**HOCHSCHULE für Angewandte Wissenschaften LANDSHUT**

**UNIVERSITY of Applied Sciences**

**FAKULTÄT Informatik**

**HOCHSCHULE LANDSHUT**

FAKULTÄT INFORMATIK

**Echtzeitbasierte Netzwerkdatenanalyse zur Ermittlung verwendeter Root Zertifikate**

**Masterarbeit**

vorgelegt von

**Sebastian Bilda**

aus Landshut

eingereicht am: .......................

Betreuer: Johann Uhrmann

# Erklärung zur Masterarbeit

Name, Vorname   
der/des Studierenden: …………………………………………….

Hiermit erkläre ich, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Landshut, den ……………. …..………………………………….  
 (Unterschrift der/des Studierenden)

Inhaltsverzeichnis

[Erklärung zur Masterarbeit II](#_Toc68510618)

[Inhaltsverzeichnis III](#_Toc68510619)

[Abbildungsverzeichnis IV](#_Toc68510620)

[Tabellenverzeichnis V](#_Toc68510621)

[Listingverzeichnis VI](#_Toc68510622)

[Abstract 1](#_Toc68510623)

[1 Einleitung 2](#_Toc68510624)

[1.1 Problemstellung und Motivation 2](#_Toc68510625)

[1.2 Zielsetzung 3](#_Toc68510626)

[1.3 Vorgehensweise 3](#_Toc68510627)

[2 Grundlagen 4](#_Toc68510628)

[2.1 Public Key Infrastructure 4](#_Toc68510629)

[2.2 Verschlüsselungsverfahren 4](#_Toc68510630)

[2.2.1 Symmetrisches Verfahren 4](#_Toc68510631)

[2.2.2 Asymmetrisches Verfahren 5](#_Toc68510632)

[2.3 Zertifizierungsstellen 6](#_Toc68510633)

[2.4 Digitale Zertifikate 6](#_Toc68510634)

[2.4.1 X.509 Zertifikate 6](#_Toc68510635)

[2.4.2 Zertifikatsketten 7](#_Toc68510636)

[2.4.3 Format und Semantik von Zertifikaten 8](#_Toc68510637)

[2.4.4 Zertifikatsperrliste 10](#_Toc68510638)

[2.5 Transport Layer Security 10](#_Toc68510639)

[2.5.1 TLS Handshake 11](#_Toc68510640)

[2.6 Windows Certificate Trust Store 13](#_Toc68510641)

[2.7 Stand der Forschung 13](#_Toc68510642)

[3 Implementierung des PcapAnalyzer 15](#_Toc68510643)

[3.1 Anforderungsanalyse 16](#_Toc68510644)

[3.2 Aufbau und Struktur 17](#_Toc68510645)

[4 Methodik 24](#_Toc68510646)

[4.1 Wahl der Methode 24](#_Toc68510647)

[4.2 Versuchsaufbau 24](#_Toc68510648)

[4.3 Vorgehensweise 26](#_Toc68510649)

[5 Ergebnisse 27](#_Toc68510650)

[6 Zusammenfassung und Ausblick 33](#_Toc68510651)

[Literaturverzeichnis 34](#_Toc68510652)

**Abbildungsverzeichnis**

[Abbildung 1: Symmetrische Verschlüsselung [21] 5](#_Toc68482481)

[Abbildung 2: Asymmetrisches Verfahren [21] 5](#_Toc68482482)

[Abbildung 3: CA-Hierarchien [13] 7](#_Toc68482483)

[Abbildung 4: Beispiel einer Zertifikatskette 8](#_Toc68482484)

[Abbildung 5: Zertifikat Datenstruktur [33] 9](#_Toc68482485)

[Abbildung 6: TLS-Protokollstruktur [35] 11](#_Toc68482486)

[Abbildung 7: TLS Handshake Ablauf [9] 12](#_Toc68482487)

[Abbildung 9: PcapAnalyzer Klasse 17](#_Toc68482488)

[Abbildung 10: Beispielausgabe list\_possible\_interfaces 18](#_Toc68482489)

[Abbildung 11: Zertifikatsklasse CertNode 19](#_Toc68482490)

[Abbildung 12: RootCATree Klasse 19](#_Toc68482491)

[Abbildung 13: Parser Klasse 21](#_Toc68482492)

[Abbildung 14: GetRootCAs Klasse 22](#_Toc68482493)

[Abbildung 15: Verwendung des PcapAnalyzers 23](#_Toc68482494)

[Abbildung 8: Aufbau des privaten Heimnetzwerkes 25](#_Toc68482495)

[Abbildung 16: Ausschnitt aus der Logdatei (subjectKeyIdentifier) 27](#_Toc68482496)

[Abbildung 17: Generelle Zahlen 28](#_Toc68482497)

[Abbildung 18: Verwendete Root Zertifizierungsstellen 28](#_Toc68482498)

[Abbildung 19: Baumstruktur der CA AAA Certificate Services 29](#_Toc68482499)

[Abbildung 20: Konvergenz der Zertifizierungsstellen 30](#_Toc68482500)

[Abbildung 21: Verwendete Cipher Suites 31](#_Toc68482501)

[Abbildung 22: Lokalität der Zertifizierungsstellen 32](#_Toc68482502)

**Tabellenverzeichnis**

[Tabelle 1: Funktionale Anforderungen 16](#_Toc68482503)

[Tabelle 2: Eckdaten der Stichproben 25](#_Toc68482504)

**Listingverzeichnis**

[Listing 1: run Methode des PcapAnalyzer 18](#_Toc68482505)

[Listing 2: Erstellung eines Zertifikatsbaum 20](#_Toc68482506)

[Listing 3: Methode zum Einfügen eines Knotens 20](#_Toc68482507)

[Listing 4: get\_roots Methode der GetRootCAs Klasse 22](#_Toc68482508)

Abstract

Authentische und verschlüsselte Verbindungen basieren meist auf TLS. Die Identität eines Servers wird mittels seines Zertifikats bewiesen, welches von einer Zertifizierungsstelle signiert sein muss. Dieser wiederum vertraut das System. Es gibt allerdings hunderte von Zertifizierungsstellen.

Das Ziel der vorliegenden Arbeit ist es, zu beantworten, ob das Vertrauen in Zertifizierungsstellen feingranularer konfigurierbar ist oder zu verifizieren ob dieses Vertrauten notwendig beziehungsweise gerechtfertigt ist. Dazu wird folgende Forschungsfrage gestellt: ‚Wie viele Zertifizierungsstellen werden in einem System wirklich verwendet beziehungsweise sind notwendig?‘.

Um diese Frage zu beantworten, wurde eine Software zur Analyse von Netzwerkdaten implementiert. Darüber hinaus wurden Stichproben des Netzwerkverkehrs in einem privaten Haushalt aufgezeichnet. Die aufgezeichneten Daten werden an die Analysesoftware übergeben, um daraus Zertifikatsketten zu extrahieren und als Baumstruktur für jede Zertifizierungsstelle zusammenzusetzen.

Die Statistiken aus dem Analysewerkzeug zeigen, dass nur ein Bruchteil aller Zertifizierungsstellen verwendet werden. Der gesamte Netzwerkverkehr kann auf genau 26 Zertifikate zurückgeführt werden, welche vertrauenswürdig für das System sein müssen, um ohne HTTPS Zertifikatswarnungen im Internet browsen zu können.

# Einleitung

## Problemstellung und Motivation

Kevin Bocek, Vice President Security Strategy & Threat Intelligence beschrieb 2017 in dem Journal Wirtschaftsinformatik & Management unter dem Namen Vertrauen im Internet in Gefahr, dass zu viele digitale Zertifikate das Internet versalzen. Durch Unternehmen wie Let`s Encrypt, ist das Erlangen von digitalen Zertifikaten für Cyber-Kriminelle genauso möglich, wie für ein renommiertes Unternehmen [2]. Forshaw, J. ist der Meinung, dass die CAs in der heutigen Zeit mehr darauf aus sind Zertifikate zu verkaufen, anstatt auf Korrektheit und Zweck konkret zu prüfen [10]. Die Anzahl an Zertifikaten, der ein System vertraut, sind in einer gemeinsamen Zertifizierungsstellen (Certificate Authority, CA) Datenbank, der Common Certificate Authority Database (CCADB) aufgelistet. Die CCADB wird unter anderem von Microsoft, Google und Mozilla verwendet. Derzeit befinden sich 415 Stamm- und Zwischenzertifikate darin [24]. Jedes Zertifikat, das von einem vertrauenswürdigen Stamm- oder Zwischenzertifikat signiert ist, ist auch für ein Windowssystem vertrauenswürdig. Durch die große Anzahl an Zertifikaten tritt immer wieder die Frage auf: Welchen Zertifikaten beziehungsweise Zertifizierungsstellen ist es notwendig zu vertrauen?

Es gibt derzeit keinen etablierten Mechanismus, um das Vertrauen in die Zertifizierungsstellen feingranularer zu konfigurieren oder überhaupt zu verifizieren, ob dieses Vertrauen notwendig bzw. gerechtfertigt ist. Daher kann aus diesem Zusammenhang, die Menge an vertrauenswürdigen Zertifizierungsstellen in Systemen oder Browsern infrage gestellt werden.

Es gibt verschiedene Ansätze das Vertrauen in die CAs zu steigern. Zum einen eine Initiative von Google, Certificate Transparency (CT), die durch eine zusätzliche öffentliche Instanz, auf der alle neu registrierten Zertifikate in einem kryptografischen Verfahren speichert. Dadurch soll ein erneutes Ausstellen eines Zertifikates für eine Domain verhindern werden [5]. Dieses Verfahren wird im Abschnitt 2.7 nochmal genauer erklärt. Zum anderen gibt es die Möglichkeit über den Certificate Authority Authorization (CAA) Record die Zertifikate genauer zu überprüfen. Hierbei entscheidet der Domaininhaber welche CA ein Zertifikat für diese Domain ausstellen darf [8]. Grundsätzlich trägt der CAA-Record zur Sicherheit bei, jedoch steigert dies nicht das Vertrauen in die Zertifizierungsstellen, da der CAA-Record von dem Domaininhabern gesetzt wird.

## Zielsetzung

Ziel ist es, am Ende dieser Masterarbeit mithilfe einer entwickelten Software eine Liste von Stamm- und Zwischenzertifikaten zur Verfügung zu stellen. Diese Liste soll für einen privaten Haushalt die notwendigen Zertifikate enthalten. Das entwickelte Analysewerkzeug soll aus einem Mitschnitt des Netzwerkverkehrs, die Zertifizierungsstellen extrahieren und einen Überblick verschaffen, welche Stammzertifizierungsstellen wirklich benötigt werden. Des Weiteren sollen anhand des Netzwerkverkehrs verschiedene Auswertungen erstellt werden. Die Statistiken sollen eine Basis dafür bilden, um die Anzahl der Zertifizierungsstellen in den Systemen einzuschränken bzw. zu verringern.

Anhand der Forschungsfrage, wie viele Stammzertifizierungsstellen in einem privaten Haushalt benötigt beziehungsweise verwendet werden, soll geklärt werden, ob das Vertrauen in die Menge an Zertifizierungsstellen gerechtfertigt ist.

## Vorgehensweise

Diese Arbeit ist in sechs Kapitel unterteilt. Im folgenden zweiten Kapitel werden Digitale Zertifikate im Detail spezifiziert, der Begriff Public Key Infrastructure (PKI) erläutert und die Aufgaben einer Zertifizierungsstelle beschrieben. Des Weiteren wird der Transport Layer Security (TLS) konkretisiert und die verschiedenen Verschlüsselungstechniken erörtert. Das dritte Kapitel beschäftigt sich mit der Implementierung der Analysesoftware, welche den Teil der Datenaufbereitung übernimmt und Statistiken erstellt. In Kapitel vier werden die Methodiken, die zur Datenerhebung und folglich zum Versuchsaufbau führen, vorgestellt. Die Vorgehensweise des Experiments wird dargestellt. Anschließend folgt die Darstellung der Ergebnisse in Abschnitt fünf. Im letzten Kapitel wird ein Fazit der Arbeit gezogen und ein Ausblick gegeben.

# Grundlagen

In diesem Kapitel sollen die Grundlagen für das Verständnis der Arbeit beschrieben werden und die Schlüsselbegriffe definiert werden. Wie in der Zielsetzung beschrieben geht es um Netzwerke, das SSL/TLS Protokoll, digitale Zertifikate, Verschlüsselung und Zertifizierungsstellen. Diese Themengebiete sollen nun grundlegend erläutert werden.

## Public Key Infrastructure

Eine Public Key Infrastructure (PKI) oder auf Deutsch übersetzt – *Infrastruktur für öffentliche Schlüssel* – ist "das gesamte Konstrukt rund um die Absicherung der Datenkommunikation und die Identitätskontrolle im Netzwerk mithilfe von Zertifikaten" [21]. Mithilfe einer PKI wird die Sicherheit der Kommunikation durch Authentifizierung, Datenverschlüsselung und digitale Signaturen erhöht. Grundsätzlich basieren die Dienste einer PKI auf der ordnungsgemäßen Verwendung von öffentlichen/privaten Schlüsselpaaren. Die öffentliche Komponente eines Schlüsselpaars wird in Form eines Public-Key-Zertifikats ausgestellt und kann in Verbindung mit dem entsprechenden Algorithmus zur Verifizierung einer digitalen Signatur, zur Verschlüsselung von Daten oder zu beidem verwendet werden [1].

## Verschlüsselungsverfahren

In einer PKI werden verschiedene Verschlüsselungsverfahren verwendet. In dem Abschnitt darüber steht der Begriff Schlüsselpaar. Der öffentliche und private Schlüssel wird zur Asymmetrischen Verschlüsselung verwendet. Im späteren Verlauf der Arbeit wird auch die symmetrische Verschlüsselung erwähnt. Es soll kurz erläutert werden wie sich die beiden Verfahren unterscheiden.

### Symmetrisches Verfahren

Bei der symmetrischen Verschlüsselung existiert nur ein geheimer Schlüssel zum Verschlüsseln und Entschlüsseln zwischen zwei Kommunikationspartnern. Der größte Vorteil dieser Methode Daten zu verschlüsseln ist, dass es sehr viel schneller ist als das asymmetrische Verschlüsselungsverfahren. Das Problem bei der symmetrischen Verfahrens ist der Austausch dieses geheimen Schlüssels [21]. Die Abbildung 1 zeigt den Ablauf des Verfahrens.

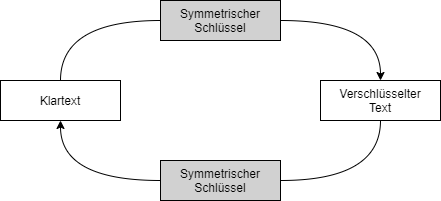


Abbildung 1: Symmetrische Verschlüsselung [21]

Dieses Verfahren wird dann eingesetzt, wenn bereits ein gemeinsamer geheimer Schlüssel vorliegt. Da kommt das asymmetrische Verschlüsselungsverfahren in das Spiel.

### Asymmetrisches Verfahren

Das asymmetrische Verschlüsselungsverfahren verwendet ein Schlüsselpaar zum Ver- und Entschlüsseln. Das Paar besteht aus einem öffentlichen und privaten Schlüssel, die durch einen mathematischen Algorithmus miteinander verbunden sind. Die Daten werden mit dem öffentlichen Schlüssel verschlüsselt und können nur von dem Besitzer des dazugehörigen privaten Schlüssels wieder entschlüsselt werden [21], wie in Abbildung 2 zu sehen.

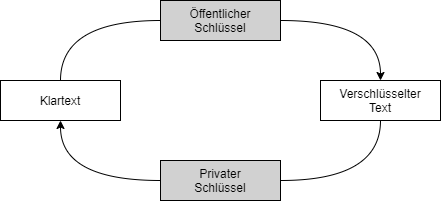


Abbildung 2: Asymmetrisches Verfahren [21]

In einer PKI wird genau dieses Verfahren eingesetzt, um den Schlüsselaustausch durchzuführen und digitale Signaturen zu erstellen.

## Zertifizierungsstellen

Zertifizierungsstellen stellen digitale Zertifikate aus, mit denen sichere und authentische Client – Server Verbindungen hergestellt werden. Sie sind die Basis der Vertrauenshierarchie im Internet. Jährlich werden Millionen von digitalen Zertifikaten ausgestellt und dazu verwendet, sicher über das Internet zu kommunizieren und die zu übermittelten Daten zu verschlüsseln [14]. Digitale Zertifikate werden im Abschnitt 2.4 näher erläutert. Eine Zertifizierungsstelle hat die Aufgabe Zertifikatsanforderungen anzunehmen und zu prüfen. Der Antragssteller wird überprüft, ob dieser berechtigt ist, ein Zertifikat für die von ihm beantragte Domain zu erhalten. Des Weiteren werden die Inhalte des digitalen Zertifikates auf unversehrt überprüft. Abhängig davon stellt die CA das Zertifikat aus oder lehnt es ab. Bei valider Klassifizierung des Zertifikates wird dieses mit CA-spezifischen Eigenschaften ergänzt und mit deren privaten Schlüssel signiert, um es vor Manipulation zu schützen. Des Weiteren ist eine CA dafür verantwortlich, Zertifikatssperrlisten zu erstellen. Diese Sperrlisten geben an welche Zertifikate nicht mehr gültig oder zurückgezogen wurden. In den Sperrlisten können nur Zertifikate enthalten sein, welche von dieser CA signiert wurden. Sie werden ebenfalls digital signiert damit eine Manipulation ausgeschlossen werden kann [21].

## Digitale Zertifikate

Peter Kleop definiert den Begriff digitales Zertifikat folgendermaßen: „Ein digitales Zertifikat bindet einen öffentlichen Schlüssel an eine Entität (Benutzer, Organisation, Computer) und beinhaltet zusätzliche Informationen, wie Sperrlisteninformationen und vieles mehr." [21]

Digitale Zertifikate sind ein Teil des Fundaments der IT-Sicherheit. Durch sie werden Public-Key-Infrastrukturen realisiert [3]. Im Folgenden soll, der am häufigsten verwendete Standard für Zertifikate beschrieben und technisch erläutert werden.

### X.509 Zertifikate

Der Name X.509 ist eigentlich der Standard. Sie werden oft als synonyme verwendet. In der aktuellen Version X.509v3 wurde er von der International Telecommunication Union (ITU-T) [20] und der International Organization for Standardization (ISO) [19] entwickelt. Der Aufbau eines X.509 Zertifikats ist in eine Baumstruktur gegliedert und wird durch die abstrakte Beschreibungssprache ASN.1[[1]](#footnote-2) beschrieben. Zur Kodierung der Zertifikate, Sperrlisten und OCSP[[2]](#footnote-3) (Online Certificate Status Protocol) wird die Distinguished Encoding Rule[[3]](#footnote-4) (DER) verwendet, welche einer Tag-Length-Value Kodierung entspricht [3].

### Zertifikatsketten

Eine Zertifikatskette beginnt immer bei einer vertrauenswürdigen Stamm Zertifizierungsstelle, welche weitere Zertifikate ausstellt. Grundsätzlich besteht aber eine CA aus einer Reihe von Wurzel-Zertifikaten. Nehmen wir zum Beispiel das Unternehmen GlobalSign, welches als eine öffentliche CA bezeichnet wird, aber eine Vielzahl an CAs betreibt. Somit bildet sich eine CA-Hierarchie, die eine Vertrauenskette schafft. In Abbildung 3: CA-Hierarchien wird eine solche Vertrauenskette abstrakt veranschaulicht.

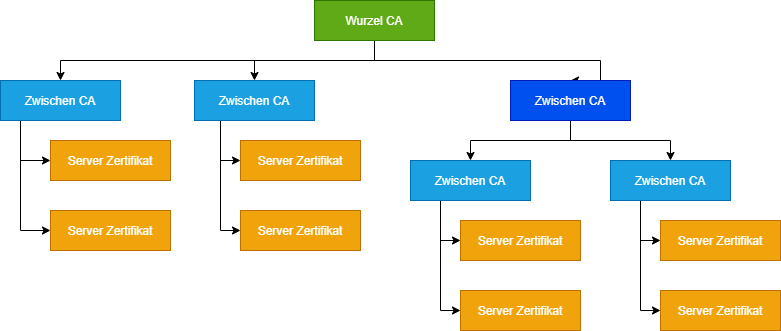


Abbildung 3: CA-Hierarchien [13]

**Wurzel CA**: Bildet die höchste Hierarchieebene und ist der Vertrauensanker. Nur wenn das Serverzertifikat mit einem Wurzelzertifikat verkettet ist und dieses im Betriebssystem oder Browser eingebettet ist, ist es vertrauenswürdig [13].

**Zwischen (Intermediate) CAs**: Sie agieren als Vermittler zwischen Wurzel- und Serverzertifikat. In jeder Zertifikatskette gibt es immer mindestens ein Zwischenzertifikat. Es gibt zwei Gründe warum Zwischenzertifikate eingeführt wurden. Zum einen können Zwischenzertifikate für einen bestimmten Zweck ausgestellt werden. Zum Beispiel nur zum Signieren von TLS oder Email Zertifikaten. Zum anderen um der Gefahr der Kompromittierung einer CA vorzubeugen. Die Root CA Zertifikate werden speziell geschützt. Im Falle einer Kompromittierung eines Intermediate Zertifikates muss nur dieses und die von ihr signierten Serverzertifikate widerrufen werden [1].

**Serverzertifikat**: Das Serverzertifikate ist dasjenige, das für die spezifische Domäne ausgestellt wurde. Es ist nicht mehr im Stande weitere Zertifikate zu signieren [1].

Da jedes Zertifikat von der übergeordneten Stelle signiert ist, sind sie miteinander verbunden und bilden eine Zertifikatskette.

Spezifischer betrachtet am Beispiel der Domain [*www.bsi.bund.de*](http://www.bsi.bund.de), welche das Endzertifikat bildet, stehen übergeordnet zwei weitere Zertifikate von D‑TRUST SSL und D‑TRUST Root (Abbildung 4).

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 4: Beispiel einer Zertifikatskette

Das Wurzel Zertifikate D-TRUST Root Class 3 CA 2 EV 2009. Das Zwischenzertifikat D‑TRUST SSL Class 3 CA 1 EV 2009, welches aufgrund des Namens vermuten lässt, dass hauptsächlich SSL/TLS Serverzertifikate damit signiert werden. Zuletzt das Serverzertifikat der Domain www.bsi.bund.de.

### Format und Semantik von Zertifikaten

Die oberste Strukturebene eines Zertifikates besitz genau drei Felder, das Feld *tbsCertificate*, *signatureAlgorithm* und *signatureValue*.

Das *tbsCertifiate* Feld enthält die Namen des Antragstellers (*subject*) und des Ausstellers (*issuer*) sowie einen öffentlichen Schlüssel, eine Zeitperiode der Gültigkeit und noch weitere Informationen. Das *signatureAlgorithm* Feld beinhaltet den kryptografischen Algorithmus, der von der jeweiligen Zertifizierungsstelle zum Signieren des Zertifikats verwendet wurde. Der *signatureValue* enthält die konkrete digitale Signatur der Zertifizierungsstelle. Mit dieser werden die Informationen des *tbsCertificate* Feldes valide. Die CA bestätigt die Verbindung zwischen öffentlichem Schlüsselmaterial und dem Besitzer des Zertifikats [33]. Im Folgenden wird die Datenstruktur eines Zertifikates in ASN.1 Notation in Abbildung 5 dargestellt.

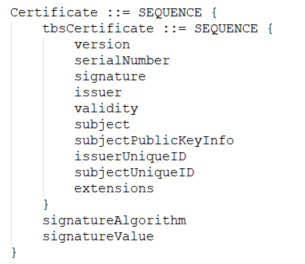


Abbildung 5: Zertifikat Datenstruktur [33]

Je nachdem von welcher Hierarchieebene das Zertifikat stammt, gibt es unterschiedliche Einschränkungen. Die wichtigsten davon, beziehungsweise diejenigen Felder die ausschlaggebend für die Implementierung und im weiteren Verlauf der Arbeit von Bedeutung sind, werden nun detailliert beschrieben.

Jedes Zertifikat wird von einem Zertifikat einer höheren Ebene signiert, welches der Issuer ist. Subject ist der Antragsteller des Zertifikates. Es dürfen keine zwei Zertifikate mit dem gleichen Subject existieren. Das heißt, dass jedes Zertifikat durch das Subject Feld eindeutig identifizierbar ist. Validity gibt den Gültigkeitszeitraum an, solange es in keiner Zertifikatsperrliste ist oder zurückgezogen wurde [3].

Ein weiteres wichtiges Feld ist, das *extension* Feld. Es beinhaltet zentrale Zertifikatserweiterungen, wie den Authority Key Identifier (AKID) und den Subject Key Identifier (SKI). Der AKID erleichtert es Anwendungen den Zertifizierungspfad zu rekonstruieren. Mit dieser Erweiterung kann das Ausstellerzertifikat, also der Issuer, hundertprozentig bestimmt werden, denn dieser ist identisch mit dem SKI des zu signierenden Zertifikats. Der SKI, welcher in jedem Zertifizierungsstellenzertifikat enthalten sein muss, ist eine Checksumme des öffentlichen Schlüssels des Zertifikats [3].

### Zertifikatsperrliste

Es werden jedes Jahr Millionen von Zertifikaten ausgestellt [15]. Was jedoch, wenn bei der Ausstellung des Zertifikats ein Fehler unterläuft oder der private Schlüssel öffentlich bekannt wird? Hierfür existiert die Zertifikatsperrliste, oder Certificate revocation list (CRL). Eine CRL ist laut dem National Institute of Standards and Technology (NIST) eine „Liste von widerrufenen Public Key Zertifikaten die von einer Zertifizierungsstelle erstellt und signiert wurde“ [7]. Das heißt im Grunde genommen ist es eine schwarze Liste von digitalen Zertifikaten, die nicht mehr vertrauenswürdig sind.

In den vorangegangenen Kapiteln wurden Digitale Zertifikate, PKI, CAs und Verschlüsselungsalgorithmen erläutert und wofür diese gebraucht werden. Nun soll auch das Transportprotokoll, über das eine PKI kommuniziert erläutert werden. Darüber werden auch die Zertifikatsketten übermittelt, die in der Analyse benötigt werden.

## Transport Layer Security

TLS (Transport Layer Security) ist der Nachfolger von SSL (Secure Sockets Layer), dabei handelt es sich um einen Internet Standard, welcher von der Internet Engineering Task Force (IETF) eingeführt wurde. In der aktuellen Version 1.3 hat TLS das primäre Ziel eine Verbindung zwischen zwei kommunizierenden Anwendungen zu gewährleisten. Die einzige Anforderung an die darunterliegende Transportschicht ist ein zuverlässiger, ordnungsgemäßer Datenstrom [34]. Die TLS-Protokollstruktur ist in Abbildung 6 zu sehen.

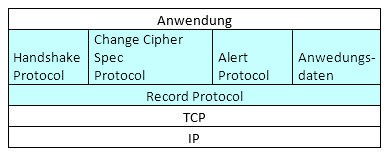


Abbildung 6: TLS-Protokollstruktur [35]

Das TLS Record Protocol, nimmt die zu übertragenden Nachrichten an, fragmentiert, komprimiert, verschlüsselt diese und verschickt das Ergebnis. Die empfangenen Daten werden verifiziert, entschlüsselt und wieder zusammengesetzt [34]. Aufbauend auf dem Record Protocol basieren vier weitere Protokolle:

1. **Handshake Protocol**, wird verwendet um die Sicherheitsparameter einer Verbindung auszuhandeln und gemeinsames Schlüsselmaterial festzulegen [34]. Auf den TLS Handshake speziell wird in Kapitel TLS Handshake 2.5.1 eingegangen.
2. **Change Cipher Spec Protocol**, Mitteilung des Wechsels auf die ausgehandelte Cipher Suite
3. **Alert Protocol**, ist für Fehler- und Alarmbehandlung zuständig.
4. **Application Data Protocol**, verantwortlich für das Zerlegen, komprimieren, verschlüsseln und übertragen der Datenblöcke.

TLS ist unabhängig vom Anwendungsprotokoll, d. h. die Protokolle höherer Ebenen können transparent darauf aufsetzten, zum Beispiel HTTP, das TLS verwendet, wird zu HTTPS. Der Standard beschreibt nicht, wie die höheren Protokolle die Sicherheit mit TLS hinzufügen oder der TLS-Handshake zu initiieren ist [34].

Um die Identität eines Servers zu verifizieren, sendet dieser zu Beginn einer Verbindung, in einem TLS Handshake die Zertifikatsnachricht. Dieser TLS Handshake soll nun genauer erklärt werden. Auf die anderen Protokolle wird in dieser Thesis nicht näher eingegangen, da sie für die Analyse der verwendeten Zertifizierungsstellen nicht relevant sind.

### TLS Handshake

Wie bereits erwähnt, wird durch den TLS Handshake eine sichere und authentische Verbindung zwischen Client und Server vereinbart. Der Handshake wird zu Beginn einer Verbindung durchgeführt. Er beinhaltet mehrere Schritte, die in Abbildung 7 aufgezeigt werden.

Beim Aufbau einer TLS Verbindung sendet der Client ein *client hello* als erste TLS Nachricht an den Server. Sie beinhaltet die Protokoll Version, eine Liste an unterstützten Cipher Suites, eine 32 Byte lange Zufallszahl, eine Liste an unterstützten Komprimierungsmethoden und eine Session ID. Der Server antwortet mit mehreren Nachrichten in genau der Reihenfolge, die in Abbildung 7 ersichtlich ist. Vorangehend mit dem *server hello*, worin die zu verwendende Cipher Suite und Kompressionsmethode festgelegt wurde. Auch enthalten ist die TLS Version, eine Zufallszahl und die Session ID. Als Nächstes kommt die Zertifikatsnachricht des Servers, *certificate*, welche die Zertifikatskette beinhaltet [34]. Die Zertifikatsketten und deren digitalen Zertifikate wurden in Kapitel 2.3 schon erläutert. Abschließend sendet der Server eine *hello done* Nachricht.

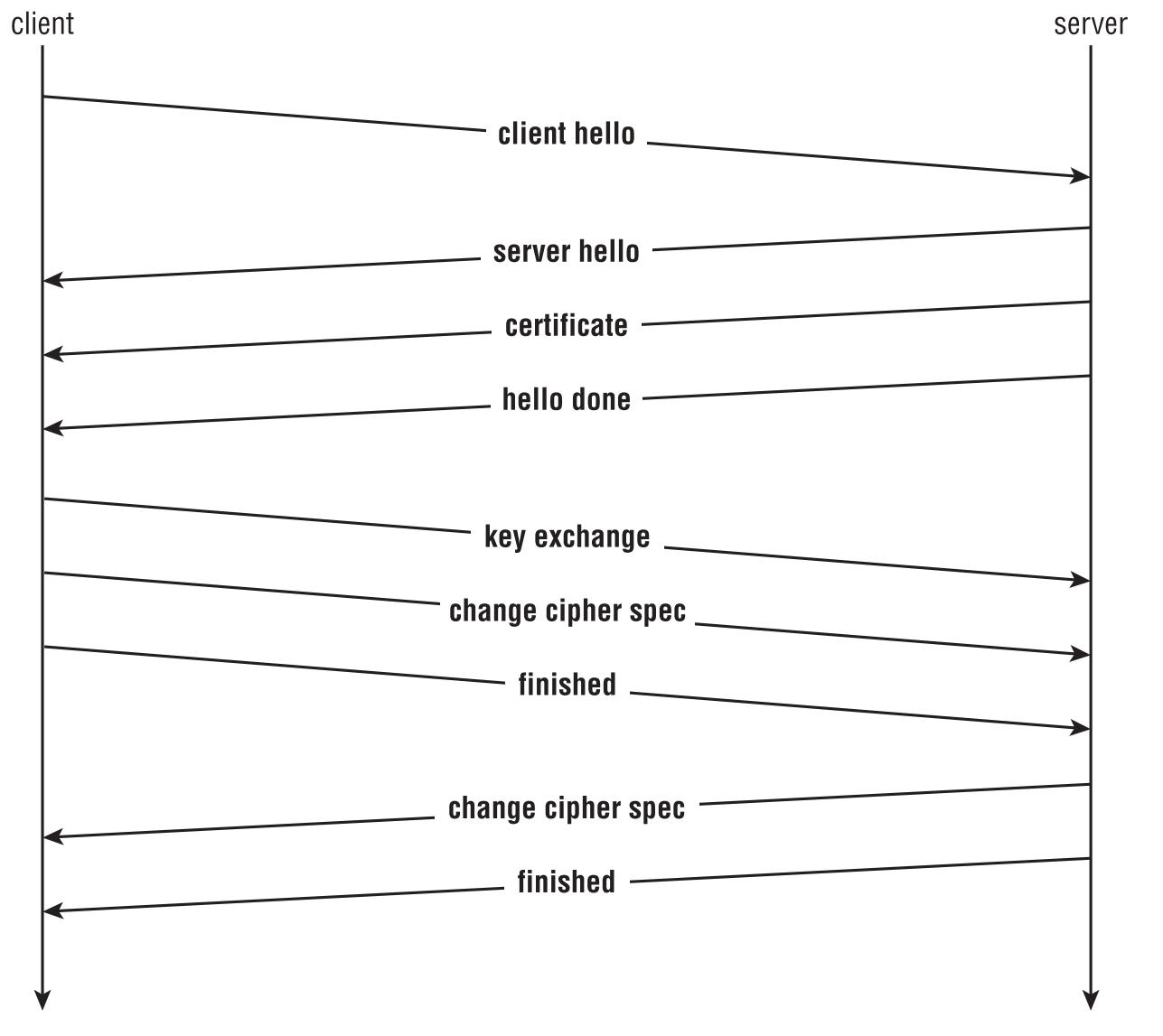


Abbildung 7: TLS Handshake Ablauf [9]

Der Client antwortet mit *key exchange* und *change cipher spec* Nachricht und zu guter Letzt die *finished* Nachricht. Der Server beendet den TLS Handshake ebenfalls mit den Nachrichten *change cipher spec* und *finished*.

Die *change cipher spec* Nachricht zeigt an, dass ab diesem Zeitpunkt, der TLS Record Layer die ausgehandelten kryptographischen Algorithmen und Schlüssel verwendet und somit gesichert kommuniziert wird. Die Finished-Nachricht bestätigt den erfolgreichen Schlüsselaustausch und der Authentifizierungsprozess. Der Empfänger der Finished-Nachricht muss diese Nachricht verifizieren, indem er den Hashwert aller ausgetauschten Nachrichten berechnet und überprüft [34].

Die wichtigsten Nachrichten in dieser Arbeit sind *server hello* und *certificate* des Servers. Sie enthalten die Zertifikatskette und die verwendeten Cipher Suites. Aus den Informationen dieser Nachrichten bestehen die Statistiken.

## Windows Certificate Trust Store

Der Windows Zertifikatsspeicher kann über die Zertifikatsverwaltung aufgerufen werden und dient zur Auflistung der vertrauenswürdigen und der nicht vertrauenswürdigen Zertifikaten. Wie aber Tom Aafloen in einem Blogbeitrag von Microsoft Security Solutions [25] beschreibt wird in der Zertifikatsverwaltung von Windows nur ein Teil der Zertifikate angezeigt, denen das System auch vertraut. Anhand eines Beispiels, bei der er eine Website aufruft die aktuelle Jobs in Hong Kong zeigt, verdeutlicht er, dass falls eine Website von einer CA signiert wurde, die nicht in der lokalen Zertifikatsverwaltung enthalten ist, wird eine Verbindung zu Windows Update hergestellt, um zu prüfen ob das Zertifikat dort aufgeführt ist. Ist dies der Fall wird es dem lokalen Speicher hinzugefügt. Aafloen betont am Ende seines Beitrages, er wisse nicht, ob das „Verstecken“ der vertrauenswürdigen CA’s den Benutzern nicht einen falschen Eindruck über die Sicherheit oder Kontrolle gibt.

Es sollte zumindest jedem absolut klar sein, dass das, was in der Oberfläche der Zertifikatsverwaltung angezeigt wird, nur eine Teilmenge aller Zertifikate ist.

Ist es dann möglich die Zertifikate über die Zertifikatsverwaltung feingranularer zu definieren, wenn der Abgleich mit einer Onlinedatenbank die Gültigkeit von Zertifikaten verifiziert?

## Stand der Forschung

Hierbei sollen die Möglichkeiten erläutert werden, die in Frage kommen können, um das Vertrauen in die Zertifizierungsstellen nicht völlig in deren Hände zu legen und die Sicherheit zu erhöhen.

Google’s Certificate Transperancy (CT) ist eine gute Alternative die Sicherheit basierend auf HTTPS und dem SSL/TLS Protokoll zu erhöhen. Der grundlegende Ansatz ist, dass jedes ausgestellte Zertifikat an einen öffentlich zugänglichen Logserver gesendet wird, der die Zertifikate in einer Merkle-Baum-Datenstruktur[[4]](#footnote-5) abspeichert und diese somit fest verwurzelt sind. Ist nun ein Angreifer in den Besitz eines gefälschten Zertifikates einer Domain gekommen, für das schon ein gültiges Zertifikat in den Logserver existiert, kann daraufhin über den Wurzelknoten bestimmt werden, ob dies das gleiche Zertifikat ist. Dennoch gibt es einige Aspekte zu beachten. CT funktioniert nur, wenn auch auf der Website das Flag für den expect CT Header gesetzt ist. Es ist nicht in allen Browsern implementiert (Mozilla beschreibt Performanceprobleme beim Einsatz von CT) [5].

Perl, Fahl et al. 2014 [27] analysierten auf einen internetweiten Datensatz von 48 Millionen HTTPS-Zertifikaten gestützt und verglichen diese mit Trust Stores aller großen Browser- und Betriebssystemhersteller. Dabei konnten sie 140 CA-Zertifikate identifizieren, die in zwölf Trust Stores aller wichtigen Plattformen enthalten sind und niemals zum Signieren von Zertifikaten verwendet werden. Basierend auf diesen Erkenntnissen wird vorgeschlagen diese CA-Zertifikate zu entfernen oder einzuschränken, um die HTTPS-Sicherheit zu verbessern. Sie validierten diese Erkenntnis mit einer zweimonatigen Sammlung von TLS-Handshake-Daten in einem Universitätsnetzwerk. Dabei entfernten Sie vorab die nicht relevanten CA-Zertifikate aus den Zertifikatsspeichern und stellten fest, dass keine einzige HTTPS-Warnmeldung hinzugekommen ist.

Das Entfernen oder Verbieten der bereits festgestellten Zertifizierungsstellen ist eine erhebliche Verbesserung der Sicherheit. Aufbauend auf diesen Erkenntnissen ist die Frage, ob nicht eine feingranularere Liste an ausgewählten CAs erstellt werden kann. In dieser Thesis geht es um den privaten Haushalt, bei denen sich in jedem Fall noch weitere Einschränkungen treffen lassen können. Des Weiteren basieren die Statistiken bei Perl, Fahl et al. 2014 auf einem statischen Datensatz, bei dem eine grundsätzliche Feststellung der Zertifikate im Vordergrund steht, welche für das Signieren von Webservern benutzt werden. Hier sollen jedoch die tatsächlichen verwendeten CA-Zertifikate aufgelistet werden, was vielmehr eine Erweiterung zu den Ergebnissen von Perl, Fahl et al. sind.

# Implementierung des PcapAnalyzer

Bei der Implementierung des PcapAnalyzer gab es zwei Möglichkeiten:

1. C/C++ mit Visual Studio
2. Python mit PyCharm

Nachfolgend sollen die beiden gegenübergestellt und zu einer Entscheidung gekommen werden. Die Möglichkeiten mit C/C++ sind sehr groß, es existieren bereits die grundlegenden Bibliotheken, die das meiste schon können. PcapPlusPlus [26], ein Wrapper der die pcaplib Funktionalitäten objektorientiert implementiert hat. Die Programmiersprache ist sehr hardwarenah. Für das Erstellen von Grafiken gibt es in C/C++ einen Wrapper MatplotPlusPlus [18], der die Matplotlib von Python in C/C++ übersetzt hat. Es wurde ein Prototyp CertificateAnalysis [12] entwickelt, der auf Github verfügbar ist, jedoch waren die Abhängigkeiten sehr hoch und schwierig einzubinden, da die MatplotPlusPlus Implementierung nur auf dem neuesten C Standard, C17, funktioniert, führte das zu Kompatibilitätsproblemen mit anderen Abhängigkeiten.

Die Interpretersprache Python ist eine einfachere und dynamisch typisierte Sprache. Es ist eine objektorientierte und plattformunabhängige Programmierung möglich. Mit den geeigneten Bibliotheken ist es möglich schnell und mit wenig Code ein Prototyp zu implementieren. Um einige verwendete Bibliotheken hier zu nennen:

* dpkt, für das Analysieren von Netzwerkverkehr
* crytography, eine sehr gute Implementierung für das Parsen von X.509 Zertifikaten
* matplotlib, für die Statistiken und Auswertungen der Daten

Es wurde ein weiterer Prototyp entwickelt und beide gegenübergestellt. Basierend auf den Implementierungen der Prototypen und der weitaus größeren Erfahrung in der Programmierung mit Python, wurde der PcapAnalyzer in Python weiterentwickelt.

Für die Implementierung wurde Python3 verwendet. Die genauen Abhängigkeiten können auf Github [29] nachgelesen werden. Der PcapAnalyzer ist angelehnt an PeterMosmans tls-protocol-analyzer [11].

## Anforderungsanalyse

Dieses Unterkapitel soll einen kurzen Überblick über die funktionalen und nicht funktionalen Anforderungen an die Software geben. Der PcapAnalyzer soll aus einer Pcap Datei die TLS Nachrichten extrahieren und aus diesen eine Übersicht geben welche CAs wirklich verwendet werden. Des Weiteren sollen die Cipher Suites, die Lokation und der kumulative zeitliche Verlauf wann jede einzelne Zertifizierungsstelle zum ersten Mal auftrat, aus den Daten dargestellt werden.

Zunächst sollen die funktionalen Anforderungen nach dem MoSCoW Prinzip [30] priorisiert in der nachfolgenden Tabelle 1 aufgelistet werden.

Tabelle 1: Funktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| Anforderung | PRIO | Beschreibung |
| [AFF-01] | MUST | Lesen einer Pcap Datei und in Echtzeit den Netzwerkverkehr mitschneiden |
| [AFF-02] | MUST | Wiederherstellen des TCP-Streams |
| [AFF-03] | MUST | Extrahieren der Zertifikatsnachricht und die Server‑Hello‑Done‑Nachricht |
| [AFF-04] | SHOULD | Extrahieren aller Zertifikatsnachrichten, um zu einem späteren Zeitpunkt alle Nachrichten zu analysieren. |
| [AFF-05] | MUST | Download der aktuellen CAs des Microsoft Trusted Root Programs und diese im aktuellen Arbeitsordner zu speichern |

Außerdem sollen die nichtfunktionalen Anforderungen wie Wartbarkeit und Zuverlässigkeit erfüllt werden. Das heißt es soll objektorientiert und modular programmiert werden, damit die Weiterentwicklung und weiteren Forschungen darauf aufbauen können.

## Aufbau und Struktur

Es wird nun auf die einzelnen Klassen eingegangen und diese kurz erläutert. Die Basis, oder Main Klasse bildet die PcapAnalyzer Klasse, welche in Abbildung 8 zu sehen ist. Sie startet das Lesen einer Pcap Datei oder auch das echtzeitbasierte mitschneiden des Netzwerkverkehrs. Des Weiteren hält sie eine Instanz der Parser Klasse und die Methoden zum Plotten der Statistiken.

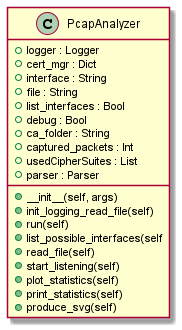
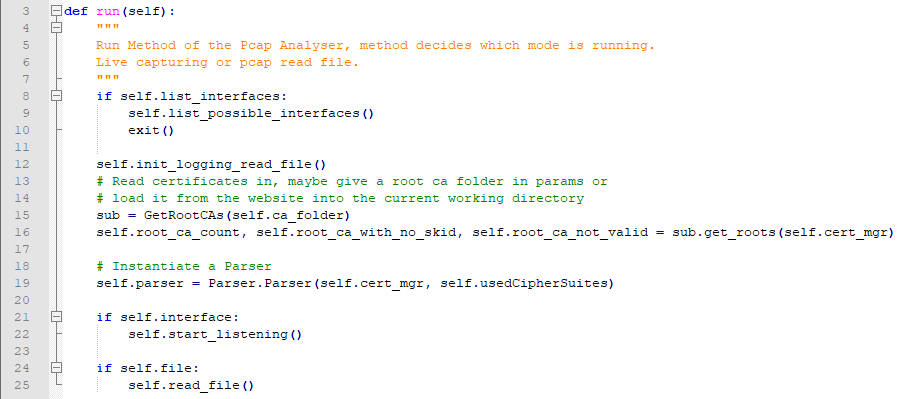


Abbildung 8: PcapAnalyzer Klasse

Die Erstellung der einzelnen Graphen über Graphviz, wird ebenfalls von dieser Klasse übernommen. Jeder Zertifikatsbaum wird in die von Graphviz unterstützte dot-Notation umgewandelt. Falls die Bibliothek installiert ist, werden automatisch die Dateien in Bilddateien umgewandelt und können dann angezeigt werden.

Wie in Abbildung 8 zu sehen verfügt der PcapAnalyzer über eine run Methode die in Listing 1 dargestellt wird.



Listing 1: run Methode des PcapAnalyzer

Sie startet den Echtzeit Netzwerkmitschnitt oder das Lesen der angegebenen Pcap Datei. Die Methode *list\_possible\_interfaces* listet für den Netzwerkmitschnitt die verfügbaren Schnittstellen auf, gibt diese aus und beendet das Programm wieder. Eine Beispielausgabe ist in Abbildung 9 zu sehen. Eine Parser Instanz wird erstellt, die jedes einzelne Netzwerkpaket erhält und analysiert.

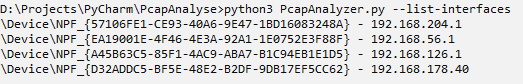


Abbildung 9: Beispielausgabe list\_possible\_interfaces

Angefangen bei einem einzelnen Knoten im Baum der Zertifikatsketten steht die CertNode Klasse, welche von der der Klasse Node der treelib Bibliothek erbt.

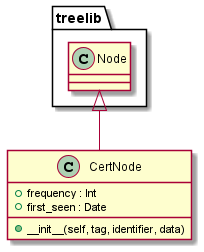


Abbildung 10: Zertifikatsklasse CertNode

Die Attribute *frequency* und *first\_seen* werden im späteren Verlauf für die Statistiken verwendet.

Die Klasse RootCATree sind die Baumstrukturen, die für jede Zertifizierungsstelle erstellt werden. Sie ist in Abbildung 11 zu sehen.

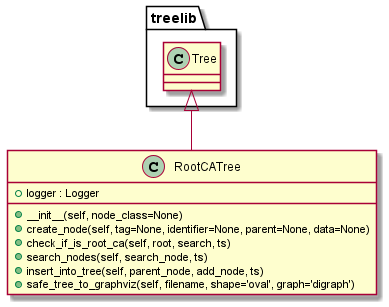
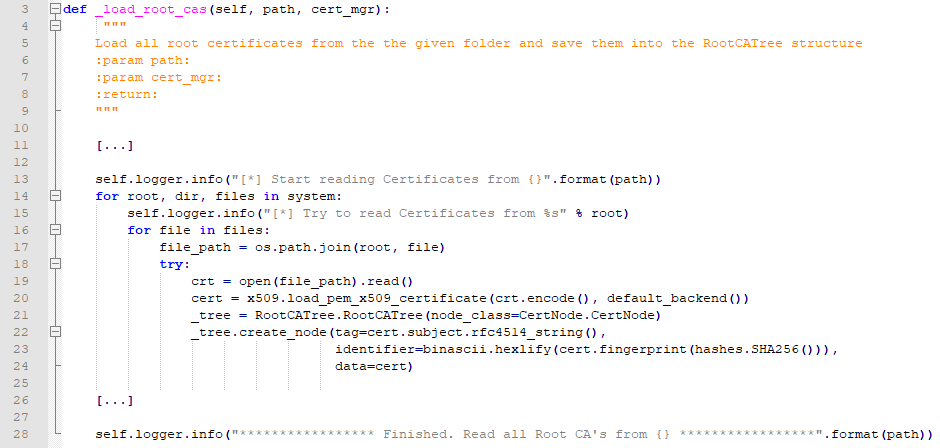


Abbildung 11: RootCATree Klasse

Der *tag* wird dazu verwendet das *subject* Feld eines Zertifikats zu repräsentieren, *data* beinhaltet das gesamte Zertifikat selbst und der *identifier* stellt den Fingerabdruck dar.

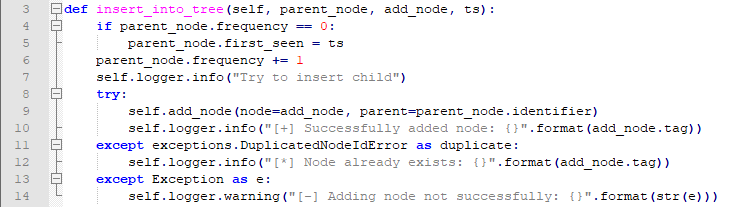
Um die Klasse CertNode als Knoten in der Klasse RootCATree zu verwenden muss dies beim Konstruktor des Baumes angegeben werden. In Listing 2 ist ein Codeausschnitt von der Klasse GetRootCAs zu sehen bei dem ein Zertifikat von der Festplatte geladen und als Wurzelknoten in die Baumstruktur des RootCATree eingebettet wird.



Listing 2: Erstellung eines Zertifikatsbaum

Mit dieser Klasse ist es möglich, dass die Zertifikatskette als Graph dargestellt werden kann und im Programm auch als Baum interpretiert wird. Die Zertifikatsketten werden somit in eine Baumstruktur umgewandelt.

Das Einfügen einer Zertifikatskette in einen vorhandenen Root CA Baum ist ebenfalls in der RootCAs Klasse implementiert, welches in Listing 3 gezeigt wird.



Listing 3: Methode zum Einfügen eines Knotens

Das Analysieren der Netzwerkpakete einer Pcap Datei oder dem Echtzeit Modus wird vollständig von der Parser Klasse übernommen. Dort wurde auch das Reassembeln des TCP-Streams implementiert, auf welches gleich noch näher eingegangen wird.

Im Netzwerk werden die Pakete auf TCP Ebene als Stream transportiert, das heißt dass es in TLS mehrere Möglichkeiten gibt, wie die Pakete an der Gegenstelle ankommen können. Für jede Verbindung, die zwischen Client und Server aufgebaut wird, wird im Programm eine Verbindungsvariable, die aus vier Teilen besteht gespeichert.

* Quell-IP-Adresse
* Quell-IP-Port
* Ziel-IP-Adresse
* Ziel-IP-Port

Diese Verbindungen werden in einer Variable *streambuffer* gespeichert. Damit kann jede Connection eindeutig identifiziert werden und die Streams wiederhergestellt werden. Die gesammelten TCP Pakete werden dann in eine von dpkt bereitgestellten Funktion *dpkt.ssl.tls\_multi\_factory()* übergeben, welche einen Puffer als Parameter bekommt und daraus die TLS Records extrahiert.

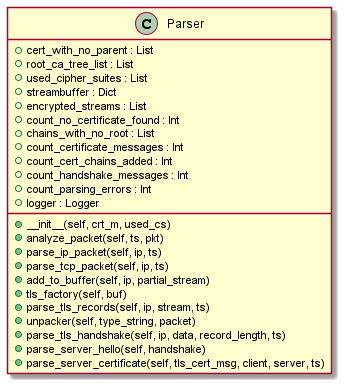


Abbildung 12: Parser Klasse

Zu Beginn des Programms werden die Zertifizierungsstellen von der Festplatte geladen, falls diese vorhanden sind. Falls kein Ordner angegeben wird, wird vom PcapAnalyzer ein Ordner im aktuellen Verzeichnis erstellt. Anschließend werden in diesen Ordner von der Microsoft Trusted Root Program Website, <https://ccadb-public.secure.force.com/microsoft/IncludedCACertificateReportForMSFT>, die aktuellen Zertifizierungsstellen heruntergeladen.

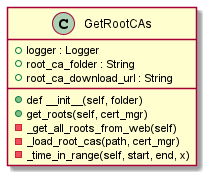
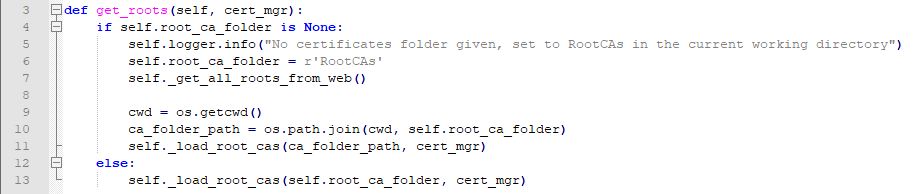


Abbildung 13: GetRootCAs Klasse

Die Methode *get\_roots* dieser Klasse entscheidet anhand der Kommandozeilenargumente ob die Zertifikate von Microsoft heruntergeladen werden sollen. In Listing 4 wird die Methode gezeigt.



Listing 4: get\_roots Methode der GetRootCAs Klasse

Das Programm wird über die Kommandozeile oder Powershell gestartet. Über die Argumentenparameter wird dann entweder die Pcap Datei oder die jeweilige Netzwerkschnittstelle übergeben. Einen Überblick über die Funktionen gibt Abbildung 14.

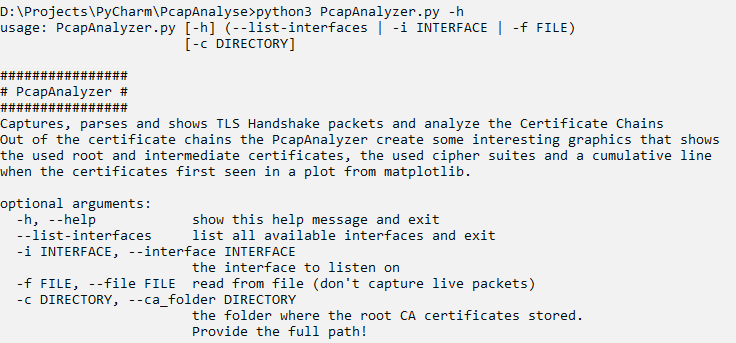


Abbildung 14: Verwendung des PcapAnalyzers

# Methodik

Nachdem die Grundlagen in Kapitel 2 und die Implementierung der Analysesoftware in Kapitel 3 erläutert wurden, folgt in diesem Kapitel die Erläuterung der Methodenwahl der im Rahmen der vorliegenden Arbeit durchzuführenden empirischen Untersuchung. Es gilt, Netzwerkdatenverkehr zu sammeln und schließlich die verwendeten Zertifizierungsstellen, mittels der vorher Implementierten Software, zu ermitteln. Hierzu wird im folgenden Kapitel die Wahl der Datenerhebungsmethode sowie die Durchführung vorgestellt.

## Wahl der Methode

Um Informationen über den Netzwerkverkehr von einem privaten Haushalt zu erhalten, bietet sich ausschließlich die Datengewinnung durch Primärerhebung an. Da es sich um schützenswerte Daten handelt existieren keine Sekundärdaten.

Es bestehen die Möglichkeiten der Datenerhebung über Befragung, Beobachtung oder Experiment [36]. Aus der in Kapitel 1.2 vorgestellten Forschungsfrage wird ersichtlich, dass sich die Methoden Befragung und Beobachtung nicht eignen. Ausschließlich das Experiment ist als Methode zu wählen, da die Netzwerkdaten nur im eigenen Heimnetzwerk legal mitgeschnitten werden können.

Hinsichtlich der Form des Experiments kann zwischen Stichproben und einem Mitschnitt über den kompletten Zeitraum gewählt werden. Ein Netzwerkmitschnitt über mehrere Monate übersteigt jedoch die Speicherkapazität. Aufgrund der Forschungsfrage und begrenzter Ressourcen wurde die Form der Stichprobe für dieses Experiment gewählt.

## Versuchsaufbau

Der Aufbau des Heimnetzwerkes wird in Abbildung 15 dargestellt. Aus Übersichtlichkeit wurde die Abbildung vereinfacht und nur die Geräte aufgezeigt, die auch relevant sind. Wie zu sehen ist, sind drei Windows Probanden für die Stichproben relevant. Da sich in einem Heimnetzwerk auch mobile Endgeräte und andere nicht Windows Geräte befinden, sich diese Thesis aber nur auf das Windows Betriebssystem beschränkt, wurde anhand der IP-Adressen der Windows Systeme gefiltert (siehe Abschnitt 4.3).

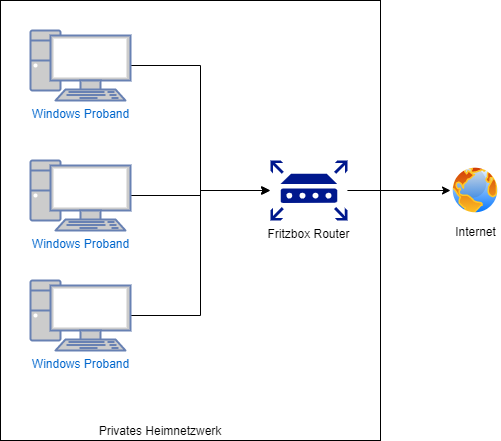


Abbildung 15: Aufbau des privaten Heimnetzwerkes

Die Probanden wurden dahingehend informiert, dass in dem Zeitraum mehrere Stichproben des Netzwerkverkehrs mitgeschnitten werden, jedoch nicht wann genau. Somit ist kein verfälschen der Daten möglich.

Es fanden drei unterschiedlich lange Stichproben statt. Eine über einen längeren Zeitraum, eine mittelfristige und eine kurzfristige Stichprobe. Die genauen Eckdaten können der nachfolgenden Tabelle entnommen werden.

Tabelle 2: Eckdaten der Stichproben

|  |  |  |  |
| --- | --- | --- | --- |
| Zeitraum | | Pakete | Dateigröße |
| 12.12.20 10:40 | 12.12.20 17:40 | 3,2 Millionen | 356 MB |
| 01.03.21 19:20 | 05.03.21 21:53 | 41 Millionen | 4325 MB |
| 15.03.21 19:17 | 22.03.2021 19:26 | 111,7 Millionen | 11 GB |

## Vorgehensweise

Um die Forschungsfrage zu beantworten, wurde ein Experiment durchgeführt, das stichprobenartig den Netzwerkverkehr eines privaten Haushaltes aufzeichnet und das entwickelte Programm die Netzwerkdaten analysiert. Da es sich um einen privaten Haushalt handelt, und dort als Betriebssystem überwiegend Microsoft Windows vertreten ist, konzentriert sich diese Thesis auf Windows.

Die verwendeten Root CAs eines Heimnetzwerks wurden mithilfe von Stichproben untersucht: Vom 01.12.20 bis 25.03.21 wurden mehrfach in unterschiedlichen zeitlichen Abständen und unterschiedlicher Länge in einem Heimnetzwerk der Netzwerkverkehr stichprobenartig mitgeschnitten.

Die Datenerhebung fand mittels eines Fritzbox Routers statt. Dieser ist in der Lage den Netzwerkverkehr von allen Geräten im Netzwerk mitzuschneiden. Auf der Support Webseite [17] des Herstellers AVM, befindet sich dafür eine detaillierte Anleitung.

Die gesammelten Stichproben wurden am Ende der Experimentlaufzeit zusammengeführt. Darauf wurde ein Filter angewendet, der die Windowssysteme anhand deren IP herausfiltert. Mit dem Programm tcpdump [31], welches ein Paket-Sniffer für die Kommandozeile ist, konnte über die IP Adressen der Windows Probanden der gewünschte Netzwerkverkehr herausgefiltert werden und daraufhin analysiert und ausgewertet.

Für die Datenanalyse wurde der Netzwerkmitschnitt in das entwickelte Programm PcapAnalyzer eingegeben. Dieser analysiert die Datei und extrahiert die TLS Nachrichten, welche unter anderem die Zertifikatsnachricht enthält. Daraufhin wurde die Zertifikatskette bis hin zur Zertifizierungsstelle rekonstruiert und in einer Liste abgespeichert. Anhand dieser Liste wurden Statistiken erstellt und ausgewertet.

Da es sich um sensitive Daten handelt und diese aus einem privaten Umfeld gesammelt wurden, wurden die Serverzertifikate, die auch die Domain beinhalten, bei der Analyse entfernt beziehungsweise nicht weiter betrachtet.

# Ergebnisse

Zu Beginn des Programms werden alle Zertifikate die in dem, dafür vorgesehenen Ordner enthalten sind, geladen. Bei dem Microsoft Trusted Root Program, wurden alle Zertifikate heruntergeladen, auch die, die auf der Webseite als Disabled gekennzeichnet sind. Die Programmlogik des Ladens der Zertifikate ist darauf ausgelegt, auf Gültigkeit und das subjectKeyIdentifier Feld eines Zertifikates zu prüfen, da dieser Pflicht in Root Zertifikaten ist. Es wurden einige Zertifikate, die aus dem Microsoft Trusted Root Program heruntergeladen wurden, erkannt, welche keinen subjectKeyIdentifier enthalten. Ein Ausschnitt aus der Logdatei (Abbildung 16) zeigt, dass das Starfield Services Root Certificate Authority – G2 Zertifikat erfolgreich geladen wird und das nächste einen Fehler enthält.

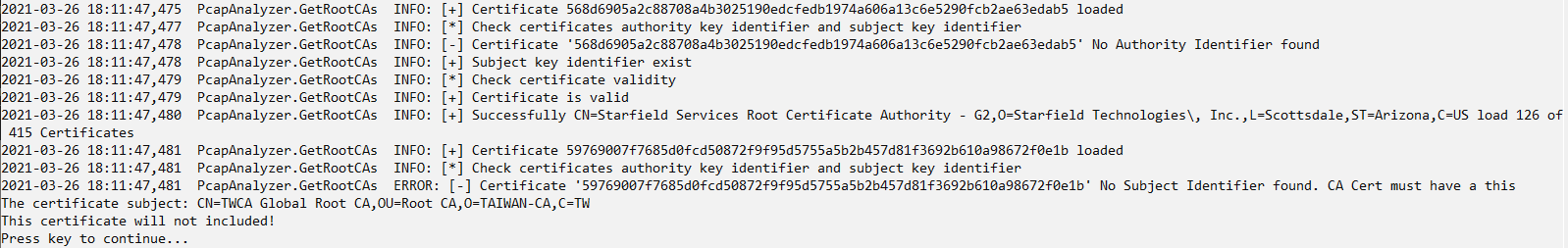


Abbildung 16: Ausschnitt aus der Logdatei (subjectKeyIdentifier)

Die Zertifikate, die keinen subjectKeyIdentifier enthalten, werden nicht in das Programm aufgenommen. In Abbildung 17, dem Ausschnitt über generelle Zahlen können die genauen Zahlen entnommen werden. Dies hat zur Folge, dass 50 Zertifikate nicht in das Programm geladen werden. Des Weiteren ist eine konkrete Rekonstruktion der Zertifikatskette ohne den subjectKeyIdentifier mit höherem Aufwand verbunden.

Um einen Überblick über die Daten zu bekommen, die analysiert wurden, wurde eine Sektion generelle Statistiken auf die Kommandozeile von Windows geschrieben, diese können aus Abbildung 17 entnommen werden. Die Zahlen geben Aufschluss darüber, wie lang die Analyze gedauert hat, wie viele Pakete analysiert wurden, wie viele TLS-Handshake Nachrichten darin waren und wie viele Zertifikatsnachrichten.

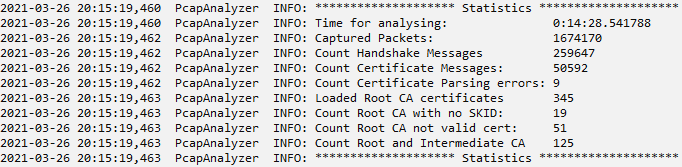


Abbildung 17: Generelle Zahlen

Bei neun Zertifikatsnachrichten konnte mindestens ein Zertifikat der Zertifikatskette nicht rekonstruiert werden, weil ein Fehler beim Laden des Zertifikats auftrat. Leider ist dies eine built-in Funktion der cryptography Bibliothek aus Python und die Fehlermeldung Unable to load certificate ist nicht sehr aufschlussreich. Nach einiger Recherchearbeit, wurde herausgefunden, dass der Byte-Stream der übertragen wird, fehlerhaft ist. Der Stream eines solchen fehlerhaften Zertifikates konnte auch bei unterschiedlichen Online Programmen, die auf das Parsen von Zertifikaten ausgelegt sind, nicht wiederhergestellt werden. Daher kann ein Programmfehler ausgeschlossen werden.

Von den 345 Zertifikaten, die zu Anfang geladen wurden, wird effektiv nur ein Bruchteil davon wirklich benötigt. Die in Abbildung 18 aufgelisteten Zertifizierungsstellen, stellen die benötigten Zertifikate dar.

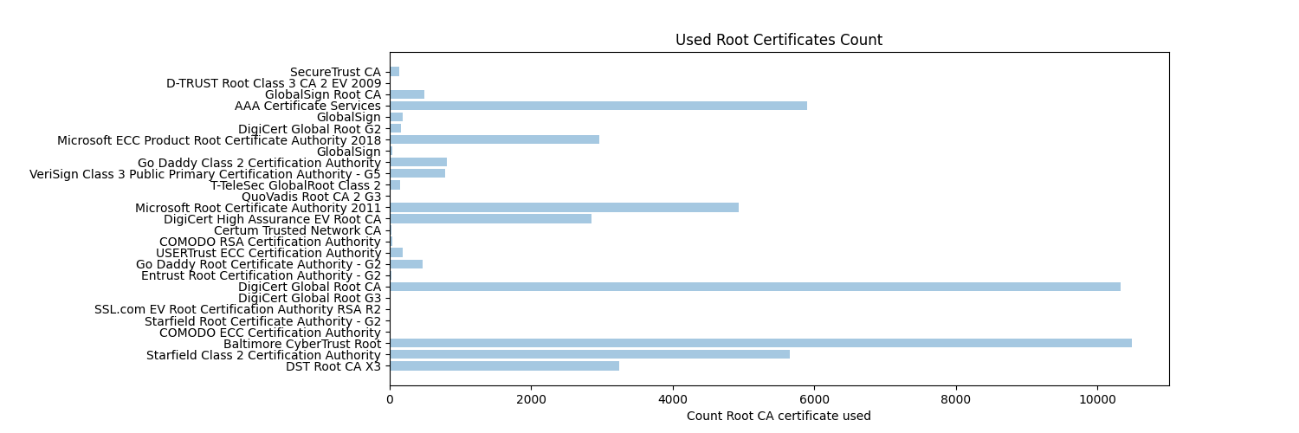


Abbildung 18: Verwendete Root Zertifizierungsstellen

Nimmt man die beiden Stichproben zusammen werden nur 26 der 345 Zertifizierungsstellen verwendet. Das entspricht einem prozentualen Anteil von 7,5 %. Das bedeutet, dass unser System viel zu vielen Zertifizierungsstellen vertraut. Ein weiterer Gesichtspunkt, der nicht außer Acht gelassen werden sollte, ist dass die aufgezeigten Zertifizierungsstellen nur die Wurzeln sind, denen wir vertrauen. Das heißt wenn eine CA beispielsweise 30 weitere Zwischenzertifizierungsstellen besitzt, wird diesen automatisch auch vertraut.

In Abbildung 17 gibt es einen Wert der genau dieses Verhalten kumuliert die tatsächliche Anzahl an Zertifikaten, denen der Endbenutzer vertraut. Sie beläuft sich auf 125. Da der PcapAnalyzer aus jedem Wurzelzertifikat eine Baumstruktur erstellt, kann jede Zertifizierungsstelle manuell überprüft wie viele Intermediate Zertifikate sie besitzt. Die CA Baltimore CyberTrust Root ist die am meisten verwendete CA. Allein diese CA Besitzt elf Zwischenzertifikaten und zusätzlich noch eine dritte Hierarchiestufe mit vier Intermediate Zertifikaten.

Der Baum in Abbildung 19 zeigt die Zertifizierungsstelle AAA Certificate Services und die signierten Zwischenzertifikate.

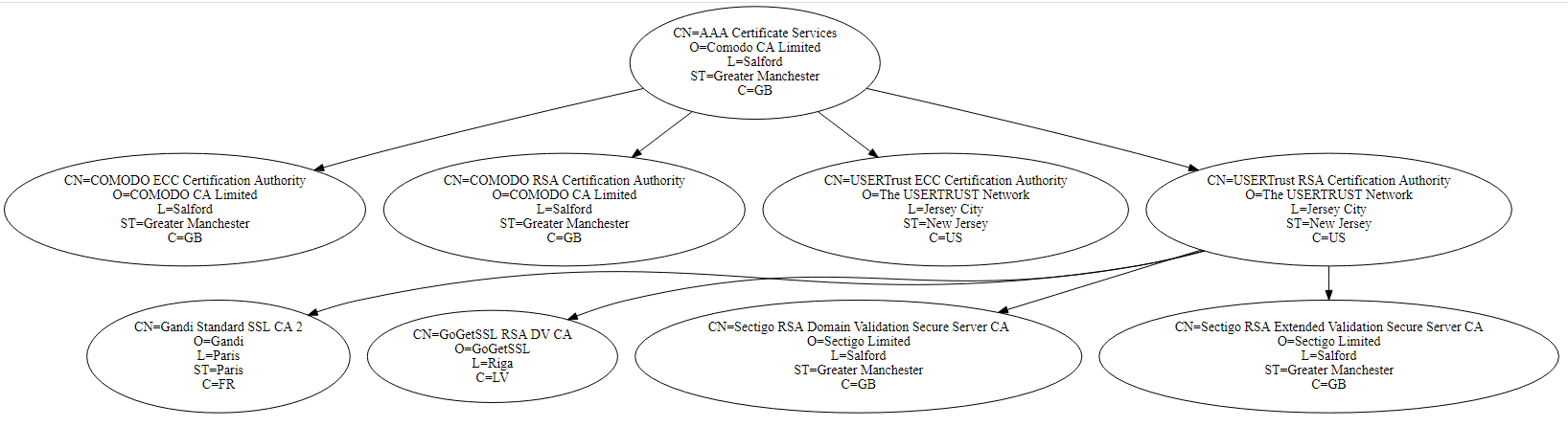


Abbildung 19: Baumstruktur der CA AAA Certificate Services

Was auch noch aus der Statistik hervorgeht, ist dass die Anzahl, wie oft eine Zertifizierungsstelle verwendet wurde, sehr hoch wird. Das liegt entweder daran, dass die Domänen, die diese CA signiert hat, sehr beliebt sind, oder dass die Anzahl der signierten Domänen sehr hoch ist. Acht Zertifizierungsstellen liegen bei mehr als 2000 Verwendungen.

Eine weitere interessante Grafik zeigt wann die Anzahl der Root CA’s stagniert, das heißt nach welcher Zeitspanne kommen keine neuen mehr hinzu. Im PcapAnalyzer wurde eine Statistik entworfen, bei der dies ersichtlich wird, jedoch nur in Bezug auf die Wurzeln, nicht inbegriffen sind die Zwischenzertifikate. Die Abbildung 20 zeigt einen Verlauf von vier Monaten über den die Stichproben erfolgt sind. Schon nach den ersten Tagen waren fast 20 Zertifikate identifiziert worden. Die Annahme ist, dass nach weiteren vier Monaten nicht einmal 5 Zertifikate neu hinzukommen würden.

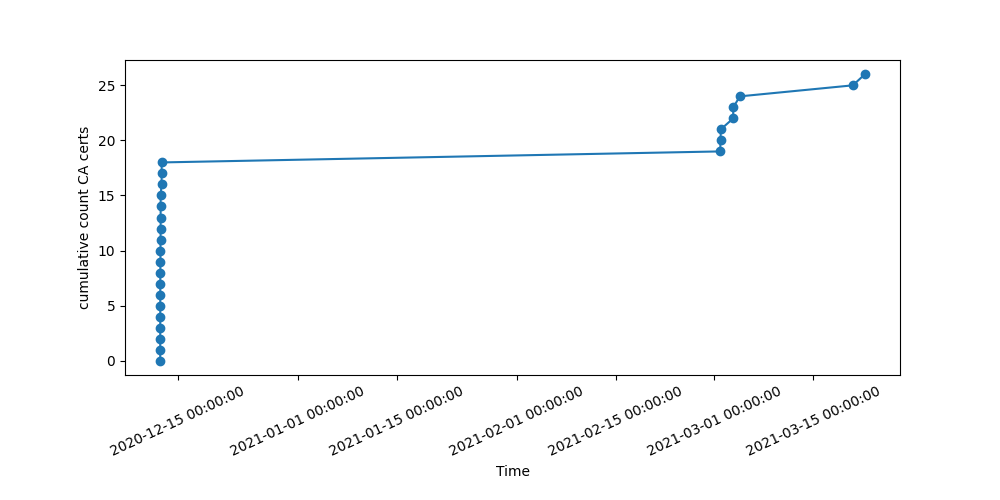


Abbildung 20: Konvergenz der Zertifizierungsstellen

Des Weiteren wurden die verwendeten Cipher Suites gesammelt. Einen Überblick darüber gibt Abbildung 21. Die Verwendung von sicheren Cipher Suites ist wichtig, da es um die verschlüsselte Kommunikation zwischen Client und Server geht. Die Hauptsächlich verwendeten Chiffresuiten, welche rund 75% einnehmen, sind TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384, TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256, TLS\_AES\_128\_GCM\_SHA256 und TLS\_AES\_256\_GCM\_SHA384. Sie sind unter den Chiffrensuiten, die auch empfohlen werden [28]. Die Namen der Cipher Suites sind immer nach dem gleichen Schema aufgebaut. Sie bestehen aus vier verschiedenen kryptographischen Algorithmen und vorangestellt das Protokoll. Jeder Algorithmus hat einen bestimmten Einsatzzweck. Betrachtet man die erst genannte Cipher Suite ist [16]

* TLS das Protokoll
* ECDHE (ephemeral Elliptic Curve Diffie Helman) der Schlüsselaustausch Algorithmus
* RSA (Rivest, Shamir und Adleman) wird für die Digitale Signature verwendet
* AES\_256\_GCM (Advanced Encryption Standard in Galois Counter Mode[[5]](#footnote-6) mit 256 Bit Schlüssel) ist für die die tatsächliche symmetrische Verschlüsselung zuständig die während einer HTTPS-Verbindung stattfindet
* SHA384 (Secure Hash Algorithm) Hash basierte Nachrichtenauthentifizierung und Datenintegrität

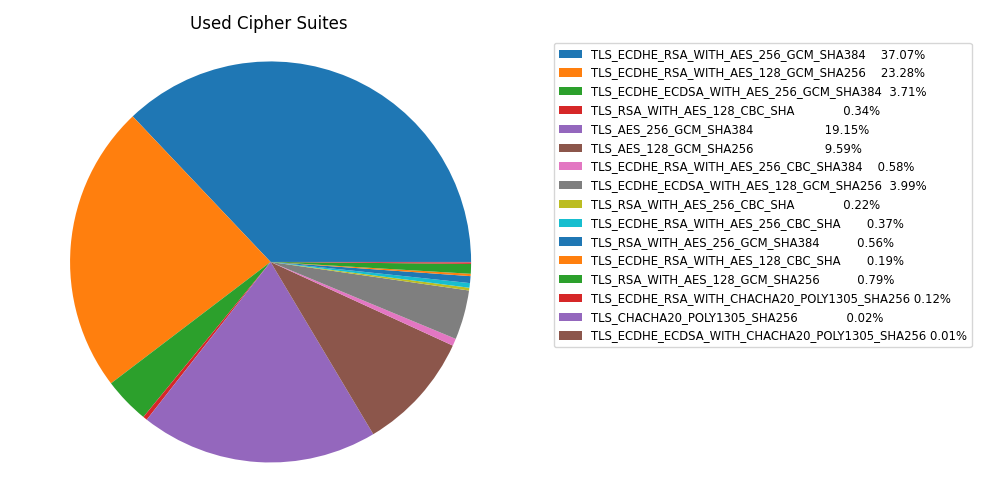


Abbildung 21: Verwendete Cipher Suites

Eine zusätzliche Funktion bietet eine Verschlüsselte TLS Sitzung, wenn der Schlüsselaustausch mittels Diffie-Hellman-Verfahren eingesetzt wurde. Bei diesem Verfahren ist es nicht notwendig, dass der Sitzungsschlüssel übertragen werden muss. Dann wird Perfect Forward Secrecy eingesetzt. Das bedeutet, falls der Hauptschlüssel bekannt wird eine nachträgliche Entschlüsselung verhindert wird, weil der Sitzungsschlüssel nicht rekonstruiert werden kann [23].

Zwei der verwendeten Chiffren sind schwache Cipher Suites. Sie nehmen zusammen jedoch nur 0,56 % der verwendet Chiffren Suiten ein.

* TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA
* TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA

Die beiden Chiffren verwenden für die Verschlüsselung AES 128 Bit mit Cipher Block Chaining[[6]](#footnote-7) mode, was anfällig für Klartextangriffe mit TLS 1.0, SSL 3.0 und niedriger ist. Die restlichen Cipher Suites sind alle sicher.

Am Rande zu erwähnen sind die Länder, in denen die Zertifizierungsstellen ihren Sitz haben. 65% aller verwendeten Zertifizierungsstellen sind aus den USA, gefolgt von Großbritannien mit 15%. Eine genauere Auflistung zeigt Abbildung 22.

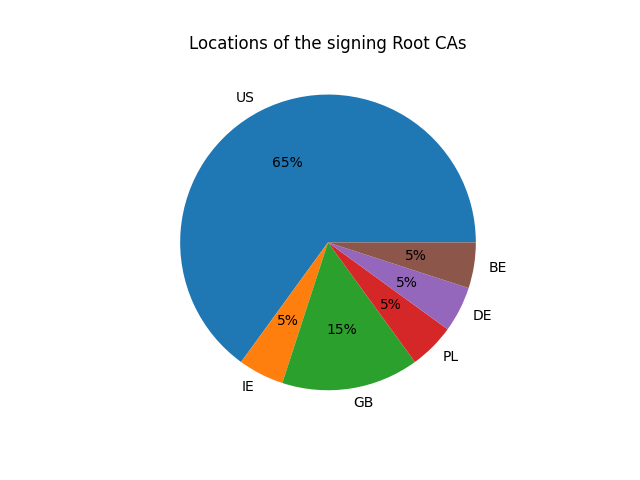


Abbildung 22: Lokalität der Zertifizierungsstellen

# Zusammenfassung und Ausblick

In der Masterarbeit wurde analysiert, welche Zertifizierungsstellen in einem privaten Haushalt wirklich benötigt werden. Die entwickelte Software beinhaltet eine echtzeitbasierte und statische Methode zur Erstellung notwendiger Statistiken, um die Frage, ob das Vertrauen in alle Root Zertifikate gerechtfertigt beziehungsweise notwendig ist, zu beantworten. Für die Beantwortung wurde eine quantitative Studie durchgeführt.

Aus den Ergebnissen geht klar hervor, dass nur ein Bruchteil, ca. 7,5 % aller Root Zertifikate auch wirklich benötigt beziehungsweise verwendet werden. Das Vertrauen in die Zertifizierungsstellen kann somit weitestgehend eingeschränkt werden. Dieser Ansatz hat zur Folge, dass die Wahrscheinlichkeit zur Verwendung eines kompromittieren Zertifikates um ein Vielfaches verringert wird. Zusammenfassend kann festgehalten werden, dass auf Basis der verwendeten Zertifizierungsstellen, die Sicherheit im Internet gesteigert werden kann.

Eine Weiterentwicklung des PcapAnalyzer mit einer graphischen Oberfläche könnte dazu führen, dass die Überprüfung von jedem Benutzer, der ein Windows System hat, selbst durchgeführt werden kann. Für Personen, die sich auf der Kommandozeile auskenne, ist dies auch jetzt schon möglich, da der PcapAnalyzer auf Github öffentlich ist. Auch ist denkbar, den Prozess der Konfiguration des Systems anhand der ermittelten Zertifikate zu automatisieren. Ein Interessanter Ansatz wäre, zu überprüfen, ob sich die verwendeten Zertifizierungsstelen auf mehrere Haushalte gleichbleibend ist oder sich verändert.

Literaturverzeichnis

[1] Anastasios Arampatzis. 2020. *The Difference between Root and Intermediate Certificates | Venafi*. https://​www.venafi.com​/​blog/​what-difference-between-root-certificates-and-intermediate-certificates. Accessed 30 December 2020.

[2] Bocek, K. 2017. Vertrauen im Internet in Gefahr. *Wirtsch Inform Manag* 9, 1, 64–69.

[3] Bundesamt für Sicherheit in der Informationstechnik. BSI Technische Richtlinie TR-02103: X.509 Zertifikate und Zertifizierungspfadvalidierung.

[4] Burr, W., Dodson, D., Nazario, N., and Polk, T. Minimum interoperability specification for PKI components (MISPC).

[5] 2020. *Certificate Transparency*. https://​www.certificate-transparency.org​/​. Accessed 7 July 2020.

[6] 2021. *Certificate Transparency - Web security | MDN*. https://​developer.mozilla.org​/​de/​docs/​Web/​Security/​Certificate\_Transparency. Accessed 15 March 2021.

[7] 2021. *CRL - Glossary | CSRC*. https://​csrc.nist.gov​/​glossary/​term/​CRL. Accessed 16 March 2021.

[8] 2020. *Das Setzen des CAA-Records trägt zur Sicherheit deiner Domain bei*. https://​www.internetx.com​/​news/​was-ist-ein-caa-record-und-warum-ist-er-so-wichtig/​. Accessed 7 July 2020.

[9] Davies, J. Implementing SSL/TLS Using Cryptography and PKI.

[10] Forshaw, J. 2018. *Netzwerkprotokolle hacken*. Dpunkt.

[11] GitHub. 2021. *PeterMosmans/tls-protocol-analyzer*. Accessed 16 March 2021.

[12] GitHub. 2021. *SebBil/CertificateAnalysis*. https://​github.com​/​SebBil/​CertificateAnalysis. Accessed 25 March 2021.

[13] GlobalSign GMO Internet, Inc. 2020. *Was sind untergeordnete CAs und wieso ist es sinnvoll, Ihre eigene zu haben?* https://​www.globalsign.com​/​de-de/​blog/​untergeordnete-cas-wieso-ist-es-sinnvoll-ihre-eigene-zu-haben. Accessed 13 March 2021.

[14] GlobalSign GMO Internet, Inc. 2020. *Zertifizierungsstellen & Vertrauenshierarchien | GlobalSign*. https://​www.globalsign.com​/​de-de/​ssl-information-center/​zertifizierungsstellen-vertrauenshierarchien. Accessed 9 March 2021.

[15] GlobalSign GMO Internet, Inc. 2020. *Zertifizierungsstellen & Vertrauenshierarchien | GlobalSign*. https://​www.globalsign.com​/​de-de/​ssl-information-center/​zertifizierungsstellen-vertrauenshierarchien. Accessed 22 March 2021.

[16] Hashed Out by The SSL Store™. 2019. *Cipher Suites: Ciphers, Algorithms and Negotiating Security Settings*. https://​www.thesslstore.com​/​blog/​cipher-suites-algorithms-security-settings/​. Accessed 2 April 2021.

[17] 2021. *Hilfe FRITZ!Box 7590 - FRITZ!Box-Support*. http://​service.avm.de​/​help/​de/​FRITZ-Box-7590/​015/​hilfe\_support. Accessed 30 March 2021.

[18] 2021. *Home - Matplot++*. https://​alandefreitas.github.io​/​matplotplusplus/​. Accessed 25 March 2021.

[19] ISO. 2021. *ISO - International Organization for Standardization*. https://​www.iso.org​/​home.html. Accessed 1 April 2021.

[20] ITU. 2021. *ITU Telecommunication Standardization Sector*. https://​www.itu.int​/​en/​ITU-T/​Pages/​default.aspx. Accessed 1 April 2021.

[21] Kloep, P. 2020. *PKI und CA in Windows-Netzwerken*. *Das umfassende Handbuch*. Rheinwerk Computing.

[22] Kuhlemann, O. 2021. *Moderne Kryptografie / Blockchiffre / Cipher Block Chaining Modus*. https://​kryptografie.de​/​kryptografie/​cbc.htm. Accessed 3 April 2021.

[23] Luber, S. 2017. Was ist Perfect Forward Secrecy (PFS)? *Security-Insider* (Dec. 2017).

[24] 2021. *MICROSOFT Included CA Certificate List*. *Common CA Database*. https://​ccadb-public.secure.force.com​/​microsoft/​IncludedCACertificateReportForMSFT. Accessed 29 March 2021.

[25] Microsoft Security Solutions. 2013. *Which Root CAs do you really trust?* https://​mssec.wordpress.com​/​2013/​01/​31/​which-root-cas-do-you-really-trust/​. Accessed 29 December 2020.

[26] 2020. *PcapPlusPlus - a multiplatform C++ library for capturing, parsing and crafting of network packets*. https://​pcapplusplus.github.io​/​. Accessed 7 July 2020.

[27] Perl, H., Fahl, S., and Smith, M. 2014. You Won’t Be Needing These Any More: On Removing Unused Certificates from Trust Stores. In *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, Berlin, Heidelberg, 307–315.

[28] Rudolph, H. C. 2021. *Cipher Suite Info*. https://​ciphersuite.info​/​. Accessed 24 March 2021.

[29] Sebastian Bilda. 2021. *PcapAnalyzer*. https://​github.com​/​SebBil/​PcapAnalyse. Accessed 14 March 2021.

[30] t2informatik. Wir entwickeln Software. 2019. Was ist die MoSCoW-Methode? - Wissen kompakt - t2informatik. *t2informatik GmbH* (Mar. 2019).

[31] 2021. *TCPDUMP (command-line sniffer/analyzer) for Windows*. https://​www.microolap.com​/​products/​network/​tcpdump/​. Accessed 30 March 2021.

[32] The Galois/Counter Mode of Operation (GCM).

[33] Tools.ietf.org, Rfcmarkup Version 1.129d On. 2021. *RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. https://​tools.ietf.org​/​html/​rfc5280. Accessed 13 March 2021.

[34] Tools.ietf.org, Rfcmarkup Version 1.129d On. 2021. *RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3*. https://​tools.ietf.org​/​html/​rfc8446. Accessed 13 March 2021.

[35] 2021. *Transport Layer Security (TLS) Funktionsweise & Erklärung*. https://​www.kryptowissen.de​/​transport-layer-security-tls.php. Accessed 22 March 2021.

[36] Walter, A. 2009. *Methodik der empirischen Forschung*. Gabler Verlag, Wiesbaden, s.l.

1. ASN.1 steht für Abstract Syntax Notation One und ist eine abstrakte Beschreibungssprache zur Definition von Datenstrukturen [↑](#footnote-ref-2)
2. Protokoll zum abfragen des Status eines digitalen Zertifikats [↑](#footnote-ref-3)
3. Regeln für die Kodierung von ASN.1 Objekten, die eine konsistente Kodierung für jeden ASN.1 Wert ergeben [4]. [↑](#footnote-ref-4)
4. In der Informatik ist der Merkle-Baum oder Hash-Baum ein Baum, bei dem die Knoten mit Hashes ihrer Kindknoten gekennzeichnet sind [6]. [↑](#footnote-ref-5)
5. GCM ist ein Blockchiffriermodus, der den Zählermodus auf den Klartext anwendet. Zusätzlich wird ein Message Authentication Code erstellt, um die Integrität sicherzustellen [32]. [↑](#footnote-ref-6)
6. CBC ist ein Blockchiffriermodus bei dem die Verschlüsselung eines Blockes abhängig von den schon vorher verschlüsselten Blöcken [22]. [↑](#footnote-ref-7)