Cuprins

1. Introducere

1.1 Context. Privire de ansamblu VoIP

VoIP este acronimul de la Voice over Internet Protocol și reprezintă tehnologia prin care are loc transferul vocii prin Internet. VoIP ofera aceleași funcționalități ca și PSTN (Public Switched Telephone Network). Acesta implică stabilirea unui canal de comunicare, digitalizarea semnalelor vocale analogice și codificarea/decodificare lor. Informația digitală este transmisă sub formă de pachete IP prin rețea. Endpoint-urile VoIP pot fi telefoane VoIP ori aplicații desktop sau mobile dedicate. Pentru a efectua comunicarea se folosesc codec-uri ca să încapsuleze semnalele audio în datagrame, care la rândul lor sunt transmise printr-o rețea care suporta IP. La primire, datagramele sunt decapsulate înapoi în semnale audio.

Primele apeluri VoIP au fost efectuate de către ARPANET în 1974, însă tehnologia VoIP așa cum o cunoaștem acum a fost inventată de VocalTec în Iseael în 1995. Peste câțiva ani după lansare VocalTec a dezvoltat posibilitatea de a face apeluri VoIP computer-telefon și telefon-telefon. În 2003 a fost lansat Skype. Utilizarea în masă a VoIP-ului a început în 2004 și a permis utilizatorilor să comunice ca și prin telefonia clasică. În prezent există mult mai multe aplicații VoIP cum ar fi Facebook Messenger, Whatsapp, Tango, Viber, folosite în special pentru apeluri internaționale și videoconferințe.

Pricipalele avantaje ale VoIP-ului asupra PSTN (Public Switched Telephone Network) sunt reducerea costurilor și refolosirea infrastructurii de Internet deja existente. De asemenea o astfel de abordare permite dezvoltarea de aplicații multimedia care includ și mesaje text și apeluri video integrate. Printre alte avantaje se numără și:

- Universalitate. VoIP permite efectuarea apelurilor între sisteme eterogene, cum ar fi computere cu diferite sisteme de operare sau computer-telefon, fără a fi necesară utilizarea adaptoarelor speciale.
- Flexibilitate. Pentru că nu există un standard pentru implementarea aplicațiilor VoIP, dezvoltatorii sunt liberi să aleagă protocoalele folosite sau funcționalitățile oferite de aplicație.

- Costuri mici. Pentru că se bazează pe o infrastructură deja existentă, costurile pentru apeluri sunt foarte mici până la gratuite. De asemenea costurile nu depind de distanța fizică între interlocutori.
- Funcționalități media adiționale. Pe langă apeluri audio, VoIP de asemenea oferă posibilitatea de a efectua apeluri video și interschimbare de mesaje text.
- Organizarea de conferințe. Se pot organiza apeluri de grup, care suporta și conferințe video. Acesta fiind un avantaj clar pentru companii în a organiza întâlniri de afaceri, fără a fi nevoie ca toți participanții să se afle în aceeași locație fizic.
- Scalabilitate. Pentru multe afaceri scalabilitatea este un punct important și VoIP facilitează adăugarea de noi utilizatori într-o rețea deja existentă, odată cu creșterea afacerii.

Există un număr mare de protocoale care pot fi combinate pentru a creea un serviciu VoIP. Acestea se împart în două categorii de bază: protocoale de semnalizare si de transport media.

Pentru transportul de pachete audio și video, cel mai folosit standard este RTP (Real-time Transport Protocol), definit de IETF (Internet Engineering Task Force) în RFC 3550. Este un protocol la nivel de aplicație și folosește UDP (User Datagram Protocol) la nivelul transport. RTP este folosit în conjuncție cu RTCP (RTP Control Protocol), care monitorizează transmisia de date și ajută la sincronizarea mai multor stream-uri în cadrul unei sesiuni. RTP rezolvă problemele legate de ordinea pachetelor și prevede mecanisme pentru a detecta pierderea pachetelor și a elimina întârzieriile. Pentru a asigura confidențialitatea comunicării, a fost dezvoltat un protocol nou, numit SRTP (Secure RTP), care pune la dispoziție criptarea, autentificarea și integritatea pachetelor audio și video. Sesiunile RTP sunt de cele mai multe ori inițializate folosind un protocol de semnalizare, cum ar fi SIP sau H.323.

H.323 și SIP (Session Initiation Protocol) au început să se dezvolte în 1995, pentru a rezolva problema inițializării comunicării între două computere în scopul de interschimba mesaje audio și video. H.323 a fost pentru prima data publicat în 1996. SIP, pe de altă parte a fost publicat abia în 1999, și re-publicat în 2002 ca RFC 3261, care este standardul recunocut astăzi.

Fundamental, H.323 și SIP fac același lucru: stabilesc o comunicare multimedia, însă difera semnificativ in design. H.323 este un protocol binar, în timp ce SIP foloseste text ASCII. Există

multe lucrări care dezbat aceste două protocoale, însă rezultatele sunt mixte, alegerea rămânând la discreția dezvoltatorilor.

SIP este un protocol de semnalizare la nivel de aplicație care poate stabili, modifica sau termina o sesiune multimedia cu doi sau mai mulți participanți, cum ar fi apelurile telefonice. SIP poate de asemenea invita participanți la sesiuni multimedia deja existente. Protocolul în sine este modelat asemenea mecanismului three-way handshake al TCP-ului. Mesajele folosite de acesta sunt INVITE, OK, ACK, BYE.

H.323 este un standard dezvolatat de ITU (International Telecommunication Union) pentru a suporta comunicațiile media peste IP. La început se axa în special pe conferințe video, dar acum include și conferințe audio. H.323 este un protocol bine structurat cu definiții rigide asupra stabilirii sesiunilor. H.323 foloseste RTP ca protocol de transport.

MGCP (Media Gateway Control Protocol) este un alt protocol de semnalizare folosit in VoIP. Acesta funcționează ca un gateway între rețelele IP și cele PSTN. Arhitectura și detaliile de implementare sunt descrise în RFC 2805. Este un protocol bazat pe mesaje text care folosește modelul de comandă-răspuns. Comenzile și răspunsurile sunt codificate în mesaje structurate și formatate cu ajutorul unor caractere speciale. Mesajele sunt transmise prin UDP. Principalele elemente ale comunicării sunt media gateway controller, media gateway endpoint si conexiunile între aceste entități. Un media gateway poate găzdui mai multe endpointuri și fiecare endpoint trebuie să suporte mai multe conexiuni.

1.2. Motivație. Vulnerabilități VoIP

Calitatea serviciilor este vitală pentru succesul aplicațiilor VoIP, întrucât puțini vor înlocui telefonia tradițională dacă VoIP nu poate oferi măcar aceași calitate a vocii. Calitatea serviciilor VoIP sunt este în principal afectată de latență, variația întârzierilor pachetelor și pierderea pachetelor. Latența se referă la timpul în care un pachet ajunge de la sursă la destinație. Limita maximă pentru acesta a fost stabilită la 150 ms pentru traficul într-o direcție. Întrucât codificarea și transportul durează 100-130 ms, implemetările de securitate sunt limitate la 20-50 ms. Variația întârzierii pachetelor introduce discontinuitate în fluxul audio,

ceea ce afectează calitatea sunetului mai mult decât întârzierile în sine și se datorează în special unei lățimi de bandă mică. Pierderea pachetelor este tolerată în anumite limite în VoIP, ăsta fiind un motiv pentru care s-a ales UDP în loc de TCP pentru transport. Pachetele sunt foarte mici, de aceea pierderile ocazionale nu sunt foarte semnificative. Ratele de pierdere tolerate sunt de 1-3%, ca să asigure o calitate cel puțin la fel de bună cu PTSN.

Sistemele VoIP se bazeaza pe rețeaua Internet, de aceea moștenesc toate problemele legate de securitate care se gasesc în rețea. În particular aplicațiile VoIP găzduite de computere pesonale sunt predispuse și vulnerabilităților legate de sistemele de operare, viruși și altele. Asigurarea securitatății în VoIP este o sarcină dificil de realizat, întrucât acesta înglobează o mulțime de protocoale care interacționează între ele. Un dezvoltator de aplicații VoIP trebuie să se asigure că fiecare protocol funcționează corect și sigur. La momentul actual nu există standarde pentru aplicațiile VoIP, de aceea calitatea lor și securitatea depinde doar de felul în care aplicația este implementată. Multe telefoane VoIP nu sunt deajuns de performante pentru a efectua criptare.

Vulnerabilitățile dispozitivelor VoIP țin de configurarea acestora și pot facilita atacurile de tip denial-of-service, eavesdropping, alterarea vocii, ceea ce rezultă în pierderea confidențialității și integrității datelor. Un exemplu ar putea fi agenții de telemarketing care ar putea utiliza informațiile astfel obținute în scopul de a-și crește profiturile.

În telefonia tradițională, pentru a intercepta mesajele transmite, atacatorul are nevoie de acces fizic la un comutator. Cu VoIP oportunitățile de interceptare sunt mult mai mari, ceea ce se datorează numărului mare de noduri prin care traversează un mesaj între două entități. Dacă un atacator compromite unul din aceste noduri, el poate accesa pachetele IP care trec prin el. Există utilitare care permit convertirea pachetelor VoIP in fișiere. Astfel conversațiile pot fi salvate și ascultate ulterior.

Un alt tip de atac este Caller ID spoofing, care constă în preluarea unui ID la alegere de către atacator. Se pot găsi situri web care pun la dispoziție acest tip de serviciu. Acestea se bazează pe headerul “From” din protocolul SIP, care poate fi ușor modificat. Atacatorul poate înlocui înregistrarea legitimă a victimei cu adresa sa, atac numit Registration hijacking. În acest caz, atacatorul primește toate apelurile victimei.

Atacurile de tip denial-of-service sunt cele mai ușor de efectuat. Cu acest tip de atac, atacatorul poate termina o conversație deja începută, ori poate preveni stabilirea uneia. În primul caz, atacatorul poate trimite cât mai multe pachete RTP victimei încât calitatea seviciului sa atingă un nivel intolerabil și astfel sesiunea să fie terminată. În al doilea caz atacatorul ar putea trimite multe mesaje de tip INVITE, îndeajuns de multe încât mesajul original să nu mai fie prelucrat de receptor.

1.3. Contribuții

În lucrarea de față am analizat soluțiile actuale pentru problemele de securitate ale protocolului Voice over IP. Din cauză că protocolul a fost dezvoltat inițial fără un modul de securitate bine pus la punct, soluțiile pentru această problemă au apărut abia mai târziu. Astfel a apărut protocolul Secure Real-time Transport Protocol, menirea căruia era asigurarea confidențialității convorbirilor. În urma studierii în detaliu a acestui protocol, am aflat că pentru a-și stabili cheia master de care are nevoie pentru a deriva tot setul de chei, a fost dezvoltat un alt protocol – ZRTP, scopul căruia este anume acesta.

În continuare, am studiat protocolul Real Privacy Management, care e unul inovativ, recent apărut. Acesta vine cu o idee netradițională în criptografie, și anume autentificarea mutuală continuă la un cost computațional acceptabil. Acesta a fost implementat cu succes în SSL/TLS, ceea ce demonstrează aplicabilitatea lui. Scopul lucrării este de a studia compatibilitatea acestuia cu SRTP. Cum urmează sa descriu în capitolul 5, integrarea RPM-ului în SRTP/ZRTP se poate face fără un efort prea mare.

1.4. Structura lucrării

Am împarțit lucrarea în cinci părți logice, care coincid cu capitolele. În primul capitol fac o introducere în contextul lucrării, și anume protocolul VoIP, unde accentuez vulnerabilitățile acestuia. În al doilea capitol abordez protocolul Secure Real-time Transport Protocol și modul în care acesta operează împreună cu funcțiile criptografice utilizate. Următorul capitol cuprinde protocolul ZRTP și o descriere detaliată a felului în care acesta funcționează. Real Privacy

Management este discutat în capitolul 4. Acesta cuprinde o privire de ansamblu asupra protocolului și tipurile de configurații propuse de dezvoltatori. Capitolul 5 abordează o posibilă soluție pentru a integra RPM în VoIP și avantajele acestei opțiuni.

2. Secure Real-Time Transport Protocol

2.1. Privire de ansamblu SRTP

SRTP este acronimul de la Secure Real-time Transport Protocol și este o extensie a RTP-ului, care asigură confidențialitatea, autentificarea și protecția impotriva replicării traficului RTP. Adițional protejării RTP-ului, SRTP prevede protecție și pentru mesajele RTCP. Întrucât mesajele RTP și RTCP sunt trimise separat și se folosesc și porturi separate, ambele trebuie să fie protejate într-o sesiune multimedia. SRTCP este echivalentul lui SRTP pentru RTP, doar că pentru RTCP. SRTP a fost dezvoltat de Cisco și Ericsson și a fost oficial publicat în martie 2004 de către IETF ca RFC 3711. Acesta este o soluție potrivită pentru VoIP întrucât este compatibil cu formatul RTP și menține costuri computaționale și lățime de bandă relativ mici.

Printre principalele funcționalități se numără derivarea de chei, autentificarea și criptarea mesajelor. Derivarea de chei reduce sarcina de stabilire a cheilor. SRTP folosește șase chei pentru un context criptografic (cheile de criptare și autentificare și salt-urile pentru SRTP si SRTCP). Acestea sunt derivate dintr-o singura cheie master și, opțional, un salt master. În acest fel protocolul de management al cheilor trebuie sa stabilească doar o cheie (cheia master). Derivarea se realizează în așa fel încât o cheie de sesiune comporomisă să nu compromită și celelalte chei folosite, chiar dacă sunt derivate din aceeași cheie master. În schimb, dacă cheia master e compromisă, aceasta dezvăluie toate cheile de sesiune derivate din ea.

Structura unui pachet SRTP se aseamănă foarte mult cu cea a unui pachet RTP, cu următoarele diferențe: mesajul trimis în text clar din RTP se înlocuiește cu criptotext, care poate fi de aceeași mărime sau poate fi mai mare; se mai adaugă MKI (Master Key Identifier), care este un parametru opțional, și tagul de autentificare. MKI are lungime configurabilă și identifică cheia master din care au fost derivate cheile de sesiune cu care a fost criptat și autentificat pachetul curent. Tagul de autentificare este de asemenea un parametru opțional, dar puternic recomandat, întrucât asigură integritatea datelor. Componentele autentificate constau în headerul RTP și criptotextul SRTP. Astfel, dacă ne dorim criptare și autentificare, trebuie mai întâi să efectuăm criptarea și abia apoi – autentificarea. De notat că parametrului opțional MKI nu îi este protejată integritatea cu ajutorul tagului.

Pentru fiecare flux SRTP, expeditorul și destinatarul trebuie să mențină o stare criptografică, care e numită context criptografic. SRTP folosește două tipuri de chei: chei master și chei de sesiune.Cheile de sesiune sunt folosite direct în transformările criptografice, iar cheile master sunt folosite pentru a deriva cheile de sesiune. Pe lângă chei, SRTP menține și o listă de parametri care sunt independenți de trasnformările criptografice folosite. Aceștia sunt:

- rollover counter (ROC) care înregistrează de câte ori numărul de secvență RTP (SEQ), trimis odata cu pachetul SRTP, a fost resetat la zero după ce a trecut de 65535;
- (doar pentru destinatar) un număr de secvență s_l, care reprezintă cel mai mare număr SEQ care a fost recepționat; Acesta ajută la detectarea mesajelor care nu vin în ordine.
- un identificator pentru algoritmul de criptare folosit;
- un identificator pentru algoritmul de autentificare folosit;
- valoarea MKI;
- o listă care conține indicii pachetelor recent primite și autentificate; Aceasta are ca scop prevenirea replicării mesajelor.
- un counter pentru fiecare cheie master, care indică câte pachete au fost trimise cu aceasta;
- lungimile cheilor de sesiune pentru criptare (n_e) și pentru autentificare (n_a).

Există cazuri în care mai multe fluxuri SRTP din aceeași sesiune RTP, identificate cu ajutorul SSRC (synchronization source), împărtășesc o mare parte din parametrii din contexul criptografic. În aceste cazuri este recomandat să se folosească liste diferite de replicare și ROC. Replicarea mesajelor are loc atunci când un adversar stochează un pachet RTP și îl re-injectează în rețea, forțând astfel destinatarul sa reutilizeze o cheie de sesiune, fapt care are consecințe grave asupra securității sesiunii.

Un context criptografic este identificat prin 3 parametri: SSRC, adresa destinatarului și portul la care se găsește. Considerând că contextul criptografic a fost deja inițializat, emițătorul ar trebui sa efectueze următorii pași pentru a construi un pachet SRTP:

1. Determină ce context criptografic trebuie utilizat.
2. Determină indicele pachetului folosind ROC și SEQ după formula i = 2^16 * ROC + SEQ.
3. Determină cheia si salt-ul master folosind indicele calculat mai sus sau MKI-ul curent.

4. Determină cheile si salt-ul de sesiune folosind cheia master și lungimile specificate în contextul criptografic.
5. Criptează mesajul RTP, folosind algoritmul de criptare specificat în contextul criptografic, cheia și salt-ul de sesiune determinate în pașii anteriori.
6. Dacă se folosește MKI, acesta se atașează la pachet.
7. Pentru autentificarea mesajului, calculează tagul de autentificare folosind ROC, algoritmul de autentificare, cheia de sesiune si lungimea ei din cadrul contextului criptografic curent.
8. Dacă e necesar, actualizează ROC-ul.

Destinatarul, la momentul recepționării unui pachet SRTP, urmează următorii pași:

1. Determină contextul criptografic care trebuie să-l folosească.
2. Determină indicele pachetului.
3. Determină cheia și salt-ul master. Dacă e folosit MKI, se ia MKI transmis în pachet, altfel se folosește indicele calculat la pasul 2.
4. Determină cheile de sesiune.
5. Pentru autentificare și protecția împotriva replicării, se verifică mai întâi dacă pachetul a fost replicat cu ajutorul listei de replicare. Dacă pachetul este stabilit ca fiind o replică, acesta este discardat și evenimentul este înregistrat.
6. Se calculează tagul de autentificare conform algoritmului specificat si cheilor de sesiune din context. Dacă acesta nu coincide cu cel din pachetul trimis, pachetul este discardat și evenimentul – înregistrat.
7. Se decriptează criptotextul folosind algoritmul si cheile de criptare din contextul criptografic.
8. Se actualizează parametrii ROC și s_l, și de asemenea lista de replicare.

2.2. Transformările criptografice utilizate

SRTP suportă numeroși algoritmi de criptare și autentificare. Voi descrie mai jos algoritmii folosiți în mod implicit de către SRTP.

Transformările criptografice folosesc indicele pachetului și cheia secretă pentru a genera un segment de keystream. Fiecare keystream este utilizat pentru a cripta un singur pachet SRTP. Procesul de criptare constă în generarea keystream-ului corespunzător pachetului și efectuarea operației de XOR între el și mesajul RTP. În cazul în care lungimea mesajului nu este un multiplu de lungimea blocului pentru criptare, cei mai nesemnificativi biți din keystream nu sunt utilizați. Decriptarea se realizează în același mod interschimbând rolul mesajului RTP cu criptotextul. Primii octeți din keystream sunt rezervați pentru a fi utilizați în codul de autentificare al unui mesaj și nu trebuie folosiți pentru criptare. Aceștia sunt numiți keystream prefix și au o lungime definită de constanta SRTP_PREFIX_LENGTH.

Felul în care acest keystream este generat depinde de codul utilizat și de modul lui de operare. Cifrul implicit folosit în SRTP este AES cu două moduri de operare: Counter Mode AES și AES în f8-mode. Conceptual, counter mode costă în a cripta întregi succesivi. Segmentul de kestream se generează în felul următor:

E(k, IV) || E(k, IV + 1 mod 2^128) || E(k, IV + 2 mod 2^128) || ...

Unde E(k, x) este AES aplicat pe cheia k și blocul x. IV este un întreg pe 128 de biți, care se calculează după formula:

IV = (k_s * 2^16) XOR (SSRC * 2^64) XOR (i * 2^16)

unde k_s este salt-ul de sesiune și i este indicele pachetului. Toți termenii folosiți în operația XOR trebuie să fie padați la stânga cu 0 până la 128 biți. Incluziunea parametrului SSRC ne oferă libertatea de a proteja mai multe fluxuri SRTP din aceeași sesiune cu aceeași cheie de sesiune.

AES în f8-mode a fost dezvoltat ca o soluție pentru UMTS (Universal Mobile Telecommunications System). El este o variantă a modului OFB (Output Feedback Mode), cu o funcție de feedback mai elaborată. AES în f8-mode pentru SRTP trrbuie să folosească aceleași lungimi ale cheii de sesiune și salt-ului ca și AES în Counter Mode. Vectorul de inițializare (IV) este de lungime 128 de biți și se calculează după formula:

IV = 0x00 || M || PT || SEQ || TS || SSRC || ROC

unde M, PT, SEQ, TS și SSRC se iau din headerul RTP, iar ROC se ia din contextul criptografic folosit.

La fel ca la AES în Counter Mode, prezența parametrului SSRC ne permite să utilizăm aceeași cheie de sesiune pentru mai multe fluxuri SRTP. Vectorul IV nu se utilizează direct, în locul lui se folosește IV', calculat: IV' = E(k_e XOR m, IV), unde k_e este cheia de criptare, iar m – masca m = k_s || 0x555..5. Keystreamul are forma S(0) ||... || S(L-1), unde L = lungimea mesajului/lungimea blocului rotunjit superior, iar S(j) se calculează:

S(j) = E(k_e, IV' XOR j XOR S(j-1)).

S(-1) e stabilit prin convenție 00..0.

Cifrul NULL se folosește atunci când nu se dorește confidențialitate pentru RTP. În acest caz putem considera keystream-ul ca fiind 00..0.

Autentificarea pachetelor SRTP are loc precum urmează: Emițătorul calculează valoarea tagului de autentificare și îl atașează pachetului. La primire, destinatarul calculează un nou tag de autentificare și îl compară cu cel primit. Dacă valorile coincid, atunci autentificarea a avut loc cu succes. În caz contrar se returnează un mesaj de eroare. Pentru autentificare se folosește HMAC-SHA1 aplicat asupra cheii de autentificare și datelor ce trebuie autentificate, iar rezultatul se trunchiază la mărimea tagului.

Indiferent de algoritmii de criptare sau autentificare utilizați, toate implementările SRTP trebuie să utilizeze derivarea de chei de sesiune specificată în RFC 3711. Odată ce rata de derivare a cheii, păstrată în parametrul key_derivation_rate a fost stabilit, nu mai este necesară comunicarea între părțile ce folosesc SRTP. Funcția de derivare trebuie apelată pentru prima dată înainte ca primul pachet sa fie trimis și după aceea, de fiecare dată când indicele pachetului mod key_derivation_rate devine egal cu 0. Valoarea pentru key_derivation_rate este ținută în contextul criptografic și este fixată pentru toată durata de viață a cheii master asociate.

Pentru derivarea cheilor se folosește o funcție sigură pseudo-random (PRF). Operația DIV este definită astfel: a DIV b returnează câtul lui a și b rotunjit inferior, cu rezultatul de aceeași lungime ca și a. Prin convenție a DIV 0 = 0 pentru orice a. <label> este o constantă pe 8 biți

specifică pentru fiecare tip de cheie. Astfel pentru cheia de criptare <label> = 0, pentru cheia de autentificare <label> = 1, iar pentru salt <label> = 2. Cheile se derivează în felul următor:

- Fie r = indicele pachetului DIV key_derivation_rate
- Fie key_id = <label> || r
- Fie x = key_id XOR salt-ul master (aliniați la dreapta)
- noua cheie = PRF(cheia master, x), cu lungimea specificată în contextul criptografic.

Derivarea de chei implicită pentru SRTP folosește AES în Counter Mode cu o cheie master de 128 biți și salt master de 112 biți. Parametrul key_derivation_rate este 0.

3. Zimmerman RTP

Denumirea de ZRTP vine de la inventatorul acestul protocol Phil Zimmerman și RTP. Este un protocol de stabilire a cheilor criptografice pentru criptare prin metoda Diffie-Hellman între două noduri în cadrul unui apel VoIP. El generează cheia master și salt-ul petru o sesiune SRTP. Protocolul nu solicită secrete împărtășite anterior și nu se bazează pe infrastructura de chei publice. El folosește chei Diffie-Hellman efemere și permite detecția atacurilor man-in-the-middle (MitM) cu ajutorul unui string de autentificare scurt (SAS – de la short authentication string) care e afișat pe ecran și citit cu voce de către interlocutori. Cheile sunt distruse la finalul apelului. ZRTP poate fi folosit cu orice protocol de semnalizare în VoIP, întrucât este intependent de nivelul de semnalizare și tot procesul de negociere a cheilor are loc prin fluxul RTP. Protocolul a fost publicat de IETF în aprilie 2011 ca RFC 6189.

3.1. Privire de ansamblu ZRTP

ZRTP se bazează pe același principiu ca și RTP folosind mesaje de tip cerere-răspuns. Comunicarea are loc pe același port ca și RTP. În plus, ZRTP folosește un header special care să-l diferențieze de RTP. Protocolul începe după ce endpoint-urile au utilizat un protocol de semnalizare cum ar fi SIP. Ambele endpoint-uri încep o sesiune ZRTP cu un mesaj Hello, scopul căruia este sa asigure că amblele endpoint-uri suportă ZRTP și să stabilească ce algoritmi au în comun. Mesajele Hello conțin opțiunile de configurare pentru SRTP și ZID-ul. Fiecare instanță de ZRTP are un ZRTP ID (ZID) random pe 96 de biți, care este generat în timpul instalării. ZID-urile sunt folosite pentru a căuta secretele pre-distribuite în memorie de la sesiuni ZRTP precedente. Răspunsul la un mesaj Hello este un mesaj de tip HelloACK, care pur și simplu confirmă recepționarea mesajului Hello. Întrucât ZRTP folosește UDP la nivel de transport, el are timere de retransmisie în caz de pierderi de pachete. Se folosesc două tipuri de timere: unul pentru mesajele Hello și altul pentru toate restul după ce a fost primit un mesaj HelloACK. Hello și alte tipuri de mesaje folosesc de asemenea un hash care are ca scop să lege mesajele între ele și să respingă mesaje false ZRTP injectate în timpul sesiunii.

După interschimbarea de mesaje Hello și HelloACK, schimbul de chei începe cu un mesaj Commit. Acesta poate fi trimis imediat după HelloACK sau poate fi amânat pentru mai târziu, după ce participanții fac un schimb necriptat de mesaje și este activat manual din interfață. Este recomandat însă să se trimită un mesaj Commit automat imediat după faza de Hello/HelloACK.

ZRTP suportă mai multe moduri de stabilire de chei, printre care modul Diffie-Hellman și modul Preshared. Inițiatorul se consideră endpointul care a trimis mesajul Commit. Pentru modul Diffie-Hellman, valorile publice DH sunt schimbate cu ajutorul mesajelor DHPart1 și DHPart2. După asta se calculează cheile și salt-urile SRTP. Autentificarea ZRTP folosește SAS (short authentication string), care în mod ideal este afișat utilizatorului pentru a fi citit. Alternativ însă, SAS poate fi trimis printr-o semnătură digitală în mesajele Confirm1 și Confirm2. Aceste mesaje au ca scop principal să asigure că calculele pentru stabilirea cheilor a avut loc cu succes și poarta și informații adiționale despre sesiune print-un șir de flag-uri. În modul Preshared calculele DH sunt omise, dacă participanții au un secret comun din sesiunea anterioară. Acest mod este convenabil pentru procesoarele mai slabe, astfel încât calculele DH să nu fie efectuate la fiecare sesiune ori pentru a restabili rapid o sesiune întreruptă. Dacă cache-ul unui participant este capturat de un oponent și următorul apel este în mod Preshared, acesta poate descifra conversația. În schimb dacă oponentul ratează urmatorul apel, secretele din cache sunt actualizate la sfarșitul fiecărei sesiuni. Există și modul multistream, atunci când se adaugă de asemenea un flux video la conversație. Nu se mai calculează secrete adiționale folosind DH, ci se derivează din cele existente deja.

3.2. Descrierea în detaliu

3.2.1. Negocierea versiunii și a algoritmilor

Fiecare participant include versiunea ZRTP suportată în mesajul Hello. Dacă ambii participanți au aceeași versiune, ei pot continua dupa protocol. Dacă suportă mai multe versiuni, ele trebuie să fie transmise de preferat sub formă de listă în timpul protocolului de semnalizare, cum ar fi SIP. Se alege versiunea cea mai mare suportată de ambele părți. În cazul în care un participant primește un mesaj Hello cu o versiune mai mare de ZRTP decât cea suportată de el, mesajul

va fi ignorat și va transmite alte mesaje Hello conform timerului. Dacă participantul primește un mesaj Hello cu o versiune mai mică decât cea pe care a inclus-o el în mesaj, și suportă acea versiune, atunci încetează să mai trimită versiunea precedentă și o înlocuiește cu cea suportată de ambii. În caz ca primește un Hello cu o versiune mai mică, pe care nu o suportă, va transmite un mesaj de eroare indicând eșuarea stabilirii unei versiuni.

Pentru stabilirea algoritmului și lungimii DH fiecare mesaj Hello conține o listă cu algoritmii în ordinea preferințelor (de ex. DH2k, DH3k, EC25, EC38=ECDH-384). Endpointurile elimină opțiunile care nu se intersectează din ambele liste, rezultând în două liste cu algoritmi comuni care sunt sau nu în aceeași ordine. În cazul în care primul element din cele două liste nu coincide, se alege algorimul cel mai rapid. Dacă se procedează conform acestei metode, atunci fiecare endpoint poate începe calculele DH imediat după recepționarea mesajului Hello. Ceilalți algoritmi sunt aleși de inițiator în mesajul Commit.

Dacă ambii participanți trimit mesaje Commit în același timp, următoarele reguli se aplică: Dacă un mesaj este pentru modul DH și celălalt pentru Preshared, atunci Commitul Preshared se discardează. Dacă ambele Commit-uri sunt pentru modul Preshared și un participant a marcat flag-ul MiTM în mesajul Hello, iar celălalt nu, atunci Commit-ul de la participantul cu flag-ul marcat e discardat. Dacă ambele mesaje Commit sunt în modul DH ori în modul Preshared , atunci mesajul cu cel mai mic hvi (hash value of initiator) pentru DH sau cea mică valoare nonce (pentru non-DH) trebuie să fie discardat.

3.2.2. Utilizarea secretelor reținute

Fiecare endpoint ZRTP menține un cache de lungă durată cu secretele împărtășite pe care le-a negociat într-o sesiune precedentă cu alți participanți. ZID-ul celuilalt participant este folosit în ca index în acest cache pentru a determina secretele reținute, dacă acestea există. Această opțiune oferă ZRTP-ului proprietatea de continuintate a cheilor. Cache-ul conține intrări pentru secretele rs1, rs2, auxsecret și pxbsecret, atunci când există o entitate de încredere drept MiTM PBX (Private Branch Exchange). Mesajele DHPart1 și DHPart2 conțin o listă de hash-uri pentru aceste secrete, pentru a permite participanților să verifice dacă aceștia împărtășesc sau nu secrete comune. Dacă nu există vreun secret în cache – se folosește o valoare random, ca să

se asigure nepotrivirea hash-urilor. Această metodă previne atacatorii de a afla ce mod se va folosi pentru sesiune.

Secretul auxiliar auxsecret este definit în afara protocolului ZRTP. În unele cazuri este furnizat de protocolul de semnalizare sau manual prin alte modalități specifice aplicației. Acestea ar putea fi un hash al unei parole stabilite anterior sau un token hardware sau ar putea fi o cheie generică specifică unei instituții pentru utilizatorii ei. Auxsecret este destinat să se folosească în afara protocolului ZRTP. Din această cauză, puține endpointuri îl vor folosi.

Pentru ambii inițiatorul și responderul vor fi calculate secretele s1, s2 și s3 ca să poată fi utilizate mai târziu pentru a calcula s0. s1 va fi rs1 sau rs2 al inițiatorului în dependență de ce se găsește în cache-ul responderului. Dacă rs1 al inițiatorului se potrivește cu rs1 sau rs2 al responderului, atunci s1 va fi setat cu valoarea rs1 al inițiatorului. Dacă potrivirea eșuează și rs2 al inițiatorului se potrivește cu rs1 sau rs2 al responderului, atunci s1 va fi setat cu valoarea rs2 al inițiatorului. Numai dacă nu există nici o potrivire între secretele inițiatorului și responderului, s1 va fi setat pe NULL. s2 va lua valoarea secretului auxsecret, numai dacă acesta coincide la ambele părti. Altfel este setat pe NULL. s3 ia valoarea secretului pbxsecret, doar dacă aceasta coincide la ambele părți, altfel este setat pe NULL. Dacă s1, s2 sau s3 au valoare NULL, se presupune că au lungime zero, în scopul calcularii valorii hash pentru s0.

Ambii participanți calculează un set de hash-uri, care sunt trunchiate la 64 biți. Funcția MAC se definește în dependență de algoritmul ales de inițiator în mesajul Commit.

rs1IDr = MAC(rs1, "Responder")

rs2IDr = MAC(rs2, "Responder")

auxsecretIDr = MAC(auxsecret, Responder's H3)

pbxsecretIDr = MAC(pbxsecret, "Responder")

rs1IDi = MAC(rs1, "Initiator")

rs2IDi = MAC(rs2, "Initiator")

auxsecretIDi = MAC(auxsecret, Initiator's H3)

pbxsecretIDi = MAC(pbxsecret, "Initiator")

unde H3 este H3 = hash (H2), H2 = hash (H1), H1 = hash (H0), iar H0 este un nonce random, diferit pentru fiecare participant. H3 se găsește în mesajul Hello al fiecărui participant.

Responderul trimite rs1IDr, rs2IDr, auxsecretIDr și pbxsecretIDr în mesajul DHPart1. Inițiatorul trimite parametrii corespunzători în mesajul DHPart2. Responderul folosește valorile calculate local pentru rs1IDi, rs2IDi, auxsecretIDi și pbxsecretIDi și le compară cu cele primite în mesajul DHPart2. Inițiatorul face la fel cu valorile corespunzătoare. Din aceste comparații s1, s2 și s3 sunt calculate cu metodele descrise mai sus. Secretele care se potrivesc după hash sunt păstrate, în timp ce restul sunt setate pe NULL. În cazul în care secretele din cache nu se potrivesc, deși ne așteptăm să fie o potrivire, atunci există două posibilități: ori are loc un atac MiTM, ori celălalt participant legitim a pierdut secretele din memorie dintr-o oarecare cauză. Acest eveniment trebuie să fie tratat ca un posibil atac și utilizatorii trebuie să fie avertizați. Dacă este posibil aceștia ar trebui sa compare valorile SAS verbal, ori, dacă nu este posibil – să poată alege dacă doresc să continuie apelul ori să-l întrerupă. Nepotrivirea secretului auxsecret trebuie tratată în dependență de implementare, dar în general e mai puțin gravă decât nepotrivirea secretelor rs1.

3.2.3. Stabilirea cheilor

Următorul pas este stabilirea secretului s0 pentru a deriva materialul de chei SRTP din el. Generarea se poate face folosind Diffie-Hellman sau în baza secretelor reținute. Scopul unui schimb DH este sa genereze un secret nou s0. În modul Preshared calculele se bazează pe secretele deja existente în cache.

Din intersecția algoritmilor din mesajele Hello primite și trimise inițiatorul alege tipul hash-ului, cifrului, tagului de autentificare, tipul SAS-ului și tipul stabilirii cheilor (DH / ECDH) care vor fi folosite. În modul Diffie-Hellman stabilirea de chei începe când inițiatorul alege o valoare secretă DH nouă (svi) în baza tipului de chei ales și calculează valoarea publică pvi = g^svi mod p. Valoarea hash (hvi) a inițiatorului include hash-ul întregului mesaj DHPart2 concatenat cu mesajul Hello al responderului și este trunchiată la 256 biți. Inițiatorul trimite valoarea hvi în mesajul Commit.

Atunci când primește mesajul Commit, responderul generează propria valoare secretă, svr, și calculează valoarea pulică pvr = g^svr mod p. Când primește mesajul DHPart2 de la inițiator,

responderul verifică valoarea publică a acestuia. Dacă e 1 sau p-1, utilizatorul trebuie să fie alertat de atac și schimbul de mesaje terminat. Altfel, responderul calculează propriul său hash public pvi în baza mesajului sau Hello și mesajul DHPart2 primit, și îl compară cu hvi primit în mesajul Commit. Dacă acestea nu coincid, atunci a avut loc un atac MiTM și protocolul trebuie să fie terminat. La final se calculează rezultatul DH conform formulei: DHResult = pvi^svr mod p.

Pentru a calcula s0, din care ulterior vor fi derivatele cheile SRTP, se calculează mai întâi un hash total după formula:

total_hash = hash(Hello al responderului || Commit || DHPart1 ||DHPart2)

În aceste calcule se includ doar mesajele ZRTP, nu tot pachetul. Pentru a calcula s0, se folosește formula:

s0 = hash(counter || DHResult || "ZRTP-HMAC-KDF" || ZIDi ||ZIDr || total_hash || len(s1) || s1 || len(s2) ||s2 || len(s3) || s3)

unde counter are tot timpul valoarea 1. După calculul s0, valorile pentru s1, s2, s3 și DHResult se șterg imediat din memorie.

Modul Preshared e util pentru dispozitivele cu resurse limitate, întrucât permite derivarea cheilor SRTP fara a se face toate calculele descrise mai sus. Acesta se folosește de secretul comun rs1. În acest mod inițiatorul trimite valoarea keyID în mesajul Commit, care este calculată după formula:

preshared_key = hash(len(rs1) || rs1 || len(auxsecret) || auxsecret ||len(pbxsecret) || pbxsecret)

keyID = MAC(preshared_key, “Prsh”)

În locul valorii hvi, în Commit se trimite un nonce random pe 16 octeți si keyID, trunchiată la 64 biți.

Responderul folosește keyID ca să verifice secretele comune pe care le are în cache cu inițiatorul. Dacă keyID calculat de responder local nu coincide cu cel primit, acesta încearcă cu un nou subset de secrete până găsește o potrivire sau consumă toate opțiunile. Dacă găsește

o potrivire, keyID este folosit pentru calculul lui s0. În caz contrar, acesta răspunde cu un mesaj Commit propriu și se trece în modul DH. În modul Preshared nu se mai trimit mesajele DHPart1 și DHPart2. Pentru a calcula total_hash pentru modul Preshared se folosește formula:

total_hash = hash(Hello al responderului || Commit)

s0 se calculează folosinf funcția de derivare ZRTP – KDF, care necesită un context. Acesta se stabilește conform: KDF_Context = (ZIDi || ZIDr || total_hash) și este unic pentru fiecare sesiune din cauza modului în care se calculează total_hash. După asta putem calcula s0 conform formulei:

s0 = KDF(preshared_key, "ZRTP PSK", KDF_Context, negotiated hash length).

3.2.4. Derivarea cheilor SRTP

Pentru a deriva cheile SRTP, ZRTP folosește funcția KDF care este o funcție bazată pe HMAC. Aceasta depinde de tipul algoritmului de hash selectat de către inițiator în mesajul Commit și are forma:

KDF(KI, Label, Context, L) = HMAC(KI, i || Label ||0x00 || Context || L)

unde KI este un secret, de exemplu s0, Label este un șir de caractere care indică scopul folosirii funcției, Context include ZIDi, ZIDr și un nonce opțional pentru unicitate, iar L este lungimea la care urmează sa fie trunchiat rezultatul, păstrând cei mai semnificativi biți și nu trebuie să depășească lungimea șirului returnat de HMAC.

Înainte de a deriva cheile SRTP, se determină o cheie ZRTP de sesiune, numită ZRTPSess, care se calculează în felul următor:

ZRTPSess = KDF(s0, "ZRTP Session Key", KDF_Context, negotiated hash length)

Această cheie se păstrează pâna la finalul sesiunii și trebuie să fie distrusă imediat după. Valoarea SAS se calculează: sashash = KDF(s0, "SAS", KDF_Context, 256) trunchiat la cei mai semnificativi 32 biți. ZRTP pune de asemenea la dispoziție o cheie “externă”, care se poate folosi în afara protocolului ZRTP, calculată după formula:

ExportedKey = KDF(s0, "Exported key", KDF_Context, negotiated hash length)

ZRPT folosește SRTP fără MKI (Master Key Identifier), autentificare pe 32 de biți HMAC-SHA1, AES în Counter Mode cu cheie de lungime 128 sau 256 biți, key_derivation_rate de 2^48 și prefix SRTP egal cu 0. Se folosesc chei master și salt-uri separate pentru fiecare flux, în fiecare direcție. Astfel inițiatorul ZRTP criptează și responderul ZRTP decriptează mesajele folosind srtpkeyi și srtpsalti, pe când responderul ZRTP criptează și inițiatorul ZRTP decriptează folosind srtpkeyr și srtpsalr.

srtpkeyi = KDF(s0, "Initiator SRTP master key", KDF_Context, negotiated AES key length)

srtpsalti = KDF(s0, "Initiator SRTP master salt", KDF_Context, 112)

srtpkeyr = KDF(s0, "Responder SRTP master key", KDF_Context, negotiated AES key length)

srtpsaltr = KDF(s0, "Responder SRTP master salt", KDF_Context, 112)

Se mai generează key ZRTP pentru a cripta mesajele Confirm1 și Confirm2:

zrtpkeyi = KDF(s0, "Initiator ZRTP key", KDF_Context, negotiated AES key length)

zrtpkeyr = KDF(s0, "Responder ZRTP key", KDF_Context, negotiated AES key length)

Toate cheile sunt distruse în momentul în care nu mai sunt folosite, nu mai târziu decât finalul apelului.

4. Real Privacy Management

4.1. Privire de ansamblu și proprietăți

RPM este acronimul de la Real Privacy Management, tehnologie dezvoltată de Relevant Security Corp, Denver. RPM este o metodă de autentificare mutuală continuă și criptare bazată pe chei secrete. A fost dezvoltată pentru sistemele care nu au putere de procesare prea mare. RPM este lightweight și deci, mai rapidă decât metodele tradiționale de securitate. Acest aspect face RPM o tehnologie potrivită pentru a fi utilizată în VoIP. S-a testat ca autentificarea RPM durează 5 microsecunde. RPM de asemenea ocupă un spațiu mult mai restrâns comparativ cu alte librării de securitate, cu o dimensiune 10 KB sau mai puțin. RPM folosește chei simetrice pentru autentificare și criptare. Acestea se fac într-un singur pas de la un participant la altul și nu implică schimbul de chei de securitate.

Fundamentele matematice folosite în securitatea din ziua de azi se bazează pe probleme care sunt considerate NP (nondeterministic polynomial time). Printre acestea se numără factorizarea numerelor mari, logaritmi discreți, etc. În criptografie asta înseamnă securitate perfectă, lasând ca opțiune doar cautarea exhaustivă. Odată cu îmbunătățirea performanței calculatoarelor, numerele utilizate trebuie să fie tot mai mari, ceea ce are un impact și asupra sistemelor care doresc asigurarea securității, întrucât cresc și costurile computaționale. RPM oferă aceleași capabilități de securitate în baza unor probleme matematice care pot fi demonstrate a fi nerezolvabile.

RPM folosește aritmetica modulară și aritmetica bazată pe poziția cifrelor, care, aplicate în anumite configurații, asigură protecția necesară pentru schimbul de informații. În baza lor RPM își moștenește caracteristicile princilale, și anume viteza și dimensiunea.

4.2. Funcții low-level folosite

Funcțiile MOD16 și MOD16D: Aceste funcții execută adunarea modulară fără transport a fiecărei cifre a celor doi parametri primiți. Așa cum implică și numele, RPM folosește chei

hexadecimale. Funcțiile folosesc o valoare secretă cunoscută de ambii participanți, adaugă un salt random sau o altă valoare și își comunică rezultatul. Folosind rezultatul, doar participanții care cunosc valoarea secretă comună vor putea determina a doua valoare adăugată. Aceste funcții sunt utile pentru a actualiza valorile secrete comune fără a le transmite direct.

Funcția One Way Cut (OCW): Această funcție folosește adunarea modulară pentru a combina cifrele unei singure valori, în conformitate cu un parametru specificat, obținându-se o valoare nouă din cea veche.

Funcția Position Digit Algebra (PDAF): Aceasta este o funcție extrem de flexibilă care primește unul sau doi parametri și îi folosește pentru a indica poziția și ordinea cifrelor utilizate în calcul. Rezultatul poate fi folosit pentru alte iterații ale funcției PDAF sau pentru alte funcții. Ea este deseori folosită în combinație cu OCW.

Funcțiile RPMCombine și RPMExtract: Această pereche de funcții crează un nou "alfabet" de tip cheie-valoare prin selectarea de valori indicate de poziții de cifre și adunarea lor modulară. Aceste două funcții oferă aceleași proprietăți one-way ca și PDAF, doar că într-o manieră puțin mai eficientă.

4.3. Tipuri de configurații RPM

În prezent există patru tipuri de configurări RPM, însă numărul acestora poate fi extins în dependență de cerințele aplicației pentru care este dezvoltat.

4.3.1. PDAF Baseline

În această configurație, fiecare participant este deținătorul a două chei secrete: ID1 și ID2, ficare pe 256 de biți, pentru a fi utilizați cu chei de criptare pe 128 de biți. Operațiile care trebuiesc urmate sunt:

1. S + ID1 = ORS
2. ID2[S] = IDtemp
3. $IDtemp_{left} + IDtemp_{right} = W$
4. AES(Msg, W) = CT
5. S[ID2] = ID1new
6. ID2[ID1] = ID2new

unde S este un salt pentru mesaj generat de algoritm pseudo-random; ID1new și ID2new sunt noile chei secrete ID1 și ID2; W este cheia de criptare pe 128 biți; $X_{left} + Y_{right}$ este funcția OCW; iar X[Y] este funcția PDAF. ORS și CT sunt parametrii trimiși recipientului.

4.3.2. PDAF Network

În această configurație fiecare participant este deținătorul a unei chei secrete master RPM-EK pe 256 biți, pentru folosirea cu chei de criptare de lungime 128 biți. Operațiile, pas cu pas, sunt:

1. Generează ID1, ID2 și S random de 256 biți
2. ID1 + RPM-EK = ORID1
3. ID2 + RPM-EK = ORID2
4. S + ID1 = ORS
5. ID2[S] = IDtemp
6. $IDtemp_{left} + IDtemp_{right} = W$
7. AES(Msg, W) = CT

unde parametrii ORID1, ORID2, OSR și CT sunt trimiși recipientului.

4.3.3. CE Baseline

Această configurație se bazează pe funcțiile Combine și Extract menționate mai sus. Fiecare participant deține o cheie ID1 și una RPM-EK, ambele pe 256 biți și sunt folosite pentru criptare cu chei de 256 biți. Operațiile sunt:

1. Generează S random
2. S + RPM-EK = ORR
3. Combine(S, ID1) = A
4. Extract(ID1, A) = W
5. AES(Msg, W) = CT

unde parametrii ORR și CT sunt trimiși recipientului.

4.3.4. CE Network

În această configurație fiecare participant are o cheie ID1 pe 256 biți și folosește chei de 256 biți pentru criptare. Operațiile efectuate pentru a trimite un mesaj sunt:

1. Generează S random
2. Combine(S, ID1) = A
3. Extract(ID1, A) = W
4. AES(Msg, W) = CT

Unde R și CT sunt trimiși prin rețea recipientului.

Dacă destinatarul un pachet RPM reușește să decripteze mesajul într-un mod corect, acest fapt garantează autenticitatea acestui mesaj. Emițătorul poate fi numai celălalt deținător al cheilor RPM-EK, ID1 sau ID2. Această garanție se bazează pe modul în care e folosit cifrul. Pentru ambele tipuri de configurații Baseline, este demonstrat că scurgerea cheii de criptare W nu duce la alte scurgeri de informații.

5. VoIP cu RPM. Soluție

După cum am descris mai sus, RPM (Real Privacy Management) este o tehnologie inovativă care vine cu soluții pentru problemele principale ale VoIP-ului: viteza și spațiul de memorie. Scopul dezvoltării VoIP-ului este să înlocuiască treptat telefonia tradițională PSTN. La momentul actual există adaptoare speciale pentru telefoane clasice pentru a suporta apeluri VoIP. Provocările cele mai mari ale dezvoltării unei aplicații VoIP sunt legate de asigurarea confidențialității în momentul comunicării. Soluțiile pentru securitate în ziua de azi se bazează pe calcule cu numere mari, ceea ce implică costuri computaționale semnificative. Acesta este un obstacol major pentru dispozitivele cu putere de calcul limitată, cum ar fi telefoanele obișnuite. În prezent pilonul securității în VoIP este protocolul descris în capitolul doi, adică SRTP, pentru care a fost dezvoltat ZRTP-ul ca o soluție pentru stabilirea cheilor.

Autentificarea mutuală continuă este o funcționalitate dorită de multe aplicații de securitate. Însă aceasta este de obicei neglijată deoarece este prea lentă. RPM-ul oferă această funcționalitate fără a afecta viteza. Aplicații reale ale RPM-ului au fost implementate în protocolul SSL/TLS, și noul produs a fost numit SSLX (Secure Socket Layer eXtended), ceea ce demonstrază aplicabilitatea lui în protocoale deja existente.

Mă voi axa în continuare pe primul tip de configurație al RPM-ului, descris în capitolul anterior, și anume PDAF Baseline. Scopul acestei lucrări este să arate că este posibil și în ce fel anume se poate extinde protocolul SRTP din VoIP cu RPM.

SRTP și ZRTP au fost concepute pentru a fi compatibile cu protocolul de transport RTP. Acest lucru este dorit și în noua implementare care suportă RPM, din care cauză se dorește o cât mai puțină implicare în felul în care funcționează acestea. Voi descrie în continuare în detaliu, modificările propuse pentru a realiza acest scop.

La fel ca și SRTP, RPM folosește criptare simetrică, ceea ce înseamnă ca are nevoie de niște secrete stabilite înaintea schimbului de mesaje. În cazul SRTP-ului acestea sunt cheia și salt-ul master. În cazul RPM-ului acestea sunt ID1 și ID2, confidențialitatea cărora se dorește a fi asigurată prin alte metode decât RPM. La fel cum cheia și salt-ul master pentru SRTP sunt

generate de ZRTP, ID1 și ID2 pot fi la rândul lor generate de ZRTP, și anume folosind cheia ExternKey, scopul căreia este anume pentru a fi folosită în afara limitelor ZRTP sau SRTP. Pentru a genera din ExternKey ID1 și ID2, se va folosi funcția de generare de chei specifică ZRTP-ului, și anume KDF. În acest mod securitatea cheilor ID1 și ID2 este asigurată la fel ca cea a cheii master din SRTP. Evident acest lucru presupune că, de fiecare dată ZRTP-ul va fi folosit in modul Diffie-Hellman, și nu Preshared.

6. Bibliografie

[1] “Generation of Cryptographic Random Sequences and Its Application to Secure Enciphering” , Hatsukazu Tanaka, 2007

[2] “An Initial Assessment of the 2factor Authentication and Key-Management System: Highlights”, Alan T. Sherman, 2005

[3] “Real Privacy Management™ (RPM) Reference Guide Version v 3.2”, Relevant Security Corp., 2014

[4] “Real Privacy Management™ (RPM) RPM Properties Description for Analysis v 2.2”, Relevant Security Corp., 2014

[5] “Security Analysis of Voice-over-IP Protocols”, Prateek Gupta, Vitaly Shmatikov

[6] “The Secure Real-time Transport Protocol (SRTP)”, RFC 3711, M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, 2004

[7] “ZRTP: Media Path Key Agreement for Unicast Secure RTP”, RFC 6189, P. Zimmermann, A. Johnston, Ed. Avaya, J. Callas, 2011

[8] https://www.packetizer.com/ipmc/papers/understanding_voip/index.html

[9] “Security Issues and Countermeasure for VoIP”, Jianqiang Xin

[10] http://www.rscorp.com/

[11] “Wiretapping End-to-End Encrypted VoIP Calls: Real-World Attacks on ZRTP”, Dominik Schürmann, Fabian Kabus, Gregor Hildermeier, Lars Wolf, 2017