Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Fabian Reiber

Darstellung bestehender E-Mail-Verschlüsselungsmethoden sowie Konzeption und Implementierung eines OpenPGP-Mailing-Verteiler-Systems

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Fabian Reiber

Darstellung bestehender E-Mail-Verschlüsselungsmethoden sowie Konzeption und Implementierung eines OpenPGP-Mailing-Verteiler-Systems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr.-Ing. Martin Hübner

Eingereicht am: 29. Juli 2016

Fabian Reiber

Thema der Arbeit

Darstellung bestehender E-Mail-Verschlüsselungsmethoden sowie Konzeption und Implementierung eines OpenPGP-Mailing-Verteiler-Systems

Stichworte

OpenPGP, Ende-zu-Ende-Verschlüsselung, E-Mail Verteiler, Massenverschlüsselung, Umverschlüsselung, S/MIME, Kryptographie, De-Mail

Kurzzusammenfassung

Diese Bachelorarbeit handelt zum einen von einigen theoretischen Netzwerk- und Kryptographie-Grundlagen und zum anderen von der Konzeption und Realisierung eines OpenPGP-Mailing-Verteiler-Systems. Zu Beginn wird kurz das OSI-Modell und einige für das Verständnis der Arbeit notwendige Schichten vorgestellt. Anschließend wird ein kleiner Überblick über entwickelte kryptographische Algorithmen und deren Kategorisierung gegeben. Auf Grundlage dieses Wissens wird jeweils ein Protokoll der Transport- und Netzwerkschicht des OSI-Modells genauer betrachtet und eingeschätzt. Nach diesen Netzwerk Grundlagen werden drei mögliche Werkzeuge für eine Ende-zu-Ende-Verschlüsselung, sowie eine Lösung zur Transportverschlüsselung von E-Mails genauer erläutert. Anschließend werden bestehende Lösungen für ein E-Mail Verteiler-System vorgestellt. Zur Realisierung eines eigenen auf OpenPGP basierende Systems wird dann ein Konzept entwickelt und hierzu die nötigen Anforderungen ausgearbeitet. Hierbei liegt ein Schwerpunkt auch in der Erläuterung von notwendigen Komponenten und Werkzeugen. Zuletzt wird das eigene System aus kryptographischer Sicht bewertet und mit den anderen zuvor vorgestellten Systemen verglichen.

Fabian Reiber

Title of the paper

Representation of existing e-mail-encryption methods as well as concept and implementation of an OpenPGP-mailing-distribution-system

Keywords

OpenPGP, End-To-End-Encryption, E-Mail Distributer, Bulk-Encryption, Re-Encryption, S/MIME, Cryptography, De-Mail

Abstract

This thesis deals on the one hand with some theoretical networking and cryptography basics and secondly on the conception and implementation of an OpenPGP-mailing-distribution-system. At the beginning the OSI model and some layers of it, which are necessary for the understanding of the thesis, are presented. Then, a brief overview of advanced cryptographic algorithms and their categorization is given. Based on this knowledge, respectively one protocol of the transport and network layer of the OSI model will be considered and evaluated in more detail. After these network basics three possible tools for end-to-end encryption, as well as a solution for transport encryption of e-mails are explained in detail. Then existing solutions for an e-mail distribution system are presented. For an own implementation based on OpenPGP the concept is being developed and the necessary requirements are being carved out. At this juncture the focus will be tended on the explanation of the necessary components and tools. Recently the own system is evaluated from cryptographic point of view and compared to the other presented systems before.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Laut einer Umfrage des Statistischen Bundesamtes aus dem Jahr 2014 nutzten 91% der Deutschen im privaten Bereich das Internet, um E-Mails zu senden und zu empfangen (vgl [20115]). Auch im geschäftlichen Bereich wächst die Nutzung von E-Mails als zentrales Kommunikationsmittel. 2015 wurden von einer einzelnen Geschäftsperson im Durchschnitt täglich 122 E-Mails versendet und empfangen (vgl. [TRG15]). Diese Zahlen führen uns vor Augen, welche Bedeutung diese Art der Kommunikation heutzutage zukommt.

Die Enthüllungen von Edward Snowden im Sommer 2013 und die anschließende Berichterstattung (vgl. [GMP13]) rückt das Thema Sicherheit in der digitalen Welt immer weiter in den Mittelpunkt der Gesellschaft. Fast wöchentlich erscheinen neue Berichte über das Abfangen von persönlichen Daten - E-Mails, Telefonate, Kurznachrichten usw. - durch staatliche Geheimdienste wie den deutschen Bundesnachrichtendienst (BND) oder die US-amerikanischen National Security Agency (NSA).

Aktuellstes und bestes Beispiel war das von Juli 2000 bis Oktober 2015 geltende sogenannte *Safe Harbor* Abkommen. Europäische Datenschutzrichtlinien sehen ein Verbot einer Weitergabe personenbezogener Daten[1] von EU-Bürgern an Staaten ohne ein mit dem EU-Recht vergleichbares Schutzniveau vor. Durch das *Safe Harbor* Abkommen wurden milliardenschwere US-Unternehmen wie Google oder Amazon (vgl. [Exp16]), ermächtigt personenbezogene Daten legal aus der EU in die USA zu übermitteln. Dafür mussten die Unternehmen dem Abkommen beitreten und bestimmte Prinzipien erfüllen, um ein adäquates Sicherheitsniveau zu gewährleisten. Allerdings wurde das Abkommen vom Europäischen Gerichtshof im Oktober 2015 für ungültig erklärt. In der Begründung hieß es, dass die USA kein angemessenes Schutzniveau der Daten gewährleiste und dadurch „das Grundrecht auf Achtung der Privatsphäre“[2] verletzte. Ebenso wies der EuGH darauf hin, dass das garantierte Grundrecht „auf

[1] Bundesdatenschutzgesetz (BDSG) §3 Abs. (1): „Personenbezogene Daten sind Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren natürlichen Person (Betroffener)“

[2] Urteil des Europäischen Gerichtshofs; Rechtssache C-362/14; Punkt 78; 06.10.2015; http://curia.europa.eu/juris/document/document.jsf?text=&docid=169195&pageIndex=0&doclang=DE&mode=req&dir=&occ=first&part=1&cid=196172; zuletzt besucht am 27.07.2016

Achtung des Privatlebens“[3] verletzt sei. Ein Zugriff staatlicher Behörden wie der NSA sei durch das Abkommen nicht ausgeschlossen und eine Möglichkeit des effizienten Rechtsschutzes bestehe für die Bürger nicht. Im Februar 2016 wurde das Folgeabkommen *EU-US Privacy Shield* entworfen, welches Mitte Juli 2016 in Kraft trat. Unternehmen können ab August 2016 dem neuen Abkommen beitreten und Daten zwischen der EU und den USA austauschen. Entsprechende Kritik folgte sofort. Das *EU-US Privacy Shield* Abkommen sei „Alter Wein in neuen Schläuchen“[4]. [fdDudI, fdDudI16, Bec16, Kom16]

In Zeiten zunehmender Überwachung und wachsender Datensammelwut seitens staatlicher Geheimdienste und privater Konzerne, ist es Aufgabe der Informatik sichere und zugleich benutzerfreundliche Kommunikationsmittel zu entwickeln. Ziel hierbei sollte es sein, die persönlichen und geschäftliche Daten vor unbefugten Zugriffen zu schützen sowie die Privatsphäre aller zu wahren.

1.2 Zielsetzung

Das Hauptziel dieser Arbeit besteht in dem Entwurf und Entwicklung eines sicheren und möglichst benutzerfreundlichen E-Mail-Verteiler-Systems. Anwender sollen Ende-zu-Ende verschlüsselte Nachrichten von einer ausgewählten Anwendergruppe empfangen und an sie versenden können. Hierdurch sollen die Inhalte der E-Mails vor unbefugter Einsichtnahme oder Manipulation geschützt werden.

Um den Ablauf einer Kommunikation im Netzwerk nachvollziehen zu können, soll zunächst ein Einblick in das sogenannte OSI-Modell gegeben werden. Darüber hinaus sollen im Anschluss bekannte kryptographische Algorithmen vorgestellt werden. Dieses Wissen dient als Grundlage für das Verständnis von bereits existierenden Verschlüsselungsprotokollen wie TLS oder IPsec, oder im Speziellen von vorhandenen E-Mail-Verschlüsselungsprotokollen wie bspw. S/MIME oder OpenPGP. Abbildung 1.1 zeigt einen Einblick in das OSI-Modell und der Ansiedlung einiger Protokolle in den Schichten des Modells.

Ein grundlegendes Problem der unverschlüsselten E-Mail-Kommunikation im Internet ist, dass sie theoretisch für jeden Menschen - mit dem nötigen Wissen und den Werkzeugen - ohne großen Aufwand einsehbar oder manipulierbar ist. Daher ist der Einsatz von Verschlüsselungswerkzeugen, insbesondere für eine Ende-zu-Ende-Verschlüsselung, ratsam, wenn bspw.

[3] Urteil des Europäischen Gerichtshofs; Rechtssache C-362/14; Punkt 94; 06.10.2015; http://curia.europa.eu/juris/document/document.jsf?text=&docid=169195&pageIndex=0&doclang=DE&mode=req&dir=&occ=first&part=1&cid=196172; zuletzt besucht am 27.07.2016

[4] Volker Tripp; Safe Harbor: Alter Wein in neuen Schläuchen; Digitale Gesellschaft e. V.; 02.02.2016; https://digitalegesellschaft.de/2016/02/safe-harbor-alter-wein/; zuletzt besucht am 27.07.2016

Schichtnummer	Schichtbezeichnung	Sicherheits-mechanismus
7	Anwendungsschicht	PGP/SMIME
4	Transportschicht	TLS
3	Netzwerkschicht	IPSEC

Abbildung 1.1: Auszug aus dem OSI-Modell

sensible Daten mit einer E-Mail versendet werden. Einige bereits vorhandene Lösungen zur E-Mail-Verschlüsselung sollen genauer diskutiert werden.

Ein Problem all dieser Lösungen ist, dass ihre Grundfunktionalität nicht für die Massenkommunikation ausgelegt ist. D.h. das Senden einer E-Mail von einem Sender an beliebig viele Empfänger ist mit einem höheren Aufwand verbunden und deutlich weniger benutzerfreundlich als das Senden einer E-Mail von einem Sender an einen Empfänger. Unterschiedliche Lösungsstrategien für dieses Problem sollen erläutert werden. Auf dieser Grundlage soll schließlich ein Entwurf für ein OpenPGP-Mailing-Verteiler-System vorgestellt sowie dessen Implementierung detailliert beschrieben werden. Dieses System soll die sichere, Ende-zu-Ende verschlüsselte Kommunikation der Nutzer eines E-Mail-Verteilers ermöglichen. Die Grundidee hierbei soll sein, dass ein dezentraler Verteiler-Server sämtliche öffentliche Schlüssel aller auf einem E-Mail-Verteiler vorhandenen Nutzer besitzt. Gleichzeitig sollen alle Nutzer den öffentlichen Schlüssel des Verteilers besitzen. Hierdurch soll ein Sender eine E-Mail, die über den Verteiler verschickt werden soll, verschlüsseln und an den Verteiler-Server schicken können. Erhält der Verteiler-Server eine E-Mail, soll diese mit dem privaten Schlüssel des Verteilers entschlüsselt werden. Im nächsten Schritt soll diese E-Mail dann mit allen öffentlichen Schlüsseln der Empfänger auf diesem Verteiler verschlüsselt und an diese versandt werden.

Anhand des Entwurfes soll im Anschluss ein Server aufgesetzt werden, der dies praktisch umsetzt. Die dafür entwickelte Software soll quelloffen und im Internet abrufbar[5] sein. Somit sollen etwaige Fehler oder Sicherheitslücken von anderen Entwicklern entdeckt und korrigiert werden können. Ebenfalls sollen die Vor- und Nachteile sowie mögliche vorhandene Schwachstellen dieses Verteiler-Systems beschrieben werden. Zudem sollen die in dieser Arbeit vorgestellten Lösungen mit der eigenen in ihren Gemeinsamkeiten und Unterschieden verglichen werden.

[5] https://github.com/fabianHAW/GnuPG-Distributer-Mailing-System; zuletzt besucht am 27.07.2016

Insgesamt soll mit dieser Arbeit gezeigt werden, wie mit den grundlegenden kryptographischen Algorithmen und Protokollen Werkzeuge entwickelt werden können, die eine sichere und benutzerfreundliche E-Mail-Massenkommunikation gewährleisten können. Des Weiteren soll ersichtlich werden, wo die bisherigen Lösungen und die eigene Lösung kryptographische und praktische Schwachstellen aufweisen. Mit diesem und dem entstehenden Wissen soll eine Grundlage geschaffen werden, um in Zukunft weitere Lösungen erforschen und entwickeln zu können.

1.3 Gliederung der Arbeit

Nach der Einleitung in Kapitel 1 geht es in Kapitel 2 um einige grundlegende Netzwerk- und Kryptographie-Grundlagen. Dort wird zunächst ein Einblick in bestimmte für diese Arbeit wichtige Schichten des OSI-Modells gegeben und ihre Aufgaben beschrieben. Im Anschluss werden einige allgemeine kryptographische Algorithmen genauer erläutert. Mit diesem Wissen wird auf zwei etablierte Protokolle zur Verschlüsselung von Daten in der Transport- und Netzwerkschicht eingegangen: TLS bzw. IPsec.

Im darauf folgenden Kapitel 3 werden die Konzepte sowie die Vor- und Nachteile einiger vorhandener Verschlüsselungs-Lösungen aufgezählt. OpenPGP und S/MIME sind für die E-Mail-Verschlüsselung von besonderer Bedeutung, da mit ihnen eine Ende-zu-Ende-Verschlüsselung realisiert wird, die es Angreifern nahezu unmöglich macht den Inhalt von E-Mails zu entschlüsseln. Mit De-Mail wird ein vom Bundesministerium initiiertes Konzept zur sicheren und verbindlichen E-Mail-Kommunikation innerhalb Deutschlands vorgestellt. Als vierte vorhandene Lösung bietet SMTP/TLS eine Erweiterung zum SMTP für einen sicheren Transport von E-Mails zwischen Client und Server.

In Kapitel 4 werden zunächst bereits bestehende Lösungen für eine sichere E-Mail-Massenkommunikation kurz aufgezeigt. Im Anschluss wird eine eigene Lösung dieses Problems in einem Entwurf beschrieben und ein theoretischer Aufbau des Systems aufgezeigt.

Die Umsetzung dieses Entwurfs wird in Kapitel 5 erläutert. Die Implementierung wird mit OpenPGP als Ende-zu-Ende-Verschlüsselungsmethode umgesetzt. Des Weiteren werden mögliche Schwachstellen des eigenen Systems aufgezeigt sowie im letzten Schritt die eigene Lösung mit den bereits vorgestellten Lösungen aus Kapitel 4.1 verglichen.

In Kapitel 6 wird diese Arbeit mit einem Fazit sowie einem Ausblick abgeschlossen.

2 Netzwerk Grundlagen

2.1 Das OSI-Modell

Das OSI-Modell ist ein Standard zur Kommunikation zwischen zwei Endsystemen in uneinheitlichen Netzen. Es wurde 1977 von der International Standardization Organization (ISO) entworfen. Das Modell ist aus zwei Sichten zu betrachten: der vertikalen und horizontalen Sicht. In der vertikalen Sicht gliedert sich das Modell in 7 Schichten ein (vgl. vertikale Pfeile in Abb. 2.1) die jeweils einen speziellen Dienst besitzen. Der Begriff Dienst wird im weiteren genauer spezifiziert. Die horizontale Sicht zeigt die logische Kommunikation (vgl. horizontale Pfeile in Abb. 2.1). Dies bedeutet, dass die gleichnamigen Schichten auf Sender- und Empfänger-Seite miteinander logisch in Verbindung stehen. [S.27f HPS00]

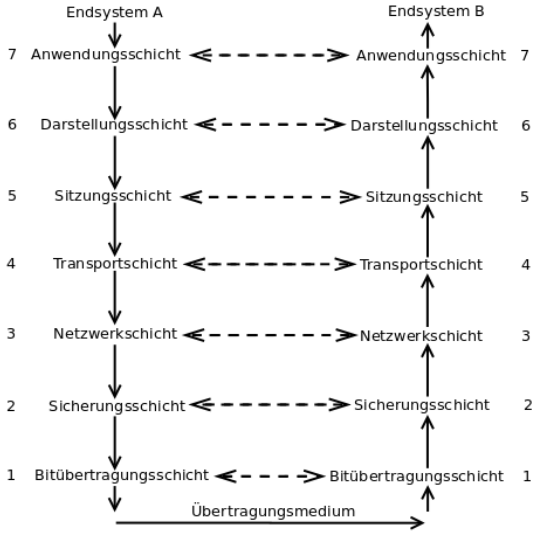

Abbildung 2.1: Vertikale und horizontale Sicht auf das OSI-Modell

Logische Kommunikation

Die logische Kommunikation bedeutet hierbei, dass es für die Anwendung auf Senderseite den Anschein macht sie würde direkt mit der kommunizierenden Anwendung auf Empfängerseite auf ein und demselben Client laufen. Die Lokalitäten der Clients, auf denen die Anwendungen laufen, können allerdings weit entfernt voneinander liegen. Durch die Verwendung dieser Kommunikation, sind die Details der physischen Infrastruktur sowie Details darunterliegender Schichten oft für die Schichten irrelevant. [S.211 KR14]

Schichten (vertikale Sicht)

Die Daten passieren vertikal auf jeder Seite - einmal von oben nach unten bzw. von unten nach oben - alle sieben Schichten des OSI-Modells, um von einem Endgerät zu einem anderen Endgerät zu gelangen. Wie bereits erwähnt, besitzt jede Schicht einen Dienst. Ein Dienst erfüllt eine spezielle Leistung. Dies kann z.B. der Datentransport zwischen zwei Endgeräten sein. Die Dienste werden durch verschiedene Protokolle erbracht. Im OSI-Modell gilt daher ein sogenanntes Dienstmodell [vgl. S.70 KR14]. Dies besagt, dass eine Schicht der übergeordneten Schicht für sie bestimmte Dienste anbietet und selber die Dienste der untergeordneten Schicht nutzt. In den Abschnitten 2.1.1, 2.1.2 und 2.1.3 werden die Dienste von drei Schichten etwas genauer erläutert. Das OSI-Modell gibt bzgl. der Schichten keine Implementierungsvorgaben vor, sondern beschreibt eher eine allgemeine Definition für jede Schicht. Somit ergibt sich, dass die Schichten unabhängig voneinander zu betrachten sind. Für eine jeweilige Schicht ist dadurch die Funktionsweise anderer Schichten unbekannt. [S.29 RLM14]

Ferner werden unterschiedliche Begriffe für die Daten in den Schichten verwendet. In den anwendungsorientierten Schichten 7 (Anwendungsschicht), 6 (Darstellungsschicht) und 5 (Sitzungsschicht) ist von Daten die Rede. In der Transportschicht (4) ist der Begriff Segment gebräuchlich. Von Paketen ist in der Netzwerkschicht (3) die Rede. Des Weiteren wird in der Sicherungsschicht (2) der Begriff Rahmen (Frame) verwendet. Auf der untersten 1. Schicht, der Bitübertragungsschicht, wird die Repräsentation der Daten als Bits bezeichnet. [S.74ff Sch06]

Protokolle (horizontale Sicht)

Um sich verständigen zu können, werden, ähnlich wie bei einer Unterhaltung zweier Personen, Regeln, Vereinbarungen und eine Sprache benötigt. Beispielsweise muss Person A eine Person B explizit nach der aktuellen Uhrzeit fragen (*Wie spät ist es?*), bevor sie diese erfahren kann. Für die Kommunikation zweier Endgeräte in einem Netzwerk werden diese Dinge ebenfalls benötigt. Daher gibt es Protokolle die die Regeln, Vereinbarungen und Sprache spezifizieren. Jede

Schicht enthält mehrere Protokolle. Mit ihnen ist gewährleistet, dass gleichnamige Schichten in dem OSI-Modell miteinander einheitlich, reibungslos und fehlerfrei kommunizieren bzw. Daten austauschen können. [S.30f RLM14, ITWa]

Ende der 60er Jahre wurde das für heute in der Internetkommunikation etablierte TCP/IP-Referenzmodell der Grundstein gelegt. In diesem aus vier Schichten bestehenden Modell, sind die nötigen Protokolle für eine Kommunikation von Rechnernetzen im Inter- oder Intranet definiert. Die Funktionalität der Schichten sind ähnlich wie die im OSI-Modell und sind wie folgt bezeichnet: Anwendungsschicht (4), Transportschicht (3), Internetschicht (2) sowie die Netzzugangsschicht (1). In den folgenden Unterkapiteln werden drei Schichten konkreter erläutert die u.a. für eine sichere Kommunikation in einer unsicheren Umgebung wichtig, aber auch für das Verständnis des weiteren Verlaufs der Arbeit notwendig sind. Die dort angesiedelten Protokolle sind jene die im TCP/IP-Referenzmodell zu finden sind und speziell für die Kommunikation im Inter- bzw. Intranet entwickelt wurden. [S.40 HPS00] [S.70ff TW12]

2.1.1 Anwendungsschicht

Zur Kommunikation in einem Netzwerk ist die Anwendungsschicht der Einstiegspunkt jeder Anwendung. Daher gehören die Anwendungen selbst nicht zu dieser Schicht, sondern geben ihre Daten an sie weiter oder erhalten diese von ihr. Die Anwendungsschicht bietet eine Vielzahl von Protokollen für die Anwendungen an. Protokolle erbringen der Anwendung einen Dienst wie z.B. das Versenden (SMTP) oder Empfangen (POP3/IMAP) einer E-Mail. Im OSI-Modell setzt die Anwendungsschicht auf die Darstellungsschicht und die wiederum auf die Sitzungsschicht auf. Da kein Bedarf mehr für die Darstellungs- und Sitzungsschicht bestand, wurden sie im TCP/IP-Referenzmodell in die Anwendungsschicht integriert. Die jeweiligen Funktionalitäten werden seitdem in der Anwendungsschicht umgesetzt.

In der Anwendung eines Endgerätes gibt es einen Prozess der die Kommunikation mit einem anderen Prozess auf einem anderen Endgerät durchführt. Man spricht auch von der sogenannten Prozess-zu-Prozess-Kommunikation. Dies geschieht über den Austausch von Nachrichten über das Netzwerk. Um diese Funktionalität anzubieten, greift der Prozess auf einen Socket zu. Der Socket ist die Schnittstelle zwischen der Anwendungs- und Transportschicht. Er besitzt zudem einen Port, damit die Transportschicht die empfangenen Nachrichten an den korrekten Socket weiterreichen kann. Das jeweilige Protokoll gibt für die Nachrichten die Struktur und andere Vorgaben vor, mit dem der Anwendungsprozess umgehen können muss. [S.112ff KR14] [S.98 Eck12] [S.73 TW12]

2.1.2 Transportschicht

Die Transportschicht übernimmt den Transport der Daten der Anwendungsschicht von einem Client zu einem anderen Client. In der Terminologie werden hier die Daten als Segmente bezeichnet. Diese werden auf Senderseite erzeugt, indem die Daten vom Prozess der darüber liegenden Schicht, sofern notwendig, in kleine Teilstücke zerlegt werden und jedem ein Header hinzugefügt wird (Multiplexing). Auf Empfängerseite erhält die Transportschicht die Segmente von der Netzwerkschicht und extrahiert die nötigen Informationen heraus, um diese an den entsprechenden Prozess der Anwendungsschicht weiterleiten zu können (Demultiplexing). Dies wird auch als Transportschicht-Multiplexing und -Demultiplexing bezeichnet. Dadurch ist bei dieser Abarbeitung auch die Rede von der Prozess-zu-Prozess-Zustellung [vgl. S.215 KR14]. Neben dieser Grundleistung wird von den zwei hauptsächlich verwendeten Protokollen der Transportschicht (Transmission Control Protocol (TCP) und User Datagram Protocol (UDP)), eine weitere angeboten: die Integritätsüberprüfung. Dabei wird dem Header eines Segmentes ein Feld für die Prüfsumme beigefügt. Wobei zu sagen ist, dass lediglich UDP diese beiden Leistungen anbietet und TCP einige mehr die im weiteren Verlauf angesprochen werden.

Die Protokolle TCP und UDP setzen die logische Kommunikation auf Transportschicht um. TCP stellt einen zuverlässigen und verbindungsorientierten Transportdienst zur Verfügung. UDP hingegen einen unzuverlässigen und verbindungslosen. [S.214ff KR14]

Bei der verbindungsorientierten Kommunikation baut der Sender vor dem Nachrichtenaustausch mit dem Empfänger eine Verbindung auf. Nach Abschluss des Austausches wird diese Verbindung wieder aufgelöst. Charakteristisch hierfür ist zum einen, dass die Nachrichten beim Empfänger in der gleichen Reihenfolge ankommen wie der Sender sie in die Verbindung gegeben hat. Und zum anderen die Bestätigungen einer empfangenen Nachricht die der Empfänger zum Sender zurückschickt. [S.60 TW12, ITWc]

Eine verbindungslose Kommunikation hingegen ist dadurch gekennzeichnet, dass keine Verbindung zwischen dem Sender und Empfänger aufgebaut wird. Die zu versendende Nachricht wird mit einer Empfängeradresse versehen und ohne Berücksichtigung nachfolgender Nachrichten schickt der Sender diese los. D.h. die Reihenfolge der eintreffenden Nachrichten kann beim Empfänger eine andere sein als beim Sender. Empfangsbestätigungen gibt es hierbei ebenfalls nicht, sodass unter Umständen Nachrichten auf dem Weg verloren gehen können. [S.60 TW12, ITWb]

Geschieht dies, liegt es in der Verantwortung der darüber liegenden Schicht wie mit einem Verlust auf Senderseite umgegangen wird. DNS empfiehlt auf Clientseite bspw. das erneute Senden einer Nachricht nach einer gewissen Zeit oder das Ausprobieren einer anderen Serveradresse (vgl. [S.32 Moc87]).

UDP

Das im RFC 768 spezifizierte UDP ist ein verbindungsloses und sehr einfach gehaltenes Protokoll zum Datentransport. Andere Funktionalitäten als das Multiplexing, Demultiplexing und der Integritätsprüfung (s. vorhergehender Abschnitt) leistet dieses Protokoll nicht. Nach dem Schichtdurchlauf wurde eine UDP-Segmentstruktur, wie in Abbildung 2.2 zu sehen ist, erzeugt. Quell- und Zielportnummer sind zur Identifikation der Anwendungsprozesse nötig.

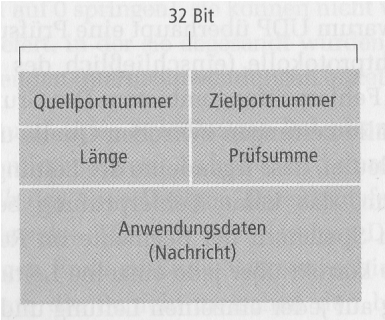

Abbildung 2.2: Segmentstruktur: UDP [S.227 KR14]

Durch die verbindungslose Kommunikation ist UDP unzuverlässig. Wie bereits geschildert, können Segmente verloren gehen oder kommen beim Empfänger in falscher Reihenfolge an. Es gibt aber dennoch Gründe warum ein unzuverlässiges Protokoll einem zuverlässigen vorzuziehen ist. Dadurch, dass es keinen Verbindungsaufbau gibt, wodurch die Segmente unmittelbar los geschickt werden, ist ein enormes Zeitersparnis und geringerer Segmentaustausch zu beobachten. Weiterhin gibt es keinen Verbindungszustand, was den Vorteil bietet, dass keine Informationen über eine Verbindung gespeichert werden. Somit steht mehr freier Speicher auf den Endsystemen zur Verfügung. Ein dritter Vorteil von UDP ist eine sehr gute Kontrolle über den Ablauf des Versendens. Es gibt beispielsweise keine Fluss- oder Fehlerkontrolle wie bei TCP, sodass ein schnelles Versenden der Segmente möglich ist. Das DNS-Protokoll nutzt i.d.R. UDP auf der Transportschicht. Dies liegt darin begründet, dass dem DNS-Dienst kleine und schlanke Segmente zum Informationsaustausch genügen, in denen die Anfragen und Antworten hinein passen. [S.223ff KR14]

TCP

TCP ist mit seiner Basisfunktionalität im RFC 793 definiert und erstreckt sich mittlerweile über viele weitere RFCs. Das liegt an der Entwicklung zu einem sehr umfangreichen und leistungsstarken Protokoll mit vielen Optionen und besonderen Regelungen. Um einen Überblick

darüber zu erhalten, wurde der RFC 4614 erstellt. Durch die verbindungsorientierte Eigenschaft ist TCP ein zuverlässiges Protokoll. Dies wird durch viele Mechanismen erreicht und benötigt entsprechend mehr Informationen diese umzusetzen. Ein Blick auf die Segmentstruktur in Abbildung 2.3 lässt dies bereits vermuten. Um die Mechanismen für die Zuverlässigkeit verstehen

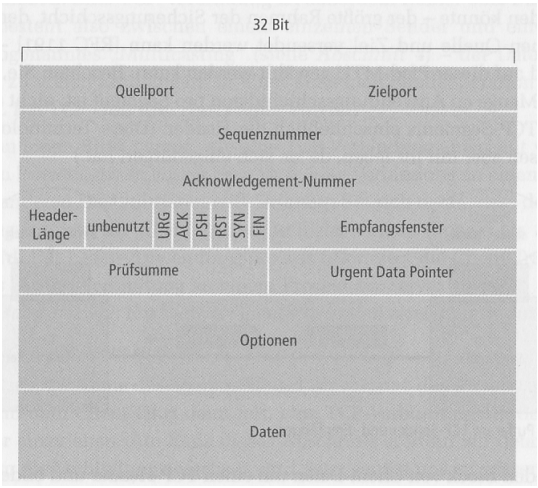

Abbildung 2.3: Segmentstruktur: TCP [S.258 KR14]

zu können, werden nun einige Felder des Segmentes weiter erläutert. [S.255ff KR14, DBEB06]

Wie bei UDP sind auch hier die Quell- und Zielportnummern sowie die Prüfsumme zu finden. Für eine zuverlässige Übertragung sind die Sequenz- und Acknowledgement-Nummer die wichtigsten Felder im Segment. Die Sequenznummer nummeriert jedes Segment des zu sendenden Datenstroms durch. Damit lassen sich die Segmente eindeutig identifizieren. Stellt der Empfänger eine Lücke in den Sequenznummern der erhaltenen Segmente fest, erkennt er, dass Segmente verloren gegangen sein müssen. Doppelte Segmente werden erkannt, wenn diese beim Empfänger eintreffen und die gleiche Sequenznummer besitzen. Die Acknowledgement-Nummer signalisiert dem Empfänger eines Segmentes, welches nächste Segment der Sender von ihm erwartet. Abbildung 2.4 verdeutlicht den Ablauf.

Das Feld für das Empfangsfenster ist für die Flusskontrolle nötig. Es wird im weiteren Verlauf konkreter darauf eingegangen. Das ACK-Flag gibt die Gültigkeit der Acknowledgement-Nummer wider. Das Flag wird als Bestätigung für den Erhalt eines oder einer Menge von Segmenten aufgefasst. Die Flags RST, SYN und FIN sind für den Aufbau und Abbau einer Verbindung nötig und werden im Folgenden weiter erläutert. Das Header-, Optionen- und Urgent-Data-Pointer-Feld sowie das PSH- und URG-Flag sind für das weitere Verständnis nicht notwendig und werden daher nicht genauer erklärt. [S.258ff KR14]

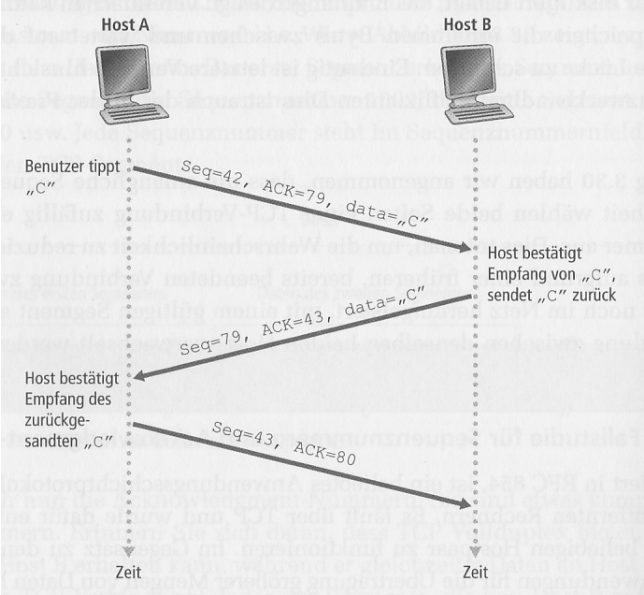

Abbildung 2.4: Demonstration Sequenz- und Acknowledgement-Nummer [S.262 KR14]

Der sogenannte Three-Way Handshake bezeichnet den Verbindungsaufbau durch TCP. Folgende 3 Schritte sind dazu nötig:

1. Der Sender schickt das sogenannte SYN-Segment ohne Nutzdaten an den Empfänger, um sich mit ihm zu synchronisieren. Dabei ist das SYN-Flag auf 1 gesetzt. Ebenfalls wird eine zufällige Sequenznummer des Senders erzeugt und in das Segment eingetragen.

2. Vorausgesetzt das SYN-Segment kommt beim Empfänger an, erzeugt dieser ein neues Segment, um darauf zu antworten. Sofern der Empfänger einen Dienst unter dem angegebenen Zielport anbietet, wird das sogenannte SYNACK-Segment versendet. Darin sind ebenfalls keine Nutzdaten enthalten. Neben dem gesetzten SYN-Flag wird eine Acknowlegement-Nummer eingetragen. Diese besteht aus der Sequenznummer des vorherigen SYN-Segments plus 1. Der Empfänger trägt nun auch seine eigene Sequenznummer in das entsprechende Feld ein. Bietet der Empfänger hingegen keinen Dienst für den Zielport an, versendet er ein Segment in dem lediglich das RST-Flag gesetzt ist.

3. Nach dem Erhalt des SYNACK-Segments bestätigt der Sender dieses. Dabei setzt er die Acknowlegement-Nummer auf die Sequenznummer des SYNACK-Segments plus 1. Das SYN-Flag muss nicht auf 1 gesetzt werden und Nutzdaten können bereits in dem Segment enthalten sein.

Nach diesen Schritten besteht eine Verbindung zwischen Sender und Empfänger, um ihre Daten miteinander austauschen zu können. [S.276f, S.281 KR14]

Entscheidet sich der Sender oder Empfänger für einen Verbindungsabbau, werden folgende 4 Segmente zwischen ihnen ausgetauscht:

1. Der Sender entscheidet sich beispielsweise für den Abbau der Verbindung und sendet ein Segment in dem das FIN-Flag auf 1 gesetzt ist. Die Sequenz- und Acknowledgement-Nummer werden, wie bereits erklärt, entsprechend gesetzt.

2. Erhält der Empfänger das Segment, antwortet er darauf mit einem Segment in dem das ACK-Flag gesetzt ist. Die Sequenz- und Acknowledgement-Nummer sind wie gehabt eingetragen.

3. Im Anschluss sendet der Empfänger ebenfalls ein Segment wie unter Punkt 1.

4. Erhält der Sender dieses, sendet er ein Segment wie unter Punkt 2 beschrieben zurück.

[S.278f KR14]

Während der Datenübertragung kommt es vor, dass Segmente verloren gehen. Daher benötigt TCP einen Mechanismus zum Wiederholen der verloren gegangenen Segmente, um einen zuverlässigen Datenaustausch gewährleisten zu können. Dies wird durch sogenannte Timer realisiert. In früheren Implementierungen wurde noch ein Timer pro Segment eingesetzt. Nach Ablauf des Timers wurde das entsprechende Segment erneut versendet. Dies führt allerdings zu einem sehr hohen Verwaltungsaufwand der Timer. Daher sollte man nun das im RFC 6298 definierte Managementverfahren für Timer verwenden. Dieses sieht nur noch einen Timer für alle Segmente vor. Mit dieser Lösung ist erstens die Integrität der Segmente gewährleistet. Zweitens fehlen keine oder treten keine doppelten Segmente im Segmentstrom auf. Und drittens kommen alle Segmente in der richtigen Reihenfolge beim Empfänger an wie sie beim Sender abgeschickt wurden. [S.266f KR14]

Damit die eingehenden Nutzdaten in ihrer korrekten Reihenfolge zwischengespeichert werden können, wird auf jeder Seite einer TCP-Verbindung ein Puffer eingesetzt. Somit kann sich die Anwendungsschicht die bereitgestellten Daten abholen. Der Puffer ist allerdings nur begrenzt groß. Ebenso müssen die Nutzdaten nicht immer rechtzeitig von der Anwendungsschicht abgeholt werden. Dies hat zur Auswirkung, dass nicht zwangsläufig freier Speicher zur Verfügung steht, um weitere ankommende Nutzdaten dort abzulegen. Daher setzt TCP eine Flusskontrolle auf Empfängerseite ein. Mit ihr kann die Sendegeschwindigkeit zwischen Sender und Empfänger angepasst werden, damit der Puffer auf Empfängerseite nicht überläuft. Hierfür wird das kürzlich erwähnte Empfangsfenster aus der TCP-Segmentstruktur eingesetzt.

Dieses gibt dem Sender vor wie viel freier Puffer auf Empfängerseite zur Verfügung steht. [S.274f KR14]

Die Überlastkontrolle hingegen findet auf Senderseite statt, indem er eine Geschwindigkeitsregelung durchführt. Er passt somit seine Sendegeschwindigkeit an das zugrunde liegende Netzwerk an. Bemerkt der Sender eine hohe Netzwerkauslastung, drosselt er sein Sendetempo. Erkennt er hingegen eine geringe Netzwerkauslastung, kann er sein Sendetempo steigern. Dieser Mechanismus wurde daher in einem TCP-Überlastkontroll-Algorithmus beschrieben und im RFC 5681 manifestiert. [S.292ff KR14]

2.1.3 Netzwerkschicht

In der Netzwerkschicht werden die Daten nicht mehr Segmente sondern als Pakete bezeichnet. Der Dienst den diese Schicht erbringt, ist die Übertragung und Weiterleitung der Pakete durch ein unterschiedlich großes Netzwerk. Dieses kann weit verzweigt und aus vielen Komponenten (z.B. Router) bestehen. Die Netzwerkschicht muss den Aufbau eines Netzwerkes, die sogenannte Topologie, kennen. Dieses Wissen benötigt sie, um passende Wege für die Pakete auf dem Weg zum Empfänger ermitteln zu können. Passend bedeutet, dass keine Konflikte auf dem Weg liegen oder um einen generellen Weg zu finden. Um dies zu erreichen, werden unterschiedliche Ansätze verfolgt die in den folgenden Abschnitten erörtert werden. Wie bereits in der Transportschicht gesehen, gibt es auch in dieser Schicht verbindungslose und verbindungsorientierte Dienste. Das bekannteste Protokoll das einen verbindungslosen Dienst anbietet ist das Internet Protocol (IP), welches gleich genauer behandelt wird. [S.412f TW12]

Die einfachste Variante Pakete auf den Weg zu schicken, ist die verbindungslose. Es ist kein Verbindungsaufbau mit dem Empfänger nötig, da die Pakete mit der entsprechenden Empfängeradresse versehen und anschließend losgeschickt werden. Entscheidend dabei ist, dass Pakete unter Umständen ganz andere Wege zurücklegen können als andere. D.h. sie sind unabhängig voneinander und können entsprechend eher oder später am Ziel ankommen. Die Wege auf die die Pakete unterwegs sind, werden durch sogenannte Routing-Algorithmen ermittelt. Diese verfolgen das Ziel, dass Wege ermittelt werden auf denen keine Konflikte, wie z.B. ein Stau, vorliegen.

Die etwas aufwendigere Variante Pakete zu versenden ist die verbindungsorientierte. Dafür muss eine Verbindung zwischen dem Router den die Pakete zuerst passieren zu dem Router den sie zuletzt passieren aufgebaut werden. Eine solche Verbindung wird Virtual Circuit (VC) genannt. Im Gegensatz zu TCP besitzt bei dieser virtuellen Verbindung jeder einzelne auf dem Weg liegende Router die Kenntnis über die Verbindung und ist auch an ihrem Aufbau beteiligt.

Bei TCP hingegen wissen nur die beiden Endsysteme - nicht aber die Router auf dem Weg - von einer bestehenden Verbindung und handeln die nötigen Parameter unter sich aus. Erst beim Bestehen der virtuellen Verbindung werden die Pakete losgeschickt. Diese wird auch für den gesamten Pakettransfer genutzt. Ein genutztes Protokoll das einen verbindungsorientierten Dienst anbietet ist das Multi-Protocol Label Switching (MPLS). Beispielsweise verwenden Internet Service Provider (ISP) dieses Protokoll innerhalb ihrer Netze. Ein Einsatzgrund kann z.B. sein, dass der ISP den Verkehr, der durch seine Netze verläuft, verwalten möchte. Dies ist vor allem durch die bestehende Verbindung möglich, da hiermit der Datenverkehr kontrollierbar ist. [S.415ff TW12] [S.343f KR14]

Verbindungslose und verbindungsorientierte Dienste auf der Netzwerkschicht können unter einigen Aspekten verglichen werden. Dabei kristallisieren sich einige Vor- und Nachteile beider Dienste heraus. An dieser Stelle wird aber nur kurz darauf eingegangen. Gerade bei der Überlastkontrolle ist der verbindungsorientierte Dienst klar im Vorteil. Er weiß bereits im Voraus welche Ressourcen (Bandbreite etc.) genutzt werden und reserviert diese in den einzelnen Routern. Bei dem verbindungslosen Dienst ist dies schwieriger, da durch die Unabhängigkeit der Pakete nicht gesagt werden kann, wann und in welchem Router Ressourcen benötigt werden. Ein Nachteil des verbindungsorientierten Dienstes ist, dass bei einem Routerausfall, alle virtuellen Verbindungen (VC) abgebrochen werden müssen die über ihm laufen. Der verbindungslose Dienst ist so konzipiert, dass bei einem Routerausfall ein neuer Weg für das verloren gegangene Paket gesucht wird. [S.418f TW12]

IP

Das Internet Protocol (IP) bietet auf der Netzwerkschicht ein Paketformat sowie eine Adressierungskonvention der Pakete an und setzt somit die logische Kommunikation auf dieser Schicht um. Es wurde 1981 im RFC 791 erstmals spezifiziert und liegt mittlerweile in der Version 4 und 6 vor. Mit IPv4 hat alles begonnen, daher ist es ebenfalls in diesem RFC spezifiziert. Wohingegen IPv6 hauptsächlich im RFC 2460 beschrieben wird. Aus verschiedenen Gründen wird nun eine Umstellung von Version 4 auf 6 vorgenommen.

Eine IP-Adresse identifiziert eine Netzwerkschnittstelle eines Hosts eindeutig in einem Netzwerk. Die Adressierungskonvention bei IPv4 lässt allerdings nur 2^{32} IP-Adressen zu. Durch das stetige Wachstum des Internetgebrauchs hat man nun gemerkt, dass eine Umstellung auf IPv6 - das bereits im Jahre 1998 standardisiert wurde - unerlässlich ist. Version 6 stellt mit 2^{128} IP-Adressen deutlich mehr als sein Vorgänger zur Verfügung. Mit dieser großen Anzahl an möglichen Adressen wird nun davon ausgegangen, dass eine nochmalige Erweiterung nicht mehr nötig sei. IPv6 bringt noch andere weitere Vorteile gegenüber IPv4 mit sich. Einer liegt

darin, dass der Header deutlich verkürzter ist als bei IPv4. Statt dreizehn Felder besitzt dieser in der Version 6 nur sieben. Dies bewirkt eine deutlich schnellere Verarbeitung der Pakete in den Routern, womit sie schneller weitergeleitet werden können. Ebenfalls wurden in IPv4 nicht immer alle Felder im Header genutzt die in Version 6 nun optional sind, da sie eigentlich nicht nötig waren. Auch dies beschleunigt die Abarbeitung der Pakete in den Routern. Ein letzter hier genannter Vorteil von IPv6 ist die automatische Verschlüsselung der Pakete mit IPsec. Dies war von Beginn an Standard in Version 6, wird aber mittlerweile auch für IPv4 angeboten und ebenfalls mit IPsec realisiert. [S.500ff und S.520ff TW12, Int81, DH98, KS05]

Wie bereits erörtert, gibt es auf der Transport- und Netzwerkschicht die beiden Kategorien von verbindungslosen und verbindungsorientierten Protokollen. Daraus ergeben sich Kombinationen von Protokollen beider Kategorien auf den jeweiligen Schichten. Somit können bspw. TCP auf Transportschicht und IP auf Netzwerkschicht miteinander kombiniert werden. 1989 wurde diese Kombination als der heutige Internetstandard festgelegt und ist als TCP/IP-Referenzmodell bekannt (vgl. [Bra89] und oben). Aber auch UDP und IP als verbindungslose Protokolle können kombiniert werden. [S.214 KR14] [S.70f TW12]

2.2 Allgemeine kryptographische Algorithmen

2.2.1 Symmetrische Verschlüsselung

Schon Gaius Iulius Caesar verwendete die symmetrische Verschlüsselung, um geheime Botschaften zu verschlüsseln, um sie vor unbefugter Einsicht zu schützen. Die Idee dahinter ist, dass jeder Buchstabe mit einem anderen Buchstaben aus dem Alphabet ersetzt wird. Die Ersetzung findet nach dem Prinzip der zyklischen Rotation um k-Buchstaben nach rechts statt. D.h. der Buchstabe *A* wird bei einem Schlüssel $k = 3$ auf den Buchstaben *D* abgebildet. Zum Entschlüsseln muss daher um k-Buchstaben nach links rotiert werden. Daraus ergibt sich das Grundprinzip der symmetrischen Verschlüsselung zwischen mehreren Parteien: der Schlüssel k muss allen Parteien bekannt sein und für Dritte geheim bleiben. Außerdem darf ein Angreifer das Verschlüsselungsverfahren durchaus kennen, da er ohne Schlüssel keine Entschlüsselung vornehmen kann. Die nachfolgenden zwei Unterpunkte zeigen kurz zwei berühmte Algorithmen aus diesem Bereich. [S.39ff und 43f Sch13a]

DES

Der Data Encryption Standard (DES) wurde in den 70er Jahren von IBM und der NSA entwickelt und 1977 vom heutigen NIST standardisiert. Trotz der kurzen Schlüssellänge von

tatsächlich 56 Bit, galt er seitdem als das etablierteste Verfahren für die symmetrische Verschlüsselung. Heutzutage ist diese kurze Schlüssellänge allerdings nicht mehr sicher genug, sodass der DES in modifizierter Form eingesetzt werden sollte. [S.81f Sch13a] [S.334f Eck12]

DES erhält die zu verschlüsselnden Daten in 64 Bit große Blöcke und führt mehrere Operationen in 16 Runden durch. Nachdem eine Initialpermutation durchgeführt wurde, wird der Block in zwei Hälften mit je 32 Bit geteilt. Die eine Hälfte fließt in eine sogenannte F-Funktion ein, in der der Block u.a. mit einem Teil des verwendeten Schlüssels verknüpft wird. Im Anschluss wird das Ergebnis mit der anderen Hälfte verknüpft. Dieser Vorgang wird weitere 15 Mal wiederholt, bis am Ende eine Endpermutation den endgültig verschlüsselten Block erzeugt. DES führt im wesentlichen drei Grundoperationen während der Verschlüsselung durch: Permutation, XOR-Verknüpfung und Substitution. Sie werden heutzutage in symmetrischen Verfahren verwendet, da die jeweiligen Operationen sehr schnell von einem System durchgeführt werden können. [S.83ff Sch13a] [S.336f Eck12]

DES wurde in den letzten Jahrzehnten von vielen Experten analysiert, die bestätigten, dass der Verschlüsselungsablauf ohne Zweifel sicher und fehlerfrei ist. Daher sind mit DES verschlüsselte Daten nur noch mit einer vollständigen Schlüsselraumsuche zu entschlüsseln. Allerdings hat man bereits 1997 einen Erfolg verzeichnen können und ist mit den heutigen Rechenleistungen noch viel schneller und effizienter. Daher sollte heutzutage Triple-DES angewendet werden, bei dem DES dreimal hintereinander mit drei voneinander unabhängigen Schlüsseln angewendet wird. Daraus ergibt sich im Endeffekt ein Verfahren mit 112 Bit - statt 168 Bit - Schlüssellänge, da immer noch Meet-in-the-middle-Angriffe möglich sind. [S.87ff Sch13a] [S.340ff Eck12]

AES

Advanced Encryption Standard (AES) ist der offizielle Nachfolger von DES und wurde 2000 vom NIST als dieser bekannt gegeben. Er wurde in einem Wettbewerb aus insgesamt 15 Kandidaten ausgewählt und hatte den Namen Rijndael. Dort dominierte Rijndael durch sein einfaches Design, einer einfachen Implementierung sowie einer guten Performanz sowohl auf Soft- als auch auf Hardwareebene. AES gilt heute als eines der praktisch sichersten symmetrischen Verschlüsselungsverfahren und wird in vielen Bereichen eingesetzt. [S.343f Eck12]

Die Blocklänge von AES beträgt standardmäßig 128 Bit, wohingegen Rijndael auch die Längen 192 und 256 Bit unterstützt. Die Schlüssellängen betragen 128, 192 oder 256 Bit. Abhängig von der Schlüssellänge durchläuft ein Block - wie bei DES - eine bestimmte Anzahl an Runden. In jeder Runde wird der Eingabeblock - außer der letzten - durch vier verschiedene Operationen bearbeitet, sodass am Ende ein verschlüsselter Block entsteht. Neben den bereits

genannten Operationen Permutation, XOR-Verknüpfung und Substitution, besitzt AES eine vierte Operation: die Diffusion. [S.129 Sch13a] [S.335f Eck12]

Die jahrelangen Analysen zeigen, dass das AES-Verfahren aus praktischer Sicht als sehr sicher gilt. Im Gegensatz zu DES, ist eine vollständige Schlüsselraumsuche derzeit praktisch nicht durchführbar. Lediglich die geringe Rundenanzahl wurde von Beginn an kritisch gesehen. Dies zeigen bereits bekannte Angriffe auf Rijndael der mit geringerer Rundenanzahl betrieben wurde (vgl. [FKL+01]). Ebenfalls gibt es einige theoretische Angriffe wie bspw. die Biclique-Kryptoanalyse (vgl. [BKR11]). Damit kann die Zeitkomplexität der vollständigen Schlüsselraumsuche um den Faktor vier verringert werden. Letztlich hat die Vergangenheit gezeigt, dass es theoretisch immer wieder möglich war das Verfahren anzugreifen. Es bleibt daher abzuwarten, wann ein wirklich praktischer Angriff auf AES entwickelt wird. [S.135ff Sch13a] [S.346f Eck12]

2.2.2 Asymmetrische Verschlüsselung

Die asymmetrische Verschlüsselung unterscheidet sich von der symmetrischen Verschlüsselung u.a. darin, dass zwei voneinander abhängige Schlüssel verwendet werden. Zum einen ein privater geheimer Schlüssel und zum anderen ein öffentlicher Schlüssel - der i.d.R. allgemein zugänglich ist. Ein weiterer Unterschied ist, dass symmetrische Verschlüsselungsverfahren deutlich schneller als asymmetrische Verfahren sind. Daher wird dieses Verfahren zur verschlüsselten Kommunikation heutzutage bevorzugt. Das Problem ist allerdings, dass der symmetrische Schlüssel allen teilnehmenden Parteien bekannt sein muss. In einem unsicheren Netzwerk sollte dieser Schlüssel nicht unverschlüsselt versendet werden. Daher ist der sichere Austausch von symmetrischen Schlüsseln der heutige Haupteinsatzzweck der asymmetrischen Verschlüsselung und wird auch als Hybridverfahren bezeichnet. Ein weiteres Einsatzgebiet ist die Signierung von Nachrichten. Im folgenden werden zwei bedeutsame und bekannte asymmetrische Verfahren kurz vorgestellt. [S.175ff Sch13a]

RSA

RSA wurde 1978 von den drei bekannten Kryptografen Rivest, Shamir und Adleman entwickelt und gilt heute als de facto Standard in der asymmetrischen Verschlüsselung. Es wird, wie bereits erwähnt, hauptsächlich für den Schlüsselaustausch, aber auch zum Signieren von Nachrichten genutzt. [S.351 Eck12]

Die Berechnung des Schlüsselpaares e und d wird im folgenden kurz erläutert. Die Arbeitsweise von RSA beruht auf einer Einwegfunktion mit Falltür. Die Einwegfunktion ist die

Multiplikation von zwei gewählten Primzahlen *p* und *q* dessen Ergebnis nur sehr aufwendig faktorisiert werden kann. Das Ergebnis ist das sogenannte Modulo *n* und wird zur Berechnung der Eulerschen ϕ-Funktion verwendet. Zu diesem Wert $\phi(n)$ muss nun ein teilerfremder Wert *e* berechnet werden. Dieser Wert und *n* stellen den öffentlichen Schlüssel dar. Anschließend wird das multiplikative Inverse *d* von *e* bezogen auf $\phi(n)$ berechnet. Die Werte *d* und *n* bilden somit den privaten Schlüssel und sind die oben angesprochene Falltür zur Entschlüsselung von Daten. Nun kann mit $C = M^e \pmod{n}$ verschlüsselt und mit $M = C^d \pmod{n}$ wieder entschlüsselt werden. Zur Signierung muss $Sig = M^d \pmod{n}$ und zur Verifizierung $M = Sig^e \pmod{n}$ berechnet werden. [S.190f Sch13a]

Durch die Abhängigkeit der Schlüssel von den generierten Primzahlen, ist eine Wahl dieser Werte von entscheidender Bedeutung für die Sicherheit von RSA. Daher müssen möglichst große und voneinander weit entfernte Primzahlen gewählt werden, damit ein Faktorisierungsangriff erschwert wird. Dies kann dadurch erreicht werden, indem bei der Schlüsselgenerierung große Schlüssellängen verwendet werden. Aktuell empfiehlt das BSI eine Schlüssellänge von mindestens 2000 Bit (vgl. [BSI16]). Grund dafür ist, dass es bereits erfolgreiche Faktorisierungsangriffe auf geringere Schlüssellängen gibt (vgl. [KAF+10]). Eine weitere Schwachstelle ist die Multiplikativität von RSA erzeugten Signaturen. Zwei Signaturen können miteinander multipliziert werden und ergeben eine dritte gültige Signatur. Wird aber der Hashwert einer Nachricht signiert, kann dies nicht mehr passieren. Abschließend kann man sagen, dass bei korrekter Implementierung und Verwendung ausreichend langer Schlüssel, die mit RSA verschlüsselten Daten praktisch nicht durch Unbefugte zu entschlüsseln sind. [S.358ff Eck12]

Diffie-Hellman

Das Diffie-Hellman-Verfahren wurde 1976 von ihren Namensgebern Whitfield Diffie und Martin Hellman entwickelt und ist ein reines Schlüsselaustauschverfahren. Die bei diesem Verfahren eingesetzte Einwegfunktion ist der diskrete Logarithmus. Wollen zwei Parteien einen gemeinsamen Sitzungsschlüssel austauschen, einigen sie sich auf die zwei Zahlen *p* und *g*. Wobei *p* eine Primzahl und *g* idealerweise ein Generator von *p* ist und öffentlich bekannt sein dürfen. Nun wählen beide Parteien einen privaten Schlüssel *x* und *y* und berechnen ihren jeweiligen öffentlichen Schlüssel: $a = g^x \pmod{p}$ bzw. $b = g^y \pmod{p}$. Nun können diese beiden Schlüssel über ein unsicheres Netzwerk miteinander getauscht werden. Der Sitzungsschlüssel wird dann wie folgt berechnet: $k = b^x \pmod{p}$ bzw. $k = a^y \pmod{p}$. Mit diesem Schlüssel *k* kann nun der ausgetauschte Datenverkehr verschlüsselt werden. [S.185ff Sch13a]

Damit der Sitzungsschlüssel berechnet werden kann, muss ein Angreifer den diskreten Logarithmus berechnen. Durch die Problematik, dass die Primzahl p meist vorgegeben ist und eine Schlüssellänge von 512 Bit gewählt wurde, ist dies heutzutage in ca. einer Minute möglich. Zusätzlich ist es möglich, dass durch einen Logjam-Angriff ein vereinbarter Schlüsselaustausch mit bspw. 2048 Bit Schlüssellänge, auf 512 Bit herunter gesetzt werden kann, wodurch die Berechnung des Sitzungsschlüssel wieder möglich ist. Daher sollte eine höhere Schlüssellänge in Betracht gezogen und Ciphersuites mit einer Schlüssellänge kleiner und gleich 1024 Bit abgeschaltet werden. Des Weiteren ist ein Man-in-the-Middle-Angriff möglich, da durch das Verfahren keine Authentizität der Parteien gewährleistet wird. Durch das Station-to-Station-Protokoll kann aber eine Authentifikation gewährleistet werden. Letztlich ist das Diffie-Hellman-Verfahren ein weit verbreitetes Konzept, was bei korrekter Verwendung und mit ausreichend langen Schlüsseln zu einem sicheren Schlüsselaustausch führt. [S.445ff Eck12, ABD+15]

2.2.3 Kryptographische Hashfunktionen

Kryptographische Hashfunktionen haben deutlich höhere Anforderungen als gewöhnliche Hashfunktionen die z.B. im Datenbankbereich zur schnellen Suche von Datensätzen eingesetzt werden. Diese sind i.d.R. einfacher aufgebaut, sodass die Wahrscheinlichkeit für das Auftreten von Kollisionen höher ist. Aus kryptographischer Sicht sollte jedoch diese Wahrscheinlichkeit möglichst gering sein. Somit kann man gewährleisten, dass ein Objekt eindeutig durch einen Hashwert charakterisiert wird und dessen Integrität sichergestellt ist. Dies ist ein Grund, weshalb kryptographische Hashfunktionen heutzutage in Verbindung mit digitalen Signaturen eingesetzt werden. Es wird zunächst der Hashwert, bspw. einer E-Mail, gebildet und dieser im Anschluss digital signiert. Dies hat ebenfalls den entscheidenden Vorteil, dass die Signatur schneller gebildet wird als bei der Signierung der gesamten E-Mail.
Kryptographische Hashfunktionen sind laut Definition schwache Hashfunktionen mit der zusätzlichen Eigenschaft, dass bei zwei unterschiedlichen Eingabewerten, es praktisch nicht möglich ist, zwei identische Hashwerte zu erhalten. Sie berechnen die Hashwerte, indem die Eingabe in Blöcke fester Länge geteilt wird. Jeder Block wird durch eine Kompressionsfunktion zu einem Hashwert gebildet, der wiederum als Eingabe für die nächste Kompressionsfunktion dient (vgl. Abbildung 2.5). Der zuletzt berechnete Wert bildet den eigentlichen Hashwert. Durch diesen Vorgang ist gewährleistet, dass die Eingabe durch diesen einen Wert repräsentiert wird. [S.376ff Eck12]

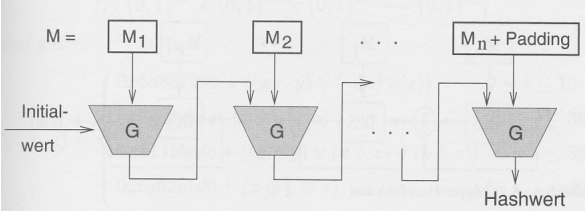

Abbildung 2.5: Allgemeine Berechnung eines Hashwertes [S.381 Eck12]

SHA-1

SHA-1 wurde 1993 vom NIST veröffentlicht und wurde zusammen mit der NSA entwickelt. Der Algorithmus gehört zur Familie der dedizierten Hashfunktionen. Die Blockgröße beträgt 512 Bit und der Hashwert besitzt eine Länge von 160 Bit. Jeder Block wird darüber hinaus in 16 Teilblöcke à 32 Bit gegliedert. Zur Verarbeitung eines Blocks werden 80 Funktionen benötigt die jeweils drei 32 Bit-Eingabeblöcke verwenden und einen 32-Bit-Ausgabeblock produziert. Diese Funktionen und vordefinierten Konstanten bilden die Kompressionsfunktion, mit der letztlich ein 160 Bit Hashwert gebildet wird. Dieser wird, wie bereits beschrieben, als Eingabe für die Verarbeitung des nächsten Blocks genutzt. [S.382 Eck12, oST15]

Seit 2005 ist bekannt, dass eine Kollision praktisch in deutlich weniger Schritten für den SHA-1 berechnet werden kann als theoretisch nötig. Demnach werden statt 2^{80} lediglich 2^{69} bzw. 2^{63} Schritte benötigt (vgl. [WYY05b, WYY05a]). Durch die derzeitigen Leistungen von modernen Computersystemen ist dies ein durchaus realistischer Angriff. Laut [BSI16] ist die Verwendung von SHA-1 heute nicht mehr zu empfehlen. [S.384 Eck12]

SHA-2

Zur SHA-2-Familie gehören mittlerweile die Verfahren SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 und SHA-512/256. Sie wurden - mit Ausnahme von SHA-224 aus dem Jahr 2004, sowie SHA-512/224 und SHA-512/256 aus dem Jahr 2015 - bereits im Jahre 2002 vom NIST im Secure Hash Standard (SHS) vorgestellt. Der letzte Wert in der Bezeichnung gibt die Länge des Hashwertes in Bits an. Die Blockgröße von SHA-224 und 256 beträgt 512 Bit, die der anderen 1024 Bit. Aus diesem Grund sind die 16 Teilblöcke 32 Bit bzw. 64 Bit groß. Im Gegensatz zu SHA-1 arbeiten diese sechs Verfahren mit 6 Funktionen und erhalten ebenfalls jeweils drei 32 bzw. 64 Bit-Eingabeblöcke und erzeugen jeweils einen 32 bzw. 64 Bit-Ausgabeblock. Auch die Konstanten unterscheiden sich zu SHA-1 in Anzahl und Länge. [oST02, oST15]

Alle SHA-2 Verfahren sind bzgl. des bekannten Angriffs von [WYY05b] bzw. [WYY05a] nicht betroffen. Andere Angriffe oder Schwachstellen sind derzeit nicht bekannt. Daher werden

sie zur Zeit - neben SHA-3 - als die sichersten kryptographischen Hashfunktionen angesehen. [S.239f Sch13a, BSI16]

2.3 Transparente Verschlüsselung im OSI-Modell

In den folgenden Unterkapiteln wird exemplarisch je ein bekanntes Protokoll der Transport- und Netzwerkschicht vorgestellt. Diese dienen der Verschlüsselung von Segmenten bzw. Paketen auf den jeweiligen Schichten. Protokolle der Anwendungsschicht werden in Kapitel 3 genauer untersucht, da diese im Fokus der Arbeit stehen. Dazu gehören bspw. OpenPGP, S/MIME und zum Teil auch SMTP/TLS.

2.3.1 Transportschicht: TLS

SSL ist das wohl bekannteste und meist verschwendete Sicherheitsprotokoll im Internet. Es wurde 1994 von Netscape entwickelt und sollte ursprünglich die Webkommunikation mit HTTP absichern. Das TLS-Protokoll ist die Weiterentwicklung von SSL 3.0 und wurde 1999 von der Internet Engineering Task Force (IETF) in der Version 1.0 im RFC 2246 standardisiert. Dabei wurden keine dramatischen Änderungen vorgenommen. Sie sind aber dennoch derart signifikant, sodass SSL 3.0 und TLS 1.0 nicht kompatibel zueinander sind. Die aktuellste Version ist TLS 1.2, wobei zur Zeit an einem Entwurf für die Version 1.3 gearbeitet wird [Res16]. Im RFC 7525 stehen Empfehlungen zur Verwendung von TLS die verlauten lassen, dass SSL in seinen alten Versionen 2.0 und 3.0 nicht weiter verwendet werden dürfen. Der Grund sind u.a. die als zu unsicher eingestuften Verschlüsselungsalgorithmen. Weitere Empfehlungen für z.B. einem sicheren Schlüsselaustausch sind diesem RFC ebenfalls zu entnehmen. [S.145ff Sch14, DA99, DR08, SHSA15, Res16]

Der Hauptzweck von TLS ist das Anbieten von Geheimhaltung, Authentizität und Integrität zwischen zwei kommunizierenden Parteien. Logisch ist TLS zwischen der Anwendungsschicht und der Transportschicht anzusiedeln. Es übernimmt somit Aufgaben der Schichten 5 und 6 im OSI-Modell. Die Schicht 5 fasst bspw. für das zustandslose Hypertext Transfer Protocol (HTTP) mehrere TCP-Verbindungen zu einer Sitzung zusammen. Dies hat den entscheidenden Vorteil, dass eine effizientere Verwaltung der Verbindungen gewährleistet wird. Der Unterschied von Sitzung und Verbindung im Bezug auf TLS wird im Laufe dieses Unterkapitels genauer erläutert. Daher ist TLS in zwei Schichten - die unterschiedliche Protokolle implementieren - einzuteilen. Auf der einen Seite dem Handshake-Protokoll zum Aufbau einer Verbindung und dem Aushandeln von Sitzungsschlüsseln sowie kryptographischen Algorithmen. Auf der anderen Seite dem Record-Protokoll zur Vorbereitung der Weiterleitung der Daten aus

der Anwendungsschicht an die Transportschicht. Die beiden weiteren Protokolle sind das Alert- und das Change Cipher Spec-Protokoll. Sie setzen, wie das Handshake-Protokoll, auf das Record-Protokoll auf (siehe Abb. 2.6). Ein entscheidender Vorteil ist die Unabhängigkeit zur Anwendungsschicht. Jede Anwendung (HTTP, SMTP etc.) kann daher auf TLS aufsetzen und muss sich über einen sicheren Datentransport keine Gedanken machen. Wie die erwähnten Protokolle genau arbeiten wird nun detaillierter beschrieben. [S.145ff Sch14] [S.791 Eck12, DR08]

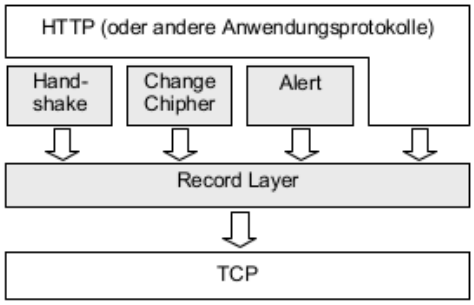

Abbildung 2.6: Aufbau TLS-Protkoll [S.158 Sch14]

Bevor mit einem verschlüsselten Datentransport begonnen werden kann, müssen bestimmte kryptographische Parameter ausgetauscht werden. Der Ablauf des TLS-Handshake-Protokolls ist in Abbildung 2.7 zu sehen:

	Client		Server	Nachricht
1		⟶		ClientHello
2		⟵		ServerHello Certificate (optional) ServerKeyExchange (optional) CertificateRequest (optional) ServerHelloDone
3		⟶		Certificate (optional) ClientKeyExchange CertificateVerify (optional) ChangeCipherSpec Finished
4		⟵		ChangeCipherSpec Finished
5		⟷		Anwendungsdaten

Abbildung 2.7: TLS-Handshake-Protokoll [S.794 Eck12]

Beginnt der Client damit eine Verbindung zu einem Server aufzubauen, schickt er eine *ClientHello-Nachricht*. Diese beinhaltet:

1. SSL Protokollversion die der Client bevorzugt. TLS 1.0: 3.1; TLS 1.1: 3.2; TLS 1.2: 3.3. Wie zu erkennen wurde die Versionierung von SSL 3.0 fortgeführt.

2. *ClientRandom*: Eine Zufallszahl zur späteren Berechnung des sogenannten *master_secret*.

3. *SessionID*: Entweder hat sie den Wert *null* oder es liegt bereits eine vom Server vergebene *ID* einer älteren Sitzung die wieder aufgenommen werden soll vor. Dies hat den Vorteil, dass ein verkürzter Handshake durchgeführt und somit schneller eine Verbindung aufgebaut werden kann. Die kryptographischen Parameter aus der bereits ausgehandelten Session werden dabei weiter genutzt.

4. *Ciphersuites*: Eine Liste möglicher kryptographischer Verfahren (*CipherSpec*) die der Client unterstützt. Sie werden durch 2 Byte codiert. Bspw. liefert *{0x00,0x2F}* die *Ciphersuite*: *TLS_RSA_WITH_AES_128_CBC_SHA*. RSA wird zum Schlüsselaustausch genutzt, AES 128 Bit für die Verschlüsselung der Daten im CBC-Modus und SHA-1 zum Bilden von Hashwerten. Diese ist unter TLS 1.2 auch die verpflichtende (*mandatory*) Suite die jede Anwendung implementieren muss, damit Client und Server mindestens eine gemeinsam haben.

5. Eine Liste von Kompressionsfunktionen wie in [Hol04] definiert.

[S161f und S.165f Sch14, DR08]

Der Server antwortet auf die *ClientHello-Nachricht* mit *ServerHello*. Er schickt dem Client nun seine ausgewählten Parameter:

1. Die vom Server gewählte Protokollversion, die der erhaltenen Version des Clients entspricht oder eine niedrigere.

2. *ServerRandom*: Eine Zufallszahl zur späteren Berechnung des sogenannten *master_secret*, die vollkommen unabhängig vom *ClientRandom* gebildet wird.

3. *SessionID*: War die *SessionID* in der *ClientHello-Nachricht null*, wird eine neue eindeutige ID generiert. War sie es nicht, prüft der Server, ob er noch alle nötigen Informationen über die Session besitzt und kann diese wiederverwenden.

4. *Ciphersuite*: Die stärkste vom Client unterstützte *CipherSuite* bzw. *CipherSpec* die er in der Liste der *CipherSuites* des Clients findet. I.d.R. die an oberster Stelle. Oder die bereits gewählte Suite unter der alten *SessionID*.

5. Eine vom Server gewählte Kompressionsfunktion. Oder die bereits gewählte Funktion unter der alten *SessionID*.

[S.161f und S.165f Sch14, DR08]

Nachdem beide Seiten die zu verwendenden Algorithmen und andere nötige Informationen ausgetauscht haben, werden noch weitere Informationen benötigt, um die entsprechenden Schlüssel generieren zu können. Um die Authentifikation des Servers zu gewährleisten, sendet dieser, sofern RSA als Schlüsselaustausch gewählt wurde, eine *Certificate-Nachricht* im Anschluss der *ServerHello-Nachricht*. In dieser ist der öffentliche Schlüssel enthalten. Dieser muss durch ein X.509v3-Zertifikat repräsentiert sein, sofern nicht anders ausgehandelt. Wurde aber bspw. der Schlüsselaustausch mit Diffie-Hellman ausgehandelt, reichen die Informationen in der *Certificate-Nachricht* nicht aus. Daher folgt direkt darauf, oder nach *ServerHello*, die *ServerKeyExchange-Nachricht*. In ihr sind die nötigen Parameter zur Diffie-Hellman Schlüsselberechnung enthalten (Generator g, Primzahl p und öffentlicher Schlüssel des Server y). Die *CertificateRequest-Nachricht* kann vom Server gesendet werden, wenn er vom Client ein Zertifikat anfordert, um sich ebenfalls zu authentifizieren. Mit *ServerHelloDone* schließt der Server den Schlüsselaustausch ab und wartet auf den Client. [S.161f und S.167f Sch14, DR08]

Hat der Client die *ServerHelloDone-Nachricht* erhalten, schickt er sein eigenes Zertifikat, sofern eine *CertificateRequest-Nachricht* vom Server geschickt wurde. Besitzt er keines, sendet er eine leere *Certificate-Nachricht*. Die *ClientKeyExchange-Nachricht* muss direkt nach der vorhergehenden, oder nach dem *ServerHelloDone* folgen. Erfolgt der Schlüsselaustausch mit RSA, generiert der Client das *pre_master_secret* und codiert dieses als PKCS#1. Anschließend verschlüsselt er dieses mit dem öffentlichen Schlüssel des Servers. Wurde hingegen Diffie-Hellman gewählt, berechnet der Client seinen öffentlichen Schlüssel aus den zuvor erhaltenen Parametern vom Server. Mit der *CertificateVerify-Nachricht* beweist er dem Server seine Identität. Er bildet einen Hashwert über alle bisher ausgetauschten Nachrichten und verschlüsselt diesen mit seinem privaten Schlüssel. Zum Verifizieren bildet der Server ebenfalls den Hashwert, entschlüsselt die *CertificateVerify-Nachricht* und gleicht die Hashwerte ab. [S.161f und S.167f Sch14, DR08]

Nun liegen auf beiden Seiten alle Informationen vor, um alle erforderlichen Schlüssel generieren zu können. Das *master_secret* wird nun aus dem *pre_master_secret*, den beiden Random-Werten sowie der Zeichenkette *'master secret'* erzeugt, indem es in eine Pseudo Random Function (PRF) gegeben wird. Mit diesem Wert, den Random-Werten und der Zeichenfolge *'key expansion'*, werden nun auf jeder Seite jeweils ein MAC-Schlüssel, ein Sitzungsschlüssel und ggf. ein Initialisierungsvektor mittels der PRF gebildet. [S.167f Sch14, DR08]

ChangeCipherSpec ist in dem Sinne keine Nachricht des TLS-Protokolls, sondern ein eigenständiges Protokoll. Damit wird nun eine neue *CipherSpec* - als die noch beim Handshake genutzte - gewählt und damit die zuvor berechneten Schlüssel ausgetauscht. Alle folgenden Nachrichten werden bereits mit diesen neuen Parametern verschlüsselt übertragen. [S.169 Sch14, DR08]

Im Anschluss signalisieren die *Finished-Nachrichten*, dass der Handshake vorbei ist. Es wird über alle zuvor ausgetauschten Nachrichten, mit Ausnahme der *ChangeCipherSpec-Nachrichten* und eventuellen *Alert-Nachrichten*, ein MAC-Wert gebildet, um die Authentizität und Integrität zu gewährleisten. [S.169 Sch14, DR08]

TLS unterscheidet zwischen Sitzungen und Verbindungen. Eine Sitzung enthält demnach die *SessionID*, X.509v3-Zertifikate der Teilnehmenden, die gewählte Kompressionsfunktion und *CipherSpec* sowie das *master_secret*. Eine Sitzung kann mehrere Verbindungen besitzen und ist durch die Random-Werte, die MAC- und Sitzungsschlüssel, dem Initialisierungsvektor und den Sequenznummern charakterisiert. [S.798 Eck12]

Kommt es während der Kommunikation zu Fehlern, wird von dem Alert-Protokoll Gebrauch gemacht. Dieses unterscheidet zwischen *warning*- und *fatal*-Fehlern. Tritt ein *fatal*-Fehler auf, muss die Verbindung sofort beendet werden. Daher muss ein neuer Handshake durchgeführt werden. Das Verhalten für *warning*-Fehler ist nicht definiert. Ein Beispiel für einen *fatal*-Fehler ist *decrypt_error*. Dies besagt u.a., dass eine kryptographische Operation während des Handshakes nicht durchgeführt werden konnte. [S.169 Sch14, DR08]

Die zuvor beschriebenen Protokolle setzen nun auf das TLS Record-Protokoll auf und nutzen dieses. Der folgende Ablauf zeigt die Funktionalität des Protokolls nachdem der Handshake erfolgte. Zu Beginn werden die Daten der darüber liegenden Schicht entgegen genommen und in Records fragmentiert. Optional kann danach die ausgehandelte Kompressionsfunktion angewendet werden, um die Recordgröße zu minimieren. Im Anschluss daran wird der MAC-Wert für das Record gebildet. Ggf. muss vor der Verschlüsselung, sofern eine Blockchiffre als Verschlüsselungsfunktion ausgehandelt wurde, ein Padding durchgeführt werden. Damit werden die zu verschlüsselnden Daten an die jeweilige Blocklänge angepasst. Zuletzt werden die Records verschlüsselt. Wie in allen anderen Schichten werden auch hier bestimmte Informationen des verwendeten Protokolls an ein Record angehängt. Darunter fällt die Versionsnummer von TLS, die Länge des darauf folgenden Records sowie der Typ des Records. Beim Typ wird unterschieden zwischen *ChangeCipherSpec-*, *Alert-*, *Handshake-* und *Anwendungsdaten*-Typ. [S.159 Sch14, DR08]

Einer der bekanntesten Angriffe war der im Jahr 1998 vorgestellte Bleichenbacher-Angriff. Er trug maßgeblich an der weiteren Entwicklung von TLS bei. Die Grundidee ist, dass das

pre_master_secret aus der *ClientKeyExchange-Nachricht* mittels einer Side-Channel-Attacke berechnet wird. Genaueres ist [Ble98] zu entnehmen. Ein weiterer bekannter Angriff aus dem Jahre 2014 ist der sogenannte Heartbleed-Angriff. Dies stellte einen recht simplen Buffer Overflow dar, womit der Arbeitsspeicher des Servers ausgelesen werden konnte und somit z.B. die Sitzungsschlüssel preisgegeben wurden. Genaueres unter [Hea14]. [S.180f Sch14]

Mit TLS ist ein sehr bedeutendes und sicheres Verfahren entwickelt worden. Dennoch hat es seine Grenzen. Zum einen ist die Verwendung von starken kryptographischen Verfahren abhängig. Z.B. ist schon länger bekannt, dass für die Hashfunktion MD5 schnell und effizient Kollisionen berechnet werden können [dBB93]. Daher ist bereits in der aktuellen Version 1.2 vermerkt, dass MD5 keine Sicherheit mehr gewährleistet. Eine weitere Grenze zeigt sich, dass durch TLS keine Verbindlichkeit gewährleistet ist. Die MAC-Verfahren sind dafür nicht ausgelegt und weitere Mechanismen gibt es nicht in dem Protokoll die es gewährleisten könnten. Des Weiteren kann ein Angreifer durch Spoofing dem Client falsche Zertifikate vorlegen und somit den vermeidlich geschützten Datenverkehr mitlesen. Ebenso ist ein Man-in-the-Middle-Angriff gut möglich. [S.800f Eck12, DR08]

2.3.2 Netzwerkschicht: IPSEC

IPsec ist eine von der IETF standardisierter Sicherheitsarchitektur für IP-Pakete auf der Netzwerkschicht. Es besteht aus mehreren Komponenten die über einige RFCs definiert sind. Die grundlegende Architektur wird im RFC 4301 spezifiziert. Da IPsec auf Schicht 3 angesiedelt ist, kann es nicht nur zwischen zwei Endsystemen eingesetzt werden, sondern auch zwischen zwei Routern oder auch einem Endsystem und einem Router. Insgesamt kann IPsec die Vertraulichkeit, Authentizität und Integrität von IP-Paketen gewährleisten. Dies geschieht über die Sicherheitsprotokolle Authentication Header (AH) und Encapsulating Security Payload (ESP) die alleine oder in Kombination miteinander verwendet werden können. Darüber hinaus werden mit dem Inter Key Exchange (IKE)-Protokoll im Vorfeld die dafür nötigen Schlüssel und Algorithmen für zwei Systeme ausgehandelt. Das IKE-Protokoll kommt aber nur zum Einsatz wenn in der Security Association Database (SAD) keine entsprechende Security Association (SA) eingetragen ist. Eine SA definiert die Schlüssel und Algorithmen für eine IPsec-Verbindung zwischen zwei Systemen und ist über den Security Parameter Index (SPI) in der SAD eindeutig identifiziert. Ferner wird durch die Security Policy Database (SPD) die Sicherheitsstrategie für IP-Pakete festgelegt. Diese soeben kurz vorgestellten Komponenten und ihre Eigenschaften werden im folgenden genauer erläutert. Abbildung 2.8 gibt einen Einblick über das grobe Zusammenspiel dieser Komponenten. [S.762ff Eck12, KS05]

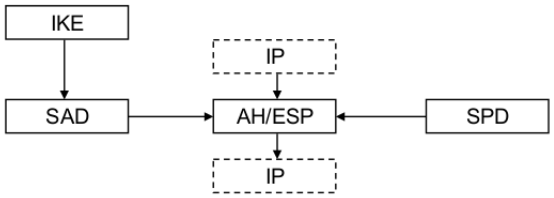

Abbildung 2.8: IPsec Komponenten und ihr grobes Zusammenspiel auf Senderseite [S.88 Sch14]

In IPsec werden zwei unterschiedliche Arten definiert, welche Daten eines IP-Pakets geschützt werden. Auf der einen Seite werden im Transportmodus nur die Daten der darüber liegenden Schicht geschützt. Auf der anderen Seite wird im Tunnelmodus das gesamte IP-Paket geschützt. D.h. IP-Header und die Daten der darüber liegenden Schicht. Der Transportmodus wird allerdings kaum noch in der Praxis verwendet. Daher wird hier die Funktionalität der Sicherheitsprotokolle im Tunnelmodus bezogen auf IPv4 beschrieben. Durch das AH- und ESP-Protokoll wird definiert wie die Daten geschützt werden. [S.764f Eck12]

Die Schlüssel, Verschlüsselungsalgorithmen und weitere kryptographische Parameter für eine IPsec-Verbindung werden als SA bezeichnet. Sie können statisch in die jeweilige SAD des Systems eingetragen oder dynamisch mit dem IKE-Protokoll ausgehandelt werden. Somit kann bspw. das Sicherheitsprotokoll AH oder ESP mit den nötigen Informationen jeweils eine eigene SA zu einer IPsec-Verbindung bilden. Um eine SA eindeutig identifizieren zu können, besitzt sie ein SPI. Damit ein empfangenes Paket - während einer IPsec-Verbindung - behandelt werden kann, dient der SPI und ggf. auch die IP-Zieladresse als Verweis auf den entsprechenden SAD-Eintrag. Somit kann festgestellt werden, welcher Schlüssel und Algorithmus bspw. zur Entschlüsselung des Paketes genutzt werden muss.
Zu der SAD gibt es zusätzlich die SPD. Sie stellt die Sicherheitsstrategien - AH, ESP, oder beide - in einer Datenbank bereit, in der festgelegt ist, wie mit ein- und ausgehenden Paketen umgegangen wird. Dabei kann IPsec auf ein Paket angewendet (*apply*), es ohne IPsec weitergeleitet (*bypass*) oder verworfen werden (*discard*). Wird IPsec auf ein ausgehendes Paket angewendet, wird geprüft, ob in der SAD ein entsprechender Eintrag für AH oder ESP vorliegt. Gibt es keinen Eintrag, muss mit dem IKE-Protokoll ein entsprechender Schlüssel für das weitere Vorgehen ausgehandelt werden. Eingehende Pakete die keinen SAD-Eintrag aufweisen werden verworfen. [S.766ff Eck12, KS05]

Das AH-Protokoll gewährleistet Authentizität, Integrität und optional den Schutz vor Reply-Angriffen der Pakete. Das gesamte IPsec-Paket wird durch einen MAC-Wert authentifiziert und bietet gleichzeitig dessen Integrität. Soll jedoch die Integrität von zusammenhängenden

Paketen gewährleistet werden muss der Schutz vor Reply-Angriffen ebenfalls angewendet werden. Dies geschieht durch eine zusätzliche Sequenznummer die fortlaufend von 0 hochgezählt wird. Wird allerdings die größte Sequenznummer ($2^{62} - 1$) erreicht muss eine neue SA ausgehandelt werden. Diese Nummer, der MAC-Wert, der SPI und weitere Informationen bilden den AH-Header. Der Header wird bei der Bildung des IPsec-Paketes vor dem eigentlichen IP-Paket (innerer IP-Header und Nutzdaten) gesetzt. Vor diesem Konstrukt wird ein äußerer IP-Header angehängt. Im äußeren Header können im Tunnelmodus andere Quell- und Zieladressen stehen als im inneren Header. Dadurch und durch das Einfließen in die MAC-Wert-Berechnung sind die eigentlichen Adressen im inneren Header vor Veränderungen geschützt. Einige Felder, wie z.B. die Time To Live (TTL) oder Header-Prüfsumme, des inneren und äußeren Headers fließen bei der MAC-Wert-Berechnung nicht mit ein, da diese sich durch den Transport der Pakete ändern. Die Verwendung des AH-Protokolls in Verbindung mit NAT ist im Tunnel- und Transportmodus nicht möglich (vgl. [AW04]). Da NAT eine Adressübersetzung vornimmt, ändert sich auch der MAC-Wert im AH-Header. Dies führt dazu, dass das Paket verworfen wird, da der gebildete MAC-Wert nicht mit dem im AH-Header übereinstimmt. [S.771ff Eck12] [S.92ff Sch14, Ken05a]

Das ESP-Protokoll hingegen gewährleistet - wie das AH-Protokoll - die Authentizität, Integrität und den Schutz gegen Reply-Angriffen, auch die Verschlüsselung der Pakete. Aus Performanzgründen kann auch nur die Verschlüsselung eingesetzt werden, wovon aber wegen möglicher Angriffe abgeraten wird. Der Zweck von MAC-Wert und den Sequenznummern ist identisch mit dem im AH-Protokoll. Ein IPsec-Paket im Tunnelmodus besteht aus einem äußeren IP-Header, dem ESP-Header, dem IP-Paket (innerer IP-Header und Nutzdaten) und einem ESP-Trailer. Im ESP-Header sind der SPI, die Sequenznummer und optional ein Initialisierungsvektor angesiedelt. Der Trailer setzt sich aus Padding-Informationen und dem MAC-Wert zusammen. Im Gegensatz zu AH sind nur der ESP-Header, das IP-Paket und die Padding-Informationen authentifiziert. Wohingegen das IP-Paket und die Padding-Informationen verschlüsselt sind. Somit bestehen bzgl. der Authentizität und Integrität keine weiteren Anforderungen an den äußeren IP-Header wie bei dem AH-Protokoll. Die Berechnung des MAC-Wertes ist daher einfacher durchzuführen. Der ESP-Header hingegen darf nicht verschlüsselt werden, da dieser ggf. Informationen für die Entschlüsselung eines Paketes enthält. Die beschriebene NAT-Problematik ist für das ESP-Protokoll ebenfalls gegeben. Allerdings kann dies umgangen werden, indem das IPsec-Paket nochmals in ein UDP-Paket gepackt wird und über den IKE-UDP-Port 500 ausgetauscht wird (vgl. [HSV+05]). [S.775ff Eck12, Ken05b]

Falls für ein ausgehendes Paket keine Schlüssel und Algorithmen in der SAD gespeichert sind, müssen diese mit dem IKE-Protokoll ausgehandelt werden. Es ist im RFC 7296 in der 2. Version spezifiziert und verwendet zum Austausch von Schlüsselinformationen generell das Diffie-Hellman-Verfahren. Nach dem Austausch sind in der SAD auf Sender- und Empfängerseite entsprechende SAs eingetragen. Der Ablauf vom IKE-Protokoll wird in zwei Phasen eingeteilt. In der ersten Phase wird eine eigene IKE-SA ausgehandelt, damit in der zweiten Phase die Informationen für die benötigte IPsec-SA sicher ausgetauscht werden können. Daher werden in der ersten Phase die nötigen Informationen für das Diffie-Hellman-Verfahren für die IKE-SA ausgetauscht. Zu Beginn dieser Phase wird ein sogenanntes Cookie erzeugt, der zum einen gegen DoS-Attacken und zum anderen für die Identifizierung der IKE-SA im weiteren Verlauf dient. Ein Cookie ist in diesem Fall ein Hashwert der von beiden Parteien zunächst erzeugt und an die andere Partei gesendet wird. Jede Partei prüft das Cookie und schickt es an den jeweiligen Sender wieder zurück. Darüber hinaus werden die nötigen Diffie-Hellman Parameter nach dem Senden des Cookies der zweiten Partei von beiden Seiten versendet. Somit kann jede Seite den entsprechenden Schlüssel für die IKE-SA generieren. Im Anschluss müssen sich die Parteien gegenseitig Authentifizieren. Dies kann bspw. mit digitalen Signaturen geschehen. Nach der erfolgreichen Authentifizierung, tauschen beide Parteien in der zweiten Phase die Informationen der IPsec-SA untereinander aus. Diese sind mit dem Schlüssel aus der IKE-SA aus der ersten Phase verschlüsselt. [S.778ff Eck12, KHN+14]

Ein großes Problem ist die Komplexität von IPsec wie es auch in [FS03] analysiert wurde. Dies bringt eine hohe Fehleranfälligkeit in der Implementierung mit sich und kann zu großen Sicherheitslücken führen. Die unübersichtliche Dokumentation des Protokolls ist ebenfalls ein großes Manko und führt zu Verständnisschwierigkeiten. Des Weiteren gibt es zu viele Möglichkeiten der Kombination von Modi und Sicherheitsprotokollen, die das Protokoll unnötig komplex gestalten. Da das AH-Protokoll keine Verschlüsselung bietet und der Transportmodus in der Praxis keine Relevanz findet ist die Verwendung des ESP-Protokolls im Tunnelmodus ausreichend. Es würde zudem das Protokoll deutlich schlanker machen. Zusätzlich ist bei IPsec keine Verbindlichkeit gewährleistet, da keine Signaturen eingesetzt werden. Dennoch ist IPsec zur Zeit die beste und auch bekannteste Möglichkeit auf IP-Ebene zu verschlüsseln. [S.783ff Eck12, FS03]

3 Vorhandene Lösungen

3.1 OpenPGP

3.1.1 Konzept

Phil Zimmermann ist der Erfinder von Pretty Good Privacy (PGP) und stellte 1991 die erste Version vor. Sein Ziel war es die Privatsphäre in den Händen der Menschen zu belassen und diese nicht von der Regierung einschränken zu lassen. Dem folgenden Zitat von Phil Zimmermann ging u.a. ein umstrittener Gesetzesentwurf der US-Regierung voraus, das Hintertüren in Verschlüsselungssoftware vorsah: „PGP empowers people to take their privacy into their own hands. There has been a growing social need for it. That's why I wrote it."[1] Der Entwurf wurde wegen des Protestes von Industrie und Zivilgesellschaft nicht umgesetzt. Während PGP mittlerweile von der Firma Symantec weiterentwickelt wird, entstand 1998 der frei zugängliche OpenPGP-Standard. Dieser basiert auf PGP Version 5.x und ist derzeit im RFC 4880 spezifiziert. Dieser RFC definiert lediglich das Nachrichtenformat. Wie dieses Format z.B. von einem E-Mail-Client zu verwenden ist, wird durch PGP/MIME im RFC 3156 beschrieben. Neben S/MIME gehört auch OpenPGP zu den derzeit einzigen Methoden für die Ende-zu-Ende-Verschlüsselung von E-Mails. Darüber hinaus können auch Dateien mit OpenPGP ver- und entschlüsselt werden. [S.198 Sch14, CDF+07, Zim91]

Die generellen Funktionen von OpenPGP sind die Verschlüsselung und Signierung sowie das Komprimieren und Kodieren von E-Mails bzw. Daten. Genau wie S/MIME setzt OpenPGP die Verschlüsselung durch die Kombination von symmetrischen und asymmetrischen Verfahren um. Mit einem erzeugten Sitzungsschlüssel wird eine E-Mail verschlüsselt und dieser anschließend mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. In Abbildung 3.1 ist der symmetrische Schlüssel als K_M und Bob's öffentlicher RSA-Schlüssel als E_B gekennzeichnet. K_M fließt zunächst in den IDEA-Algorithmus ein und wird anschließend mit E_B verschlüsselt. Vor der Kodierung mit *base64*, werden die verschlüsselte Nachricht und der verschlüsselte symmetrische Schlüssel miteinander verknüpft. Der Empfänger kann nun mit seinem privaten

[1] Phil Zimmermann; Why I wrote PGP; Boulder, Colorado; 1991 (1999 aktualisiert); https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html; zuletzt besucht am 27.07.2016

Schlüssel den Sitzungsschlüssel und damit die E-Mail entschlüsseln. Die digitale Signatur wird durch einem mit dem privaten Schlüssel des Senders verschlüsselten Hashwert der Nachricht repräsentiert. In Abbildung 3.1 wird dieser Ablauf ganz links verdeutlicht. Der Originaltext kann hierbei zunächst gehasht werden und wird im Anschluss mit dem privaten RSA-Schlüssel von Alice verschlüsselt. Der Empfänger entschlüsselt den empfangenen Hashwert mit dem öffentlichen Schlüssel des Senders und bildet ebenfalls einen eigenen Hashwert der empfangenen Nachricht. Die Signatur ist verifiziert, wenn die beiden Werte identisch sind. [S.6ff CDF+07]

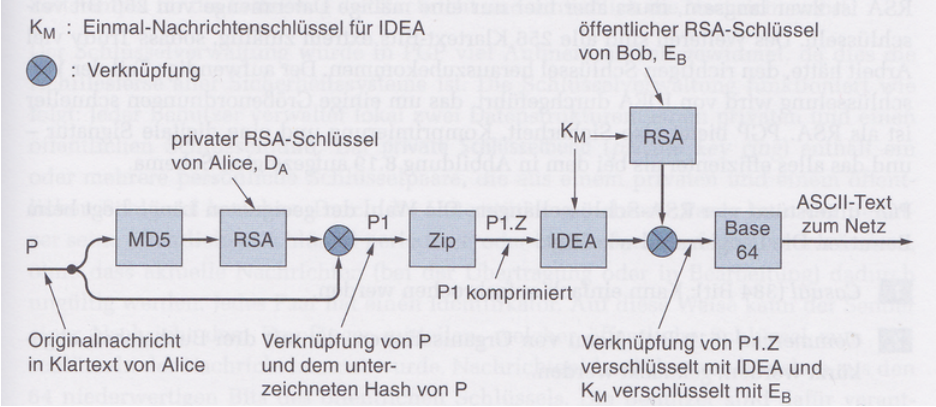

Abbildung 3.1: Signierung und Verschlüsselung einer OpenPGP-Nachricht [S.951 TW12]

Ein OpenPGP Datensatz besteht aus verschiedenen Paketen die wiederum andere Pakete enthalten können. Ein Paket besteht aus einem Header und einem Body. Der Header setzt sich zum einen aus einem sogenannten *Tag* zusammen, der die Art des Paketes bestimmt. Als auch zum anderen aus der Länge der im Body enthaltenen Daten im Header. Abhängig vom *Tag* im Header ist der Inhalt des Bodys definiert. Z.B. sind in dem Pakettyp *Public-Key Encryption Session Key Packet* mit dem *Tag* 1 die Versionsnummer des Pakets, die Schlüssel-ID des öffentlichen Schlüssels, die Nummer des Algorithmus des öffentlichen Schlüssels sowie der verschlüsselte Sitzungsschlüssel als Zeichenkette repräsentiert enthalten. In Abbildung 3.2 ist ein OpenPGP Datensatz abgebildet der eine signierte und verschlüsselte Nachricht darstellt. [S.208f Sch14, CDF+07]

Das *Literal Data Packet* spiegelt die eigentlichen Daten wider. Diese können in Binär- oder Textform vorliegen und werden mit einem *b*, *t* oder *u* im Body gekennzeichnet. *u* bedeutet wie *t*, dass die Daten in Textform vorliegen, diese allerdings in UTF-8 kodiert sind. [S.211 Sch14, CDF+07]

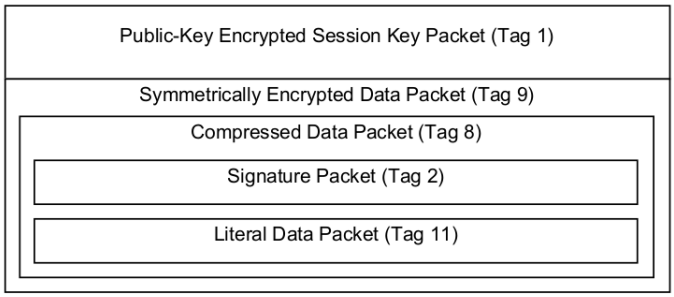

Abbildung 3.2: Beispiel für eine OpenPGP-Paketstruktur [S.216 Sch14]

Im OpenPGP-Standard gibt es einige unterschiedliche Arten von Signaturen, die in dem *Signature Packet* (*Tag* 2) kodiert werden. Es können bspw. das *Literal Data Packet* oder Schlüssel signiert werden. Ebenso ist es möglich Zertifizierungen über die User-ID und dem öffentlichen Schlüssel durch eine Signatur zu erzeugen. In dem hier angegebenen Beispiel wird eine Signatur für ein *Literal Data Packet* erzeugt. Es besitzt im Body deshalb ein Feld für den entsprechenden Typen der Daten für das Binär- oder Textformat. Darüber hinaus besteht der Body aus weiteren Dingen wie der Versionsnummer, dem Zeitstempel, der *Signer Key ID* die den verwendeten Schlüssel zur Erzeugung der Signatur kennzeichnet, dem verwendeten Signieralgorithmus (RSA oder DSA), dem verwendeten Hash-Algorithmus (SHA-1, SHA256 etc.), die Länge des Hashwertes, die ersten beiden Bytes des Hashwertes zur zügigen Erkennung, ob die Signatur gültig ist und der Signatur selbst. Die Signatur wird durch den Hashwert der Daten, dem Datentyp und einem Zeitstempel erzeugt. [S.211ff Sch14, CDF+07]

Nach der Erzeugung des *Signature Packet*, werden dieses und das *Literal Data Packet* typischerweise komprimiert. Die Komprimierung stellt einen optionalen Schritt dar, wird aber dennoch sehr häufig angewendet. Der Body ist sehr schlank aufgebaut und enthält, neben den komprimierten Daten, den verwendeten Komprimierungsalgorithmus. [S.214 Sch14, CDF+07]

Das vorliegende Gesamtpaket wird nun mit einem Sitzungsschlüssel verschlüsselt. Neben dem *Symmetrically Encrypted Data Packet* (*Tag* 9) kann auch das *Symmetrically Encrypted Integrity Protected Data Packet* (*Tag* 18) genutzt werden. Dieses stellt durch einen zusätzlichen Hashwert die Integrität sicher und verhindert somit eine unbemerkte Modifikation der verschlüsselten Daten. Das *Symmetrically Encrypted Data Packet* besitzt im Body lediglich die verschlüsselten Daten bzw. Pakete. Der verwendete Algorithmus und Schlüssel, zur Verschlüsselung der Daten, stehen entweder im darüber liegenden *Public-Key Encrypted Session Key Packet (Tag 1)* oder im *Symmetric-Key Encrypted Session Key Packet (Tag 3)*. Wird keines von diesen beiden Paketen eingesetzt, wird ein Standard-Algorithmus zur Verschlüsselung verwendet. In jedem Fall werden die Daten bzw. Pakete im CFB-Modus verschlüsselt. Der

Initialisierungsvektor ist bei OpenPGP immer gleich 0. Damit aber bei gleichem Sitzungsschlüssel und gleichen Daten nicht der selbe Chiffretext entsteht, muss dies entsprechend gelöst werden (vgl. [S.46 CDF+07]). [S.214f Sch14]

Zuletzt muss der Sitzungsschlüssel mit dem öffentlichen Schlüssel des Empfängers verschlüsselt werden. Die im Body des *Public-Key Encrypted Session Key Packet* (*Tag* 1) enthaltenen Informationen sind wie oben beschrieben. Im zuletzt angegebenen Feld des Bodys, das den Sitzungsschlüssel enthält, sind die noch fehlenden wichtigen Informationen zu dem Sitzungsschlüssel enthalten. Zum einen der verwendete Algorithmus für den Sitzungsschlüssel und zum anderen eine Prüfsumme des Sitzungsschlüssel. Diese beiden Werte werden konkateniert und als PKCS#1 kodiert. [S.215f Sch14, CDF+07]

Somit ist der Vorgang zur Erstellung eines OpenPGP-Datensatzes des angegebenen Beispiels beendet. Um nun die Interoperabilität zu gewährleisten, bietet OpenPGP eine Radix-64-Konvertierung an. Diese besteht aus der base64-Kodierung - wie auch bei MIME - und einer Prüfsumme. Somit ist eine korrekte Darstellung der Daten bspw. in Form einer E-Mail gegeben. [216f Sch14, CDF+07]

In PGP müssen die u.a. verwendeten asymmetrischen Schlüsselpaare verwaltet werden. Dies geschieht in den sogenannten *Keyrings*. Dafür gibt es einen separaten *Private-* und *Public-Keyring*. Der *Private-Keyring* stellt im Endeffekt eine Liste aller vom Anwender erzeugten Schlüsselpaare dar. Zum Schutz wird der private Schlüssel im Gegensatz zum öffentlichen Schlüssel verschlüsselt abgespeichert. Dazu muss der Anwender ein Passwort verwenden. Dieses wird durch einen Hashwert repräsentiert der anschließend den privaten Schlüssel verschlüsselt. Sobald der private Schlüssel benötigt wird, muss der Anwender das Passwort eingeben, woraufhin der Hashwert gebildet wird und den Schlüssel entschlüsselt. Ein Eintrag im *Private-Keyring* besteht u.a. aus dem öffentlichen und privaten Schlüssel sowie dem Zeitstempel der Erzeugung des Paares, der Schlüssel- und Benutzer-ID. Um die öffentlichen Schlüssel anderer Anwender zu verwalten, wird ein eigener *Public-Keyring* verwaltet. Dieser stellt ebenfalls eine Liste aller Schlüssel dar. Ein Eintrag enthält u.a. einen Zeitstempel der Eintragung des öffentlichen Schlüssels, den Schlüssel selbst sowie die Schlüsselbesitzer- und Schlüssel-ID. [S.814f Eck12]

Um die Glaubwürdigkeit der öffentlichen Schlüssel von PGP zu bewerten, wird das Vertrauensmodell *Web of Trust* eingesetzt. Dieses besagt, dass jeder Anwender ein Zertifikat für einen öffentlichen Schlüssel erzeugen kann, indem dieser signiert wird. Die Signatur besagt, dass der Schlüssel zu dem angegebenen Besitzer gehört und ihm vertraut werden kann. Je mehr Signaturen an einem öffentlichen Schlüssel haften, desto vertrauenswürdiger ist er. Dieser Vertrauensgrad wird in PGP durch das Key Legitimation Field (KLF) repräsentiert und ist

ebenfalls zu jedem öffentlichen Schlüssel im *Public-Keyring* verzeichnet. Dieser Wert wird auf Basis der bisher erhaltenen Zertifikate berechnet. Daher kann man bei der Eintragung eines öffentlichen Schlüssels in den eigenen *Public-Keyring* eine Vertrauenseinschätzung abgeben womit dieser zertifiziert wird. Die Einschätzung besitzt fünf Stufen wie z.B. der Benutzer ist unbekannt (1), geringfügig vertrauenswürdig (3) oder ultimativ vertrauenswürdig (5). Damit die öffentlichen Schlüssel an andere Anwender verteilt werden können, gibt es die Möglichkeit auf sogenannte öffentlich zugängliche Schlüsselserver zurückzugreifen. Ebenso können die Schlüssel gewöhnlich als E-Mail Anhang verschickt oder auf Internetseiten veröffentlicht werden. [S.817f Eck12, Inc99]

3.1.2 Verwendung

PGP eignet sich sowohl für die E-Mail- als auch für die Datenverschlüsselung. Mit GnuPG gibt es ein Werkzeug für die gängigsten Betriebssysteme wie Windows, Linux und Mac OS. Damit lassen sich ausschließlich Dateien ver- und entschlüsseln sowie signieren und verifizieren. Darüber hinaus wird z.B. bei der Linux Distribution Debian GnuPG zum Verifizieren von Paketen genutzt. Damit ist es möglich, Pakete aus sicheren Quellen zu installieren, sofern dem öffentlichen Schlüssel der Quelle vertraut wird. Standardmäßig werden einige öffentliche Schlüssel bei der Installation des Systems mitgegeben. Anhand dieser kann ein z.B. aus dem Internet heruntergeladenes Paket verifiziert und vertraut werden. Zur Verschlüsselung von E-Mails gibt es für die gängigsten E-Mail-Clients, wie Thunderbird oder MS Outlook, Plugins die installiert werden können. Zu nennen sind hier Ggp4win mit der Komponente GpgOL für Outlook oder Enigmail für Thunderbird. Mittlerweile bieten auch einige Mailanbieter wie GMX eine Verschlüsselung über eine eigene OpenPGP-Implementierung an. Dies hat den Vorteil auch im Webbrowser die E-Mails verschlüsseln zu können. Dafür wird allerdings noch ein zusätzliches Plugin für den Webbrowser benötigt. [gCG, Asl, Gmbb, Pro, Gmba]

3.1.3 Vor- und Nachteile

OpenPGP gilt heute als ein sehr guter und sicherer Standard zur E-Mail- und Datenverschlüsselung. Die im Standard definierten Algorithmen gelten nach wie vor als sehr stark, vertrauenswürdig und zukunftsgerichtet. Zwar gibt es immer noch schwache Algorithmen in der Spezifikation wie bspw. den MD5-Algorithmus, aber dieser wird bereits im RFC als überholt gekennzeichnet. OpenPGP hat seid jeher eine loyale Fangemeinde die ihm auch weiterhin treu bleiben wird. Das Einsatzgebiet erstreckt sich daher vor allem im privaten

Umfeld und in kleineren Unternehmen. Es stellt zu S/MIME eine gute und in der Verwendung etwas unkompliziertere Alternative dar. [S.669f Sch13a] [S.818f Eck12, CDF+07]

Die Kehrseite von OpenPGP besitzt allerdings viele Argumente gegen seinen Einsatz. Das Hauptproblem ist dabei die eher aufwendige Bedienung bzw. schlechten Bedienoberflächen eines OpenPGP-Werkzeuges. Der Mensch ist bequem und möchte sich nicht mit zeitintensiven Tutorials zu einem Werkzeug auseinandersetzen. Eine sichere Ende-zu-Ende-Verschlüsselung der E-Mails soll einfach funktionieren, ohne dass man sich mit der Materie auseinander setzen muss. [Gre14] äußert sich kritisch über die sehr unhandlichen Schlüssel und dem transparenten Schlüsselaustausch. Die Handhabung der Schlüssel ist weniger für den Menschen konzipiert, was das Problem der Identifizierung eines erhaltenen öffentlichen Schlüssel mit sich bringt. Daher ist es schwierig, ob man einem erhaltenen Schlüssel tatsächlich vertrauen kann, oder ob es sich nicht um einen Man-In-The-Middle-Angriff handelt. Der Einsatz eines hierarchischen Vertrauensmodells, wie bei den X.509-Zertifikaten, ist, bezogen auf den Schlüsselaustausch, schwer bis gar nicht umsetzbar. Dies liegt an dem eigenen OpenPGP-Zertifikatsformat. Dies besitzt nur bestimmte Felder und ist mit den gegebenen Standards, wie X.509 oder dem PKCS, nicht kompatibel. Aus der technischen Sicht betrachtet ist OpenPGP, wie bereits erwähnt, gut aufgestellt. Allerdings liegt es immer in der Hand der Entwickler wie sicher ein nach einem Standard implementiertes Werkzeug ist. So ist es häufig immer noch möglich für die Generierung eines RSA-Schlüsselpaares eine nicht ausreichend lange Schlüssellänge zu wählen. So ist laut [S.15 BSI16] gegenwärtig die Verwendung eines mindestens 2000 Bit langen Schlüssels zu empfehlen. Nach dem Jahr 2016 sollte sogar mindestens ein 3000 Bit langer Schlüssel eingesetzt werden. In [S.82 CDF+07] ist immer noch eine Länge von mindestens 1024 Bit empfohlen. Weiterhin unterliegt OpenPGP nicht dem Perfect Forward Secrecy (PFS) Prinzip. D.h. sobald der private Schlüssel bekannt ist, können damit rückwirkend alle - mit dem dazugehörigen öffentlichen Schlüssel - verschlüsselten E-Mails entschlüsselt werden. Dies entwickelt sich zu einem immer häufiger diskutiertem Thema, da die NSA schon lange viele verschlüsselte Daten aus dem Netzwerkverkehr sammelt und hofft diese in Zukunft entschlüsseln zu können. Es ist daher ein ernsthaftes Problem, wenn die verwendete Verschlüsselungsmethode nicht unter das PFS-Prinzip fällt. [S.669f Sch13a, Gre14, Sch13b]

Schlussendlich wird OpenPGP von vielen Experten als nicht mehr zeitgemäßer Verschlüsselungsstandard angesehen. Es wird daher von einem Neustart der Ende-zu-Ende-Verschlüsselung gesprochen. Es ist vor allem nötig eine Technologie zu entwickeln die eine sichere E-Mail-Verschlüsselung intuitiver und wesentlich unkomplizierter in der Benutzung gestaltet. Teilweise wird dies bereits durch einige Messanger-Dienste wie *TextSecure* oder *Signal* umgesetzt. Daher wird gefordert, dass die bisherigen Ansätze an die heutigen Verhältnisse anzupassen

sind. D.h. aus dem entwickelten OpenPGP-Standard sollten neue Standards, die den gegenwärtigen Kritikpunkten entsprechen, entwickelt werden. [Mar15, Gre14, Jä15]

3.2 S/MIME

3.2.1 Konzept

Secure/Multipurpose Internet Mail Extension (S/MIME) ist heutzutage der de facto Standard für die Ende-zu-Ende Verschlüsselung und zum Signierung von E-Mails in professionellen Umgebungen. S/MIME ist in der aktuellsten Version 3.2 im RFC 5751 spezifiziert. Neben S/MIME hat die Firma *RSA Data Security Inc.* auch den sogenannten Public Key Cryptography Standard (PKCS) entwickelt. Dieser umfasst mehrere Standards für die asymmetrische Kryptographie. S/MIME stützte sich zu Beginn der Entstehung überwiegend auf den PKCS#7. Dieser definiert ein einheitliches Datenformat für verschlüsselte und signierte Daten. Cryptographic Message Syntax (CMS) wurde aus diesem Standard abgeleitet und ist ab S/MIME Version 3 Standard. Es wird nun ein kleiner Einblick in MIME gegeben und anschließend etwas ausführlicher über S/MIME diskutiert. [S.225f Sch14] [S.668 Sch13a, Kal98, Ram99, Hou09]

Seit den 80er Jahren wurde eine E-Mail grundsätzlich in Header und Body aufgeteilt. Im Header werden die Metadaten von Empfänger und Absender geführt, wie z.B. Absender-, Empfängeradresse, Datum, etc.. Im Body steht die eigentliche Nachricht im ASCII-Format. Dies brachte im Laufe der Zeit einige Probleme mit sich, sodass bspw. nicht ASCII-Nachrichten (z.B. Dateianhänge in Binärformat) konvertiert werden mussten. Die Konvertierung war allerdings ein komplexes Unterfangen. Daher entschied man sich einen neuen Format-Standard für E-Mails zu definieren: Multipurpose Internet Mail Extension (MIME). Dieser ist in den RFCs 2045, 2046, 2047, 2049 sowie 4289 und 6838 definiert. Die wichtigsten Erweiterungen von MIME umfassen zusätzliche Metadaten zur Beschreibung des E-Mail Inhaltes und eine einheitliche Codierung zum Transfer einer E-Mail. Die wichtigsten neuen Metadaten sind Content-Type und Content-Transfer-Encoding. Der Content-Type drückt aus um welche Art von E-Mail Inhalt es sich handelt. Mit *multipart/mixed* bspw. beschreibt man eine E-Mail die aus mehreren Teilen, sogenannten MIME-Entitäten (z.B. einem Textteil und einem Bildanhang), besteht. Content-Transfer-Encoding drückt aus um welchen Algorithmus zur Transfercodierung es sich handelt. Je nach Eigenschaft des E-Mail Inhaltes kann ein solches Verfahren gewählt werden. Z.B. wird für Binärdaten die base64-Kodierung eingesetzt. Diese Neuerung hat den Zweck, dass eine E-Mail keine Manipulation durch das E-Mail-System erfährt. [S.226ff Sch14] [S.807 Eck12, Res08]

Nr.	Content-Type	CMS-Objekt	Dateiendung	Beschreibung
1	application/pkcs7-mime	enveloped-data	.p7m	Verschlüsselte S/MIME-Entität
2	application/pkcs7-mime	signed data	.p7m	Signierte S/MIME-Entität
3	application/pkcs7-signature	-	.p7s	2. Teil des Content-Type multipart/signed
4	multipart/signed	-	-	Signierte Nachricht aus zwei Teilen: 1. Nachricht 2. Signierte Nachricht

Tabelle 3.1: S/MIME Content-Type Beispiele

S/MIME hat zur Aufgabe das zuvor erläuterte MIME, um die Sicherheitsaspekte der Verschlüsselung und Signierung zu erweitern. Somit können die MIME-Entitäten entsprechend geschützt werden. Während es sich bei der Verschlüsselung um ein hybrides Verschlüsselungsverfahren handelt, werden bei der Signierung - neben den asymmetrischen Verschlüsselungsaspekten - auch kryptographische Hashfunktionen eingesetzt. Es gibt einige Varianten zur Verschlüsselung und/oder Signierung von E-Mails, wovon vier davon nun grob erläutert werden. Details sind aus [RT10] zu entnehmen. [S.230ff Sch14]

Verschlüsseln

Das alleinige Verschlüsseln einer Nachricht und ohne die Verwendung einer Signatur gewährleistet lediglich die Geheimhaltung der Daten. Die Integrität kann nicht gegeben werden, da bspw. ein verschlüsselter Abschnitt der Nachricht ausgetauscht werden kann, ohne dass dem Empfänger dies auffällt. Der erste Schritt zur Verschlüsselung ist das Vorbereiten der Nachricht durch Kanonisierung und Transfercodierung. Anschließend generiert S/MIME einen Sitzungsschlüssel nach einem bestimmten symmetrischen Verschlüsselungsverfahren. Dieser wird durch den öffentlichen Schlüssel des Empfängers verschlüsselt. Alle nötigen Informationen wie z.B. der verschlüsselte symmetrische Schlüssel, der verwendete Algorithmus etc. werden in ein CMS-Objekt vom Typ *enveloped-data* verpackt. Dieses Objekt umfasst alle relevanten Content-Informationen und wird in ein *application/pkcs7-mime* Content-Type-Objekt integriert (siehe Tabelle 3.1 Nr. 1). Der Empfänger kann nun den symmetrischen Schlüssel mit seinem privaten Schlüssel entschlüsseln und die Nachricht lesbar machen. [S.235ff Sch14, RT10]

Signieren

Unter S/MIME gibt es zwei unterschiedliche Wege eine Nachricht zu signieren. Die eine Variante besteht aus einem und die andere aus zwei Datenteilen. Zu Beginn wird die erste Variante beschrieben. Die Bildung der Signatur ist in beiden Varianten dieselbe. Anfangs muss auch hier die Nachricht kanonisiert und für den Transfer entsprechend codiert werden. Danach wird eine Hashfunktion gewählt und mit ihr ein Hashwert für die zu signierende Nachricht erzeugt. Im Anschluss wird dieser Hashwert mit dem privaten Schlüssel des Senders verschlüsselt und entspricht somit der Signatur. Aus der Signatur wird ein CMS-Objekt vom Typ *signed data* mit allen nötigen Informationen erzeugt. Die Content-Informationen werden auch hier in ein *application/pkcs7-mime* Content-Type-Objekt integriert (siehe Tabelle 3.1 Nr. 2). Erhält der Empfänger die Signatur, kann er sie mit dem öffentlichen Schlüssel des Senders verifizieren. [S.237ff Sch14][S.392ff Eck12, RT10]

Die zweiteilige Variante besteht im ersten Teil aus der Nachricht als MIME-Entität. Der zweite Teil stellt die Signatur dar, die wie in der ersten Variante gebildet wird. Die Signatur wird auch hier in ein CMS-Objekt verpackt und zusätzlich transportcodiert. Anschließend wird das Objekt in das Content-Type-Objekt *application/pkcs7-signature* integriert. Die Entität und das Objekt werden anschließend in den Content-Type *multipart/signed* verpackt (siehe Tabelle 3.1 Nr. 3 bzw. 4) und besitzt u.a. den Parameter *micalg*. Dieser repräsentiert die verwendete kryptographische Hashfunktion (SHA-1, SHA-224 etc.). Abbildung 3.3 zeigt ein Beispiel. Die

```
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

Abbildung 3.3: S/MIME multipart/signed Beispiel [S.29 RT10]

erste Variante zur Bildung einer Signatur bringt das Problem mit sich, dass ein Empfänger ohne einen S/MIME-fähigen Client die Nachricht nicht lesen kann. Da diese in dem vom Sender

erzeugten S/MIME-typischen *application-pkcs7-mime*-Objekt verpackt ist. Die zweite Variante hingegen lässt zu, dass die Nachricht gelesen werden kann, auch wenn kein S/MIME-fähiger Client genutzt wird. Allerdings muss zur Verifikation der Signatur S/MIME vorhanden sein. Ein Nachteil an der zweiten Variante ist allerdings, dass die Integrität und Authentizität durch E-Mail-Gateways zunichte gemacht werden können. Dies kommt häufig vor, da die MIME-Struktur derart verändert wird, dass die Signatur unbrauchbar gemacht wird. Zum einen ist die erste Variante zu bevorzugen, wenn ein Gateway auf dem Weg zum Empfänger liegt und zum anderen die Anforderungen an Authentizität und Integrität gewährleistet sein müssen. [RT10]

Verschlüsseln und Signieren

Durch die Eigenschaft, dass bei den vorgestellten Varianten immer neue MIME-Entitäten entstehen (die wiederum MIME-Entitäten enthalten), ist eine Verschachtelung beliebig möglich (vgl. Abbildung 3.4). D.h. es liegt in der Hand der Nutzer, ob zuerst signiert und dann verschlüsselt wird, oder umgekehrt. Es gibt für beide Möglichkeiten Vor- sowie Nachteile. Wurde bspw. eine Nachricht zuerst verschlüsselt und dann signiert, ergibt sich zum einen, dass die verschlüsselte Nachricht definitiv unverändert ist. Zum anderen aber kann keinerlei Beziehung zwischen Unterzeichner und entschlüsselter Nachricht hergestellt werden. Daher muss man immer abwägen in welchem Einsatzgebiet S/MIME verwendet wird. Ein allgemeiner Nachteil ist allerdings, dass durch die beliebige Verschachtelung ein Empfänger in der Lage sein muss diese aufzulösen. [S.237ff Sch14, RT10]

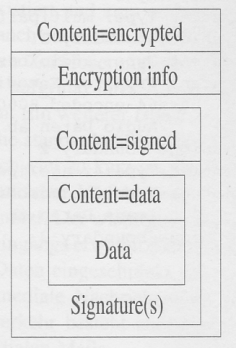

Abbildung 3.4: S/MIME Verschachtelung [S.808 Eck12]

3.2.2 Verwendung

Im Gegensatz zu PGP können mit S/MIME nur E-Mails ver- und entschlüsselt sowie signiert werden. Um dies tun zu können, benötigt man allerdings ein X.509-Zertifikat. Es beglaubigt die Zuordnung eines öffentlichen Schlüssels zu einem Anwender. Die Zertifikate basieren gegenüber PGP auf das hierarchische Vertrauensmodell und werden von einer Certificate Authority (CA) ausgestellt. Zertifikate können kostenpflichtig oder kostenfrei von verschiedenen Anbietern im Internet ausgestellt werden[2]. X.509-Zertifikate werden in vier Klassen (0-3) unterteilt, die die Art der Authentifizierung eines Anwenders und die Vertrauenswürdigkeit eines Zertifikates kategorisiert. Bei *Class* 1 Zertifikaten muss der Anwender den Erhalt der E-Mail vom Anbieter bestätigen und beweist somit den Zugriff auf die E-Mail Adresse. Durch dieses Verfahren der Identifizierung haben die Zertifikate eine geringe Vertrauenswürdigkeit. *Class* 3 Zertifikate hingegen erfordern zusätzlich eine persönliche Identifizierung durch einen gültigen Ausweis des Anwenders beim Anbieter. Sie sind somit die am vertrauenswürdigsten Zertifikate von allen. S/MIME ist in den gängigsten E-Mail Clients wie Thunderbird oder MS Outlook bereits integriert. Nachdem ein Zertifikat beantragt und die Identifikation erfolgt ist, wird es i.d.R. auf dem Computer des Anwenders generiert. Das Zertifikat wird dabei als PKCS#12-Datei abgespeichert. PKCS#12 definiert das Format für die passwortgeschützte Speicherung des privaten Schlüssels sowie des zugehörigen Zertifikates. Im Anschluss muss das Zertifikat in den E-Mail Client importiert werden. Zuletzt muss bei den E-Mail Konteneinstellungen das von S/MIME zu verwendende importiere Zertifikat angegeben werden. Über eine Schaltfläche im E-Mail Client können verfasste E-Mails problemlos verschlüsselt und/oder signiert werden. Nach dem Erhalt einer signierten E-Mail wird das Zertifikat des Absenders automatisch - sofern das Root-Zertifikat der CA dem E-Mail Client bekannt ist - in den Zertifikatsspeicher gespeichert. Andernfalls muss das Root-Zertifikat händisch importiert werden. [Ess14, Gmb10]

3.2.3 Vor- und Nachteile

S/MIME ist genau wie PGP ein sicheres und bewährtes Konzept zur Ende-zu-Ende-Verschlüsselung und Signierung von E-Mails. Die im RFC angegebenen kryptographischen Algorithmen sind wie bei PGP stark und zukunftsorientiert. Mit Ausnahme von einigen schwächeren Algorithmen wie MD5 die nicht mehr verwendet werden sollten, aber noch aus Kompatibilitätsgründen dort aufgeführt sind. Des Weiteren ist es für den Anwender recht angenehm S/MIME zu nutzen, da es in den gängigsten E-Mail Clients bereits vorinstalliert ist. Ebenso gibt es bereits einige Anbieter die kostenfreie X.509 Zertifikate anbieten, womit die Hürde zur Verwendung von

[2] vgl. https://www.cacert.org/ oder https://www.comodo.com/; zuletzt besucht am 27.07.2016

S/MIME zumindest etwas herab gesetzt wird. Zudem ist theoretisch eine hohe Vertraulichkeit durch die persönliche Identifizierung von *Class* 3 Zertifikaten gegeben. Ein weiterer Vorteil von S/MIME ist die Verwendung von Smartcards. [S.668 Sch13a]

Dennoch wird die Benutzerfreundlichkeit von S/MIME kritisiert. Dies liegt daran, dass es speziell mit Thunderbird Interoperabilitätsprobleme zu anderen E-Mail Clients gibt. Z.B. werden von MS Outlook stammende E-Mails bei der Weiterleitung mit Thunderbird in eine Datei verpackt und in den Anhang gehängt.

Weiterhin gibt es eine starke Abhängigkeit zu der CA. Dieser muss bedingungslos vertraut werden, dass die Verifizierung der Identität unter strengen Auflagen durchgeführt wurde. Zusätzlich muss man der CA vertrauen, dass angemessen lange Schlüssel generiert werden (vgl. [BSI16]). Dies setzt auch voraus, dass der Anwender ein Zertifikat beantragt hat, was wiederum eine gewisse Hürde darstellt. Zusätzlich werden derzeit kostenfreie Zertifikate - die gerade für den privaten Anwender interessant sind - mit einer Gültigkeitsdauer von einem Jahr ausgestellt. Dies birgt das Problem, dass abgelaufene Zertifikate weiterhin im Zertifikatsspeicher verbleiben müssen, damit ältere E-Mails weiterhin entschlüsselt werden können. Aus kryptographischer Sicht stellt auch hier das PFS-Prinzip ein Problem dar. Wurde der private Schlüssel entwendet können nachträglich die gesamten abgehörten E-Mails entschlüsselt werden. [Ess14]

Letztlich unterscheiden sich PGP und S/MIME nur geringfügig. Der größte Unterschied ist das Vertrauensmodell, welches jeweils seine Vor- und Nachteile besitzt. Auch deshalb gilt für S/MIME, dass die bisherigen Erfahrungen genutzt werden sollten, um sichere und benutzerfreundlichere neue Standards für die E-Mail Verschlüsselung zu entwickeln.

3.3 De-Mail

3.3.1 Konzept

De-Mail ist ein u.a. vom Bundesministerium des Innern initiierter Dienst zur sicheren und verbindlichen Übermittlung von Nachrichten und Dokumenten. Um diesen Dienst rechtlich zu sichern und Vorgaben an Betreiber und Anwender zu machen, trat am 03.05.2011 das sogenannte De-Mail-Gesetz in Kraft. Unter dem Slogan: „So einfach wie E-Mail, so sicher wie Papierpost“[3] können nun Unternehmen, Behörden und Bürger in Deutschland Dokumente und Nachrichten untereinander verschlüsselt und bequem austauschen. Neben dem Gesetz

[3] Bundesministerium des Innern; Informationsbroschüre De-Mail; Berlin; März 2012; https://www.bmi.bund.de/SharedDocs/Downloads/DE/Themen/OED_Verwaltung/Informationsgesellschaft/de_mail.pdf; zuletzt besucht am 27.07.2016

bilden die sogenannten Technischen Richtlinien des BSI die Grundlage für diesen Dienst. Seid 2012 gibt es bereits diverse Anbieter von De-Mail[4] [fSidI14]

De-Mail bietet den Nutzern einen Dienst zum Versenden und Empfangen von E-Mails (vgl. §1 Abs. (2) Satz 1 De-MailG) mit weiteren Eigenschaften die ihr bisher fehlten. Zum einen ist die Identität von Sender und Empfänger nachweislich gesichert und kann nicht gefälscht werden. Zum anderen werden die E-Mails auf den Servern ausschließlich verschlüsselt gespeichert und über eine Transportverschlüsselung durch einen sicheren Kanal übermittelt. Ebenfalls ist sichergestellt, dass die E-Mails auch tatsächlich ankommen und nicht verloren gehen. Darüber hinaus können E-Mails Ende-zu-Ende verschlüsselt werden. Dies ist allerdings kein angebotener Dienst von De-Mail und benötigt die Verwendung von zusätzlicher Software wie bspw. S/MIME oder PGP/OpenPGP. Um den Austausch der öffentlichen Schlüssel zu vereinfachen, kann der Verzeichnisdienst des De-Mail-Dienstanbieter (DMDA) genutzt werden. Dieser muss bereitgestellt werden (vgl. §1 Abs. (2) Satz 1 De-MailG) und kann zusätzlich dazu genutzt werden persönliche Kontaktdaten eines Nutzers zu veröffentlichen. Somit kann ein Sender z.B. die De-Mail-Adresse des Empfängers problemlos finden. Um Dokumente sicher aufbewahren zu können, kann ein DMDA zusätzlich einen Dokumentenablagedienst anbieten (vgl. §1 Abs. (2) Satz 1 De-MailG). In diesem persönlichen Safe werden die Dokumente verschlüsselt bei dem Anbieter aufbewahrt. Erst durch das Abrufen des Inhabers werden die Dokumente entschlüsselt. [fSidI14, Bun11]

Die Kernfunktionalität von De-Mail ist der sogenannte Postfach- und Versanddienst. Das Ziel dieses Dienstes ist die Zustellung von Nachrichten und Dokumenten im Internet so sicher, einfach und verbindlich zu machen, wie es die heutige Papierpost bereitstellt. E-Mails innerhalb von De-Mail können allerdings nur von Nutzern mit einer De-Mail-Adresse untereinander ausgetauscht werden. Nachrichten von anderen nicht De-Mail-Adressen können nicht empfangen werden. Die funktionalen Anforderungen des Postfach- und Versanddienst sind in [fSidI16] beschrieben. In Abbildung 3.5 ist die Architektur des Postfach- und Versanddienstes angegeben.

Der Postfachdienst stellt für den Anwender das Versenden, Empfangen sowie Verwalten von E-Mails bereit. Darüber hinaus wird die Integrität einer Nachricht durch eine Prüfsumme oder einer qualifizierten Elektronischen Signatur beim DMDA des Absenders gesichert. Diese wird beim DMDA des Empfängers nach Empfang der Nachricht entsprechend geprüft. Ebenfalls werden die Metadaten im Postfachdienst validiert und ergänzt. Damit dem Empfänger keine Viren geschickt werden, wird die E-Mail auf diese geprüft. Dies bedingt eine kurzzeitige

[4] Bundesministerium des Innern; Gesetzlich geregeltes Sicherheitsniveau; 2016; http://www.cio.bund.de/Web/DE/Innovative-Vorhaben/De-Mail/Gesetzlich-geregeltes-Sicherheitsniveau/gesetzlich_geregeltes_sicherheitsniveau_node.html; zuletzt besucht am 27.07.2016

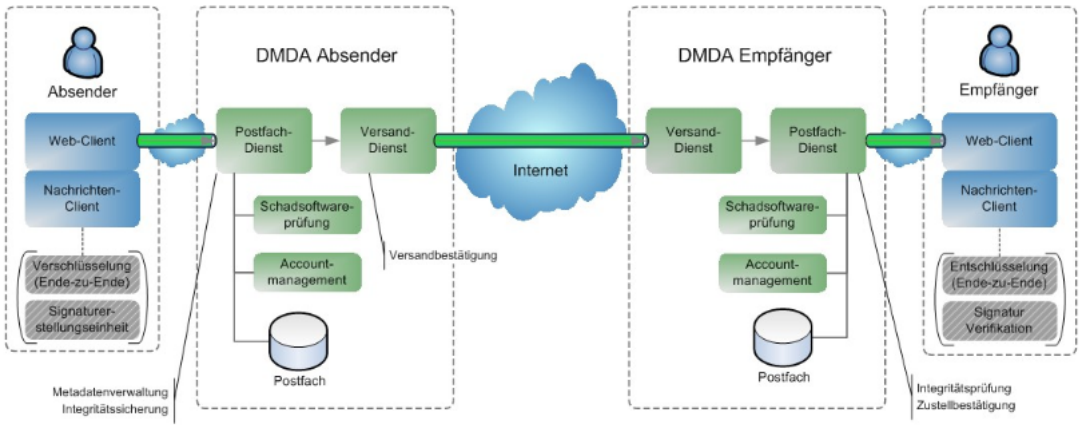

Abbildung 3.5: Architektur Postfach- und Versanddienst [S.6 fSidI16]

Entschlüsselung der Transportverschlüsselung. Die Eingangsbestätigung ist eine zusätzliche Option die gewählt werden kann. Der Postfachdienst des Empfängers bestätigt somit den Erhalt der E-Mail, womit der Sender einen eindeutigen Nachweis darüber besitzt (vgl. §5 Abs. (8) De-MailG). Der Versanddienst hingegen ist für die verbindliche Übermittlung von E-Mails zwischen zwei DMDAs zuständig. Dabei kann der Sender eine Versandbestätigung vom eigenen DMDA verlangen die ihm beweist, dass die E-Mail versendet wurde (vgl. §5 Abs. (7) De-MailG). [fSidI14]

Zur Authentifizierung bietet De-Mail zwei Anmeldeverfahren an, um einen Anwender eindeutig zu identifizieren. Auf der einen Seite kann ein Anwender eine Ein-Faktor-Authentifizierung und auf der anderen eine Zwei-Faktor-Authentifizierung durchführen. Die Ein-Faktor-Authentifizierung stellt ein *normales* Sicherheitsniveau unter De-Mail dar und wird durch die sogenannte Authentifizierung durch Wissen realisiert. Dabei muss der Anwender lediglich Benutzername und Passwort bei der Anmeldung an das De-Mail-Konto angeben. Ein *hohes* Sicherheitsniveau wird durch die Zwei-Faktor-Authentifizierung gewährleistet und durch die Authentifizierung durch Wissen und Besitz umgesetzt. Neben dem Benutzernamen und dem Passwort wird ein sogenannter *Token* benötigt. Dieser kann bspw. eine Chipkarte (z.B. der neue Personalausweis mit eID-Funktion) oder auch ein USB-Stick mit entsprechender Authentifizierungsfunktion sein. Des Weiteren muss ein DMDA mindestens zwei unterschiedliche Tokens unterstützen unter denen der Anwender zur Authentifizierung frei wählen kann. Mindestens muss jedoch das Zwei-Faktor-Authentifizierungsverfahren durch den neuen Personalausweis mit eID-Funktion angeboten werden (vgl. §4 Abs. (2) De-MailG). Neben den bereits beschriebenen Versandarten *Eingangsbestätigung* und *Versandbestätigung* kann ein Anwender zusätzliche

Versandoptionen wählen, sofern er sich mit der Zwei-Faktor-Authentifizierung angemeldet hat. Der Anwender kann nach dem Versenden einer Nachricht die Optionen *Persönlich* oder *Absender-bestätigt* wählen. Mit der *Persönlich*-Option darf der Empfänger die Nachricht nur lesen, wenn er sich ebenfalls mit einer Zwei-Faktor-Authentifizierung angemeldet hat. Die *Absender-bestätigt*-Option hingegen bestätigt dem Empfänger, dass der Sender sich mit einer Zwei-Faktor-Authentifizierung angemeldet hat. [fSidI14]

Ein zukünftiger DMDA, der die De-Mail-Dienste zur Verfügung stellen will, muss „sich auf schriftlichen Antrag von der zuständigen Behörde akkreditieren lassen."[5]. Die zuständige Behörde ist in diesem Fall das BSI (vgl. §2 De-MailG). Dafür wurden in §18 strenge Anforderungen festgehalten die der zukünftige DMDA erbringen muss, damit ein hoher Sicherheitsstandard gewährleistet sein kann. Diese können in vier Bereiche eingeteilt werden: Sicherheit, Funktionalität, Interoperabilität und Datenschutz. Für die Prüfung von Sicherheit, Funktionalität und Interoperabilität ist das BSI verantwortlich. Hierbei erstellen die vom BSI anerkannten und unabhängigen Prüfstellen, Auditoren und Personen Prüfberichte. Die Berichte werden im Anschluss von unabhängigen und vom BSI zugelassenen IT-Sicherheitsdienstleister validiert und erstellen sogenannte *Testate*. Die *Testate* weisen nach, dass der zukünftige DMDA die Anforderungen in den drei Bereichen erbringt. Für den vierten Bereich ist hingegen die Bundesbeauftragte für den Datenschutz und die Informationsfreiheit (BfDI) verantwortlich. Diese prüft ob die Anforderungen an den Datenschutz durch den zukünftigen DMDA erbracht werden und stellt bei positiver Prüfung ein Zertifikat aus. Mit den *Testaten*, dem Zertifikat und Nachweise über die Zuverlässigkeit, Fachkunde und Deckungsvorsoge (vgl. §18 Abs. (1) Satz 1 und Abs. (3) Satz 1 sowie §18 Abs. (1) Satz 2 und Abs. (3) Satz 2) kann nun das BSI den Anbieter zulassen. Erst nach dieser intensiven Prüfung darf der Anbieter die De-Mail-Dienste zur Verfügung stellen und muss u.a. alle drei Jahre erneut akkreditiert werden (vgl. §17 Abs. (3) De-MailG). [fSidI14, dI16, fSidI11]

Über dem De-Mail-Gesetz hinaus wurden anschließend zwei weitere Gesetze erlassen. Zum einen das E-Government-Gesetz (EGovG) und zum anderen das E-Justice-Gesetz. Mit dem EGovG sollen die öffentlich-rechtlichen Verwaltungstätigkeiten in elektronischer Weise von Bund, Ländern und Kommunen einfacher, effizienter und benutzerfreundlicher gestaltet werden. De-Mail findet durch ihre Funktionalität mit der Absendebestätigung hier entsprechend Anwendung. Das E-Justice-Gesetz hingegen soll den elektronischen Rechtsverkehr der Gerichte fördern und erweitern. Über De-Mail können daher elektronische Dokumente einem Gericht zugespielt werden. [dI15]

[5] De-Mail-Gesetz; §17 Abs. (1); April 2011; https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/De_Mail/Gesetz-De-Mail.pdf; zuletzt besucht am 27.07.2016

Nach der Vorstellung des Konzeptes von De-Mail ist zu erkennen, dass aus kryptographischer Sicht die Punkte Vertraulichkeit, Integrität und Authentizität in gewissem Maße umgesetzt werden. Es ist dennoch möglich einen Man-in-the-middle-Angriff durchzuführen welcher in Kapitel 3.3.3 kurz erläutert wird. Im folgenden wird die von De-Mail angegebene bequeme Benutzung des Dienstes dargestellt.

3.3.2 Verwendung

Die Verwendung von De-Mail-Diensten wird als besonders benutzerfreundlich angegeben. Zuvor muss der Anwender sich bei einem zugelassenen DMDA registrieren und identifizieren lassen. Im folgenden wird dieser Prozess anhand von GMX als DMDA beispielhaft erläutert, da er sich von Anbieter zu Anbieter unterscheiden kann. Bei der Anmeldemaske muss der Anwender seine persönlichen Daten angeben, die auf dem angegebenen Ausweisdokument zu finden sind. Für eine sichere Anmeldung muss zusätzlich die Mobilfunknummer angegeben werden. Die E-Mail-Adresse muss aus dem Vor- und Nachnamen des Anwenders bestehen, wobei der Vorname abgekürzt sein darf. Daher wird ihm im nächsten Schritt eine Auswahl von noch verfügbaren Adresse angeboten. Im Anschluss muss die Identität des Anwenders geprüft werden. Dies erfolgt persönlich durch ein Treffen bei dem die vorher angegebenen Daten mit den Daten auf dem Ausweisdokument abgeglichen werden. Alternativ kann der Anwender sich - mit einem weiteren entsprechendem Formular - auch bei einem Paketshop identifizieren lassen. War die Identifizierung erfolgreich kann der De-Mail-Dienst in Anspruch genommen werden. Für eine bequemere Identifizierung kann ein DMDA auch die Online-Ausweisfunktion (eID-Funktion) durch den neuen Personalausweis anbieten. Die Anmeldung einer De-Mail-Adresse bei GMX ist kostenpflichtig. Der Versand einer E-Mail hingegen kostenfrei, da automatisch eine Flatrate verwendet wird. Lediglich ein Einschreiben oder die Versandoptionen *Persönlich* und *Vertraulich* weisen zusätzliche Kosten auf. Die Verwendung von De-Mail ist identisch mit der einer herkömmlichen E-Mail. Der Benutzer kann durch die Weboberfläche des DMDA die Dienste in Anspruch nehmen oder durch einen E-Mail-Client das De-Mail-Konto einrichten. Es wird keinerlei zusätzliche Software benötigt. Sobald der Anwender eine Ende-zu-Ende-Verschlüsselung einsetzen möchte, muss - wie bei der herkömmlichen E-Mail - ggf. entsprechende Software installiert werden. Wird ein eigener E-Mail-Server betrieben kann dieser problemlos und ohne viel Aufwand durch ein Gateway an den De-Mail-Dienst angebunden werden. [Gmb16, Gmb14]

3.3.3 Vor- und Nachteile

Die klaren Vorteile die De-Mail mit sich bringt ist die Benutzerfreundlichkeit sowie die automatische Transportverschlüsselung. Nach einer erfolgreichen Registrierung und Identifizierung kann der Anwender wie gewohnt und ohne zusätzlicher Software E-Mails versenden und empfangen. Der Weg den eine E-Mail im System hinter sich lässt ist zudem verschlüsselt und soll für Dritte nicht zugreifbar sein. Weiterhin bilden die gesetzlich geschaffenen Grundlagen das Fundament für De-Mails. Daher ist eine De-Mail rechtlich abgesichert und soll eine Alternative zur herkömmlichen Papierpost darstellen. Die Nachteile der Papierpost, wie Porto-, oder Materialkosten sowie Personal- und Arbeitszeitkosten können durch De-Mails deutlich eingespart werden. [fSidI14, Gel10]

Dem gegenüber stehen eine Menge kritischer Stimmen gegen den Einsatz von De-Mail. Vor allem wird der nicht verpflichtende Einsatz einer Ende-zu-Ende-Verschlüsselung negativ gesehen. Diese gibt es bereits seit Jahrzehnten und gilt aus kryptographischer Sicht als sehr sicher. Als Begründung hat die Bundesregierung gesagt, dass es gegen das eigentliche Ziel der Benutzerfreundlichkeit spricht eine derartige Verschlüsselung verpflichtend zu machen (vgl. [Bun10]). Andere Meinungen berichten darüber, dass die Sicherheitsbehörden ein gewisses Interesse daran haben die E-Mails einsehen zu können. Zudem kann De-Mail nicht so sicher wie die Papierpost sein. Die Ende-zu-Ende-Verschlüsselung ist bei der Papierpost vergleichbar mit dem Umschlag. In ihm bleibt der Inhalt während des gesamten Versandweges verschlossen und somit unzugänglich. Nach jahrelanger scharfer Kritik wird ab April 2015 ein Browser-Plugin für De-Mail zur Ende-zu-Ende-Verschlüsselung bereitgestellt. Dies soll die Verwendung einfacher gestalten, ist aber dennoch nicht verpflichtend. Ein weiteres Manko ist die *Umverschlüsselung* der Nachricht auf den Servern der DMDAs. Für kurze Zeit liegt die E-Mail unverschlüsselt auf dem Server und kann von Unbefugten eingesehen werden und ermöglicht eine bewusste Überwachung. Das Gegenargument lautet, dass die E-Mails auf Schadsoftware geprüft wird. Allerdings würde man eine gezielte Verbreitung von Schadsoftware nicht mit einem kostenpflichtigen Dienst durchführen der es zusätzlich ermöglicht die Person zu identifizieren die diesen Angriff initiiert hat (vgl. [Neu13]). Durch die bisher wenigen zentralen DMDAs ergibt sich außerdem ein sehr gutes Angriffsziel, um an die kurzzeitig unverschlüsselten Nachrichten zu gelangen. Ein weiterer Punkt ist die Intransparenz der Technischen Richtlinien, da nicht genau angegeben ist mit welchem Verfahren die Transportverschlüsselung durchgeführt wird. Daher muss in Betracht gezogen werden, dass ein Man-in-the-middle-Angriff möglich ist. Bereits die Integritätssicherung von De-Mails beim Auslassen der Versandoption *Absender-bestätigt* weist sicherheitskritische Mängel auf. [Lec11] weist daraufhin, dass zur Prüfung der Integrität lediglich der Hashwert geprüft wird. Bei

einem Man-in-the-middle-Angriff kann ein Angreifer den Inhalt und den Hashwert anpassen, womit die Prüfung der Integrität beim Empfänger positiv ausfallen würde. Durch die Option *Absender-bestätigt* wird hingegen der Hashwert signiert. Somit würde bei einer geänderten Nachricht die Prüfung negativ ausfallen. Dennoch wird das gesamte Integritätsverfahren als kritisch gesehen, da der DMDA den Hashwert und die Signatur erzeugt und nicht der Absender. Aus diesem Grund kann der DMDA oder ein Angreifer der sich bspw. zwischen Absender und DMDA aufhält selber rechtskräftige E-Mails versenden von dem der Absender nichts erfährt.

Letztendlich ist De-Mail immer noch eine Insellösung für den Versand von E-Mails innerhalb Deutschlands, was u.a. auf die Interoperabilität zur herkömmlichen E-Mail zurückzuführen ist. Zwar gibt es bereits über eine Millionen angemeldete Benutzer, aber wie häufig De-Mail genutzt wird und wie hoch das Datenaufkommen innerhalb des Systems ist, ist unbekannt. [Bor15, Bis15, Lec11, Wel11] Daher trifft, nach dieser Betrachtung, der zu Beginn zitierte Slogan nur zur Hälfte zu. De-Mail ist zwar einfach in der Bedienung, aber so sicher wie Papierpost auf keinen Fall.

3.4 SMTP/TLS

3.4.1 Konzept

Um ein grundlegendes Verständnis von SMTP zu bekommen, werden zu Beginn einzelne Komponenten eines E-Mail Systems kurz erläutert. Im Anschluss wird das SMTP in groben Zügen beschrieben, um danach die Möglichkeit der Verschlüsselung von SMTP diskutieren zu können.

Damit ein Sender eine E-Mail erfolgreich von seinem Sender an den oder die entsprechenden Empfänger senden kann, durchläuft die E-Mail mehrere Komponenten eines E-Mail Systems. Um eine gemeinsames Verständnis davon zu bekommen, wurde eine Internet Mail Architektur entwickelt. Sie ist im RFC 5598 spezifiziert und sammelt das bisher angesammelte Wissen über den Ablauf der E-Mail Zustellung. [S.709 TW12, Cro09]

Der Anwender verfasst eine E-Mail i.d.R. in einem E-Mail Client oder in der Weboberfläche seines E-Mail Anbieters. Ferner kann er erhaltene E-Mails dort einsehen, weiterleiten, beantworten oder löschen. Diese Komponente wird fachlich als Message User Agent (MUA) bezeichnet. Die entscheidende Aufgabe des MUA - aus Sendersicht - ist das Zustellen der E-Mail an den sogenannten Message Submission Agent (MSA). Aus Empfängersicht bezieht der MUA die E-Mails vom E-Mail System durch entsprechende Protokolle wie z.B. dem POP3 oder IMAP, da das SMTP dafür nicht konzipiert ist. [S.709 TW12, Cro09]

Der MSA dient als Zwischenstation bevor die E-Mail durch das System weitergeleitet wird. Dies hat einige historische Gründe. Jede Person konnte ohne weiteres E-Mails an die Server der Anbieter senden damit diese sie weiterleiteten. Dadurch wurde die Zunahme von Spam-Mails gefördert und war nicht im Interesse der Anbieter. Damit nur Berechtigte den Dienst des E-Mail Servers in Anspruch nehmen dürfen, kann heutzutage ein Authentifizierungsmechanismus eingesetzt. Ebenso führt der MSA weitere Überprüfungen und Korrekturen des Nachrichtenformates der vom MUA erzeugten E-Mail durch. Diese Funktionalitäten kann das SMTP allerdings nicht leisten, welches zwischen dem MUA und MSA eingesetzt wird. Daher wurde das *Message Submission*-Protokoll auf Basis von SMTP eingeführt und soll seitdem generell eingesetzt werden (vgl. [Kle08, GK11]). [S.728f TW12, Cro09]

Die nächste Komponente ist der Message Transfer Agent (MTA) der die E-Mail vom MSA erhält, der sowohl eine Client- als auch eine Serverrolle in dem System einnimmt. Eine E-Mail passiert i.d.R. mehrere MTAs bevor sie beim Empfänger ankommt. Die MTAs kommunizieren mit dem SMTP untereinander und verwenden ein einheitliches Nachrichtenformat dafür (vgl. [Res08]). Der MTA der die E-Mail vom MSA erhalten hat baut eine direkte Verbindung zu dem auf dem E-Mail Server des Empfängers gelegenen MTA auf. Gelingt dies wird die E-Mail weitergeleitet. Ist der MTA allerdings nicht erreichbar, kann der MTA die E-Mail zwischenspeichern und es zu einem späteren Zeitpunkt nochmals versuchen. [S.730 TW12, Cro09]

Die auf dem E-Mail Server des Empfängers eingetroffene E-Mail wird dem Message Delivery Agent (MDA) - die letzte Komponente und ein spezieller MTA - übergeben. Dieser hat zwei Funktionen. Zum einen stellt er die E-Mail dem MUA des Empfängers zur Verfügung. Zum anderen kann er die E-Mail an eine vom Empfänger eingetragene andere E-Mail Adresse weiterleiten. Somit kann der MUA des Empfängers die E-Mail mit POP3 oder IMAP vom Server herunterladen oder über die Weboberfläche einsehen. [Cro09]

Das SMTP ist ein reines Übertragungsprotokoll von elektronischen Nachrichten in Rechnernetzwerken und ist im RFC 5321 definiert. Es ist ein Protokoll der Anwendungsschicht und setzt i.d.R. auf das TCP auf. Für den Nachrichtentransfer ist Port 25 reserviert. Durch die Erweiterung des *Message Submission*-Protokolls wurde zudem der Port 587 freigehalten. Heutzutage hat SMTP auf der einen Seite die Aufgabe eine E-Mail von einem MUA entgegen zu nehmen. Auf der anderen Seite übernimmt das SMTP die Übertragung zwischen den MTAs. Das Protokoll ist ein sehr einfach gehaltenes auf ASCII-Text basiertes Protokoll. Man kann daher über *telnet* eine Verbindung zum E-Mail Server aufbauen und eine Nachricht versenden. Es umfasst wenige Befehle, um eine E-Mail erfolgreich an einen Server zu übermitteln. Zusätzlich gibt es einige durchnummerierte Antwortcodes die entscheidend für das weitere Verfahren

sind. Bspw. deuten Antworten des Servers mit einer führenden 5 im Antwortcode auf einen fatalen Serverfehler hin. Wohingegen der Code 250 signalisiert, dass der empfangenen Befehl akzeptiert wurde. Wurde bspw. eine Verbindung zum Server über den Port 25 aufgebaut und bestätigt sendet der Client eine *HELO*-Nachricht. Nach dessen Bestätigung kann der Sender mit *MAIL FROM: <Senderadresse>* und *RCPT TO: <Empfängeradresse>* die Adressen von Sender und Empfänger angeben. Konnten diese Adressen vom Server bestätigt werden sendet der Client eine *DATA*-Nachricht. Er wird im positiven Fall dazu aufgefordert die E-Mail zu senden und dessen Ende mit einem . in einer extra Zeile abzuschließen. Letztlich schließt der Client mit *QUIT* die Sitzung. [S.725ff TW12, Kle08]

SMTP ist einfach aufgebaut und funktioniert sehr gut, weist aber einige Nachteile auf. Zum einen ist keine Authentifizierung des Senders vorgesehen, wodurch die Senderadresse manipuliert werden kann. Des Weiteren versendet SMTP die E-Mails unverschlüsselt durch das Netzwerk. Daher wurde SMTP um einige Mechanismen erweitert (ESMTP) die in den RFC integriert wurden. Das sogenannte Extended SMTP (ESMTP) sieht vor, dass der Client zu Beginn keine *HELO*-, sondern eine *EHLO*-Nachricht sendet, um die Erweiterungen in Anspruch nehmen zu können. Die Verbindung kann sowohl über Port 25 oder 587 erfolgen. Wird die Nachricht vom Server akzeptiert, sendet er seine zur Verfügung stehenden Erweiterungen an den Client. Z.B. kann sich der Client mit der Erweiterung *AUTH* gegenüber dem Server authentifizieren (vgl. [EE07]). Wohingegen die Erweiterung *STARTTLS* einen Wechsel zu einer mit TLS verschlüsselten Verbindung vorsieht (vgl. [Hof02]). [S.727f TW12, Kle08]

Mit der STARTTLS-Erweiterung kann auch SMTP wie andere Protokolle der Anwendungsschicht - z.B. HTTP - auf eine Transportverschlüsselung mit TLS zurückgreifen. Sie wurde 1999 im - mittlerweile obsoleten - RFC 2487 standardisiert. Somit ist Authentizität, Integrität und Geheimhaltung des E-Mail Versandes zwischen zwei Endsystemen gewährleistet. Abbildung 3.6 zeigt einen Auszug aus einem Kommunikationsdialog zum Aufbau einer STARTTLS-Verbindung zwischen einem Client (C) und einem Server (S). Nach der Auswahl dieser Erweiterung, erhält der Client zunächst die Bestätigung (Code 220) und Aufforderung (*Go ahead* eine TLS-Verbindung aufzubauen. Dies muss der Client tun bevor er weitere SMTP-Befehle an den Server senden darf. Ferner gibt es zu dem Code 220 auch den Code 454. Er weist darauf hin, dass eine TLS Verbindung zur Zeit nicht möglich ist. Daher muss der Client sich entscheiden wie er weiter verfährt. Er kann es bspw. später noch einmal versuchen oder die E-Mail unverschlüsselt senden. Nach der Aushandlung der TLS-Verbindung müssen Client und Server sich entscheiden, ob sie weiterhin miteinander kommunizieren wollen. Erachtet der Client oder Server die ausgehandelten Parameter als nicht sicher genug kann der Client dies entsprechend mit einer *QUIT*-Nachricht oder der Server mit einer Fehlermeldung deutlich

machen. Entscheiden sich beide Seiten für die Fortführung der TLS-Verbindung, sollte der Client eine weitere *EHLO*-Nachricht senden und kann mit dem SMTP fortführen. [Hof02]

```
S: <waits for connection on TCP port 25>
C: <opens connection>
S: 220 mail.imc.org SMTP service ready
C: EHLO mail.example.com
S: 250-mail.imc.org offers a warm hug of welcome
S: 250-8BITMIME
S: 250-STARTTLS
S: 250 DSN
C: STARTTLS
S: 220 Go ahead
C: <starts TLS negotiation>
C & S: <negotiate a TLS session>
C & S: <check result of negotiation>
C: EHLO mail.example.com
S: 250-mail.imc.org touches your hand gently for a moment
S: 250-8BITMIME
S: 250 DSN
```

Abbildung 3.6: Aufbau einer SMTP-Verbindung mit STARTTLS [Hof02]

Neben STARTTLS gab es Ende der 90er Jahre für kurze Zeit den von der IANA für SMTPS reservierten Port 465. Dieser ermöglichte ebenfalls ein abgesichertes SMTP über eine TLS-Verbindung. Im Gegensatz zu STARTTLS wird bei SMTPS eine sogenannte implizite TLS-Verbindung aufgebaut. Bevor SMTP-Befehle zwischen Client und Server ausgetauscht werden, wird eine sichere Verbindung mit TLS aufgebaut. Erst bei bestehender Verbindung wird mit dem SMTP die Kommunikation fortgeführt. Laut [Hof98] ist der spezielle Port allerdings bedenklich hinsichtlich der Sicherheit, Performanz und Interoperabilität von SMTP. Daher wurde die Reservierung des Ports mit der Standardisierung von STARTTLS wieder zurückgezogen. Dennoch wird SMTPS über diesen Port, z.B. unter Thunderbird, weiterhin eingesetzt. [Hof98, Web08]

3.4.2 Verwendung

Sowohl STARTTLS als auch SMTPS können in den gängigen E-Mail Clients wie Thunderbird und MS Outlook problemlos eingestellt werden. Dafür muss entweder im Einrichtungsassistenten oder manuell der jeweilige Server des E-Mail Anbieters angegeben werden. Nötig dafür ist allerdings, dass der Anbieter einen E-Mail Server mit dem jeweiligen Verschlüsselungsprotokoll anbietet. Darüber hinaus muss neben dem Servernamen auch der entsprechende Port angegeben werden: 587 für STARTTLS und 465 für SMTPS. Diese Konfiguration gewährleistet eine Transportverschlüsselung vom E-Mail Client zum E-Mail Server des Anbieters. Ob die E-Mail vom Anbieter-Server des Senders zum Anbieter-Server des Empfängers verschlüsselt ist, können Sender und Empfänger nicht beeinflussen.

Darüber hinaus kann man mit dem Werkzeug *netcat* testen, ob ein Server eine STARTTLS Verbindung anbietet. Dies geschieht mit dem Kommando *nc <E-Mail-Server_des_Anbieters> smtp*. Sobald der E-Mail Server mit dem Code 220 antwortet, kann man eine *EHLO*-Nachricht senden. Im Anschluss erhält man, sofern unterstützt, eine Liste der ESMTP-Befehle die der Server anbietet. [Sch09]

3.4.3 Vor- und Nachteile

Nachdem SMTP ohne Verschlüsselungsmechanismus Anfang 1980 entwickelt wurde, bietet die Erweiterung um STARTTLS mittlerweile eine gute und zeitgemäße Transportverschlüsselung auf Basis von TLS. Zudem ist es positiv zu bewerten, dass eine TLS-Verbindung ausgehandelt werden muss, nachdem der Client die Bestätigung vom Server erhalten hat, dass STARTTLS genutzt wird. Somit darf der Client vorerst nicht mit dem SMTP-Verfahren fortführen bis nicht eine geschützte Verbindung besteht. Dies bedingt allerdings, dass Client und Server sich an die im RFC vorgeschriebene Bedingung halten. Zusätzlich wurde 2015 festgestellt, dass die größten E-Mail Anbieter STARTTLS zur Verfügung stellen (vgl. [DAM+15]). [Hof02]

Zwar bietet STARTTLS aus theoretischer Sicht eine sichere Verschlüsselung, da sie das aus kryptographischer Sicht solide TLS-Protokoll einsetzt. Dennoch gibt es aus praktischer Sicht einiges zu bemängeln. Eine Ende-zu-Ende-Verschlüsselung wird von der Erweiterung nicht angeboten. Die E-Mails sind nur auf dem Transportweg zwischen Client und Server bzw. zwischen den E-Mail Servern untereinander verschlüsselt. Dies bedeutet, dass die E-Mails zwischenzeitlich im Klartext auf den Servern vorzufinden sind. Darüber hinaus müssen die E-Mail Server untereinander kein STARTTLS einsetzen. Ebenso besteht die Gefahr, dass auch der Client oder Server eine STARTTLS-Verbindung, obwohl sie eingestellt ist, nicht verwenden, da sie die ausgehandelten kryptographischen Parameter nicht für ausreichend sicher erachten. Daher muss der Anwender dem Client und den Servern vertrauen und hat keinen Einfluss darauf. In [DAM+15] wird der Einsatz von STARTTLS in der Praxis in einer aktuellen Studie gezeigt. Im April 2015 wurden 80% STARTTLS-Verbindungen von Gmail Servern zu anderen Servern für ausgehende E-Mails verzeichnet. Dem gegenüber stehen STARTTLS-Verbindungen für eingehende E-Mails auf die Server von Gmail. Zum gleichen Zeitpunkt wurden 60% der Verbindungen zu diesen Servern mit TLS verschlüsselt. Die Tendenz zur Verschlüsselung ist für ein- und ausgehende E-Mails steigend. Ein weiterer Punkt sind die verwendeten Ciphersuites der TLS-Verbindung. Es ist zu sehen, dass immer noch schwache Verschlüsselungsalgorithmen wie der RC4 von den Clients verwendet werden. Laut [DAM+15] sind es 45% der eingehenden STARTTLS-Verbindungen auf Gmail Server. Ferner sind Man-in-the-middle-Angriffe möglich. Ein Server könnte bspw. die Option *250-STARTTLS* aus der ESMTP-Auswahl löschen. Der

Client kann daher keine TLS-Verbindung aufbauen und muss sich entscheiden wie er weiter verfährt. Im schlimmsten Fall baut er eine unverschlüsselte Verbindung auf und sendet die E-Mail als Klartext an den Server. Ebenfalls kann ein Server die STARTTLS-Erweiterung anbieten, aber die Anfrage des Clients und die Antwort des Servers für eine TLS-Verbindung ändern. Damit ist es möglich schwache Ciphersuites zu verwenden die eine Verschlüsselung hinfällig machen. [Hof02, DAM+15]

Zusammenfassend ist zu sehen, dass es stark von der Implementierung der E-Mail Clients und Server abhängt, ob und in wie weit eine gute Transportverschlüsselung eingesetzt wird. Aus [DAM+15] wird dennoch deutlich, dass die größten E-Mail Anbieter den STARTTLS-Standard vermehrt umsetzen. Allerdings kann die Frage gestellt werden, warum dieser Zuwachs erst nach 17 Jahren - nach der Standardisierung von STARTTLS - verzeichnet wird. Zudem besteht das Problem der schwachen Ciphersuites die aus Kompatibilitätsgründen weiterhin eingesetzt werden. Infolgedessen muss den Anbietern ein großes Vertrauen für eine zeitgemäße Verschlüsselung des Transportweges entgegengebracht werden.

4 Konzept eines OpenPGP-Verteilers

Im Folgenden wird zunächst kurz auf bereits bestehende ältere und aktuellere Implementierungen und Lösungen von verschlüsselten Mailinglisten eingegangen. Im Anschluss wird das eigene Lösungskonzept sowie der dazugehörige Entwurf vorgestellt.

4.1 Bisherige Lösungen

4.1.1 Schleuder

Schleuder ist eine unter GNU GPL stehende Open Source Mailinglisten-Implementierung. Es verwendet die GnuPG-Implementierung zum Ver-, und Entschlüsseln sowie zum Signieren von E-Mails. Neben den Mitgliedern auf einer Liste können auch Nicht-Mitglieder E-Mails an die Schleuder-Mailinglisten senden und empfangen. Auch unverschlüsselte E-Mails werden von dem System weitergeleitet. Sobald der Schleuder-Server eine mit dem öffentlichen Schlüssel der jeweiligen Mailingliste verschlüsselte E-Mail erhält wird diese entschlüsselt. Nachdem die eventuell vorhandene Signatur verifiziert wurde, wird die Nachricht mit jedem öffentlichen Schlüssel der Listenmitglieder verschlüsselt sowie mit dem eigenen privaten Schlüssel signiert und abgesendet. Somit muss lediglich der öffentliche Schlüssel der Mailingliste vom Server bekannt sein, um eine Ende-zu-Ende-Verschlüsselung zu gewährleisten. Ein weiterer Vorteil ist die Anonymisierung der Metadaten einer E-Mail. Die E-Mail- und IP-Adresse des Senders wird aus der eingehenden verschlüsselten E-Mail gelöscht und gegen die des Servers ersetzt. Ein Nachteil ist allerdings der Verbleib des privaten Schlüssels und des Passwortes zentral auf dem Server. Somit muss dem ein großes Vertrauen gegeben werden, dass dieser nicht kompromittiert wurde. Ebenso kann es zu Skalierungsproblemen kommen. [Sch]

4.1.2 GPGrelay

Eine weitere freie Open Source Implementierung unter der GNU GPL auf Basis von GnuPG ist GPGrelay. Dieses System fungiert als Proxy-E-Mail-Server lokal auf dem Client. Nachteile für den Anwender gängiger Implementierungen von Ende-zu-Ende-Verschlüsselungen sind bspw. die Schlüsselverwaltung, oder dass die E-Mail entschlüsselt im E-Mail-Client gespeichert

ist und bei jeder Einsicht explizit entschlüsselt werden muss. Damit einhergehend ist eine Textsuche in einer verschlüsselten E-Mail unmöglich. Diese Nachteile werden durch GPGrelay beseitigt. Dabei muss die GPGrelay-Software auf dem Client installiert und konfiguriert werden. Sobald eine E-Mail versendet wird, übernimmt der Proxy-Server die Verschlüsselung und Signierung für einen Sender. Er leitet die bearbeitet E-Mail eigenständig an den zuständigen SMTP-Server weiter. Erhält ein Empfänger eine verschlüsselte bzw. signierte E-Mail wird diese im Vorfeld von GPGrely entschlüsselt bzw. die Signatur verifiziert. Somit erhält der Anwender eine E-Mail im Klartext und hat nichts mit der Ver-, und Entschlüsselung sowie Signierung zu tun. Durch diesen Automatismus wird eine Transparenz für den Anwender geschaffen der auch gleichzeitig ein Nachteil sein kann. Bspw. können Konfigurationsfehler nicht erkannt werden, was einen Versand einer unverschlüsselten E-Mail nach sich ziehen kann. Für größere Unternehmen stellt die Verwendung von GPGrelay ggf. auch keine Alternative dar. S/MIME wird nicht unterstützt und zur Einrichtung des Proxy-Servers muss der Anwender einige Fachkenntnisse mitbringen. [Joh, Fox08]

4.1.3 Sichere E-Mail-Verteiler: Ein praxisorientierte Ansatz

Auch [Hasnt] macht sich wie die beiden bisher beschriebenen Systeme das Prinzip eines zentralen Servers zunutze, der das Ver- und Entschlüsseln einer E-Mail übernimmt und diese neu verschlüsselt an eine Mailingliste weiterleitet. Eine genaue Bezeichnung dieser Lösung kann der Quelle nicht entnommen werden. Als Standard zur E-Mail-Verschlüsselung wird S/MIME angewendet. Es soll ein benutzerfreundlicher Dienst angeboten werden, welcher eine sichere Ende-zu-Ende-Kommunikation umsetzt. Dieser Dienst führt eine sogenannte *'Umverschlüsselung'* durch. Dabei wird der verwendete Sitzungsschlüssel auf dem Server entschlüsselt und mit den entsprechenden öffentlichen Schlüsseln der Empfänger verschlüsselt. Allerdings besitzt diese Hybrid-Methode einen entscheidenden Nachteil. Der Server kann die E-Mail ggf. im Klartext mitlesen. Um diesem Problem ein wenig entgegenzuwirken, wird das *Separation of Duty*-Prinzip eingesetzt. Dieses verteilt die Verantwortungen auf zwei Komponenten. Das sogenannte *Mail Distribution Center* empfängt eine E-Mail und entnimmt ihr den verschlüsselten Sitzungsschlüssel. Dieser wird an das *Key Management Center* weitergeleitet, welches über den privaten Schlüssel des Servers verfügt. An dieser Stelle wird nun die *'Umverschlüsselung'* durchgeführt, indem der Sitzungsschlüssel entschlüsselt und mit den jeweiligen öffentlichen Schlüsseln der Empfänger verschlüsselt wird. Die neu erzeugten verschlüsselten Sitzungsschlüssel werden nun an das *Mail Distribution Center* zurückgegeben. Dieses fügt die ursprüngliche E-Mail jeweils mit den neu verschlüsselten Sitzungsschlüsseln zusammen und sendet sie an die Empfänger. [Hasnt]

4.1.4 Practical Encrypted Mailing Lists

Entgegen der bisher vorgestellten Lösungen verfolgt [Wal16] einen ganz anderen Ansatz. Bisherige Ideen einer verschlüsselten Kommunikation über eine Mailingliste sahen vor, dass entweder ein Sender eine E-Mail einzeln mit allen öffentlichen Schlüsseln der Empfänger verschlüsseln muss oder ein zentraler Server dies übernimmt und eine *'Umverschlüsselung'* vornimmt. Dies ist entweder nicht praxistauglich oder es muss vollstes Vertrauen in den Server gesteckt werden. Um eine wirklich sichere, praxistaugliche und vertrauenswürdige Mailingliste bereitzustellen wurde ein Ansatz entwickelt den OpenPGP-Standard für die Verwendung von Mailinglisten zu erweitern. Dabei soll das Versenden einer E-Mail an eine Liste identisch zu dem Versenden einer E-Mail an einen einzelnen Empfänger sein. Sobald ein Sender wie gewohnt eine E-Mail verschlüsselt senden will, wählt er den öffentlichen Schlüssel der Mailingliste aus. Die OpenPGP-Implementierung verschlüsselt nun nicht die Nachricht mit dem gewählten Schlüssel, sondern verwendet dafür die ihm bekannten öffentlichen Schlüssel der Mailinglisten-Mitglieder die mit diesem Schlüssel korrespondieren. Diese sind in einem sogenannten Schlüsselblock zusammengefasst. Das grundlegende Problem besteht darin, dass ein Sender immer den aktuellen Schlüsselblock der Mailingliste besitzen muss. Dafür wird die bereits existierende Infrastruktur von Schlüsselservern in OpenPGP - worüber die öffentlichen Schlüssel von Anwendern verteilt werden - genutzt. Dieser Ansatz besitzt den entscheidenden Vorteil, dass tatsächlich nur die Listenmitglieder den Klartext einsehen können und der Server die verschlüsselte E-Mail wie gewohnt weiterleitet. Weiterhin wird kein neuer Workflow definiert oder andere große Änderungen an der existierenden OpenPGP-Implementierung vorgenommen. [Wal16]

4.2 Analyse

4.2.1 Problemstellung

Wie bereits in Kapitel 1 und 3 erläutert, gibt es bereits Werkzeuge die eine sichere Ende-zu-Ende-Verschlüsselung von E-Mails zwischen zwei Parteien gewährleisten. Sie sind allerdings für eine Massenkommunikation nicht praxistauglich. Dies liegt zum einen an der sehr aufwendigen Schlüsselverwaltung und zum anderen daran, dass der E-Mail Client eine E-Mail für jeden Empfänger separat verschlüsseln muss. Dies nimmt bei vielen Empfängern und großen E-Mail Anhängen sehr viel Zeit in Anspruch und ist somit nicht gewollt. Daher müssen auf Grundlage dieser Werkzeuge Anwendungen dafür entwickelt werden die zudem die Massenverschlüsselung übernehmen und benutzerfreundlich sind. Ebenso muss die Schlüsselverwaltung

für den Anwender einfacher gestaltet werden. Wie im vorangegangenen Unterkapitel 4.1 beschrieben, gibt es bereits funktionierende und eingesetzte Anwendungen die unterschiedliche Eigenschaften aufweisen. Im folgenden wird zunächst der theoretische Aufbau und die gebotenen Eigenschaften eines eigenen dezentralen OpenPGP-Servers dargestellt. Mit ihm kann sich jeder seinen eigenen Verteilerserver zu Hause einrichten und verwalten.

4.2.2 Funktionalität des Systems

Das System soll auf dem im RFC 4880 definierten OpenPGP-Standard basieren. Ebenfalls sollen die E-Mail Formate PGP/MIME und Inline-PGP unterstützt werden. Es stellt den Anwendern fünf unterschiedliche Operationen zur Verfügung. Zum einen werden E-Mails vom Sender an den oder die im E-Mail-Header angegebenen Verteiler *umverschlüsselt*. Zum anderen hat der Anwender die Möglichkeit sich an die im E-Mail-Header angegebenen Verteiler an- und wieder abzumelden, seinen öffentlichen Schlüssel für diese Verteiler zu ändern oder die öffentlichen Schlüssel dieser Verteiler anzufragen. Die *Umverschlüsselung* bedeutet, dass die E-Mail mit dem privaten Schlüssel des jeweiligen Verteilers zunächst entschlüsselt und deren Signatur verifiziert wird. Dies bedingt die vorherige Signierung und Verschlüsselung der E-Mail mit dem privaten Schlüssel des Absenders bzw. dem öffentlichen Schlüssel des Verteilers. Im Anschluss muss die Anwendung die im Klartext vorliegende E-Mail mit dem eigenen privaten Schlüssel signieren und jedem öffentlichen Schlüssel der Verteilermitglieder verschlüsseln. Ist dies geschehen, übermittelt das System die individuell verschlüsselte E-Mail an die jeweiligen SMTP-Server des E-Mail Anbieters der Empfänger. Die weiteren vier Operationen werden durch eine ausschließlich signierte E-Mail über spezifische Befehle im E-Mail Betreff getriggert. Eine Übersicht der Operationen stellt Tabelle 5.1 mit den Einträgen Nr. 1-4 bereit. Der Absender erhält im Anschluss an einer dieser vier Operationen eine - wenn möglich - entsprechende signierte und verschlüsselte E-Mail mit dem positiven oder negativen Ergebnis des Befehls. Darüber hinaus besteht die Möglichkeit für den Administrator, über eine Web-Schnittstelle Anwender an einen Verteiler an- und wieder abzumelden.

4.2.3 Funktionale Anforderungen

Die funktionalen Anforderungen sind in Tabelle 4.1 zusammengefasst und wurden teilweise in groben Zügen angesprochen. Es wird nun auf einige Anforderungen eingegangen und kurz ihre Bewandtnis begründet.

Nr.	Beschreibung
1.0	Das System soll eine empfangene E-Mail mit einem der vier Befehle im Betreff bearbeiten und dem Sender eine spezifische signierte und verschlüsselte E-Mail zurück senden.
1.1	Das System soll eine empfangene E-Mail zur *Umverschlüsselung* an den oder die im E-Mail-Header angegebenen Verteiler weiterleiten.
1.2	Das System muss die empfangene E-Mail mit dem privaten Schlüssel des entsprechenden Verteilers entschlüsseln und die Signatur des Sender verifizieren.
1.3	Das System soll neben der Verschlüsselung auch eine Signierung (Authentizität, Integrität und Verbindlichkeit) von E-Mails durchführen.
1.4	Das System muss die entschlüsselte und bereits von ihm signierte E-Mail mit den öffentlichen Schlüsseln der Empfänger auf dem jeweiligen Verteiler verschlüsseln.
1.5	Das System soll die signierte und neu verschlüsselte E-Mail an alle Empfänger des Verteilers senden.
1.6	Zur Signierung soll das System ein zweites anderes Schlüsselpaar als zur Ver- und Entschlüsselung verwenden.
1.7	Das System muss die im Klartext vorliegende Nachricht unverzüglich aus dem Arbeitsspeicher löschen, wenn diese nicht mehr gebraucht wird.
2.0	Das System muss die öffentlichen Schlüssel der einzelnen Anwender eines Verteilers verwalten.
2.1	Das System muss es zulassen, dass öffentliche Schlüssel der Anwender zu ändern sind.
2.2	Das System muss auf Anfrage eines Anwenders die öffentlichen Schlüssel des im E-Mail-Header angegebenen Verteilers an ihn senden.
3.0	Das System sollte E-Mails an unbekannte Verteiler zurückweisen.
3.1	Das System soll E-Mails zur *Umverschlüsselung* nur weiterverarbeiten, wenn sie korrekt signiert und korrekt verschlüsselt sind. Andernfalls wird dem Absender dies mitgeteilt.
3.2	E-Mails mit einem der vier Befehle im Betreff sollen nur vom System angenommen werden, wenn sie korrekt signiert sind. Eine Verschlüsselung dieser E-Mails ist nicht nötig und wird vom System nicht akzeptiert.
4.0	Das System muss eine Schnittstelle anbieten, die es ermöglicht, dass sich Anwender an einen Verteiler anmelden können.

Nr.	Beschreibung
4.1	Das System muss eine Schnittstelle anbieten, die es ermöglicht, dass sich Anwender von einem Verteiler abmelden können.
4.2	Es sollten schnell und problemlos durch den Administrator E-Mail-Adressen zu einem Verteiler hinzugefügt oder von ihm gelöscht werden können.
5.0	In dem System sollen vorerst nicht-moderierte Verteiler eingerichtet werden können.
6.0	Das System muss gemäß SMTP die E-Mails die es nicht weiterleiten konnte vorhalten und es zu einem späteren Zeitpunkt noch einmal versuchen.
7.0	Das System soll dezentral eingesetzt werden können, damit die Vertraulichkeit der Verteiler im Gegensatz zu zentralen Systemen erhöht wird.

Tabelle 4.1: Funktionale Anforderungen

Damit dem Spoofing von E-Mail-Adressen entgegengewirkt werden kann, müssen neben den ausgehenden auch die eingehenden E-Mails korrekt signiert sein (vgl. Anforderungen 1.0, 1.3, 3.1 und 3.2). Mit der Signatur ist die Authentizität, Integrität sowie die Verbindlichkeit der E-Mail gewährleistet und beugt somit einer Verfälschung vor. Daher ist es notwendig, dass das System die Signaturen verifiziert und selber eine einsetzt. Es wurde bewusst entschieden, dass die E-Mails signiert und erst dann verschlüsselt werden. Zum einen ist dies in [S.69 CDF+07] festgeschrieben und zum anderen ist es aus kryptographischer Sicht nicht sinnvoll zuerst zu verschlüsseln und im Anschluss zu signieren. Dies liegt daran, dass ein Angreifer die E-Mail abfangen und die Signatur entfernen kann und somit über den verschlüsselten Inhalt verfügt. Er kann nun diesen Inhalt selber signieren und sich als Urheber dessen ausgeben. Ein zusätzlicher sinnvoller allgemeiner Schritt wäre die zunächst signierte und dann verschlüsselte E-Mail nochmals zu signieren. Dadurch hat der Sender tatsächlich die E-Mail verschlüsselt und kann nicht von einem Angreifer ausgetauscht werden. Dies ist allerdings nicht im OpenPGP Nachrichtenformat definiert und wird daher kritisiert (vgl. [Dav01]). Allerdings ist für die eigene Implementierung eine zusätzliche Signierung der verschlüsselten E-Mail nicht notwendig. Kann ein Angreifer die E-Mail entschlüsseln, weil er z.B. Mitglied auf dem Verteiler ist, kann er sie nicht an Dritte weiterleiten, ohne dass sie es feststellen. Dies liegt daran, dass die E-Mail vom Verteiler signiert wurde und nicht vom Urheber der E-Mail. Ebenfalls ergänzt die eigene Implementierung - nach erfolgreicher Verifikation - die E-Mail Adresse des Urhebers zu der ursprünglichen E-Mail, bevor diese sie signiert.

Die mathematische Theorie hinter der Erzeugung von Signaturen und verschlüsselten Nachrichten ist bei RSA identisch. Der einzige Unterschied liegt in der Verwendung der einzelnen Schlüssel eines Schlüsselpaares. Zur Signierung wird der private und zur Verifikation der öffentliche Schlüssel genutzt. Zum Ver- bzw. Entschlüsseln von Nachrichten wird der öffentliche bzw. private Schlüssel eingesetzt. Somit ist es zum einen theoretisch möglich, dass man als Angreifer seinem gewählten Opfer einen Klartext zum Entschlüsseln gibt und eine gültige Signatur erhält. Und zum anderen kann ein verschlüsselter Text dem Opfer zum signieren gegeben werden, um dadurch den Klartext zu erhalten. Um dieser Schwachstelle zu entgehen, soll das System wie in Anforderung 1.6 beschrieben unterschiedliche Schlüsselpaare zum Signieren und Verschlüsseln einsetzen.

Anforderung 1.7 ist nötig, da man nicht mit Gewissheit aussagen kann, wann der Speicherbereich einer Variable vom Garbage Collector der eingesetzten Programmiersprache wieder freigegeben wird. Aus diesem Grund ist es durchaus möglich, dass der Inhalt der Variable - die bspw. den Klartext einer E-Mail darstellt - für unbestimmte Zeit aus dem Speicher gelesen werden kann. Somit sollte dieser Speicherbereich durch manuelles Aufrufen des Garbage Collectors im Quellcode explizit freigegeben werden.

Für eine E-Mail mit einem der vier spezifischen Befehle im Betreff ist es ausreichend, dass diese nur signiert und nicht verschlüsselt ist (vgl. Anforderung 3.2). Die Verschlüsselung bezieht sich lediglich auf den E-Mail-Body, nicht aber auf den E-Mail-Header in dem der Betreff verzeichnet ist. Diese Entscheidung ermöglicht es allerdings, dass bspw. Nachrichtendienste den E-Mail-Header auswerten können - sofern keine Transportverschlüsselung eingesetzt wird - und somit wissen zu welchem Zeitpunk sich z.B. ein Anwender an einem Verteiler anmeldet. Bettet man aber nun den Befehl im E-Mail-Body ein, muss sie zunächst entschlüsselt, zergliedert und analysiert werden, um feststellen zu können, um welche Art von E-Mail es sich handelt. Dieser Vorgang ist sehr rechenintensiv und zu aufwendig, sofern es sich nicht um eine E-Mail zur *Umverschlüsselung* handelt. Ebenfalls soll eine Analyse der E-Mails nicht Aufgabe der Anwendung sein. Die bereits angesprochene Auswertung des E-Mail-Headers ist allerdings auch bei verschlüsselten E-Mails möglich womit ein Bewegungsprofil erzeugt werden kann.

Anforderungen 4.0 und 4.1 werden durch jeweils einen der vier Befehle im Betreff der E-Mail umgesetzt. Durch die Webschnittstelle kann ein Verteiler-Administrator E-Mail Adressen hinzufügen oder löschen (vgl. Anforderung 4.2). Der Administrator authentifiziert sich gegenüber dem System mit der Verteileradresse und einem, bei der Erzeugung des Verteilers angegebenen, Passwortes.

Nicht-moderierte Verteiler (vgl. Anforderung 5.0) sind so konzipiert, dass jeder Anwender E-Mails an einen Verteiler senden kann, sofern er auch Mitglied des Verteilers ist. E-Mails von Nicht-Mitgliedern werden generell abgewiesen. Moderierte Verteiler hingegen halten die E-Mails von Nicht-Mitgliedern auf und können vom Administrator zugelassen werden. Diese Verwaltung benötigt einen zusätzlichen Aufwand und kann nachträglich ergänzt werden (siehe auch nicht-funktionale Anforderung Nr. 1).

Die Vertraulichkeit ist durch ein dezentrales System höher als durch ein zentrales (vgl. Anforderung 7.0). Bei zentralen Servern muss man sich auf den Anbieter verlassen, dass die dort abgelegten Daten in sicheren Händen sind. Auch bei Open Source Software bleibt durch die Zentralität ein gewisses Restrisiko, da von außen nicht genau festzustellen ist wie das System intern konkret arbeitet. Bei dezentralen Servern hingegen haben die Administratoren eigenständigen und flexiblen Zugriff auf das System. Damit können sie entsprechende Sicherheitsmechanismen anwenden und regelmäßig anpassen um die dort vorhandenen Daten vertraulich abzusichern.

4.2.4 Nicht-Funktionale Anforderungen

Die funktionalen Anforderungen sind in Tabelle 4.2 zusammengefasst. Es wird nun auf einige Anforderungen eingegangen und kurz ihre Bewandtnis begründet.

Nr.	Beschreibung
1	Das System wird zunächst nach dem KISS (Keep it Small and Simple) Prinzip entworfen.
2	Das System soll modular aufgebaut werden, um Erweiterungen vornehmen zu können.
3	Das System soll problemlos in eine andere ähnliche Umgebung portierbar sein.
4	Das System sollte im Mittel skalierbar sein.
5	Das System soll bei aufgetretenen Fehlern und Ausnahmen mit einer entsprechenden E-Mail an den Sender darauf hinweisen.
6	Das System soll einfach in der Bedienung sein.

Tabelle 4.2: Nicht-Funktionale Anforderungen

Zunächst sollen die Hauptfunktionalitäten - gemäß Anforderung 1 - umgesetzt werden. Diese sind die bereits angesprochenen fünf Operationen und der Versandt von E-Mails. Durch ein kleines und einfaches System kann zudem das Augenmerk verstärkt auf die Sicherheit

des Systems gelegt werden. Dies ermöglicht das Auffinden von potenziellen Schwachstellen und deren rechtzeitigen Beseitigung. Bereits solche minimalen Systeme können sehr komplex werden in denen die Übersicht zu den Sicherheitsaspekten schnell verloren gehen kann. Weitere Module, die zusätzliche oder ergänzende Aufgaben übernehmen, können im Nachhinein entsprechend problemlos erweitert werden (vgl. Anforderung 2).

Der Aspekt der Dezentralität macht es unerlässlich, dass das System in andere ähnliche Umgebungen portierbar sein sollte (vgl. Anforderung 3). Damit kann das System zügig und problemlos an weitere Administratoren verteilt und eingesetzt werden. Etwaige Fehler sollten dabei nicht auftreten, da die Hardwareumgebung identisch bzw. nahezu identisch ist. Lediglich einige Konfigurationen müssen vorgenommen werden, um den Server in Betrieb nehmen zu können.

Die Fehlerbehandlung stellt ebenfalls ein Qualitätsmerkmal dar und ist unerlässlich in einem soliden, sicheren und gut funktionierenden System (vgl. Anforderung 5). Entsprechend müssen die Anwender darüber informiert werden, damit diese darauf reagieren können. Daher sollen in Fehler- und Ausnahmefällen sinnvolle Benachrichtigungen versandt und ggf. diese geloggt werden.

Ein weiteres intensiv diskutiertes Themenfeld ist die Benutzerfreundlichkeit von Ende-zu-Ende-Verschlüsselungssystemen (vgl. Anforderung 6). Viele Anwender sind davon abgeschreckt ihre Privatsphäre zu schützen, wenn sie zusätzliche Software bedienen oder Funktionalitäten einsetzen müssen die ihnen Fremd sind. Dem zufolge sollte das System möglichst bedienerfreundlich und einfach in der Handhabung sein, um diese Hürde möglichst tief zu halten.

5 Realisierung eines OpenPGP-Verteilers

Nachdem ein Überblick über die theoretische Funktionalität des Verteilersystems gegeben wurde, wird im Folgenden Kapitel die Architektur und die Spezifikation beschrieben. D.h. es wird erörtert wie und mit welchen Mitteln die Anforderungen aus Kapitel 4.2 umgesetzt werden. Darüber hinaus werden am Ende des Kapitels mögliche Schwachstellen sowie ein Vergleich aller bereits in Kapitel 4.1 vorgestellten Verteilersysteme gezeigt.

5.1 Architektur

5.1.1 Kontextsicht

Die in Abbildung 5.1 dargestellte Kontextsicht stellt die zuvor diskutierten Anforderungen grob visuell dar. Sie zeigt die wichtigsten Anwendungsfälle für die Benutzer und Administratoren. Zudem lässt sie die Verknüpfungen zwischen den einzelnen Subsystemen erkennen.

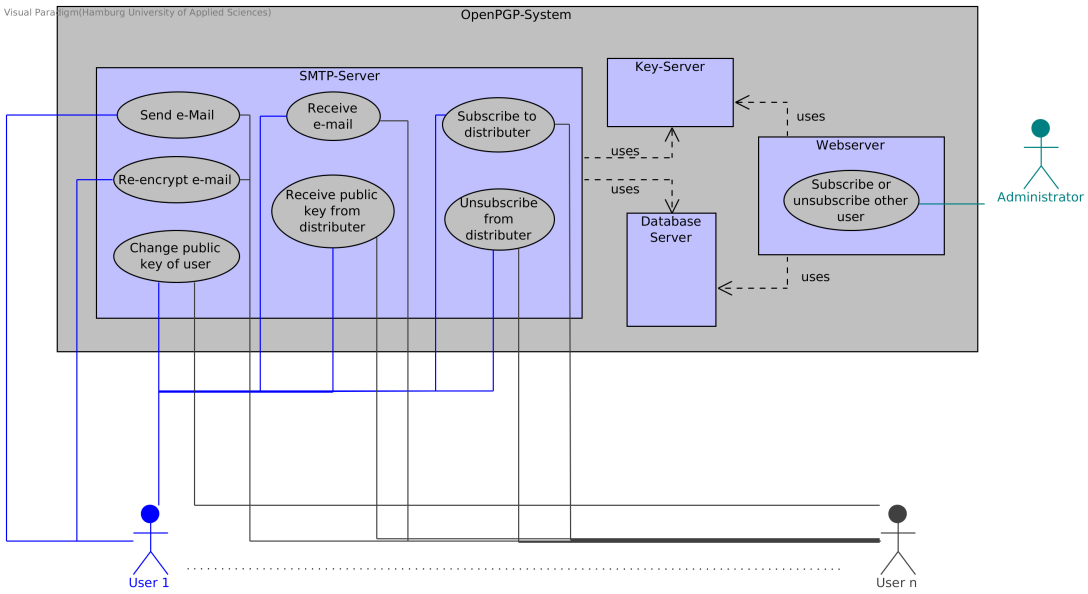

Abbildung 5.1: Kontextsicht für den OpenPGP-Verteiler

5.1.2 Komponenten-Diagramm

In Abbildung 5.2 ist das UML Komponentendiagramm dargestellt. Es besteht aus insgesamt sechs Komponenten. Die beiden Instanzen *GPG-Instance* und *Distributer Database* gehören logisch zu den jeweiligen Komponenten und sind aus Gründen des besseren Verständnisses außerhalb derer angesiedelt. Des Weiteren bestehen mit Ausnahme der *Web Service* Komponente, alle weiteren aus zwei Unterpaketen. Auf der einen Seite ein Paket in dem die Hauptfunktionalität der Komponente umgesetzt wird. Auf der anderen Seite ein zweites Paket in denen die spezifischen Exceptions abgelegt sind. In der folgenden Unterteilung werden die Funktionalitäten der einzelnen Komponenten und ihren Schnittstellen die sie anbieten oder verwenden genauer beschrieben. Die Fehlerbehandlung wird im darauf folgenden Unterkapitel behandelt.

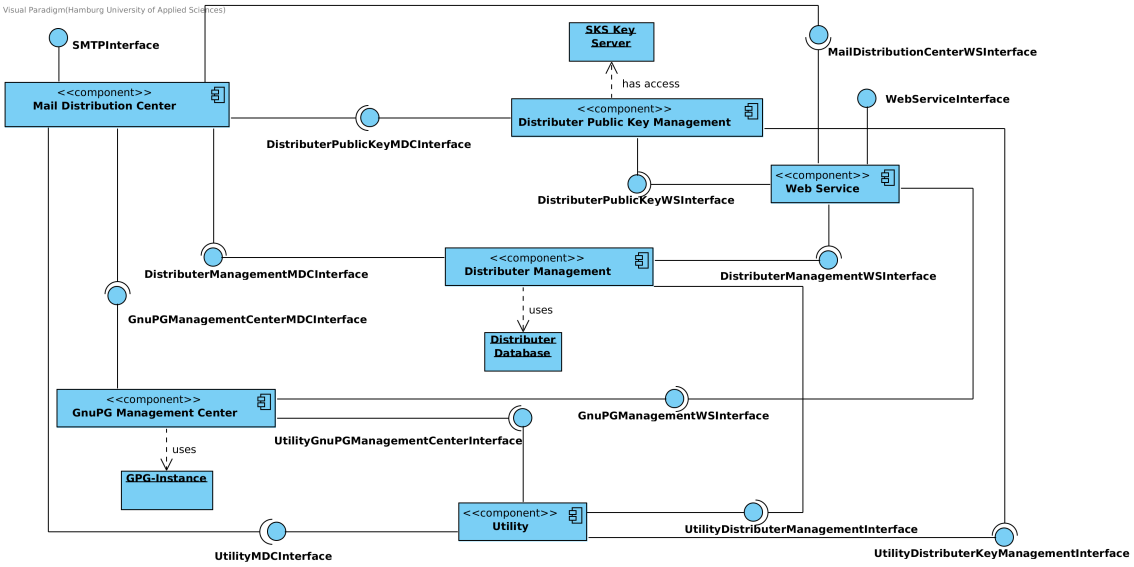

Abbildung 5.2: Komponentendiagramm für den OpenPGP-Verteiler

Mail Distribution Center

Im *Mail Distribution Center* gibt es zwei Hauptmodule, die die Kernfunktionalität dieser Komponente bereitstellen. Das *SMTPChannel* Modul wartet auf eingehende E-Mails - zu erkennen an dem *SMTPInterface* - und bearbeitet sie. *MTASendThread* stellt das zweite Modul dar mit dem E-Mails versendet werden. Zu Beginn muss die angenommene E-Mail geparst und die nötigen Header-Werte und der Body in Instanzvariablen gespeichert werden. Im

Anschluss verwendet das Modul die Enumeration *SubjectCommand*, um die Befehle im Betreff unterscheiden zu können. Tabelle 5.1 gibt einen Einblick in die vorhandenen Befehle.

Nr.	Variable	Wert	Beschreibung
1	ADDTODIST	'ADDTODIST'	Hinzufügen der E-Mail Adresse zum angegebenen Verteiler.
2	CHANGEPK	'CHANGEPK'	Ändern des öffentlichen Schlüssels der E-Mail Adresse auf dem angegebenen Verteiler.
3	DELFROMDIST	'DELFROMDIST'	Löschen der E-Mail Adresse vom angegebenen Verteiler.
4	GETPK	'GETPK'	Öffentliche Schlüssel des angegebenen Verteilers anfordern.
5	REENCRYPT	'REENCRYPT'	*Umverschlüsselung* der empfangenen E-Mail.

Tabelle 5.1: Übersicht der Betreff-Befehle

Hat der E-Mail Betreff die Länge eins und besteht er aus einem der entsprechenden Befehle - mit Ausnahme des *re-encrypt*-Befehls - , wird die dafür zuständige Methode aufgerufen. Andernfalls handelt es sich immer um eine E-Mail zur *Umverschlüsselung*. Bei den Befehlen 1 bis 4 wird zunächst immer erst die Signatur geprüft. Dabei bedient sich die Komponente an den Funktionalitäten anderer Komponenten. Um die Signatur mit dem *GnuPGManagementCenterMDCInterface* verifizieren zu können, muss sie zunächst aus der E-Mail extrahiert werden. Dies geschieht über das *UtilityMDCInterface*. Des Weiteren werden der Fingerprint und der öffentliche Schlüssel des Absenders zur Verifizierung benötigt. Diese werden in Abhängigkeit der Operation unterschiedlich ermittelt. Wird eine E-Mail Adresse zu einem Verteiler hinzugefügt (Befehl Nr. 1) oder ein Absenderschlüssel getauscht (Befehl Nr. 2), wird der Fingerprint aus dem Absenderschlüssel der sich im Anhang befindet extrahiert. Dies bedeutet, dass die E-Mail mit dem neuen Schlüssel signiert sein muss. Zunächst wird der Schlüssel aus dem Anhang durch die *Utility* Komponente extrahiert. Danach kann der Fingerprint aus dem Schlüssel durch das *GnuPGManagementCenter* ermittelt werden. Für das Löschen bzw. Anfragen der Verteilerschlüssel (Befehl Nur. 3 bzw. 4) wird der Fingerprint über das *DistributerManagementMDCInterface* und der Schlüssel über das *DistributerPublicKeyMDCInterface* ermittelt.

Nach erfolgreicher Verifizierung wird - bezogen auf die Befehle Nr. 1 und 3 - ein neuer Eintrag über das *DistributerManagement* vorgenommen bzw. der entsprechende Eintrag gelöscht. Ebenfalls wird der Fingerprint darüber hinzugefügt bzw. geändert (Befehl 1 bzw. 2).

Damit der korrekte Absenderschlüssel im System hinterlegt ist, muss dieser über das *DistributerPublicKeyMDCInterface* eingetragen bzw. geändert werden (Befehl 1 bzw. 2). Findet zudem der Absenderschlüssel keine Verwendung mehr, da die E-Mail Adresse komplett gelöscht wurde, wird dieser ebenfalls gelöscht (Befehl 3). Wird eine Adresse zu einem Verteiler hinzugefügt oder die Schlüssel eines Verteilers angefordert (Befehl 1 bzw. 4) werden zunächst die Schlüssel-IDs der Verteiler über das *GnuPGManagementCenterMDCInterface* ermittelt. Mit den IDs werden ebenfalls darüber die Verteilerschlüssel festgestellt die dem Absender mit der E-Mail übermittelt werden.

Handelt es sich allerdings um eine E-Mail zur *Umverschlüsselung* werden Fingerprint und Absenderadresse wie beschrieben durch die zuständigen Schnittstellen ermittelt. Über das *DistributerManagement* werden zudem die E-Mail Adressen mit ihren zugehörigen Fingerprints des Verteilers lokalisiert. Mit dieser Information müssen die Schlüssel aller Empfänger über das *DistributerPublicKeyManagementMDCInterface* ausfindig gemacht werden. Im Anschluss muss der verschlüsselte E-Mail Body über die *Utility* Komponenten extrahiert werden. Mit diesen Informationen kann durch das *GnuPGManagementCenter* die *Umverschlüsselung* durchgeführt werden. Sind keinerlei Fehler aufgetreten wird zuletzt ein *MTASendThread* erzeugt und ihm die nötigen Informationen übergeben. Die Hauptfunktionalität besteht zum einen darin die *umverschlüsselten* E-Mails an die Empfänger eines Verteilers korrekt zuzustellen und zum anderen eine E-Mail zu generieren, sofern es sich um einen der Betreff-Befehle 1 bis 4 handelt. Sofern eine E-Mail erzeugt werden muss, wird diese im positiven Fall signiert und verschlüsselt oder im negativen Fall nur signiert. Negative Fälle treten auf, wenn während der Abarbeitung Fehler aufgetreten sind. In jedem Fall wird dem Absender mit der E-Mail mitgeteilt was geschehen ist. Das Erzeugen der E-Mail geschieht über das *UtilityMDCInterface*, welche im Anschluss über das *GnuPGManagementCenterMDCInterface* signiert und verschlüsselt bzw. nur signiert wird. Bei vorliegender E-Mail wird im Anschluss gemäß SMTP die E-Mail versandt. Treten dabei Fehler auf, wartet der Thread einige Zeit und versucht es erneut. Dies geschieht entweder solange bis die E-Mail versandt wurde, oder drei Tage abgelaufen sind. In diesem Fall wird dem Absender eine E-Mail geschickt die ihn darauf hinweist, dass der Versandt an bestimmte Empfänger nicht möglich war.

GnuPG Management Center

Das *GnuPG Management Center* besitzt Zugriff auf eine GunPG-Instanz und ist hauptsächlich für die *Umverschlüsselung* zuständig. Sie bietet die Dienste für das *Mail Distribution Center* und den *Web Service* an. Eine *Umverschlüsselung* bedeutet, dass der zuvor extrahierte E-Mail Body entschlüsselt und verifiziert wird. Im Anschluss wird eine Signatur der entschlüsselten E-Mail

erzeugt. Diese und die E-Mail werden laut [ETLR01] in eine E-Mail im PGP/MIME-Format vom Content-Type *multipart/signed* verpackt und für jeden einzelnen Empfänger separat verschlüsselt. Der verschlüsselte Teil wird ebenfalls als PGP/MIME-Format in eine E-Mail verpackt und erhält den Content-Type *multipart/encrypted*. Die Funktionalität zur Erzeugung von PGP/MIME E-Mails wird durch das *UtilityGnuPGManagementCenterInterface* angeboten. Die verschlüsselte E-Mail wird nun mit der E-Mail Adresse in einem Wörterbuch abgespeichert und kann so im weiteren Verlauf versandt werden. Wichtig hierbei ist, die mit dem Klartext belegten Variablen aus dem Arbeitsspeicher zu löschen. D.h. der Garbage Collector muss hier manuell aufgerufen werden (vgl. funkt. Anforderung 1.7). Ist es zudem nötig eine E-Mail, die kurz vor dem Versandt durch den *MTASendThread* steht, zu signieren und zu verschlüsseln bzw. nur zu signieren, bietet dieses Modul eine entsprechende Funktion dafür an. Darüber hinaus werden über das Center die Schlüssel-IDs für einen gegebenen Verteiler ermittelt. Zudem werden die Schlüssel eines Verteilers über die angegebenen Schlüssel-IDs ermittelt. Diese beiden Funktionen sind in diesem Modul angesiedelt, da die Informationen über ein Schlüsselpaar eines Verteilers nur durch die GnuPG-Instanz abrufbar sind. Sie greift dafür auf das Home-Verzeichnis von GnuPG zu.

Distributer Management

Durch das *Distributer Management* hat das System Zugriff auf die zugrunde liegende Datenbank. In ihr werden die Verteiler, E-Mail Adressen und Fingerprints gespeichert. Ferner gibt es eine Beziehung zwischen den Verteilern und den Adressen. Daher kann das *Mail Distribution Center* und der *Web Service* E-Mails auf Verteiler hinzufügen oder von einem löschen. Weiterhin besteht die Möglichkeit alle E-Mail Adressen eines Verteilers mit den zugehörigen Fingerprints abzufragen und einen einzelnen Fingerprint zu erhalten oder zu ändern. Die genutzte Schnittstelle *UtilityDistributerManagementInterface* dient nur für die Nutzung eines Loggings sofern keine Verbindung zur Datenbank aufgebaut werden kann.

Distributer Public Key Management

Die *Distributer Public Key Management* Komponente besitzt den Zugriff auf den Schlüsselserver. Dieser findet über die Web-Schnittstelle des Servers statt. Damit können Schlüssel hinzugefügt, gelöscht oder angezeigt werden. Mit den beiden angebotenen Schnittstellen *DistributerPublicKeyMDCInterface* und *DistributerPublicKeyWSInterface* werden diese Dienste den beiden Komponenten zur Verfügung gestellt. Die Notwendigkeit der Verwendung der *Utility* Komponente ist ebenfalls auf das Logging begründet sowie das dynamische Ermitteln der eigenen lokalen IP-Adresse für den Serverzugriff.

Utility

Außer der *Web Service* Komponente bietet die *Utility* Komponente allen anderen hilfreiche Funktionalitäten an. Hauptsächlich ist sie für die Generierung von E-Mails in diversen MIME-Formaten und dem Extrahieren von spezifischen PGP-Inhalten aus den E-Mails zuständig. Ebenfalls bietet sie ein Logging für alle Komponenten an, um spezielle Systemfehler zu protokollieren.

Web Service

Mit dem *Web Service* besteht zusätzlich die Möglichkeit für einen Administrator manuell E-Mail Adressen auf einen Verteiler hinzuzufügen oder zu löschen. Aus diesem Grund muss diese Komponente Zugriff auf die Datenbank und dem Schlüsselserver haben. Ein Anwender darf dies nicht, da keinerlei Authentifizierungsmechanismen vorhanden sind. Dritte könnten daher z.B. E-Mail Adressen ohne Einverständnis des Anwenders von einem Verteiler löschen. Der Administrator muss sich zu Beginn an der Weboberfläche mit einem Benutzernamen und Passwort anmelden. Der Benutzername entspricht der zu bearbeitenden Verteileradresse. Ein individuell zu wählendes Passwort wird bei der Erstellung eines neuen Verteilers in der Datenbank verzeichnet. Mit diesem Zugang kann der authentifizierte Administrator eine anzugebende E-Mail Adresse für den gewählten Verteiler an- oder abmelden. Der öffentliche Schlüssel muss bei der Anmeldung ebenfalls angegeben werden. Er kann zudem bei der Abmeldung der angegebenen E-Mail Adresse aus dem System gelöscht werden. Traten bei der An- oder Abmeldung keinerlei Fehler auf, wird über das *MailDistributionCenterWSInterface* eine E-Mail mit der entsprechenden Information an den E-Mail Adressaten versandt.

5.1.3 Datenbank-Modell

Die zugrunde liegende relationale Datenbank wurde gewählt, da es Beziehungen zwischen den Verteilern und den E-Mail Adressen gibt. Dies lässt sich mit relationalen Datenbanken sehr gut umsetzen. Ebenfalls bieten sie ein striktes Konsistenzmodell an, sodass die Datensätze und ihre Beziehungen untereinander stets korrekt sind. Abbildung 5.3 zeigt das Entity-Relationship-Modell für das System. Es besitzt zwei Haupttabellen für Verteiler und E-Mail Adressen. Die Zwischentabelle *distributer_mail_address* bildet die Beziehung zwischen diesen beiden Tabellen ab. Sie wird durch die automatisch generierten Primärschlüssel *distID* und *maID* hergestellt. In der Zwischentabelle ist ebenfalls der verpflichtende Fingerprint auf exakt 40 Zeichen festgelegt, womit ungültige Werte unzulässig sind. Ein Verteilereintrag muss stets eine in der Datenbank noch nicht vorhandene E-Mails Adresse, ein Passwort sowie einen *Salt*-Wert besitzen. Eine

zusätzliche Bezeichnung für den Verteiler kann zudem angegeben werden. Generell werden E-Mail Adressen nicht länger als 255 Zeichen lang sein, was die Begrenzung der Attribute in den Tabellen begründet. Das Passwort wird verwendet, damit sich der Administrator am *Web Service* anmelden kann und wird als Hashwert in der Datenbank abgelegt. Dafür muss der *Salt*-Wert an das gewählte Passwort konkateniert und anschließend mit SHA-512 gehasht werden. Der *Salt*-Wert dient u.a. dazu, um Wörterbuch-Attacken zu erschweren und ist eine zufällig generierte Zeichenfolge. Bei der Anmeldung des Administrators wird das eingegebene Passwort mit dem aus der Datenbank korrespondierenden *Salt*-Wert konkateniert. Dieser Wert wird ebenfalls mit SHA-512 gehasht und mit dem hinterlegten Passwort aus der Datenbank verglichen. Somit ist ein für die heutige Zeit solides und sicheres Hashverfahren eingesetzt.

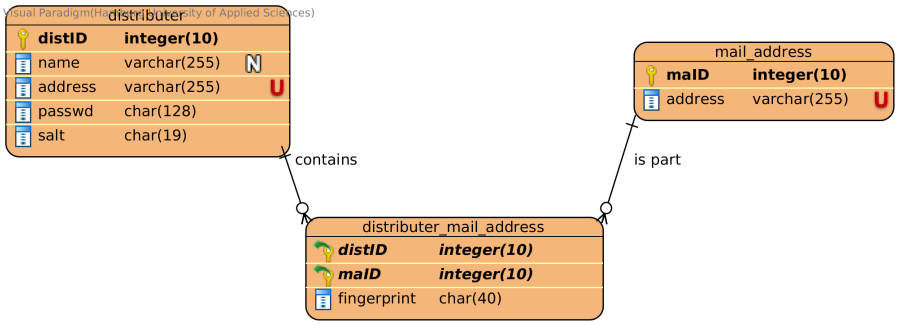

Abbildung 5.3: ER-Modell für den OpenPGP-Verteiler

5.2 Spezifikation

Damit die bereits angesprochene Dezentralität des System gewährleistet sein kann (vgl. funkt. Anforderung 7.0) muss eine entsprechende Hardware-Umgebung genutzt werden. Es wurde daher der Bastelcomputer Raspberry Pi 2 Model B mit einer 16 Gb Speicherkarte gewählt. Dieser verfügt über ausreichend Rechenleistung für ein solches System. Das installierte Betriebssystem ist das Raspbian GNU/Linux Version 7 (wheezy), eine auf Debian basierte Linux-Distribution. Als Programmiersprache wurde Python in der Version 3.4 gewählt. Python ist eine einfach zu erlernende und performante Skriptsprache. Die Version 3.4 wurde aus Kompatibilitätsgründen zu der gnupg-Bibliothek gewählt die im weiteren genauer beschrieben wird.

5.2.1 Externe Schnittstellen

SKS Key-Server

Die SKS Key-Server[1] Implementierung ist ein Open Source Schlüsselserver für GnuPG. Die installierte Version ist 1.1.3 und stammt aus den offiziellen Paketquellen. Der Key-Server ist derzeit so konfiguriert, dass er nur lokal über den Port 11371 erreichbar ist. Damit eine spätere Anbindung an andere Schlüsselserver gewährleistet ist - sofern dies gewollt ist - , werden die Schlüssel nicht in der zugrunde liegenden Datenbank verwaltet. Schlüssel können über HTTP hinzugefügt und abgefragt werden. Eine Statistik über die auf dem Server liegenden Schlüssel kann ebenfalls darüber ermittelt werden. Somit ist es möglich verschiedene Informationen der Schlüssel über die URL zu erfahren. Bspw. kann über die URL *http://<lokale IP-Adresse>:11371/pks/lookup?op=get&search=0x<fingerprint>* ein Schlüssel unter der Verwendung des Fingerprints durch das HTTP mit der GET-Methode im Programmcode oder Browser abgefragt werden. Das Löschen von Schlüsseln muss allerdings über die Kommandozeile geschehen.

MySQL Datenbank

Wie bereits erwähnt, wird eine MySQL[2] Datenbank genutzt. Sie wurde ebenfalls aus den offiziellen Paketquellen installiert und hat die Version 5.5.47. Für die Anwendung wurde eine eigene Datenbank *db_distributer* angelegt. Für die Verwaltung wurde ebenfalls ein eigener Benutzer *www-data* mit eigenem Passwort erstellt. Diesem wurde lediglich der Zugriff auf diese Datenbank gestattet, um die Zugriffe auf andere Datenbanken zu verhindern, bzw. den Zugriff von nicht berechtigten Benutzern zu unterbinden.

Apache Server

Der installierte Apache2-Webserver hat die Version 2.2.22 und wurde ebenfalls aus den offiziellen Paketquellen heruntergeladen. Dieser ist notwendig, damit der - in PHP geschriebene - *Web Service* genutzt werden kann. Damit der Zugriff auch über eine eingerichtete Domäne funktioniert und unterschiedliche Webseiten auf dem Server eingerichtet werden können, muss zudem ein virtueller Host[3] konfiguriert werden.

[1] https://sks-keyservers.net/; zuletzt besucht am 27.07.2016
[2] https://www.mysql.de/; zuletzt besucht am 27.07.2016
[3] https://httpd.apache.org/docs/2.2/de/vhosts/; zuletzt besucht am 27.07.2016

5.2.2 Verwendete Bibliotheken

Das System nutzt einige von Python bereitgestellte Bibliotheken. Im folgenden werden einige wichtige kurz vorgestellt.

Die wichtigste Python Bibliothek für die Inbetriebnahme des Verteiler-Systems ist *gnupg*[4] in der Version 2.0.2. Diese Schnittstelle stellt einige Operationen für GnuPG bereit. Es besteht aus dem Hauptmodul *gnupg*, welches drei weitere Hilfsmodule verwendet: *_meta*, *_parsers* und *_util*. Das *gnupg*-Modul stellt einige GnuPG Operationen in der GPG-Klasse bereit mit denen u.a. Schlüssel erzeugt, exportiert, importiert oder Daten verschlüsselst, entschlüsselt sowie signiert und verifiziert werden können. Bisher ist die Bibliothek noch kein 100%-iges Abbild von GnuPG, da bspw. nicht mehr als ein Subkey bei der Generierung von Schlüsselpaaren erzeugt werden kann. Das *_meta*-Modul enthält Meta- und Basis-Klassen von denen die GPG-Klasse ableitet. Sie verbirgt einige interne Funktionen und sollte zudem nicht direkt genutzt werden. In diesem Modul wird z.B. bei der Verschlüsselung von Daten ein Unterprozess erzeugt, der die notwendigen Befehle an GnuPG leitet und ausführen lässt. Mit dem *_parsers*-Modul werden Benutzereingaben und die Rückgabewerte von GnuPG behandelt. Sobald eine GnuPG-Operation ausgeführt wurde, wird ein entsprechendes, durch das *_parser*-Modul erzeugte, Objekt mit Statusinformationen über die Operation erzeugt. Mit diesen Informationen kann geprüft werden, ob z.B. signierte Daten erfolgreich verifiziert werden konnten. Das *_util*-Modul wird für weitere interne Abläufe im *gnupg*-Modul eingesetzt. Damit werden bspw. Eingabe- und Ausgabe-Operationen gesteuert, Daten (de-)codiert oder ein Logging durchgeführt.

Das *mysqlclient*-Modul[5] in Version 1.3.7 stellt die Schnittstelle zur MySQL-Datenbank bereit. Mit ihr wird ein Datenbank-Objekt erzeugt, mit dem die Daten in der angegebenen Datenbank gepflegt werden können. Da die zuvor erzeugte Datenbank mit einem Passwort geschützt ist, muss dieses bei der Erzeugung einer Datenbankinstanz angegeben werden. Mit dem Modul kann das Passwort sowie weitere Parameter, wie bspw. der Datenbank-Benutzername und der Datenbankname, über eine extra Konfigurationsdatei eingelesen werden (vgl. Abbildung 5.4). Die Zugriffsrechte müssen für die Datei entsprechend angepasst werden, damit nicht jeder

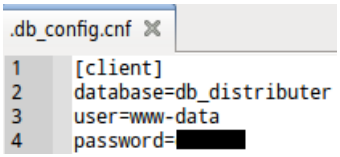

Abbildung 5.4: Konfigurationsdatei für die Datenbank des OpenPGP-Verteilers

[4] https://pythonhosted.org/gnupg/; zuletzt besucht am 27.07.2016
[5] https://mysqlclient.readthedocs.io/en/latest/; zuletzt besucht am 27.07.2016

Benutzer des Systems darauf zugreifen darf. Durch diese Lösung ist das Passwort gut geschützt und steht nicht im Quellcode.

Des Weiteren ist es notwendig die *email*-Bibliothek[6] von Python zu verwenden. Damit lassen sich ankommende E-Mails in Datenobjekte parsen und eigene E-Mails erzeugen. Es unterstützt zudem MIME und weitere im RFC 2822 definierte Internet Nachrichten Formate. Diese Bibliothek ist lediglich für das Nachrichtenformat zuständig. Damit E-Mails empfangen und versendet werden können, muss zudem die *smtplib*-Bibliothek[7] eingesetzt werden. Hierüber wird eine Verbindung zu einem SMTP-Server hergestellt und die E-Mail versandt. Es unterstützt außerdem STARTTLS für die Transportverschlüsselung.

Weitere notwendige Bibliotheken für die Netzwerkkommunikation sind *dnspython3*[8] (Version 1.12.0), *netifaces*[9] (Version 0.10.4) und *requests*[10] (Version 2.9.1). Durch *dnspython3* kann für eine angegebene Domain u.a. der MX-Record aufgelöst werden. Dieser ist beim Versenden der E-Mails notwendig, da darüber der SMTP-Server der Domain angesprochen wird. Mittels *netifaces* können die auf dem System installierten Netzwerkschnittstellen ermittelt werden. Dies ist insofern hilfreich, als dass die Netzwerkadresse nicht direkt in den Quellcode codiert werden muss, sondern dynamisch ermittelt wird. Wie bereits erwähnt wird der SKS Key-Server über HTTP angesprochen. Um dies in Python zu lösen, wird die *request*-Bibliothek eingesetzt. Mit ihr lassen sich HTTP-Anfragen ohne viel Quellcode erzeugen und an einen Server senden.

5.2.3 Fehlerbehandlung

Wie schon angedeutet ist in allen Komponenten - mit Ausnahme der Web Service Komponente - ein Paket für spezifische Exceptions enthalten. Jede Exception bildet eine eigene Klasse die von der Python bereitgestellten Exception-Klasse ableitet. Der Inhalt der eigenen Exception-Klassen besteht bisher nur aus der *__str__(*args, **kwargs)*-Methode die die selbige aus der Exception-Klasse überschreibt und kann individuell erweitert werden. Der Grund für den Einsatz von spezifischen Exceptions ist das individuelle Behandeln der Fehler. Kann bspw. ein Schlüssel nicht auf dem Schlüsselserver gespeichert werden, muss die zuvor in die Datenbank hinzugefügte E-Mail Adresse wieder gelöscht werden (Betreff-Befehl 1). Bisher wird eine Exception mit einem für den Benutzer verständlichen Text ausgelöst. Dieser Text wird nach Auslösen der Exception an den *MTASendThread* weitergeleitet. Dieser erzeugt eine entsprechende E-Mail und schickt sie an den Absender.

[6] https://docs.python.org/3.4/library/email.html; zuletzt besucht am 27.07.2016
[7] https://docs.python.org/3.4/library/smtplib.html; zuletzt besucht am 27.07.2016
[8] http://www.dnspython.org/; zuletzt besucht am 27.07.2016
[9] https://bitbucket.org/al45tair/netifaces; zuletzt besucht am 27.07.2016
[10] http://docs.python-requests.org/en/master/; zuletzt besucht am 27.07.2016

5.2.4 Mögliche Systemerweiterungen

Das Verteiler-System kann an diversen Stellen erweitert und ergänzt werden. Auf der einen Seite sollte es möglich bzw. sogar verpflichten sein, dass auch die Anwender zwei unterschiedliche Schlüsselpaare für ihre Mitgliedschaft auf einem Verteiler verwenden. Ein Schlüssel wird zur Signierung/Verifizierung und der andere zur Verschlüsselung/Entschlüsselung eingesetzt. Der Grund ist identisch mit der Aussage im zweiten Absatz unterhalb der Tabelle in Kapitel 4.2.3 für die funktionale Anforderung 1.6. Auf der anderen Seite sollte ein Verteiler seine beiden Schlüsselpaare in regelmäßigen Abständen erneuern und die öffentlichen Schlüssel propagieren. Dies hängt mit dem PFS-Prinzip und den privaten Schlüsseln, die keinerlei Passphrase zum Entschlüsseln verwenden, zusammen. Sobald der private Schlüssel zum Ver- und Entschlüsseln einem Angreifer vorliegt, können alle mitgeschnittenen E-Mails aus der Vergangenheit entschlüsselt werden. Ebenso kann sich ein Angreifer, mit dem privaten Schlüssel zur Signierung, als Verteiler identifizieren und E-Mails mit falschem Inhalt versenden. Diese Szenarien werden besonders wegen der fehlenden Passphrase realistischer. Eine weitere Erweiterung sind sinnvolle organisatorische Richtlinien für die öffentlichen Schlüssel. Dadurch werden dem Anwender strengere Auflagen aufgezwungen, die die Sicherheit erhöhen. Es sollten bspw. nur Algorithmen verwendet werden die aus kryptographischer Sicht als sicher gelten. Ebenfalls spielt die Schlüssellänge eine wichtige Rolle die entsprechend hoch sein sollte (vgl. [BSI16]).

5.3 Vorhandene Schwachstellen

Das Konzept und die Implementierung des Verteiler-Systems besitzen eine grundlegende Schwachstelle. Es ist theoretisch möglich den Inhalt einer E-Mail die zur *Umverschlüsselung* im System verarbeitet wird als Klartext auszulesen. Dies ist durch die kurzzeitige Entschlüsselung des Nachrichteninhaltes bedingt. Allerdings ist die Ausnutzung dieser Schwachstelle mit viel Aufwand verbunden. Ein Angreifer müsste sich zunächst einen direkten Zugriff auf den Server verschaffen, um die kurzzeitig entschlüsselten E-Mails auslesen zu können. Dieses Szenario stellt aber eine besondere Situation dar. Letztlich gibt es niemals 100%-ige Sicherheit, für den hier verfolgten Zweck ist der erreichte Sicherheitsgrad gleichwohl als ausreichend anzusehen.

Die Entscheidung zur Verwendung von Betreff-Befehlen (vgl. funktionale Anforderung Nr. 1 in Kapitel 4.2.3) kann ebenfalls als Manko gesehen werden. Mit diesen Befehlen aus Tabelle 5.1 ist es möglich Bewegungsprofile zu erzeugen. Daraus ist ersichtlich zu welchem Zeitpunkt sich Anwender z.B. an einem Verteiler an- oder abmelden. Dies ist bereits langjährige Praxis der NSA und wird in einem großen Umfang durchgeführt, wie [Bal13] berichtet.

Weiterhin ist es sinnvoll eine gute Passphrase für den privaten Schlüssel anzugeben, wodurch dieser verschlüsselt im Dateisystem abgelegt wird. Dies ist bei einem automatisierten System aber nicht umzusetzen, da es keinen Anwender gibt der die Passphrase eingibt, um Zugriff auf den Schlüssel zu erhalten. Ein Ablegen der Passphrase in eine Datei ergibt keinen nennenswerten Vorteil. Daher sind alle in diesem System angelegten privaten Schlüssel ohne Passphrase hinterlegt. Sobald diese von einem Angreifer entwendet werden, können sie problemlos genutzt werden.

5.4 Ideen zur Lösung der Schwachstellen

Ein genereller Ansatz der die Massenverschlüsselung mit OpenPGP in einem wichtigen Punkt verbessern würde, ist meiner Meinung nach ein neues Protokoll für die Anwendungsschicht, welches zusätzlich den OpenPGP-Standard verwendet. Dieses sieht zunächst vor, dass der Client und der Server einen symmetrischen Schlüssel aushandeln. Das Aushandeln der Sitzungsschlüssel ist mit heutigen Verfahren, wie z.B. mit dem Diffie-Hellman-Schlüsselaustauschverfahren, problemlos möglich. Im Anschluss erzeugt der Client eine OpenPGP-Paketstruktur (vgl. Abbildung 5.5) und verpackt die zu sendende Nachricht darin. Zur Verschlüsselung der Nachricht wird der zuvor ausgehandelten Schlüssel eingesetzt und nicht lokal beim Client erzeugt, wie es im OpenPGP-Standard definiert ist.

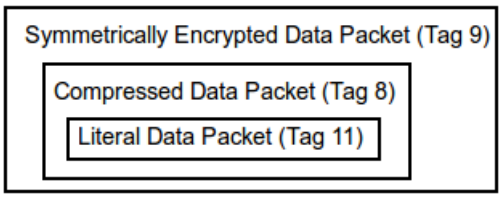

Abbildung 5.5: Verschlüsselte OpenPGP Paketstruktur

Das *Symmetrically Encrypted Data Packet (Tag 9)* repräsentiert somit die mit dem symmetrischen Schlüssel verschlüsselte OpenPGP-Nachricht. Dieses Vorgehen ist identisch mit dem generellen OpenPGP-Verfahren zur Verschlüsselung von OpenPGP-Paketen, außer dass der verwendete Schlüssel vorher von Client und Server ausgehandelt wurde. Danach wird das verschlüsselte Paket an den Server versandt. An dieser Stelle ist es nicht mehr notwendig, dass der Server eine *Umverschlüsselung* vornehmen muss, da er bereits den verwendeten symmetrischen Schlüssel kennt. Nun kann der Server alle öffentlichen Schlüssel der Mitglieder auf dem Verteiler ermitteln und für jedes Mitglied ein individuelles OpenPGP-Paket erzeugen. Es

besteht aus der empfangenen verschlüsselten Paketstruktur sowie einer zusätzlichen Schicht. Diese Schicht ist ein *Public-Key Encrypted Session Key Packet (Tag 1)* in dem der ausgehandelte Schlüssel mit dem jeweiligen öffentlichen Schlüssel eines Mitgliedes verschlüsselt verpackt wird.

Der wichtigste Vorteil bei diesem Ablauf ist, dass die E-Mail vom Sender bis zu deren Empfänger durchgehend Ende-zu-Ende verschlüsselt ist. Ebenso hat es den Vorteil, dass das Schlüsselpaar des Servers zum Ver- und Entschlüsseln nicht mehr benötigt wird. Dadurch wird ebenfalls der Nachteil einer fehlenden Passphrase automatisch gelöst. Auf der anderen Seite gibt es zum einen den Nachteil, dass ein neues Protokoll entwickelt und in die E-Mail Clients eingebunden werden muss. Zum anderen kann eine mögliche vorhandene Signatur nur aufwendig verifiziert werden. Die Verifizierung der E-Mails hat in der vorgestellten Lösung in dieser Arbeit der Server übernommen. Dies ist mit diesem Konzept allerdings nicht mehr möglich, da u.a. die Signatur mit dem ausgehandelten symmetrischen Schlüssel verschlüsselt ist und der Server diese nicht mehr entschlüsselt. Daher müssen die Empfänger den öffentlichen Schlüssel des Senders besitzen, um eine Verifikation durchführen zu können. Diese Problematik hat wiederum den Nachteil, dass der Einsatz eines Verteiler-Systems zunichte gemacht wird, da das System u.a. wegen der aufwendigen Schlüsselverwaltung eingesetzt wird.

5.5 Vergleich der vorgestellten Lösungen

In Tabelle 5.2 sind alle in Kapitel 4 und 5 vorgestellten Lösungen für ein Verteiler-System gegenübergestellt.

Positiv hervorzuheben ist die Offenlegung des Quellcodes von drei der fünf Lösungen. Das Konzept in Kapitel 4.1.4 ist allerdings sehr neu und aus [Wal16] kann man durchaus entnehmen, dass dieses auch quelloffen werden soll. Zentrale und dezentrale Lösungen haben sowohl Vor- als auch Nachteile. Zentrale Lösungen bieten dem Anwender die Möglichkeit sich schnell und mit wenig Aufwand einen Verteiler zu erstellen. Andererseits muss man diesen Systemen vollstes Vertrauen für die Verarbeitung von E-Mails geben. Anwender mit etwas Wissen und Interesse können ein dezentrales System schnell zu Hause einrichten und in Betrieb nehmen. Wie aus der eigenen Entwicklung eines Verteiler-Systems auf Basis der *Umverschlüsselung* zu erkennen ist hat dies einen entscheidenden Nachteil. Eine Ende-zu-Ende-Verschlüsselung ist zwar gegeben, allerdings nur zwischen Sender und Server sowie zwischen Server und Empfänger. Daher besteht immer die Gefahr, dass der Klartext der E-Mail auf dem Server von Dritten eingesehen werden kann. Idealerweise sollte die Ende-zu-Ende-Verschlüsselung, ohne eine weitere Instanz dazwischen, zwischen Sender und Empfänger

Bezeichnung	Open-Source	Lokalität	Standard	Umverschlüsselung	Schlüsselserveranbindung	Signierung	PFS
4.1.1 Schleuder	Ja	Zentral	OpenPGP	Ja	Nein	freiwillig	Nein
4.1.2 GPGrelay	Ja	Dezentral	OpenPGP	Nein	Nein	freiwillig	Nein
4.1.3 Sichere E-Mail-Verteiler: Ein praxisorientierte Ansatz	Nein	Zentral	S/MIME	Ja	Nein	freiwillig	Nein
4.1.4 Practical Encrypted Mailing Lists	unbekannt	Dezentral	OpenPGP	Nein	Ja	freiwillig	Nein
Eigene Implementierung	Ja	Dezentral	OpenPGP	Ja	möglich	verpflichtend	Nein

Tabelle 5.2: Vergleich der vorgestellten Lösungen

gelten. Die Schlüsselserveranbindung hat neben einer effektiven Schlüsselverteilung auch einen nicht zu ignorierenden Nachteil. Die auf den Servern hinterlegten Schlüssel sind nicht verifiziert, sodass der eindeutige Ursprung eines Schlüssels nicht ersichtlich ist. Innerhalb eines Verteiler-Systems kann dies allerdings gewährleistet werden, indem der Anwenderschlüssel beim Eintragen einer Adresse auf einem Verteiler signiert wird. Die Signierung der E-Mails ist bei allen anderen Lösungen - als der in dieser Arbeit behandelten - nur freiwillig und stellt einen großen Nachteil dar. Ein weiterer wichtiger Punkt ist auch hier das PFS-Prinzip. Dies ist durch die verwendeten Standards der Lösungen bedingt, da die Schlüsselpaare generell für eine längere Einsatzzeit generiert und verwendet werden. Dadurch führt es zwangsläufig dazu, dass die Verfahren *Imperfect Forward Secrecy* sind.

6 Abschließende Bemerkungen

6.1 Fazit

In dieser Arbeit ging es zum einen um allgemein vorhandene Lösungen zur verschlüsselten und authentifizierten Kommunikation in Netzwerken, zum anderen um die Massenverschlüsselung von E-Mails im Speziellen. Zunächst wurde ein Überblick über einige Netzwerkgrundlagen gegeben. Das OSI-Modell stellt eine theoretische Einteilung der Netzwerkkommunikation in Schichten dar. Hier wurden drei Schichten aus dem Modell betrachtet und ihre konkreten Aufgaben beschrieben. Auf der Anwendungsschicht gibt es eine Vielzahl von Protokollen, welche die Kommunikation von Anwendungen abwickeln. Die Transportschicht sorgt dafür, dass die Daten der Anwendungsschicht an den korrekten Empfänger geleitet werden. Hierfür sind u.a. die beiden Protokolle TCP und UDP zuständig, welche ausführlich beschrieben wurden. Die darunter liegende Netzwerkschicht erhält die Segmente von der Transportschicht und leitet die Pakete über die verschiedenen Komponenten (z.B. Router) zur Zielkomponente.

Nach den Grundlagen der Netzwerkkommunikation wurden drei Kategorien von kryptographischen Algorithmen und ihre Einsatzzwecke beschrieben. Zusätzlich wurden jeweils zwei Vertreter aus diesen Bereichen vorgestellt und in ihrer Sicherheit eingeschätzt. Im Anschluss wurde das TLS- bzw. IPsec-Protokoll zur Verschlüsselung und Authentifizierung der Daten auf der Transport- bzw. Netzwerkschicht detailliert erörtert. Es hat sich herauskristallisiert, dass sie die zur Zeit bedeutsamsten Vertreter auf diesen Schichten im Bereich der Verschlüsselung sind.

Im Anschluss daran wurden bestehende Lösungen für die E-Mail Verschlüsselung dargestellt. OpenPGP und S/MIME implementieren eine Ende-zu-Ende-Verschlüsselung auf der Anwendungsschicht und sind derzeit die am weit verbreitetsten Vertreter, während De-Mail in jedem Fall eine Transportverschlüsselung durchführt und eine Ende-zu-Ende-Verschlüsselung optional anbietet. SMTP/TLS hingegen verschlüsselt den Weg von E-Mails durch ein Netzwerk auf der Transportschicht. Alle vier Lösungen wurden in ihrem Konzept detailliert vorgestellt.

Hauptziel dieser Arbeit war die Entwicklung eines eigenen OpenPGP-Mailing-Verteiler-Systems zur Massenverschlüsselung und dessen Vergleich zu anderen bestehenden Systemen.

Deshalb wurden zunächst in Kapitel 4.1 bestehende Lösungen für ein solches System kurz vorgestellt.

Mithilfe der ausgearbeiteten Anforderungen in Kapitel 4.2 wurde zunächst eine theoretische Basis für das System geschaffen. Auf Grundlage dieser Erkenntnisse konnte ein Entwurf angefertigt werden, der die nötigen Komponenten und deren Abhängigkeiten untereinander und zu anderen eingesetzten Werkzeugen aufzeigt. Nachdem der Entwurf fertiggestellt war, konnte das System in der Programmiersprache Python umgesetzt und auf einem Raspberry Pi 2 erfolgreich eingesetzt werden. Zudem ist die Implementierung im Internet frei zugänglich und kann von anderen Entwicklern verwendet, korrigiert und erweitert werden. Durch die erfolgreiche Inbetriebnahme auf einem Raspberry Pi 2 wurde ein dezentrales System entwickelt, welches ohne großen Aufwand verwendet werden kann.

Bei der Umsetzung war zu erkennen, dass der Entwurf an einigen Stellen nachgebessert werden musste. Somit ergab sich im Laufe der Entwicklung des Systems ein zyklischer Ablauf, der gewonnene Erkenntnisse in den Entwurf nachträglich einfließen ließ. Durch diese strukturelle und systematische Vorgehensweise konnte erfolgreich ein OpenPGP-Mailing-Verteiler-System umgesetzt werden, welcher die Anforderungen vollständig berücksichtigte. Insgesamt konnten die in Kapitel 1.2 dargelegten Zielsetzungen mit Erfolg erreicht werden.

Bei der Anfertigung der Anforderungen und des Entwurfs stellte sich heraus, dass das System in einigen Punkten erweitert oder ergänzt werden kann. Diese Punkte wurden im Laufe der Arbeit kurz angerissen und können zusätzlich implementiert werden. Weiterhin war festzustellen, dass das System einige Schwachstellen aufweist, welche ebenfalls erläutert wurden. Eine dieser Schwachstellen ist die *Umverschlüsselung* auf dem Server. Dieser Umstand führte zu der Erkenntnis, dass keine durchgehende Ende-zu-Ende-Verschlüsselung vorliegt, da eine gesendete E-Mail kurzzeitig im Klartext auf dem Server vorzufinden ist. Dies stellt ein Defizit dar, welches die Sicherheit aber nicht in einem hohen Maß beeinträchtigt und mit der vorgestellten Lösung in Kapitel 5.4 beseitigt werden kann. Diese Lösung sieht ein neues Protokoll auf der Anwendungsschicht vor, welches in Kombination mit OpenPGP die *Umverschlüsselung* umgeht.

Zuletzt wurden alle Lösungen für ein Mailing-Verteiler-System tabellarisch gegenübergestellt. Es zeigte sich, dass jede Lösung für sich genommen seine Vor- und Nachteile besitzt. In dem Vergleichspunkt PFS weisen allerdings alle Systeme eine zur Zeit viel diskutierte Schwachstelle auf, welches auf die zugrunde liegenden Verschlüsselungsstandards OpenPGP bzw. S/MIME zurück zu führen ist. Weiterhin stellte sich bei der Gegenüberstellung heraus, dass ein Großteil der diskutierten Lösungen keine durchgehende Ende-zu-Ende-Verschlüsselung umsetzt. Um sichere und durchgehend Ende-zu-Ende verschlüsselte Systeme zu entwerfen,

sollte man daher zukünftig andere Ansätze verfolgen, welche das PFS-Problem beseitigen und im besten Fall eine *Umverschlüsselung* vermeiden.

6.2 Ausblick

Anhand dieser Arbeit kann man sehen, dass es bereits einige Ansätze zur Massenverschlüsselung von E-Mails gibt. Sie basieren auf praktisch sicheren und soliden kryptographischen Standards und unterscheiden sich an einigen wenigen Stellen. Dennoch werden die herkömmlichen Standards für eine Ende-zu-Ende-Verschlüsselung - wie OpenPGP oder S/MIME - in der heutigen digitalen Welt nur sehr begrenzt eingesetzt. Hauptgrund hierfür ist die fehlende Benutzerakzeptanz. Die meisten Anwender schrecken vom Einsatz von OpenPGP und S/MIME zurück. Die Bereitschaft, sich mit technischen Einzelheiten zu befassen, ist gering. Vielen Menschen erscheint E-Mail Verschlüsselung auch heutzutage noch als zu kompliziert. Dies umfasst sowohl die Installation der nötigen Software als auch die korrekte Anwendung. Die in der Regel erforderlichen Passworteingaben zum Einsehen des E-Mail Inhaltes werden vor allem als lästig empfunden.

Phil Zimmermann sagte einmal: „Privacy is a right like any other. You have to exercise it or risk losing it.“[1]. Zum Schutz der Privatsphäre in der digitalen Welt, braucht es zunächst die Akzeptanz der Anwender. Darüber hinaus ist das Vorhandensein von einfacher und benutzerfreundlicher Software jedoch unerlässlich. Daher muss zukünftige Software entsprechend dieser Vorgaben ausgerichtet und konzipiert sein.

Bezogen auf die Massenverschlüsselung von Textnachrichten und Telefongesprächen gibt es z.B. die von Open Whisper Systems entwickelte Software Signal[2]. Diese setzt automatisch eine Ende-zu-Ende-Verschlüsselung ein, sodass der Anwender keinen zusätzlichen Aufwand betreiben muss, um seine Privatsphäre zu schützen. Das Signal-Protokoll[3] berücksichtigt zudem das PFS-Prinzip, da keine Langzeitschlüssel wie bei OpenPGP oder S/MIME verwendet werden. Für die Zukunft könnte auf Grundlage dieses Protokolls ein Konzept für eine Ende-zu-Ende-Verschlüsselung von E-Mails entwickelt werden. Mit einer derart benutzerfreundlichen Anwendung könnte eine flächendeckende Verschlüsselung von E-Mails eines Tages Wirklichkeit werden.

[1] Javier Bernal; Issue Seven: Religion Online & Techno-Spiritualism; University of Lincolnshire & Humberside; http://www.cybersociology.com/files/7_bigbrotheronline.html; zuletzt besucht am 27.07.2016
[2] https://whispersystems.org/blog/just-signal/; zuletzt besucht am 27.07.2016
[3] https://whispersystems.org/blog/advanced-ratcheting/; zuletzt besucht am 27.07.2016

Literaturverzeichnis

[20115] Statistisches Bundesamt Wiesbaden 2015. Wirtschaftsrechnungen Private Haushalte in der Informationsgesellschaft - Nutzung von Informations- und Kommunikationstechnologien. https://www.destatis.de/DE/Publikationen/Thematisch/EinkommenKonsumLebensbedingungen/PrivateHaushalte/PrivateHaushalteIKT2150400147004.pdf, 2015. [Zuletzt besucht am 27.07.2016].

[ABD+15] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. Technical report, 2015. https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf. [Zuletzt besucht am 27.07.2016].

[Asl] AslamKarachiwala. Secureapt - all about secure apt. https://wiki.debian.org/SecureApt. [Zuletzt besucht am 27.07.2016].

[AW04] B. Aboba and W.Dixon. *IPsec-Network Address Translation (NAT) Compatibility Requirements*. Internet Engineering Task Force (IETF), März 2004. https://tools.ietf.org/html/rfc3715. [Zuletzt besucht am 27.07.2016].

[Bal13] James Ball. NSA stores metadata of millions of web users for up to a year, secret files show. https://www.theguardian.com/world/2013/sep/30/nsa-americans-metadata-year-documents, 2013. [Zuletzt besucht am 27.07.2016].

[Bec16] Markus Beckedahl. Logo ist vor Einigung fertig! Safe Harbour 2.0 heißt jetzt Privacy Shield! (Update). https://netzpolitik.org/2016/logo-ist-vor-einigung-fertig-safe-harbour-2-0-heisst-jetzt-privacy-shield/, 2016. [Zuletzt besucht am 27.07.2016].

[Bis15] Anna Biselli. De-Mail: Das tote Pferd wird weitergeritten, wie viel das kostet, soll geheim bleiben. https://netzpolitik.org/2015/

de-mail-das-tote-pferd-wird-weitergeritten-wieviel-das-kostet-soll-geheim-bleiben/, 2015. [Zuletzt besucht am 27.07.2016].

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. Technical report, K.U. Leuven, Belgium; Microsoft Research Redmond, USA; ENS Paris and Chaire France Telecom, France, 2011. https://eprint.iacr.org/2011/449.pdf. [Zuletzt besucht am 27.07.2016].

[Ble98] Daniel Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. Technical report, Bell Laboratories, 1998. http://archiv.infsec.ethz.ch/education/fs08/secsem/Bleichenbacher98.pdf. [Zuletzt besucht am 27.07.2016].

[Bor15] Detlef Borchers. De-Mail integriert Ende-zu-Ende-Verschlüsselung mit PGP. http://www.heise.de/newsticker/meldung/De-Mail-integriert-Ende-zu-Ende-Verschluesselung-mit-PGP-2570632.html, 2015. [Zuletzt besucht am 27.07.2016].

[Bra89] R. Braden. *Requirements for Internet Hosts – Communication Layers*. Internet Engineering Task Force (IETF), Oktober 1989. https://tools.ietf.org/html/rfc1122. [Zuletzt besucht am 27.07.2016].

[BSI16] BSI. Kryptographische verfahren: Empfehlungen und schlussellängen. Technical report, Bundesamt für Sicherheit in der Informationstechnik, 2016. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf. [Zuletzt besucht am 27.07.2016].

[Bun10] Deutscher Bundestag. Entwurf eines Gesetzes zur Regelung von De-Mail-Diensten und zur Änderung weiterer Vorschriften: Drucksache 17/3630. http://dip21.bundestag.de/dip21/btd/17/041/1704145.pdf, 2010. [Zuletzt besucht am 27.07.2016].

[Bun11] Bundesgesetzblatt. Gesetz zur Regelung von De-Mail-Diensten und zur Änderung weiterer Vorschriften. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/De_Mail/Gesetz-De-Mail.pdf, 2011. [Zuletzt besucht am 27.07.2016].

[CDF+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. *OpenPGP Message Format*. Internet Engineering Task Force (IETF), November 2007. https://tools.ietf.org/html/rfc4880. [Zuletzt besucht am 27.07.2016].

[Cro09] D. Crocker. *Internet Mail Architecture.* Internet Engineering Task Force (IETF), Juli 2009. https://tools.ietf.org/html/rfc5598. [Zuletzt besucht am 27.07.2016].

[DA99] T. Dierks and C. Allen. *The TLS Protocol Version 1.0.* Internet Engineering Task Force (IETF), Januar 1999. https://tools.ietf.org/html/rfc2246. [Zuletzt besucht am 27.07.2016].

[DAM+15] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. Neither Snow Nor Rain Nor MITM...An Empirical Analysis of Email Delivery Security. Technical report, 2015. http://conferences2.sigcomm.org/imc/2015/papers/p27.pdf. [Zuletzt besucht am 27.07.2016].

[Dav01] Don Davis. Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. http://world.std.com/~dtd/sign_encrypt/sign_encrypt7.PDF, 2001. [Zuletzt besucht am 27.07.2016].

[dBB93] Bert den Boer and Antoon Bosselaers. Collisions for the compression function of MD5. Technical report, Philips Crypto B.V und ESAT Laboratory, K.U. Leuven, 1993. https://securewww.esat.kuleuven.be/cosic/publications/article-143.pdf. [Zuletzt besucht am 27.07.2016].

[DBEB06] M. Duke, R. Braden, W. Eddy, and E. Blanton. *A Roadmap for Transmission Control Protocol (TCP) Specification Documents.* Internet Engineering Task Force (IETF), September 2006. https://tools.ietf.org/html/rfc4614. [Zuletzt besucht am 27.07.2016].

[DH98] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification.* Internet Engineering Task Force (IETF), Dezember 1998. https://tools.ietf.org/html/rfc2460. [Zuletzt besucht am 27.07.2016].

[dI15] Bundesministerium des Innern. De-Mail-Leitfaden für Behörden. http://www.cio.bund.de/SharedDocs/Publikationen/DE/Innovative-Vorhaben/De-Mail/De-Mail_Leitfaden.pdf, 2015. [Zuletzt besucht am 27.07.2016].

[dI16] Bundesministerium des Innern. De-Mail-Anbieter werden. http://www.cio.bund.de/Web/DE/Innovative-Vorhaben/De-Mail/De-Mail-Anbieter-werden/de_mail_anbieter_werden_node.html, 2016. [Zuletzt besucht am 27.07.2016].

[DR08] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. Internet Engineering Task Force (IETF), August 2008. https://tools.ietf.org/html/rfc5246. [Zuletzt besucht am 27.07.2016].

[Eck12] Claudia Eckert. *IT-Sicherheit – Konzepte, Verfahren, Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, 2012.

[EE07] R. Siemborski Ed. and A. Melnikov Ed. *SMTP Service Extension for Authentication*. Internet Engineering Task Force (IETF), Juli 2007. https://tools.ietf.org/html/rfc4954. [Zuletzt besucht am 27.07.2016].

[Ess14] Prof. Bernhard Esslinger. Sichere E-Mail mit S/MIME - Eine Anleitung aus Anwenderperspektive. *Datenschutz und Datensicherheit*, pages 305–313, Mai 2014.

[ETLR01] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. *MIME Security with OpenPGP*. Internet Engineering Task Force (IETF), August 2001. https://tools.ietf.org/html/rfc3156. [Zuletzt besucht am 27.07.2016].

[Exp16] Export.gov. U.S.-EU SAFE HARBOR LIST. https://safeharbor.export.gov/list.aspx, 2016. [Zuletzt besucht am 27.07.2016].

[fdDudI] Die Bundesbeauftragte für den Datenschutz und die Informationsfreiheit. Safe Harbor. https://www.bfdi.bund.de/DE/Europa_International/International/Artikel/SafeHarbor.html. [Zuletzt besucht am 27.07.2016].

[fdDudI16] Die Bundesbeauftragte für den Datenschutz und die Informationsfreiheit. Datenschutz-Wiki: Safe Harbor. https://www.bfdi.bund.de/bfdi_wiki/index.php/Safe_Harbor, 2016. [Zuletzt besucht am 27.07.2016].

[FKL+01] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting. *Improved Cryptanalysis of Rijndael*, pages 213–230. Springer-Verlag Berlin Heidelberg New-York, 2001. http://link.springer.com/content/pdf/10.1007%2F3-540-44706-7_15.pdf. [Zuletzt besucht am 27.07.2016].

[Fox08] Dirk Fox. GPGrelay Die 'lokale Poststelle'. *Datenschutz und Datensicherheit*, pages 684–687, Oktober 2008.

[FS03] Niels Ferguson and Bruce Schneier. A Cryptographic Evaluation of IPsec. Technical report, Counterpane Internet Security, Inc., 2003. https://www.schneier.com/cryptography/paperfiles/paper-ipsec.pdf. [Zuletzt besucht am 27.07.2016].

[fSidI11] Bundesamt für Sicherheit in der Informationstechnik. Akkreditierung von De-Mail-Diensteanbietern. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/De_Mail/De-Mail-Akkreditierung-Prozessuebersicht.pdf, 2011. [Zuletzt besucht am 27.07.2016].

[fSidI14] Bundesamt für Sicherheit in der Informationstechnik. De-Mail Sicherer elektronischer Nachrichtenverkehr - einfach und nachweisbar. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Broschueren/De-Mail-Broschuere.pdf, 2014. [Zuletzt besucht am 27.07.2016].

[fSidI16] Bundesamt für Sicherheit in der Informationstechnik. Postfach- und Versanddienst Funktionalitätsspezifikation. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/De_Mail/TR_De_Mail_PVD_FU.pdf, 2016. [Zuletzt besucht am 27.07.2016].

[gCG] g10 Code GmbH. The GNU Privacy Guard. https://www.gnupg.org/. [Zuletzt besucht am 27.07.2016].

[Gel10] Sven Gelzhäuser. Erfolgreiches De-Mail Pilotprojekt: Teilnehmer ziehen Bilanz. *Datenschutz und Datensicherheit*, pages 646–648, September 2010.

[GK11] R. Gellens and J. Klensin. *Message Submission for Mail.* Internet Engineering Task Force (IETF), November 2011. https://tools.ietf.org/html/rfc6409. [Zuletzt besucht am 27.07.2016].

[Gmba] 1&1 Mail & Media GmbH. Verschlüsselte Kommunikation. https://hilfe.gmx.net/sicherheit/pgp.html. [Zuletzt besucht am 27.07.2016].

[Gmbb] Intevation GmbH. Gpg4win. https://www.gpg4win.de/. [Zuletzt besucht am 27.07.2016].

[Gmb10] TC Trust Center GmbH. *TC TrustCenter-Zertifizierungsrichtlinien*, Januar 2010. http://www.trustcenter.de/media/CPD-TCTrustCenter-de.pdf. [Zuletzt besucht am 27.07.2016].

[Gmb14] 1&1 Mail & Media GmbH. De-Mail Preisliste. https://ident.gmx.net/document/price-list/109223, 2014. [Zuletzt besucht am 27.07.2016].

[Gmb16] 1&1 Mail & Media GmbH. GMX Hilfe. https://hilfe.gmx.net/demail/ident.html, 2016. [Zuletzt besucht am 27.07.2016].

[GMP13] Glenn Greenwald, Ewen MacAskill, and Laura Poitras. Edward Snowden: the whistleblower behind the NSA surveillance revelations. http://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance?CMP=twt_gu, 2013. [Zuletzt besucht am 27.07.2016].

[Gre14] Matthew Green. What's the matter with PGP? http://blog.cryptographyengineering.com/2014/08/whats-matter-with-pgp.html, 2014. [Zuletzt besucht am 27.07.2016].

[Hasnt] Jens Hasselbach. Sichere E-Mail-Verteiler: Ein praxisorientierter Ansatz. Technical report, Frauenhofer AEMT - Security for Virtual Goods, Jahr unbekannt. http://subs.emis.de/LNI/Proceedings/Proceedings36/GI-Proceedings.36-10.pdf. [Zuletzt besucht am 27.07.2016].

[Hea14] Heartbleed. The Heartbleed Bug. http://heartbleed.com/, 2014. [Zuletzt besucht am 27.07.2016].

[Hof98] Paul Hoffman. *SMTP Service Extension for Secure SMTP over TLS*, Dezember 1998. https://tools.ietf.org/html/draft-hoffman-smtp-ssl-07. [Zuletzt besucht am 27.07.2016].

[Hof02] P. Hoffman. *SMTP Service Extension for Secure SMTP over Transport Layer Security.* Internet Engineering Task Force (IETF), Februar 2002. https://tools.ietf.org/html/rfc3207. [Zuletzt besucht am 27.07.2016].

[Hol04] S. Hollenbeck. *Transport Layer Security Protocol Compression Methods.* Internet Engineering Task Force (IETF), Mai 2004. https://tools.ietf.org/html/rfc3749. [Zuletzt besucht am 27.07.2016].

[Hou09] R. Housley. *Cryptographic Message Syntax (CMS).* Internet Engineering Task Force (IETF), September 2009. https://tools.ietf.org/html/rfc5652. [Zuletzt besucht am 27.07.2016].

[HPS00] Heiko Häckelmann, Hans J. Petzold, and Susanne Strahringer. *Kommunikationssysteme - Technik und Anwendungen.* Springer-Verlag Berlin Heidelberg GmbH, 2000.

[HSV+05] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M.Stenberg. *UDP Encapsulation of IPsec ESP Packets.* Internet Engineering Task Force (IETF), Januar 2005. https://tools.ietf.org/html/rfc3948. [Zuletzt besucht am 27.07.2016].

[Inc99] Network Associates Inc. How PGP works. http://www.pgpi.org/doc/pgpintro/, 1990 bis 1999. [Zuletzt besucht am 27.07.2016].

[Int81] Internet Engineering Task Force (IETF). *INTERNET PROTOCOL - PROTOCOL SPECIFICATION*, September 1981. https://tools.ietf.org/html/rfc791. [Zuletzt besucht am 27.07.2016].

[ITWa] ITWissen. Definition: Protokoll. http://www.itwissen.info/definition/lexikon/Protokoll-protocol.html. [Zuletzt besucht am 27.07.2016].

[ITWb] ITWissen. Definition: Verbindungslos. http://www.itwissen.info/definition/lexikon/Verbindungslos-CL-connectionless.html. [Zuletzt besucht am 27.07.2016].

[ITWc] ITWissen. Definition: Verbindungsorientiert. http://www.itwissen.info/definition/lexikon/Verbindungsorientiert-CO-connection-oriented.html. [Zuletzt besucht am 27.07.2016].

[Joh] Andreas John. GPGrelay. http://sites.inka.de/tesla/gpgrelay.html. [Zuletzt besucht am 27.07.2016].

[Jä15] Moritz Jäger. PGP: Hochsicher, kaum genutzt, völlig veraltet. http://www.golem.de/news/pgp-hochsicher-kaum-genutzt-voellig-veraltet-1506-114797-3.html, 2015. [Zuletzt besucht am 27.07.2016].

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. Technical report, 2010. https://eprint.iacr.org/2010/006.pdf. [Zuletzt besucht am 27.07.2016].

[Kal98] B. Kaliski. *PKCS#7: Cryptographic Message Syntax Version 1.5.* Internet Engineering Task Force (IETF), März 1998. https://tools.ietf.org/html/rfc2315. [Zuletzt besucht am 27.07.2016].

[Ken05a] S. Kent. *IP Authentication Header.* Internet Engineering Task Force (IETF), Dezember 2005. https://tools.ietf.org/html/rfc4302. [Zuletzt besucht am 27.07.2016].

[Ken05b] S. Kent. *IP Encapsulating Security Payload (ESP).* Internet Engineering Task Force (IETF), Dezember 2005. https://tools.ietf.org/html/rfc4303. [Zuletzt besucht am 27.07.2016].

[KHN+14] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. *Internet Key Exchange Protocol Version 2 (IKEv2).* Internet Engineering Task Force (IETF), Oktober 2014. https://tools.ietf.org/html/rfc7296. [Zuletzt besucht am 27.07.2016].

[Kle08] J. Klensin. *Simple Mail Transfer Protocol.* Internet Engineering Task Force (IETF), Oktober 2008. https://tools.ietf.org/html/rfc5321. [Zuletzt besucht am 27.07.2016].

[Kom16] Europäische Kommission. Europäische Kommission lanciert EU-US-Datenschutzschild: besserer Schutz für den transatlantischen Datenverkehr. http://europa.eu/rapid/press-release_IP-16-2461_de.htm, 2016. [Zuletzt besucht am 27.07.2016].

[KR14] James Kurose and Keith Ross. *Computernetzwerke - Der Top-Down-Ansatz.* Pearson Deutschland GmbH, 2014.

[KS05] S. Kent and K. Seo. *Security Architecture for the Internet Protocol.* Internet Engineering Task Force (IETF), Dezember 2005. https://tools.ietf.org/html/rfc4301. [Zuletzt besucht am 27.07.2016].

[Lec11] Dr. Jens Lechtenbörger. Zur Sicherheit von De-Mail. *Datenschutz und Datensicherheit*, pages 268–269, April 2011.

[Mar15] Moxie Marlinspike. GPG And Me. http://thoughtcrime.org/blog/gpg-and-me/, 2015. [Zuletzt besucht am 27.07.2016].

[Moc87] P. Mockapetris. *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION.* Internet Engineering Task Force (IETF), November 1987. https://tools.ietf.org/html/rfc1035. [Zuletzt besucht am 27.07.2016].

[Neu13] Linus Neumann. Bullshit made in Germany - So hosten Sie Ihre De-Mail, E-Mail und Cloud direkt beim BND! https://media.ccc.de/v/30C3_-_5210_-_de_-_saal_g_-_201312282030_-_bullshit_made_in_germany_-_linus_neumann, 2013. [Zuletzt besucht am 27.07.2016].

[oST02] National Institute of Standards and Technology. *Federal Information Processing Standards Publication 180-2 - SECURE HASH STANDARD*, August 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf. [Zuletzt besucht am 27.07.2016].

[oST15] National Institute of Standards and Technology. *Federal Information Processing Standards Publication 180-4 - Secure Hash Standard (SHS)*, August 2015.

http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. [Zuletzt besucht am 27.07.2016].

[Pro] The Enigmail Project. A simple interface for OpenPGP email security. https://www.enigmail.net/index.php/en/. [Zuletzt besucht am 27.07.2016].

[Ram99] B. Ramsdell. *S/MIME Version 3 Message Specification.* Internet Engineering Task Force (IETF), Juni 1999. https://tools.ietf.org/html/rfc2633. [Zuletzt besucht am 27.07.2016].

[Res08] P. Resnick. *Internet Message Format.* Internet Engineering Task Force (IETF), Oktober 2008. https://tools.ietf.org/html/rfc5322. [Zuletzt besucht am 27.07.2016].

[Res16] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3.* RTFM, Inc., August 2016. https://tlswg.github.io/tls13-spec/. [Zuletzt besucht am 27.07.2016].

[RLM14] Wolfgang Riggert, Michael Lutz, and Christian Märtin. *Rechnernetze - Grundlagen, Ethernet, Internet.* Hanser Verlag GmbH und Co. KG, 2014.

[RT10] B. Ramsdell and S. Turner. *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification.* Internet Engineering Task Force (IETF), Januar 2010. https://tools.ietf.org/html/rfc5751. [Zuletzt besucht am 27.07.2016].

[Sch] Schleuder. GPG-Mailingliste für eine sichere Gruppenkommunikation. https://schleuder2.nadir.org/. [Zuletzt besucht am 27.07.2016].

[Sch06] Jürgen Scherff. *Grundkurs Computernetze - Eine kompakte Einführung in die Rechnerkommunikation - Anschaulich, verständlich, praxisnah.* Friedr. Vieweg und Sohn Verlag | GWV Fachverlage GmbH, 2006.

[Sch09] Jürgen Schmidt. E-Mail-Verschlüsselung austesten - Diagnose von POP3, IMAP und SMTP via SSL. http://www.heise.de/security/artikel/E-Mail-Verschluesselung-austesten-785451.html, 2009. [Zuletzt besucht am 27.07.2016].

[Sch13a] Klaus Schmeh. *Kryptografie -Verfahren, Protokolle, Infrastruktur.* dpunkt.verlag GmbH, 2013.

[Sch13b] Jürgen Schmidt. Zukunftssicher verschlüsseln mit Perfect Forward Secrecy. http://www.heise.de/security/artikel/

Zukunftssicher-Verschluesseln-mit-Perfect-Forward-Secrecy-1923800.html, 2013. [Zuletzt besucht am 27.07.2016].

[Sch14] Jörg Schwenk. *Sicherheit und Kryptographie im Internet – Theorie und Praxis*. Springer Vieweg, 2014.

[SHSA15] Y. Sheffer, R. Holz, and P. Saint-Andre. *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. Internet Engineering Task Force (IETF), Mai 2015. https://tools.ietf.org/html/rfc7525. [Zuletzt besucht am 27.07.2016].

[TRG15] Inc. The Radicati Group. Email Statistics Report, 2015-2019. http://www.radicati.com/wp/wp-content/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf, 2015. [Zuletzt besucht am 27.07.2016].

[TW12] Andrew S. Tanenbaum and David J. Wetherall. *Computernetzwerke*. Pearson Deutschland GmbH, 2012.

[Wal16] Neal H. Walfield. Practical Encrypted Mailing Lists. Technical report, Johns Hopkins University, 2016. http://hssl.cs.jhu.edu/~neal/encrypted-mailing-lists.pdf. [Zuletzt besucht am 27.07.2016].

[Web08] Webdav101. System.Net.Mail with SSL to authenticate against port 465. https://blogs.msdn.microsoft.com/webdav_101/2008/06/02/system-net-mail-with-ssl-to-authenticate-against-port-465/, 2008. [Zuletzt besucht am 27.07.2016].

[Wel11] Harald Welte. Sichere und vertrauenswürdige elektronische Kommunikation via De-Mail. https://www.ccc.de/system/uploads/64/original/CCC-de-mail-2011.pdf, 2011. [Zuletzt besucht am 27.07.2016].

[WYY05a] Xiaoyun Wang, Andrew C Yao, and Frances Yao. Cryptanalysis on SHA-1. Technical report, Tsinghua University & Shandong Unversity; Tsinghua University; City University of Hong Kong, 2005. http://csrc.nist.gov/groups/ST/hash/documents/Wang_SHA1-New-Result.pdf. [Zuletzt besucht am 27.07.2016].

[WYY05b] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. Technical report, Shandong University, Jinan 250100, China; Independent Security Consultant, Greenwich CT, US; Shandong University, Jinan250100, China,

2005. https://www.iacr.org/archive/crypto2005/36210017/36210017.pdf. [Zuletzt besucht am 27.07.2016].

[Zim91] Phil Zimmermann. Why I Wrote PGP. https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html, 1991. [Zuletzt besucht am 27.07.2016].

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
AH	Authentication Header
ASCII	American Standard Code for Information Interchange
BfDI	Bundesbeauftragte für den Datenschutz und die Informationsfreiheit
BND	Bundesnachrichtendienst
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Certificate Authority
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CMS	Cryptographic Message Syntax
DES	Data Encryption Standard
DMDA	De-Mail-Dienstanbieter
DNS	Domain Name System
DoS	Denial of Service
DSA	Digital Signature Algorithm
ESMTP	Extended SMTP
ESP	Encapsulating Security Payload
GNU GPL	GNU General Public Licensex
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IBM	International Business Machines Corporation
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IKE	Inter Key Exchange
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPsec	IP Security Protocol
IPv4	Internet Protocol version 4

IPv6	Internet Protocol version 6
ISO	International Standardization Organization
ISP	Internet Service Provider
KLF	Key Legitimation Field
MAC	Message Authentication Code
MD5	Message Digest No. 5
MIME	Multipurpose Internet Mail Extension
MPLS	Multi-Protocol Label Switching
MDA	Message Delivery Agent
MSA	Message Submission Agent
MTA	Message Transfer Agent
MUA	Message User Agent
NAT	Network Address Translation
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OSI	Open Systems Interconnection
PFS	Perfect Forward Secrecy
PGP	Pretty Good Privacy
PHP	Hypertext Preprocessor
PKCS	Public Key Cryptography Standard
POP3	Post Office Protocol 3
PRF	Pseudo Random Function
RC4	Rivest Cipher 4
RFC	Request For Comments
RSA	Rivest-Shamir-Adleman
SA	Security Association
SAD	Security Association Database
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SMTP	Simple Mail Transfer Protocol
SPD	Security Policy Database
SPI	Security Parameter Index
SSL	Secure Socket Layer-Protocol
S/MIME	Secure/Multipurpose Internet Mail Extension
TCP	Transmission Control Protocol

TTL	Time To Live
TLS	Transport Layer Security
UDP	User Datagram Protocol
UTF-8	Unicode Transformation Format 8
VC	Virtual Circuit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29. Juli 2016 Fabian Reiber