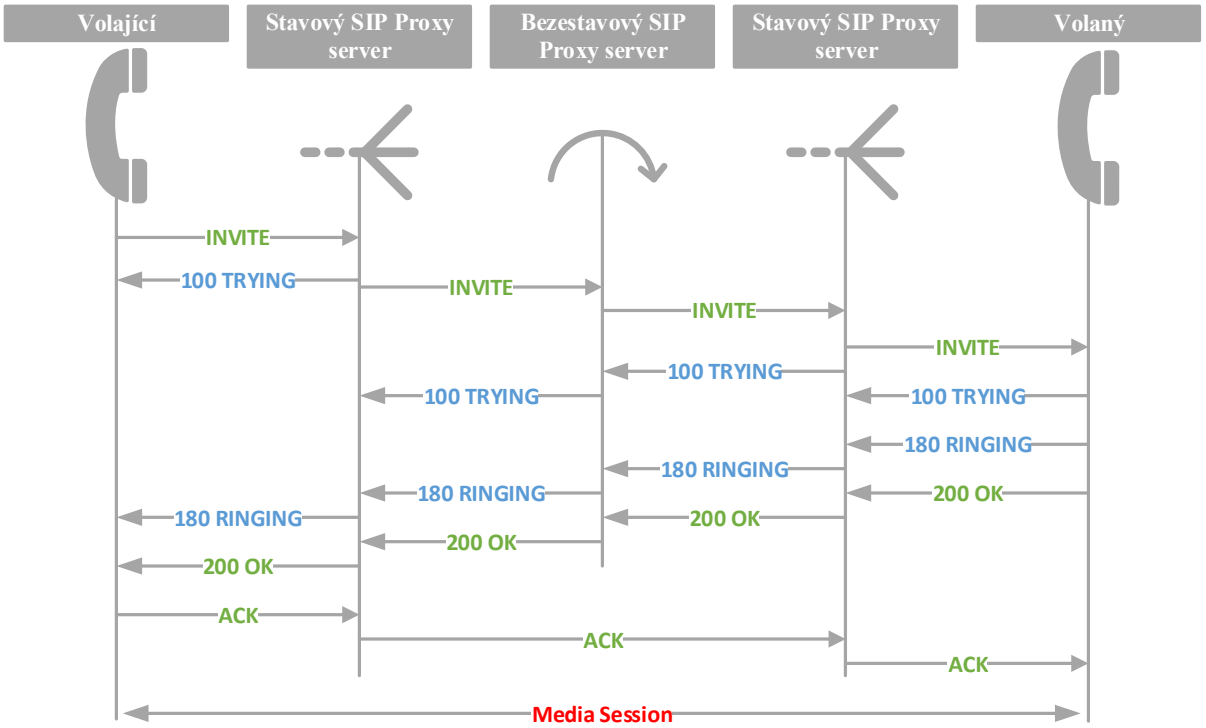
Existuje také B2BUA (Back to Back User Agent), což je speciální typ UA sloužící k přemostění dvou UA. Pomocí B2BUA je možné přemostit spojení dvou UA, ale takovýto B2BUA není výkonný jako SIP Proxy. Servery mohou fungovat jako zástupci (Proxy Server), v takovém případě zastupují klienty při předávání požadavků SIP na další server. Servery také mohou podporovat přesměrování (Redirect Server) a klienta tak informují o dalším skoku v sítí, kam se má zpráva odeslat a klient či Proxy server následně kontaktuje sám doporučené zařízení. O zpracování aktuálních umístění klientů se starají registrátoři (Registrar Server), ti informace o uživatelských agentech aktualizují v serveru umístění (Location Server) nebo v databázi. Příkladem B2BUA je Asterisk s mnoha doplňkovými službami pro koncové uživatele, který je používán jako pobočková ústředna a dokáže obsloužit řádově tisíce uživatelů. Jako zástupce SIP Proxy může být Kamailio až pro stovky tisíc uživatelů [3][5].

SIP Proxy Server

Infrastruktura sítě hostitelů se v SIP systému nazývá SIP Proxy servery. SIP Proxy servery jsou důležitými prvky infrastruktury sítě. Na SIP Proxy server mohou UA zasílat zprávy. Hlavní funkcí je směrování žádostí o spojení blíže k volanému podle aktuálního umístění příjemce, mohou také provádět ověřování, účtování a realizaci doplňkových služeb (např. přesměrování). Při sestavování spojení je základním úkolem SIP Proxy nalézt další SIP Proxy (Next Hop), který vyřídí požadavek a na něj zprávu odeslat. Pro nalezení Next Hop SIP Proxy lze využít nejen statického záznamu, ale lze vyhledávat i v DNS. Požadavek k sestavení spojení je směrován přes jednotlivé SIP Proxy a volaný žádost přijme nebo odmítne. Existují dva základní typy SIP Proxy serverů:

- *Stateless* - bezestavový server.
- *Stateful* - zachovává informace o stavech.

Výhodou stavového SIP Proxy serveru oproti bezestavovému je, že pokud přijme INVITE odpoví na něj 100 TRYING (pouze jednou). Tím server okamžitě potvrzuje přijetí INVITE a odesílatel je tak o tomto stavu informován (neodesílá znovu INVITE). Bezestavový server automaticky přepošle všechny zprávy, které přijme, jelikož nezachovává žádné informace o stavech transakce. Chování těchto dvou typů SIP Proxy, viz obrázek níže.



Obrázek 2.1: *Stateless a Stateful SIP Proxy server.*

Stateless SIP Proxy

Tyto servery pouze přeposílají zprávy nezávisle na jejích vzájemných vazbách. Jsou tedy jednoduché a rychlejší než Stateful SIP Proxy. Což v praxi znamená, že neumí kontrolovat výměnu

zpráv z hlediska smysluplnosti, nedetekují replikaci zpráv a detekce smyček jim trvá podstatně déle než Stateful SIP Proxy, které navíc umí přesměrování a větvení. Využívají se proto jako balanční servery pro překládání a směrování zpráv.

Stateful SIP Proxy

Stateful jsou oproti Stateless SIP Proxy komplexnější. Server vždy po přijetí požadavku vytváří záznam o stavu a uchovává důležité informace po dobu transakce nebo dialogu. Dělí se tedy na dva typy:

- Transakční - uchovává stavy k žádosti, dokud není vyřízena.
- Dialogové - drží stavy dialogu, dokud se neukončí celé spojení.

Jelikož tento server musí uchovávat informace o stavech po celou dobu konání transakce, je jeho výkon limitován. Jak již bylo zmíněno na rozdíl od Stateless SIP Proxy, Stateful SIP Proxy umožňuje detekci replikací zpráv a aplikaci komplikovanějších metod nalezení uživatele (např. volat na telefon do zaměstnání a pokud je nedostupný přesměrovat hovor na osobní mobilní telefon). To vše lze provést díky uchovávání stavových informací. SIP Proxy servery obvykle bývají Stateful, často nabízející větvení, generování záznamů o spojeních, podpora NAT. Přiřazování SIP zpráv do transakcí umožňuje také serveru zvláštní možnosti, například větvení, kdy přijme jednu zprávu a může odeslat více zpráv (uživatel s vícenásobnou registrací).

SIP Redirect Server

SIP Redirect server je prvek SIP systému přijímající požadavek a odesílající zpět odpověď, která obsahuje umístění uživatele v síti. Vyhledává tedy konkrétní příjemce v lokalizační databázi (vytvořené Registrar serverem) na základě přijatých požadavků. Poté vytváří seznam aktuálních umístění uživatele a ty zasílá autorovi požadavku v odpovědi třídy 3xx (Redirect - přesměrování). Autor požadavku tedy dostává seznam lokací a odesílá další požadavky přímo na tyto lokace. Celý proces je vyobrazen na obrázku níže.

SIP Registrar Server

SIP Registrar server zvláštní částí SIP serveru přijímající požadavky na registraci od uživatelů. Tím shromažďuje údaje o jejich aktuální poloze v síti (uživatelské jméno, IP adresa a komunikační port) do lokalizační databáze (Location Database).

2.3.3 Metody

Metody nebo také žádosti jsou používány k sestavení, modifikaci nebo ukončení spojení. Následují základní metody SIP protokolu definované v RFC 3261 [6]:

- Metoda *INVITE* - jedná se o metodu pro vytvoření spojení. Touto metodou lze také zažádat o změnu parametrů spojení (re-INVITE).
- Metoda *ACK* - potvrzuje přijetí žádosti INVITE
- Metoda *BYE* - ukončuje navázané spojení.
- Metoda *CANCEL* - ukončuje zatím nenavázané spojení (tedy pokud ještě není sestaven dialog).
- Metoda *REGISTER* - registruje, aktualizuje nebo odregistruje uživatele v Registrar serveru.

- Metoda **OPTIONS** - je to speciální metoda určená k získání vlastností SIP zařízení. Strukturou je stejné jako INVITE, ale nesestavuje spojení. Používá se například pro zjištění podporovaných funkcí, periodickému zjišťování dostupnosti (mezi registracemi) nebo si naopak může UA udržovat průchodnost zvenčí periodickým zasíláním OPTIONS přes NAT.

Další metody, které nepochází z RFC 3261:

- *INFO (RFC 2976)* - přenos informací během relace.
- *PRACK (RFC 3262)* - potvrzení dočasné odpovědi (1xx).
- *SUBSCRIBE (RFC 3265)* - přihlášení k upozornění na událost.
- *NOTIFY (RFC 3265)* - doručení o události.
- *UPDATE (RFC 3311)* - aktualizace stavu relace.
- *MESSAGE (RFC 3428* - zprávy pro IM (Instant Messaging).
- *REFER (RFC 3515)* - požadavek jiného UA k relaci.
- *PUBLISH (RFC 3903)* - aktualizace prezence.

2.3.4 Žádosti a Odpovědi

Žádosti

Žádosti se většinou používají k registraci uživatele, sestavení, modifikaci, ukončení spojení nebo se může jednat o jiné služby (Presence, Instant Messaging). SIP žádost může vypadat následovně:

```
Request-Line: INVITE sip:0738331699@asterisk.vsb.cz SIP/2.0
Via: SIP/2.0/UDP
158.196.192.32;branch=z9hG4bK9ec4c0248acd48724710d7;rport
From: "SJphone" <sip:7002@asterisk.vsb.cz>;tag=27df31582de
To: <sip:0738331699@asterisk.vsb.cz>
Contact: <sip:7002@158.196.192.32>
Call-ID: C317880624584EB9B1443F8B448CC2830x9ec4c020
CSeq: 2 INVITE
Max-Forwards: 70
User-Agent: SJphone/1.65.377a (SJ Labs)
Content-Length: 321
Content-Type: application/sdp
```

Odpovědi

Obdrží-li UAS žádost, pak na tuto žádost odesílá svou odpověď. Na všechny žádosti, kromě žádosti ACK, musí být odeslána odpověď. Struktura odpovědi je velmi podobná žádosti, hlavním rozdílem je první řádek, kde je obsažena verze protokolu (SIP/2.0) a kód odpovědi (Reply Code). Následuje ukázka, jak může vypadat typická žádost:

```
Status-Line: SIP/2.0 200 OK
Via: SIP/2.0/UDP
158.196.192.32;branch=z9hG4bK9ec4c02048ace1a554;rport=5060
From: "SJphone" <sip:7002@asterisk.vsb.cz>;tag=164235a64f6
To: <sip:1194@asterisk.vsb.cz>;tag=as04503766
Call-ID: 7798248D73F3443CABCD68EE653C791C0x9ec4c020
CSeq: 2 INVITE
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Contact: <sip:1194@158.196.146.12>
Content-Type: application/sdp
Content-Length: 312
```

Celkem existuje 6 tříd opovědí, které mohou být informativního charakteru (1xx) nebo charakteru finálního (2xx - 6xx). Rozdělení tříd je následovné (v závorkách jsou uvedeny některé příklady):

- *1xx* - dočasné informativní odpovědi (100 - TRYING, 180 - RINGING, 183 - SESSION PROGRESS).
- *2xx* - pozitivní finální odpovědi (200 - OK).
- *3xx* - přesměrování (300 - MULTIPLE CHOICES, 301 - MOVED TEMPORARLY, 302 - MOVED PERMANENTLY).
- *4xx* - chyba na straně klienta (400 - BAD REQUEST, 403 - FORBIDDEN, 404 - NOT FOUND: USER NOT FOUND).
- *5xx* - chyba na straně serveru. (500 - SERVER INTERNAL ERROR, 502 - BAD GATEWAY).
- *6xx* - globální chyba. (600 - BUSY EVERYWHERE, 603 - DECLINE).

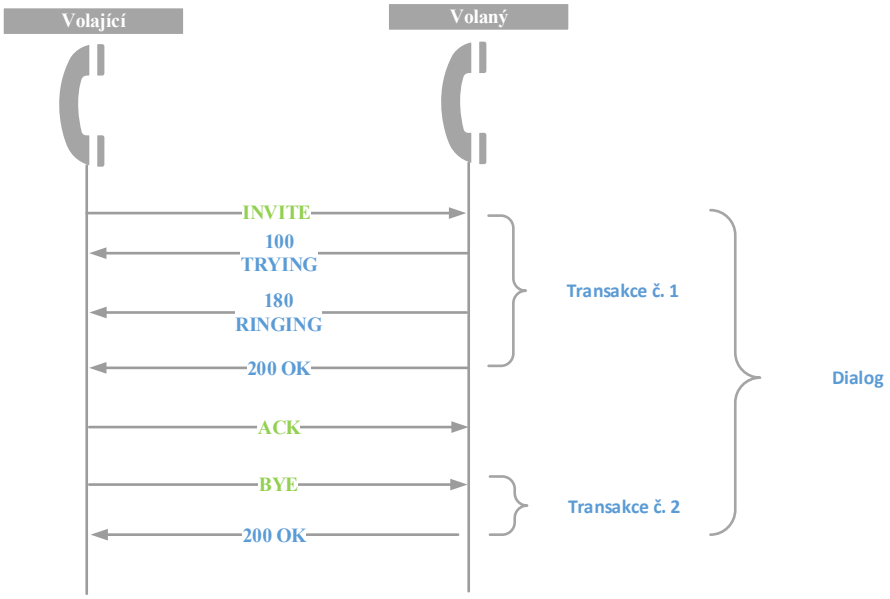
2.3.5 Transakce a Dialog

Transakce

SIP zprávy jsou většinou UA nebo SIP Proxy servery uspořádány do tzv. transakcí. Ty jsou důležité pro tok SIP zpráv a jejich obsluhu, jelikož většina SIP prvku jsou transakční stroje.

Transakce je tedy sekvence SIP zpráv, jež se vyměňují mezi SIP prvky sítě. V transakci figuruje jedna žádost a všechny odpovědi jsou vztaženy k této žádosti. To může prakticky znamenat žádnou až několik dočasných odpovědí a jednu či více odpovědí konečných (např. při větvení v SIP Proxy serveru na zprávu INVITE). Tyto transakce sledují tzv. stateful zařízení, vytvořený stav se tak připojí k žádosti a s transakcí je spojen po celou dobu jejího trvání. Každá transakce má svůj vlastní identifikátor, aby bylo možné příchozí žádosti a odpovědi přiřadit ke správné transakci. Z toho vyplývá, že zprávy v sobě nesou identifikátor náležící nějaké transakci. Ve starším SIP v1.0 (RFC 2543) se identifikátor odvozoval jako hash z hlavičky zprávy, což bylo zdlouhavé a výpočetně náročné. Proto se v novější verzi SIP v 2.0 (RFC 3261) od tohoto způsobu upustilo a jako identifikátor

se používá položka branch (pod polem Via). Tento nový identifikátor lze poznat podle začínající sekvence znaků z9hG4bK (např. branch= z9hG4bK013c15aa). Příklad transakce je vyobrazen na Obrázku 2.2.



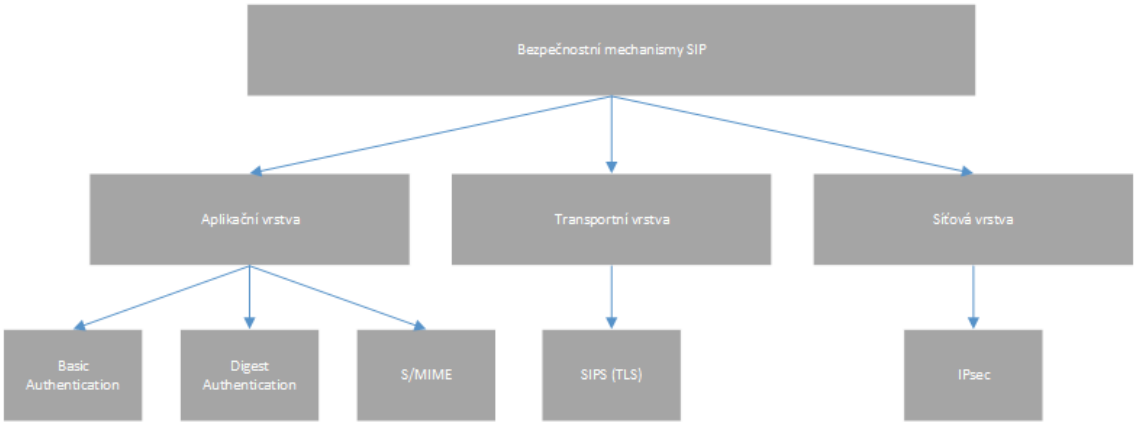
Obrázek 2.2: *SIP Transakce a SIP dialog.*

Pokud byla transakce zahájena žádostí INVITE a zároveň nepatří finální odpověď do třídy 2xx, pak se do této transakce zahrnuje i zpráva ACK (dle RFC 3261). Tzn., že při přijetí finální odpovědi např. třídy 4xx, žádost ACK do transakce patřit bude.

2.3.6 Zabezpečení signalizace

Signalizační protokoly nemají žádné zabezpečení proti potencionálním útokům, proto jsou k dispozici bezpečnostní mechanismy, které řeší problematiku bezpečnosti nad již vytvořenými standardy.

Dle RFC 3261 [6] je považováno za nejlepší techniku pro zachování důvěry v signalizaci použití úplného šifrování zpráv, které zaručuje, že zprávy nebyly nijak modifikovány třetí stranou. Koncepce protokolu SIP nedovoluje šifrovat žádosti a odpovědi stylem End to End v tzv. plném rozsahu, jelikož některé z polí jako např. Route a Via musí být v hlavičce viditelné a přístupné pro Proxy servery většiny síťových architektur i SIP servery. Některé Proxy servery potřebují upravit prvky zpráv (např. přidáním hodnot do pole Via v hlavičce). Používají se tedy zabezpečovací mechanismy nižších vrstev šifrující celé SIP žádosti a odpovědi Hop by Hop metodou. V následujícím obrázku jsou vyobrazeny mechanismy pomocí, kterých lze zabezpečit SIP protokol (tyto mechanismy budou dále rozebrány):

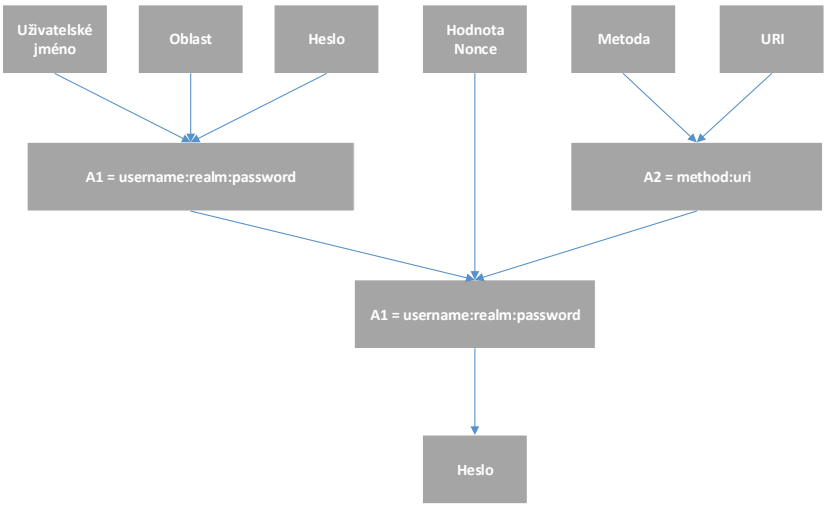


Obrázek 2.3: *Bezpečnostní mechanismy SIP protokolu.*

Digest Authentication

S protokolem SIP je možné použít autentizační metodu, která je založena na autentizaci v HTTP (RFC 2617). Tento nespojový způsob ověřování je založen na výzvě. Pomocí něj lze dosáhnout aby Proxy, Registrar nebo UA zjistil totožnost zadavatele příchozí žádosti. Digest Authentication využívá starší metody Basic Authentication, ale namísto přenosu otevřeného hesla se vytváří pomocí hashovací funkce MD5 řetězce, které se pak porovnávají. Příklad: UA zašle požadavek REGISTER na SIP Registrar server nebo INVITE požadavek na SIP Proxy server. Žádost je zamítnuta a je odeslána odpověď. V případě Proxy Serveru je to odpověď typu 407 (Proxy Authentication Required) a v případě Registrar serveru se jedná o odpověď 401 (Unauthorized Response). Tyto odpovědi obsahují zprávu, jak se má požadavek autentizovat a jaká metoda má být použita. V závislosti na typu požadavku je pak do hlavičky přidáno odpovídající pole. Jedná-li se o Proxy server, vyžaduje se pole Proxy-Authenticate, u Registrar serveru je to pak pole WWW-Authenticate. Obě pole obsahují následující parametry:

- *Authentication Scheme* - neboli použité autentizační schéma.
- *Realm* - určuje pole rozsahu platnosti autentizace s platností v rámci domény.
- *Nonce* - hodnota, kterou generuje UA autentizující se na Registrar nebo SIP Proxy server.



Obrázek 2.4: *Digest Authentication (výpočet odpovědi) [4].*

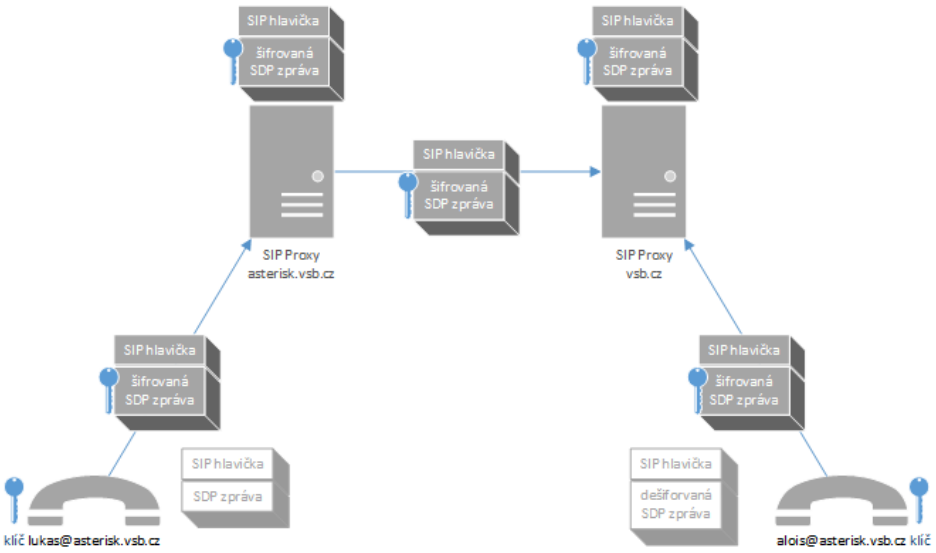
Autentizační pole v hlavičce může obsahovat více parametrů (například typ použitého šifrovacího algoritmu). Není-li žádný algoritmus definován, používá se hashovací funkce MD5. UA, inicializující požadavek, odpovídá na výzvu k ověření vypočtením nového, nyní již ověřeného požadavku. Výpočet hodnoty lze vidět na obrázku výše, který se skládá z následujících parametrů:

- *Username* - uživatelské jméno známé v UA, který zasílá žádost a v SIP serveru požadujícího autentizaci. Toto jméno je součástí SIP URI (např. Username z SIP URI sip:alice@voip.cz je alice).
- *Realm* - určuje pole rozsahu platnosti autentizace s platností v rámci domény.
- *Password* - je sdílené heslo známé oběma stranám účastnících se ověřování. Takovéto heslo je potřeba získat tzv. z jiného prostředí, což znamená, že protokol SIP samotný neposkytuje mechanismus pro definování hesla.
- *Nonce* - hodnota, kterou generuje UA autentizující se na Registrar nebo SIP Proxy server.
- *SIP method*: Původní SIP žádost, která inicializovala ověřování, se použije i pro výpočet hodnoty.
- *Request URI* - představuje URI cílové stanice. Pokud je například poslána žádost REGISTER, tak Request URI v tomto případě bývá adresa Registrar serveru.

Takto vypočtená hodnota (klientem) pro daný požadavek je odeslána zpět na server, který žádal o autentizaci. Server poté vypočte ze stejných parametrů svou vlastní hodnotu a porovná identičnost těchto hodnot. Mechanismus SIP Digest je nejčastěji používán v infrastruktuře s jednou doménou, kde mezi uživatelem a poskytovatelem dochází k výměně sdíleného hesla (může se jednat i o kombinaci uživatelského jména a hesla). Digest Authentication tedy zaručuje pouze integritu polí hlavičky (URI a Method) zahrnutých do výpočtů autentizace, nikoliv však integritu a autenticitu celé SIP zprávy.

S/MIME

Zkratka S/MIME znamená Secure/Multipurpose Internet Mail Extension. Tento protokol byl vyvinut za účelem bezpečnostního rozšíření standardu MIME, který slouží pro šifrování a podepisování těl e-mailových zpráv.



Obrázek 2.5: *SIP přes S/MIME.*

S/MIME není vázaná jen pro e-mailové zprávy. SIP zprávy mohou také obsahovat tělo MIME a proto je možné tento protokol využít i u SIP protokolu. S/MIME umožnuje SIP UA šifrování těl MIME zpráv SIP žádostech a odpovědích typu konec-konec bez ovlivnění hlaviček zpráv. Směrování takovýchto zpráv je tedy zachováno a síťové prvky nejsou schopny měnit obsah zpráv. S/MIME tedy zajišťuje důvěru a integritu zpráv směrem od volajícího až k volanému, což u SIP přes TLS nebylo možné provést.

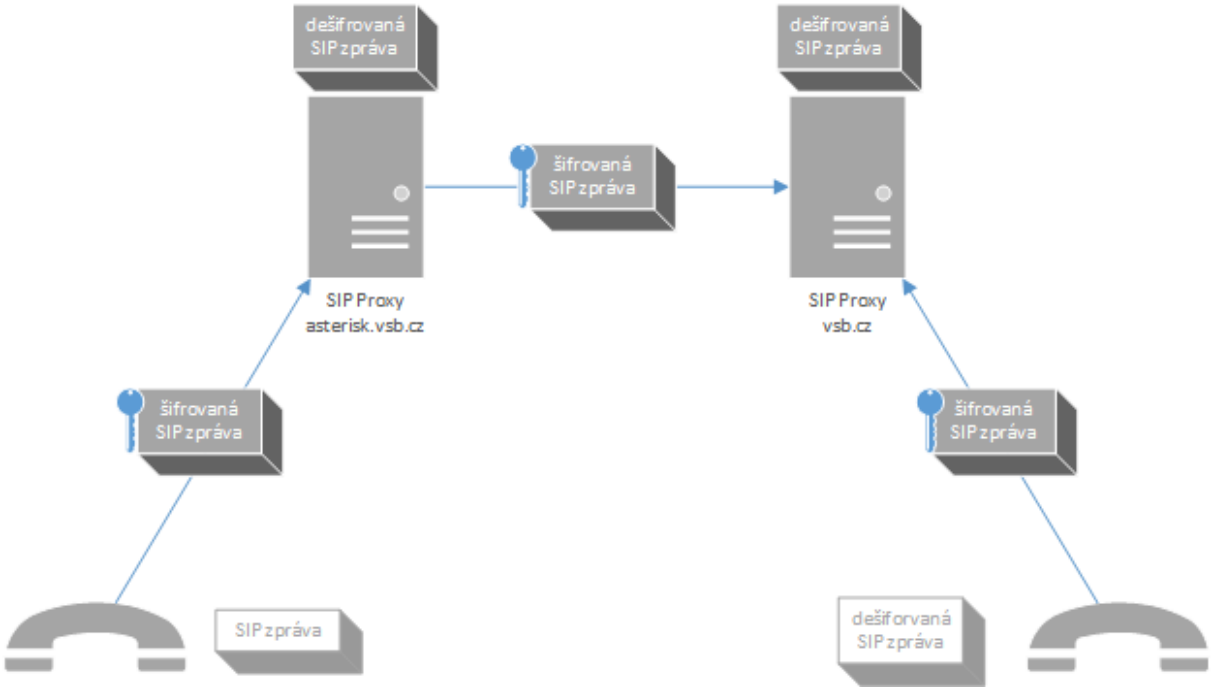
Tělo zprávy je podepsáno soukromým klíčem odesílatele a zašifrované veřejným klíčem příjemce. S/MIME využívá různé šifrovací algoritmy mezi, které patří RSA, 3DES, AES a různé hashovací algoritmy pro podpis jako například MD5 nebo SHA-1. Nutností pro podepsání zprávy jsou znalosti odesílatele o veřejném klíči příjemce pro účely ověření podpisu. Dále jsou zapotřebí soukromé klíče a certifikáty. SIP již sice obsahuje metody pro výměnu klíčů, ale využívá se PKI (Public Key Infrastructure), kde odesílatel musí znát veřejný klíč příjemce, aby byl schopen zašifrovat zprávu. Z toho je zřejmé, že zprávu odesílatele může dešifrovat pouze právoplatný příjemce a ne žádný jiný síťový prvek, přes které zpráva prochází. V některých případech nelze S/MIME použít, například pokud jsou některé síťové prvky (SIP Proxy) založeny na prohlížení či úpravě obsahu SIP zpráv (zejména SDP). Dalším řešením je tzv. SIP Tunneling, kde je celá SIP zpráva zabalena do těla S/MIME těla. Při použití této metody lze zabezpečit celou SIP zprávu a při dešifrování zprávy, obsahující i kopii hlavičky, může příjemce porovnat tuto hlavičku s nezašifrovanou a je možné tak odhalit případnou změnu zprávy.

SIPS (SIP Security)

SIPS neboli SIP přes TLS (Transport Layer Security), stanoví mechanismus pro hop-by-hop přenos SIP zašifrovaných zpráv a byl přejat od HTTPS. TLS pracuje na transportní vrstvě OSI modelu a je tvořen několika částmi:

- *Handshake Protocol* - Sjednávání a koordinace komunikačních parametrů (šifrovací nebo hashovací algoritmus, symetrický klíč).
- *Record Protocol* - Komprese a Fragmentace datagramů. Šifrování a hashování.

V průběhu TLS spojení vyžaduje UAC platný certifikát od UAS a tento certifikát musí být vygenerován důvěryhodnou certifikační autoritou CA. Na straně klienta se doporučuje mít certifikát ověřený tzv. kořenovým certifikátem. K tomu je potřeba PKI. Ve chvíli, kdy je celá zpráva zašifrována pomocí TLS scénáře, samotnou komunikaci lze zabezpečit pouze metodou hop-by-hop. Tzn. klient, který využívá protokolu SIPS, sestavuje zabezpečenou relaci s první SIP Proxy a šifrované zprávy jsou přenášeny mezi nimi.



Obrázek 2.6: *SIP přes TLS.*

Proxy následně dešifruje obdržené zprávy a sestavuje zabezpečenou relaci s dalším prvkem SIP infrastruktury. Síťové prvky potřebují získat přístup k polím v hlavičce (např. Via, From nebo To), pomocí dešifrování obsahu zprávy. Bez přístupu k těmto polím nelze zprávu směrovat sítí. SIPS tedy zaručuje důvěrnost a integritu SIP zpráv pouze mezi dvěma prvky sítě, ne na celé trase od začátku až po konec spojení.

Režim SIPS URI

Režimem SIPS URI (podobný HTTPS) je umožněno UAC požadovat zabezpečenou komunikaci po celý průběh signalizace. Příklad SIPS URI:

```
sips:lukas@asterisk.vsb.cz
```

Je-li vyžadována URI ve tvaru SIPS, je zaručeno, že každý síťový prvek zpracovávající tento požadavek přenáší data zabezpečeně. Dosáhne-li žádost cílové Proxy, místní zabezpečení a směrovací politiky jsou informovány o použití TLS (nepovinné).

IPSec

IPSec (Internet Protocol Security) je souborem algoritmů a mechanismů na síťové vrstvě OSI modelu. IPSec definuje přidání těchto mechanismů do standardní IP vrstvy. Definuje tak dva bezpečnostní mechanismy:

- *Autentizace* - Definuje vlastní původ dat, příjemce dat je schopen si ověřit, že IP paket pochází doopravdy od toho, kdo jej vyslal.
- *Šifrování* - Pouze hlavička IP paketu zůstává nezašifrovaná, zbytek paketu je zašifrován pomocí předem domluveného algoritmu (před přenosem dat se obě strany musejí dohodnout na způsobu šifrování).

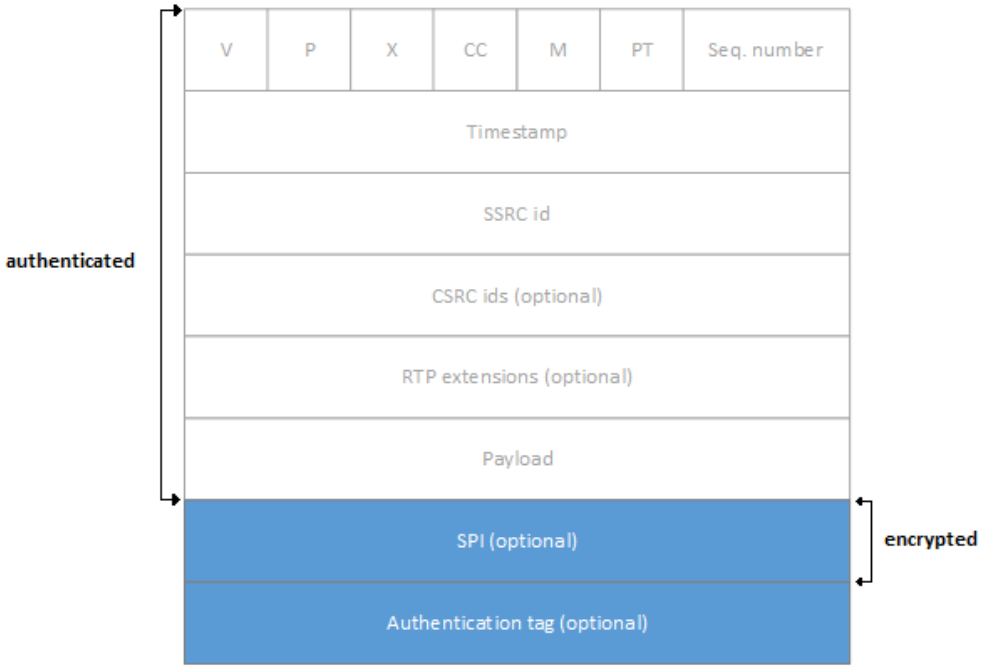
IPSec umožnuje důvěrný a autentizovaný přenos IP paketů a nejčastěji je využíván v architekturách, kde mají uživatelé nebo domény (mezi doménová důvěra) již mezi sebou vytvořenou důvěru. IPSec je většinou implementován na úrovni operačního systému uživatele nebo na bezpečnostní bráně, která se stará o zajištění důvěry a integrity pro celkový příchozí provoz na konkrétním rozhraní (VPN architektura). IPSec lze také využít pro hop-by-hop metodu komunikace. Nejčastější využití nachází IPSec tam, kde lze jen těžce implementovat zabezpečení přímo u SIP uživatelů. UA, kteří vlastní již vyměněné klíče se SIP Proxy ve vzdálenosti jednoho hopu, jsou také dobrými kandidáty pro použití IPSec.

2.4 Transportní protokol RTP a možnosti jeho zabezpečení

Nejdůležitějším faktorem, který je nutno potlačit při zabezpečování transportních protokolů, je zpoždění. Jelikož transportní protokoly přenáší data v reálném čase, je nutné docílit co nejmenšího možného zpoždění při implementování zabezpečovacích mechanismů. Tak jako u signalizačních protokolů je asi nejpoužívanější protokol RTP (Real-Time Transport Protocol). Ten sám nedisponuje žádnými bezpečnostní mechanismy či metodami. Proto byly definovány protokoly SRTP a ZRTP, jenž již tyto mechanismy obsahují. ZRTP byl navržen jako nadstavba pro SRTP, k ulehčení a implementaci.

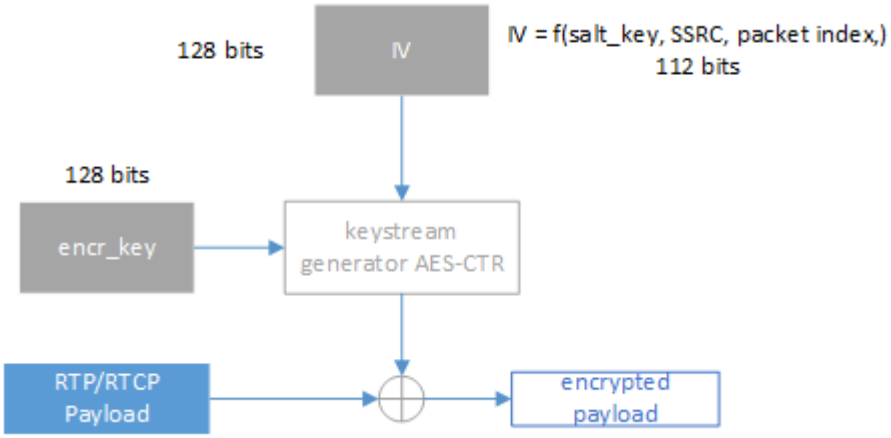
2.4.1 Protokol SRTP

SRTP rozšiřuje protokol RTP o bezpečnostní mechanismy integrity, ověření autenticity, zaručení důvěrnosti a ochrana proti přeposílání zpráv. Protokol SRTP pracuje na portu 5004 a SRTCP na portu 5005.



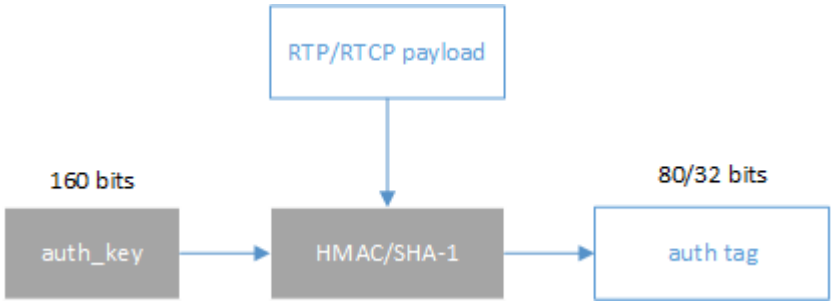
Obrázek 2.7: *SRTP paket.*

O důvěrnost přenášených dat se stará kryptografický primitiv AES, fungující jako generátor náhodných čísel. Vstupem do generátoru je inicializační vektor VI, který se skládá z kontrolního součtu salt key, SSRC a packet index. Výsledný klíč je aplikován pomocí logické operace XOR na nešifrovaný obsah paketu.



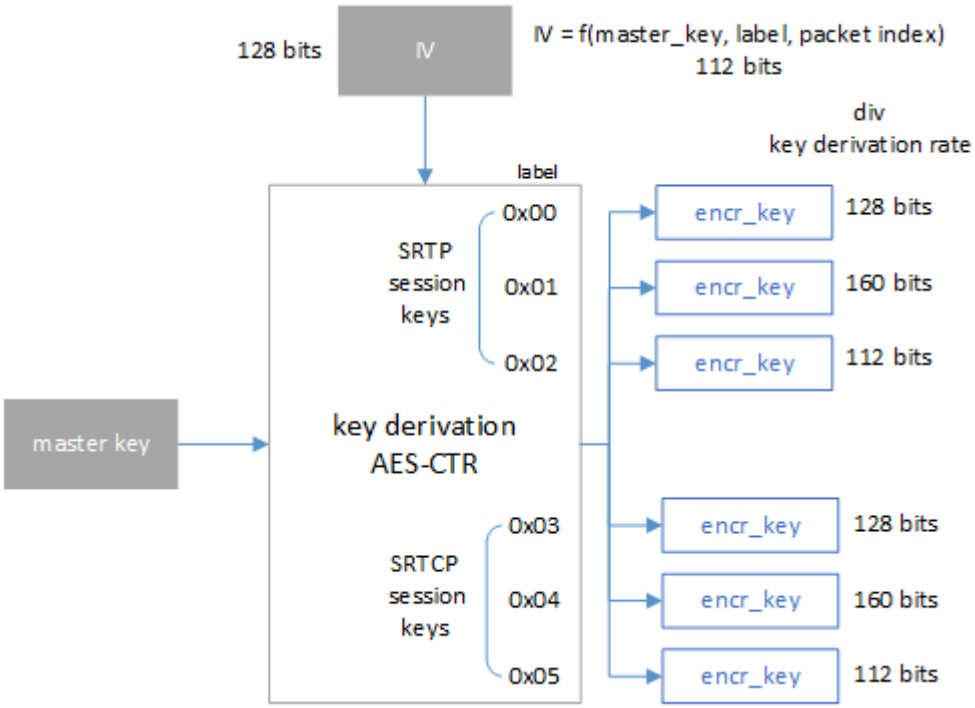
Obrázek 2.8: *Kódování obsahu paketu v SRTP.*

Algoritmus zajišťující autentizaci se nazývá HMAC-SHA 1. Tento algoritmus provádí součet z hlavičky a obsahu SRTP paketu a ten následně uloží do pole authentication tag.



Obrázek 2.9: *HMAC/SHA-1 princip fungování.*

Pro odlehčení zátěže přenosového pásma byl kontrolní součet zkrácen z původních 160 bitů na pouhých 80 bitů případně až na 32 bitů. Při použití zmiňovaných algoritmů je nutné, aby obě komunikující strany znaly tajný symetrický klíč neboli session key. V praxi se generují session klíče pomocí jednoho master klíče.



Obrázek 2.10: *Generování session klíčů pomocí master klíče.*

Na obrázku výše lze vidět, že master klíč může nabývat velikostí 128, 192 nebo 256 bitů. K jeho distribuci se používá protokol SDP, ten ale není chráněn proti potencionálním útokům (tak jako většina jiných protokolů). Proto se k němu navíc využívají bezpečnostní protokoly TLS nebo IPSec [7].

2.4.2 Protokol ZRTP

Jak již bylo zmíněno protokol ZRTP je nadstavbou pro již hojně používaný protokol SRTP. ZRTP rozšiřuje tento protokol o počáteční výměnu symetrických klíčů a o ochranu proti útokům "Man in the middle". Při výměně klíčů využívá Diffie-Hellmannův algoritmus. Ten funguje tak, že vypočte hash několika utajovaných hodnot spolu s krátkým autentizačním řetězcem, který nahlas přečtou obě volající strany. Takto vypočtená hodnota je použita jen jednou, její část se, ale ukládá a používá pro budoucí spojení.

ZRTP využívá tři fáze k zjištění a nastavení SRTP master klíče a přechodu na SRTP mód:

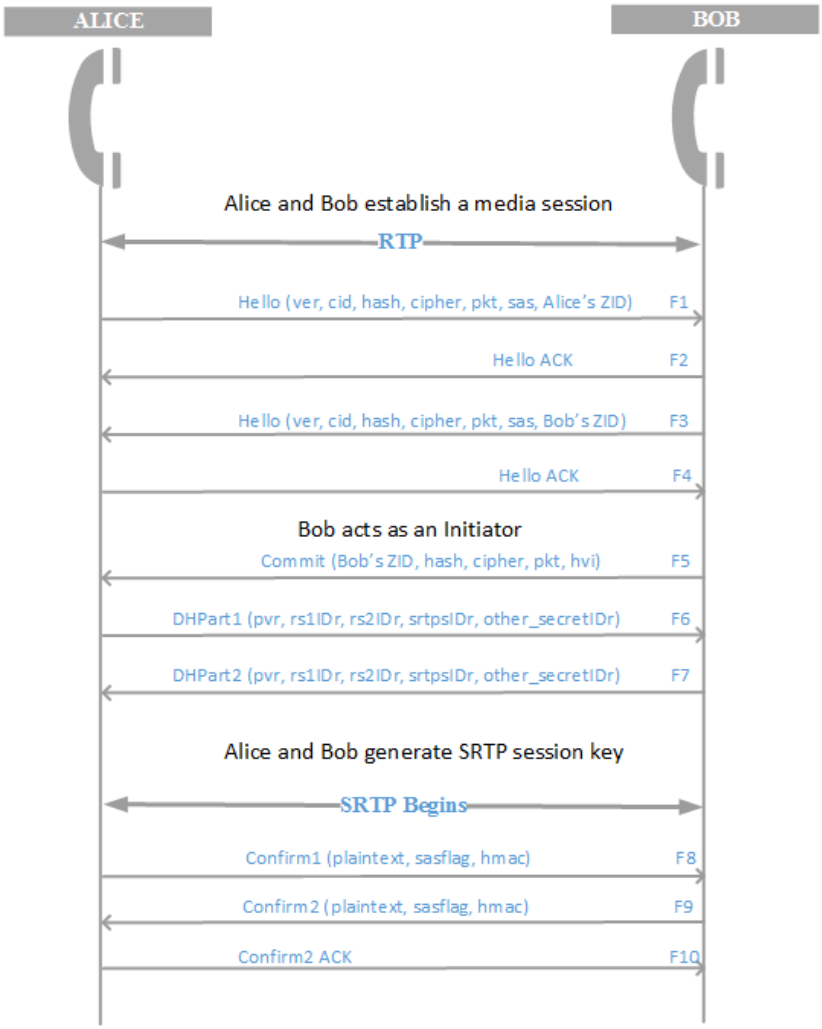
1. Detekce, podpory ZRTP implementací u účastníků.
2. Výměna symetrických klíčů.
3. Přepnutí do módu SRTP a úspěšné používání SRTP.

Během první fáze si oba ZRTP účastníci vymění informace o dohodnutých klíčích, šifrování a způsobech autentizace, které budou dále používat. ZRTP specifikace také definuje tyto šifrovací režimy:

- AES Counter Mode s 128 bitovou délkou klíče.
- AES Counter Mode s 256 bitovou délkou klíče.

K výměně klíčů se využívá ZRTP Diffie-Hellmanův algoritmus s různými módy:

- 3072 bitové Diffie-Hellman hodnoty.
- 4096 bitové Diffie-Hellman hodnoty.



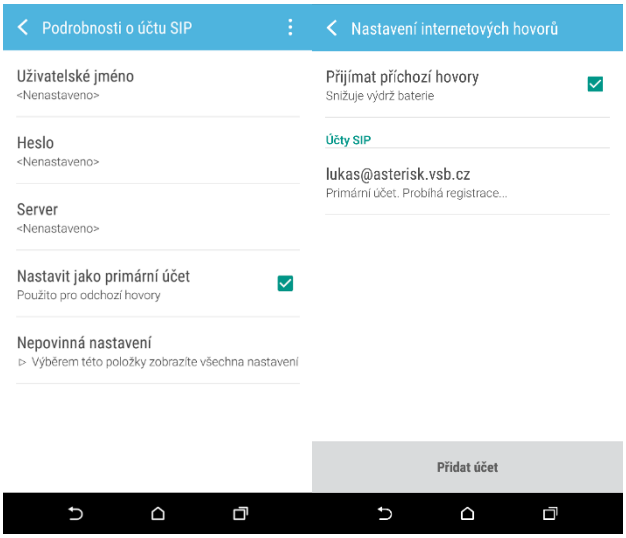
Obrázek 2.11: *Sestavení SRTP spojení s použitím ZRTP.*

3 Analýza současných SIP řešení pro OS Android

V současné době existuje již mnoho SIP klientů pro operační systém Android. Výběr začíná u nativního klienta, jenž je zabudován v systému Android až po propracovanější a komplexnější aplikace typu Zoiper. Veškeré vybrané a dále popisované aplikace jsou dostupné ke stažení zdarma na Google Play, u některých z nich je možné dokoupit funkce a rozšíření přímo v aplikaci.

3.1 Nativní Android klient

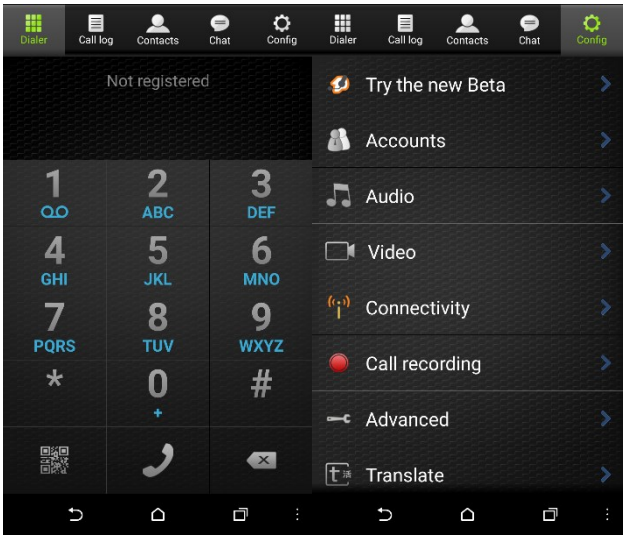
Již od samotného začátku nativního klienta v systému Android se nedostalo výrazných změn. Celková struktura klienta a nastavení v současné verzi systému Android 6.0 (Marshmallow) je prakticky totožná s první verzí klienta. U většiny mobilních telefonů se do nastavení SIP účtu klienta, lze dostat z nabídky volání a zde stisknutím tlačítka nastavení. Zobrazí se nabídka pro přidání SIP účtu a nastavení jeho parametrů (viz Obrázek 3.1). Po nastavení účtu, je nutné zaškrtnout políčko pro přijímání SIP hovorů (aktivace SIP klienta). Bohužel je u tohoto klienta absence šifrování. Naopak výhodou je zakomponování přímo v systému Android a tak lze očekávat vysokou spolehlivost a kompatibilitu s ostatními zařízeními využívající systém Android. U nativního klienta je však absence zabezpečení komunikace [8].



Obrázek 3.1: *Nativní SIP klient v OS Android 6.0.*

3.2 Zoiper IAX SIP VOIP Softphone

Aplikace Zoiper podporuje jak SIP tak i IAX (Inter Asterisk Exchange), což je nativní komunikační protokol vytvořený skupinou Asterisk. Mezi hlavní funkce aplikace Zoiper patří: integrace s OS Android, šifrování (TLS + SRTP), videohovory, nahrávání hovorů a zesílení reproduktoru. Jako placené doplňkové funkce, dostupné pro tzv. "Gold Users" jsou: konferenční hovory, přesměrování hovorů a šifrování za pomocí ZRTP. Dále je možné dokoupit podporu kodeku G.729 a nebo H.264. Grafické uživatelské rozhraní je možno vidět na obrázku níže [9].



Obrázek 3.2: *Zoiper*

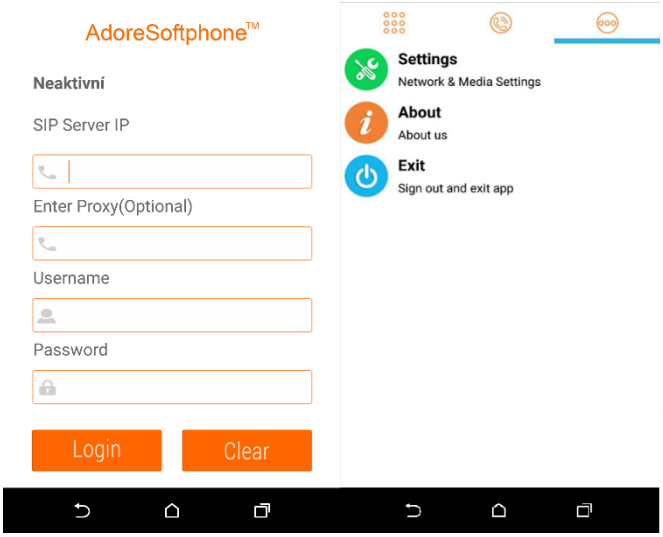
3.3 CSipSimple

Tato aplikace je vytvořena jako OpenSource pod licencí GPL. Výrobce udává vysoký výkon, podporu posílání zpráv pomocí protokolu SIMPLE, nahrávání hovorů, kodeky G.729 pro RTP a H.264 pro video hovory jsou zdarma, potlačení echa, menu se snadnou konfigurací pro začínající uživatele SIP, nastavitelné schémata GUI a šifrování SIP pomocí TLS a RTP pomocí SRTP nebo ZRTP. Celkový dojem z aplikace je velmi dobrý, u některých uživatelů se bohužel projevuje nekompatibilita s jejich zařízeními. Aplikace je zobrazena na obrázku 3.3 [10].

Obrázek 3.3: *CSipSimple*

3.4 Adore Softphone

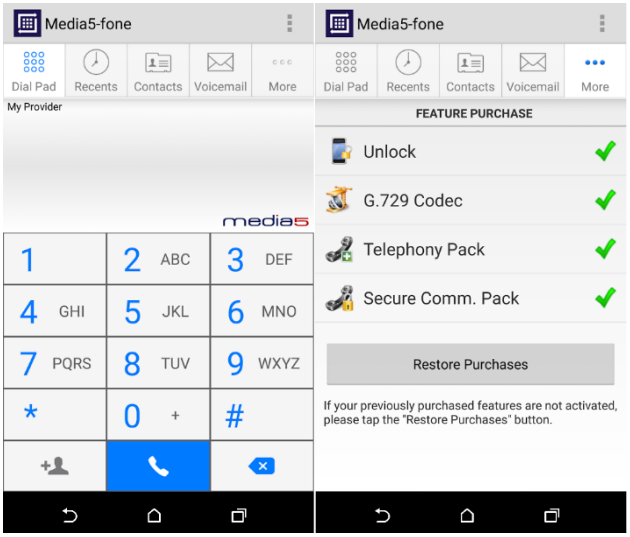
Patří k dalším nezpoplatněným aplikacím pro SIP komunikaci s vlídným GUI a kvalitním ovládáním (viz Obrázek 3.4). Tato aplikace mimo standardních funkcí nabízí: vícejazyčnou podporu, nahrávání, přesměrování a pozdržení hovorů, potlačení echa a ticha. Od června roku 2016 již podporuje i Android 6.0 (v 3.4.1) [11].



Obrázek 3.4: *Adore Softphone*

3.5 Media5-fone VoIP SIP Softphone

Aplikaci Media5-fone si z Google Play nainstalovalo mezi sto tisíc až půl milionu uživatelů. Patří tak k mezi nejrozšířenější. Aplikace nabízí mnoho funkcí jako například: podpora dialplan, nastavitelné vyzvánění, podpora hlasových zpráv, blacklist, podpora kodeku G.711, G.722 a G.729 Annex-B je možné dokoupit (viz Obrázek 3.5), šifrování SIP přes TLS a RTP pomocí SRTP. Mnoho uživatelů ve svých recenzích vyzdvihuje kvalitu hovorů a naopak nedostatek tzv. "skinů" neboli barevných schémat, mezi které patřil oblíbený černý, ale ten autoři vyřadili [12].

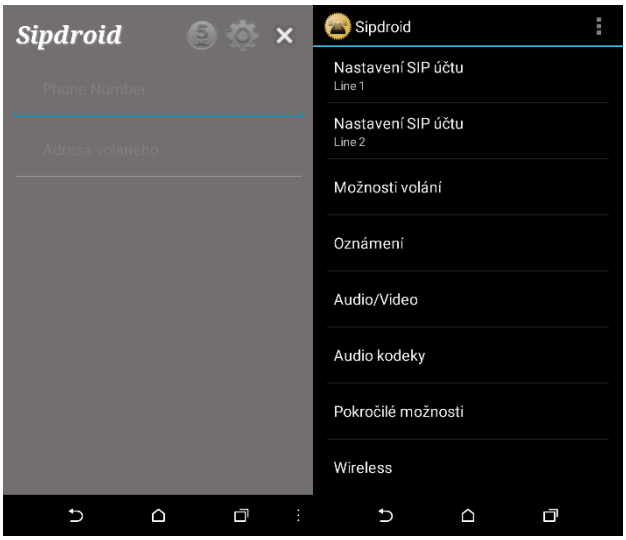


Obrázek 3.5: *Media5-fone*

3.6 Sipdroid

OpenSource aplikace Sipdroid je nejrozšířenější aplikace zaměřená na SIP komunikaci v Google Play. Vyzkoušelo si ji již mezi jedním až pěti milióny uživatelů. Aplikace nabízí, mimo standardní funkce, například: využití PBXes a VPN, nastavení dvou SIP účtů, podpora audio kodeků (G.722, PCMA, PCMU, speex, GSM a BV16), podpora video hovorů ve dvou rozlišeních (176x144 a

352x288) a možnost nastavení vlastního vyzvánění. Aplikace má intuitivní a jednoduché ovládání (viz Obrázek 3.6), ale design GUI je poněkud zastaralý [13].



Obrázek 3.6: *Sipdroid*

4 Návrh vlastního SIP klienta s možností zabezpečení

Před samotným započetím implementace klienta, bylo nutné zvolit programovací jazyk, ve kterém bude klient vytvořen. Jelikož je tato aplikace vyvíjena pro platformu Android, byl pochopitelně zvolen programovací jazyk Java. V dalším kroku bylo potřeba vyhledat nejvhodnější knihovnu, která bude zajišťovat podporu protokolu SIP v aplikaci. Zde již připadalo v úvahu více možností. Jako první se nabízela knihovna MjSip, jejíž nespornou výhodou je kompletní implementace v jazyce Java. Bohužel tato knihovna není nijak vyvíjena a podporována od dubna roku 2012, je tedy značně zastaralá, a proto bylo rozhodnuto o zvolení jiné knihovny. Další knihovnu v implementovanou v jazyce Java jsem nalezl na internetu, jedná se o knihovnu Vocal vyvíjené stejnojmennou firmou v New Yorku. Po emailové komunikaci se zástupci firmy, jsem byl obeznámen s podmínkami užití knihovny a její cenou, která však byla příliš vysoká i pro studentské účely. Z toho důvodu byla nakonec zvolena open source knihovna PJSIP. Tato knihovna je sice implementována v jazyce C, ale pomocí nástroje SWIG (Simplified Wrapper and Interface Generator) bylo možné PJSIP využít také pro programování v jazyce Java [14][15][16][17].

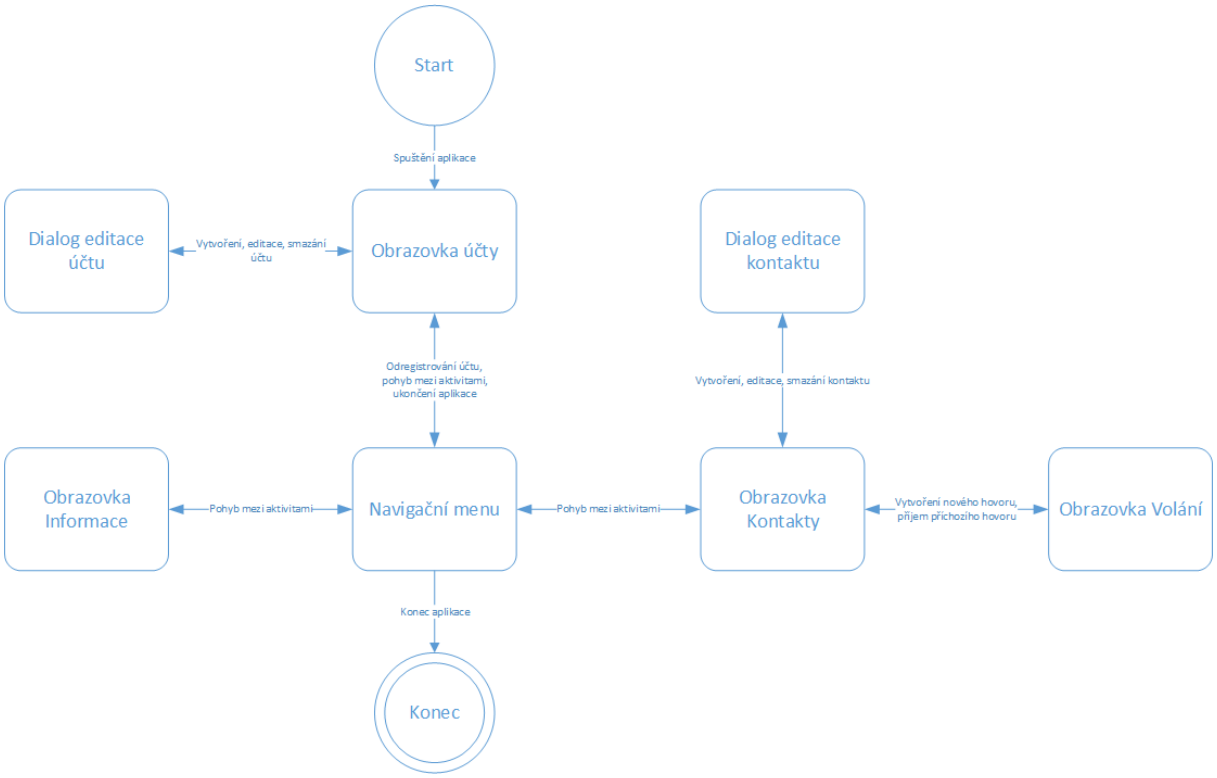
4.1 Vývojové prostředí

V srpnu roku 2015 bylo oznámeno ukončení podpory a vývoje pluginu pro vývojové prostředí Eclipse a proto bylo zvoleno jako vývojové prostředí aplikace Android Studio verze 2.3. K vývoji aplikací pro platformu Android využívá vývojové prostředí nástroje ADT (Android Development Toolkit), který obsahuje [18]:

- Nástroje pro kompilaci a ladění kódu.
- Nástroje pro převod Java bytecode do dx.
- Emulátor systému Android pro vybrané mobilní zařízení.
- Nástroje pro komunikaci s reálně připojeným mobilním zařízením.

4.2 Funkcionalita

Klient, dle zadání diplomové práce, musí umožňovat základní operace s klientskými účty a kontakty. Dále by měl obsahovat notifikace a stavové ikony pomocí, kterých bude uživatel obeznámen se současným stavem účtu. Největší pozornost by však měla být věnována možnosti zabezpečení hlasového hovoru včetně signalizačních informací. V rámci návrhu základní funkcionality aplikace byl kladen důraz na tyto aspekty a bylo také vytvořeno grafické schéma, popisující orientaci celou aplikací (viz obrázek 4.1).



Obrázek 4.1: *Grafická reprezentace funkcionality aplikace*

4.3 Návrh grafického uživatelského rozhraní

Základními prvky této aplikace jsou Android Aktivity a dialogy. Jedna aktivita prakticky znázorňuje jednu obrazovku ve výsledné aplikaci. Pomocí dialogů je potom uživatel vyzýván k zadávání jednotlivých údajů či k interakci se samotnou aplikací.

4.3.1 Aktivity

Pro všechny aktivity bylo navrhnuto rozložení GUI (Graphical User Interface). Ve všech aktivitách je společným prvkem v levé horní části logo aplikace a v pravé horní části navigační ikony.

První aktivita (viz obrázek 4.2, zleva) je AccountsActivity umožňující veškerou manipulaci s klientskými účty jako tvorba nových, editace a smazání stávajících anebo registrování a odregistrování účtu ze serveru. Tato aktivita obsahuje textové pole s oznámením o posledním stavu registrace účtu, v případě, že se účet registruje po delší dobu, pak je zobrazena načítací ikona, dále Jméno a URI účtu a ve spodní části obrazovky je situována ikona stavu registrace účtu.

Druhou aktivitou je BuddyActivity, která pracuje s kontakty a umožňuje vytáčení nových hlasových hovorů. Do horní části obrazovky je zasazen seznam kontaktů a k němu odpovídající obslužné tlačítka. V druhé části obrazovky je číselník s displejem pro možnost rychlého vytáčení, bez nutnosti vytváření nového kontaktu.

Ve třetí aktivitě InfoActivity je zobrazeno logo aplikace, informace o licenci a kontakt na vývojáře.

Obrázek 4.2: *AccountsActivity, BuddyActivity a InfoActivity GUI*

Poslední aktivitou je CallActivity, která je vyvolána vždy při odchozím nebo příchozím hovoru. Celá aktivita je tvořena textovým polem identifikujícím volajícího, ikonou zobrazující zabezpečený či nezabezpečený přenos hlasových dat, tlačítky pro příjem a odmítnutí hovoru a doplňkovými tlačítky pro hlasitý odposlech, regulaci hlasitosti hovoru a ztlumení mikrofonu. Tyto tlačítka jsou v rámci situace dynamicky měněny. Obrazovky příchozího, odchozího a probíhajícího hovoru jsou vyobrazeny na obrázku 4.3 (jdoucí zleva) [19].

Obrázek 4.3: *CallActivity GUI*

4.3.2 **Dialogy**

Samotné dialogy tvoří jakýsi mezistupeň komunikace mezi aplikací a uživatelem. Pomocí takovýchto dialogů Uživatel zadává vstupní informace pro aplikaci nebo je naopak upozorněn při zadání nesprávného vstupu. Toho je využito hlavně při tvorbě klientských účtů a kontaktů (viz obrázek 4.4). Možné využití pak také nachází v oznámeních uživateli, které jej nasměrují správným směrem např. při nepovolených akcích [20].

Obrázek 4.4: Dialogy GUI

4.4 Prezentace aplikace

Důležitým a často opomíjeným faktorem při tvorbě aplikací je samotná prezentace celé aplikace a její zviditelnění. Velké množství kvalitních aplikací neklade dostatečný důraz na tyto aspekty, ale pouze na funkčnost aplikace. Z tohoto důvodu aplikace dostala název caMEll. Ten je odvozen od anglického sousloví "Call Me" (v překladu "zavolej mi"). Název tak na první pohled připomíná jiné anglické slovo "Camel" (v překladu "velbloud"). Dle tohoto názvu bylo také vytvořeno logo aplikace (viz obrázek 4.4), kde je vyobrazen stojící velbloud s Headset zařízením na hlavě zvířete. Při propagaci výrobků se v oblasti marketingu velmi často využívá zábavná forma sdělení a humor, jenž byl zvolen jako hlavní složka prezentace výsledné aplikace.

Obrázek 4.5: *Logo aplikace*

5 Implementace vlastního SIP klienta s možností zabezpečení

V této kapitole bude mimo vývoji aplikace, věnován prostor podrobnému popisu, jak správně zkompilovat knihovnu PJSIP pro platformu Android. Knihovna PJSIP sice na webových stránkách výrobce obsahuje návod, jak zapojit knihovnu do vlastního projektu a jak provést kompilaci pro různé platformy, ale bohužel postrádá velké množství důležitých informací. Tyto informace a postupy byly získány právě při implementaci vlastního řešení. Z tohoto důvodu budou v rámci této kapitoly uvedeny získané informace pro budoucí vývojáře, aby při řešení problémů spojených s kompilací neztráceli čas.

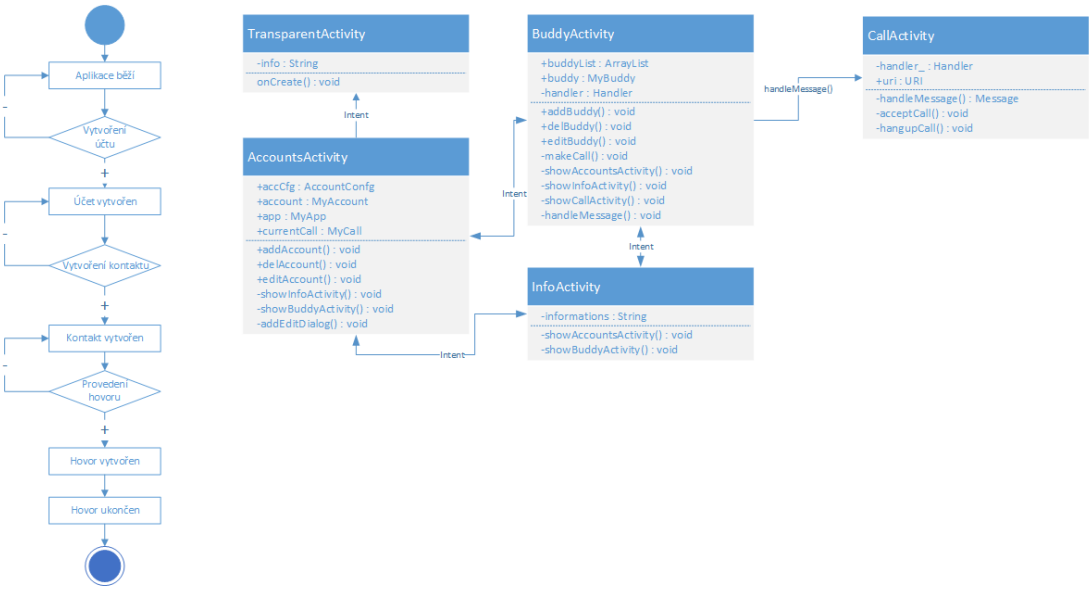
Výchozím bodem pro vytvoření této aplikace bylo API (Application Programming Interface) s názvem PJSUA2. PJSUA2 je objektově orientované API umožňující konstrukci SIP klientů v jazyce Java [21].

Jak již bylo popsáno v kapitole 4, pro aplikaci byl vybrán programovací jazyk Java. Jako vývojové prostředí bylo zvoleno Android Studio verze 2.3 a pro implementaci SIP jádra posloužila knihovna PJSIP. Projekt byl vytvořen s minimální podporou Androidu verze 4.0 (API 14). Aplikace byla vyvíjena předně na mobilním zařízení HTC One M8 (Android 6.0), ale po dokončení implementace, byla otestována na více zařízeních pro zjištění kompatibility a stability (viz kapitola 4) [33].

5.1 Diagram

Již před započetím vývoje aplikace bylo potřeba stanovit základní koncepci celé aplikace. Vznikl tak počáteční návrh 4 základních tříd. Tyto třídy zajišťují základní funkcionalitu celé aplikace. V průběhu vývoje vyvstala potřeba přidat třídu *TransparentActivity*, která se stará pouze o zobrazení oznámení důležitých informací uživateli. Pro jednodušší pochopení principu fungování aplikace a propojení jednotlivých tříd vznikl třídní diagram spolu s životním cyklem aplikace, který je znázorněn na obrázku 4.1. Základní koncept je pak tvořen z těchto tříd:

- *AccountsActivtity* – třída spravující klientské účty a jejich registraci.
- *BuddyActivity* – tato třída spravuje kontakty a umožňuje následné volání.
- *CallActivity* – třída, která zajištuje ovládání hlasových hovorů.
- *InfoActivity* – informativní třída, která zobrazuje obrazovku s informacemi o aplikaci.



Obrázek 5.1: *Životní cyklus aplikace a třídní diagram*

5.2 Kompilace knihoven

Ke kompilaci knihovny PJSIP pro operační systém Android je stanoven jako výchozí bod stručný návod vývojářů na webových stránkách produktu, který byl následně upraven do funkční podoby.

Základními požadavky pro kompilaci PJSIP s podporou šifrování jsou:

- Zdrojový projekt PjProject verze 2.6.
- Nástroj SWIG verze 3.0.12.
- Android SDK a NDK verze r13b [22].

Ze všeho nejprve je nutné stáhnout knihovnu PJSIP z *http://www.pjsip.org/download.htm*. Archív rozbalit a přesunout se do rozbalené složky.

5.2.1 Úprava zdrojové knihovny

Prvním krokem před samotným započetím kompilace je změna zdrojové knihovny */pjproject-2.6/pjmedia/src/pjmedia/transport_srtp.c*. Jedná se o část z knihoven zajišťujících podporu SRTP. Knihovna se nachází v PJSIP projektu, konkrétní změnu je nutné provést ve struktuře *crypto_suite* a zde zaměnit pořadí délky klíčů šifry AES, tak aby se struktura naplnila nejprve délkou klíče 128 bitů tzn., že musí začínat tímto:

```
static crypto_suite crypto_suites[] = {
    /* plain RTP/RTCP (no cipher & no auth) */
    {"NULL", NULL_CIPHER, 0, NULL_AUTH, 0, 0, 0, sec_serv_none},
#if defined(PJMEDIA_SRTP_HAS_AES_GCM_128) && \
    (PJMEDIA_SRTP_HAS_AES_GCM_128 != 0)
    /* cipher AES_GCM, NULL auth, auth tag len = 16 octets */
```

```
    {"AEAD_AES_128_GCM", AES_128_GCM, AES_128_GCM_KEYSIZE_WSALT,
     NULL_AUTH, 0, 16, 16, sec_serv_conf_and_auth,
&aes_gcm_128_openssl},

    /* cipher AES_GCM, NULL auth, auth tag len = 8 octets */
    {"AEAD_AES_128_GCM_8", AES_128_GCM, AES_128_GCM_KEYSIZE_WSALT,
     NULL_AUTH, 0, 8, 8, sec_serv_conf_and_auth,
&aes_gcm_128_openssl},
#endif
#if defined(PJMEDIA_SRTP_HAS_AES_CM_128) && \
    (PJMEDIA_SRTP_HAS_AES_CM_128 != 0)
    /* cipher AES_CM_128, auth HMAC_SHA1, auth tag len = 10 octets
*/
    {"AES_CM_128_HMAC_SHA1_80", AES_ICM, 30, HMAC_SHA1, 20, 10, 10,
     sec_serv_conf_and_auth},
    /* cipher AES_CM_128, auth HMAC_SHA1, auth tag len = 4 octets */
    {"AES_CM_128_HMAC_SHA1_32", AES_ICM, 30, HMAC_SHA1, 20, 4, 10,
     sec_serv_conf_and_auth},
#endif
...
};
```

Je možné, že tento postup nebude potřeba praktikovat u všech projektů implementujících knihovnu PJSIP. Problém totiž nastával ve chvíli, kdy Asterisk vyjednával délky klíčů šifry AES (128, 192 nebo 256 bitů).

5.2.2 Konfigurace knihovny OpenSSL pro Android

Dalším krokem je stažení knihovny OpenSSL, v tomto projektu byla použita verze 1.0.2k, z *https://www.openssl.org/source/*. Dále je nutné přiřadit systémovým proměnným správné cesty ke kompilátorům, NDK apod. a nakonec knihovnu zkompilovat následujícím způsobem **Chyba! Nenalezen zdroj odkazů.**:

```
export NDK=/…/android-ndk-r13b
$NDK/build/tools/make-standalone-toolchain.sh --platform=android-21
--toolchain=arm-linux-androideabi-4.9 --install-dir=`pwd`/android-
toolchain-arm
export TOOLCHAIN_PATH=`pwd`/android-toolchain-arm/bin
export TOOL=arm-linux-androideabi
```

```
export NDK_TOOLCHAIN_BASENAME=${TOOLCHAIN_PATH}/${TOOL}
export CC=$NDK_TOOLCHAIN_BASENAME-gcc
export CXX=$NDK_TOOLCHAIN_BASENAME-g++
export LINK=${CXX}
export LD=$NDK_TOOLCHAIN_BASENAME-ld
export AR=$NDK_TOOLCHAIN_BASENAME-ar
export RANLIB=$NDK_TOOLCHAIN_BASENAME-ranlib
export STRIP=$NDK_TOOLCHAIN_BASENAME-strip
export ARCH_FLAGS=
export ARCH_LINK=
export CPPFLAGS=" ${ARCH_FLAGS} -fpic -ffunction-sections -funwind-
tables -fstack-protector -fno-strict-aliasing -finline-limit=64 "
export CXXFLAGS=" ${ARCH_FLAGS} -fpic -ffunction-sections -funwind-
tables -fstack-protector -fno-strict-aliasing -finline-limit=64 -
frtti -fexceptions "
export CFLAGS=" ${ARCH_FLAGS} -fpic -ffunction-sections -funwind-
tables -fstack-protector -fno-strict-aliasing -finline-limit=64 "
export LDFLAGS=" ${ARCH_LINK} "

./Configure android-armv7
PATH=$TOOLCHAIN_PATH:$PATH make
```

Po úspěšném nakonfigurování knihoven pro platformu Android, je nezbytné vytvořit adresář /lib a do něj tyto knihovny nakopírovat.

```
mkdir lib
cp lib*.a lib/
```

5.2.3 Nastavení parametrů pro kompilaci

V hlavičkovém souboru */pjlib/include/pj/config_site.h* se nastavují veškeré parametry pro kompilaci knihovny PJSIP. Mezi tyto parametry například patří podpora SRTP, OpenSSL nebo také nastavení maximálního počtu hovorů či kontaktů. Většina z těchto parametrů je již předem definována v dokumentacích PJSIP, ale obsah samotného souboru je po rozbalení archívu prázdný. Proto je nutné jej upravit alespoň do následující podoby:

```
/* PJLIB nastavení*/
/* Zamezení podpory desetinné čárky */
#undef PJ_HAS_FLOATING_POINT
#define PJ_HAS_FLOATING_POINT                    0
```

```
#define PJ_HAS_SSL_SOCK                                 1
/*PJMEDIA nastavení*/
#define PJMEDIA_AUDIO_DEV_HAS_PORTAUDIO                 0
#define PJMEDIA_AUDIO_DEV_HAS_WMME                      0
#define PJMEDIA_AUDIO_DEV_HAS_OPENSL                    0
#define PJMEDIA_AUDIO_DEV_HAS_ANDROID_JNI               1
/* Vyladění Speex's základního nastavení pro nejlepší výkon */
#define PJMEDIA_CODEC_SPEEX_DEFAULT_QUALITY             5
/*PJSUA nastavení*/
#define PJSUA_DEFAULT_CODEC_QUALITY                     4
/* Nastavení trasakcí volání*/
#define PJSIP_MAX_TSX_COUNT                             31
#define PJSIP_MAX_DIALOG_COUNT                          31
#define PJSUA_MAX_CALLS                                 4
/* Další nastavení PJSUA */
#define PJSUA_MAX_ACC                                   100
#define PJSUA_MAX_PLAYERS                               100
#define PJSUA_MAX_RECORDERS                             100
#define PJSUA_MAX_CONF_PORTS (PJSUA_MAX_CALLS+2*PJSUA_MAX_PLAYERS)
#define PJSUA_MAX_BUDDIES                               1000
```

5.2.4 Kompilace PJSIP

V tuto chvíli jsou již všechny prerekvizity nastaveny a nainstalovány, tudíž je možné přistoupit k samotné kompilaci PJSIP pomocí příkazu:

```
NDK_TOOLCHAIN_VERSION=4.9 TARGET_ABI=armeabi-v7a ./configure-android
--use-ndk-cflags --with-ssl=/…/openssl-1.0.2k/
```

Pokud proběhla kompilace bez chyb, je možné pokračovat příkazy:

```
Make dep && Make
```

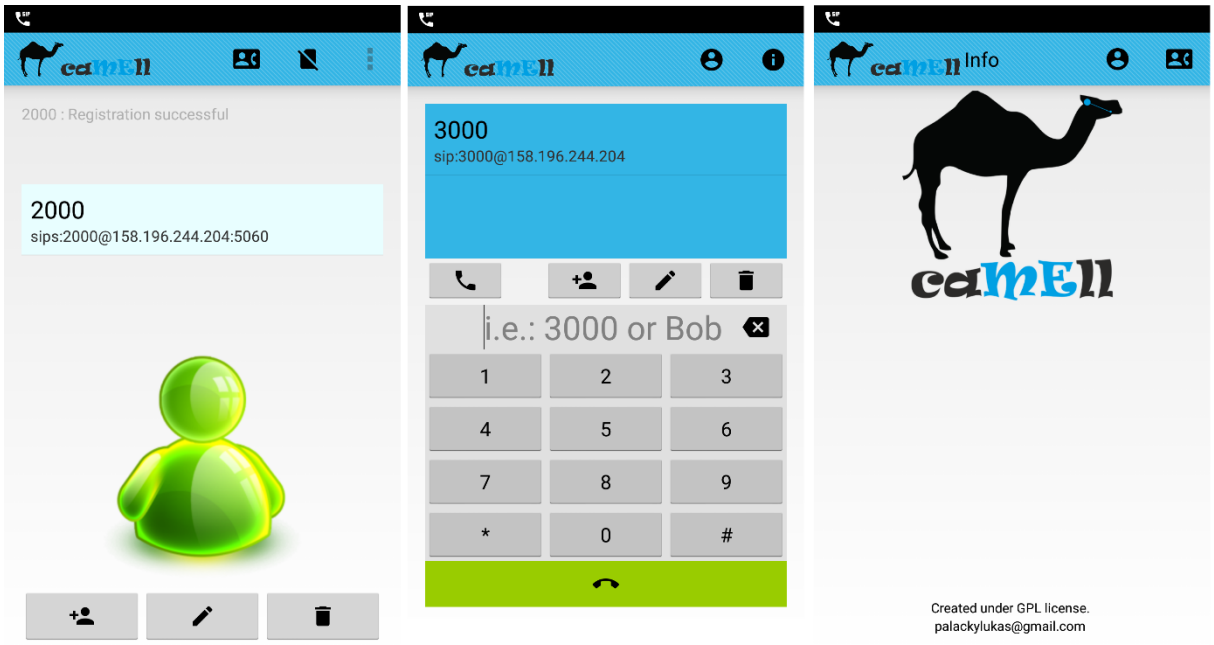
Jestliže se i tyto instrukce úspěšně provedly, pak je již knihovna PJSIP zkompilována. Pro vytvoření potřebných Java tříd a příkladů, které jsou použity pro řešení u platformy Android, se používá nástroj SWIG. Z adresáře */pjproject/pjsip-apps/src/swig/* stačí vyvolat příkaz *make*, o zbytek se již postará SWIG. V adresáři budou vytvořeny nové podadresáře s příklady PJSUA2 API, které bylo základem pro vypracování této práce.

5.3 Struktura aplikace

V kapitole 4 byl již popsán základní koncept aplikace a složení tříd a aktivit. Tento koncept byl dodržen a v následujících podkapitolách budou vyobrazeny reálné výsledky a popis funkčnosti jednotlivých částí aplikace.

5.3.1 Správa klientských účtů

Třída AccountsActivity je hlavní obrazovkou, která je vyvolána při spuštění aplikace. Na této obrazovce je možné vytvářet, editovat a mazat účty. U již zaregistrovaného účtu je možné sledovat současný stav registrace pomocí stavové ikony v dolní části obrazovky a v horní části obrazovky je formou textu zpracován poslední známý stav klientského účtu. K odregistrování účtu od serveru slouží ikona ve tvaru přeškrtnuté karty umístěna v navigačním menu (viz obrázek 5.2).



Obrázek 5.2: *Obrazovky účtů, kontaktů a informací*

5.3.2 Správa kontaktů

Obdobně jako správa klientských účtů funguje i správa kontaktů. Třída spravující kontakty a vytváření hovorů se jmenuje BuddyActivity. V horní části obrazovky je možné pomocí tlačítek s kontakty pracovat a na vybraný kontakt je možné vytočit nový hlasový hovor (viz Obrázek 5.2).

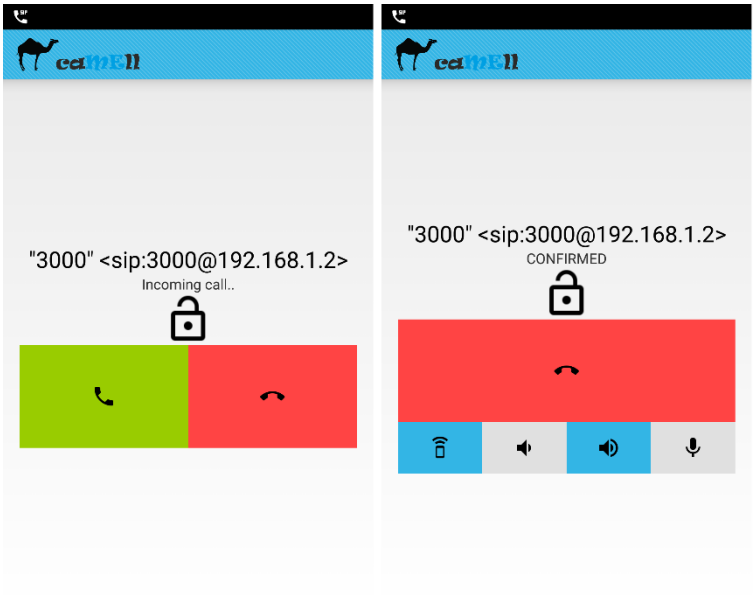
Další možností jak provést hovor je rychlou volbou přímo z číselníku integrovaného ve druhé části obrazovky.

5.3.3 Informace

Krátkou formou sdělení informací o licencování aplikace a kontaktu na vývojáře, využívá třída InfoActivity. Jedná se v podstatě o jednu obrazovku s logem a informacemi v její spodní části (viz obrázek 5.2).

5.3.4 Volání

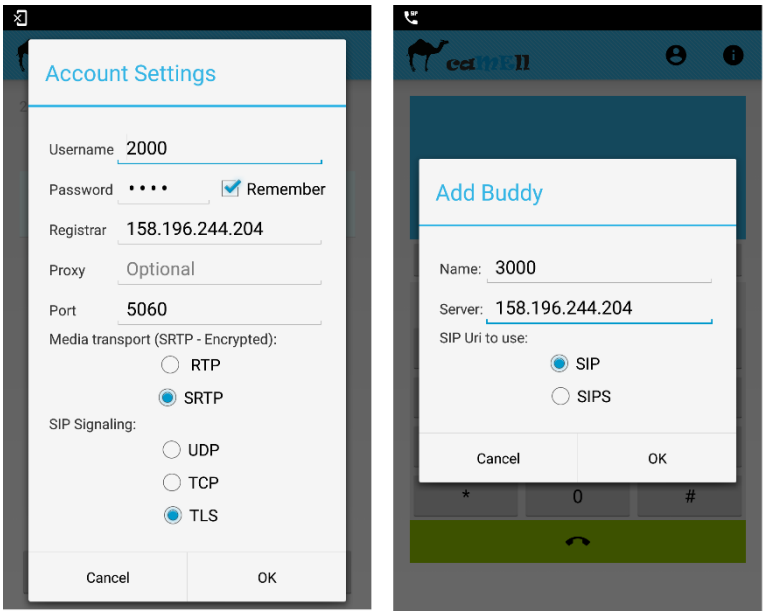
Pro manipulaci s příchozími, odchozími a sestavenými hovory pak slouží třída CallActivity. Tato třída implementuje sadu obslužných tlačítek a informací o hovoru. Mimo přijetí a odmítnutí hovoru je tedy možné v probíhajícím hovoru např. ztlumit mikrofon, ovládat hlasitost nebo zapnout hlasitý odposlech. Při každém hovoru je zobrazována identita volajícího (popř. volaného) a ikona zámku značící použití SRTP protokolu (viz obrázek 5.3).



Obrázek 5.3: *Obrazovky hovoru*

5.3.5 Dialogy

Pomocí dialogů jsou v aplikaci vytvářeny klientské účty a kontakty nebo také upozornění určená uživatelům (viz obrázek 5.4). Takovýto dialog je vyvolán vždy po stisknutí tlačítka.



Obrázek 5.4: *Dialogy*

6 Testování vlastního SIP klienta s možností zabezpečení

K testování aplikace bylo využito dvou Asterisk serverů ve verzi 13 s podporou SRTP a TLS. Jeden z těchto serverů byl zaveden na osobním notebooku a druhý byl poskytnut vedoucím diplomové práce v rámci školní sítě. Samotná aplikace byla testována na následujících zařízeních:

- *HTC One M8* [23]
    - *Verze operačního systému Android:* 6.0.
    - *Operační paměť:* 2 GB RAM.
    - *Procesor:* Quad-Core 2.3 GHz (Snapdragon 801).
- *Huawei P8 Lite* [25]
    - *Verze operačního systému Android:* 5.0.2.
    - *Operační paměť:* 2 GB RAM.
    - *Procesor:* Octa-Core 1.2 GHz (Cortex-A53).
- *Samsung Galaxy A Tab* [26]
    - *Verze operačního systému Android:* 5.0.
    - *Operační paměť:* 2 GB RAM.
    - *Procesor:* Quad-Core 1.2 GHz (Cortex-A53).
- *Lenovo Vibe S1* [27]
    - *Verze operačního systému Android:* 5.0.
    - *Operační paměť:* 3 GB RAM.
    - *Procesor:* Octa-Core 1.7 GHz (Cortex-A53).

Na všech výše zmiňovaných zařízeních aplikace vykazovala stejné známky chování, jediný rozdíl je v udílení oprávnění na základně rozdílných verzí systému Android (verze 5 a 6). Pozorovaná funkčnost a stabilita aplikace byla na všech zařízeních totožná.

6.1 Asterisk

Asterisk je open-source digitální pobočková ústředna (PBX), kterou vytvořil roku 1999 Mark Spencer. Trend nasazování IP telefonie je dlouhodobý a za tuto dobu se Asterisk stal jakýmsi fenoménem. Proto je také Asterisk od firmy Digium ™ v současné době nejrozšířenější volně dostupnou softwarovou realizací pobočkové ústředny [28].

Z těchto důvodů nebylo pochyb o využití právě Asterisku pro testování aplikace. V následujících podkapitolách budou uvedeny postupy pro instalaci a konfiguraci Asterisku, nastavení Dialplanu a generování TLS klíčů a certifikátů.

6.1.1 Instalace

Asterisk byl vytvořen pro operační systémy Linux. Tato instalace byla prováděna na operačním systému Ubuntu ve verzi 14.04 [29].

Nejprve je nutné otevřít terminál a do něj se přihlásit pod uživatelem ROOT. Poté provést update systému a pokračovat instalací knihoven a podpůrných balíčků. Poté je potřeba pomocí příkazu *wget* stáhnout současnou verzi Asterisku z webových stránek výrobce. Dále rozbalit stažený archív a přesunout se do rozbalené složky. Následuje spuštění konfiguračních skriptů s uvedenými parametry a

instalace příkazy *make && make install*, vytvoření tzv. ukázkových a konfiguračních souborů příkazy *make samples && make config*. Instalace je hotova, v tuto chvíli je možné spustit službu Asterisk, příkazem *asterisk* a *asterisk -rvvvvv* připojí uživatele do konzole serveru. Všechny příkazy ve správném pořadí jsou uvedeny níže.

```
sudo -i
apt-get update
apt-get install build-essential
apt-get install git-core subversion libjansson-dev libsqlite3-dev
apt-get install uuid-dev libsqlite3-dev libc-dev libncurses-dev
apt-get install libssl-dev libxml2-dev uuid-dev g++ zlib1g-dev
apt-get install unixodbc-dev librstp-dev libmyodbc
wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-13-
current.tar.gzgunzip asterisk-13-current.tar.gz
tar –xzvf asterisk-13-current.tar
cd asterisk-13.x.x/
./contrib/scripts/install_prereq install
./bootstrap.sh
./configure --with-pjproject-bundled
make && make install
make samples && make config
asterisk
asterisk -rvvvvvvv
```

6.1.2 Konfigurace Asterisku

V následujícím textu je popsán obsah konfiguračního souboru *sip.conf*, který je umístěn v adresáři */etc/asterisk/*. Před první konfigurací konfiguračních souborů je dobrým zvykem vygenerovaný obsah konfiguračního souboru vymazat a nahradit vlastní konfigurací. Takto lze účinně předejít problémům, ke kterým často v budoucnu dochází.

```
[general]                   ;obecné nastavení
tlsenable=yes               ;podpora protokolu TLS
tlsbindaddr=0.0.0.0         ;naslouchání TLS na všech IP adresách
tlscertfile=/home/…/asterisk/keys/asterisk.pem ;Asterisk certifikát
tlscafile=/home/…/keys/ca.crt ;certifikát certifikační autority
tlscipher=ALL               ;podpora šifrovacích algoritmů TLS
tlsclientmethod=tlsv1       ;verze TLS
udpbindaddr=0.0.0.0         ;naslouchání UDP na všech IP adresách
tlsdontverifyserver=yes     ;TLS neověřuje server
```

```
directmedia=no              ;přímý přenos médií (mezi dvěma uzly)
[2000]                      ;název klientského účtu
type=friend                 ;typ účtu
secret=1234                 ;heslo účtu
host=dynamic                ;dynamicky/staticky přidělená IP adresa
transport=tls               ;protokol pro přenos SIP informací
encryption=yes              ;šifrování médií (SRTP)
context=default             ;kontext, který má být použit v dialplanu

[2001]                      ;název klientského účtu
type=friend                 ;zařízení s obousměrnou komunikací
secret=1234                 ;heslo účtu
host=dynamic                ;dynamicky/staticky přidělená IP adresa
transport=tls               ;protokol pro přenos SIP informací
encryption=yes              ;šifrování médií (SRTP)
context=default             ;kontext, který má být použit v dialplanu
```

6.1.3 Konfigurace Dialplanu

Dialplan, někdy také označován jako číslovací plán, vytáčecí plán apod., určuje, jak se zachází s relacemi, které vznikají v Asterisku nebo přicházejí zvenčí. Je hlavní pohonnou jednotkou Asterisku a rozhoduje se v něm o tom, jaké činnosti budou vykonány při příchozím nebo odchozím volání. Dělí se do 4 základních koncepcí: kontexty, pobočky, priority a aplikace. Dialplan se nachází v souboru *extensions.conf* v adresáři */etc/asterisk*, ale nekonfiguruje se přímo tento soubor. Ke konfiguraci se využívá skriptovacího jazyka AEL a konkrétně tedy souboru *extensions.ael*. Na následující ukázce je nakonfigurován základní Dialplan, který pouze zřizuje hovory mezi všemi stanicemi.

```
context default {           ;název kontextu
  _X. => {                  ;schéma, pro které je kontext platný
    Dial(SIP/${EXTEN});     ;akce (v tomto případě vytočení čísla)
  };
};
```

6.1.4 Generování TLS certifikátů a klíčů

TLS vytváří šifrované stavové spojení (Session), ve kterém se na začátku vždy provádí Handshake, kdy si obě strany domluví parametry spojení (spolu s šifrou). Certifikát na serverové straně se využívá vždy (dochází také k ověření identity). Klientský certifikát se použít může, ale nemusí. Klient generuje Pre-Master Secret (tajemství), které zašifruje veřejným klíčem z certifikátu

serveru a odešle. Je zde tedy využita asymetrická kryptografie (šifrování s veřejným klíčem) a rozšifrovat data může pouze server, který vlastní privátní klíč [30].

Klient i server provedou definovanou sérii kroků nad Pre-Master Secret a tím vytvoří Master Secret. Na tento klíč použijí dohodnutou Hashovací funkci a vytvoří tak Session Key. Session Key je symetrický klíč, využívaný k šifrování a dešifrování spojení. Vlastní komunikace se pak tedy šifruje pomocí symetrické kryptografie s využitím Session Key.

Pro vygenerování klientských, případně serverových klíčů je možné postupovat následujícím způsobem.

Serverová část

Nejdříve je potřeba vytvořit adresář pro klíče příkazem *mkdir /etc/asterisk/keys*. K vytvoření Self-Signed (podepsané sama sebou) autority a certifikátu pro Asterisk se používá skript *ast_tls_cert* umístěný v kořenovém adresáři Asterisku */contrib/scripts*. Spuštění skriptu se provádí s následujícími parametry:

- "*-C*" se využívá k definování hosta (IP adresou nebo doménovým jménem).
- "*-O*" značí název organizační jednotky.
- "*-d*" specifikuje výstupní adresář pro uložení klíčů.

```
./ast_tls_cert -C „IP adresa Asterisku“ -O "Název organizace" -d
/etc/asterisk/keys
```

Poté bude uživatel postupně vyzván k následujícím akcím:

1. K zadání bezpečnostní fráze pro */etc/asterisk/keys/ca.key*.

2. Předchozí bod vytvoří certifikát */etc/asterisk/keys/ca.crt*.

3. Opětovné zadání bezpečnostní fráze, které vytvoří klíč */etc/asterisk/keys/asterisk.key*.

4. Automaticky bude také vygenerován certifikát */etc/asterisk/keys/asterisk.crt*.

5. Poslední výzva k zadání bezpečnostní fráze vytvoří certifikát */etc/asterisk/keys/asterisk.pem* (tento certifikát je vytvořen kombinací klíče *asterisk.key* a certifikátu *asterisk.crt*)

Klientská část

Nyní je možné vytvořit klientský certifikát s použitím stejného skriptu, ale jinými parametry:

- "*-m client*" tento parametr specifikuje typ certifikátu (klientský).
- "*-c /etc/asterisk/keys/ca.crt*" definuje jaká certifikační autorita má byt použita.
- "*-k /etc/asterisk/keys/ca.key*" poskytuje klíč, který má být použit.
- "*-C*" udává jméno nebo IP adresu klienta.
- "*-O*" značí název organizační jednotky.
- "*-d*" specifikuje výstupní adresář pro uložení klíčů.
- "*-o*" název vygenerovaného klíče.

```
./ast_tls_cert -m client -c /etc/asterisk/keys/ca.crt -k
/etc/asterisk/keys/ca.key -C "IP adresa nebo doménové jméno serveru"
-O "Název organizace" -d /etc/asterisk/keys -o phone
```

V následujícím kroku bude uživatel vyzván k zadání bezpečnostní fráze z dřívějška pomocí, kterého bude zpřístupněn klíč */etc/asterisk/keys/ca.key*. Po úspěšném provedení všech předešlých kroků bude složka */etc/asterisk/keys* obsahovat následující soubory:

- asterisk.crt.
- asterisk.csr.
- asterisk.key.
- asterisk.pem.
- phone.crt.
- phone.csr.
- phone.key.
- phone.pem.
- ca.cfg.
- ca.crt.
- ca.key.
- tmp.cfg.

Klientská aplikace pak využívá certifikáty *phone.pem* a *ca.crt*, které je nutné přesunout do požadovaného adresáře.

6.2 Wireshark

Po dokončení aplikace bylo nutné prakticky zkontrolovat, jestli se data přenášená sítí opravdu šifrují a využívají protokoly TLS a SRTP anebo se aplikace pouze navenek jeví zabezpečeně. Z toho důvodu bylo nutné použít analyzátor síťového provozu a odchytit síťový provoz během komunikace klienta s Asterisk serverem [31].

Pro tyto účely byl zvolen program Wireshark a Tcpdump. Wireshark je zdaleka nejpoužívanější protokolový analyzátor a zároveň tzv. paketový Sniffer (v překladu štěnice). Mezi jeho přednosti patří grafické uživatelské rozhraní, jednoduchá obsluha a podpora prakticky všech, v současné době, používaných síťových komunikačních protokolů. Wireshark byl použit pro analýzu zachyceného provozu, samotný provoz byl zachycen pomocí programu Tcpdump přímo na serveru a uložen ve formátu PCAP. Obsluha programu Tcpdump je jednoduchá, pomocí jediného příkazu lze definovat na jakém rozhraní má program naslouchat a provoz následně uloží do souboru. Použitelný příkaz, který bude naslouchat na síťovém rozhraní eth0 a uloží výsledný provoz do souboru provoz.pcap, vypadá následovně [32]:

```
tcpdump -w provoz.pcap -i eth0
```

6.2.1 SIP přes UDP

V prvním případě byl zachycen provoz, kdy klient používal pro přenos hlasových dat protokol SRTP, ale pro přenos signalizačních dat protokol UDP. Což ve výsledku znamená, že jsou sice hlasové informace šifrovány, ale signalizační ne (viz obrázek 6.1). Je tak prakticky možné odchytit celkový provoz na síti (viz obrázek 6.2 a 6.3). Jelikož se v protokolu SDP vyjednávají Master Key, pomocí nichž se následně generují všechny potřebné klíče sezení, je nutné signalizační data zabezpečit proti odposlechu také.

```
Session Initiation Protocol (INVITE)
  Request-Line: INVITE sip:1999@158.196.244.204 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 158.196.194.227:5065;rport;branch=z9hG4bKPj3ceea90a-9ec9-43d9-bea9-3dd52fa79012
    Max-Forwards: 70
    From: sip:2000@158.196.244.204;tag=81c4dabf-1bb5-4d0b-a6ae-cc3e6fd0e546
    To: sip:1999@158.196.244.204
    Contact: <sip:2000@158.196.194.227:5065;ob>;+sip.ice
    Call-ID: e0680b3b-4716-486b-ac35-464cde02f697
    CSeq: 15430 INVITE
    Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS
    Supported: replaces, 100rel, timer, norefersub
    Session-Expires: 1800
    Min-SE: 90
    User-Agent: HTC 2.6
    Authorization: Digest username="2000", realm="asterisk", nonce="1333ea13", uri="sip:1999@158.196.244.204", response="75cdfe4dddcd9b732f5e9853bce6fb0e", algorithm=MD5
    Content-Type: application/sdp
    Content-Length:  1187
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): - 3698047120 3698047120 IN IP4 158.196.194.227
      Session Name (s): pjmedia
      Bandwidth Information (b): AS:84
      Time Description, active time (t): 0 0
      Session Attribute (a): X-nat:8
      Media Description, name and address (m): audio 60580 RTP/SAVP 98 97 99 104 3 0 8 9 96
      Connection Information (c): IN IP4 158.196.194.227
      Bandwidth Information (b): TIAS:64000
      Media Attribute (a): rtcp:60581 IN IP4 158.196.194.227
      Media Attribute (a): sendrecv
      Media Attribute (a): rtpmap:98 speex/16000
      Media Attribute (a): rtpmap:97 speex/8000
      Media Attribute (a): rtpmap:99 speex/32000
      Media Attribute (a): rtpmap:104 iLBC/8000
      Media Attribute (a): fmtp:104 mode=30
      Media Attribute (a): rtpmap:3 GSM/8000
      Media Attribute (a): rtpmap:0 PCMU/8000
      Media Attribute (a): rtpmap:8 PCMA/8000
      Media Attribute (a): rtpmap:9 G722/8000
      Media Attribute (a): rtpmap:96 telephone-event/8000
      Media Attribute (a): fmtp:96 0-16
      Media Attribute (a): crypto:1 AES_CM_128_HMAC_SHA1_80 inline:A9kZq5DEYcf2WXt0X5SJ61afy7otfr+7GjSZgg8u
      Media Attribute (a): crypto:2 AES_CM_128_HMAC_SHA1_32 inline:fvQ9mC45X8X/HT1SKLARJMgSLrXEUyB2DlaiSsEp
      Media Attribute (a): crypto:3 AES_256_CM_HMAC_SHA1_80 inline:lwxRtkPwo/mjNjCmeloO0eAiXlga11QTNLQNtl1HvwesKD0USauI2w9+T4qD/Q==
      Media Attribute (a): crypto:4 AES_256_CM_HMAC_SHA1_32 inline:3e0154IYLw1VPokIwvMw6SH0Q2lzsn7IpRR581Y4U60TkmeBXrgteXvcUg8+Zg==
      Media Attribute (a): ice-ufrag:6ea0b8b2
      Media Attribute (a): ice-pwd:3068a3f1
      Media Attribute (a): candidate:H9ec4c2e3 1 UDP 2130706431 158.196.194.227 60580 typ host
      Media Attribute (a): candidate:Hc0a80103 1 UDP 2130706431 192.168.1.3 60580 typ host
      Media Attribute (a): candidate:H9ec4c2e3 2 UDP 2130706430 158.196.194.227 34434 typ host
      Media Attribute (a): candidate:Hc0a80103 2 UDP 2130706430 192.168.1.3 34434 typ host
```

Obrázek 6.1: Zachycený obsah protokolu SIP a SDP

No.	Time	Source	Destination	Protocol	Length	Info
701	20.492256	158.196.194.227	158.196.244.204	SIP/SDP	388	Request: INVITE sip:1999@158.196.244.204 |
702	20.492770	158.196.244.204	158.196.194.227	SIP	628	Status: 401 Unauthorized |
704	20.525636	158.196.194.227	158.196.244.204	SIP	400	Request: ACK sip:1999@158.196.244.204 |
706	20.526357	158.196.194.227	158.196.244.204	SIP/SDP	555	Request: INVITE sip:1999@158.196.244.204 |
707	20.527431	158.196.244.204	158.196.194.227	SIP	610	Status: 100 Trying |
708	20.528560	158.196.244.204	158.196.194.227	SIP/SDP	1007	Request: INVITE sip:1999@158.196.194.227:56263;ob |
710	20.557315	158.196.194.227	158.196.244.204	SIP	337	Status: 100 Trying |
711	20.584701	158.196.194.227	158.196.244.204	SIP	519	Status: 180 Ringing |
712	20.584888	158.196.244.204	158.196.194.227	SIP	626	Status: 180 Ringing |
851	24.126324	158.196.194.227	158.196.244.204	SIP/SDP	956	Status: 200 OK |
852	24.126608	158.196.244.204	158.196.194.227	SIP	472	Request: ACK sip:1999@158.196.194.227:56263;ob |
853	24.127142	158.196.244.204	158.196.194.227	SIP/SDP	1044	Status: 200 OK |
937	24.626751	158.196.244.204	158.196.194.227	SIP/SDP	1044	Status: 200 OK |
944	24.658985	158.196.194.227	158.196.244.204	SIP	405	Request: ACK sip:1999@158.196.244.204:5060 |
959	24.735118	158.196.194.227	158.196.244.204	SIP/SDP	1061	Request: INVITE sip:1999@158.196.244.204:5060, in-dialog |
960	24.735286	158.196.244.204	158.196.194.227	SIP	625	Status: 100 Trying |
961	24.735372	158.196.244.204	158.196.194.227	SIP/SDP	997	Status: 200 OK |
973	24.774752	158.196.194.227	158.196.244.204	SIP	405	Request: ACK sip:1999@158.196.244.204:5060 |
3697	38.458581	158.196.194.227	158.196.244.204	SIP	462	Request: BYE sip:2000@158.196.244.204:5060 |
3698	38.458921	158.196.244.204	158.196.194.227	SIP	557	Status: 200 OK |
3699	38.460132	158.196.244.204	158.196.194.227	SIP	650	Request: BYE sip:2000@158.196.194.227:5065;ob |
3700	38.482778	158.196.194.227	158.196.244.204	SIP	362	Status: 200 OK |

Obrázek 6.2: *SIP protokol, šířený přes UDP*

No.	Time	Source	Destination	Protocol	Length	Info
845	24.109348	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22080, Time=160, Mark
846	24.110803	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22081, Time=320
847	24.110811	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22082, Time=480
848	24.116934	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22083, Time=640
849	24.116942	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22084, Time=800
850	24.117081	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22085, Time=960
854	24.138651	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22086, Time=1120
855	24.139613	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22087, Time=1280
857	24.173045	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22088, Time=1440
858	24.173054	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x335B5969, Seq=22089, Time=1600
860	24.190103	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x5197DF8B, Seq=7137, Time=160, Mark
861	24.190180	158.196.244.204	158.196.194.227	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x633508EB, Seq=25330, Time=160
862	24.190242	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x5197DF8B, Seq=7138, Time=320
863	24.190248	158.196.194.227	158.196.244.204	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x5197DF8B, Seq=7139, Time=480
864	24.190313	158.196.244.204	158.196.194.227	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x633508EB, Seq=25331, Time=320

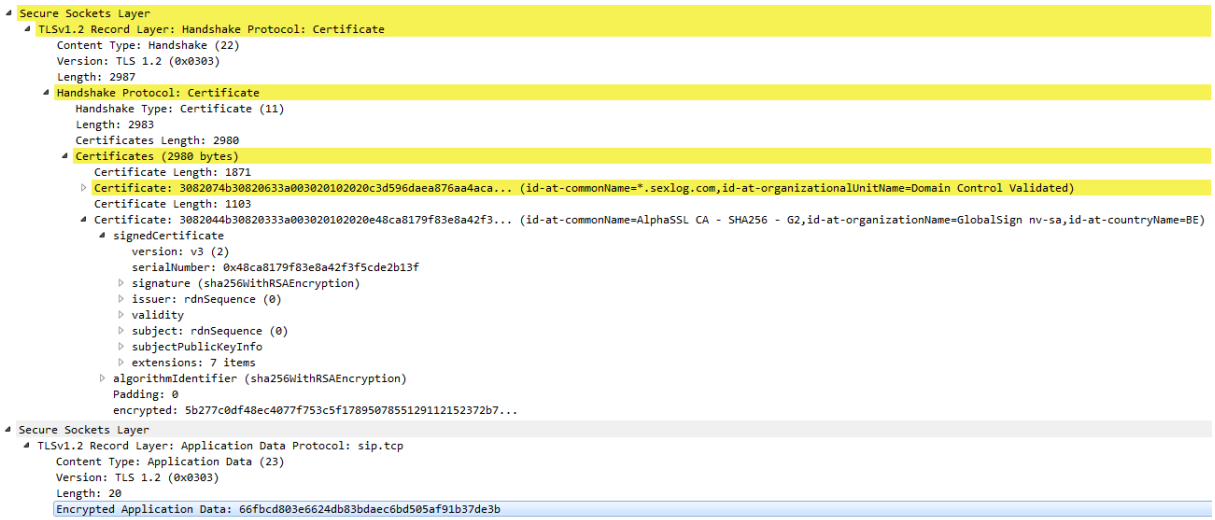
Obrázek 6.*3: SRTP protokol, šířený přes UDP*

6.2.2 SIP přes TLS

V druhém případě již byla situace podstatně lepší. Klient byl nastaven tak, aby používal pro přenos signalizačních dat protokol TLS. Pro přenos médií byl opět použit protokol SRTP. Při analýze síťového provozu, byly v tomto případě zachyceny pouze pakety protokolu TLS (viz obrázek 6.4). Z takto odchycených paketů je možné vyčíst například použité šifrování (viz obrázek 6.5), ale protokol SDP je již zabezpečen proti odposlechu a není tak možné zachytit pakety protokolu SRTP.

No.	Time	Source	Destination	Protocol	Length	Info
78	1.159779	158.196.244.131	192.175.111.53	TLSv1.2	571	Client Hello
80	1.275454	192.175.111.53	158.196.244.131	TLSv1.2	1514	Server Hello
84	1.276142	192.175.111.53	158.196.244.131	TLSv1.2	539	Certificate
86	1.277274	158.196.244.131	192.175.111.53	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
88	1.404163	192.175.111.53	158.196.244.131	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
89	1.404730	158.196.244.131	192.175.111.53	TLSv1.2	335	Application Data
93	1.516181	192.175.111.53	158.196.244.131	TLSv1.2	403	Application Data
94	1.516187	192.175.111.53	158.196.244.131	TLSv1.2	85	Encrypted Alert
96	1.517073	158.196.244.131	192.175.111.53	TLSv1.2	85	Encrypted Alert
222	2.923309	158.196.244.131	192.175.111.54	TLSv1.2	571	Client Hello
228	3.037208	192.175.111.54	158.196.244.131	TLSv1.2	1514	Server Hello
232	3.037999	192.175.111.54	158.196.244.131	TLSv1.2	539	Certificate
234	3.039163	158.196.244.131	192.175.111.54	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
244	3.156137	192.175.111.54	158.196.244.131	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
245	3.156602	158.196.244.131	192.175.111.54	TLSv1.2	335	Application Data
254	3.267601	192.175.111.54	158.196.244.131	TLSv1.2	403	Application Data
255	3.267607	192.175.111.54	158.196.244.131	TLSv1.2	85	Encrypted Alert
257	3.267773	158.196.244.131	192.175.111.54	TLSv1.2	85	Encrypted Alert
302	3.867231	158.196.194.146	158.196.244.204	TLSv1.2	91	Application Data
305	4.182309	158.196.194.146	158.196.244.204	TLSv1.2	651	Application Data
307	4.182719	158.196.244.204	158.196.194.146	TLSv1.2	688	Application Data
309	4.213715	158.196.194.146	158.196.244.204	TLSv1.2	814	Application Data
310	4.213944	158.196.244.204	158.196.194.146	TLSv1.2	651	Application Data
340	5.369552	158.196.244.204	158.196.194.146	TLSv1.2	89	Encrypted Alert
509	10.191045	158.196.244.131	192.175.111.53	TLSv1.2	571	Client Hello
511	10.305376	192.175.111.53	158.196.244.131	TLSv1.2	1514	Server Hello
515	10.306121	192.175.111.53	158.196.244.131	TLSv1.2	539	Certificate
517	10.308617	158.196.244.131	192.175.111.53	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
518	10.420265	192.175.111.53	158.196.244.131	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
519	10.421256	158.196.244.131	192.175.111.53	TLSv1.2	335	Application Data
522	10.532470	192.175.111.53	158.196.244.131	TLSv1.2	403	Application Data

Obrázek 6.4: SIP protokol, šířený přes TLS



Obrázek 6.5: Obsah zprávy Certificate protokolu TLS

7 Závěr

V prvním úseku teoretické části práce byla podrobně popsána technologie VoIP se zaměřením na signalizační protokol SIP a bezpečnostní mechanismy. V druhém úseku byla provedena teoretická analýza současných SIP klientů pod operačním systémem Android. Byly vybrány jen ty řešení, které uvádějí podporu zabezpečené komunikace.

V praktické části byl vytvořen SIP klient pro operační systém Android. Tento klient využívá knihovny PJSIP a základem pro jeho tvorbu se stalo API PJSUA2. Klient podporuje zabezpečenou komunikaci jak signalizačních dat, tak dat hlasových. U signalizačních dat je možnost přenášet informace protokolem TLS zatímco pro hlasová data je možné využít protokolu SRTP. Tento klient byl otestován na čtyřech mobilních zařízeních různých výrobců s odlišnými verzemi operačního systému Android. Na všech bylo pozorováno stejné chování aplikace i stabilita. Během testování aplikace komunikovala s pobočkovou ústřednou Asterisk, která zajišťovala mimo jiné podporu zabezpečené komunikace. Poměrnou část vývoje aplikace představovala kompilace knihoven PJSIP a OpenSSL. Veškeré pokusy o zkompilování knihovny PJSIP podle instrukcí poskytnutých vývojáři byly neúspěšné. V průběhu kompilace se objevovaly stále nové chyby, které bylo potřeba řešit. Jelikož vyhledávání řešení těchto chyb zabralo téměř třetinu celkového času pro celý vývoj aplikace, byla do práce zakomponována podkapitola (viz kapitola 5.2), která se podrobně věnuje problematice kompilace knihoven a jejich správné konfiguraci pro platformu Android.

Pro praktické ověření funkčnosti bezpečnostních mechanismů vytvořeného klienta byl zaznamenán průběh síťové komunikace pomocí nástroje Tcpdump. Následná analýza provozu programem Wireshark potvrdila, že klient funguje správně a používá jím udávané bezpečnostní prvky.

Budoucnost VoIP klientů se udává směrem technologie WebRTC. Ta umožňuje uživatelům využívat webového prohlížeče k uskutečňování hlasových hovorů, videohovorů, přenosům souborů a zasílání textových zpráv bez nutnosti instalace VoIP klienta. V současné době se však stále vyskytují problémy se zabezpečením komunikace na straně Asterisku, který sice podporuje WSS (Web Socket Security), ale prakticky tato možnost nefunguje.

Hovoříme-li o budoucím rozšiřování aplikace, pak je do aplikace možné dále implementovat podporu videohovorů a komunikaci pomocí textových zpráv. Aplikace je na tato rozšíření připravena, ale předmětem této práce nebylo jejich vypracování. Proto bylo věnováno větší úsilí k optimalizaci aplikace.

Součástí této práce je uživatelská a programová dokumentace. Programová dokumentace byla vytvořena vývojovým prostředím Android Studio 2.3. Obě dokumentace jsou součástí přílohy této práce.

Použitá literatura

[1] PUŽMANOVÁ, Rita. Moderní komunikační sítě od A do Z: [technologie pro datovou, hlasovou i multimediální komunikaci]. 2., aktualiz. vyd. Brno: Computer Press, 2006. ISBN 80-251-1278-0.

[2] HARWOOD, Mike. Internet security: how to defend against attackers on the web. Second Edition. Burlington, MA: Jones, 2016. ISBN 978-128-4090-550.

[3] PUŽMANOVÁ, Rita. TCP/IP v kostce. 1. vyd. České Budějovice: Kopp, 2004. ISBN 80-723-2236-2.

[4] ŘEZÁČ, Filip, Miroslav VOZŇÁK a Jan ROZHON. Bezpečnost v komunikacích. 2. rozšířené. Ostrava: VŠB-TU Ostrava, 2013. ISBN skript.

[5] VOZŇÁK, Miroslav. Voice over IP. 1. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2008. ISBN 978-80-248-1828-3.

[6] RFC 3261: SIP: Session Initiation Protocol [online]. IETF, 2002 [cit. 2016-02-12]. Dostupné z: https://www.ietf.org/rfc/rfc3261.txt

[7] VOZŇÁK, Miroslav. Technologie a protokoly multimediálních komunikací pro integrovanou výuku VUT a VŠB-TUO. 1. vyd. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2014. ISBN 978-80-248-3326-2.

[8] Android [online]. Google, 2017 [cit. 2017-04-04]. Dostupné z: https://www.android.com/

[9] Zoiper [online]. Zoiper, 2017 [cit. 2017-04-04]. Dostupné z: https://www.zoiper.com/en

[10] CSipSimple [online]. GooglePlay, 2017 [cit. 2017-04-04]. Dostupné z: https://play.google.com/store/apps/details?id=com.csipsimple&hl=cs

[11] Adore Softphone [online]. Adore, 2017 [cit. 2017-04-04]. Dostupné z: http://www.adoresoftphone.com/

[12] Media5-fone [online]. Media5 Corporation, 2017 [cit. 2017-04-04]. Dostupné z: http://www.media5corp.com/products/softclients-and-provisioning/media5-fone/

[13] SipDroid [online]. GooglePlay, 2017 [cit. 2017-04-04]. Dostupné z: https://play.google.com/store/apps/details?id=org.sipdroid.sipua&hl=cs

[14] MjSip [online]. MjSip, 2017 [cit. 2017-04-04]. Dostupné z: http://www.mjsip.org/

[15] Vocal VoIP Software [online]. New York: VOCAL Technologies, 2016 [cit. 2017-04-04]. Dostupné z: https://www.vocal.com/software-modules/voip-software/

[16] PJSIP Library [online]. PJSIP, 2016 [cit. 2017-04-04]. Dostupné z: http://www.pjsip.org/

[17] Simplified Wrapper and Interface Generator [online]. SWIG, 2017 [cit. 2017-04-04]. Dostupné z: www.swig.org

[18] Android Studio [online]. Google, 2017 [cit. 2017-04-04]. Dostupné z: https://developer.android.com/studio/index.html

[19] Dialogs. Android developers [online]. Google, 2017 [cit. 2017-04-06]. Dostupné z: https://developer.android.com/guide/topics/ui/dialogs.html

[20] Activity. Android developers [online]. Google, 2017 [cit. 2017-04-06]. Dostupné z: https://developer.android.com/reference/android/app/Activity.html

[21] PJSUA2 Documentation [online]. PJSIP, 2017 [cit. 2017-04-06]. Dostupné z: http://www.pjsip.org/docs/book-latest/html/

[22] Android NDK [online]. Google, 2017 [cit. 2017-04-06]. Dostupné z: https://developer.android.com/ndk/index.html

[23] VIEGA, John, Matt MESSIER a Pravir CHANDRA. Network Security with OpenSSL. Sebastopol, CA, USA: O'Reilly Media, 2002. ISBN 0-596-00270-X.

[24] HTC ONE M8 [online]. HTC Corporation, 2017 [cit. 2017-04-06]. Dostupné z: http://www.htc.com/cz/support/htc-one-m8/

[25] Huawei P8 Lite [online]. Huawei Technologies Co., 2017 [cit. 2017-04-06]. Dostupné z: http://consumer.huawei.com/en/mobile-phones/p8lite/index.htm

[26] Samsung Galaxy Tab A [online]. Samsung, 2017 [cit. 2017-04-06]. Dostupné z: http://www.samsung.com/cz/tablets/

[27] Lenovo Vibe S1 [online]. Lenovo, 2017 [cit. 2017-04-06]. Dostupné z: http://www.lenovo.com/cz/cs/smart-devices/lenovo-smartphones/vibe-series/Vibe-S1/p/PPIPPIS1A40

[28] Asterisk [online]. Digium, 2017 [cit. 2017-04-06]. Dostupné z: http://www.asterisk.org/

[29] Ubuntu [online]. Canonical, 2017 [cit. 2017-04-06]. Dostupné z: http://www.ubuntu.cz/

[30] RFC 5246 - The Transport Layer Security Protocol [online]. IETF, 2017 [cit. 2017-04-06]. Dostupné z: https://tools.ietf.org/html/rfc5246

[31] Wireshark [online]. Wireshark foundation, 2017 [cit. 2017-04-06]. Dostupné z: https://www.wireshark.org/

[32] TCPDUMP&LIBPCAP [online]. Tcpdump/Libpcap, 2017 [cit. 2017-04-06]. Dostupné z: http://www.tcpdump.org/

[33] SWIFT, OS. Android: App Development & Programming Guide: Learn In A Day!. 2. OS Swift, 2015. ISBN 9781517640095.

Seznam příloh

Součástí DP je archív ve formátu ZIP.

Adresářová struktura přiloženého archívu:

- Uživatelská_dokumentace.pdf
- caMEll.apk
- caMell (složka) obsahující Android projekt
- JavaDoc (složka) obsahující programovou dokumentaci:
- index-files (složka) obsahující soubory s indexy
- org (složka) obsahující JavaDoc soubory
- allclasses-frame.html
- allclases-noframe.html
- constant-values.html
- help-doc.html
- index.html
- overview-frame.html
- overview-summary.html
- overview-tree.html
- package-list
- script.js
- serialized-form.html
- stylesheet.css

Uživatelská dokumentace

Bc. Lukáš Palacký

PAL0075

Obsah

1. Funkce aplikace

Aplikace caMEll je klient vytvořený k vzájemné komunikaci pomocí signalizačního protokolu SIP a transportních protokolů RTP a SRTP. Mezi hlavní vlastnosti klienta patří:

- Možnost zabezpečeného přenosu signalizačních i hovorových informací.
- Uložení více SIP účtů.
- Možnost uložení kontaktů do seznamu kontaktů.
- Vytáčení čísel pomocí softwarové klávesnice.
- Oznámení o zmeškaném hovoru formou notifikace.

2. Ovládání aplikace

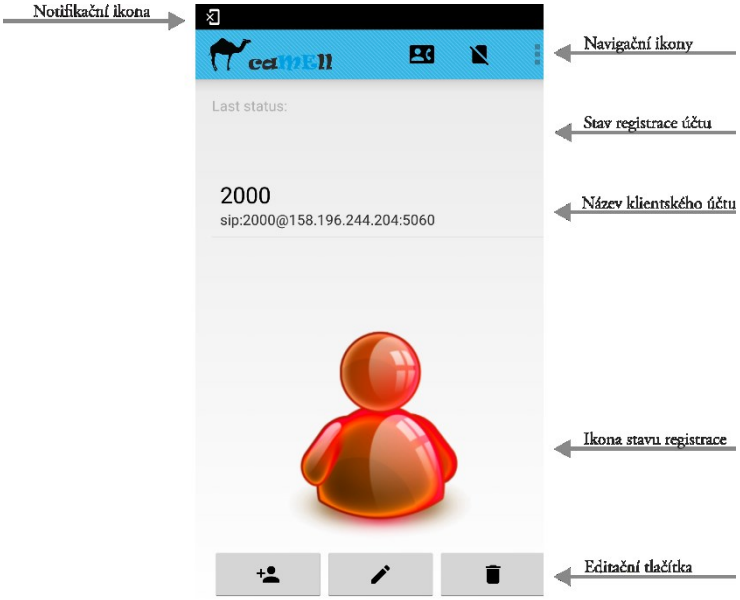
Aplikace obsahuje 4 hlavní obrazovky mezi, kterými se uživatel v rámci aplikace neustále pohybuje:

- *Účty* – tvorba, editace a přihlašování/odhlašování účtu.
- *Kontakty* – vytváření a editování kontaktů, volání na kontakty.
- *Volání* – obrazovka umožňující přijmutí/odmítnutí hovoru a následný hovor.
- *Informace* – kontakt na vývojáře a informace o licenci.

Mezi těmito obrazovkami (s výjimkou obrazovky pro *volání*) se může uživatel přesouvat posunem prstu do stran nebo navigačními ikonami v navigačním menu. Po prvním spuštění aplikace se zobrazí obrazovka s *účty* a zde je nutné registrovat nový SIP účet. Následně je možné pokračovat v obrazovce *kontakty*, s tvorbou kontaktů nebo přímým voláním z klávesnice. K odregistrování účtu ze serveru slouží ikonka v navigačním menu (tvar přeškrtnuté karty). Pro ukončení aplikace je nutné stisknout ikonku pravém horním rohu navigačního menu (tvar křížku) nebo pomocí rozbalovacího menu.

2.1 Hlavní obrazovka

Po spuštění aplikace se uživateli zobrazí hlavní obrazovka. V té je možné pracovat s klientskými účty. Pomocí tlačítek ve spodní části obrazovky je možné účet přidat, editovat anebo odebrat. V horní části obrazovky je možné sledovat stav registrace klientského účtu, který je zároveň indikován notifikační ikonou a barevnou ikonou na hlavní obrazovce. Červená barva ikony na hlavní obrazovce značí nezaregistrovaný účet, zatímco barva zelená úspěšně zaregistrovaný. Nad touto ikonou je vyhrazena část obrazovky pro položku s názvem a URI klientského účtu. V rámci celé aplikace je možné se pohybovat pomocí navigačních ikon umístěných v navigační liště, tato lišta obsahuje také ikonu ve tvaru přeškrtnuté karty v pravém horním rohu, jak již bylo zmíněno, tato ikona slouží k odregistrování účtu (viz obrázek 1).

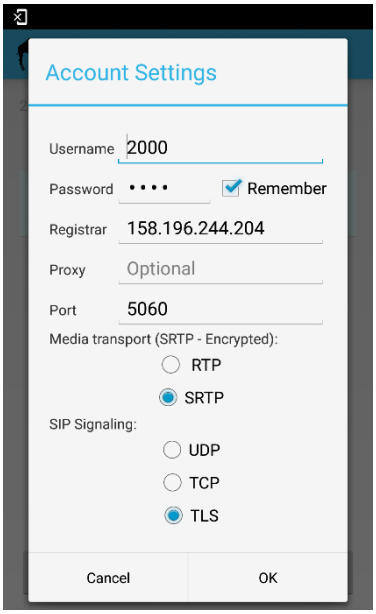


Obrázek 1: Hlavní obrazovka

2.2 Vytvoření (editace) klientského účtu

Po stisknutí tlačítka přidat účet nebo editovat účet na hlavní obrazovce se zobrazí dialog vyzývající uživatele zadat následující údaje o klientském účtu (viz obrázek 2):

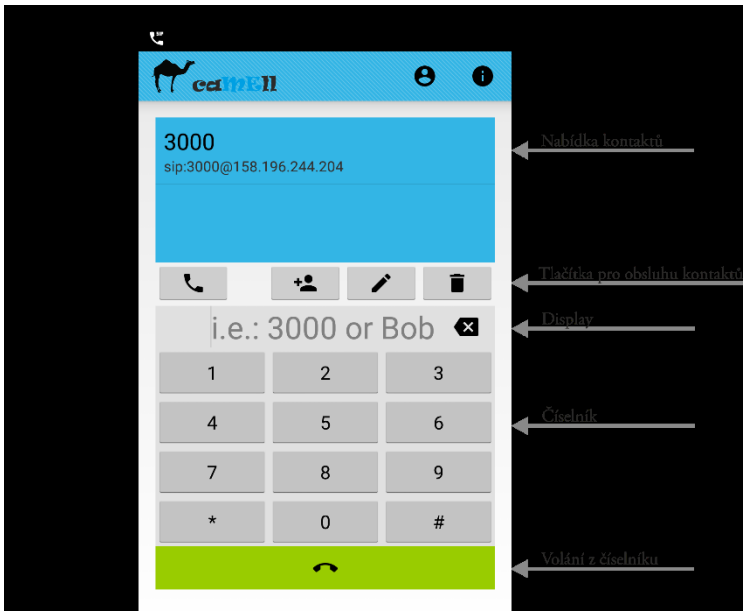
- *Username* (uživatelské jméno) – první část SIP URI (před „@“, bez „sip:“/„sips:“).
- *Password* (heslo) – heslo k účtu.
- *Registrar* – IP adresa serveru, ke kterému se uživatel přihlašuje.
- *Proxy* – IP adresa proxy serveru (nepovinný údaj).
- *Port* – číslo portu, na kterém má aplikace naslouchat (nepovinný údaj).
- *Media transport* – protokol pro přenos hovorových dat RTP/SRTP.
- *SIP signaling* – protokol pro přenos signalizačních dat UDP/TCP/TLS.



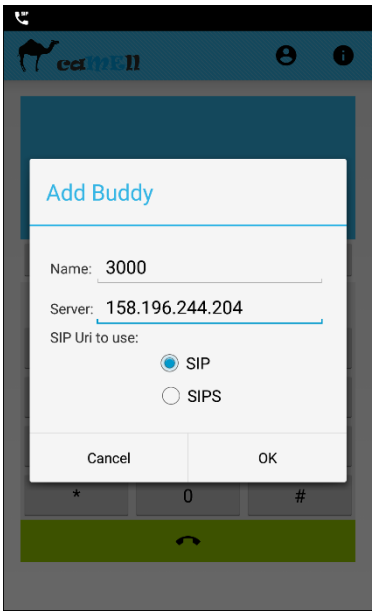
Obrázek 2: Dialog editace klientského účtu

2.3 Tvorba kontaktů

Pro přidání nového kontaktu je nejprve nutné, aby měl uživatel úspěšně zaregistrovaný účet. Poté stisknutím tlačítka pro přidání nebo editaci kontaktu, z nabídky tlačítek pro obsluhu kontaktů, vyvolá dialog (viz obrázek 4). V tomto dialogu je potřeba zadat jméno kontaktu, IP adresu serveru a schéma SIP („sip“: nebo „sips“). Po úspěšném přidání kontaktu se, nový kontakt zobrazí v nabídce kontaktů. Druhou možností jak volat kontakt (na stejném serveru, ke kterému je uživatel přihlášen) je rychlá volba na číselníku, případně lze zadat alfanumerické znaky přímo do displaye (viz obrázek 3).



Obrázek 3: Obrazovka kontakty

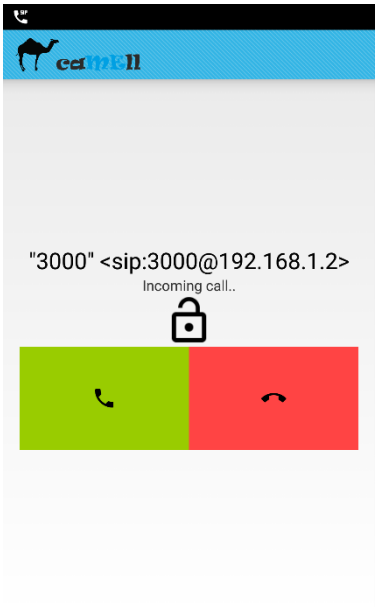


Obrázek 4: Dialog pro editaci kontaktu

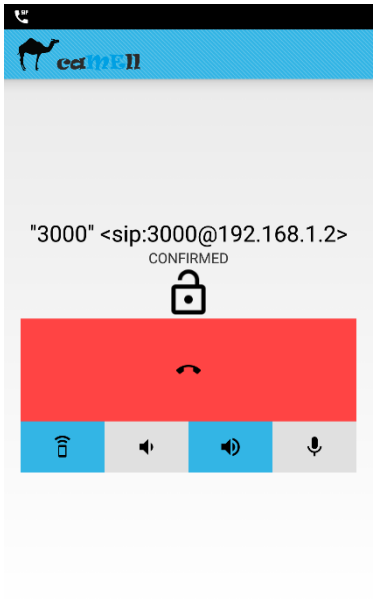
2.4 Volání v aplikaci

Volání je prakticky stejné jako v systému android. Při příchozím hovoru je možné hovor přijmout nebo odmítnout. V průběhu hovoru je v horní části obrazovky zobrazena ikona ve tvaru zámku (RTP – odemčený zámek, SRTP – uzamčený zámek). Následuje informativní stav hovoru a dále následují tlačítka (viz obrázek 5 a 6) pro:

- Ukončení hovoru.
- Hlasitý odposlech (Reproduktor).
- Snížení a zvýšení hlasitosti hovoru.
- Ztlumení mikrofonu.



Obrázek 5: Příchozí hovor



Obrázek 6: Probíhající hovor

2.5 Informace o aplikaci

Poslední obrazovkou jsou informace o aplikaci. Zde je možné nalézt kontakt na vývojáře a informace o licenci, pod kterou byla aplikace vytvořena (viz obrázek 7).

Obrázek 7: Informační obrazovka