h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Wintersemester 2009 / 2010

Master-Seminar

Future Internet und Next Generation (Mobile) Networks

Media Security: Real-time Transport Protocol, Secure Real-time Transport Protocol und Key Exchange Protocol

von
Marco Münch

bei
Prof. Dr. Michael Massoth

Abstract

In dieser Seminararbeit werden besondere Protokolle für das Thema *Media Security* in Netzwerken besprochen.

Diese Umfasst die Erläuterung wie Audio- und Videodaten mit Hilfe von dem Real-time Transport Protocol übertragen wird. Damit diese unverschlüsselten Daten sicher übertragen werden können, wurde das Secure Real-time Transport Protocol eingeführt, welches ebenfalls in dieser Arbeit besprochen wird.

Um die Sicherheit im Secure Real-time Transport Protocol zu gewährleisten, wird ein Key Management Protocol eingesetzt. Innerhalb dieses Key Management Protocol existieren zur Zeit vier verschiedene Protokolle, welche den gesicherten Schlüsselaustausch vornehmen können: SDP Security Description for Media Streams, Multimedia Key Exchange, Zimmermann RTP und das Datagram Transport Layer Security-Secure Real-time Transport Protocol.

In dieser Arbeit werden alle vier Protokolle vorgestellt und besprochen, der Hauptaugenmerk liegt hierbei auf dem noch recht neuen Zimmermann RTP, welches gründlicher besprochen wird.

Schlüsselwörter: *Media Security, Real-time Transport Protocol, RTP, Real-time Transport Control Protocol, RTCP, Secure Real-time Transport Protocol, SRTP, Secure Real-time Transport Control Protocol, STRCP, Key Management Protocol, SDP Security Description for Media Streams, SDES, Multimedia Key Exchange, MIKEY, Zimmermann RTP, ZRTP, Datagram Transport Layer Security-Secure Real-time Transport Protocol, DTLS-SRTP*

Inhaltsverzeichnis

1 Einleitung

Es werden immer mehr Audio- und Videoinhalte über Netzwerkstrukturen verteilt. Dabei werden sie häufig in Echtzeit übertragen. Dies können Multicast[1]-Übertragungen wie Fußball oder Formel-1-Rennen über IPTV oder Internet-Radio-Streams, aber auch Unicast[2]-Übertragungen über Internet-Telefonie oder Video-Konferenzen sein.

Während im erst genannten Beispiel die Übertragung ungesichert übertragen werden kann, sind Unicast-Übertragungen meist Privat. Daher müssen sie besonders geschützt werden. Diese Übertragungen sollen nicht von Dritten ausgespäht werden, um am Ende damit Schaden anzurichten. Welche Angriffsmöglichkeiten in Netzwerken existieren wird in Kapitel 2 erläutert.

Ein Kryptologisches Verfahren, welches sehr häufig dazu verwendet wird, um einen geheimen Schlüssel zwischen zwei Kommunikationsteilnehmern auszutauschen wird in Kapitel 3 vorgestellt. Dabei handelt es sich um das Diffie-Hellman-Verfahren.

Das weit verbreitete Media-Protokoll RTP, dass in Kapitel 4 besprochen wird, besitzt keinen Ausreichenden Schutz vor Angriffen.

Daher wurde das Protokoll SRTP entwickelt, auf welches in Kapitel 5 eingegangen wird. Es verfügt über eine Echtzeit-Verschlüsselung. Die notwendigen Schlüssel dafür können in vier sogenannten Key Management Protokollen übertragen werden. Es handelt sich dabei um SDP Security Description for Media Streams, Multimedia Internet KEYing, Zimmermann Real−time Transport Protocol und Datagram Transport Layer Security-Secure Real−time Transport Protocol und werden in Kapitel 6 erläutert.

[1] Übertragung von einem Punkt zu einer Gruppe
[2] Übertragung von einem Punkt zu einem anderen Punkt

2 Angriffsszenarien

Es gib unterschiedliche Arten um Voice over Internet Protocol (VOIP)- oder Videosignale, die über das Internet oder dem eigenen Netz übertragen werden, abzuhören. Diese Möglichkeiten werden werden in diesem Kapitel besprochen.

2.1 Sniffing

Bei einem Sniffing Angriff werden die einzelnen Datenpakete von dem potenziellen Angreifer ausgelesen. Bei Übertragung von VOIP- oder Videosignalen können somit mit Hilfe von öffentlichen Programmen wie Womit, Wireshark oder Cain & Abel [Rey08] sehr einfach unverschlüsselte Daten abgehört werden. Über das Internet ist es zwar möglich, aber aufwendig an die Daten zu kommen. Deswegen versuchen Angreifer direkt beim Sender oder Empfänger die Datenpakete auszulesen. Eine Möglichkeit wird auch ARP-Spoofing genannt.

2.1.1 ARP-Spoofing

Normalerweise kann bei einem Switch in einem Netzwerk nicht so einfach abgefangen werden, da die Datenpakete, im Gegensatz zu einem Hub, nur an einem Port in das Teilnetz versendet werden, in dem sich das Zielgerät befindet. Um dieses ARP-Spoofing, zu realisieren schickt der Angreifer jeden der beiden Nutzer eine Address Resolution Protocol (ARP)-Nachricht und gibt sich jeweils als der andere Nutzer aus. Durch die ARP-Nachricht werden die MAC-Adressen der jeweiligen Geräte ausgetauscht. Damit werden nun alle Datenpakete zuerst an den Angreifer gesendet, um sie zu speichern oder anderweitig zu verarbeiten, um sie danach an den richtigen Empfänger zu verschicken. Dies wurde in Abbildung 1 verdeutlicht. [Ber09]

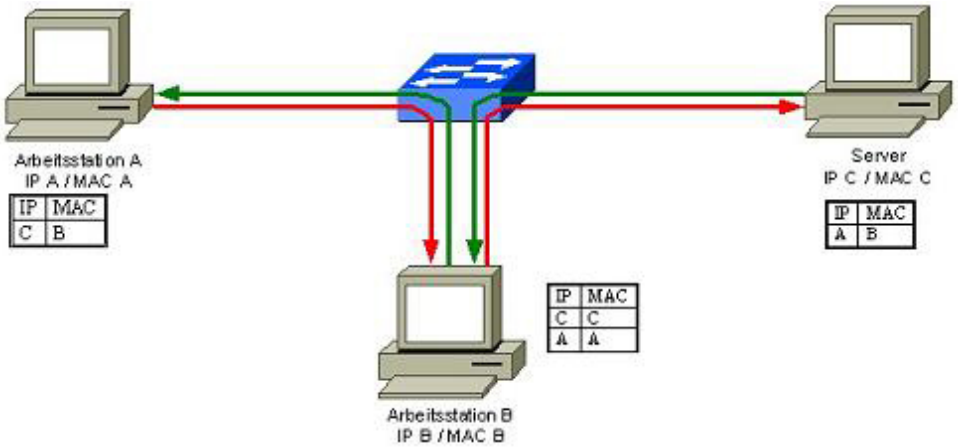

Abbildung 1: ARP-Spoofing [Rue05]

2.2 Man-in-the-Middle Angriff

Bei einem Man-in-the-Middle Angriff klinkt sich der Angreifer in eine Verbindung ein, verarbeitet die Daten und sendet sie weiter zum eigentlichen Kommunikationsteilnehmer, wie in Abbildung 2 zu sehen.

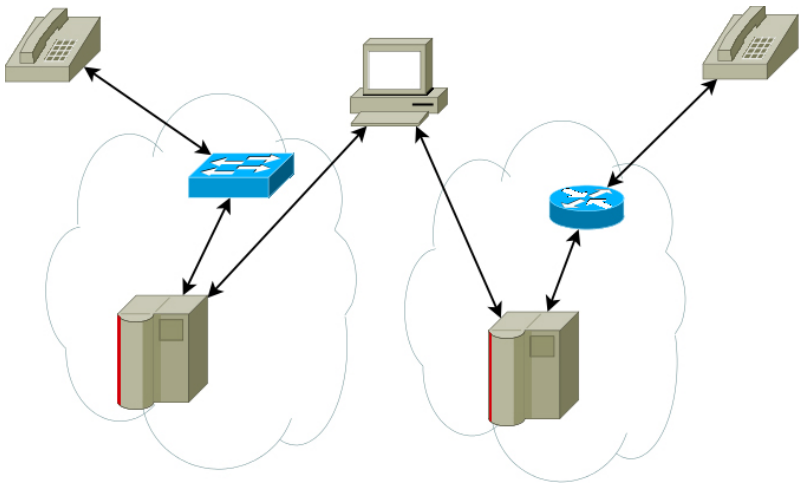

Abbildung 2: Man-in-the-Middle Angriff

Somit kommuniziert die Teilnehmer nicht miteinander, sondern immer nur über den Angreifer. Im Gegensatz zu Sniffing agiert der Angreifer aktiv.

2.3 Replay-Attacken

Bei einer Replay-Attacke werden von einem Angreifer vorher aufgezeichnete Daten einem Nutzer vorgespielt um eine falsche Identität vorzutäuschen. Dies können Audioinhalte sein aber auch Passwörter sein. So kann der Angreifer eine Kommunikation zwischen Nutzer Alice und Bob abfangen und ein übertragenes Passwort aufzeichnen. Anschließend kann der Angreifer mit Hilfe dieses Passwortes die Identität von Bob annehmen [Mal02].

2.4 Weitere Angriffsmöglichkeiten

Neben diesen genannten Attacken, die darauf Zielen an übertragene Media-Inhalte zu kommen, gibt es noch weitere Angriffsmöglichkeiten. Diese Zielen hauptsächlich darauf ab, die Kommunikation der Nutzer zu stören.

Spam over Internet Telefonie Spam over Internet Telefonie (SPIT) Attacken sind eine moderne Version von Spam, also Nachrichten, die einem Empfänger unaufgefordert zugesendet werden. So ist laut einer Studie der Firma Symantec im Mai 2009 über 90% der weltweit verschickten E-Mail Nachrichten dem Spam zuzuordnen [Whi09].

Das liegt daran das der E-Mail versand extrem günstig ist und mit Viren und Trojanern jeder Desktop-Rechner ungewollt zu einem E-Mail Sender werden kann. Zum anderen ist E-Mail sehr weit verbreitet und so können die Spam-Nachrichten viele Personen erreichen.

SPIT wird mit zunehmender Verbreitung der Voice-over-IP-Telefonie immer Beliebter. Die Vorzeichen dafür sind die gleichen wie bei der E-Mail: es wird günstiger und man kann viele Leute erreichen [Her08].

Denial of Service Attack Bei Denial of Service (DoS) oder Distributed Denial of Service (DDoS) Attacken wird der Host-Rechner angegriffen, so dass er seine Dienste nicht mehr richtig verrichten kann. Hierfür wird der Host mit sehr vielen Anfragen bombardiert, so dass er neue Aufträge nicht mehr abarbeiten kann. Wenn die Zeitdauer zu lang ist, werden diese Verbindungen komplett unterbrochen. [Her08]

Um diese Angriffe zu ermöglichen muss eine große Anzahl von Rechnern gleichzeitig den Host-Rechner mit Anfragen überhäufen. Dies wird oft mit Hilfe von sogenannten Bot-Netzwerken erreicht. Es wird mit Hilfe von Viren oder Trojanern fremde Rechner übernommen die zusammen das Bot-Netzwerk bilden.

Diese Angriffsart wurde Mitte 2007 genutzt um verschiedene Institutionen in Estland komplett lahmzulegen. [Tit08]

Van-Eck-Phreaking Das Van-Eck-Phreaking wurde nach Wim van Eck und seiner Arbeit „Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?“ [van07] benannt. Hierbei handelt es sich um ein Verfahren um elektrische Geräte „abzuhören“ indem die elektromagnetischen Signale des Gerätes aufgezeichnet und rekonstruiert werden. Bei dem Versuchsaufbau von van Eck wurden Signale eines Monitors aufgezeichnet und deren Bild sichtbar gemacht. Dieses Verfahren lässt sich generell auf alle elektrischen Geräte ausweiten um an die übertragenen Datensignale zu kommen.

Vishing Ähnlich wie SPIT ist Phishing over VoIP (Vishing) eine Methode die ihren Ursprung bei der E-Mail Versendung hat. Bei Phisching Attacken wird eine Website erstellt, die einer anderen Seiten nachempfunden ist, und potenzielle Opfer mit einer E-Mail auf diese falsche Seite gelockt. Dort soll das Opfer nun persönliche Daten eingeben, wie Kontonummer und Geheimzahl.

Bei Vishing wird diesen Verhalten nachempfunden. So ruft der Angreifer persönlich oder mit einer Bandansage das Opfer an und gaukelt ihn vor Mitarbeiter einer Firma zu sein. Durch diesen „persönlicheren“ Kontrakt ist das Opfer eher geneigt seine Daten herauszugeben als per E-Mail. [Her08]

Um die ersten drei Angriffsmethoden erfolgreich wiederstehen zu können, muss die Kommunikation verschlüsselt werden. Für die Verschlüsselung müssen alle Teilnehmer einen gemeinsamen, aber nur ihnen bekannten Schlüssel besitzen.

Da in einigen Fällen der Schlüssel erst am Anfang der Kommunikation ausgetauscht werden kann, wird ein Verfahren benötigt, das diesen Austausch sichert.

Ein Verfahren, das sich hier als besonders Effektiv herausgestellt hat und daher in dieser Arbeit häufig erwähnt wird, ist der Diffie-Hellman-Schlüsselaustausch, der in Kapitel 3 erläutert wird.

3 Diffie-Hellman-Verfahren

Wie im vorhergehenden Kapitel erwähnt, muss eine verschlüsselte Kommunikation durch einen geheimen Schlüssel gesichert sein. Dieser Schlüssel muss in den meisten Fällen vor der eigentlichen Kommunikation ausgetauscht werden.

Ein sehr häufig eingesetzter Schlüsseltausch ist das sogenannte Diffie-Hellman-Verfahren. Mit Diffie-Hellman ist es möglich einen geheimen Schlüssel aus zwei Teilschlüsseln zu erzeugen, die sich die Teilnehmer gegenseitig zuschicken. Der Schlüsseltausch läuft wie folgt ab: [Mey09]

Schritt 1: Zuerst einigen sich der Initiator (Alice) und der Responder (Bob) auf eine Primzahl p und eine Primitivwurzel g. Die Primitivwurzel ist eine Zahl, die als Potenz der Primitivwurzel dargestellt werden kann. Um eine möglichst guten Schlüssel zu erzeugen benötigt man eine sehr große Primzahl, damit der mögliche Schlüssel nicht durch ausprobieren gefunden werden kann. Für dieses Beispiel wir eine kleine Primzahl verwendet, damit die Übersicht erhalten bleibt.

p=13, g=2

Schritt 2: Alice und der Bob wählen eine Zufallszahl, die nur sie kennen.

Alice	Bob
a = 5	b = 7

Schritt 3: Alice und der Bob berechnen nun $g^{Zufallszahl}\ mod\ p$ und sendet das Ergebnis jeweils dem anderen Teilnehmer zu.

Alice	Bob
$A = 2^5\ mod\ 13 = 6$	$B = 2^7\ mod\ 13 = 11$

Schritt 4: Alice und der Bob berechnen nun den gemeinsamen Schlüssel K mit $erhaltener\ Teilschluessel^{Zufallszahl}\ mod\ p$

Alice	Bob
$K = 11^5\ mod\ 13 = 7$	$K = 6^7\ mod\ 13 = 7$

Angreifer können alle übertragenen Parameter p, g, A und B mithören, aber aufgrund der fehlenden Zufallszahl ist es dem Angreifer unmöglich den geheimen Schlüssel K zu berechnen, da die Berechnung sehr schwer umkehrbar ist.

Problematisch ist der Diffie-Hellman-Schlüsselaustausch nur bei einem Man-in-the-Middle Angriff. Der Angreifer fängt die jeweiligen Teilschlüssel von Alice und Bob ab und schickt im Gegenzug ein eigenen Teilschlüssel. Dies wird in Abbildung 3 verdeutlicht.

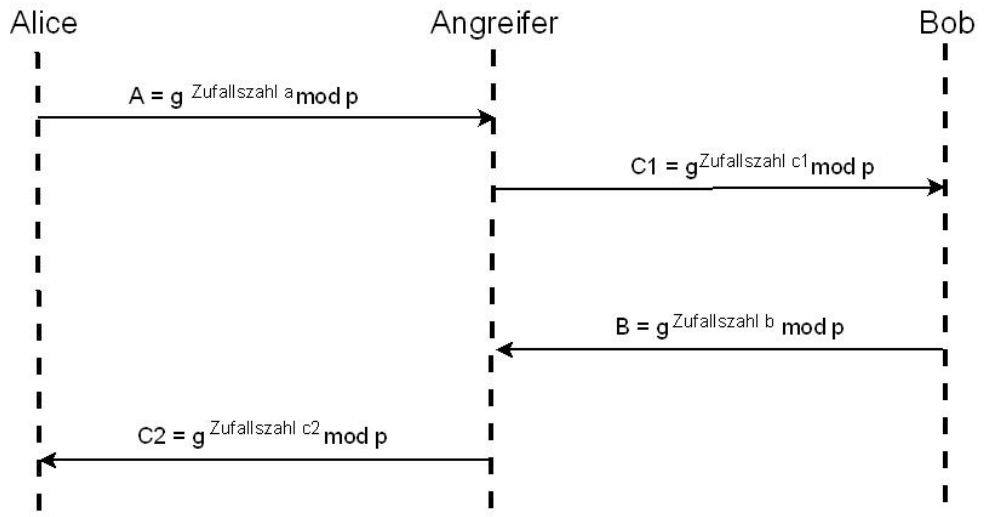

Abbildung 3: Angriff auf Diffie-Hellman

Somit existiert jeweils ein geheimer Schlüssel zwischen Alice und dem Angreifer und Bob und den Angreifer. Daher muss zusätzlich überprüft werden, ob der Schlüssel tatsächlich vom richtigen Teilnehmer stammt.

4 Real-Time Transport Protocol (RTP)

Um Media-Daten wie Bilder oder Sprache zu übertragen werden besondere Protokolle benötigt.

Für Echtzeitorientierte Anwendung, wie Audio- und Videostreams, sind einige Besonderheiten zu beachten:

- die Datenpakete müssen schnell gesendet werden
- geringer Verlust von Datenpaketen ist hinnehmbar

Auf Grund dieser Vorgaben ist Transmission Control Protocol (TCP)/IP mit seinem vielen Informationen überdimensioniert, während User Datagram Protocol (UDP)/IP schnell, aber zu wenig Informationen bietet. So fehlen Informationen wie ein Zeitstempel, aus welcher Quelle das Paket stammt und in welcher Reihenfolge die Pakete beim Empfänger wieder zusammen gesetzt werden müssen. Deshalb wurde ein neues Protokoll entwickelt, welches auf UDP/IP aufsetzt und um die fehlenden Informationen erweitert. Dies ist das Real−time Transport Protocol

Real−time Transport Protocol (RTP) ist ein Protokoll, das Echtzeit-Übertragungen über ein IP-basiertes Netzwerk übertragen kann. Die entsprechenden Daten können sowohl Audio- als auch Videoinhalte sein. Allgemein sind es Anwendungen wie IP-Telefonie, Video-Telefonie oder Video-Streams von Fernsehsendern. RTP wurde zu ersten Mal in Jahr 1996 im RFC 1886 standardisiert [Sch96]. Im July 2003 wurde im RFC 3550 eine bis heute gültige Version von RTP veröffentlicht [Sch03].

Damit diese Echtzeit-Daten über das Netzwerk übertragen werden, müssen sie zuerst in kleine Datenpakete zerlegt werden. Auf diese Pakete wird der sogenannte RTP-Header eingefügt (Siehe Abbildung 4) um das ganze Datenpaket über UDP zu versenden. Der Anwendungsbereich können sowohl Unicast-Übertragungen als auch Multicast-Übertragungen sein.

Abbildung 4: Eingliederung des RTP-Headers

Mit diesem RTP-Header werden nur die reinen Daten übertragen, ein Qualitätskontrolle findet nicht statt. Hierfür wurde ein zusätzliches Protokoll entwickelt, welches Periodisch zwischen den Sender und den Empfänger ausgetauscht wird. Dieses Protokoll heißt Real−time Transport Control Protocol (RTCP). Es werden Quality of Service (QoS) Parameter ausgetauscht damit die jeweilige Seite Veränderungen, zum Beispiel an der Übertragungsrate

(stärkere Codierung für kleinere Datenpakete), vornehmen kann. Dieses Paket wird meistens über TCP übertragen.

In den folgenden Unterkapitel wird das Format dieser beiden Protokolle erläutert.

4.1 RTP-Header

Da RTP darauf Konzipiert wurde Echtzeit-Anwendungen zu übertragen, enthält das Paket auch nur die nötigsten Informationen. In Abbildung 5 wird der Aufbau des RTP-Headers gezeigt.

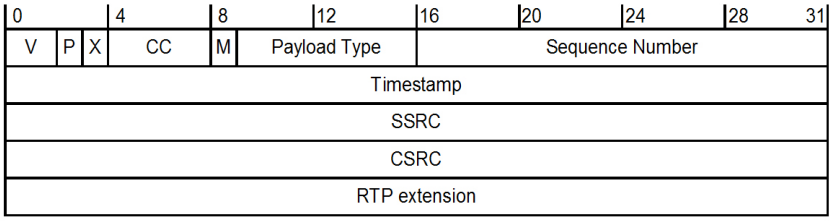

Abbildung 5: RTP-Header [Sis09]

Die einzelnen Abkürzungen stehen für [Bad04]:

- **V** = Version von Secure Real−time Transport Protocol (SRTP), aktuell Version 2
- **P** = Padding, wenn P = 1, wenn das Paket zusätzliche Füllbits enthält
- **X** = Extension, wenn X = 1, kommt ein Erweiterungspaket
- **CC** = CSRC count, Anzahl der zusätzlichen Quellen
- **M** = Marker, hängt vom Profil ab
- **PT** = Payload Type, welche Art von Audio- oder Videodaten das Paket überträgt
- **sequence number** = Sequenz number, zuerst ein Zufallswert, der für jedes weitere Paket um Eins erhöht wird. Damit werden Verluste festgestellt und die Pakete können beim Empfänger in die richtige Reihenfolge gebracht werden.
- **timestamp** = Zeitstempel, dient dazu den Zeitpunkt der Generierung vom Payload zu markieren. Der Anfangswert ist ein Zufallswert. Damit sollen Schwankungen in der Übertragungszeit ausgeglichen werden. Mehrere Pakete können den gleichen Zeitstempel besitzen.
- **synchronization source (SSRC) identifier** = Synchronization Source Identifier, identifiziert eine eindeutige Quelle. Jede Audio- oder Videoquelle vom einem Sender benötigt daher eine eindeutige SSRC.

- **contributing source (CSRC) identifiers** = Contributing Source Identifiers, identifiziert zusätzliche Quellen. Unter Umständen können Multicast-Anwendungen über einen externen Server laufen, der die einzelnen Streams zusammenmixt, so das nur noch ein Stream beim Empfänger ankommt. Diese Quelle wird den CSRC identifiziert.
- **RTP extension (OPTIONAL)** = Das Vorhandensein dieser Erweiterung des RTP-Headers wird durch X kenntlich gemacht. Darin können weitere Daten übermittelt werden, die nicht in diesem Header vorkommen.

4.2 RTCP-Header

Im Gegensatz zu RTP-Header ist der RTCP-Header umfangreicher, da er die ganzen QoS-Parameter überträgt. Es wird zwischen dem Sender Report und dem Receiver Report unterschieden. Während der Sender Report vom Sender ausgehend zu jedem Empfänger geschickt wird, wird der Receiver Report von dem Empfängern zum Sender geschickt. Der Aufbau des Headers ist identisch, nur wird im Packet Type unterschiedliche Flags benutzt.

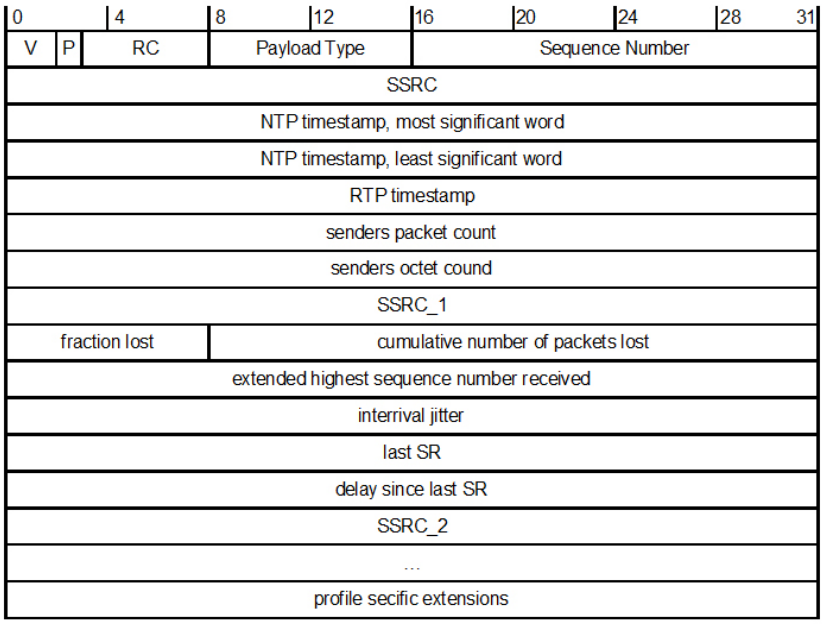

Abbildung 6: RTCP-Header [Sch03]

Der RTCP-Header besteht aus Folgenden Informationen [Bad04], die sich in drei Teile gliedern :

Header

- **V** = Version von SRTP, aktuell Version 2

- **P** = Padding, wenn P = 1, wenn das Paket zusätzliche Füllbits enthält
- **RC** = Reception Report Count, Anzahl der Meldung in diesem Report
- **PT** = Packet Type, PT=200 für Identifikation als Sender Report, PT=201 bedeutet Receiver Report
- **length** = Länge des Paketes
- **synchronization source (SSRC) identifier** = Synchronization Source Identifier, identifiziert eine eindeutige Quelle. Jede Quelle vom einem Sender benötigt daher eine eindeutige SSRC.

Sender-Informationen Den zweiten Block enthält Zeitstempel und Zähler, die folgende Bedeutung haben

- **NTP Timestamp** = Enthält einen Zeitstempel der durch das Network Time Protocol (NTP) synchronisiert wurde. Darin werden die Sekunden seit dem 1. Januar 1900 bis zu aktuellen Zeit übertragen.
- **RTP Timestamp** = Der Zeitstempel dieses RTP Paketes
- **Sender's Packet Count** = Anzahl der bisher gesendeten RTP Pakete
- **Sender's Octet Count** = Anzahl der bisher gesendeten Bytes innerhalb der RTP Pakete

Report Block Im Report Block werden die einzelnen Informationen zu den einzelnen Quellen übertragen. Ein Paket kann mehrere Report Blocks enthalten.

- **SSRC-n** = Identifiziert eine eindeutige Quelle zu dem dieser Report Block gehört.
- **fraction lost** = Verlustquote der RTP-Paketen seit dem letzten Sende Report
- **cumulative number of packets lost** = Anzahl aller verlorenen RTP-Pakete seit Beginn der Übertragung
- **extended highest sequence number received** = niederwertigsten 16 bit geben die höchste Sequenznummer an, die angekommen sind. Die höherwertigen 16 bit enthalten die Anzahl der empfangenen Sequenznummern
- **interarrival jitter** = Ergebnis einer Berechnung die den Jitter angibt
- **last SR timestamp** = Zeitstempel aus dem letzten Sender Report
- **delay since last SR** = Verzögerung zwischen dem Empfangen des letzten Sender Reports und dem senden des Report Blocks

Das Übertragungsprotokoll RTP besitzt keine Schutzmöglichkeiten um Angriffe wie Sniffing oder Man-in-the-Middle Attacken zu verhindern. Daher wurde das Protokoll um die Möglichkeit einer gesicherten Verbindung erweitert, welch im folgenden Kapitel erläutert wird.

5 Secure Real-time Transport Protocol

Die Übertragung über das RTP besitzt den Nachteil das sämtliche Audio- und Videoinhalte unverschlüsselt übertragen werden. Mögliche Hacker können sich an Knotenpunkten einklinken und dort die empfangenen Pakete abfangen und damit abhören. Um das zu verhindern müssen die Inhalte verschlüsselt werden. Hierfür sollte RTP erweitert werden, aber folgende Punkte [Sis09] beachten :

1. Header mit möglichst geringer Größe
2. Headerkompression soll möglich sein
3. Entschlüsselung trotz fehlender Pakete
4. Schnelle Ver- und Entschlüsselung
5. Einfacher Kryptographischer Algorithmus

Aus diesen Vorgaben wurde im Jahre 2004 das SRTP im RFC 3711 [Sch03] vorgestellt. Wie auch RTP besitzt SRTP ein Control Protocol, das Secure Real−time Transport Control Protocol (SRTCP). Das Format der beiden Header wird in den Kapiteln 5.2 und 5.3 erklärt. Der generelle Aufbau der SRTP-Architektur kann der Abbildung 7 entnommen werden.

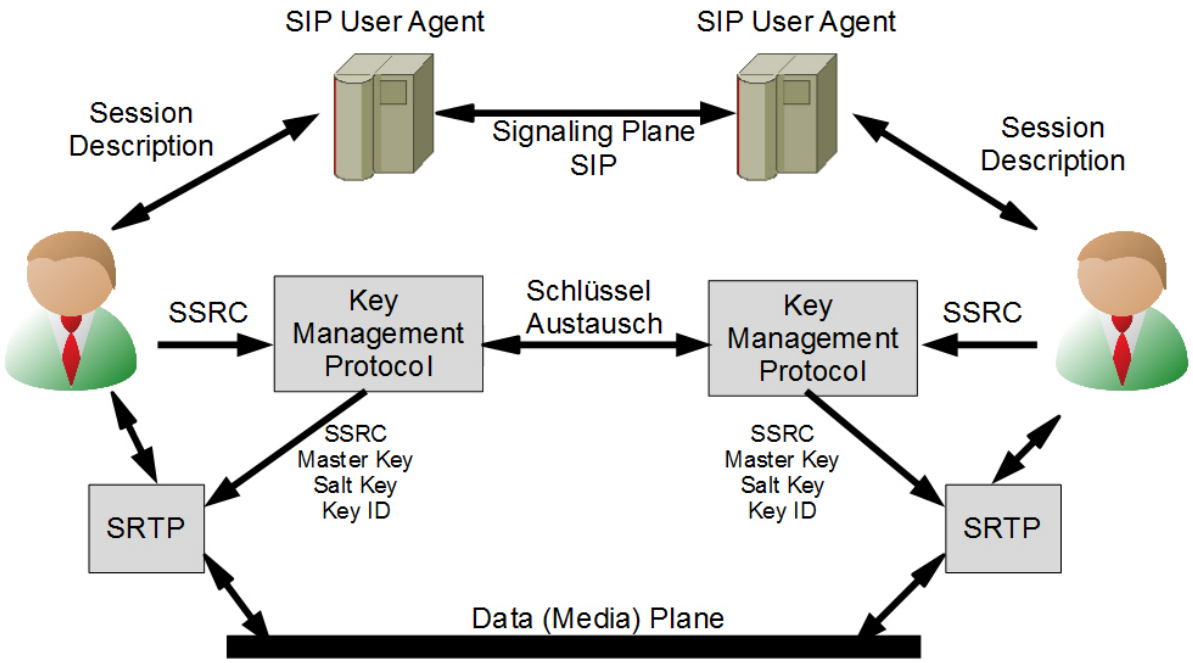

Abbildung 7: SRTP Aufbau

Ein wichtiger Aspekt der Architektur liegt darin, dass zwei getrennte Verbindungen existieren. Eine Verbindung realisiert die Authentifizierung der Teilnehmer mit Hilfe einer gesicherten Session Initiation Protocol (SIP)-Architektur, der sogenannten *Signaling Plane*. Über die andere Verbindung, genannt *Data Plane*, werden die Media-Daten verschlüsselt übertragen.
Der Aufbau der Verbindung erfolgt in drei Schritten:

Der **erste Schritt** ist die Authentifizierung der einzelnen Teilnehmer. Hierfür wird eine SIP-Sicherheitsarchitektur benötigt, welche die nötigen Sicherheitsmechanismen in den SIP User Agents beinhalten. [Sis09]

Der Austausch verläuft über die Signaling Plane und beinhaltet die Authentifizierung der Teilnehmer, welche Verschlüsselung angewendet werden soll (Siehe Anhang A) und über welches Key Management Protocol die zur Verschlüsselung benötigten Schlüssel übertragen werden.

Nachdem sich die Teilnehmer untereinander bekannt gemacht haben erfolgt der **zweite Schritt:** Der Austausch der Schlüssel über ein Key Management Protocol. Dabei handelt es sich um vier standardisierte Möglichkeiten den Schlüssel auszutauschen.
Da wären zum einen das SDP Security Description for Media Streams und Multimedia Internet KEYing, die den Schlüsselaustausch über die SIP User Agents vornimmt.
Bei den anderen handelt es sich um Zimmermann Real−time Transport Protocol und Datagram Transport Layer Security-Secure Real−time Transport Protocol dessen Austausch der Schlüssel über die Data Plane vorgenommen wird.
Wie diese Key Management Protokolle im einzelnen Funktionieren wird im Kapitel 6 erläutert.

Das Protokoll selbst erzeugt für jede Medienquelle einen eindeutigen Synchronization Source Identifier (SSRC) um jedes Datenpaket einem oder mehreren Quellen zu ordnen zu können.

Der Master Key und der Master Salt sind zufällige Schlüssel, die 128 Bit und 112 Bit lang sind, und dazu dienen den endgültigen Schlüssel zu berechnen (Siehe Anhang A). Wie das geschieht wird in Kapitel 5.1 gezeigt.

Die Key ID dient dazu den Schlüssel zu Zertifizieren, welcher aktuell für die Verschlüsselung verwendet wird. Es handelt sich dabei um einen Hash-Wert Schlüssels, der als SRTP Authentifizierung tag im Protokoll übertragen wird.

Der SSRC, der Master Key, der Master Salt und die Key ID werden dann mit Secure Real−time Transport Protocol übertragen.

Im **dritten** und letzten Schritt kann die eigentliche Übertragung beginnen. Es wird der Medien-Daten-Strom in Päckchen zerteilt, wie in Kapitel 5.1 dargestellt, verschlüsselt und anschließend über die Data Plane übertragen.

5.1 Ver- und Entschlüsselung

Die Übertragung des Master Key und des Master Salt Schlüssel reicht nicht aus um den Media-Stream zu verschlüsseln. Aus diesem Grund wird der Media-Stream, bevor er über die Media Plane übertragen wird, über eine zusätzliche Funktionen verschlüsselt. Diese sieht wie Abbildung 8 aus:

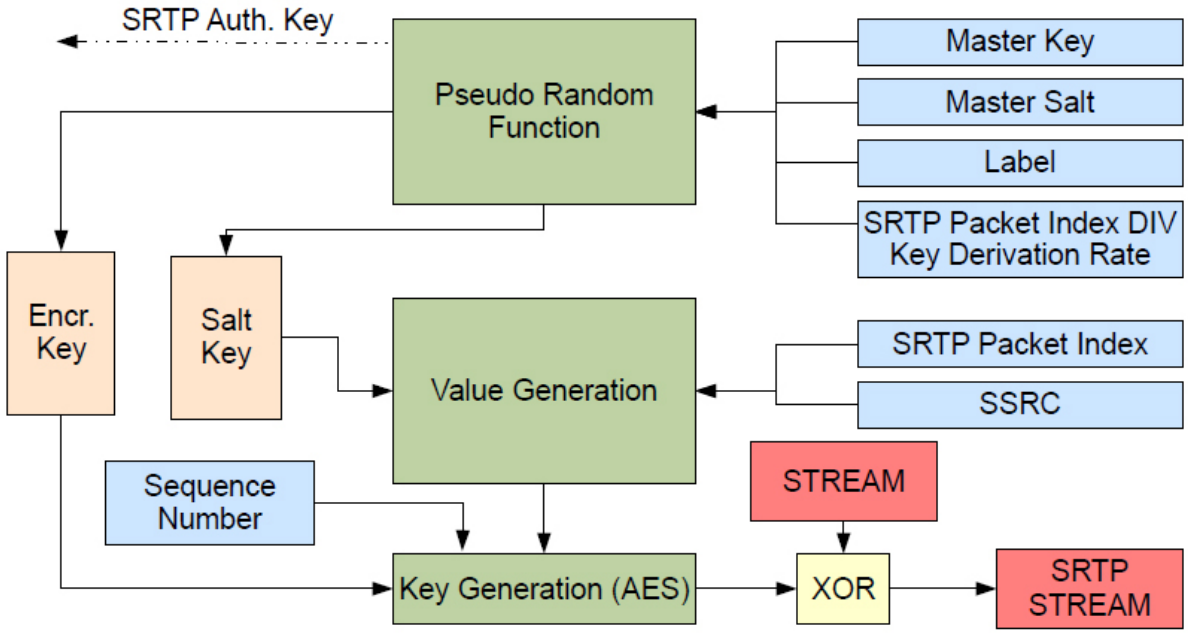

Abbildung 8: Verschlüsselung SRTP

Zuerst werden der wichtige Encryption Key und der Salt Key durch eine **Pseudo-Random-Function** erzeugt. Als Parameter benötigt die Pseudo-Random-Function die durch das Key Management Protocol übertragenen *Master Key* und den *Master Salt*. Zusätzlich benötigt sie ein *Label*. Das Label besteht aus sechs vorher ausgemachten Parametern. Der vierte und letzte Parameter wird aus zwei weiteren Elementen berechnet. Es wird der *SRTP Packet Index* durch die *Key Derivation Rate* geteilt. Falls vor der eigentlichen Übertragung keine Key Derivation Rate ausgehandelt wurde, so fließt dieser Parameter als Null in die Pseudo-Random-Function ein. Zusätzlich generiert die Pseudo-Random-Function einen SRTP Authention Key. Bei ihm handelt es sich um einen Hashwert, der den Encryption Key und den Salt Key authentifiziert. Dies wurde eingeführt, um bei dem Wechsel eines Schlüssels auf den anderen zu erkennen.

Im nächsten Schritt wird ein Wert in der **Value Generation** erstellt. Dazu wird der *Salt Key*, der *SRTP Packet Index* und der *Synchronization Source Identifier* benötigt.

Dieser Wert fließt nun mit dem *Master Key* und der *Sequenznummer* des Paketes in die **AES Key Generation** ein. Dort wird nun der endgültige Schlüssel erstellt und mit dem Media-Stream mit XOR verknüpft. Das Ergebnis daraus ist nun die verschlüsselten Media-Daten, die über SRTP und der Media Plane übertragen werden. Der Empfänger der Daten berechnet den Schlüssel auf die gleich Art und Weise und verknüpft den verschlüsselten Daten-Strom ebenfalls mit XOR und erhält damit die unverschlüsselten Daten.

Die SRTCP-Daten werden auf die gleich Weise berechnet, nur wird statt dem *SRTP Packet Index* der *SRTCP Packet Index* verwendet.

5.2 SRTP-Header

Das SRTP bietet die Möglichkeit den Payload zu verschlüsseln. Dazu wurde der RTP-Header leicht abgeändert, wie in Abbildung 9 zu sehen.

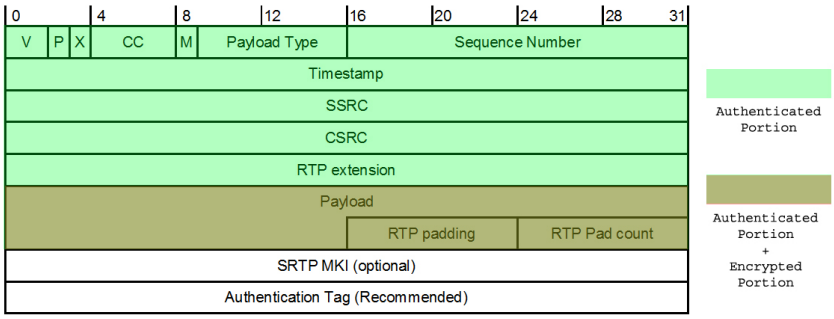

Abbildung 9: SRTP-Header [Bau04]

Der Header setzt sich wie folgt zusammen [Par06]:

- **V** = Version von SRTP, aktuell Version 2
- **P** = Padding, wenn P = 1, wenn das Paket zusätzliche Füllbits enthält
- **X** = Extension, wenn X = 1, kommt ein Erweiterungspaket
- **CC** = CSRC count, Anzahl der zusätzlichen Quellen
- **M** = Marker, hängt vom Profil ab
- **PT** = Payload Type, welche Art von Daten das Paket überträgt

- **sequence number** = Sequenz number, zuerst ein Zufallswert, der für jedes weitere Paket um Eins erhöht wird. Damit werden Verluste festgestellt und die Pakete können beim Empfänger in die richtige Reihenfolge gebracht werden.
- **timestamp** = Timestamp, dient dazu den Zeitpunkt der Generierung vom Payload zu markieren. Der Anfangswert ist ein Zufallswert. Damit sollen Schwankungen in der Übertragungszeit ausgeglichen werden. Mehrere Pakete können den gleichen Zeitstempel besitzen.
- **synchronization source (SSRC) identifier** = Synchronization Source Identifier, identifiziert eine eindeutige Quelle. Jede Quelle vom einem Sender benötigt daher eine eindeutige SSRC.
- **contributing source (CSRC) identifiers** = Contributing Source Identifiers, identifiziert zusätzliche Quellen. Unter Umständen können Multicast-Anwendungen über einen externen Server laufen, der die einzelnen Streams zusammenmixt, so das nur noch ein Stream beim Empfänger ankommt. Diese Quelle wird den CSRC identifiziert.
- **RTP extension (OPTIONAL)** = Das Vorhandensein dieser Erweiterung des RTP-Headers wird durch X kenntlich gemacht. Darin können weitere Daten übermittelt werden, die nicht in diesem Header vorkommen.
- **payload** = Enthält die eigentlichen Mediendaten in verschlüsselter Form.
- **RTP padding** = Füllbits
- **RTP pad count** = Anzahl der Füllbits
- **SRTP MKI (OPTIONAL)** = Der Master Key Identifier (MKI) identifiziert den Master Key der aus den Session Keys abgeleitet wurde
- **authentication tag** = Das Feld enthält einen Authentifizierung Wert über den SRTP-Header und dem verschlüsselten Payload.

5.3 SRTCP-Header

- **V** = Version von SRTCP, aktuell Version 2
- **P** = Padding, wenn P = 1, wenn das Paket zusätzliche Füllbits enthält
- **RC** = Reception Report Count, Anzahl der Meldung in diesem Report
- **PT=SR or RR** = Welcher Art der Payload ist.
- **length** = Länge des Paketes
- **SSRC of sender** = Synchronization Source Identifier des Senders, identifiziert eine eindeutige Quelle. Jede Quelle vom einem Sender benötigt daher eine eindeutige SSRC.

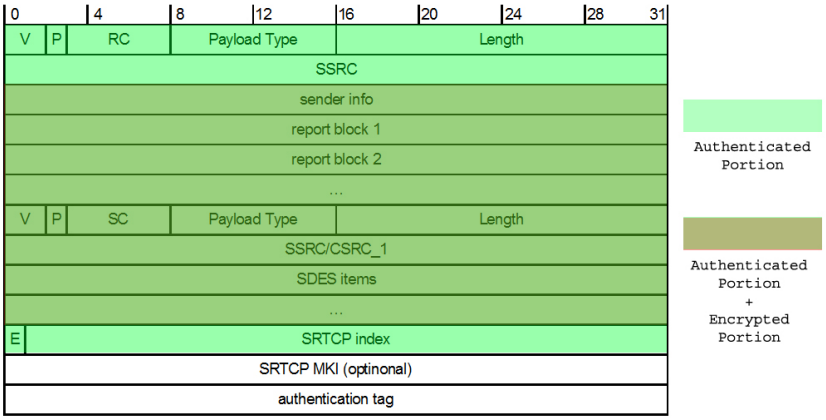

Abbildung 10: SRTCP-Header [Bau04]

- **sender info** = Hier befindet sich ein kompletter Block mit Sender-Informationen. Siehe dazu Kapitel 4.2.
- **report block** = Hier sind nun einen oder mehrere Report Blocks. Siehe dazu Kapitel 4.2.
- **SC** = Sender Count
- **PT** = Payload Type, welche Art von Daten das Paket überträgt. Für SDP Security Description for Media Streams (SDES) in diesem Fall 202
- **SSRC/CSRC_1** = Synchronization Source Identifier bzw. Contributing Source Identifiers des aktuellen Paketes.
- **SDES items** = SDES items
- **E** = Das E-flag ist 1 wenn der vorhergehende Teil verschlüsselt nicht, NULL wenn er nicht verschlüsselt ist.
- **SRTCP index** = Jedes SRTCP Paket wird durchnummeriert. Es beginnt bei NULL und wird immer um Eins erhöht.
- **SRTCP MKI** = SRTCP MKI
- **authentication tag** = Das Feld enthält den SRTP Authention Key über den SRTP-Header und dem verschlüsselten Payload.

6 Key Management Protocol

Damit SRTP die notwendigen Schlüssel sicher übertragen kann, wurden vier unterschiedliche Key Management Protokolle entwickelt: SDP Security Description for Media Streams, Multimedia Internet KEYing, Zimmermann Real−time Transport Protocol und Datagram Transport Layer Security-Secure Real−time Transport Protocol.

Auf diese unterschiedlichen Protokolle wird in den folgenden Unterkapiteln näher eingegangen.

6.1 SDP Security Description for Media Streams

SDES wurde im July 2006 von der Internet Engineering Task Force (IETF) im RFC 4568 [And06] standardisiert und realisiert einen Zwei-Personen-Schlüsselaustausch. Wie der Name SDP im Namen schon andeutet wird der Schlüssel mit Hilfe einer sicheren SIP-Verbindung, der Signaling Plane, übertragen (Siehe Abbildung 7).

6.1.1 Schlüsselübertragung

Die Schlüssel, Master Key und Master Salt, werden bei SDES unverschlüsselt über die Signal Plane, also auch über die SIP-Server übertragen. Die Übertragung zwischen User und SIP-Server ist durch Secure / Multipurpose Internet Mail Extensions (SMIME) oder Transport Layer Security (TLS) gesichert.

Die Übertragung findet im *SDP offer* und in der Antwort *SDP answer* statt. Dafür wurde ein neues SDP Attribut mit dem Namen *crypto* eingeführt. Es besitzt folgende Struktur:

$$a = crypto :< tag >< crypto\_suite >< key - params > [< session - params >]$$

Die Tags haben folgende Bedeutung [Sis09]:

- <tag> = Dieses Feld enthält eine eindeutige Nummer und dient dazu während des Austausches die einzelnen Parameter einem Datenstrom in der Media Plane zuzuordnen
- <crypto_suite> = Gibt den Verschlüsselung-Algorithmus an, mit dem der Media-Stream verschlüsselt wird. Standardmässig können drei Algorithmen angewendet werden:
  - AES_CM_128_HMAC_SHA1_80 : AES Counter Mode Verschlüsselung mit 80 bit SRTP Authentifizierung tag
  - AES_CM_128_HMAC_SHA1_32 : AES Counter Mode Verschlüsselung mit 32 bit SRTP Authentifizierung tag

– F8_128_HMAC_SHA1_80 : AES F8 Mode Verschlüsselung mit 80 bit SRTP Authentifizierung tag

Mehr über die einzelnen Eigenschaften im Anhang A.

- <key-params> = In diesem Feld werden die Schlüssel übertragen

  $inline : base64(< key > \| < salt >)[\| < lifetime >][\| < MKI >:< MKI - Lenght >]$

  – $base64 < key > \| < salt >)$ = Der Master Key und der Master Salt kodiert als base64.

  – $< lifetime >$ = Optional ist die maximale Lebensdauer dieses Schlüssels. Falls keine Lebensdauer angegeben wurde beträgt der Standard Wert 2^{48} SRTP-Pakete und 2^{31} SRTCP-Pakete.

  – $< MKI >:< MKI - Lenght >$ = Optional kann auch der Master Key Identifier und dessen Länge angeben werden.

- <session-params> = Dieser optionale Wert bietet unterschiedliche Einstellungsmöglichkeiten, die im weiteren näher erläutert werden:

  – KDR - Key Derivation Rate, gibt an wie oft ein Encryption Key und ein Encryption Salt durch die Pseudo-Random Funktion erstellt wird.

  – UNENCRYPTED_SRTP, UNENCRYPTED_SRTCP und UNAUTHNTICATED_SRTP gibt an ob einige Teile SRTP/SRTCP Pakete nicht verschlüsselt werden.

  – WSH - Window Size Hint wird für die Replay Protection [Bau04] benötigt

  – FEC_ORDER - gibt an ob Forward Error Correction vor (FEC_SRTP) oder nach (SRTP_FEC) SRTP benutzt wird. Standardwert ist FEC_SRTP. [And06]

  – FEC_KEY - wenn die Forward Error Correction über einen eigenen Stream übertragen wird, so benötigt dieser einen eigenen Master Key und einen Master Salt. Dieser wird im selben Format wie <key-params> übertragen. [And06]

Als Veranschaulichung eines SIP-Austausches von SDP offer und SDP answer dient folgendes Beispiel [And06]:

SDP offer

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
```

```
  FEC_ORDER=FEC_SRTP
a=crypto:2 F8_128_HMAC_SHA1_80
  inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
  inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
  FEC_ORDER=FEC_SRTP
```

SDP answer

```
v=0
o=jill 25690844 8070842634 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=homer@example.com (Homer Simpson)
c=IN IP4 168.2.17.11
t=2873397526 2873405696
m=audio 32640 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

6.2 Multimedia Internet KEYing

Multimedia Internet KEYing (MIKEY) wurde von der IETF als RFC 3830 [Ark04] veröffentlicht. Wie schon SDES (Siehe Kapitel 6.1) überträgt Multimedia Internet KEYing (MIKEY) die Schlüssel über die Signaling Plane (Siehe Abbildung 7). Aber im Gegensatz zu SDES werden die Schlüssel nicht als SDP Attribut ausgetauscht, sondern direkt verschlüsselt im Payload dieser Nachrichten.

6.2.1 Schlüsselübertragung

MIKEY bietet zur Übertragen des Master Key und des Master Salt insgesamt fünf verschiedene Methoden: mit Hilfe eines Pre-shared Key (PSK), über zwei Verfahren der RSA-Verschlüsselung (RSA & RSA-Reverse) und über zwei Arten eines Diffie-Hellman Schlüsselaustausch (DHSIGN & DHMAC).

MIKEY-PSK

Bei dem MIKEY-PSK Verfahren wird der Master Key und der Master Salt vor der ersten Kommunikation übertragen und durch einen privaten Schlüssel verschlüsselt. Der private Schlüssel wurde zwischen den Nutzern auf einen anderen Weg vorher ausgetauscht. Das kann über Datenträger wie CD oder USB-Stick sein, aber auch über ein anderes Kommunikationsmedium wie E-Mail. Mit diesem Schlüssel werden Teile des ersten Austausches verschlüsselt.

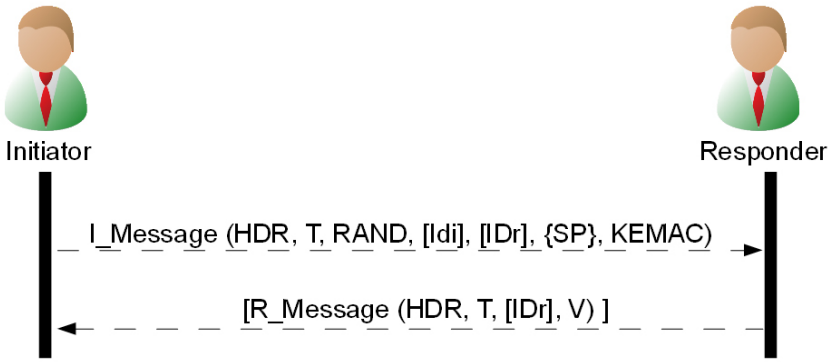

Abbildung 11: Mikey-PSK

Der eigentliche Austausch erfolgt wie in Abbildung 16 zu sehen. Der Initiator muss eine Mitteilung (I_MESSAGE) an den Responder schicken. Der Header (HDR) enthält die Crypto Session Bundle ID (CSBID) mit der dazugehörigen Crypto Session ID Map (CSIDMAP). Ebenso wird ein Zeitstempel (T) und ein zufälliger Wert (RAND) übertragen. RAND wird zur Erzeugung

des Master Key und des Master Salt benötigt. Als nächstes können optional eine ID des Initiators (IDi) oder des Responders (IDr) angeben werden. Danach werden die Security Policy (SP) übertragen. Das können Null oder mehr SP sein. Am Ende wird Key Data Transport Payload (KEMAC) übertragen. Dieser enthält den Encryption Key, einen oder mehrere verschlüsselte TEKs und ein Message Authentication Code über die gesamte MIKEY Mitteilung.

$$KEMAC = E(encr\_key, \{TGK\}) \mid\mid MAC$$

Optional kann der Responder eine Verifizierung der Mitteilung zurückschicken. Dazu wird aus dem Zeitstempel (T) des Initiators, der ID des Initiators (IDi), der ID des Responders (IDr) und der MAC dieser Nachricht eine Verification Message (V) berechnet und an den Initator geschickt.

PSK wird für One-to-One und sehr kleinen One-to-Many Übertragungen verwendet. Grund hierfür liegt in der problematischen Skalierbarkeit der Schlüsselweitergabe.

MIKEY-RSA

Mit MIKEY-RSA wird der Schlüsselaustausch mit einem Public-Key-Verfahren gesichert. Dafür wird eine Struktur benötigt, die digitale Zertifikate ausstellen und diese Prüfen kann.

Es wird ein öffentlicher Schlüssel (Public-Key (PK)) benötigt, der öffentlich zur Verfügung gestellt wird. Es muss dabei aber sichergestellt werden, das dieser Schlüssel keinem Betrüger gehört. Um dies zu vermeiden ist der PK durch ein digitales Zertifikat gesichert, das bei einer Zertifizierungsstelle authentifiziert wird.

Der Encryption Key, die verschlüsselten TEKs und der Message Authentication Code werden mit Hilfe eines sogenannten Envelope Key verschlüsselt:

$$KEMAC = E(encr\_key, IDi \mid\mid \{TGK\}) \mid\mid MAC$$

Der Envelope Key wird durch den PK verschlüsselt im Public-Key Envelope (PKE) an den Responder gesendet:

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

Falls der Responder über mehrere PK verfügt, so kann dieser mit dem Cert hash payload (CHASH) identifiziert werden. In CHASH wird der PK als

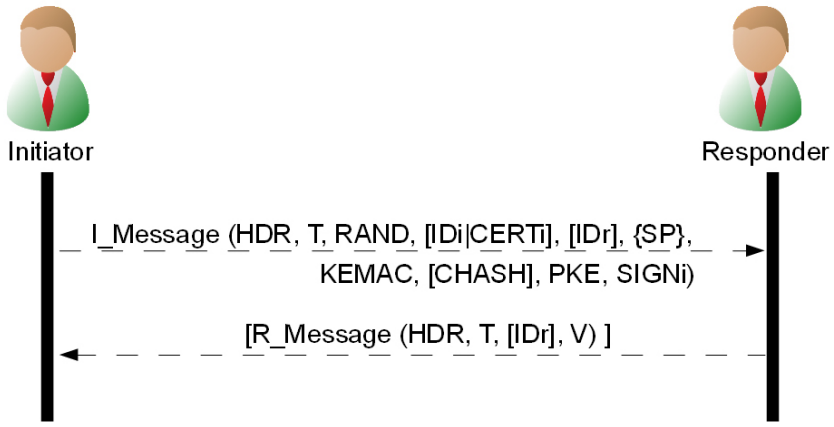

Abbildung 12: Mikey-RSA

Hashwert übertragen.

MIKEY-DHSIGN

Ähnlich wie bei den anderen beiden Verfahren sieht der Schlüsselaustausch wie in Abbildung 13 aus:

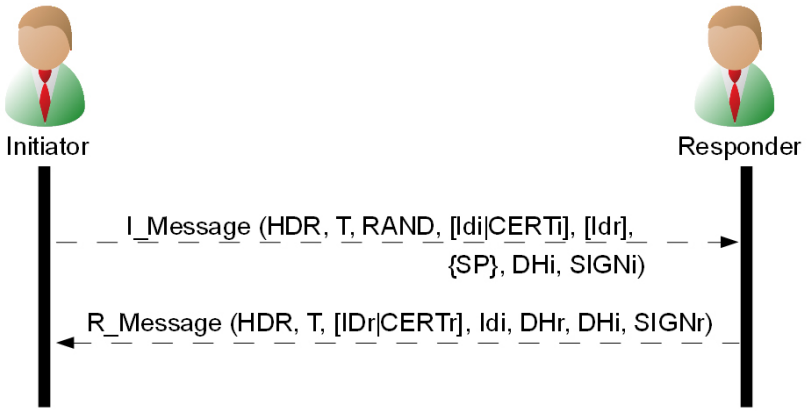

Abbildung 13: MIKEY-DHSIGN

Besonders hervorzuheben ist dieses Fall der Schlüssel nach dem Diffie-Hellman-Verfahren ausgetauscht wird (Siehe Kapitel 3. In MIKEY-Austausch werden sie mit den Kürzeln DHi, für Diffie-Hellman Initiator, und DHr, für Diffie-Hellman Responder, abgekürzt. Darin werden die jeweiligen TeilSchlüssel mit übertragen.

MIKEY-DHMAC

Diese Variante liefert den Schlüssel mit Hilfe des Diffie-Helman-Austausch aber die Benutzer Authentifizierung über einen Public Key ähnlichen Schema. Diese Variante war, im Gegensatz zu den anderen drei Varianten nicht im MIKEY Standard enthalten, sondern wurde von der IETF als RFC 4650 [Euc06] nachtraäglich standardisiert. Da eine Public Key Infrastruktur aufwendig ist bzw. nicht immer verfügbar ist fungiert der MIKEY als ein Group key management architecture (GKMARCH). Mehrere Nutzer können sich beim Responder registrieren und handelt mit ihm einen Diffie-Hellman Schlüssel aus.

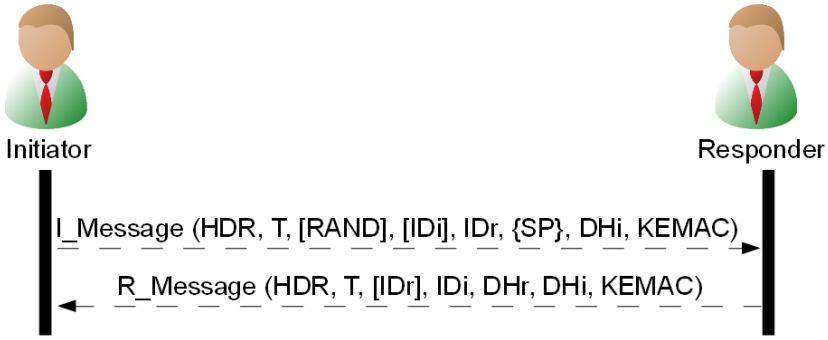

Abbildung 14: MIKEY-DHMAC

Wie in Abbildung 14 verläuft der übrige Schlüsseltausch wie bei MIKEY-DHSIGN.

Im Gegensatz zu den anderen drei Vorgestellten Varianten eignet sich MIKEY-DHMAC auch für die verschlüsselte Übertragung in kleineren Gruppen.

MIKEY-RSA-Reverse

MIKEY-RSA-Reverse basiert auf MIKEY-RSA, wobei diese Erweiterung erst 2006 nachträglich erarbeitet wurde und als RFC 4738 [Ign06] veröffentlicht. Es wurde erarbeitet um zwei Szenarios, die MIKEY-RSA nicht abdeckt, zu ermöglichen.

Dabei handelt es sich zum einen um den Fall, wenn der Initiator die Indentiät des Responders nicht bestimmen kann. Bei diesem Szenario handelt es sich typischerweise um ein Unicast-Szenario.

Bei dem anderen Fall handelt es sich um ein Multicast-Szenario, dass den TEK Generation Key (TGK) Schlüssel über einen externen Zugang für alle Teilnahmer zur Verfügung stellt.

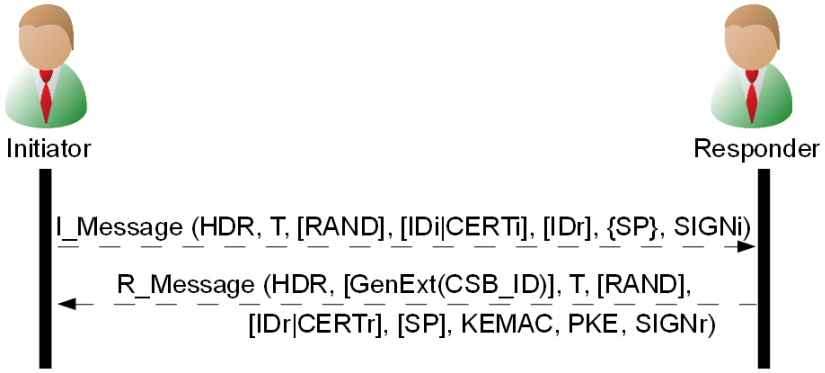

Abbildung 15: MIKEY-RSA-R im Unicast Mode

Im MIKEY-RSA-Reverse Unicast Mode spezifiziert der Initiator die Security Policy und die CS ID Map der RTP Streams und sendet sie an den Responder. Der Responder generiert nun den Envelope Key. Dazu überprüft er die Signatur SIGNi mit dem Zertifikat CERTi und verschlüsselt damit den Envelope Key. Mit der R_Message wird der Envelope Key im PKE an den Responder zurückgesendet.

In MIKEY-RSA-Reverse übernimmt, im Gegensatz zu MIKEY-RSA, der Reponder die Erstellung des Envelope Key.

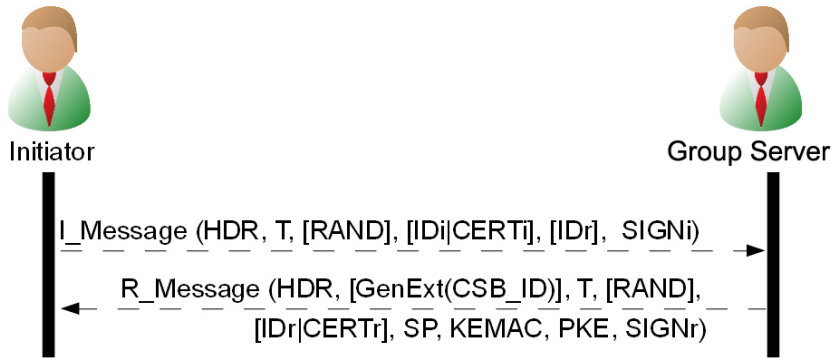

Abbildung 16: MIKEY-RSA-R im Multicast Mode

Im MIKEY-RSA-Reverse Multicast Mode benötigt eine Gruppen-Management-Architektur, welche die einzelnen Teilnehmer verwaltet. Der Schlüsselaustausch erfolgt über diesen Server. Das Verfahren ist absonsten identisch mit dem Unicast Mode.

6.2.2 Schlüsselgenerierung

Im Gegensatz zu SDES wird der Encryption Key und der Salt Key über einen Algorithmus berechnet, der in Abbildung 17 zu sehen ist.

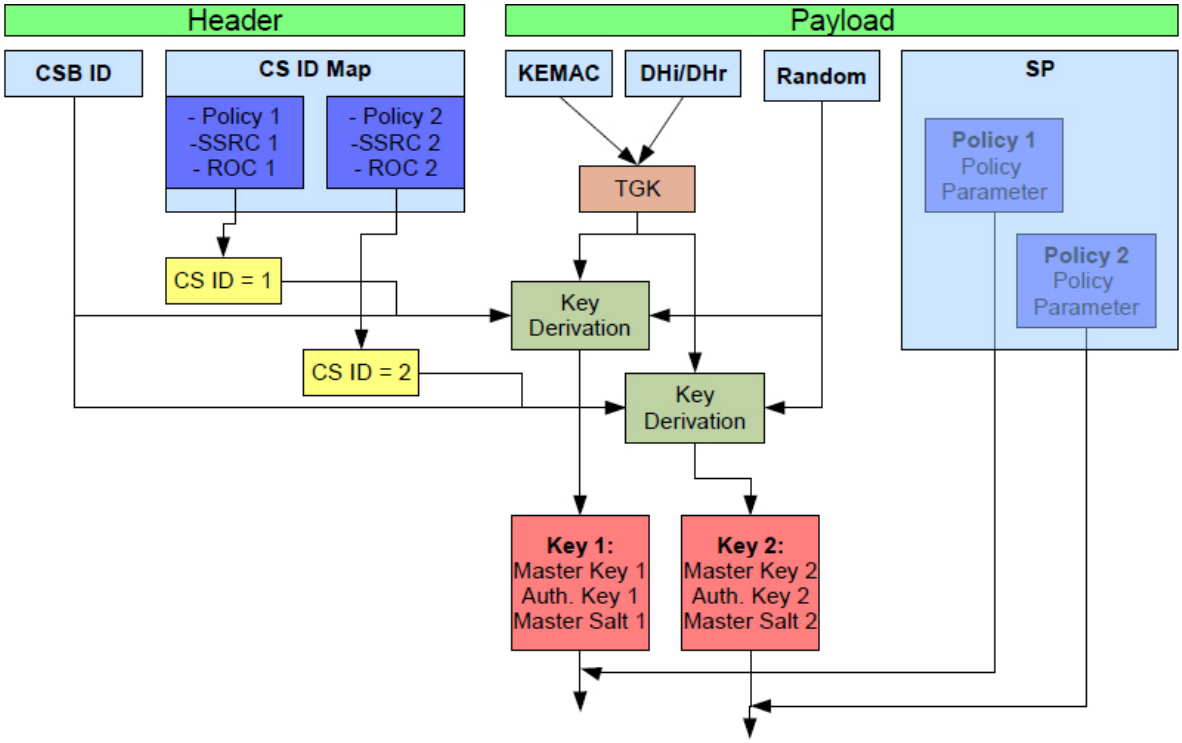

Abbildung 17: Schlüsselgenerierung MIKEY

Für die Berechnung (**Key Derivation**) wird zuerst eine CSBID und eine Crypto Session (CS) benötigt. Die CS befindet sich innerhalb einer CSIDMAP und enthält für die eindeutige Identifizierung eines Datenstroms den entsprechenden SSRC, einen Rollout Counter (ROC) und eine Verschlüsselungs-Verfahren (Policy). Welches Verschlüsselungs-Verfahren verwendet wird, kann durch mehrere SP ausgehandelt werden, die im Payload des MIKEY-Protokolls mit übertragen wird.

Für die Schlüsselerzeugung wird zusätzlich ein TGK benötigt, der entweder über einen KEMAC (PSK, RSA, RSA-Reverse) oder einen Diffie-Hellman Austausch (DHSIGN, DHMAC) übertragen / erstellt wird.

Zu guter Letzt wird ein Zufälliger Wert (random) im Payload übertragen um den Master Key, Master Salt und einen Authentification Key zu erzeugen.

6.3 Datagram Transport Layer Security-Secure Real-time Transport Protocol

Das Datagram Transport Layer Security-Secure Real−time Transport Protocol (DTLS-SRTP) überträgt seine Daten über die Data Plane. Wie alle vorhergehenden Protokolle ist DTLS-SRTP ein Standard und wurde von der IETF als RFC 4650 [Res06] veröffentlicht. DTLS-SRTP basiert auf dem weit verbreitenden Standard TLS, der eine Kommunikation auf OSI-Schicht 6 (Darstellungsschicht) sichert. TLS ist nur für TCP ausgelegt, bei Echtzeit-Anwendungen wie Audio und Video wird, wie in Kapitel 4 besprochen, das verbindungslose Protokoll UDP benötigt. Daher wurde TLS zusätzlich um UDP erweitert und ist als DTLS-SRTP benannt.

6.3.1 Schlüsselgenerierung

In Abbildung 18 wird die Schlüsselgenerierung von DTLS-SRTP gezeigt.

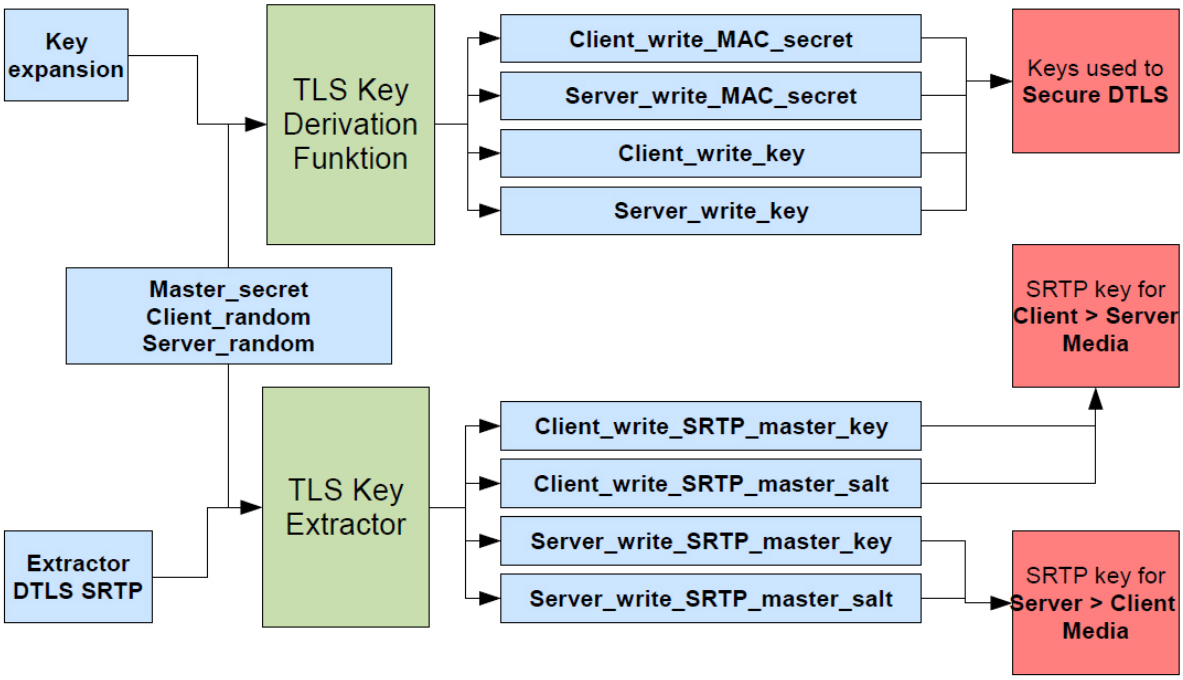

Abbildung 18: Schlüsselgenerierung DTLS-SRTP

Über die gesicherte Verbindung werden Master_secret, Client_random und Server_random übertragen.

Mit **Key expansion** und einer Schlüssel Funktion werden die Schlüssel Client_write_MAC_secret, Sever_write_MAC_secret, Client_write_Key und Sever_write_Key erzeugt, welche die DTLS Verbindung sichern.

Die Schlüssel Master_secret, Client_random und Server_random werden mit dem **Extractor DTLS SRTP** und dem TLS Key Extractor die Master Keys und die Master Salt für den Client (Client_write_SRTP_Master_Key und Client_write_SRTP_Master_Salt) und den Server (Server_write_SRTP_Master_Key und Server_write_SRTP_Master_Salt) erzeugt.

6.4 Zimmermann Real-time Transport Protocol

Das Zimmermann Real−time Transport Protocol (ZRTP) wurde von Phil Zimmermann für seine kostenlose VOIP-Anwendung Zfone im Jahr 2006 entwickelt. [Rüt07] Seine Idee war es SRTP um ein weiteres Protkoll zu erweitern, aber den Schlüsselaustausch nicht wie bei SDES oder MIKEY über eine zentrale Rechnerarchitektur zu realisieren sondern direkt zwischen den Teilnehmern in der ungesicherten Data Plane. Das Protokoll ist nicht als RFC von der IETF standardisiert, aber Patentiert. Solange der patentierte Standard eingehalten wird, steht ZRTP zu freien Verfügung. [Zim10]

Der kryptographische Schlüsselaustausch ist abermals das Diffie-Hellman-Verfahren, welches , wie in Kapitel 3 gezeigt wurde, anfällig gegen den Man-in-the-Middle Angriff ist. Damit der Man-in-the-Middle Angriff schwer zu realisieren ist, wurde ein aktive Zertifizierung eingeführt. Hierbei wird ein Short Authentication String (SAS) übermittelt. Bei ihm handelt es sich um einen vier Zeichen lange Buchstabenkombination, die sich die Kommunikationspartner über eine unverschlüsselte VOIP-Verbindung laut vorlesen. Gesichert wird das komplette Aushandeln des Schlüssels über Hash-Werte gesichert (Näheres dazu in Kapitel 6.4.1). Damit hören sie ihre Stimmen, was den Einbruch eines Angreifers schwer macht. Oft kennen sich die Teilnehmer schon und wenn nicht, dann würde der Angriff bemerkt werden wenn die Stimme, die den SAS mitgeteilt hat, und die Stimme, welche das normale Gespräch führt, unterschiedlich sind.

Für zusätzliche Sicherheit wird für jede Richtung jeweils ein Master Key und ein Master Salt erzeugt.

Des Weiteren kann ZRTP die Schlüssel auf drei verschiedene Arten erzeugen: mit dem Diffie-Hellman Mode, dem Preshared Mode und dem Multistream Mode

Diffie-Hellman Mode

Mittels des Diffie-Hellman Schlüsseltauschs (Siehe Kapitel 3) wird ein geheimer Schlüssel s0 generiert. Zusätzlich werden frühere Schlüssel in die Generierung das Schlüssel mit einbezogen. Die genaue Bestandteile des Schlüssels wird in Kapitel 6.4.2 erläutert. Wenn mehrere Media Streams zwischen den Teilnehmern aufgebaut werden soll, zum Beispiel für Videotelefonie (ein Stream für Video, ein Stream für Audio), so wird für jeden Stream ein eigener geheimer Schlüssel generiert.

Preshared Mode

Mit dem Preshared Mode kann ein geheimer Schlüssel ohne den Austausch weiterer Schlüssel, zum Beispiel mit Hilfe von Diffie-Hellman berechnet wer-

den. Hierfür wir ein Preshared Key aus alten Elementen berechnet. Damit Initator und Responder sicher gehen können, dass sie den gleichen Preshared Key verwenden, wird bei Schlüsselübertragung eine KeyID und ein Fingerprint übertragen. Die KeyID definiert die alten Elemente, während der Fingerprint ein hash-Wert des Preshared Key ist.

Falls es hier zu Abweichungen kommt, so muss ein neuer geheimer Schlüssel mit dem Diffie-Hellman Mode ausgetauscht werden.

Multistream Mode

Der Multistream Mode wird verwendet um mehrere Verbindung mit unterschiedlichen Streams zu realisieren. Jeder Stream nach dem ersten muss im Multistream Mode erfolgen. Voraussetzung hierfür ist das vorhanden sein eines aktiven Streams, welcher mit einem der vorher besprochenen Modis aufgebaut wurde.

Aus den Elementen der vorhergehenden Modi und der Hello- und Commit-Nachricht wird der für diesen Stream gültige geheime Schlüssel berechnet.

6.4.1 Schlüsselübertragung

Der Schlüsselaustausch erfolgt, im Gegensatz zu SDES und MIKEY, über die Data Plane, in der auch die entsprechenden SRTP-Pakete gesendet werden. Da diese Verbindung ungesichert ist, müssen gewisse Vorkehrungen getroffen werden. In Abbildung 19 ist der erste Schritt, die Bekanntmachung, zu sehen.

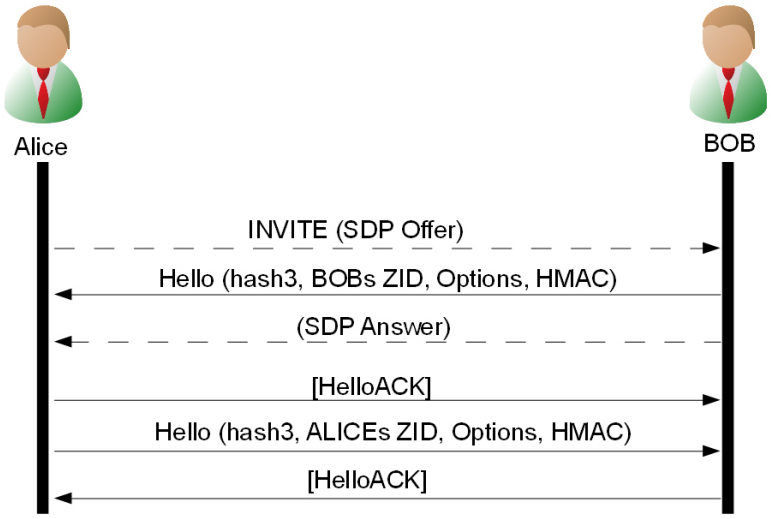

Abbildung 19: ZRTP Bekanntmachung

Zuerst schickt Alice über die gesicherte SIP-Architektur und damit über die Signaling Plane eine Einladung (Invite) an Bob. Er schickt eine **Hallo-Nachricht** über ZRTP und damit der Data Plane an Alice zurück.

Damit die Teilnehmer sicher sein können, dass alle Nachrichten von ein und der selben Person stammen, wurden eine Reihe von Hashwerten erzeugt. Hashwerte haben die Eigenschaft, dass sie (nahezu) eindeutig sind und schwer umkehrbar sind. Man kann sehr leicht von Wert A den Hashwert B errechnen, aber aus den Hashwert B nicht A. Diese Eigenschaft nutzt ZRTP aus. Hierfür wird ein 256-bit Zufallswert für jeden Teilnehmer erzeugt und daraus eine Reihe von Hashwerten generiert. Somit wird folgende Abhängigkeiten erzeugt:

$$\begin{aligned} \text{hash0} &= \text{256-bit Zufallswert} \\ \text{hash1} &= \text{hash}(\text{hash0}) \\ \text{hash2} &= \text{hash}(\text{hash1}) \\ \text{hash3} &= \text{hash}(\text{hash2}) \end{aligned}$$

Diese Hashwerte werden nun in umgekehrter Reihenfolge durch alle Nachrichten hindurch mit gesendet. ZRTP überprüft jeweils ob der Hashwert der Hashwert der neuen Nachricht ist. In der Hallo-Nachricht wird er letzte, also hash3, übertragen.

Die Hallo-Nachricht enthält überdies eine für jeden Teilnehmer eindeutige ZRTP ID (ZID). Sie ist eine eindeutige 96-bit große Zufallszahl.

Darüber hinaus werden einige Optionen ausgehandelt. Wie zum Beispiel welche Verschlüsselung verwendet werden soll, welcher Art von HMAC-Algorithmus verwendet wird oder welchen Format der SAS besitzen soll.

Gesichert wird die komplette Nachricht am Ende durch den Hash Message Authentication Code (HMAC). Das ist wiederum eine Hashfunktion, die auf der hmac-Spezifikation beruht. Der 64-bit große Hashwert, wird durch über die komplette Hallo-Nachricht gebildet, natürlich ausgenommen der HMAC. Der Empfänger einer Nachricht kann anhand das HMAC erkennen ob ein potenzieller Angreifer die Daten manipuliert hat.

Gleichzeitig mit der Hallo-Nachricht wird von BOB über die SIP-Architektur eine Antwort geschickt, das die Verbindung geklappt hat.

Nun kann Alice zuerst optional eine Bestätigung (**HelloACK**) senden, das sie die Hallo-Nachricht erhalten hat. Das kann notwendig werden, das ZRTP ein sehr schnell seine Hallo-Nachrichten wiederholt (20 Wiederholungen in 4 Sekunden).

Alice führt nun ihre Hallo-Nachricht aus, welche von Bob ebenfalls bestätigt werden kann.

Nachdem nun beide Hallo-Nachrichten ausgetauscht wurden, kann Alice oder Bob als Initator der weiteren ZRTP-Nachrichten sein. In Abbildung 20 ist der weitere Austausch zu sehen. Dabei gibt es je nach Modi kleine Unterschiede.

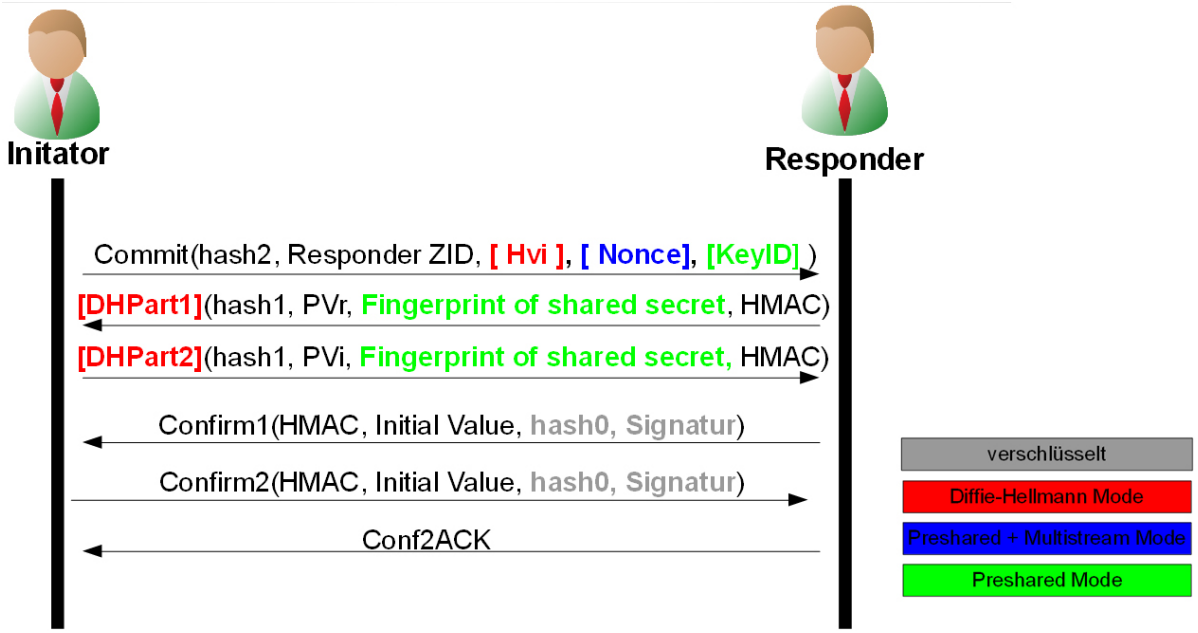

Abbildung 20: ZRTP Schlüsselaustausch

Der Initiator beginnt damit eine **Commit-Nachricht** an den Resonder zu senden. Sie enthält den zweiten 256-bit Hashwert hash2 und die ZID des Responders. Nun gibt es Unterschiede in der Nachricht, je nachdem welcher Modi verwendet wird.

Im Diffie-Hellman Mode wird zusätzlich ein Hash Hvi übertragen. Hvi wird dabei aus dem DHPart2, welcher erst bei der nächsten Übertragung verschickt wird, und aus der Hallo-Nachricht des Responders erzeugt.

Im Preshared Modus eine KeyID übertragen. Die KeyID ist eine ID um die gespeicherten Schlüssel zu verifizieren, mit denen den neuen Master Key und den Master Salt zu berechnen.

Im Multistream Modus und Preshared Modus wird zusätzlich ein Nonce übertragen. Das ist eine zufallsgenerierte Zahl und soll der Commit-Nachricht mehr Varianz verleihen. Der Grund hierfür liegt daran, das die Commit-Message zur späteren Berechnung des total_hash verwendet wird.

Die komplette Nachricht wird wieder von HMAC gesichert.

Der Resonder antwortet, wenn die Commit-Nachricht in Ordnung war, mit der **DHPart1-Nachricht**. Sie enthält im Diffie-Hellman Modus seinen Teilschlüssel für den Diffie-Hellman Austausch.

Im weiteren den ersten Hashwert hash1 und PVr. PV steht für Prime Value.

Wenn die Commit-Nachricht im Preshared Modus ist, so enthält sie auch noch 64-bit Hashwerte der vorher übertragenen Schlüssel. Damit sollen Missverständnisse von evtl. falschen Encryption Key und Salt Key verhindert werden.

Auch hier wird die Nachricht wieder über den HMAC vor Veränderungen geschützt.

Die **DHPart2-Nachricht** ist gleich aufgebaut wie die des Responders. Nur wird hierbei der Wert von Diffie-Hellman des Initiator übertragen, falls notwendig ist.

Da nun alle für die Übertragung notwendigen Elemente übertragen worden sind, wird mit der **Confirm1- und Confirm2-Nachricht** der Schlüsseltausch abgeschlossen. Bei den Confirm-Nachrichten steht dieses Mal der HMACzuerst im Paket.

Danach kommt ein zufälliger 128-bit Wert (Cipher Feedback Mode (CFB) Initial Value), welcher den SAS in dem verschlüsselten Teil dieser Nachricht, entschlüsseln kann.

Die restlichen Daten sind alle mit dem in SRTP verwendeten Schlüssel verschlüsselt. Damit wird geprüft, ob die Erstellung des Schlüssels geklappt hat.

In dem verschlüsselten Teil wird der letzte Hashwert h0 übertragen. Damit lässt sich nun der komplette Nachrichten Austausch nachvollziehen. Denn nur wenn der Hashwert h3 aus h0 ableiten lässt, kann man sicher sein, dass es die selbe Person ist.

Ebenso enthält es noch verschiedene Flags und eine Signatur. Die Signatur kann optional den SAS enthalten. Der SAS wird, wie in Kapitel 6.4 erläutert, über eine unverschlüsselte Verbindung vorgelesen, und, falls sie richtig ist, akzeptiert.

Wenn alles in Ordnung ist kann der Initiator seine Confirm2-Nachricht senden.

Und wenn der Responder diese Nachricht akzeptiert, so schickt er noch eine Akzeptiere-Nachricht **Conf2ACK** an den Initiator und somit ist der komplette Schlüsseltausch abgeschlossen.

6.4.2 Schlüsselgenerierung

Aus dem gemeinsamen Schlüssel s0 und s0n lassen sich alle benötigten Schlüssel ableiten. s0 wird dabei für die drei Modi jeweils unterschiedlich berechnet. Mit welchen Elementen s0 berechnet wird, wird im weiteren Verlauf erklärt. In der Abbildung 21 wird der Grundlegende Aufbau der Schlüsselgenerierung für den Diffie-Hellman Mode gezeigt. Der Blaue Kasten enthält die Elemente, die zur Generierung von s0 benötigt werden und ist daher

für jeden Modi unterschiedlich. Der restliche Aufbau ist identisch. 21

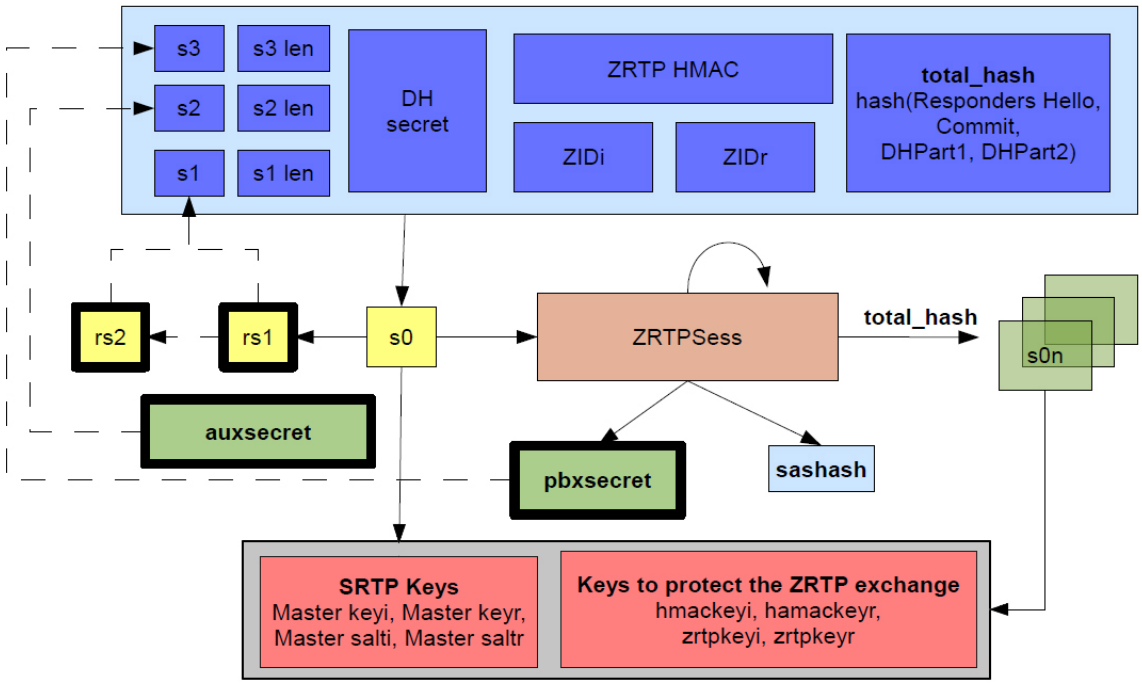

Abbildung 21: Schlüsselgenerierung ZRTP (Diffie-Hellman Mode)

Aus s0 und s0n kann der Master Key (srtpkeyi) und der der Master Salt (srtpsalti) auf der Seite des Initiator (i) und der Master Key (srtpkeyr) und der der Master Salt (srtpsaltr) auf der Seite des Responders (r) erstellt werden. Das wären jeweils die Schlüssel, die für die Berechnung des Encryption Key und dem Salt Key (Siehe Kapitel 5) benötigt werden.

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)

Für jeden Schlüssel wird ein KDF_Context benötigt. Diese wird aus der ZID des Initators und des Respondern und dem total_hash gebildet. Wie der total_hash gebildet wird, wird in den einzelnen Modi näher beschrieben, da er sich jeweils aus anderen Elementen erzeugt wird:

KDF_Context = (ZIDi || ZIDr || total_hash)

Zusätzlich werden aus s0 und s0n Schlüssel erzeugt, welche den ZRTP Austausch sichern. Dies wären der hmackeyi und zrtpkeyi für den Initator und der hmackeyr und zrtpkeyr für den Responder.

hmackeyi = KDF(s0, Ïnitiator HMAC key", KDF_Context,
negotiated hash length)

hmackeyr = KDF(s0, "Responder HMAC key", KDF_Context,
negotiated hash length)

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

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

Nachdem eine Session beendet ist wird auf den Schlüssel s0 eine Key Derivation Function (KDF) angewendet und in rs1 kopiert:

rs1 = KDF(s0, „retained secret", KDF_Context, 256)

Der Inhalt von rs1 wird aber vorher nach rs2 kopiert. Der Ursprüngliche Inhalt aus rs2 wird gelöscht. Der alte Schlüssel, der nun rs1 heißt wird im Preshared Mode benötigt. Aus rs1 und rs2 wird das Element s1 generiert. rs1 und rs2 habem bei der ersten Generierung des Schlüssel den Wert Null. Erst wenn einmal eine Verbindung zustande gekommen ist, kann der rs1 gesetzt werden.

s2 ist ein Hilfsschlüssel (*auxsecret*), der auf unterschiedliche Arten generiert werden kann. Es kann vorkonfiguriert sein, kann über einen alternativen Weg übertragen werden. Damit wird ZRTP noch sicherer. Falls kein *auxsecret* ausgehandelt wurde, so besitzt dieser den Wert Null.

s3 wird aus ZRTPSess erzeugt und wird abgekürzt *pbxsecret*. ZRTPSess wird mit einer KDF aus s0 erstellt:

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

Neben pbxsecret wird aus ZRTPSess noch der SAS für die aktive Zertifizierung erstellt.

sashash = KDF(s0, „SAS“, KDF_Context, 256)

sasvalue = sashash [truncated to leftmost 32 bits]

ZRTPSess dient auch zusätzlich dazu einen Schlüssel s0n zu erzeugen, der für den Multistream Mode verwendet wird.

Diffie-Hellman Mode

Die geheimen Schlüssel werden im Diffie Hellman Mode wie folgt erzeugt:

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

Der Counter ist immer auf Eins gesetzt. DHResult ist der geheime Schlüssel der durch den Diffie-Hellman Austausch erzeugt wurde (Siehe Kapitel 3). „ZRTP-HMAC-KDF“ ist ein String der besagt, für was s0 verwendet wird. Die ZIDi und ZIDr ist jeweils eine eindeutige ID des Initiatos (i) oder des Responders (r). Der total_hash wird im Diffie-Hellman Mode mit folgenden Elementen gebildet:

total_hash = hash(Hello of responder || Commit || DHPart1 || DHPart2)

Und zum Schluss fließen die Werte s1, s2 und s3 und deren Länge in die Erzeugung der Schlüssel mit ein.

Preshared Mode

Im Preshared Mode wird ein Preshared Key mit einer Hash Funktion aus den schon vorher übertragenen Elementen erzeugt:

preshared_key = hash(len(rs1) || rs1 || len(s2) || s2 || len(s3) || s3)

So wird der alte Schlüssel s0 in Form von rs1 wiederverwendet. Ebenso das vorher definierte (*auxsecret*) s2 und, falls ausgehandelt, der *pbxsecret* s3.

Um s0 noch mehr Varianz zu geben wird ein KDF_Context aus dem total_hash und den ZIDs der Teilnehmer erzeugt. Der total_hash wird, im Gegensatz zu Diffie-Hellman Mode, ohne die beiden Ausgetauschten Diffie-Hellman Secrets erzeugt:

total_hash = hash(Hello of responder || Commit)

Damit wären nun alle Elemente vorhanden, um s0 zu erzeugen:

s0 = KDF(preshared_key, „ZRTP PSK“, KDF_Context, negotiated hash length)

Multistream Mode

Da der Multistream Mode nur verwendet wird, wenn eine bestehende Verbindung existiert. Im Multistream Mode wird der Schlüssel s0n mit einer verstehenden ZRTP Session (ZRTPSess) generiert:

s0n = KDF(ZRTPSess, „ZRTP MSK“, KDF_Context, negotiated hash length)

KDF_Context wird genau so berechnet, wie er schon für den Preshared_key berechnet wurde. Kurz gefasst ist im Multistream Mode die ZRTPSess der Preshared_key. Die Varianz bei mehr als zwei Streams kommt dadurch, dass für jeden weiteren Stream ein komplett neuer Schlüsselaustausch vollzogen wird. Und dadurch ändert sich immer der KDF_Context der aus Elementen des Schlüsselaustauschs erzeugt wird.

7 Zusammenfassung

Secure Real−time Transport Protocol ist das wichtigste Protokoll wenn Media Daten in Echtzeit übertragen werden sollen. Voraussetzung hierfür ist eine vorhandene, gesicherte SIP-Architektur, welche eine Authentifizierung der Teilnehmer vornimmt.

Die eigentlichen Nutzdaten werden über eine ungesicherte UDP-Verbindung gesendet. Daher ist es nötig, dass diese Datenpakete mit einen geheimen Schlüssel einzeln verschlüsselt werden. Dieser geheime Schlüssel darf nur dem Sender und Empfänger bekannt sein. In Secure Real−time Transport Protocol können insgesamt vier verschiedene Key Management Protokolle verwendet werden um diese geheimen Schlüssel zu übertragen. Dies wären SDP Security Description for Media Streams, Multimedia Internet KEYing, Zimmermann Real−time Transport Protocol und Datagram Transport Layer Security-Secure Real−time Transport Protocol.

SDES überträgt den geheimen Schlüssel im Klartext über die SIP-Architektur. Die jeweiligen Betreiber der SIP-Server können daher den Schlüssel abfangen und auswerten und damit den Medien-Stream entschlüsseln. Wenn der Betreiber des SIP-Servers unbekannt ist, so ist diese Übertragung der Schlüssel mit SDES nicht zu empfehlen.

Auch MIKEY überträgt den Schlüssel über die SIP-Architektur, allerdings verschlüsselt. Diese Verschlüsselung kann bei MIKEY über fünf verschiedenen Methoden geschehen. Das wäre eine Methoden durch den vorherigen Austausch eines Pre-shared Key, über zwei Verfahren der RSA-Verschlüsselung (RSA & RSA-Reverse) und über zwei Arten eines Diffie-Hellman Schlüsselaustausch (DHSIGN & DHMAC). Alle diese Verfahren sind als sicher eingestuft und können über unsichere SIP-Server verwendet werden. MIKEY ist daher SDES vorzuziehen, wenn die Schlüsseltausch über die SIP-Architektur abgewickelt werden soll.

Datagram Transport Layer Security-Secure Real−time Transport Protocol hingegen wird über die ungesicherte UDP-Verbindung übertragen. Die Sicherung wird dabei innerhalb der OSI-Schicht 6 realisiert. Es bietet ebenfalls eine gute Sicherheit.

Das neuste Key Management Protokoll ZRTP wird wie Datagram Transport Layer Security-Secure Real−time Transport Protocol über die ungesicherte UDP-Verbindung übertragen. Das patentierte Protokoll wird hauptsächlich für VOIP-Anwendungen verwendet. Es bietet viele Möglichkeiten den geheimen Schlüssel zu generieren. Ob über einen Pre-shared Key oder über Diffie-Hellman Schlüsselaustausch mit verschiedenen

zusätzlichen Parametern sind viele Variationen möglich. Besonders Hervorzuheben ist bei ZRTP die aktive Authentifizierung, bei der sich die Teilnehmer einen generierten String laut vorlesen.

Da ZRTP den Schlüssel auf verschiedene Arten erzeugen kann uns eine aktive Authentifizierung besitzt ist es viel Universeller als DTLS-SRTP und ist diesem daher vorzuziehen.

Durch diese verschiedenen Schlüsselprotokolle und dem ungebrochenen Verschlüsselungsalgorithmus Advanced Encryption Standard (AES) ist SRTP heute und auf lange Sicht eine sehr sichere Übertragungsmethode für Echtzeit-Anwendungen, deren Schwachstelle im allgemeinen der Schlüsseltausch ist.

Anhang

A Überblick Crypto-Suites

In SRTP werden drei sogenannte Crypto-Suits mit unterschiedlichen Eigenschaften vordefiniert. In diesem Teil des Anhangs werden diese Eigenschaften übersichtlich dargestellt [And06]:

	AES_CM_128_ HMAC_SHA1_80	AES_CM_128_ HMAC_SHA1_32	F8_128_ HMAC_SHA1_80
Master key length	128 bits	128 bits	128 bits
Master salt length	112 bits	112 bits	112 bits
SRTP lifetime	2^{48} packets	2^{48} packets	2^{48} packets
SRTCP lifetime	2^{31} packets	2^{31} packets	2^{31} packets
Cipher	AES Counter Mode	AES Counter Mode	AES F8 Mode
Encryption key	128 bits	128 bits	128 bits
MAC	HMAC-SHA1	HMAC-SHA1	HMAC-SHA1
SRTP auth. tag	80 bits	32 bits	80 bits
SRTCP auth. tag	80 bits	80 bits	80 bits
SRTP auth. key len.	160 bits	160 bits	160 bits
SRTCP auth. key len.	160 bits	160 bits	160 bits

Tabelle 1: Überblick Crypto-Suits

B Überblick der Request for Comments

In diesem Teil des Anhangs werden alle Request for Comments (RFC) der IETF übersichtlich dargestellt:

- RTP / RTCP - **RFC 3550** - RTP: A Transport Protocol for Real-Time Applications
- SRTP / SRTCP - **RFC 3711** - The Secure Real-time Transport Protocol (SRTP)
- SDES - **RFC 4568** - Session Description Protocol (SDP) Security Descriptions for Media Streams
- MIKEY - **RFC 3830** - MIKEY: Multimedia Internet KEYing
- Erweiterung von MIKEY
  - MIKEY-DHMAC - **RFC 4650** - HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY)
  - MIKEY-RSA-R - **RFC 4738** - MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)
- ZRTP - noch nicht offiziell in der RFC veröffentlicht
- DTLS - **RFC 4347** - Datagram Transport Layer Security

Außer ZRTP sind alle Protokolle standardisiert. Eine erste Vorschlag um ZRTP als Standard einzuführen wurde im Januar 2010 eingereicht. [Zim10]

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
CFB	Cipher Feedback Mode
CHASH	Cert hash payload
CS	Crypto Session
CSIDMAP	Crypto Session ID Map
CSBID	Crypto Session Bundle ID
DDoS	Distributed Denial of Service
DoS	Denial of Service
DTLS-SRTP	Datagram Transport Layer Security-Secure Real−time Transport Protocol
FEC	Forward Error Correction
GKMARCH	Group key management architecture
HDR	Header
HMAC	Hash Message Authentication Code
IETF	Internet Engineering Task Force
IP	Internet Protocol
KDR	Key Derivation Rate
KDF	Key Derivation Function
KEMAC	Key Data Transport Payload
MAC	Message Authentication Code
MIKEY	Multimedia Internet KEYing
MKI	Master Key Identifier
NTP	Network Time Protocol
PK	Public-Key
PKE	Public-Key Envelope
PSK	Pre-shared Key
QoS	Quality of Service
RFC	Request for Comments
ROC	Rollout Counter
RTCP	Real−time Transport Control Protocol
RTP	Real−time Transport Protocol
SAS	Short Authentication String
SDES	SDP Security Description for Media Streams
SDP	Session Description Protocol
SIP	Session Initiation Protocol

SMIME	Secure / Multipurpose Internet Mail Extensions
SP	Security Policy
SPIT	Spam over Internet Telefonie
SRTCP	Secure Real−time Transport Control Protocol
SRTP	Secure Real−time Transport Protocol
SSRC	Synchronization Source Identifier
TCP	Transmission Control Protocol
TEK	Traffic-Encrypting Key
TGK	TEK Generation Key
TLS	Transport Layer Security
UA	SIP User Agents
UDP	User Datagram Protocol
VOIP	Voice over Internet Protocol
Vishing	Phishing over VoIP
WSH	Window Size Hint
ZID	ZRTP ID
ZRTP	Zimmermann Real−time Transport Protocol

Abbildungsverzeichnis

Literatur

[And06] Andreasen, F.: Session Description Protocol (SDP) Security Descriptions for Media Streams, 2006. Online verfügbar unter http://www.networksorcery.com/enp/rfc/rfc4568.txt, zuletzt besucht am 13.12.2009.

[Ark04] Arkko, J.: MIKEY: Multimedia Internet KEYing, 2004. Online verfügbar unter http://www.networksorcery.com/enp/rfc/rfc3830.txt, zuletzt besucht am 13.12.2009.

[Bad04] Badach, Anatol: Voice over IP - Die Technik. Hanser Verlag, 2004. ISBN 978 3 446 22627 4.

[Bau04] Baugher, Mark: The Secure Real-time Transport Protocol (SRTP), 2004. Online verfügbar unter http://www.networksorcery.com/enp/rfc/rfc3711.txt, zuletzt besucht am 09.12.2009.

[Ber09] Berezowski, Björn; Brandt, Holger: Bedrohungsanalyse im IMS Umfeld, 2009. Online verfügbar unter http://www.fbi.h-da.de/fileadmin/personal/m.massoth/Master-Seminar_WS0910/IMS_Bedrohungsanalyse_MPSE_SS09.pdf, zuletzt besucht am 06.01.2010.

[Euc06] Euchner, M.: HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY), 2006. Online verfügbar unter http://www.networksorcery.com/enp/rfc/rfc4650.txt, zuletzt besucht am 16.02.2010.

[Her08] Herff, Sebastian von; Keinz, Markus; Miess, Florian: Sicherheit in IP Multimedia Systemen, 2008. Online verfügbar unter http://www.fbi.h-da.de/fileadmin/personal/m.massoth/Master-Seminar_WS0910/SIP_Security_Hausarbeit_2008-04-21.pdf, zuletzt besucht am 06.01.2010.

[Ign06] Ignjatic, D.: MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY), 2006. Online verfügbar unter http://www.networksorcery.com/enp/rfc/rfc4738.txt, zuletzt besucht am 28.12.2009.

[Mal02] Malladi, Sreekanth; Alves-Foss, Jim; Heckendorn, Robert B.: On Preventing Replay Attacks on Security Protocols, 2002. Online verfügbar unter http://www.csds.uidaho.edu/papers/Malladi02b.pdf, zuletzt besucht am 06.01.2010.

[Mey09] Meyer, Prof.Dr. Herwig: Skript für Kryptographie, 2009. Skript 2009 der Mastervorlesung Kryptographie.

[Par06] Parziale, Lydia; Britt, David T.; Davis, Chuck; Forrester, Jason : TCP/IP Tutorial and Technical Overview. IBM Redbooks, 2006.

[Res06] Rescorla, E.: Datagram Transport Layer Security, 2006. Online verfügbar unter `http://www.networksorcery.com/enp/rfc/rfc4347.txt`, zuletzt besucht am 04.03.2010.

[Rey08] Rey, Enno: Voice-over-IP Security, 2008. Online verfügbar unter `http://ernw.de/content/e7/e181/e1212/download1216/1B05_ernw_voipsec_ger.pdf`, zuletzt besucht am 18.11.2009.

[Rüt07] Rütten, Christiane: Patent verschlüsselt, 2007. Online verfügbar unter `http://www.heise.de/ct/artikel/Patent-verschluesselt-290822.html`, zuletzt besucht am 18.02.2010.

[Rue05] Ruetten, Gereon; Stutzke, Oliver: Angriff von innen, 2005. Online verfügbar unter `http://www.heise.de/security/artikel/Angriff-von-innen-270632.html`, zuletzt besucht am 18.11.2009.

[Sch96] Schulzrinne, Henning: RTP: A Transport Protocol for Real-Time Applications, 1996. Online verfügbar unter `http://www.networksorcery.com/enp/rfc/rfc1889.txt`, zuletzt besucht am 09.12.2009.

[Sch03] Schulzrinne, Henning: RTP: A Transport Protocol for Real-Time Applications, 2003. Online verfügbar unter `http://www.networksorcery.com/enp/rfc/rfc3550.txt`, zuletzt besucht am 09.12.2009.

[Sis09] Sisalem, Dorgham; Floroiu, John; Kutham, Jiri; Abend, Ulrich; Schulzrinne, Henning: SIP Security. John Wiley and Sons Lzd., 2009. ISBN 978 0 470 51636 2.

[Tit08] Tittelbach, Bernhard: Angriff auf Estland, 2008. Präsentation, Online verfügbar unter `http://www.iaik.tugraz.at/downloads/public/KIIS/tittelbach_angriff_auf_estland.pdf`, zuletzt besucht am 11.02.2010.

[van07] van Eck, Wim: Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?, 2007. Online verfügbar unter `http://jya.com/emr.pdf`, zuletzt besucht am 11.02.2010.

[Whi09] Whitney, Lance; Beiersmann, Stefan: Weltweites Spam-Aufkommen übersteigt 90-Prozent-Marke, 2009. Online verfügbar unter http://www.zdnet.de/news/wirtschaft_sicherheit_security_weltweites_spam_aufkommen_uebersteigt_90_prozent_marke_story-39001024-41004583-1.htm, zuletzt besucht am 06.01.2010.

[Zim10] Zimmermann, Phil: ZRTP: Media Path Key Agreement for Secure RTP draft-zimmermann-avt-zrtp-17, 2010. Online verfügbar unter http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-17, zuletzt besucht am 16.02.2010.