

% Abschlussarbeit

% Autor\_in

% Prüfer\_in

% Betreuer\_in

---

Fakultät für  
Informatik und  
Mathematik



HOCHSCHULE  
FÜR ANGEWANDTE  
WISSENSCHAFTEN  
**MÜNCHEN**

# Authentifizierung von Systemen: Beschreibung eines Verfahrens zur sicheren Generierung, Verwahrung und Aktualisierung von Zertifikaten

Richard Reik

Bachelorarbeit Informatik

Prüfer:

Prof. Dr. , Hochschule München

Betreuer:

Dr. , Firma GmbH

01.03.2018

# Erklärung

Richard Reik, geb. 27.08.1998 (IF8, SS 2021)

Hiermit erkläre ich, dass ich die Bachelorarbeit 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.

München, 01.03.2018

.....

Unterschrift

# Zusammenfassung

(Worum geht es?) Sicherheit ist ein Thema das zunehmend an Wichtigkeit gewinnt, vor allem im Internet wird Kommunikation zunehmend verschlüsselt. Nachrichten die heutzutage unverschlüsselt verschickt werden sind nichtmehr die Regel sondern stellen eine Ausnahme dar. Daher ist es wichtig zu verstehen wie Sicherheit gewährleistet werden kann, auch wenn Geräte sich nichtmehr in eigener Hand befinden.

Zielstellung dieser Arbeit ist verwalten von Zertifikaten, welche benutzt werden können um jene Sicherheit zu gewährleisten. Dabei soll besondersDie sichere Erstellung, Aktualisierung und Speicherung von Zertifikaten. Problemstellung: Forschungsfrage:

(Wie bin ich vorgegangen?) Methoden:

(Was sind meine wichtigsten Ergebnisse?) Ergebnisse/Fazit:

(Was bedeuten meine Ergebnisse?) Diskussionsgrundlage/Empfehlung:

Das ACME Protokoll wird verwendet um es Servern zu ermöglichen

Tabelle 1: Informationsverlust in %, realer Datensatz

k	l	t	Informationsverlust
2	-	-	-64,288 %
3	-	-	-69,002 %
4	-	-	-69,008 %
4	3	-	-69,233 %
4	5	-	-69,652 %
4	-	0,7	-70,164 %
4	-	0,1	-70,409 %
4	-	0,01	-82,463 %

# Inhaltsverzeichnis

<b>Zusammenfassung</b>	<b>i</b>
<b>Abbildungsverzeichnis</b>	<b>iii</b>
<b>Tabellenverzeichnis</b>	<b>iv</b>
<b>Abkürzungsverzeichnis</b>	<b>v</b>
<b>1 Einführung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problemstellung . . . . .	1
1.3 Verwandte Arbeiten . . . . .	2
1.4 Übersicht über die Bachelorarbeit . . . . .	2
<b>2 Grundlagen</b>	<b>3</b>

## INHALTSVERZEICHNIS

---

2.1	TPM Chip . . . . .	3
2.2	Raspberrypi . . . . .	4
2.3	ACME . . . . .	4
2.3.1	Hintergrund . . . . .	5
2.3.2	Ablauf . . . . .	5
2.3.2.1	Die erste Nonce . . . . .	6
2.3.2.2	Account erstellen . . . . .	7
2.3.2.3	Order platzieren . . . . .	7
2.3.2.4	Challenge aktivieren . . . . .	7
2.3.2.5	Challenge erfüllen . . . . .	8
2.3.2.6	CSR wird an den Server geschickt . . . . .	8
2.3.2.7	Certifikat wird erhalten . . . . .	8
2.3.2.8	Weitere mögliche Schritte . . . . .	8
<b>3</b>	<b>Theoretische Umsetzung</b>	<b>9</b>
3.1	Aufbau der neuen ACME Challenge . . . . .	9
3.1.1	Vorbereitung . . . . .	9
3.1.2	Account erstellen und verwalten . . . . .	10
3.1.3	Challenge . . . . .	10
3.1.4	CSR . . . . .	11
3.2	Erweiterung des ACME Protokolls . . . . .	11
3.2.1	Clientseitig . . . . .	11
3.2.2	Serverseitig . . . . .	11
<b>4</b>	<b>Umsetzung</b>	<b>12</b>
4.1	Architektur . . . . .	12
4.1.1	Einrichtung des Raspberry Pi inklusive TPM Chip . . . . .	12
4.1.2	Aufsetzen des ACME Servers . . . . .	12
4.2	Implementierung . . . . .	13
4.2.1	Implementierung des regulären ACME Ablaufs . . . . .	13
4.2.2	Erweiterung des ACME Protokolls um die neue Challenge	14
4.2.2.1	Vorbereitungen . . . . .	14
4.2.2.2	Order platzieren . . . . .	15

---

## INHALTSVERZEICHNIS

---

4.2.2.3	Challenge aktivieren . . . . .	15
4.2.2.4	Challenge erfüllen . . . . .	15
4.2.2.5	CSR wird an den Server geschickt . . . . .	15
4.2.2.6	Certifikat wird erhalten . . . . .	15
<b>5</b>	<b>Evaluation</b>	<b>16</b>
5.1	Testlauf der ACME Erweiterung . . . . .	16
5.2	Vergleich mit DNS und HTTP Challenges . . . . .	16
5.3	Angriffsvektoren . . . . .	16
5.3.1	Mögliche Angriffe . . . . .	16
5.3.2	Schutzmechanismen . . . . .	16
5.4	Mögliche Erweiterungen . . . . .	16
5.5	Ergebnisse . . . . .	16
5.6	Auseinandersetzung . . . . .	17
<b>6</b>	<b>Fazit</b>	<b>18</b>
6.1	Zusammenfassung . . . . .	18
6.2	Future Work . . . . .	18
	<b>Anhang 1: Einige Extras</b>	<b>19</b>
	<b>Anhang 2: Noch mehr Extras</b>	<b>20</b>
	<b>Literatur</b>	<b>21</b>

---

# **Abbildungsverzeichnis**



# Tabellenverzeichnis

1	Informationsverlust in %, realer Datensatz . . . . .
---	--

# Abkürzungsverzeichnis

API: **A**pplication **P**rogramming **I**nterface

JSON: **J**ava**S**cript **O**bject **N**otation

JOSE:

JWS:

JWT:

ACME:

PI:

JWK:

TPM:

Nonce:

**API**            **A**pplication **P**rogramming **I**nterface

**JSON**        **J**ava**S**cript **O**bject **N**otation

# 1 | Einführung

## 1.1 Motivation

Nicht nur für die Kommunikation im Internet werden Zertifikate zur Prüfung der Authentizität der Kommunikationspartner verwendet. Zertifikate und Schlüssel stellen einen zentralen Bestandteil der Sicherheit in der Informatik dar und deren Verlust kann schwerwiegende Folgen haben. Beispielsweise können Geräte und Nutzer nicht mehr eindeutig identifiziert werden und jede Kommunikation zu und mit diesen wird unsicher. Das erlaubt es Angreifern sich als normalerweise vertrauenswürdige Kommunikationspartner zu tarnen um u.a. private Daten abzufragen oder Industriespionage zu betreiben. Dabei wird unterschieden zwischen der Prüfung von Nutzern und Endgeräten. Eine Kommunikation kann von Anfang an abgelehnt werden, wenn sicher gestellt werden kann, dass das anfragende Gerät nicht vertrauenswürdig ist. Aus diesem Grund muss eine Möglichkeit geschaffen werden Zertifikate sicher bereitzustellen und abzuspeichern, ohne dass diese von einem Nutzer manipuliert werden könnten.

## 1.2 Problemstellung

Es muss sicher gestellt werden, dass das Zertifikat nicht missbraucht werden kann ohne dessen Nutzen einzuschränken. Wäre die Verwendung des Zertifikates zu kompliziert, ist das Verfahren nutzlos, da es unbrauchbar wird. Gleichzeitig muss ein Grad an Sicherheit vorherrschen der es so schwer wie möglich macht das

Zertifikat zu missbrauchen. Das gilt auch für den Initialen Prozess des anfragens des Zertifikates, so wie dessen aktualisierung. Es muss also ein Gleichgewicht zwischen Funktionalität, Sicherheit sowie Verwaltbarkeit geben.

## **1.3 Verwandte Arbeiten**

(TODO: microsoft projekt angeben)

## **1.4 Übersicht über die Bachelorarbeit**

## 2 | Grundlagen

### 2.1 TPM Chip

Das TPM ist ein Chip mit dem Funktionen der Sicherheit auf einer Physischen Ebene umgesetzt werden sollen. Auch wenn der Chip erstmals 2009 von der International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC) [1] beschrieben wurde, existierten bereits 2011 300 Millionen dieser Chips[2]. Eine Zahl die mit der Ankündigung von Windows, für ihr neues Betriebssystem Windows 11 TPM2.0 Chips als Anforderung zu stellen, wahrscheinlich weiter steigen wird[3]. Die Idee des TPM Chips ist dabei die Limitationen und Probleme die sicherheits Software und standartmäßige Hardware haben zu beheben. Darunter fallen unsicherer Speicher, volatiler Speicher sowie unsichere Kryptographische Hardware. So können Sensible Daten auf dem gesicherten persistenten Speicher des Chips abgespeichert werden. Zusätzlich stellt der Chip neben dem Speicher auch eine Kryptographische Einheit zur Verfügung, die es dem Chip erlaubt durch verschiedene cryptographische Verfahren, Schlüssel zu erzeugen so wie Informationen zu ver- und entschlüsseln. Zu verwendung des Chips verfügt dieser über eine eigene umfassende API. Die Funktionalitäten und Funktionsweisen des Chips sind dabei auf über 700 Seiten Dokumentation zusammengefasst[2]. Für diese Arbeit reicht es zu verstehen, dass der Chip, je nach gewähltem Verfahren ein Schlüsselpaar erstellt kann indem nur der Public Key aus dem Chip gelesen werden kann, der Private Key diesen jedoch, da er im Chip erstellt und direkt im gesicherten Speicher gespeichert wird, nie verlässt. Diese Tatsache kann verwendet werden um das Gerät welchen den TPM Chip

verwendet an diesem zu identifizieren. Jeder Chip verfügt über einen sogenannten Endorsment Key. Dabei handelt es sich um drei Teile: einen private Key im gesicherten Speicher auf den nicht zugegriffen werden kann, einen public Key auf den zugegriffen werden kann, sowie ein Certifikat welches vom Hersteller signiert wurde.[4] An diesem Certifikat in Verbindung mit dem entsprechenden private Key kann der TPM Chip eindeutig identifiziert werden.

(TODO: MARKIEREN 1:1 aus A Formal Analysis of Authentication in the TPM)

## 2.2 Raspberrypi

Der Raspberrypi ist ein kleiner vollwertiger Computer. Er wird hier einerseits verwendet, da nichts vorinstalliert oder eingerichtet wurde, andererseits stellt der Pi Pins zur Verfügung an denen ein TPM Chip installiert werden kann. Wie bereits angedeutet gibt es bereits eine vergleichbare Lösung für Windows Systeme, also wird ein Linuxsystem verwendet werden. Der Raspberrypi stellt also die Grundlage für ein System auf dem das Verfahren entwickelt werden soll.

## 2.3 ACME

ACME steht für Automatic Certificate Management Enviroment, also eine Automatische Zertifikats Management Umgebung. Diese besteht aus zwei Teilen: erstens einer CA welche Zertifikate ausstellt und verwaltet und zweitens einer automatisierten Schnittstelle welche erst prüft ob Anfragen valide sind und diese dann an die CA weiter gibt. Dadurch können Zertifikate Automatisch anzufragen werden. Dazu wird auf dem Gerät welches ein Zertifikat benötigt ein ACME Client ausgeführt, welcher eine Anfrage an den entsprechenden ACME Server stellt. Dieser prüft dann mit sogenannten Challenges, dass der Client tatsächlich ist für den er sich ausgibt. Ist der Client in der Lage die Challenge zu erfüllen kann dieser ein Zertifikat anfragen. Dieses Prinzip soll sich für diese Bachelorarbeit zu nutze gemacht werden. Auch hier soll mit einem Automatisierten Verfahren erst eine

---

Prüfung des Client getätigt werden und dann ein Zertifikat ausgestellt werden. Der ACME Server arbeitet dabei mit einer Datenbank in der jeder Client der ein Zertifikat anfragen möchte erstmal einen Account erstellen muss. Für diesen Account kann der Client dann Anfragen nach Zertifikaten schicken, muss dabei jedoch für jede Anfrage eine Challenge erfüllen. Die Art der Challenge, ob nun DNS oder HTTPS kann der Client dabei frei wählen.

### 2.3.1 Hintergrund

Ein ACME Server der nicht gleichzeitig als Webserver dient muss laut RFC-8555 mindestens eine Schnittstelle zur Verfügung stellen, die unverschlüsselt erreicht werden kann. Diese dient als Directory und Übersicht über alle URLs die benötigt werden, damit der Client mit dem Server Kommunizieren kann. Darunter finden sich unter anderem URLs um ein Zertifikat zu widerrufen, eine Replay-Nonce zu erhalten und einen Account zu erstellen. Jede Kommunikation mit Ausnahme des GET-Requests zum erhalten des Directories sowie dem erhalten der Replay-Nonce ist verschlüsselt und muss mit einer Replay-Nonce abgesichert werden. Dabei übersendet der Server bei jeder Anfrage des Client auch eine Replay Nonce im Header des Responses mit, sodass diese nicht jedes mal neu angefragt werden muss. Jede Kommunikation, ausgenommen der genannten zwei, muss als POST oder POST-as-GET Request durchgeführt werden. POST-as-GET bedeutet, dass jede Anfrage die normalerweise ein GET-Request wäre, nun als POST Request mit leerem, aber Verschlüsseltem body, geschickt wird.

POST-as-GET Requests sind unabdingbar, da jede Kommunikation durch JWS gesichert ist. Die Schlüsselpaare die für die Kommunikation notwendig sind müssen vorher Clientseitig erstellt werden.

### 2.3.2 Ablauf

Folgend soll der Typische Ablauf einer ACME Kommunikation von der ersten Nachricht, bis zum erhalten des Zertifikates dargestellt werden. Der grundsätz-

---

liche Ablauf des ACME Protokolls ändert sich abhängig davon ob der Client bereits über ein Account auf dem Server verfügt. Hier wird jedoch davon ausgegangen, dass Client und Server noch keinerlei Kontakt mit einander hatten. Der Einfachheit halber wird die Kommunikation aus Sicht des Clients dargestellt und die internen Abläufe des ACME Servers, da sie von Server zu Server unterschiedlich sein können, außer Acht gelassen. Zu allererst muss der Client eine Anfrage an das Directory stellen um zu erfahren welche URL für welche Kommunikation benötigt wird. Sind die URLs bereits bekannt, kann dieser vorbereitungs Schritt übersprungen werden.

### 2.3.2.1 Die erste Nonce

Die Nonce wird im weiteren Ablauf des Protokolls in jedem Response des Servers mitgeschickt, damit der Client nicht wieder eine neue Anfrage senden muss. Dadurch entsteht eine Kette die aus Anfrage des Clients mit erhaltener Nonce und Antwort des Servers mit neuer Nonce besteht. Wenn die Nonce jedoch abgelaufen ist, da die letzte Kommunikation länger zurückliegt, oder der Client noch gar keine Anfrage gestellt hat und somit noch gar keine Nonce erhalten hat, kann der Client eine neue Nonce anfordern. Dazu schickt der Client einen HEAD Request an den Server an die über das Directory erhaltene URL. Da jede Kommunikation die nun beschrieben wird eine Replay-Nonce verwendet wurde darauf verzichtet dies immer wieder anzuführen. Bilder können mit der folgenden Syntax eingefügt werden:

```
HEAD /acme/new-nonce HTTP/1.1
```

```
Host: example.com
```

```
HTTP/1.1 200 OK
```

```
Replay-Nonce: oFvn1FP1wIhR1YS2jTaXbA
```

```
Cache-Control: no-store
```

```
Link: <https://example.com/acme/directory>;rel="index"
```

den:



### 2.3.2.2 Account erstellen

Jeder Request wird fest an einen Account gebunden und so muss ein neuer Account erstellt werden. Dazu sendet der Client in seiner JWS Payload Mailadressen die mit diesem Account verknüpft werden sollen, sowie eine Bestätigung dass der Client mit den Nutzungsbedingungen einverstanden ist an den Server. Optional kann hier auch ein bereits vorhandenes Konto verknüpft werden. Da noch keine KID vorhanden ist wird hier im Header an dessen Stelle JWK mit dem entsprechenden Privaten Schlüssel an den Server gesendet. Der Server antwortet in der Payload mit dem Status des Accounts, sowie mit der KID des Client und der entsprechenden URL des Accounts.

### 2.3.2.3 Order platzieren

Mit den im letzten Schritt erhaltenen Informationen kann der Client nun eine Order aufgeben. Dazu sendet er in der Payload ein Array mit allen Ordnern die er aufgeben möchte, sowie einen Zeitlichen Rahmen in dem die Validierung durchgeführt werden soll, an den Server. Dabei wird in dem Array nicht nur definiert mit welchem Verfahren sondern auch mit gegen welchen Wert die Validierung stattfinden soll. In der Antwort schickt der Server den Status, wann die Anfrage ungültig wird, sowie ein Array an Links zur Validierung der Order. Für jede Order wird ein Link erstellt. Zusätzlich antwortet der Server mit einer Finalisierungs URL die im späteren Verlauf benötigt wird.

### 2.3.2.4 Challenge aktivieren

Je nachdem welchen Challenge Typ es geht, ist dieser Schritt unterschiedlich. Bei der DNS Challenge wird nun eine Nachricht an den Server geschickt um ihn wissen zu lassen dass jetzt eine Validierung durchgeführt werden kann. Der Server antwortet mit einem Wert den der Client entschlüsselt in einem TXT Dokument, für den Server zugänglich abspeichert. Der Server kann nun dieses Dokument anfragen und so bestimmen dass der Client tatsächlich die Kontrolle über die

---

angegebene DNS besitzt.

#### **2.3.2.5 Challenge erfüllen**

Auch dieser Schritt ist abhängig von der jeweiligen Challenge, grob kann gesagt werden, dass die Challenge erfüllt wird und der Server den Status für diese Challenge anpasst. Erst wenn der Status der Challenge von "pending" zu "processing" und abschließend zu valid geändert wird, kann der Client hierfür ein

#### **2.3.2.6 CSR wird an den Server geschickt**

Ist der Status für diese Order als Valid gesetzt, also wurden alle Challenges erfüllt, so kann der Client sein Zertifikat anfragen. Dazu sendet er dem Server eine CSR, welcher dieser verwendet um ein Zertifikat zu erstellen.

#### **2.3.2.7 Zertifikat wird erhalten**

Das im vorherigen Schritt erstellte Zertifikat wird auf dem Server gespeichert und kann vom Client angefordert werden. Dazu sendet dieser einen POST-as-GET Request an die im letzten Schritt mitgeteilte URL. Der Server antwortet mit dem Zertifikat im Body der Antwort.

#### **2.3.2.8 Weitere mögliche Schritte**

Damit ist die Kommunikation abgeschlossen. Der Client kann nun über den erstellten Account die Zertifikate aktualisieren lassen, ohne die Challenges noch einmal durchlaufen zu müssen. Der Client kann den Server bitten den Account zu löschen oder ein Zertifikat zu widerrufen.

## 3 | Theoretische Umsetzung

### 3.1 Aufbau der neuen ACME Challenge

Wie im Grundlagen Kapitel beschrieben ist die Challenge das Herzstück des ACME Protokolls. Das Ziel dieser Bachelorarbeit besteht darin, eine neue Challenge zu erstellen die sich an den anderen beiden Challenges orientiert. Es soll die gleiche Sicherheit mit POST und POST-as-GET Requests erzeugt werden und der Ablauf soll vergleichbar sein. Auch die Replay Noncen sollen weiterhin verwendet werden. Der Klassische Aufbau in dem erst ein Account erstellt wird, dann eine Challenge erzeugt, diese dann beantwortet und dann das Zertifikat mithilfe eines CSR erzeugt wird bleibt also erhalten.

#### 3.1.1 Vorbereitung

Gegensätzlich zu den anderen beiden Challenge Arten soll die neue Challenge Kontrolle über das Gerät für welches sich der Client ausgibt. Der erste Schritt besteht darin entweder vom Hersteller oder per Hand den Public Key des EK aus dem TPM Chip zu erhalten. Der sogenannte EK ist ein fester Bestandteil des TPM Chips und wird bei dessen Erstellung vom Hersteller mit eingebaut. Der Private Teil kann dabei von außen weder auslesen werden, noch angefragt werden. Diese Tatsache wird verwendet um das Gerät zu identifizieren. Der Öffentliche Teil des EK wird Serverseitig mit einer für dieses Gerät festgelegten Domain gespeichert. Dadurch kann der Server abgleichen ob eine Anfrage tatsächlich von

einem Gerät kommt welches Serverseitig registriert ist und die Kommunikation frühzeitig beenden, falls das nicht der Fall ist.

### 3.1.2 Account erstellen und verwalten

Die Kommunikation zum erstellen und verwalten des Accounts läuft ab wie im Grundlagenkapitel beschrieben, hier ändert sich für den Client so wie für den Server nichts.

### 3.1.3 Challenge

Die Challenge besteht aus zwei Teilen, erst dem Absenden der Order, dann dem Erfüllen der Challenge. Für die Challenge wird Clientseitig ein sogenannter AK erstellt, dass ist wichtig, da der EK abhängig von der TPM Implementierung eventuell nicht in der Lage ist selbst zu Verschlüsseln. AK sowie EK werden nun zusammen als Value für den Identifier bestimmt, der Type trägt nun den Namen der neuen Challenge, hier wurde "ek", als Wert gewählt. Abgesehen von dieser Änderung wird die Anfrage regulär an den Server gesendet. Dieser kann nun überprüfen ob der EK Wert in seiner Datenbank vorkommt. Falls nein wird dem Client ein 400 Fehler zurückgegeben, kommt der EK Value jedoch vor, kann nun mit der eigentlichen Challenge begonnen werden. Dazu erstellt der Server, unter Verwendung der AK und EK Werte ein Geheimniss.

Der Client kann daraufhin mit einem POST-as-GET Request dieses Geheimniss, sowie die dazugehörige domain name erfragen. Das Geheimniss kann mithilfe des TPM Chips entschlüsselt werden und wird entschlüsselt wieder zurück and den Server gesendet. Wurde das Geheimniss korrekt entschlüsselt gilt die Challenge als erfüllt und der Server ändert den Status der Challenge von "pending" zu "processing" zu "valid", falls das Geheimniss richtig ist. Dadurch wird sichergestellt, dass der Client Kontrolle über den TPM Chip besitzt. Die Prüfung der Identität des Client Geräts ist also eine Prüfung des TPM Chips.

### **3.1.4 CSR**

Für die Erstellung des CSR verwendet der Client nun den TPM Chip, da dieser über eine eingebaute cryptographische Funktion verfügt und weil der Private Key diesen Chip nicht verlassen darf. Der Public Key wird nun mit anderen Daten die in der CSR stehen sollen, darunter auch der DNS Wert den der Server in der Datenbank hinterlegt hat, mit dem TPM Chip verschlüsselt. Diese Nachricht wird an den Server gesendet der das Zertifikat erstellt und dem Client zur Verfügung stellt, sodass dieser es per POST-as-GET Request abfragen kann. Das so erhaltene Zertifikat wird nun auf dem TPM Chip gespeichert.

## **3.2 Erweiterung des ACME Protokolls**

### **3.2.1 Clientseitig**

Hier muss zuallererst die Kommunikation zwischen Gerät und TPM Chip ermöglicht werden. Anschließend wird ein System Daemon geschaffen in dem der ACME Client verbaut ist. Dieser Daemon wird Teil des Bootprozesses des Servers und überprüft das aktuelle Zertifikat. Ist keins vorhanden wird ein neuer Account erstellt und eine Order aufgegeben. Diese Kommunikation läuft im Hintergrund ab und ist für den Nutzer des Gerätes unsichtbar. Wurde erfolgreich ein Zertifikat erhalten, wird dieses im TPM Chip gespeichert und dem Nutzer so zur Verfügung gestellt.

### **3.2.2 Serverseitig**

Der ACME Server wird um die Funktionalität der Datenbank sowie der Bearbeitung der Challenge erweitert. Dabei soll es einem Systemadministrator einfach gemacht werden die Liste der bekannten EK Public Keys zu erweitern und mit DNS Namen zu versehen.

---

## 4 | Umsetzung

### 4.1 Architektur

#### 4.1.1 Einrichtung des Raspberry Pi inklusive TPM Chip

Hierbei gilt sich an das Einrichten des Raspberry Pi sowie die Anweisung des TPM Chips zur Installation zu halten. Dazu wurde zuerst eine Linux Distribution auf dem Raspberry installiert. Dadurch kann der Chip, sobald er physisch am Board befestigt wird, ebenfalls installiert werden. In diesem Schritt der Einrichtung der Hardware muss nur sichergestellt werden, dass die Kommunikation mit dem TPM Chip nicht durch andere Prozesse blockiert wird. Ein einfacher Test wäre

```
sudo cat dev/tpm0
```

Antwortet der TPM Chip mit "resource is busy" blockiert ein anderer Prozess. Vor allem bei Geräten, auf denen bereits mit dem Chip gearbeitet wurde, muss hierauf geachtet werden.

#### 4.1.2 Aufsetzen des ACME Servers

Hierfür wurde sich bereits existierender Software bedient. Pebble ist ein ACME Server, der von letsencrypt zur Verfügung gestellt wird. Dabei handelt es sich um eine leichte Version, die für Testzwecke, aber nicht auf Live-Systemen verwendet

werden kann. Diese Stellt alle Grundlegenden Funktionen die für die Folgenden Schritte benötigt werden zur Verfügung.

## 4.2 Implementierung

Das Projekt wurde sowohl Serverseitig wie auch Clientseitig in GO geschrieben. Als Server wurde mein eigener Rechner Verwendet, der Client lief auf dem Raspberrypi, der Code für beide wurde vorrangig auf dem Rechner geschrieben. Die IP Adressen sind beispielhafte Werte.

### 4.2.1 Implementierung des regulären ACME Ablaufs

Wie bereits in den Grundlagen beschrieben bedient sich die Kommunikation zwischen dem Client und Server Replay Noncen sowie JWS. Um die Replay Nonce zu erhalten reicht es einen HEAD Request an die von Pebble für diesen Zweck bereit gestellt Schnittstelle zu senden.

```
func (n dummyNonceSource) Nonce() (string, error) {
    if globNonce != "" {
        return globNonce, nil
    }
    tr := &http.Transport{
        TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
    }
    client := &http.Client{Transport: tr}

    res, err := client.Head("https://192.168.1.2:14000/nonce-plz")
    if err != nil {
        panic(err)
    }
    ua := res.Header.Get("Replay-Nonce")
```

---

```
    return ua, nil
}
```

In dieser Methode wird zuallererst geprüft ob bereits einen Nonce, hier globNonce genannt, vorhanden ist. Falls nein wird ein Request ausgesendet und der Response Header für die Replay-Nonce ausgelesen.

Zum Signieren der Nachrichten wird Clientseitig ein Schlüsselpaar generiert. Beispielcode aus der Methode für den Account erstellungs Request:

```
var signerOpts = jose.SignerOptions{NonceSource: dummyNonceSource{}}
signerOpts.WithHeader("jwk", jose.JSONWebKey{Key: globPrivateKey.Public()})
signerOpts.WithHeader("url", signMeUpURL)
signer, err := jose.NewSigner(jose.SigningKey{Algorithm: jose.RS256, Key: globPr
if err != nil {
    panic(err)
}
```

Nur dieser Request übersendet den Public key. Im späteren Verlauf, sobald der Account erstellt wurde, wird anstelle von "jwk" die "kid" die vom Server mitgeteilt wurde verwendet.

## 4.2.2 Erweiterung des ACME Protokolls um die neue Challenge

### 4.2.2.1 Vorbereitungen

Da Pebble nur über eine Volatile Datenbank verfügt und es sich nur um einen Proof of concept handelt wurde der Öffentliche Teil des EK Hard in den Server gecodet. Serverseitig müssen keine weiteren Vorbereitungen getroffen werden.

```
const pubPEM = `
-----BEGIN RSA PUBLIC KEY-----
```

---



```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAh2o0FWso2nWgrA/6SIcJ
xznL4ZHw1rVnphcqYVChhzC8tXdZ6eZmPWbIP4xgKtZsYSAkPbo1Lf3dPF1A+G5W
xuXpE5QRn1bIo3Rx0CxLwduy/z7Eak8HNI32eb1U2jPYqCMCeLRStNjNnqZEoji4
//cqss1B1pXWJCH8VckfpSiXBvA+0Jyk5ceY83VCVYoKBwLVhRnTEFI2TeWU0FDn
136c85//Yd+Mohx9aoTyYTiC84eP0/sJoNdKaFl8JjgsqxYPFxcCguzeCacvA/Jr
Ps853EGOS152FuBj21CeB8QJUrnPabT/kFM9kBW6HQvWEgASv00FTJ421Cx80Ecv
mQIDAQAB
-----END RSA PUBLIC KEY-----`
```

#### 4.2.2.2 Order platzieren

Um den TPM Chip verifizieren zu können wird der Public Key des EK aus dem TPM Chip gelesen und auch mithilfe des Chips ein AK erstellt. Dazu wird auf das Projekt go-attestation von Google zurückgegriffen. Der AK besitzt sogenannte Attestation Parameters, die später vom Server verwendet werden können. Um diese Information zusammen mit der Art der zu verwenden Challenge zu versenden, wird ein "identifier" mit dem "type": "ek" und "value" : "[ek+ak]" gesetzt. (todo: Bilder einfügen)

#### 4.2.2.3 Challenge aktivieren

#### 4.2.2.4 Challenge erfüllen

#### 4.2.2.5 CSR wird an den Server geschickt

#### 4.2.2.6 Zertifikat wird erhalten

---

## **5 | Evaluation**

### **5.1 Testlauf der ACME Erweiterung**

### **5.2 Vergleich mit DNS und HTTP Challenges**

### **5.3 Angriffsvektoren**

#### **5.3.1 Mögliche Angriffe**

#### **5.3.2 Schutzmechanismen**

### **5.4 Mögliche Erweiterungen**

### **5.5 Ergebnisse**

Das sind die Ergebnisse. In vitae odio at libero elementum fermentum vel iaculis enim. Nullam finibus sapien in congue condimentum. Curabitur et ligula et ipsum mollis fringilla.

## 5.6 Auseinandersetzung

Abbildung ?? zeigt wie man eine Abbildung einfügen kann. Donec ut lacinia nibh. Nam tincidunt augue et tristique cursus. Vestibulum sagittis odio nisl, a malesuada turpis blandit quis. Cras ultrices metus tempor laoreet sodales. Nam molestie ipsum ac imperdiet laoreet. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

## **6 | Fazit**

In diesem Kapitel sollen die Möglichen Angriffsvektoren besprochen werden die bei dem DNS sowie dem IP verfahren bestanden, sowie neue die durch die neue Methode erst hinzugekommen sind. Dabei wird unterteilt in Vektoren die in allen Verfahren relevant sind, jene die speziell auf das neue Verfahren zugeschnitten sind und abschließend sollen mögliche Einfallstore besprochen werden.

### **6.1 Zusammenfassung**

### **6.2 Future Work**

## Anhang 1: Einige Extras

Füge Anhang 1 hier hinzu. Vivamus hendrerit rhoncus interdum. Sed ullamcorper et augue at porta. Suspendisse facilisis imperdiet urna, eu pellentesque purus suscipit in. Integer dignissim mattis ex aliquam blandit. Curabitur lobortis quam varius turpis ultrices egestas.

## Anhang 2: Noch mehr Extras

Füge Anhang 2 hier hinzu. Aliquam rhoncus mauris ac neque imperdiet, in mattis eros aliquam. Etiam sed massa et risus posuere rutrum vel et mauris. Integer id mauris sed arcu venenatis finibus. Etiam nec hendrerit purus, sed cursus nunc. Pellentesque ac luctus magna. Aenean non posuere enim, nec hendrerit lacus. Etiam lacinia facilisis tempor. Aenean dictum nunc id felis rhoncus aliquam.

# Literatur

- [1] ISO/IEC 11889-1:2009. Abgerufen 7. August 2021 von <https://www.iso.org/standard/50970.html>
- [2] Pierpaolo Degano; Sandro Etalle; Joshua Guttman. 2016. *Formal Aspects of Security and Trust*. Springer. Abgerufen 1. November 2017 von [https://github.com/tompollard/phd\\_thesis\\_markdown/tree/v1.0](https://github.com/tompollard/phd_thesis_markdown/tree/v1.0)
- [3] Christof Windeck. 2021. Trusted Platform Module 2.0 in Windows 11. Abgerufen 7. August 2021 von <https://www.heise.de/ratgeber/Trusted-Platform-Module-2-0-in-Windows-11-6135986.html>
- [4] Will Arthur; David Challener; Kenneth Goldman. 2015. *A Practical Guide to TPM 2.0*. Apress open.