

**HOCHSCHULE  
HANNOVER**  
UNIVERSITY OF  
APPLIED SCIENCES  
AND ARTS  
–  
*Fakultät IV  
Wirtschaft und  
Informatik*

# **Implementierung von IEEE 802.1X Authentifizierung in der Abteilung Informatik**

Simon Rose

Bachelor-Arbeit im Studiengang „Angewandte Informatik“

20. Dezember 2018



- Autor:** Simon Rose  
Matrikelnummer: 1381627  
[mail@simonrose.link](mailto:mail@simonrose.link)
- Erstprüfer:** Prof. Dr. Stefan Wohlfeil  
Abteilung Informatik, Fakultät IV  
Hochschule Hannover  
[stefan.wohlfeil@hs-hannover.de](mailto:stefan.wohlfeil@hs-hannover.de)
- Zweitprüfer:** Dipl. Ing. Frank Müller M.Sc  
Abteilung Informatik, Fakultät IV  
Hochschule Hannover  
[frank.mueller@hs-hannover.de](mailto:frank.mueller@hs-hannover.de)

### **Selbständigkeitserklärung**

Hiermit erkläre ich, dass ich die eingereichte Bachelor-Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Hannover, den 20. Dezember 2018

Unterschrift

# Inhaltsverzeichnis

<b>1. Einführung</b>	<b>4</b>
<b>2. Technische Hintergründe</b>	<b>5</b>
2.1. IEEE 802.1X	5
2.2. Extensible Authentication Protocol	7
2.3. Remote Authentication Dial-In User Service	11
2.4. Datenspeicher	15
<b>3. Konzept</b>	<b>17</b>
3.1. Status quo des Netzwerks	17
3.2. Auswahl der EAP-Methoden	20
3.3. RADIUS-Accounting	22
3.4. Umgang mit Geräten ohne 802.1X-Support	23
3.5. Hochverfügbarkeit des RADIUS-Dienstes	24
3.6. Softwareauswahl	27
<b>4. Realisierung des Konzepts</b>	<b>32</b>
4.1. FreeRADIUS	32
4.2. Konfiguration der Authentifizierungsserver	37
4.3. Konfiguration der Switches	46
4.4. Konfiguration der Supplicants	47
<b>5. Fazit</b>	<b>52</b>
<b>Glossar</b>	<b>55</b>
<b>Literatur</b>	<b>56</b>
<b>A. Dokumentation der Authentifizierungsserver</b>	<b>62</b>
<b>B. Konfigurationsanleitung für die 802.1X-Authentifizierung</b>	<b>81</b>
<b>C. Datenträger</b>	<b>102</b>

# 1. Einführung

Computernetzwerke sind schon seit vielen Jahren ein nicht mehr wegzudenkender Teil der Infrastruktur nahezu aller Unternehmen und Institutionen. Sie werden genutzt um sowohl öffentliche als auch private und sicherheitskritische Informationen bereitzustellen. Aus diesem Grund ist die Netzwerksicherheit immer ein relevantes Thema, das sehr viele Aspekte hat. Neben einer gesicherten Übertragung von Daten, ist die *Netzwerkzugriffskontrolle* ein wichtiger Teil der Netzwerksicherheit. Insbesondere für öffentlich zugängliche Institutionen, wie die Hochschule Hannover, ist es wichtig, den Netzwerkzugriff zu beschränken.

Zur Zeit wird im kabelgebundenen Netzwerk der Abteilung Informatik der Hochschule Hannover ein Sicherheitskonzept auf Basis von [MAC](#)-Adressen genutzt. Dieses Konzept bietet nur ein geringes Maß an Sicherheit und hält einem gezielten Angriff nicht Stand. Eine effektive Netzwerkzugriffskontrolle findet nicht statt. Eine bessere Alternative ist der Standard [IEEE 802.1X](#), der eine Netzwerkzugriffskontrolle unter Verwendung verschiedener Authentifizierungsmethoden ermöglicht.

Das Ziel dieser Arbeit ist die Erstellung eines Konzepts für die Implementierung dieses Standards im Netzwerk der Abteilung Informatik. Dieses Konzept soll gewährleisten, dass ein Netzwerkzugriff ausschließlich für autorisierte Geräte und Personen möglich ist. Zu diesem Zweck wird analysiert, welche Teile des Netzwerks von 802.1X profitieren und welche Authentifizierungsmethoden sich für diese am besten eignen. Bei der Erstellung des Konzepts werden unterschiedliche Möglichkeiten zum Umgang mit Geräten ohne Unterstützung für den 802.1X-Standard geprüft. Darüberhinaus wird auch eine Hochverfügbarkeitslösung für den Authentifizierungsdienst erarbeitet, um sicherzustellen, dass ein Netzwerkzugriff auch nach der Implementierung von 802.1X jederzeit möglich ist. Abschließend wird die Realisierbarkeit des Konzepts durch die Implementierung in einer Testumgebung geprüft.

## 2. Technische Hintergründe

In den folgenden Abschnitten werden die technischen Grundlagen erläutert. Der IEEE-802.1X-Standard wird in [Abschnitt 2.1](#) erklärt, während in den [Abschnitten 2.2 bis 2.4](#) die Protokolle und Konzepte erläutert werden, auf denen der Standard aufbaut.

### 2.1. IEEE 802.1X

IEEE 802.1X – im Folgenden 802.1X genannt – ist ein Standard für „Network Access Control“ (NAC; deutsch: *Netzwerkzugriffskontrolle*). Durch 802.1X wird sichergestellt, dass ausschließlich Geräte und Personen, die sich authentifiziert haben, Zugriff auf das Netzwerk erhalten.

#### 2.1.1. Beteiligte Geräte

Der 802.1X-Standard [[LAN10](#)] definiert drei verschiedene Gerätegruppen. [Abb. 1](#) ist angelehnt an eine Grafik von Cudbard-Bell [[Cud10](#)] und zeigt die beteiligten Gerätegruppen und Protokolle.

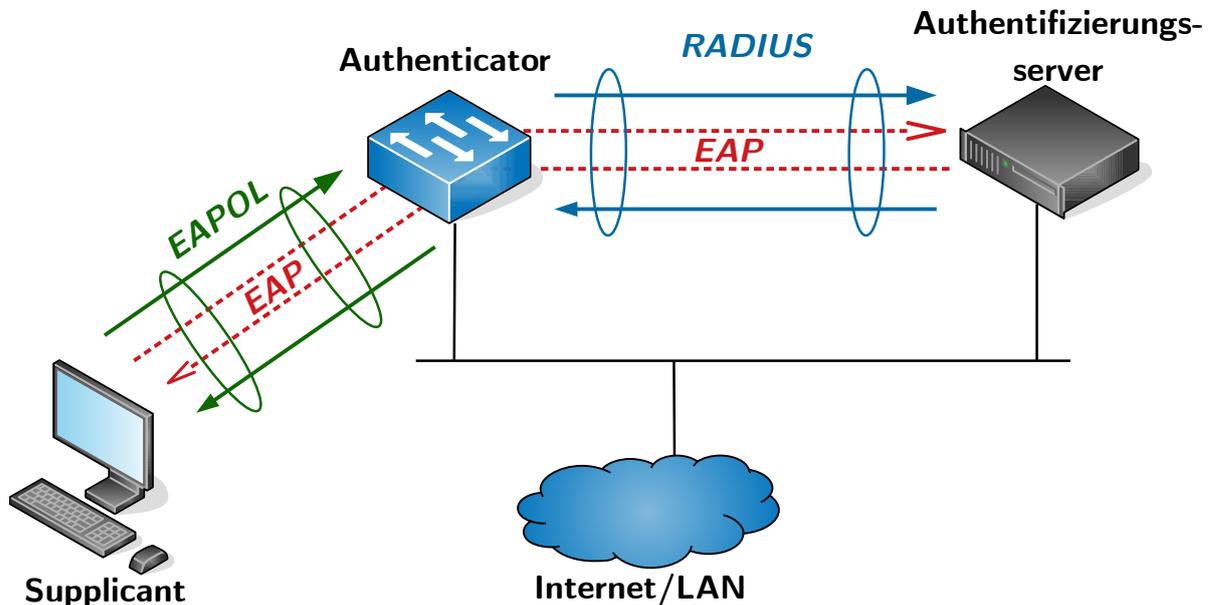


Abbildung 1: Schematische Darstellung der beteiligten Geräte und Protokolle bei der 802.1X-Authentifizierung

Der **Supplicant** (deutsch: *Bittsteller*) ist das Gerät, welches sich authentifizieren möchte. [vgl. [LAN10](#)] In der Regel ist dies eine Software auf einem Client-Gerät, wie zum Beispiel ein Notebook oder ein Mobiltelefon. Im Kontext des EAP-Protokolls, das in

[Abschnitt 2.2](#) genauer erläutert wird, verwendet man statt Supplicant den Begriff „Peer“. [[Abo+04](#)] In dieser Arbeit werden jedoch die Begriffe des 802.1X-Standards genutzt. Der **Authenticator** (deutsch: *Beglaubiger*) ist das Gerät, welches den Supplicant zur Authentifizierung auffordert. [vgl. [LAN10](#)] In der Regel ist dies das Netzwerkgerät, mit dem der Supplicant direkt verbunden ist, wie zum Beispiel ein Switch oder ein Access-Point. Der Authenticator schützt das Netzwerk, indem alle Netzwerkzugriffe blockiert werden, bis die Authentizität des Supplicants bestätigt werden konnte. Der 802.1X-Standard beschreibt den „Back End Authentication Server“ (deutsch: *Back End Authentifizierungsserver*), im folgenden **Authentifizierungsserver** genannt, als das Gerät, welches für den Authenticator die Authentifizierung des Supplicants durchführt. Anhand der vom Supplicant bereitgestellten Anmeldedaten entscheidet der Authentifizierungsserver, ob der Supplicant berechtigt ist auf die vom Authenticator bereitgestellten Netzwerkressourcen zuzugreifen [vgl. [LAN10](#)]

### 2.1.2. Zugrundeliegende Protokolle

802.1X baut auf zwei etablierten Protokollen auf – EAP und RADIUS. Zur Authentifizierung wird EAP genutzt, welches in [Abschnitt 2.2](#) genauer erläutert wird. Da EAP nicht direkt über das LAN übertragen werden kann, definiert 802.1X das Protokoll „EAP over LAN“ (EAPOL). [vgl. [LAN10](#)] Wie in [Abb. 1](#) zu sehen ist, werden die EAP-Pakete innerhalb von EAPOL-Paketen gekapselt, was eine Kommunikation zwischen Supplicant und Authenticator über das LAN ermöglicht. Sobald ein Supplicant an den Authenticator angeschlossen wird, fordert der Authenticator den Supplicant dazu auf sich über EAP zu authentifizieren. Solange dies nicht geschehen ist, wird jedes vom Supplicant verschickte Paket, das nicht zum EAP-Authentifizierungsprozess gehört, vom Authenticator verworfen. Wenn der Supplicant seine Anmeldedaten bereitgestellt hat, müssen diese an den Authentifizierungsserver weitergegeben werden. Dabei müssen die EAP-Pakete in einem anderen Protokoll gekapselt werden (vgl. [Abb. 1](#)) Diese Aufgabe übernimmt das Protokoll RADIUS, auf das in [Abschnitt 2.3](#) genauer eingegangen wird. [vgl. [LAN10](#)] Der Authentifizierungsserver muss nun die Anmeldedaten des Supplicants überprüfen. Je nach Art der Anmeldedaten, muss der Authentifizierungsserver auf verschiedene Datenspeicher zugreifen, um sie zu verifizieren. Die verschiedenen Arten von Anmeldedaten werden in [Abschnitt 2.2.3](#) erläutert. Auf die Datenspeicher wird in [Abschnitt 2.4](#) genauer eingegangen. Das Ergebnis der Authentifizierung wird über das RADIUS-Protokoll an den Authenticator zurückgeschickt. Wenn die Authentifizierung erfolgreich war, öffnet der Authenticator den Port, an dem der Supplicant angeschlossen ist. Wenn die Authentifizierung nicht erfolgreich war, bleibt der Port verschlossen und der Zugriff auf das Netzwerk wird verwehrt.

## 2.2. Extensible Authentication Protocol

Das **Extensible Authentication Protocol (EAP)** (deutsch: *Erweiterbares Authentifizierungsprotokoll*) ist ein Authentifizierungs-Framework und ist in dem [RFC 3748 \[Abo+04\]](#) definiert. Dies bedeutet, dass es keine konkrete Implementierung einer speziellen Authentifizierungsmethode ist, sondern vielmehr nur die Kommunikation zwischen den am Authentifizierungsprozess teilnehmenden Geräten regelt. Die eigentliche Authentifizierung wird von den sogenannten *EAP-Methoden* festgelegt. [vgl. [Abo+04](#)] Diese Methoden implementieren zum Beispiel Authentifizierung auf Basis von digitalen Zertifikaten oder Passwörtern. Auf die verschiedenen EAP-Methoden wird in [Abschnitt 2.2.3](#) genauer eingegangen. Unabhängig davon, welche konkrete EAP-Methode zur Authentifizierung genutzt wird, ist der von EAP festgelegte Ablauf der Kommunikation gleich.

### 2.2.1. Aufbau von EAP-Paketen

Üblicherweise wird EAP direkt über die Protokolle der Sicherungsschicht (engl.: *Data Link Layer*) des OSI-Modells, wie zum Beispiel Ethernet und [PPP](#) übertragen. Protokolle der Vermittlungsschicht (engl.: *Network Layer*), wie zum Beispiel IP, werden nicht zur Übertragung benötigt. [vgl. [Abo+04](#)] Alle EAP-Pakete haben den gleichen Aufbau. Die Felder und deren Größe sind in [Abb. 2](#) dargestellt.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Packet Code (1 Byte)								Identifizier (1 Byte)							
Length (2 Byte)															
Type (1 Byte)								Type-Data (Variable Länge)							

Abbildung 2: Aufbau eines EAP-Pakets [vgl. [Abo+04](#)]

Im Folgenden wird die Bedeutung der einzelnen Felder kurz erläutert. Im ersten Byte eines EAP-Pakets wird der sogenannte **Packet Code** (deutsch: *Paketkennzahl*) gespeichert. Die Paketkennzahl gibt die Art des EAP-Pakets an. Für den Authentifizierungsprozess werden die in [Tabelle 1](#) dargestellten Kennzahlen genutzt. In dem [RFC 3748 \[Abo+04\]](#) wird ein Paket mit der Kennzahl 1 (**Request**) als Aufforderung zur Authentifizierung definiert. Die Kennzahl 2 (**Response**) ist eine Antwort auf eine solche Aufforderung. 3 (**Success**) wird bei einer erfolgreichen Authentifizierung genutzt und 4 (**Failure**) steht für eine fehlgeschlagene Authentifizierung. [vgl. [Abo+04](#)]

Im zweiten Byte wird eine Identifikationsnummer, der sogenannte **Identifizier**, gespeichert. Dieser dient dazu Antwortpakete den dazugehörigen Aufforderungspaketen zuzuordnen. [vgl. [Abo+04](#)] Die Identifikationsnummer muss bei beiden Paketen gleich sein. Des Weiteren wird die Identifikationsnummer genutzt, um Aufforderungspakete nach einer Zeitüberschreitung neu zu übertragen und Duplikate zu erkennen. [vgl. [Abo+04](#)] Auch hier muss die Nummer identisch sein.

Im dritten und vierten Byte wird die **Länge** des EAP-Pakets gespeichert. Alle Felder des EAP-Pakets, inklusive Paketkennzahl, Identifikationsnummer, Typ und der Länge selbst, werden mitgezählt. [vgl. Abo+04] Die Größe wird in Byte angegeben.

Das fünfte Byte ist das **Type-Feld**, welches die Art des Aufforderungs- oder Antwortpakets angibt. In dem RFC 3748 [Abo+04] werden nur die ersten drei Werte definiert, da diese eine besondere Funktion haben (vgl. Tabelle 2). Höhere Werte werden für die verschiedenen EAP-Methoden genutzt, die in Abschnitt 2.2.3 genauer erläutert werden.

Tabelle 1: EAP-Paketkennzahlen [Int18a]

Wert	Bedeutung
1	Request
2	Response
3	Success
4	Failure

Tabelle 2: EAP-Typen [Int18a]

Wert	Bedeutung
1	Identity
2	Notification
3	Legacy Nak
{4..55}	EAP-Methoden

Der Wert 1 (**Identity**) (deutsch: *Identität*) ist hierbei nicht der Account- oder Computernamen der zu authentifizierenden Person oder ihres Geräts. Die Identität kann zur Auswahl einer passenden EAP-Methode genutzt werden und enthält eine beliebige Zeichenkette. Pakete mit Typ 2 (**Notification**) (deutsch: *Benachrichtigung*) enthalten Nachrichten, die auf dem Supplicant angezeigt werden. Sie haben keinen Einfluss auf den Authentifizierungsprozess. Wert 3 (**Legacy Nak**) steht für „Negative Acknowledgement“ (deutsch: *Negative Rückmeldung*). Antwortpakete mit diesem Wert werden genutzt, um zu kennzeichnen, dass die vorherige Aufforderung nicht korrekt beantwortet werden kann. [vgl. Abo+04]

Das **Type-Data-Feld** (deutsch: *Typ-Daten*) hat keine feste Größe. Der Inhalt ist je nach Kombination aus Paketkennzahl und Typ unterschiedlich und enthält die zu übertragenden Daten. [Abo+04] Das Feld kann auch weggelassen werden, wenn es keine zu übertragenden Daten gibt. Dies ist zum Beispiel bei den Paketkennzahlen 3 (**Success**) und 4 (**Failure**) der Fall.

### 2.2.2. Ablauf der Authentifizierung

Bei der EAP-Authentifizierung, sind wie bei 802.1X *Supplicant*, *Authenticator* und *Authentifizierungsserver* involviert (vgl. Abschnitt 2.1.1). Jedoch ist nach der RFC der Authentifizierungsserver nicht zwingend erforderlich und wird nur genutzt, wenn der Authenticator die Authentifizierung nicht selbst durchführen kann. [vgl. Abo+04] Bei der Mehrzahl der EAP-Methoden wird jedoch ein Authentifizierungsserver benötigt.

Im Folgenden wird zuerst der Ablauf der Authentifizierung *ohne* separaten Authentifizierungsserver geschildert. Der EAP-Authentifizierungsprozess wird in dem RFC 3748 [Abo+04] beschrieben und läuft in vier Schritten ab. Im *ersten Schritt* wird der EAP-Authentifizierungsprozess vom Authenticator initiiert. [vgl. Abo+04] Üblicherweise

fordert der Authenticator den Supplicant zuerst auf, seine Identität zu senden. Dazu schickt er ein Paket mit der Kennzahl 1 (**Request**) und dem Typ 1 (**Identity**) an den Supplicant. Als Identifikationsnummer wird eine zufällige Zahl verwendet. In seltenen Fällen wird die Identität nicht vom Supplicant erfragt, sondern aus einer anderen Quelle, wie zum Beispiel dem Switchport, an dem das Clientgerät angeschlossen ist, ausgelesen. [vgl. [Abo+04](#)] Wenn dies der Fall ist, werden die ersten beiden Schritte übersprungen.

Im *zweiten Schritt* reagiert der Supplicant auf die Aufforderung des Authenticators, indem er ein Paket mit der Kennzahl 2 (**Response**) und dem Typ 1 (**Identity**) und der gleichen Identifikationsnummer an den Authenticator sendet. [vgl. [Abo+04](#)] In den Typ-Daten ist die Identität im Klartext enthalten.

Im *dritten Schritt* fordert der Authenticator den Supplicant dazu auf, sich mit einer bestimmten EAP-Methode zu authentifizieren. Dazu sendet er ein Paket mit der Kennzahl 1 (**Request**) und dem Typ, der zu der EAP-Methode passt, zum Beispiel 13 (**EAP-TLS**). Als Identifikationsnummer wird eine neue Zufallszahl verwendet. Welche EAP-Methode der Authenticator wählt, hängt von seiner Konfiguration ab. Wie in [Abschnitt 2.2.1](#) bereits erwähnt wurde, ist es möglich anhand der Identität eine passende EAP-Methode zu wählen. Alternativ kann der Authenticator auch eine Liste bevorzugter Methoden benutzen. Der Supplicant reagiert auf diese Aufforderung, indem er ein Paket mit der Kennzahl 2 (**Response**) und dem Typ aus dem Aufforderungspaket sendet (Beispiel: 13 (**EAP-TLS**)). Die Identifikationsnummer wird wieder aus dem Aufforderungspaket übernommen. Das Typ-Daten Feld enthält die vom Authenticator angefragten Daten. Sollte der Supplicant die vom Authenticator angeforderte EAP-Methode nicht unterstützen, ändert er im Antwortpaket den Typ auf 3 (**Legacy Nak**) und die Typ-Daten bleiben leer. In dem [RFC](#) wird erläutert, dass dieser Schritt beliebig oft wiederholt werden kann, jedoch ist es dem Authenticator erst möglich eine neue Aufforderung zu schicken, wenn er ein Antwortpaket zu der vorherigen Aufforderung erhalten hat. [vgl. [Abo+04](#)] Ein Grund für eine Wiederholung sind zum Beispiel EAP-Methoden, die eine Folge von mehreren Aufforderungs- und Antwortpaketen definieren, um die benötigten Informationen zwischen Authenticator und Supplicant auszutauschen. Ein weiterer Grund ist die Ablehnung einer EAP-Methode durch den Supplicant über das Senden des Typs 3 (**Legacy Nak**) im Antwortpaket.

Im *vierten Schritt* wird der Authentifizierungsprozess vom Authenticator durch das Senden der Paketkennzahl 3 (**Success**) oder 4 (**Failure**) an den Supplicant beendet. [vgl. [Abo+04](#)] Die Paketkennzahl 3 wird nach einer erfolgreichen Authentifizierung gesendet. Die Paketkennzahl 4 wird gesendet, wenn der Authenticator alle möglichen EAP-Methoden erfolglos ausgeschöpft hat, oder er zu viele fehlerhafte Antwortpakete erhalten hat.

Bei Verwendung eines separaten Authentifizierungsservers, arbeitet der Authenticator bei einigen oder auch allen EAP-Methoden als „Pass-Through“ (deutsch: *Durchreicher*). Bei allen Paketen dieser EAP-Methoden führt der Authenticator nur Prüfungen auf fehlerhafte Pakete durch, während der Authentifizierungsserver die eigentliche Authentifizierung durchführt. [vgl. [Abo+04](#)]

### 2.2.3. EAP-Methoden

Zur Zeit umfasst die Liste der EAP-Methoden, die von der [IANA](#) gepflegt wird, ungefähr 50 Einträge. [vgl. [Int18a](#)] Ein vollständiger Vergleich aller Methoden würde den Rahmen dieser Arbeit sprengen. Daher werden nur einige ausgewählte EAP-Methoden verglichen.

**EAP-PSK:** Diese EAP-Methode hat die Typ-Nummer 47 und ist in dem [RFC 4764](#) [[BT07](#)] definiert. Es wird ein „Pre-Shared Key“ (PSK; deutsch: *Vorher vereinbarter Schlüssel*) zur Authentifizierung genutzt. Ein PSK ist ein gemeinsamer Schlüssel, der schon vor der Authentifizierung über einen beliebigen anderen Weg zwischen Authenticator und Supplicant ausgetauscht wird. Der [RFC](#) beschreibt, dass EAP-PSK mit dem Ziel entworfen wurde, eine einfach zu implementierende und sichere beidseitige Authentifizierung zu bieten, die sich für alle Arten von Netzwerken eignet, insbesondere aber für [IEEE-802.11](#)-Netzwerke (WLAN). [vgl. [BT07](#)] Die beidseitige Authentifizierung baut darauf auf, dass sowohl der Authentifizierungsserver als auch der Supplicant im Besitz des selben PSK sind. Von diesem PSK wird ein weiterer Schlüssel abgeleitet, der für die Verschlüsselung von Nachrichten mit [AES-128](#) genutzt wird. Andere Verschlüsselungsalgorithmen oder Schlüssellängen sind in dem [RFC](#) nicht vorgesehen. Über [HMAC](#)-Nachrichten wird geprüft, dass beide Parteien den selben Schlüssel errechnet haben, was nur mit dem korrekten PSK möglich ist. [vgl. [BT07](#)]

**EAP-MD5:** Diese EAP-Methode hat die Typ-Nummer 4 und ist in dem [RFC 2284](#) [[BV98](#)] definiert. EAP-MD5 implementiert eine Authentifizierung auf Basis von Accountnamen und Passwörtern. Der [RFC](#) definiert, dass für die Authentifizierung das Protokoll [CHAP](#) mit [MD5](#) als Hash-Algorithmus verwendet wird. [[BV98](#)] Die [CHAP](#)-Authentifizierung ist in dem [RFC 1994](#) [[Sim96](#)] definiert und läuft in drei Schritten ab. Zuerst sendet der Authentifizierungsserver eine Zufallszahl, die sogenannte „Challenge“ (deutsch: *Herausforderung*), an den Supplicant. Der Supplicant bildet danach aus dem Passwort des Accounts und der Challenge eine Hashsumme mit dem namensgebenden [MD5](#)-Algorithmus und überträgt die Hashsumme zusammen mit dem Accountnamen an den Authentifizierungsserver. Der Authentifizierungsserver bildet nun selbst eine Hashsumme aus dem gespeicherten Passwort des Accounts und der Challenge und vergleicht diese Summe mit der, die er vom Supplicant erhalten hat. [vgl. [Sim96](#)] Wenn beide Hashsummen übereinstimmen, bedeutet dies, dass der Supplicant das korrekte Passwort genutzt hat.

**EAP-TLS:** Diese EAP-Methode hat die Typ-Nummer 13 und ist in dem [RFC 5216](#) [[SAH08](#)] definiert. EAP-TLS ist eine EAP-Methode, die einen Schlüsselaustausch und eine beidseitige Authentifizierung zwischen dem Authentifizierungsserver und dem Supplicant auf Basis von [TLS](#) implementiert. Die beidseitige Authentifizierung basiert auf digitalen Zertifikaten. Es wird ein [TLS](#)-Handshake (deutsch: *Handschlag*) durchgeführt, ähnlich wie man ihn von dem weit verbreiteten Protokoll [HTTPS](#) kennt. Ein wichtiger Unterschied

ist jedoch, dass bei EAP-TLS die Authentifizierung des Supplicants durch ein digitales Zertifikat zwingend erforderlich ist, während bei HTTPS die Authentifizierung des Clients optional ist. [vgl. SAH08]

**EAP-TTLS und PEAP:** Sowohl EAP-TTLS („Tunneled TLS“; deutsch *Getunneltes TLS*) als auch PEAP („Protected EAP“; deutsch: *Geschütztes EAP*) nutzen die starke Transportverschlüsselung von TLS, um ältere und unsichere Authentifizierungsprotokolle abzusichern. Anders als bei EAP-TLS, ist die Authentifizierung des Supplicants mit einem Zertifikat bei EAP-TTLS und PEAP optional. Sie sind beide zweistufige Protokolle, die zuerst einen sogenannten *äußeren Tunnel* aufbauen, um dann die Anmeldedaten über unterschiedliche Methoden im *inneren Tunnel* zu übertragen.

Allerdings unterscheiden sich die beiden Protokolle auch in einigen Punkten. PEAP hat die Typ-Nummer 25 und ist in einem Internet Draft definiert. Bei PEAP wird nach dem Aufbau des äußeren Tunnels eine zweite EAP-Authentifizierung im inneren Tunnel durchgeführt. [vgl. KPW02] Üblicherweise wird für diese innere Authentifizierung die Methode EAP-MSCHAPv2 genutzt, eine EAP-Methode, die auf Microsofts Abwandlung von CHAP aufbaut. EAP-TTLS hat die Typ-Nummer 21 und ist in dem RFC 5281 definiert. Diese EAP-Methode überträgt stattdessen im inneren Tunnel sogenannte „Attribute-Value-Pairs“ (AVPs; deutsch: *Attribut-Wert-Paare*) des Diameter-Protokolls. [vgl. FB08] Diameter ist sehr ähnlich zum RADIUS-Protokoll, das in Abschnitt 2.3 genauer erläutert wird. Die AVPs werden genutzt, um eine innere Authentifizierung mit verschiedensten Methoden zu implementieren, wie zum Beispiel CHAP, PAP, MSCHAPv2 oder auch EAP. EAP-TTLS ist also bei der inneren Authentifizierung flexibler, als das auf EAP beschränkte PEAP, aber auch deutlich komplexer.

### 2.3. Remote Authentication Dial-In User Service

**Remote Authentication Dial-In User Service (RADIUS)** ist ein AAA-Protokoll („Authentifizierung, Autorisierung und Accounting“). Es wird häufig von Internetanbietern zur Authentifizierung der Kunden genutzt, findet aber auch in anderen Bereichen Anwendung, unter anderem bei 802.1X.

#### 2.3.1. Authentifizierung, Autorisierung und Accounting

Das RADIUS-Protokoll erfüllt drei Funktionen: Das Authentifizieren und Autorisieren von Personen und Geräten, sowie das „Accounting“ (deutsch: *Buchführung*). **RADIUS-Authentifizierung** ist in dem RFC 2865 [Rig+00] definiert. Wenn eine Person Zugriff auf eine Ressource im Netzwerk verlangt, muss zuerst die Identität der Person geprüft werden. Die Übertragung der Anmeldedaten kann über verschiedene Protokolle geschehen, die in RADIUS eingebettet werden können, zum Beispiel PAP oder CHAP. [vgl. Rig+00] Üblicherweise wird jedoch das modernere und flexiblere EAP genutzt, das in Abschnitt 2.2

erläutert wurde. Die Anmeldedaten werden gegen einen passenden Datenspeicher geprüft. Auf die verschiedenen Datenspeicher wird in [Abschnitt 2.4](#) genauer eingegangen.

**RADIUS-Autorisierung** ist ebenfalls in dem [RFC 2865](#) definiert. Nachdem die Anmeldedaten der Person überprüft wurden, muss geprüft werden, ob die Person auch berechtigt ist, auf die Netzwerkressource zuzugreifen. So könnte zum Beispiel in einem Firmennetzwerk der Netzwerkzugriff in einem Abteilungsnetzwerk auf dessen Angestellte beschränkt werden oder die Nutzung eines [VPN-Zugangs](#) nur für Angestellte gestattet sein, die von zu Hause arbeiten müssen. Die Informationen zu den Zugriffsrechten einer Person können ebenfalls einem Datenspeicher entnommen werden (Siehe [Abschnitt 2.4](#)). Bevor die Person authentifiziert und autorisiert ist, wird kein Zugriff zu der Netzwerkressource gestattet.

**RADIUS-Accounting** ist in dem [RFC 2866](#) [[Rig00](#)] definiert. Accounting bedeutet, dass nach der Authentifizierung und der Autorisierung für jede Person festgehalten wird, über welchen Zeitraum die Person sich im Netzwerk aufhält. Diese Metadaten werden oft zur Netzwerküberwachung und für Statistiken verwendet oder um Personen die Netzwerknutzung in Rechnung zu stellen.

### 2.3.2. Aufbau von RADIUS-Paketen

RADIUS ist ein Client-Server-Protokoll. Clients werden im RADIUS-Kontext auch „Network Access Server“ genannt (NAS; deutsch: *Netzwerkzugriffs-Server*), jedoch wird in dieser Arbeit zugunsten der Einheitlichkeit der Begriff *Authenticator* verwendet. Aus dem gleichen Grund wird auch weiterhin der Begriff *Authentifizierungsserver* statt „RADIUS-Server“ verwendet. RADIUS gehört zu den Protokollen der Anwendungsschicht (engl.: *Application Layer*) des OSI-Modells. RADIUS-Authentifizierungspakete werden über den UDP-Port 1812 übertragen, während für RADIUS-Accounting der UDP-Port 1813 genutzt wird. [Abb. 3](#) zeigt den Aufbau eines RADIUS-Pakets.

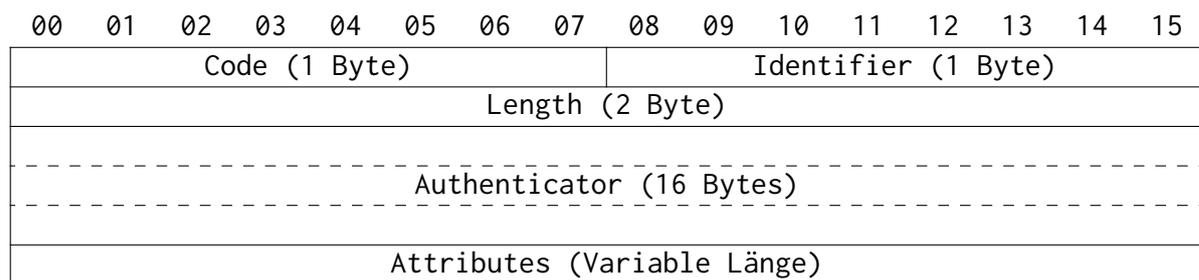


Abbildung 3: Aufbau eines RADIUS-Pakets [[Rig+00](#)]

Im Folgenden wird die Bedeutung der einzelnen Felder des Pakets kurz erklärt. Im ersten Byte wird der **Code** (deutsch: *Kennzahl*) gespeichert. Wie auch die Paketkennzahl eines EAP-Pakets, dient diese dazu, verschiedene Arten von RADIUS-Paketen zu unterscheiden (vgl [Abschnitt 2.2.1](#)). Die wichtigsten Kennzahlen werden in den [RFCs](#)

2865 und 2866 definiert. Pakete mit der Kennzahl 1 (**Access-Request**) dienen dazu, Informationen über die zu authentifizierende Person zu übertragen. Die Kennzahl 11 (**Access-Challenge**) wird genutzt, wenn der Authentifizierungsserver zusätzliche Informationen vom Authenticator benötigt, oder innerhalb der RADIUS-Pakete komplexere EAP-Methoden genutzt werden. 2 (**Access-Accept**) und 3 (**Access-Reject**) haben die Funktion, den Authenticator über eine erfolgreiche, beziehungsweise fehlgeschlagene Authentifizierung und Autorisierung zu informieren. [vgl. [Rig+00](#)] Pakete mit der Kennzahl 4 (**Accounting Request**) werden in periodischen Abständen an den Authentifizierungsserver gesendet und enthalten relevante Informationen für das Accounting. Die Kennzahl 5 (**Accounting-Response**) wird genutzt, um den Erhalt der Accounting-Informationen zu bestätigen. [vgl. [Rig00](#)]

Tabelle 3: Liste der RADIUS-Kennzahlen, die in den [RFCs](#) 2865 [[Rig+00](#)] und 2866 [[Rig00](#)] definiert werden. Eine vollständige Liste wird von der [IANA](#) [[Int18b](#)] veröffentlicht.

Wert	Bedeutung
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request
5	Accounting-Response
11	Access-Challenge

Die Felder **Identifer** und **Length** sind gleichbedeutend mit den gleichnamigen Feldern eines EAP-Pakets (vgl. [Abschnitt 2.2.1](#)). Die Identifikationsnummer markiert zusammengehörende Pakete und die Länge speichert die Größe des gesamten RADIUS-Pakets. [vgl. [Rig+00](#)]

Anhand des 16 Byte großen **Authenticator-Felds** kann der Authenticator feststellen, ob ein RADIUS-Paket von einem vertrauenswürdigen Authentifizierungsserver gesendet wurde. [vgl. [Rig+00](#)] Dadurch ist es nicht möglich, gefälschte Pakete mit der Kennung 3 (**Access-Accept**) an den Authenticator zu schicken, um Zugriff auf das Netzwerk zu bekommen.

Im **Attributes-Feld** werden die eigentlichen Nutzdaten des RADIUS-Protokolls als sogenannte „Attribute Value Pairs“ (AVPs) übertragen. Ein RADIUS-Paket kann mehrere AVPs enthalten. [Abb. 4](#) zeigt den Aufbau eines AVP. Das Attribut eines AVP ist im „Type-Feld“, der dazugehörige Wert im „Value-Feld“ gespeichert. Im „Length-Feld“ wird die Größe des gesamten AVP in Byte gespeichert. Die [IANA](#) hat über 200 verschiedene Attribute für RADIUS-AVPs definiert. [vgl. [Int18b](#)]

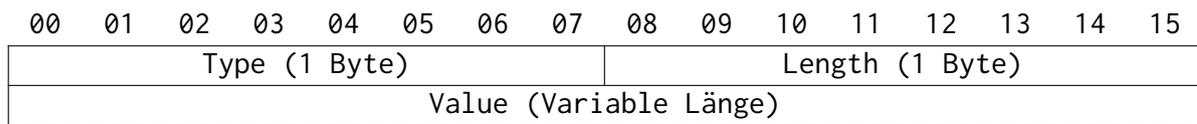


Abbildung 4: Aufbau eines RADIUS-AVP [vgl. Rig+00]

### 2.3.3. Ablauf der Kommunikation

In diesem Abschnitt wird nur der Ablauf der RADIUS-Authentifizierung beschrieben, da RADIUS-Accounting nicht der Fokus dieser Arbeit ist.

Die Kommunikation über das RADIUS-Protokoll beginnt, nachdem der Authenticator die Identität des Supplicants über EAP erhalten hat (vgl. Abschnitt 2.2.2). Der Authenticator muss die Identität an den Authentifizierungsserver weiterleiten. Dazu sendet er ein Paket mit der Kennzahl 1 (**Access-Request**) und einer zufälligen Identifikationsnummer. Wie bereits in Abschnitt 2.3.2 angedeutet wurde, muss im *Authenticator-Feld* ein Wert gesendet werden, anhand dessen der Authenticator die Vertrauenswürdigkeit des Authentifizierungsservers überprüfen kann. Die Berechnung dieses Wertes wird in dem RFC 2865 geschildert. Der Authentifizierungsserver und der Authenticator teilen ein gemeinsames Passwort  $S$ . Der Authenticator erzeugt eine 16 Byte lange Zufallszahl  $R$ . Sei  $A$  das **Access-Request**-Paket und die Verknüpfung  $\oplus$  das exklusive Oder. Dann gilt für den Authenticator-Wert: [vgl. Rig+00]

$$A_{auth} = MD5(R \oplus S)$$

Am Ende des Pakets stehen ca. ein Dutzend verschiedene AVPs. Diese enthalten unter anderem die **MAC**-Adresse des Supplicants und des Authenticators, die IP-Adresse des Authenticators und die Nummer des Ports, an dem der Supplicant angeschlossen ist. Darüber hinaus ist ein AVP mit dem Attribut 79 (**EAP-Message**) enthalten. Als zugehöriger Wert dieses AVPs ist das EAP-Identity-Response-Paket enthalten, das an den Authentifizierungsserver weitergeleitet werden muss (vgl. Code-Beispiel 1). Die Kapselung von EAP-Paketen in RADIUS-AVPs ist in dem RFC 2869 beschrieben.

```
AVP: t=EAP-Message(79) l=10 Last Segment[1]
  Type: 79
  Length: 10
  EAP fragment: 0201000801687368
  Extensible Authentication Protocol
    Code: Response (2)
    Id: 1
    Length: 8
    Type: Identity (1)
    Identity: hsh
```

Code-Beispiel 1: Ein in einem RADIUS-AVP gekapseltes EAP-Identity-Response-Paket

Der Authentifizierungsserver antwortet nur dann auf das **Access-Request**-Paket, wenn es von einem bekannten Authenticator gesendet wurde. Pakete von unbekanntem Authenticatoren – mit denen der Authentifizierungsserver kein gemeinsames Passwort

vereinbart hat – werden ohne Bestätigung fallen gelassen. [vgl. Rig+00] Das Antwortpaket hat die Kennzahl 11 (**Access-Challenge**). Der Authenticator-Wert des Pakets muss eine Überprüfung durch den Authenticator zulassen, darf aber nicht das geteilte Passwort  $S$  preisgeben. Sei  $B$  das **Access-Challenge**-Paket und die Verknüpfung  $\circ$  die Konkatenation. Dann gilt für den Authenticator-Wert: [vgl. Rig+00]

$$B_{auth} = \text{MD5}(B_{code} \circ B_{len} \circ A_{auth} \circ B_{attr} \circ S)$$

Die AVPs enthalten auch hier wieder ein EAP-Paket, welches in der Regel eine Aufforderung zur Authentifizierung ist. Diese Abfolge von **Access-Request**-Paketen und **Access-Challenge**-Paketen wird solange wiederholt, bis die EAP-Authentifizierung beendet ist. Die RADIUS-Kommunikation wird durch den Authentifizierungsserver beendet, indem das Ergebnis der EAP-Authentifizierung (**Success** oder **Failure**; vgl. [Abschnitt 2.2.2](#)) gekapselt in einem RADIUS-**Access-Accept**- oder **-Access-Reject**-Paket an den Authenticator übertragen wird.

### 2.4. Datenspeicher

In den [Abschnitten 2.1.2](#) und [2.3.1](#) wurde bereits erwähnt, dass die Anmeldedaten gegen einen sogenannten „Datenspeicher“ (engl.: *Data Store*) abgeglichen werden. Weder der 802.1X-Standard noch die [RFCs](#) zu EAP und RADIUS definieren unterschiedliche Arten von Datenspeichern, aber da sie für die Authentifizierung unerlässlich sind, werden sie an dieser Stelle erwähnt. Datenspeicher sind nur für EAP-Methoden, die anhand von Accountnamen und Passwörtern authentifizieren, erforderlich, da diese auf eine beliebig große Anzahl an Accountdaten zugreifen müssen. Grundsätzlich lassen sich die Datenspeicher in vier Gruppen einteilen:

- Verzeichnisdienste, zum Beispiel Active Directory (AD), OpenLDAP und FreeIPA
- Datenbanksysteme, zum Beispiel PostgreSQL, SQLite, Redis und MySQL
- Lokale Accounts
- Klartextdateien (*Flat Files*)

**Verzeichnisdienste** sind hierarchische Datenbanksysteme, die für die Verwaltung von großen Mengen an Accountdaten optimiert sind. Der Zugriff auf Verzeichnisdienste erfolgt üblicherweise über das offene und herstellerunabhängige „Lightweight Directory Access Protocol“ (LDAP; deutsch: *Leichtgewichtiges Verzeichniszugriffsprotokoll*). Von einigen Verzeichnisdiensten werden zusätzliche proprietäre Zugriffsmethoden implementiert, wie zum Beispiel NTLMv2, das zur Authentifizierung bei AD verwendet werden kann. Verzeichnisdienste werden oft in Unternehmensnetzwerken genutzt, da die Accountverwaltung im Vergleich zu reinen Datenbanksystemen einfacher ist und viele Dienste eine einfache Anbindung an Verzeichnisdienste bieten.

**Datenbanksysteme** sind im Gegensatz zu Verzeichnisdiensten auf die Speicherung beliebiger Datensätze ausgelegt und bieten in der Regel keine Komfortfunktionen zur vereinfachten Accountverwaltung. Der Zugriff auf die meisten Datenbanksysteme erfolgt über die Sprache „Structured Query Language“ (SQL; deutsch *Strukturierte Abfragesprache*). Die Speicherung von Accounting-Daten ist ausschließlich in Datenbanken möglich.

Bei der Verwendung von **lokalen Accounts** wird auf die Accounts zugegriffen, die sich am Betriebssystem des Authentifizierungsservers anmelden können. Auf welche Art dieser Zugriff erfolgt, hängt in diesem Fall natürlich von dem verwendeten Betriebssystem ab. Dieser Datenspeicher ist sehr einfach zu implementieren und zu pflegen. Allerdings ist es oft nicht erwünscht, dass sich jeder Account, der sich über 802.1X authentifizieren kann, auch am Authentifizierungsserver anmelden kann.

**Klartextdateien** sind für den produktiven Einsatz eher ungeeignet und werden meist nur zum Debugging verwendet. Die Datei enthält für jede Person einen Eintrag mit Accountname und Passwort. Das Passwort kann dabei gehasht sein oder auch im Klartext vorliegen. Klartextdateien als Datenspeicher sind zwar schnell und einfach implementiert, jedoch sind sie sehr unflexibel und werden bei vielen Accounts schnell unübersichtlich.

## 3. Konzept

In den folgenden Abschnitten wird das Konzept für die Implementierung von 802.1X in der Abteilung Informatik erarbeitet. Neben einer Erläuterung des aktuellen Zustands des Netzwerks in [Abschnitt 3.1](#), werden in [Abschnitt 3.2](#) die passenden EAP-Methoden für die unterschiedlichen Geräte im Netzwerk gewählt. In den [Abschnitten 3.4](#) und [3.5](#) werden verschiedene Möglichkeiten zum Umgang mit Geräten ohne 802.1X-Unterstützung, sowie zur Gewährleistung der Hochverfügbarkeit des RADIUS-Dienstes verglichen. Abschließend wird in [Abschnitt 3.6](#) die passende Software für die Umsetzung des Konzepts ausgewählt.

### 3.1. Status quo des Netzwerks

Die nächsten zwei Abschnitte erläutern den aktuellen Zustand des Netzwerks, sowie das derzeit eingesetzte Sicherheitskonzept.

#### 3.1.1. Netzwerkaufbau

Das Netzwerk der Abteilung Informatik besteht aus mehreren Teilnetzwerken, die in [Abb. 5](#) zu sehen sind.

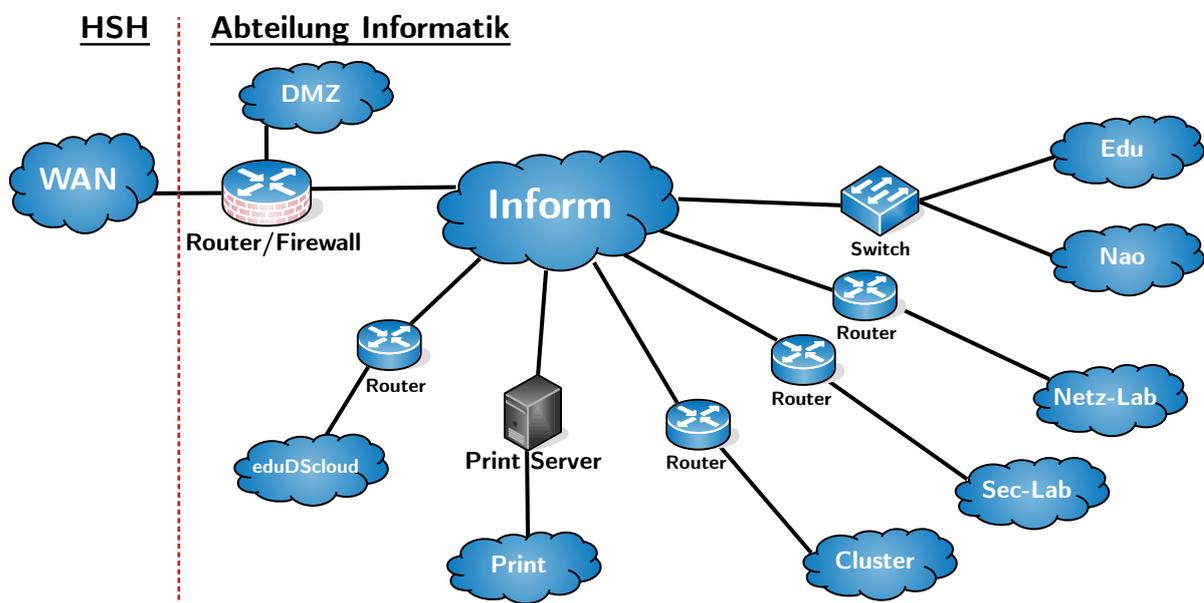


Abbildung 5: Übersicht der Netzwerke der Abteilung Informatik

Das Netzwerk ist durch eine Firewall vom Netzwerk der Hochschule Hannover getrennt. Der Großteil der Server und die PCs des Personals der Abteilung Informatik sind Teil des **Inform**-Netzes. Das Netzwerk enthält unter anderem den zentralen *DNS-Server*, sowie

den *Active Directory Domain Controller*, der die Accountdaten des Abteilungspersonals, sowie der Studierenden verwaltet. Das **Edu**- und das **Nao**-Netzwerk sind über einen Switch mit dem Inform-Netzwerk verbunden, sind aber in jeweils unterschiedlichen **VLANs**. Das Nao-Netzwerk wird für die Projekte mit den namensgebenden Nao-Robotern genutzt. Im Edu-Netzwerk befinden sich die *Pool-PCs*, sowie einige Projekträume, wie zum Beispiel das CG-Labor. Die Netzwerkdrucker sind Teil des **Print**-Netzwerks, welches nur über den Print-Server mit dem Inform-Netz verbunden ist. Der Print-Server ist Teil beider Netzwerke, leitet aber nur Druckaufträge weiter, statt zwischen den Netzwerken zu routen. Die Computer des Security-Labors und des Netze-Labors befinden sich in den Netzwerken **Sec-Lab** und **Netz-Lab**. Diese Netzwerke werden für Übungen und Projekte genutzt und sind über einen Router vom Inform-Netz getrennt. Das Netzwerk **Cluster** wird für Projekte und Experimente der Lehrenden genutzt und ist ebenfalls mit einem Router vom Inform-Netz getrennt. Das **eduDScIoud**-Netzwerk wird für das gleichnamige Projekt genutzt, um isolierte Arbeitsumgebungen für Studienprojekte bereitzustellen. [vgl. SKH18] Die **DMZ** ist über die Firewall sowohl von dem Netz der Hochschule als auch der Abteilung Informatik getrennt. Dieses Netz wird genutzt, um Dienste bereitzustellen, auf die ein Zugriff von außerhalb des Netzes der Abteilung Informatik möglich sein soll.

Die Implementierung der 802.1X-Authentifizierung ist nicht in jedem dieser Teilnetze eine geeignete Maßnahme. In Netzen, die für Projekte und experimentelle Zwecke genutzt werden, wäre 802.1X ein Hindernis. Dies betrifft die Netze Cluster, Sec-Lab und Netz-Lab. Für Netzwerke wie die DMZ und die eduDScIoud, deren Hosts sich in nicht öffentlich zugänglichen Räumen, wie dem Serverraum befinden, bietet 802.1X keinen Mehrwert, weil nicht autorisierte Personen keinen Zugriff auf die Switchports haben. Da das Print Netzwerk keine Route zum Netzwerk der Abteilung Informatik oder das Internet hat, bietet auch hier die Implementierung keinerlei Vorteile. Auch für das Nao-Netz bietet 802.1X wenig Vorteile, da die Nao-Roboter diesen Standard nicht unterstützen. Bei der Erstellung des Konzepts werden daher nur das Edu-Netz und das Inform-Netz betrachtet. Für die Sicherheit des Edu-Netzes ist 802.1X relevant, da die Pool-Räume öffentlich zugänglich und daher ungeschützt sind. Im Inform-Netz kann 802.1X genutzt werden, um die Büroräume des Abteilungspersonals abzusichern. Für andere Geräte, wie zum Beispiel die Server, sollte auch in diesem Netzwerk kein 802.1X verwendet werden, da diese ebenfalls im Serverraum stehen, der nicht öffentlich zugänglich ist.

Um die Geräte in diesen Netzwerken miteinander zu verbinden, werden verschiedene Switch-Modelle der Marke Cisco eingesetzt (s. [Tabelle 4](#)). Die ersten vier Modelle der Tabelle unterstützen den 802.1X-Standard. Diese Information kann den Datenblättern der Switches entnommen werden. [Cis18a; Cis18b; Cis18c] Das Modell SG110D-08 ist ein Switch mit acht Ports, der in einigen Räumen eingesetzt wird, in denen es nicht genügend Zuleitungen aus den Serverräumen gibt, um alle Computer anzuschließen. Statt eines Computers wird also ein SG110D-08-Switch an eine der Zuleitungen angeschlossen und die Computer werden wiederum an den SG110D-08-Switch angeschlossen. Diese Acht-Port-Switches sind *unmanaged*, verfügen also über keine erweiterten Funktionen. [Cis17] Sie können deshalb nicht als Authenticator fungieren und sind nicht für die

Tabelle 4: Übersicht der in der Abteilung Informatik eingesetzten Switchmodelle

Modell	OS-Version
Cisco WS-C3750X-48	IOS 15.0(2)SE12
Cisco WS-C3850-48T	IOS XE 16.3.7 Denali
Cisco WS-C2960L-24TS-LL	IOS 15.2(6)E1
Cisco WS-C2960L-8TS-LL	IOS 15.2(6)E1
Cisco SG110D-08	unmanaged

Implementierung von 802.1X geeignet. Für die Umsetzung des Konzepts, das in diesem Kapitel erarbeitet wird, ist es erforderlich, diese Switches durch *Managed-Switches* zu ersetzen.

### 3.1.2. Aktuelles Sicherheitskonzept

Momentan wird eine *DHCP-Weißliste* im Edu- und im Inform-Netz eingesetzt. DHCP wird in RFC 2131 [Dro97] als „Framework zur Vergabe von Konfigurations-Informationen an Geräte innerhalb eines TCP/IP-Netzwerks“ beschrieben. In der Regel gibt es in jedem Netzwerk nur einen DHCP-Server. Wenn ein Gerät an ein Netzwerk angeschlossen wird, fordert es eine Konfiguration vom DHCP-Server an. Der DHCP-Server kann dem Client neben IP-Adresse, Subnetzmaske und Standard-Gateway auch weitere Informationen, wie eine Liste von DNS- und NTP-Servern oder auch einen Host- und Domänen-Namen, zuweisen. IP-Adressen werden dabei oft aus einem *Pool* vergeben, sodass alle Clients eine zufällige IP-Adresse haben. Wenn stattdessen gewünscht ist, dass Geräte bei jeder neuen Verbindung die selbe, vorher festgelegte IP-Adresse bekommen, dann kann der DHCP-Server die Geräte anhand ihrer MAC-Adresse erkennen und IP-Adressen aus einer Liste fester Zuweisungen vergeben.

Diese Funktion wird in der Abteilung Informatik genutzt, um die bereits erwähnte DHCP-Weißliste zu implementieren. Die DHCP-Server im Edu- und Inform-Netz haben für jedes bekannte Gerät in diesen Netzwerken eine eindeutige Zuweisung von der MAC-Adresse auf eine IP-Adresse. Ein Adress-Pool zur Vergabe zufälliger IP-Adressen wird nicht genutzt. Dies hat zur Folge, dass Geräte, deren MAC-Adressen dem DHCP-Server nicht bekannt sind, keine IP-Adresse zugewiesen bekommen und somit auch keinen Zugriff zum Netzwerk haben. Wie bereits in der Einführung dieser Arbeit erwähnt wurde, lässt sich diese Sicherheitsmaßnahme sehr einfach umgehen: Zum einen sind Client-Geräte nicht dazu gezwungen, DHCP zu nutzen, was bedeutet, dass nicht autorisierte Geräte durch eine manuelle Konfiguration der Netzwerkkarte Zugriff auf das Netzwerk erhalten können. Zum anderen lassen sich MAC-Adressen sehr leicht ändern – unter GNU/Linux sind dazu nur drei Kommandozeilenbefehle notwendig.

```
ip link set dev enp1s0 down
ip link set dev enp1s0 address 18:62:2C:C0:FF:EE
ip link set dev enp1s0 up
```

Wird die **MAC**-Adresse eines nicht autorisierten Gerätes auf die eines autorisierten Gerätes abgeändert, so ist es für den DHCP-Server unmöglich, diese Fälschung der **MAC**-Adresse festzustellen. Das nicht autorisierte Gerät wird die vollständige Konfiguration des autorisierten Geräts erhalten.

## 3.2. Auswahl der EAP-Methoden

In diesem Kapitel werden die passenden EAP-Methoden für die Authentifizierung der PCs des Abteilungspersonals sowie der Pool-PCs gewählt.

### 3.2.1. Authentifizierung der PCs des Abteilungspersonals

Für die Authentifizierung der PCs des Personals der Abteilung Informatik ist es besonders wichtig, dass Mitarbeitende die 802.1X-Authentifizierung auf ihren Geräten selbst konfigurieren können. Damit dies sicher umgesetzt werden kann, müssen alle Personen ihre eigenen, persönlichen Anmeldedaten besitzen. Das Teilen der Anmeldedaten zwischen allen oder Gruppen von Mitarbeitenden hätte zur Folge, dass die Anmeldedaten geändert werden müssen, wenn eine dieser Personen kündigt. Grundsätzlich ist es möglich, alle Arten von Anmeldedaten zu personalisieren. Es wäre möglich für alle Mitarbeitenden digitale Zertifikate auszustellen, allerdings ist der Konfigurationsaufwand höher, als bei der Verwendung der Anmeldedaten, die bereits im Active Directory der Hochschule gespeichert sind. Da ohnehin für alle Mitarbeitenden ein Accountname und ein Passwort im Active Directory festgelegt werden muss, entsteht kein zusätzlicher Konfigurationsaufwand. Es sollen nur die Active-Directory-Accounts des Abteilungspersonals authentifiziert werden. Eine Authentifizierung der Studierenden-Accounts ist nicht notwendig, da die Studierenden entweder die Pool-PCs nutzen können, auf die in [Abschnitt 3.2.2](#) eingegangen wird, oder mit ihren eigenen Geräten das Eduroam-WLAN nutzen können, das bereits 802.1X nutzt. Die Verwendung des Active Directorys als Datenspeicher schränkt allerdings die Zahl der möglichen EAP-Methoden ein. Die Windows Dokumentation von Microsoft nennt drei mögliche EAP-Methoden auf Basis von Accountname und Passwort: [vgl. [Mic18](#)]

- MD5-Challenge (4)
- MS-EAP-Authentication (26)
- PEAP (25)

„MD5-Challenge“ ist der Typ-Name der EAP-Methode **EAP-MD5** (s. [Abschnitt 2.2.3](#)). EAP-MD5 ist nach der Microsoft Knowledge Base für alle Windows Betriebssysteme als veraltet eingestuft und ist standardmäßig deaktiviert. [vgl. [Mic12](#)] Darüberhinaus ist EAP-MD5 nicht sicher. 2012 zeigten Liu und Xie [[LX12](#)], dass aus den übertragenen **MD5**-Hashsummen mithilfe einer [Rainbow Table](#) das Passwort extrahiert werden

kann. [vgl. [LX12](#)] Daher ist diese EAP-Methode nicht für die Authentifizierung des Abteilungspersonals geeignet.

„MS-EAP-Authentication“ ist der Typ-Name der EAP-Methode **EAP-MSCHAPv2**, die von Kamath, Palekar und Hurst [[KPH07](#)] in einem Internet Draft beschrieben wurde. Da MSCHAPv2 eine Abwandlung des Challenge Handshake Protokolls ([CHAP](#)) ist, werden auch bei diesem Protokoll nur Hashsummen der Passwörter übertragen. Allerdings ist auch die MSCHAPv2-Authentifizierung nicht sicher, wie 1999 von Schneier, Mudge und Wagner [[SMW99](#)] gezeigt wurde. Passwörter werden vor dem hashen in Großbuchstaben umgewandelt, was die Anzahl möglicher Kombinationen deutlich verkleinert. Darüberhinaus werden lange Passwörter in sieben Byte große Stücke „zerteilt“, die einzeln gehasht werden und somit auch einzeln geknackt werden können. Aus diesen Gründen ist MSCHAPv2 sehr anfällig für [Wörterbuchangriffe](#). [vgl. [SMW99](#)] Da EAP-MSCHAPv2 lediglich MSCHAPv2-Pakete über EAP überträgt, sind alle Sicherheitslücken von MSCHAPv2 auch bei EAP-MSCHAPv2 vorhanden. [vgl. [KPH07](#)] Aufgrund dieser Lücken, ist auch diese EAP-Methode nicht für die Authentifizierung des Abteilungspersonals geeignet.

Die dritte mögliche EAP-Methode ist **PEAP**, die von Kamath, Palekar und Wodrich [[KPW02](#)] als Internet Draft definiert wurde. Wie bereits in [Abschnitt 2.2.3](#) erläutert wurde, nutzt PEAP das [TLS](#)-Protokoll, um einen sicheren äußeren Tunnel aufzubauen um darin weniger sichere EAP-Methoden zu übertragen. Wie sicher der äußere Tunnel tatsächlich ist, hängt natürlich von den verwendeten [TLS](#)-Versionen und Verschlüsselungsalgorithmen ab. Die aktuelle Version 1.3 des [TLS](#)-Protokolls ist in dem [RFC 8446](#) [[Res18](#)] definiert. [TLS](#) unterstützt verschiedene Verschlüsselungsalgorithmen. Der Algorithmus, der bei einer [TLS](#)-Verbindung genutzt wird, wird von Client und Server während des Handshakes „ausgehandelt“. [vgl. [Res18](#)] Der Authentifizierungsserver nutzt eine Implementierung des [TLS](#)-Protokolls für die Verschlüsselung, zum Beispiel *OpenSSL*. Diese Implementierungen erlauben es, über Konfigurationsoptionen festzulegen, welche [TLS](#)-Versionen und Verschlüsselungsalgorithmen unterstützt werden. Auf diese Weise kann die Aushandlung von schwachen Algorithmen und veralteten [TLS](#)-Versionen unterbunden werden. Für den inneren Tunnel können verschiedene EAP-Methoden verwendet werden, jedoch können für die Authentifizierung gegen ein Active Directory nur die bereits genannten Methoden EAP-MD5 und EAP-MSCHAPv2 genutzt werden. Da EAP-MSCHAPv2 als innerer Tunnel sowohl von Windows, macOS und GNU/Linux unterstützt wird und EAP-MD5 unter Windows veraltet ist, ist EAP-MSCHAPv2 die bessere Wahl für ein heterogenes Netzwerk. Die Kombination aus PEAP und EAP-MSCHAPv2 als innerem Tunnel wird auch *PEAP-MSCHAPv2* genannt. Da PEAP-MSCHAPv2 eine ausreichende Sicherheit bietet und mit allen relevanten Client-Betriebssystemen kompatibel ist, ist es die beste Option für die Authentifizierung des Abteilungspersonals.

### 3.2.2. Authentifizierung der Pool-PCs

Die Pool-PCs stehen allen Studierenden der Abteilung Informatik zur Verfügung und werden, anders als die PCs des Abteilungspersonals, vom IT-Team verwaltet. Als Betriebssystem wird *Ubuntu 16.04* (Xenial Xerus) verwendet. Sie sind an das Active Directory angebunden und lösen darüber Accounts und deren Gruppenzugehörigkeiten auf. Dies hat den Vorteil, dass nicht für jeden Studierenden auf jedem Pool-PC ein lokales Konto vorhanden sein muss und reduziert somit den Administrationsaufwand erheblich. Nach dem Eingeben von Accountname und Passwort in die Anmeldemaske, werden LDAP-Pakete an den *Active Directory Domain Controller* gesendet, um die Anmeldedaten zu verifizieren. Dies bedeutet, dass zum Zeitpunkt der Anmeldung bereits eine Netzwerkverbindung aufgebaut sein muss. Darüberhinaus müssen sich die zu nutzenden Anmeldedaten leicht auf alle Pool-PCs verteilen lassen. EAP-Methoden auf Basis von Accountname und Passwort sind für die Authentifizierung der Pool-PCs nicht geeignet, da es nicht möglich ist, die Anmeldedaten der Studierenden schon vor der Anmeldung am Betriebssystem für die 802.1X-Authentifizierung zu nutzen. Digitale Zertifikate eignen sich deutlich besser für diesen Zweck, da diese nach dem Systemstart sofort im Dateisystem zur Verfügung stehen und für eine 802.1X-Authentifizierung vor der Betriebssystemanmeldung genutzt werden können. Da digitale Zertifikate normale Dateien sind, lassen sie sich sehr einfach auf mehrere Pool-PCs verteilen, indem ein beliebiges Dateitransferprogramm wie `rsync` oder `sftp` verwendet wird. Bei der Verwendung von Zertifikaten muss allerdings sichergestellt werden, dass ausschließlich befugte Personen Zugriff auf den privaten Schlüssel des Zertifikats haben.

Die passende EAP-Methode für eine Authentifizierung mit digitalen Zertifikaten ist **EAP-TLS** (s. [Abschnitt 2.2.3](#)). Die beidseitige Authentifizierung zwischen dem Supplicant und dem Authentifizierungsserver macht EAP-TLS zu einer der sichersten EAP-Methoden, da sich bei den meisten Alternativen nur eine Partei authentifiziert. Bei PEAP und EAP-TTLS ist beim Aufbau des äußeren Tunnels nur die Authentifizierung des Authentifizierungsservers gegenüber dem Supplicant erforderlich. [vgl. [KPW02](#); [FB08](#)] Die Authentifizierung des Clients ist möglich, wird aber meistens ausgelassen, um den Konfigurationsaufwand zu reduzieren.

### 3.3. RADIUS-Accounting

Für die Implementierung von 802.1X ist es erforderlich, die AAA-Funktionen Authentifizierung und Autorisierung zu nutzen. In [Abschnitt 3.2](#) wurde beschrieben, wie diese Funktionen für die Pool-PCs und die Accounts des Abteilungspersonals genutzt werden. Die Nutzung der AAA-Funktion *Accounting* ist optional. Mit RADIUS-Accounting können durchaus nützliche Informationen für Statistiken und Monitoring gewonnen werden, allerdings würde dadurch auch das Konzept deutlich komplizierter werden. Ein Datenbankserver müsste zur Speicherung der Accountingdaten installiert werden. Des Weiteren

müssten auch datenschutzrechtliche Fragen geklärt werden, da nach der Datenschutz-Grundverordnung IP-Adressen personenbezogene Daten sind. [vgl. [Eur16](#)] Aus diesen Gründen wird auf RADIUS-Accounting verzichtet.

### 3.4. Umgang mit Geräten ohne 802.1X-Support

Obwohl 802.1X ein weit verbreiteter Standard ist und von allen relevanten Desktop- und Mobil-Betriebssystemen implementiert wird, gibt es trotzdem Geräte die 802.1X nicht unterstützen. Dies betrifft vor allem Netzwerkdrucker, aber auch manche [NAS](#)-Systeme und Smart-Devices. Solche Geräte können nicht ohne Weiteres an einem Switchport betrieben werden, an dem die 802.1X-Authentifizierung erzwungen wird. In diesem Kapitel werden zwei Lösungsansätze für dieses Problem verglichen.

#### 3.4.1. MAC Authentication Bypass

Die erste Möglichkeit dieses Problem zu lösen ist *MAC Authentication Bypass*, im Folgenden *MAB* genannt. MAB ist eine von Cisco entwickelte, proprietäre Technologie, die es ermöglicht, Geräte, die 802.1X nicht unterstützen, anhand ihrer [MAC](#)-Adresse zu authentifizieren. Darüberhinaus ermöglicht MAB auch eine dynamische [VLAN](#)-Zuweisung auf Basis der [MAC](#)-Adresse. [vgl. [Cis11](#)]

Wenn ein Gerät ohne Unterstützung für 802.1X an den Authenticator angeschlossen wird, sendet dieser ein RADIUS-Paket, welches die [MAC](#)-Adresse des Supplicants enthält, an den Authentifizierungsserver. Der Authentifizierungsserver greift auf einen Datenspeicher zu, auf dem eine Weißliste bekannter [MAC](#)-Adressen gepflegt wird. Wenn sich die [MAC](#)-Adresse des Supplicants auf dieser Liste befindet, sendet der Authentifizierungsserver ein [Access-Accept](#)-Paket an den Authenticator. Anderenfalls wird der Netzwerkzugriff per [Access-Reject](#)-Paket verweigert (vgl. [Abschnitt 2.3.3](#)).

#### 3.4.2. Nutzung der bestehenden DHCP-Weißliste

Die zweite Möglichkeit, das Problem zu lösen, ist die weitere Nutzung der bereits bestehenden DHCP-Weißliste. In diesem Fall wird an den Switchports, die von den Geräten ohne 802.1X-Unterstützung genutzt werden, der Authenticator gänzlich abgeschaltet, sodass keine 802.1X-Authentifizierung stattfindet. Angeschlossene Geräte versuchen sich über DHCP eine IP-Adresse zu beziehen. Nur wenn der DHCP-Server die [MAC](#)-Adresse des Clients kennt, wird er ihm eine IP-Adresse zuweisen. Anderenfalls bekommt der Client keine IP-Adresse und damit auch keinen Netzwerkzugriff.

### 3.4.3. Vergleich der Lösungsansätze

MAB bietet einige Vorteile gegenüber der Verwendung der DHCP-Weißliste. Scheitert die MAB-Authentifizierung, bleibt der Switchport blockiert. Bei der Verwendung der DHCP-Weißliste ist dies nicht der Fall. Das bedeutet, dass ein Client, der nicht auf der Weißliste steht, sich durch die Konfiguration einer festen IP-Adresse Zugang zum Netzwerk verschaffen kann. Des Weiteren ist statt der dynamischen VLAN-Konfiguration, die MAB ermöglicht, bei der Verwendung der DHCP-Weißliste, nur eine statische VLAN-Konfiguration möglich. Auf der anderen Seite hat die DHCP-Weißliste den Vorteil, dass diese bereits implementiert ist und gut funktioniert, wodurch kein zusätzlicher Konfigurationsaufwand anfällt. Schlussendlich haben beide Ansätze das gleiche Problem: Die Sicherheit basiert auf MAC-Adressen, deren Fälschung trivial ist (s. Abschnitt 3.1) Diese Tatsache macht die Sicherheitsvorteile von MAB deutlich weniger interessant. Unter Anbetracht des hohen Konfigurationsaufwands und der vernachlässigbaren Sicherheitsvorteile von MAB, ist die Weiterverwendung der bestehenden DHCP-Weißliste die bessere Lösung für die Abteilung Informatik.

## 3.5. Hochverfügbarkeit des RADIUS-Dienstes

Die Nutzung von 802.1X bringt eine Herausforderung mit sich: Der Netzwerkzugriff ist vollständig abhängig von der Verfügbarkeit des RADIUS-Dienstes. Jeder Switchport, auf dem die 802.1X-Authentifizierung erzwungen wird, bleibt gesperrt, wenn der Authentifizierungsserver nicht antwortet. Der Authentifizierungsserver ist also ein *Single Point of Failure* im Netzwerk – eine einzelne Systemkomponente, deren Versagen den Ausfall des gesamten Systems nach sich zieht. Um dies zu verhindern, muss ein zweiter Authentifizierungsserver betrieben werden, der die gleiche Konfiguration hat. Es muss gewährleistet sein, dass der Ausfall des ersten Authentifizierungsservers erkannt wird und der zweite Authentifizierungsserver, ohne manuelle Konfigurationsänderung, den Dienst übernimmt. Hierfür gibt es mehrere Lösungsansätze, von denen zwei in den kommenden Abschnitten erläutert und verglichen werden.

### 3.5.1. Aktiv-Passiv-Cluster mit Floating IP

Eine Möglichkeit Hochverfügbarkeit zu implementieren, ist die Nutzung einer sogenannten **Floating IP** (deutsch: *schwebende IP*). Diese Floating IP ist eine von zwei Servern „geteilte“, virtuelle IP-Adresse, zu der auch eine virtuelle MAC-Adresse gehört. Sie ist nur auf einem der beiden Server aktiv, kann dafür aber auf den anderen Server verschoben werden (vgl. Abb. 6). Bei diesem Ansatz wird auf dem Authenticator die Floating-IP als einzig verfügbarer Authentifizierungsserver eingetragen, wodurch dieser immer mit dem aktiven Server kommuniziert. Wenn nun der RADIUS-Dienst auf dem „Auth-Server-1“ ausfällt, wird die Floating IP sofort auf „Auth-Server-2“ aktiv geschaltet. Aus der Sicht des Authenticators ändert sich nichts, da dieser nach wie vor mit der

Floating IP kommuniziert. Dieser Prozess wird durch das *Virtual Router Redundancy Protocol* (VRRP) definiert. Eine frei verfügbare Implementierung dieses Protokolls für GNU/Linux ist *keepalived*, welche es erlaubt, den Zustand eines Dienstes mit Hilfe von Skripten zu überprüfen. Anhand des Rückgabewertes des Skripts wird entschieden, ob ein Wechsel des aktiven und passiven Servers nötig ist.

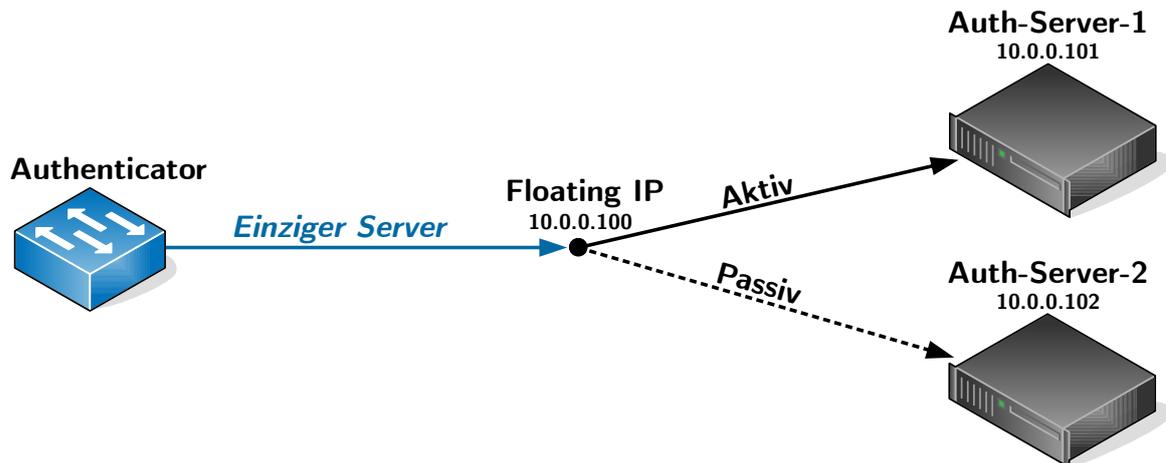


Abbildung 6: Hochverfügbarer RADIUS-Dienst durch Nutzung eines Aktiv-Passiv-Clusters auf Basis einer Floating IP

### 3.5.2. Authentifizierungsserver-Gruppe auf dem Authenticator

Cisco-Switche bieten die Möglichkeit, sogenannte **AAA Server Groups** anzulegen. In diesen Gruppen können mehrere Authentifizierungsserver eingetragen werden. Der Authentifizierungsserver an der ersten Position in der Gruppe ist der *primäre Server*, während der Authentifizierungsserver an Position zwei der *sekundäre Server* ist (vgl. Abb. 7). Es ist natürlich auch möglich, mehr als zwei Server in die Gruppe einzutragen, jedoch reichen zwei Server aus, um den Ausfall eines Servers abzufangen. Dieser Ablauf wird im RADIUS-Konfigurationshandbuch für Cisco IOS 15 beschrieben: Wenn sich ein Supplicant authentifizieren möchte, wird das RADIUS-Paket zuerst an den primären Server gesendet. Erhält der Authenticator innerhalb eines vorgegebenen Zeitfensters keine Antwort, so wird der Server als *tot* markiert und der sogenannte *Deadtimer* wird gestartet. Während der Timer läuft, prüft der Authenticator in regelmäßigen Abständen, ob der Server wieder erreichbar ist. Da der Authenticator vom primären Server keine Antwort erhalten hat, wird das Paket an den nächsten Server der Gruppe gesendet und von diesem bearbeitet. Alle nachfolgenden Pakete werden an den ersten Server der Gruppe gesendet, der nicht als *tot* markiert ist – in diesem Fall der sekundäre Server. Die Markierung eines Servers als *tot* wird erst dann wieder entfernt, wenn der Deadtimer abgelaufen ist oder der Server schon vorher wieder erreichbar ist. [vgl. Cis14]

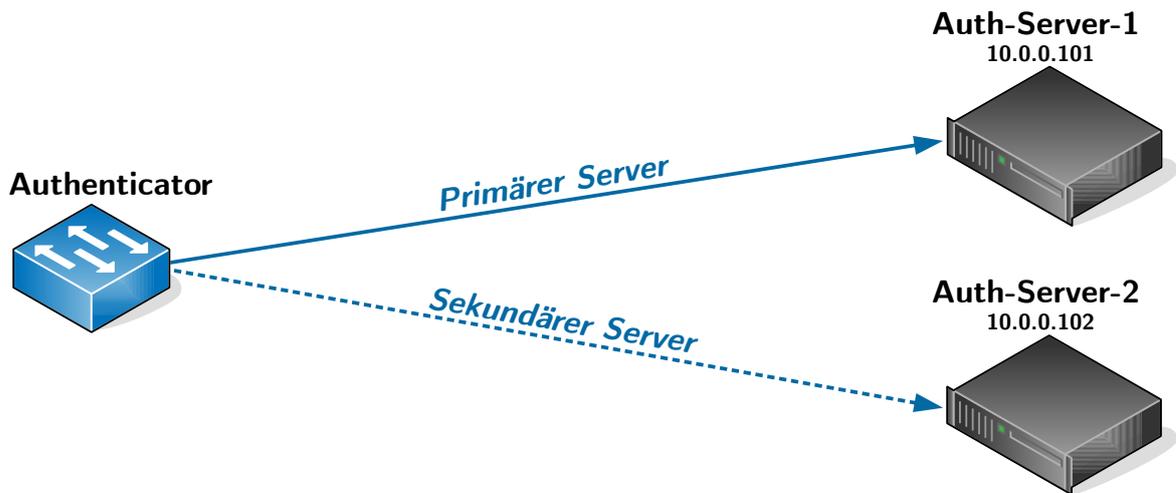


Abbildung 7: Hochverfügbarer RADIUS-Dienst mit Authentifizierungsserver-Gruppe

Die Konfiguration einer Authentifizierungsserver-Gruppe auf dem Authenticator löst das Verfügbarkeitsproblem, genau wie der Floating-IP-Ansatz, mit einem aktiven Server, an den alle Anfragen geschickt werden, solange es keine Probleme gibt, und einem passiven Server, der im Fehlerfall übernehmen kann. Da aber durch den erforderlichen Betrieb eines zweiten Authentifizierungsservers ohnehin Kosten für Hardware und Strom anfallen, ist es wünschenswert die Last auf beide Server gleichmäßig zu verteilen. Um dies umzusetzen, kann innerhalb einer Authentifizierungsserver-Gruppe ein **Lastenausgleich** (engl.: *Load Balancing*) konfiguriert werden. Der Ablauf des Lastenausgleich wird im RADIUS-Konfigurationshandbuch für Cisco IOS 15 beschrieben. Dabei werden die eingehenden Anfragen in Stapeln (engl.: *Batches*) abgearbeitet, deren Größe konfigurierbar ist und standardmäßig 25 beträgt. Kleine Stapel erhöhen die CPU-Auslastung des Authenticators, während große Stapel eher das Netzwerk belasten. Die Auswahl eines Servers ist ein vierschrüttiger Prozess: [Cis14]

1. Der Authenticator erhält die erste Anfrage eines neuen Stapels.
2. Die Liste der ausstehenden Anfragen wird für jeden Server überprüft.
3. Der Server mit der niedrigsten Anzahl ausstehender Anfragen wird identifiziert.
4. Dieser Server wird für die Bearbeitung des nächsten Stapels ausgewählt.

### 3.5.3. Vergleich der Lösungsansätze

Die Nutzung der Authentifizierungsserver-Gruppe auf dem Authenticator hat den klaren Vorteil, dass durch den Lastenausgleich zwei aktive Authentifizierungsserver verwendet werden können. Allerdings ist im Netzwerk der Abteilung Informatik nicht mit einer starken Auslastung der Authentifizierungsserver zu rechnen, sodass der Lastenausgleich nicht zwingend erforderlich, sondern eher ein willkommener Bonus ist. Für

den Floating-IP-Ansatz spricht, dass dieses Konzept bereits bei einigen Diensten in der Abteilung Informatik im Einsatz ist und gut funktioniert. Allerdings ist es auch der komplexere der beiden Ansätze, da die Installation eines zusätzlichen Dienstes auf beiden Authentifizierungsservern erforderlich ist. Darüberhinaus ist es nicht trivial, in dem von `keepalived` genutzten Skript zu überprüfen, ob die Authentifizierung noch funktioniert. Oft wird in diesen Skripten nur geprüft, ob der zu überwachende Dienst noch läuft. Dies ist aber oft nicht ausreichend, da es durchaus zu Fehlern kommen kann, obwohl die überwachten Dienste noch laufen. Dies macht den Floating-IP-Ansatz etwas fehleranfälliger, als die Nutzung der Authentifizierungsserver-Gruppe. Aus diesem Grund ist die Nutzung der Authentifizierungsserver-Gruppe die besser geeignete Lösung für die Abteilung Informatik.

## 3.6. Softwareauswahl

In diesem Kapitel wird die passende Software für die Umsetzung des Konzepts ausgewählt. Dies betrifft nur den Supplicant für die Pool-PCs sowie die RADIUS-Server-Software. Als Authenticator werden die vorhandenen Cisco-Switche genutzt.

### 3.6.1. Supplicant

Der Supplicant, der auf den Pool-PCs genutzt werden soll, muss die aktuelle Betriebssystemversion der PCs, sowie die EAP-Methode EAP-TLS unterstützen. Zur Zeit wird Ubuntu 16.04 als Betriebssystem der Pool-PCs eingesetzt. In diesem Kapitel werden drei Supplicants verglichen. Der **Xsupplicant** ist ein vom Open1X-Projekt entwickelter, freier Supplicant für Windows und GNU/Linux. Der Webseite des Projekts kann entnommen werden, dass EAP-TLS neben PEAP und EAP-FAST und einigen anderen EAP-Methoden unterstützt wird. [vgl. [Ope10](#)] Eine grafische Oberfläche zur Konfiguration gibt es nicht, stattdessen werden Konfigurationsdateien verwendet. Xsupplicant ist nicht in den offiziellen Paketquellen von Ubuntu 16.04 enthalten und muss daher selbst kompiliert werden. Die Aktuelle stabile Version von Xsupplicant ist 2.2.0 und wurde im Januar 2010 veröffentlicht.

Eine Alternative zum Xsupplicant ist der **wpa\_supplicant**. Dies ist ein freier Supplicant für Windows, GNU/Linux und andere Unix-Systeme, wie zum Beispiel FreeBSD und Solaris. Der offiziellen Webseite kann entnommen werden, dass `wpa_supplicant` alle relevanten EAP-Methoden unterstützt, unter anderem auch EAP-TLS. [vgl. [Mal13](#)] Das Programm ist für die Nutzung mit kabellosen Netzwerken optimiert, kann aber problemlos auch für kabelgebundene Netzwerke verwendet werden. Für die Implementierung kryptografischer Funktionen wird standardmäßig die weit verbreitete Bibliothek *OpenSSL* verwendet. Der `wpa_supplicant` kann über Konfigurationsdateien, ein Kommandozeilenprogramm oder eine grafische Oberfläche konfiguriert werden. Darüber hinaus wird der `wpa_supplicant` vom *NetworkManager* genutzt – dem am weitesten verbreiteten

Netzwerkverwaltungsprogramm für GNU/Linux-Desktops. Der NetworkManager und `wpa_supplicant` sind für Ubuntu 16.04 paketiert und standardmäßig vorinstalliert. Die aktuelle Version ist 2.7 und wurde im Dezember 2018 veröffentlicht. Ubuntu 16.04 wird mit der Version 2.4 vom März 2015 ausgeliefert.

Eine weitere Alternative ist **IWD**. Dieser freie Supplicant wird von Intel als Ersatz für den komplexen `wpa_supplicant` entwickelt. Aus diesem Grund ist IWD nur für GNU/Linux verfügbar. EAP-TLS wird neben einigen anderen EAP-Methoden unterstützt, was der Dokumentation entnommen werden kann. [vgl. Zab18] Auch IWD ist für die Nutzung mit kabellosen Netzwerken optimiert, kann aber auch mit kabelgebundenen Netzwerken verwendet werden. Das Programm wird über Konfigurationsdateien und ein Kommandozeilenprogramm verwaltet und kann ebenfalls vom NetworkManager genutzt werden. Zur weiteren Reduzierung der Komplexität, werden statt OpenSSL die Kryptografiefunktionen des Linux-Kernels genutzt. Allerdings sind die nötigen Funktionen erst in der Kernel-Version 4.20 enthalten, die vermutlich Ende Dezember 2018 veröffentlicht wird. Ältere Kernel-Versionen müssen von Hand gepatcht und selbst kompiliert werden. Für Ubuntu 16.04 ist IWD nicht paketiert. Es ist also erforderlich das Programm selbst zu kompilieren.

Xsupplicant ist aus mehreren Gründen schlecht geeignet. Die fehlende Integration in den NetworkManager macht die Administration im Vergleich zu IWD und `wpa_supplicant` deutlich komplizierter. Darüber hinaus könnte Xsupplicant ein Sicherheitsrisiko darstellen, da es seit knapp neun Jahren nicht mehr gepflegt wird. Da das Programm nicht für Ubuntu 16.04 paketiert wird, ist der Installationsprozess komplexer und fehleranfälliger, als bei dem `wpa_supplicant`. Auch für IWD ist der komplizierte Installationsprozess ein Nachteil. Dies ist vor allem dem noch recht frühen Entwicklungsstadium von IWD geschuldet. Der deutlich weniger komplexe Aufbau gegenüber `wpa_supplicant` ist zwar ein klarer Vorteil, trotzdem ist IWD für den produktiven Einsatz noch nicht ausgereift genug. Da der `wpa_supplicant` unter Ubuntu 16.04 bereits vorinstalliert ist, EAP-TLS unterstützt und sich gut in den Ubuntu-Desktop integriert, ist der `wpa_supplicant` die beste der drei Optionen.

#### 3.6.2. RADIUS-Server-Software

In diesem Kapitel wird die Wahl des passenden RADIUS-Servers getroffen. Es existieren viele verschiedene RADIUS-Server-Implementierungen. Daher muss die Auswahl der zu vergleichenden Server beschränkt werden. Ausgeschlossen sind Cloud-basierte Lösungen, sowie RADIUS-Server, die stark an eine bestimmte Hardware gebunden sind, sogenannte *Appliances*. Im Folgenden werden fünf der populärsten RADIUS-Server miteinander verglichen: Aradial, ClearBox, FreeRADIUS, GNU Radius und TekRADIUS. In der folgenden Tabelle werden die Server anhand der unterstützten Betriebssysteme, EAP-Methoden und Datenspeicher sowie deren Preis, Administrationsinterface, Lizenz und Versionsstand gegenübergestellt.

Tabelle 5: RADIUS-Server im Vergleich

	Aradial	ClearBox <sup>1</sup>	FreeRADIUS	GNU Radius	TekRADIUS
<b>Preis</b>	n. a.	722,03 €	Kostenlos	Kostenlos	ca. 510,- €
<b>Unterstützte Betriebssysteme</b>	<ul style="list-style-type: none"> <li>• GNU/Linux<sup>2</sup></li> <li>• Windows<sup>2</sup></li> <li>• Solaris</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Windows:</b> <ul style="list-style-type: none"> <li>– XP – 8</li> </ul> </li> <li>• <b>Win. Server:</b> <ul style="list-style-type: none"> <li>– 2000 – 2008</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>GNU/Linux:</b> <ul style="list-style-type: none"> <li>– CentOS</li> <li>– RHEL</li> <li>– Debian</li> <li>– Ubuntu</li> </ul> </li> <li>• FreeBSD</li> <li>• macOS</li> <li>• Solaris</li> </ul>	GNU/Linux <sup>2</sup>	<ul style="list-style-type: none"> <li>• <b>Windows</b> <ul style="list-style-type: none"> <li>– ab XP</li> </ul> </li> </ul>
<b>Unterstützte EAP-Methoden</b>	<ul style="list-style-type: none"> <li>• <b>EAP:</b> <ul style="list-style-type: none"> <li>– TLS</li> <li>– MD5</li> <li>– TTLS<sup>3</sup></li> <li>– SIM</li> <li>– AKA</li> </ul> </li> <li>• PEAP<sup>3</sup></li> </ul>	<ul style="list-style-type: none"> <li>• <b>EAP:</b> <ul style="list-style-type: none"> <li>– TLS</li> </ul> </li> <li>• <b>PEAP:</b> <ul style="list-style-type: none"> <li>– TLS</li> <li>– MSCHAPv2</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><b>EAP:</b> <ul style="list-style-type: none"> <li>– TLS</li> <li>– MSCHAPv2</li> <li>– GTC</li> <li>– MD5</li> <li>– SIM</li> <li>– TTLS<sup>4</sup></li> <li>– IKEv2</li> </ul> </li> <li>• <b>PEAP:</b> <ul style="list-style-type: none"> <li>– MSCHAPv2</li> </ul> </li> </ul>	Keine	<ul style="list-style-type: none"> <li><b>EAP:</b> <ul style="list-style-type: none"> <li>– TLS</li> <li>– TTLS</li> <li>– MSCHAPv2</li> <li>– MD5</li> <li>– SIM</li> </ul> </li> <li>• <b>PEAP:</b> <ul style="list-style-type: none"> <li>– MSCHAPv2</li> <li>– MD5</li> </ul> </li> </ul>
<b>Unterstützte Datenspeicher</b>	<ul style="list-style-type: none"> <li>• <b>VZ-Dienste:</b> <ul style="list-style-type: none"> <li>– AD</li> <li>– LDAP<sup>5</sup></li> </ul> </li> <li>• <b>DB-Systeme:</b> <ul style="list-style-type: none"> <li>– MS SQL</li> <li>– MySQL</li> <li>– Oracle DB</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>VZ-Dienste:</b> <ul style="list-style-type: none"> <li>– AD</li> <li>– LDAP<sup>5</sup></li> </ul> </li> <li>• <b>DB-Systeme:</b> <ul style="list-style-type: none"> <li>– SQL<sup>6</sup></li> <li>• Lokale Accounts</li> <li>• Klartextdatei</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>VZ-Dienste:</b> <ul style="list-style-type: none"> <li>– AD</li> <li>– LDAP<sup>5</sup></li> </ul> </li> <li>• <b>DB-Systeme:</b> <ul style="list-style-type: none"> <li>– PostgreSQL</li> <li>– SQLite</li> <li>– Redis</li> <li>– MySQL</li> <li>– SQL<sup>6</sup></li> <li>• Lokale Accounts</li> <li>• Klartextdatei</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>DB-Systeme:</b> <ul style="list-style-type: none"> <li>– PostgreSQL</li> <li>– MySQL</li> <li>– SQL<sup>6</sup></li> <li>• Lokale Accounts</li> <li>• Klartextdatei</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• <b>VZ-Dienste:</b> <ul style="list-style-type: none"> <li>– AD</li> </ul> </li> <li>• <b>DB-Systeme:</b> <ul style="list-style-type: none"> <li>– MS SQL</li> <li>– SQLite</li> </ul> </li> </ul>
<b>Interface</b>	Webbasiert	GUI	CLI	CLI	<ul style="list-style-type: none"> <li>• GUI</li> <li>• CLI</li> </ul>
<b>Lizenz</b>	Proprietär	Proprietär	GPLv2	GPLv3	Proprietär
<b>Aktuelle Version</b>	n. a.	6.6.0 (2018-11)	3.0.17 (2018-04)	1.6.1 (2008-12)	5.4.4 (2018-08)

<sup>1</sup>ClearBox Enterprise RADIUS Server<sup>2</sup>Genauere Informationen über kompatible Betriebssystemversionen und -distributionen werden nicht bereitgestellt.<sup>3</sup>Informationen über kompatible innere Authentifizierungen werden nicht bereitgestellt.<sup>4</sup>Innere Authentifizierungen: PAP, CHAP, MSCHAP, MSCHAPv2, EAP-MSCHAPv2, EAP-MD5, EAP-GTC, EAP-TLS<sup>5</sup>Jeder LDAP-kompatible Verzeichnisdienst kann über einen generischen Treiber genutzt werden.<sup>6</sup>Jedes SQL-konforme Datenbanksystem kann über einen generischen Treiber genutzt werden.

Eine für die Umsetzung des Konzepts geeignete RADIUS-Server-Software muss die EAP-Methoden EAP-TLS und PEAP-MSCHAPv2 unterstützen. Darüberhinaus muss ein Active-Directory als Datenspeicher unterstützt werden. Als Betriebssystem für die Authentifizierungsserver soll die GNU/Linux-Distribution Debian verwendet werden, da dieses Betriebssystem bereits auf vielen Servern der Abteilung Informatik zum Einsatz kommt. Eine Administration über die Kommandozeile (CLI) wird anderen Administrationsinterfaces, wie einer GUI oder einer Weboberfläche, vorgezogen, da keine zusätzliche Software installiert werden muss, die die Sicherheit des Servers beeinträchtigen kann, wie zum Beispiel ein Webserver oder ein Displayserver.

Die in der [Tabelle 5](#) dargestellten Informationen zu dem **Aradial** RADIUS Server sind dem Produktdatenblatt [[Ara18](#)] entnommen, das der Hersteller „Aradial Technologies“ bereitstellt. Allerdings finden sich in verschiedenen offiziellen Dokumenten von Aradial Technologies unterschiedliche, teilweise widersprüchliche Informationen. Aradial unterstützt GNU/Linux, allerdings gibt der Hersteller nicht an, welche Distributionen unterstützt werden. Die EAP Methoden EAP-TLS und PEAP werden unterstützt, jedoch fehlen Informationen über unterstützte innere Authentifizierungen für PEAP. Da Active Directory als Datenspeicher unterstützt wird, ist es naheliegend, dass auch PEAP-MSCHAPv2 als innerer Tunnel genutzt werden kann. Aradial wird über eine Weboberfläche administriert. [vgl. [Ara18](#)] Informationen über den Preis einer Lizenz und die aktuelle Version sind nicht öffentlich. Aradial erfüllt die technischen Anforderungen für das Konzept, allerdings hinterlassen die teilweise widersprüchlichen Aussagen in verschiedenen Dokumenten und die fehlenden Informationen zum Preis einen eher schlechten Eindruck.

Der Hersteller „XPerience Technologies“ stellt die in der [Tabelle 5](#) dargestellten Informationen zu dem **ClearBox Enterprise RADIUS Server** auf der Webseite [[XPe18a](#)] zur Verfügung. Die Software unterstützt die für die Umsetzung des Konzepts notwendigen Datenspeicher (AD) und EAP-Methoden (EAP-TLS, PEAP-MSCHAPv2). [vgl. [XPe18a](#)] Allerdings wird nur Windows als Betriebssystem unterstützt und der Server kann nur über eine GUI administriert werden. [vgl. [XPe18c](#); [XPe18d](#)] Darüberhinaus fallen für eine Lizenz Kosten in Höhe von 722,03 € an. Bei einem Erwerb von zwei oder mehr Lizenzen sinkt der Preis pro Lizenz auf 517,10 €. Da zwei Authentifizierungsserver betrieben werden müssen (s. [Abschnitt 3.5](#)), betragen die Lizenzkosten 1.034,20 €. [vgl. [XPe18b](#), Preise in Dollar exkl. MwSt] Aus diesen Gründen kommt der ClearBox Enterprise RADIUS Server für die Umsetzung des Konzepts nicht in Frage.

Die Daten zu **FreeRADIUS** wurden dem Handbuch der Software [[Net14b](#); [Net14a](#)] entnommen. FreeRADIUS ist freie, kostenlose Software und unterstützt neben GNU/Linux ein breites Spektrum an unixoiden Betriebssystemen. Die offiziellen Debian-Repositorys enthalten alle zur Installation von FreeRADIUS notwendigen Pakete. Alle relevanten EAP-Methoden, darunter auch EAP-TLS und PEAP-MSCHAPv2, können mit FreeRADIUS genutzt werden. [vgl. [Net14a](#)] Des Weiteren wird auch Active Directory als Datenspeicher unterstützt. Das Programm wird über die Kommandozeile gesteuert. [vgl. [Net14b](#)] FreeRADIUS wird aktiv entwickelt. Die aktuelle Version 3.0.17 erschien im

April 2018. Darüberhinaus pflegen die Entwickler einen sehr transparenten Umgang mit Sicherheitslücken [vgl. [Fre18a](#)] FreeRADIUS erfüllt alle der genannten Anforderungen und ist daher für die Umsetzung des Konzepts geeignet.

Die Informationen zu **GNU Radius** sind im Handbuch der Software [[Poz08](#)] zu finden. Als Teil des GNU-Projekts ist GNU Radius freie, kostenlose Software. Die Betriebssystem-Unterstützung beschränkt sich auf GNU/Linux, was für die Umsetzung des Konzepts ausreichend ist. Der Server wird über die Kommandozeile gesteuert und mit Konfigurationsdateien verwaltet. Allerdings wird die Software ausschließlich als Quellcode bereitgestellt, was den Installationsvorgang erschwert. Zusätzlich werden Active Directory als Datenspeicher und das gesamte EAP-Framework nicht unterstützt. Eine Authentifizierung ist nur über andere Protokolle wie [PAP](#) und [CHAP](#) möglich. [vgl. [Poz08](#)] Da die aktuelle Version im Dezember 2008 veröffentlicht wurde und die Software seitdem nur sporadisch weiterentwickelt wurde, ist auch in Zukunft nicht mit EAP-Unterstützung zu rechnen, was GNU Radius für die Implementierung von 802.1X untauglich macht.

Die in der [Tabelle 5](#) dargestellten Informationen zu **TekRADIUS** können dem Datenblatt der aktuellen Version 5.4 [[Kap18b](#)] des TekRADIUS-Servers entnommen werden. Der RADIUS-Server unterstützt die benötigten EAP-Methoden EAP-TLS und PEAP-MSCHAPv2. Active Directory als Datenspeicher wird ebenfalls unterstützt. Die Administration kann sowohl über eine GUI als auch über die Kommandozeile durchgeführt werden. Allerdings ist TekRADIUS genau wie der ClearBox Enterprise RADIUS Server nur für Windows erhältlich. [vgl. [Kap18b](#)] Darüberhinaus kostet eine Lizenz 510 €. Informationen über Rabatte beim Erwerb mehrerer Lizenzen werden nicht bereitgestellt. [vgl. [Kap18a](#)] Aufgrund des Preises und der Betriebssystem-Unterstützung ist TekRADIUS für die Umsetzung des Konzepts nicht geeignet.

Von den fünf RADIUS-Servern erfüllen nur Aradial und FreeRADIUS alle technischen Anforderungen. Von diesen beiden Optionen ist FreeRADIUS besser geeignet, da dieser Server mehr EAP-Methoden, Datenspeicher und Betriebssysteme unterstützt, keinen Webserver benötigt und darüberhinaus freie und kostenlose Software ist.

## 4. Realisierung des Konzepts

In den folgenden Abschnitten wird die Realisierbarkeit des Konzepts durch eine Implementierung in einer Testumgebung geprüft. Zunächst wird in [Abschnitt 4.1](#) die Funktionsweise von FreeRADIUS erklärt, bevor in den [Abschnitten 4.2 bis 4.4](#) die Konfiguration der Authentifizierungsserver, Switches und Supplicants erläutert wird.

### 4.1. FreeRADIUS

In diesem Kapitel werden der Aufbau der FreeRADIUS-Software, sowie die wichtigsten Konzepte und Konfigurationsdateien erläutert. Abschließend wird der Ablauf der Autorisierung und Authentifizierung schrittweise erklärt.

#### 4.1.1. Modularer Aufbau

Eines der wichtigsten Konzepte von FreeRADIUS ist der modulare Aufbau der Software. Dies bedeutet, dass Funktionen, wie zum Beispiel LDAP-, PAP- oder CHAP-Unterstützung in Module ausgelagert sind und erst genutzt werden können, wenn die entsprechenden Module aktiviert sind. Im FreeRADIUS-Handbuch wird erläutert, dass deaktivierte Module keinen Einfluss auf die Performance, den Speicherverbrauch oder die Sicherheit des FreeRADIUS-Servers haben. [vgl. [Net14b](#)] Dadurch ist es möglich die Komplexität einer FreeRADIUS-Konfiguration auf das Nötigste zu reduzieren und somit mehr Ressourcen für die benötigten Funktionen bereitzustellen und die Fehlersuche zu vereinfachen.

#### 4.1.2. Virtuelle Server

Die Richtlinien zur Autorisierung und Authentifizierung, sowie zum Accounting, werden in sogenannten *virtuellen Servern* festgelegt. In diesen Richtlinien wird definiert, welche Protokolle zur Autorisierung und Authentifizierung genutzt werden können, indem die Module aufgerufen werden, die diese Protokolle implementieren. Darüberhinaus können die in den AVPs eines RADIUS-Pakets enthaltenen Daten abgefragt werden, was eine sehr feine Kontrolle über die Autorisierung und Authentifizierung ermöglicht. Ein FreeRADIUS-Server kann mehrere virtuelle Server gleichzeitig ausführen. Virtuelle Server erhöhen die Performance und Flexibilität, da große Richtlinien, die viele Abfragen für spezielle Fälle enthalten, durch mehrere spezialisierte virtuelle Server ersetzt werden können. [vgl. [Net18a](#)] So könnte zum Beispiel ein FreeRADIUS-Server, der Dienste für mehrere Domänen anbietet, mit einem virtuellen Server pro Domäne konfiguriert werden. Eintreffende Anfragen können dann anhand des Domänennamens an den zuständigen virtuellen Server weitergereicht werden, anstatt den vollständigen Richtlinienatz für alle Domänen zu durchlaufen.

### 4.1.3. Verzeichnisstruktur und Konfigurationsdateien

Die Konfigurationsdateien von FreeRADIUS befinden sich je nach der Quelle, aus der die Software bezogen wird, an einem anderen Ort im Verzeichnisbaum. In der Dokumentation wird für diesen Ort der Name `raddb` verwendet, der auch in dieser Arbeit genutzt wird. Die Konfigurationsdateien werden in Kategorien eingeteilt und in entsprechenden Unterverzeichnissen abgelegt:

- `raddb/mods-available/` – In diesem Ordner werden Konfigurationsdateien von FreeRADIUS-Modulen gespeichert.
- `raddb/sites-available/` – Hier werden Konfigurationsdateien von virtuellen Servern gespeichert.
- `raddb/certs/` – Hier werden [TLS](#)-Zertifikate gespeichert, sowie Konfigurationsdateien und Skripte, um diese zu erstellen.

Einige Konfigurationsdateien, die sich nicht in eine dieser Kategorien einteilen lassen, befinden sich stattdessen direkt unterhalb von `raddb/`. FreeRADIUS umfasst über 180 verschiedene Konfigurationsdateien, von denen die folgenden am wichtigsten sind:

- `raddb/radiusd.conf` – Diese Datei definiert die Konfigurationsparameter des FreeRADIUS-Servers und enthält Referenzen zu allen anderen Konfigurationsdateien.
- `raddb/clients.conf` – In dieser Datei werden alle Informationen zur Konfiguration der Authentikatoren gespeichert, wie zum Beispiel IP-Adressen und gemeinsame Passwörter.
- `raddb/mods-available/eap` – Dies ist die Konfigurationsdatei des Moduls, das die EAP-Authentifizierung implementiert.
- `raddb/sites-enabled/default` – Hierbei handelt es sich um die Konfigurationsdatei des standardmäßigen virtuellen Servers.
- `raddb/sites-enabled/inner-tunnel` – Dies ist die Konfigurationsdatei des virtuellen Servers, der die Authentifizierungen des inneren Tunnels von PEAP und EAP-TTLS verarbeitet.

Alle Konfigurationsdateien haben die gleiche Syntax, die in der FreeRADIUS-Dokumentation [[Net18b](#)] beschrieben wird. Konfigurationswerte werden mit dem Gleichheitszeichen gesetzt, Kommentare werden mit einer Raute eingefügt. [vgl. [Net18b](#)]

```
# Dies ist ein Kommentar
variable = wert # Kommentare koennen auch am Ende einer Zeile stehen
zweite_variable = "Strings mit Leerzeichen stehen in Anfuhrungsstrichen"
dritte_variable = "Sehr lange Strings koennen mit einem Backslash in\
mehrere Zeilen geteilt werden"
```

Zusammengehörige Konfigurationswerte können mit geschwungenen Klammern in Gruppen eingeteilt werden. Diese Gruppen werden *Sections* genannt. [vgl. [Net18b](#)]

```
meine_section {
    anzahl_variablen = 3
    typ = beispiel
    eingerueckt = yes
}
```

Darüberhinaus können in Konfigurationsdateien weitere Dateien mit dem Schlüsselwort `$INCLUDE` eingebunden werden. [vgl. [Net18b](#)]

```
$INCLUDE other.conf # Einzelne Datei einbinden
$INCLUDE folder/   # Alle Dateien eines Unterverzeichnisses einbinden
```

### 4.1.4. Die zentrale Konfigurationsdatei

Die Datei `raddb/radiusd.conf` enthält alle grundlegenden Konfigurationswerte, die nicht in Module ausgelagert werden können. Einige Beispiele der Konfigurationswerte sind:

- `name` – Der Name des Prozesses des FreeRADIUS-Servers
- `raddbdir` – Der Pfad des Konfigurationsordners
- `max_requests` – Die maximale Anzahl unbearbeiteter Anfragen, bevor der Server weitere Anfragen ablehnt
- `logdir` – Der Pfad des Verzeichnisses, in dem Logdateien abgelegt werden
- `certdir` – Der Pfad des Verzeichnisses, in dem die [TLS](#)-Zertifikate des Servers abgelegt sind

Da beim Start des FreeRADIUS-Servers nur diese Datei eingelesen wird, müssen andere Konfigurationsdateien über `$INCLUDE` eingebunden werden. [vgl. [Net18c](#)] Neben einzelnen Konfigurationsdateien, wie zum Beispiel `clients.conf`, werden die Verzeichnisse `raddb/mods-enabled/` und `raddb/sites-enabled/` eingebunden. Diese Verzeichnisse enthalten Links auf die Konfigurationsdateien der Module in `raddb/mods-available/` beziehungsweise der virtuellen Server in `raddb/sites-available/`. Dadurch wird die Verwaltung der Module und virtuellen Server vereinfacht, da diese durch das Setzen oder Entfernen des Links aktiviert beziehungsweise deaktiviert werden können, ohne die eigentliche Konfigurationsdatei zu bearbeiten.

### 4.1.5. Konfiguration von virtuellen Servern

Jeder virtueller Server wird innerhalb einer `server`-Section konfiguriert, die darüberhinaus noch eine oder mehrere `listen`-Sections enthält. In den `listen`-Sections wird unter anderem definiert, welche IP-Adresse und welchen Port der virtuelle Server nutzen soll. Der Konfigurationswert `type` gibt an, ob der Server zum Accounting (`acct`) oder zur

Authentifizierung (**auth**) genutzt wird. Mehrere **listen**-Sections können für verschiedene IP-Adressen oder Typen definiert werden. [vgl. [Net18d](#)] Das folgende Beispiel zeigt die ersten Zeilen der Konfiguration eines virtuellen Servers mit dem Namen **example**.

```
server example {
  listen {
    ipaddr = 10.0.0.1
    port = 1812
    type = auth
  }
  [...]
}
```

Die Richtlinien zur Autorisierung und Authentifizierung werden in den Sections **authorize** und **authenticate** konfiguriert.

```
authorize {
  filter_username
  ldap
  mschap
  [...]
}
authenticate {
  Auth-Type MS-CHAP {
    mschap
  }
  [...]
}
```

Der Inhalt dieser Sections wird in der Dokumentation [[Net18d](#)] beschrieben. Die **authorize**-Section enthält die Namen von Modulen, die zur Autorisierung genutzt werden. Im obigen Beispiel sind dies die Module **ldap** und **mschap**. Des Weiteren ist es möglich, *Unlang-Funktionen* aufzurufen, wie zum Beispiel **filter\_username**. [vgl. [Net18d](#)] Unlang ist eine Sprache, in der Abfragen zu den in den RADIUS-AVPs enthaltenen Daten definiert werden können. Die **authorize**-Section darf keine Aufrufe von Unlang-Funktionen enthalten, sondern definiert nur die Module, die zur Authentifizierung zur Verfügung stehen. [vgl. [Net18d](#)]

In der Standardeinstellung nutzt FreeRADIUS die zwei virtuellen Server **default** und **inner-tunnel**. Der Server **default** ist der Standard-Server und bearbeitet zunächst alle eingehenden Anfragen. Der Server **inner-tunnel** bearbeitet die Anfragen in den inneren Tunneln von PEAP und EAP-TTLS und reagiert deshalb nur auf Anfragen, die von *Localhost* versendet wurden.

### 4.1.6. Ablauf von Autorisierung und Authentifizierung

In diesem Abschnitt wird anhand eines Beispiels beschrieben, wie ein virtueller Server die Autorisierung und Authentifizierung durchführt. In der **authorize**-Section des virtuellen Servers sind die Unlang-Funktion **filter\_username** sowie die Module **chap**, **mschap**, **eap**,

`files` und `pap` definiert. Bis auf das Modul `files` sind in der `authenticate`-Section die selben Module definiert. Der FreeRADIUS-Server nutzt die Klartextdatei `users.conf` als Datenspeicher, die einen Eintrag für den Benutzer `bob` mit dem Passwort `passwort123` enthält. Der Benutzer schickt ein `Access-Request`-Paket mit seinen Anmeldedaten an den RADIUS-Server. Der RADIUS-Server zeigt den Eingang des Pakets und die darin enthalten AVPs in den Zeilen eins bis sechs des Server-Logs an.

```
1 (0) Received Access-Request Id 217 from 127.0.0.1:56213 to 127.0.0.1:1812 length 75
2 (0)   User-Name = "bob"
3 (0)   User-Password = "passwort123"
4 (0)   NAS-IP-Address = 127.0.1.1
5 (0)   NAS-Port = 0
6 (0)   Message-Authenticator = 0xc920223b4050054b5ffa22236d21fece
```

Nach dem Erhalt des Pakets beginnt der virtuelle Server `default` die Anfrage zu bearbeiten, indem die `authorize`-Section aufgerufen wird (s. Zeilen 7–8).

```
7 (0) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default
8 (0)   authorize {
9 (0)     policy filter_username {
10 [...]
11 (0)     } # policy filter_username = noop
12 (0)     [chap] = noop
13 (0)     [mschap] = noop
14 [...]
15 (0) eap: No EAP-Message, not doing EAP
16 (0)     [eap] = noop
17 (0) files: users: Matched entry bob at line 1
18 (0)     [files] = ok
19 [...]
20 (0)     [pap] = updated
21 (0)   } # authorize = updated
```

Nun werden die Einträge der `authorize`-Section der Reihe nach ausgeführt, um den Benutzer `bob` zu autorisieren. Zuerst wird die Unlang-Funktion `filter_username` aufgerufen (s. Zeilen 9–11), die sicherstellt, dass Anfragen mit ungültigen `User-Name`-Werten abgelehnt werden. Da der Accountname `bob` gültig ist und die Funktion daher nichts getan hat, wird der Rückgabewert auf `noop` (*No Operation*) gesetzt (s. Zeile 11). Danach durchsuchen die Module `chap` (s. Zeile 12), `mschap` (s. Zeile 13) und `eap` (s. Zeilen 15–16) die AVPs des `Access-Request`-Pakets nach bestimmten Schlüsselwörtern, die in der Dokumentation erläutert werden. Das `mschap`-Modul sucht zum Beispiel nach AVPs, deren Attributnamen mit `MS-CHAP-` beginnen. [vgl. [Net18e](#)] Da die drei Module keine passenden AVPs gefunden haben, setzen sie den Rückgabewert auf `noop`. Danach wird das `files`-Modul aufgerufen, das die Unterstützung für lokale Dateien als Datenspeicher implementiert. Das Modul liest den Wert `bob` aus dem Attribut `User-Name` und durchsucht die Datei `users.conf` nach einem passenden Eintrag. Da ein passender Eintrag vorhanden ist, wird das dazugehörige Passwort in der Variable `Cleartext-Password` gespeichert. [vgl. [Net18f](#)] Das `files`-Modul war erfolgreich und setzt daher den Rückgabewert auf `ok` (s. Zeilen 17–18). Als letztes Modul in der `authorize`-Section wird

`pap` aufgerufen. Dieses Modul durchsucht die AVPs nach dem Attribut `User-Password`. Da das Attribut vorhanden ist (s. Zeile 3) und die `Cleartext-Password`-Variable vom `files`-Modul gesetzt wurde, setzt das `pap`-Modul die Variable `Auth-Type` auf `PAP` und den Rückgabewert auf `updated` (s. Zeile 20). Sobald die Variable `Auth-Type` gesetzt wurde, ist die Autorisierung abgeschlossen.

Nun beginnt die Authentifizierung des Benutzers `bob`.

```
22 (0) Found Auth-Type = PAP
23 (0) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
24 (0)   Auth-Type PAP {
25 (0)   pap: Login attempt with password
26 (0)   pap: Comparing with "known good" Cleartext-Password
27 (0)   pap: User authenticated successfully
28 (0)     [pap] = ok
29 (0)   } # Auth-Type PAP = ok
30 [...]
31 (0) Sent Access-Accept Id 217 from 127.0.0.1:1812 to 127.0.0.1:56213 length 0
32 (0) Finished request
```

Anders als bei der Autorisierung wird bei der Authentifizierung nicht jedes Modul der Reihe nach aufgerufen. Stattdessen wird anhand der Variable `Auth-Type` das passende Modul ausgewählt. [vgl. [Net18d](#)] In diesem Fall wird das Modul `pap` aufgerufen, welches das `User-Password` aus den RADIUS-AVPs mit dem vom `files`-Modul gesetzten `Cleartext-Password` vergleicht (s. Zeilen 24–29). Da die Passwörter identisch sind, wird der Rückgabewert auf `ok` gesetzt und die Authentifizierung ist abgeschlossen. Sowohl die Autorisierung als auch die Authentifizierung waren erfolgreich, daher wird ein `Access-Accept`-Paket an den Benutzer `bob` gesendet.

## 4.2. Konfiguration der Authentifizierungsserver

In diesem Kapitel wird die Konfiguration der Authentifizierungsserver beschrieben. Neben einer kurzen Beschreibung der Installation des Betriebssystems und des Domänenbeitritts, werden Vor- und Nachteile verschiedener Arten der FreeRADIUS-Installation, sowie die Konfiguration des FreeRADIUS-Servers selbst, erläutert.

### 4.2.1. Betriebssystem

Wie bereits in [Abschnitt 3.6.2](#) erwähnt wurde, wird als Betriebssystem der Server *Debian* in der aktuellen Stable-Version 9 (Stretch) verwendet, da dieses Betriebssystem bereits auf vielen Servern der Abteilung Informatik zum Einsatz kommt. Details zur Installation des Betriebssystems sind im [Anhang A](#) ersichtlich. Bis auf den Hostnamen und die IP-Adresse werden beide Server gleich konfiguriert. Der erste Server hat den Hostnamen `radius01` und die IP-Adresse `141.71.31.41`. Analog dazu hat der zweite Server den Hostname `radius02` und die IP-Adresse `141.71.31.42`. Da auch die FreeRADIUS-Konfiguration für

beide Server fast identisch ist, wird im Folgenden nur von *den Servern* gesprochen. Bei Unterschieden zwischen den beiden Servern werden stattdessen die Hostnamen `radius01` und `radius02` explizit erwähnt.

### 4.2.2. Auswahl der Bezugsquelle für FreeRADIUS

Um FreeRADIUS auf einem Debian-System zu installieren, kann die Software aus verschiedenen Quellen bezogen werden. Die erste Möglichkeit ist die Nutzung der **offiziellen Paketquellen** von Debian. Zur Zeit wird für Debian 9 die Version 3.0.12<sup>7</sup> paketierte, die im September 2016 veröffentlicht wurde. Die neueste FreeRADIUS-Version (3.0.17) wurde im April 2018 veröffentlicht. [Fre18b] Dies bedeutet allerdings nicht, dass die von Debian bereitgestellten Pakete unsicher sind, da sicherheitsrelevante Patches für aktuelle FreeRADIUS-Versionen von den Debian-Paket-Maintainern für die Version 3.0.12 zurückportiert werden. Lediglich auf neue Funktionen, die seit der Veröffentlichung von 3.0.12 implementiert wurden, muss bei der Verwendung der Debian-Pakete verzichtet werden. Die Nutzung der offiziellen Debian Paketquellen hat den Vorteil, dass Sicherheitsupdates für den FreeRADIUS-Server und für das restliche Betriebssystem aus der gleichen Quelle bezogen werden.

Die von dem FreeRADIUS-Gründer Alan DeKok geleitete Firma **Network RADIUS** stellt ebenfalls Pakete für Debian bereit.<sup>8</sup> Um diese zu nutzen, muss lediglich ein weiteres *Repository* in der Datei `/etc/apt/sources.list` hinzugefügt werden. Das Repository enthält Pakete für die FreeRADIUS-Version 3.0.16, allerdings können diese nur mit dem veralteten Debian 7 genutzt werden, dessen Support am 31. Mai 2018 endete, weshalb diese Bezugsquelle nicht in Frage kommt.

Die dritte Möglichkeit ist die Installation des FreeRADIUS-Servers in einem **Docker-Container**. Das FreeRADIUS-Projekt stellt offizielle Docker-Images<sup>9</sup> mit der aktuellen Version 3.0.17 zur Verfügung. In der Dokumentation von Docker werden Docker-Container-Images als leichtgewichtige, ausführbare Softwarepakete beschrieben, die alles, was zur Ausführung einer Anwendung erforderlich ist, enthalten. Dies betrifft alle von der Anwendung genutzten Bibliotheken, Laufzeitumgebungen und Konfigurationsdateien. Von einem *Docker-Container* spricht man, wenn das Image ausgeführt wird. [vgl. Doc18] Die Container können daher ohne Anpassungen auf jedem von Docker unterstützten Betriebssystem ausgeführt werden. Docker-Container sind leichtgewichtiger als virtuelle Maschinen, da Komponenten, wie der Kernel, zwischen dem Host-Betriebssystem und dem Container geteilt werden und keine vollständige Emulation eines Client-Betriebssystems durchgeführt wird. Da Docker-Container für die Ausführung von einer einzelnen Anwendung optimiert sind, haben sie weder ein Init-System, noch Möglichkeiten zur Interprozesskommunikation. Bei Anwendungen, die mit anderen Prozessen zusammenarbeiten müssen,

---

<sup>7</sup>Debian 9 Paket: <https://packages.debian.org/stretch/freeradius>

<sup>8</sup>Debian Repository von Network RADIUS: <http://packages.networkradius.com/debian>

<sup>9</sup>Offizielles Docker-Image: <https://hub.docker.com/r/freeradius/freeradius-server/>

kann dies zu Problemen führen, wie Liebel [Lie18] in seinem Handbuch „Skalierbare Container-Infrastrukturen“ beschreibt:

Eines der größten Probleme der *no-init*-Philosophie von Containern ist sehr elementar und nicht unbedeutend. Neben der Tatsache, dass sich selbst ein minimaler, kleiner Helper-Dienst im Container nicht ohne größeren (Scripting-)Aufwand mit *dem* Hauptprozess im Container darüber abstimmen kann, ob, wann, wie und in welcher Reihenfolge (Dependency-Modell) sie bitte schön starten möchten, fehlt schlicht und einfach so gut wie jede Kommunikationsmöglichkeit zwischen den Diensten, die beispielsweise ein *systemd*-Init standardmäßig via *DBUS* bereitstellt. Als Folge davon können im schlimmsten Fall *Container-Zombie-Prozesse* [...] innerhalb eines Containers entstehen, ohne dass *die* PID 1 via *SIGCHLD* davon Kenntnis erhält und die betreffenden Prozesse daher auch nicht beenden kann.

In der Dokumentation zur Active-Directory-Integration von FreeRADIUS wird erläutert, dass der Server der Domäne beitreten muss, damit eine Authentifizierung von Active-Directory-Accounts möglich ist. Zu diesem Zweck werden die Dienste *winbindd*, *smbd* und *nmbd* benötigt, die untereinander und mit dem FreeRADIUS-Server kommunizieren müssen. [vgl. Cud18] Bei der Installation in einem Docker-Container kann dies zu den von Liebel beschriebenen Problemen führen, weshalb eine klassische Installation unter Verwendung der offiziellen Debian-Paketquellen besser geeignet ist.

### 4.2.3. Domänenbeitritt

Der Domänenbeitritt der Server wird ebenfalls vor der FreeRADIUS-Installation durchgeführt, da dies eine Voraussetzung für die Authentifizierung der Active-Directory-Accounts ist, die in [Abschnitt 4.2.6](#) konfiguriert wird. Hierfür wird die freie Software-Suite *Samba* genutzt, die für eine bessere Kompatibilität zwischen Unix- und Windows-Systemen sorgt. [vgl. Sam18c] Um den Domänenbeitritt zu ermöglichen, werden die drei Dienste *smbd*, *nmbd* und *winbindd* benötigt, die Teil der Samba-Suite sind. Der Dienst *smbd* ist eine freie Reimplementierung des *SMB-Protokolls* („Server Message Block“), das von Windows zur Freigabe von Dateien genutzt wird. [vgl. Sam18b] Der *nmbd*-Dienst implementiert die *NetBIOS-Namensauflösung*, die von den Computern in einer Active-Directory-Domäne genutzt wird. [vgl. Sam18a] Der Dienst *winbindd* wird genutzt, um die Namen und Gruppenzugehörigkeiten von Active-Directory-Accounts aufzulösen und diese zu authentifizieren. [vgl. Sam18d]

Um diese Dienste auf einem Debian-System zu installieren, werden die Pakete *samba* (für *smbd* und *nmbd*), sowie *winbindd* benötigt. Die drei Dienste nutzen die Konfigurationsdatei */etc/samba/smb.conf*, die unter anderem Informationen wie den Domänennamen enthält. Der *winbindd*-Dienst nutzt für die Authentifizierung von Active-Directory-Accounts das Authentifizierungsprotokoll *Kerberos*. Neuman und Ts'o beschreiben Kerberos als ein verteiltes Authentifizierungsprotokoll, bei dem keine Informationen über das Netzwerk

übertragen werden, die einen Diebstahl der Identität der sich authentifizierenden Person erlauben. [vgl. NT94] Die benötigten Kerberos-Client-Programme werden von dem Paket `krb5-user` bereitgestellt. Die Anbindung dieser Programme an den Domain Controller der Abteilung Informatik wird in der Datei `/etc/krb5.conf` konfiguriert. Anschließend kann der Domänenbeitritt mit dem von Programm `net`, welches Teil der Samba-Suite ist, durchgeführt werden. Genauere Informationen zum Domänenbeitritt und dessen Verifizierung können im [Anhang A](#) nachgelesen werden.

### 4.2.4. Installation von FreeRADIUS

Nach der Vorbereitung des Betriebssystems und dem Domänenbeitritt, kann die Installation von FreeRADIUS erfolgen. Der FreeRADIUS-Server wird unter Debian von dem Paket `freeradius` bereitgestellt. Das Paket `freeradius-utils` enthält Client-Programme, die das konfigurieren und testen von FreeRADIUS vereinfachen. Da `apt`, der Paketmanager von Debian, standardmäßig alle empfohlenen Pakete installiert, wird bei der Installation der Parameter `--no-install-recommends` übergeben. Dies verhindert, dass viele unnötige Pakete, wie zum Beispiel die Java Runtime Environment und mehrere Icon-Pakete und Schriftarten installiert werden. Als `raddb`-Verzeichnis wird in der Standardkonfiguration des Debian-Pakets das Verzeichnis `/etc/freeradius/3.0/` genutzt.

### 4.2.5. Zertifikatsbasierte Authentifizierung

Um EAP-TLS zu konfigurieren, werden digitale Zertifikate für den FreeRADIUS-Server und die Supplicants (die Pool-PCs) benötigt, die von einer *Certificate Authority* (CA) signiert sind, der sowohl der Server als auch die Supplicants vertrauen. Eine CA ist eine zentrale Vergabestelle für digitale Zertifikate, die es ermöglicht, dass die Kommunikationspartner bei einer asymmetrischen Verschlüsselung die Authentizität ihres Gegenübers verifizieren können. Für alle Pool-PCs und die beiden Server werden insgesamt nur zwei Zertifikate benötigt – ein Server-Zertifikat und ein Supplicant-Zertifikat. Dadurch wird der Administrationsaufwand verringert, da eine automatisierte Verteilung von einem Zertifikat auf viele Pool-PCs einfacher zu implementieren ist, als die Verteilung individueller Zertifikate. Darüberhinaus wird die Erstellung und die regelmäßige Erneuerung der Zertifikate deutlich vereinfacht, wenn Zertifikate wiederverwendet werden. Der Ordner `raddb/certs/` enthält eine *Makefile* zur Erstellung dieser Zertifikate. Diese *Makefile* erzeugt ein selbstsigniertes CA-Zertifikat, das genutzt wird, um das Server- und Supplicant-Zertifikat zu signieren. Da das CA-Zertifikat selbstsigniert ist, muss es zusätzlich zu dem Supplicant-Zertifikat auf die Pool-PCs kopiert werden und dort als vertrauenswürdigen CA-Zertifikat eingetragen werden. Das IT-Team der Abteilung Informatik hat bereits eine eigene, selbstsignierte CA implementiert, der die Pool-PCs vertrauen. Es wäre daher möglich, die CA des IT-Teams zu nutzen, um die Server- und Supplicant-Zertifikate zu signieren, was die Verwaltung der Zertifikate vereinfacht. Allerdings wird in der FreeRADIUS-Dokumentation dazu geraten, die von FreeRADIUS

bereitgestellte Makefile zu verwenden, da die damit erzeugten Zertifikate mit allen Client-Betriebssystemen kompatibel sind. [vgl. [Smi18](#)] Darüberhinaus würde die Nutzung der CA des IT-Teams dazu führen, dass *jedes* von dieser CA signierte Zertifikat für die Authentifizierung über EAP-TLS gültig ist, was nicht gewünscht ist.

In den Dateien `ca.cnf`, `server.cnf` und `client.cnf`, die sich ebenfalls im Verzeichnis `raddb/certs/` befinden, wird die Gültigkeitsdauer der Zertifikate, sowie die Informationen, die diese enthalten, festgelegt. Für die Supplicant- und Server-Zertifikate wird eine Gültigkeit von *einem Jahr* festgelegt, während das CA-Zertifikat *zehn Jahre* gültig ist. Darüberhinaus werden in den Konfigurationsdateien Passwörter definiert, mit denen die privaten Schlüssel der Zertifikate verschlüsselt werden. Als Kontaktdaten werden in den drei Konfigurationsdateien die selben Werte benutzt, lediglich der Zertifikatname ist unterschiedlich:

- Land: *DE*
- Bundesland: *Niedersachsen*
- Ort: *Hannover*
- Organisation: *Hochschule Hannover*
- Email: *F4-I-IT-Team@hs-hannover.de*
- Name:
  - CA-Zertifikat: *HSH 802.1X Certificate Authority*
  - Server-Zertifikat: *HSH 802.1X RADIUS Server*
  - Supplicant-Zertifikat: *HSH 802.1X Client*

Die Makefile, die zur Erstellung der Zertifikate genutzt wird, enthält verschiedene Regeln, um die CA-, Server- und Supplicant-Zertifikate mit dem Programm *OpenSSL* zu erstellen und zu signieren. Mit dem Befehl `make all` werden alle drei Zertifikate erstellt. Es ist jedoch auch möglich mit den Befehlen `make ca`, `make server` und `make client` eines der Zertifikate zu spezifizieren. Um ein Zertifikat zu erneuern, muss zuerst das bestehende Zertifikat widerrufen und gelöscht werden. Die Makefile enthält hierfür keine Regeln, weshalb diese hinzugefügt werden:

```
revokeserver:
    openssl ca -revoke server.pem -keyfile ca.key -key $(PASSWORD_CA) -cert ca.pem
    -config ca.cnf

destroyserver:
    rm -f server.crt server.csr server.key server.p12 server.pem

revokeclient:
    openssl ca -revoke client.pem -keyfile ca.key -key $(PASSWORD_CA) -cert ca.pem
    -config ca.cnf

destroyclient:
    rm -f client.crt client.csr client.key client.p12 client.pem $(USER_NAME).pem
```

Durch das Aufrufen dieser Regeln ist es möglich, das Supplicant-Zertifikat mit dem Befehl `make revokeclient destroyclient client` zu erneuern. Regeln für die Erneuerung des CA-Zertifikats sind nicht nötig, da nach der Änderung dieses Zertifikats ohnehin alle Zertifikate erneuert werden müssen, was mit dem Befehl `make destroycerts all` möglich ist. Die Erstellung und Erneuerung der Zertifikate wird nur auf dem Server `radius01` durchgeführt. Anschließend werden das CA-Zertifikat und das Server-Zertifikat, sowie dessen privater Schlüssel auf den Server `radius02` kopiert.

Damit EAP-TLS-Anfragen von FreeRADIUS bearbeitet werden können, muss das `eap`-Modul in der `authenticate`- und `authorize`-Sections des virtuellen Servers `default` aufgelistet sein. Das Modul implementiert neben EAP-TLS auch weitere EAP-Methoden, wie PEAP und EAP-MD5, und ist standardmäßig in der Konfiguration des virtuellen Servers `default` aktiviert. [vgl. [Net18g](#)] Die Konfigurationsdatei des Moduls ist `raddb/mods-available/eap`. Die Datei enthält eine Section mit dem Namen `eap`, die wiederum weitere Sections für jede unterstützte EAP-Methode (`tls`, `peap`, `md5` etc.) enthält. Damit bei der Konfiguration der TLS-basierten EAP-Methoden (EAP-TLS, EAP-TTLS und PEAP) nicht die gleichen Konfigurationswerte in drei verschiedenen Sections gesetzt werden müssen, besteht die Möglichkeit, die grundlegende TLS-Konfiguration in die Section `tls-config tls-common` auszulagern. Innerhalb der `tls`-, `peap`- und `ttls`-Sections kann diese Konfiguration mit dem Setzen des Konfigurationswerts `tls = tls-common` referenziert werden, sodass nur noch die spezifischen Optionen für jede EAP-Methode angegeben werden müssen. Nahezu alle Konfigurationswerte für die `tls`-Section können in die `tls-common`-Section ausgelagert werden, sodass `tls = tls-common` in der Regel der einzige Eintrag in der `tls`-Section ist. [vgl. [Net18h](#)] Der folgende Auszug aus der Konfigurationsdatei zeigt diese Referenzierung.

```
eap {
  tls-config tls-common {
    private_key_file = /etc/freeradius/3.0/certs/server.key
    certificate_file = /etc/freeradius/3.0/certs/server.crt
    ca_file = /etc/freeradius/3.0/certs/ca.pem
    [...]
  }
  tls {
    tls = tls-common
  }
  peap {
    tls = tls-common
    virtual_server = "inner-tunnel"
    [...]
  }
}
```

Code-Beispiel 2: Mehrere TLS-basierte EAP-Typen referenzieren die gemeinsame TLS-Konfiguration in der `tls-common`-Section

In der `tls-common`-Subsection werden die Pfade des privaten Schlüssels des Servers, sowie die Zertifikate der CA und des Servers angegeben. Die anderen Werte der Section müssen

nicht geändert werden. Über den Konfigurationswert `cipher_list` kann eine OpenSSL-Chiffrenliste angegeben werden. Die standardmäßig ausgewählte Liste `DEFAULT`, deren Inhalt bei der Kompilierung von OpenSSL durch die Debian-Entwickler festgelegt wurde, könnte durch die Liste `HIGH` ersetzt werden, die von den OpenSSL-Entwicklern festgelegt wurde und nur Chiffren mit einer Schlüssellänge von mindestens 128 Bit enthält. Da die `DEFAULT`-Liste von Debian keine Chiffren mit einer Schlüssellänge von unter 128 Bit oder unsichere Chiffren wie RC4 enthält, bringt eine Änderung der Chiffrenliste keinerlei Vorteile mit sich. Auch der Zufallszahlengenerator (`random_file`) ist mit `/dev/urandom` bereits gut gewählt. Der Handbuchseite zu `random(4)` kann entnommen werden, dass sowohl `/dev/urandom` als auch die Alternative `/dev/random` einen sogenannten *Entropiepool* nutzen, um pseudozufällige Zahlen zu erzeugen. Wenn der Entropiepool leer ist, blockiert `/dev/random` bis genügend Entropie vorhanden ist. Daraus folgt, dass Programme, die aus `/dev/random` lesen, in einer solchen Situation warten müssen. [vgl. Lin17] Dies würde bedeuten, dass FreeRADIUS eine Zeit lang nicht auf Anfragen reagieren könnte, was nicht akzeptabel ist. Da `/dev/urandom` *nicht* blockiert, wenn der Entropiepool leer ist, wird der Zufallszahlengenerator in der `tls-common`-Section nicht geändert.

Um die Implementierung von EAP-TLS und deren Skalierbarkeit zu testen, wird das Programm `eapol_test` verwendet. Es wird eine Konfigurationsdatei mit dem Namen `eap-tls.conf` und den folgenden Parametern angelegt.

```
network={
  identity="anonymous"
  eap=TLS
  ca_cert="./ca.pem"
  client_cert="./client.crt"
  private_key="./client.key"
  private_key_passwd="*****"
}
```

Um zu testen, ob die Authentifizierung erfolgreich ist, wird das Programm `eapol_test` mit dieser Konfiguration aufgerufen. Das Programm ist ein Authenticator, daher muss mit dem Parameter `-s testing123` das Passwort angegeben werden, das der Authentifizierungs-server und der Authenticator teilen. „testing123“ ist bei FreeRADIUS standardmäßig als das Passwort für den Localhost festgelegt.

```
eapol_test -c eap-tls.conf -s testing123
[...]
```

```
SUCCESS
```

Die Ausgabe `SUCCESS` gibt an, dass die Authentifizierung erfolgreich war. Zusätzlich kann mit dem Programm `time` gemessen werden, wie viel Zeit der Authentifizierungsprozess in Anspruch nimmt. Die Ausgabe des Kommandos wird nach `/dev/null` umgeleitet, damit ausschließlich die Bearbeitungszeit gemessen wird und die Ausgabe auf der Konsole das Ergebnis nicht verfälscht.

```
time eapol_test -c eap-tls.conf -s testing123 > /dev/null
real    0m0.025s
user    0m0.016s
sys     0m0.000s
```

Der Zeitwert `real` gibt die Zeit an, die vom Aufruf von `eapol_test` bis zu dessen erfolgreicher Beendigung vergangen ist. Der angegebene Wert ist der Durchschnitt von 20 Authentifizierungen, die zwischen 19 und 35 Millisekunden dauerten. Dies bedeutet, dass ein Server im Durchschnitt 40 EAP-TLS Anfragen pro Sekunde beantworten kann. In [Abschnitt 4.3](#) wird der Lastenausgleich zwischen beiden Authentifizierungsservern konfiguriert. Daraus folgt, dass insgesamt im Durchschnitt 80 EAP-TLS Anfragen beantwortet werden können, was mehr als ausreichend für die Abteilung Informatik ist.

### 4.2.6. Authentifizierung von Active-Directory-Accounts

In [Abschnitt 3.2.1](#) wurde festgelegt, dass die Authentifizierung der Active-Directory-Accounts des Abteilungspersonals über PEAP-MSCHAPv2 ablaufen soll. Damit FreeRADIUS diese Anfragen bearbeiten kann, müssen die virtuellen Server wie folgt konfiguriert werden.

1. Der virtuelle Server `default` muss PEAP-Anfragen bearbeiten und die im inneren Tunnel enthaltenen EAP-Pakete an den virtuellen Server `inner-tunnel` weiterreichen.
2. Der virtuelle Server `inner-tunnel` muss EAP-MSCHAPv2-Anfragen bearbeiten können und muss gewährleisten, dass nur die AD-Accounts des Abteilungspersonals autorisiert und authentifiziert werden.

Da PEAP ebenfalls vom Modul `eap` implementiert wird, können die TLS-spezifischen Einstellungen referenziert werden, die im Rahmen der Konfiguration von EAP-TLS getätigt wurden. Der virtuelle Server, an den die EAP-Pakete des inneren Tunnels weitergereicht werden, wird mit dem Konfigurationswert `virtual_server` festgelegt. Die Konfiguration ist in [Code-Beispiel 2](#) abgebildet.

Die Unterstützung von EAP-MSCHAPv2 wird von den Modulen `eap` und `mschap` implementiert. Damit der virtuelle Server `inner-tunnel` Anfragen mit diesem EAP-Typ bearbeiten kann, werden in der `authorize`-Section das `eap`-Modul und in der `authenticate`-Section die Module `eap` und `mschap` definiert. Durch diese Konfiguration wird bei der Bearbeitung der eingehenden Anfragen in der Autorisierungsphase der `Auth-Type` auf EAP gesetzt, da das `eap`-Modul nur prüft, ob die Anfrage ein EAP-Paket enthält. In der Authentifizierungsphase überprüft das `eap`-Modul den EAP-Typ der Anfrage und ruft das `mschap`-Modul auf, um die eigentliche Authentifizierung durchzuführen. Die Konfigurationsdatei des `mschap`-Moduls ist `raddb/mods-available/mschap`. In dieser Datei wird der Pfad, sowie die Parameter des Programms `ntlm_auth` definiert, das Teil der Samba-Suite ist und es ermöglicht, die Authentifizierungsfunktion des `winbindd`-Dienstes zu nutzen.

```
ntlm_auth = "/usr/bin/ntlm_auth --request-nt-key \  
--username=%{%{Stripped-User-Name}:-%{%{User-Name}:-None}} \  
--challenge=%{%{mschap:Challenge}:-00} \  
--nt-response=%{%{mschap:NT-Response}:-00}"
```

Durch die in den Parametern enthaltenen Variablen wird das Programm mit den im EAP-MSCHAPv2-Paket enthaltenen Anmeldedaten aufgerufen, wodurch eine Authentifizierung von *allen* AD-Accounts möglich ist.

Um die Authentifizierung auf das Abteilungspersonal zu beschränken, wird eine Gruppe mit dem Namen *F4-I-802.1X-Users* im Active Directory angelegt, die alle Accounts und Gruppen enthält, für die eine Authentifizierung möglich sein soll. Das `ldap`-Modul kann in der *Autorisierungsphase* genutzt werden, um Accounts herauszufiltern, die nicht Mitglied der AD-Gruppe sind. Dieses Modul verbindet sich mit dem Domain Controller und sucht in der Gruppe *F4-I-802.1X-Users* nach dem Accountnamen, der in der Anfrage enthalten ist. Wenn der Account nicht gefunden wird, setzt das `ldap`-Modul den Rückgabewert auf `notfound` und signalisiert damit eine gescheiterte Autorisierung, woraufhin keine Authentifizierung durchgeführt wird. [vgl. [Net18i](#)] Wird der Account stattdessen gefunden, wird mit der normalen Authentifizierung fortgefahren. Alternativ kann die Prüfung der Gruppenzugehörigkeit vom `mschap`-Modul in der *Authentifizierungsphase* durchgeführt werden. Bei dieser Variante wird jeder Account autorisiert, jedoch gelingt die Authentifizierung nur wenn die Anmeldedaten korrekt sind und der Account Teil der AD-Gruppe ist. Der Vorteil der ersten Variante ist die klarere Trennung von Autorisierung und Authentifizierung, was in der Praxis allerdings keine Rolle spielt, da die Resultate die selben sind. Die zweite Variante ist dafür aber weniger komplex und fehleranfällig, da das `ldap`-Modul nicht benötigt wird und deaktiviert werden kann, weshalb diese Variante genutzt wird. Um die Überprüfung der Gruppenzugehörigkeit in der Authentifizierungsphase zu implementieren, wird dem `ntlm_auth`-Aufruf in der Konfigurationsdatei des `mschap`-Moduls folgender Parameter hinzugefügt.

```
--require-membership-of='FH-H\F4-I-802.1X-Users'
```

Genauere Informationen zu der Konfiguration der Authentifizierung von Active-Directory-Accounts können dem [Anhang A](#) entnommen werden.

Die korrekte Implementierung von PEAP-MSCHAPv2 kann ebenfalls mit `eapol_test` überprüft werden. Die Konfigurationsdatei `peap-mschapv2.conf` hat den folgenden Inhalt.

```
network={
  eap=PEAP
  identity="rzm-p24"
  anonymous_identity="anonymous"
  password="*****"
  phase2="authheap=MSCHAPV2"
}
```

Das Programm `eapol_test` wird auf dem Authentifizierungsserver aufgerufen.

```
eapol_test -c peap-mschapv2.conf -s testing123
[...]  
SUCCESS
```

Die Ausgabe `SUCCESS` gibt an, dass die Authentifizierung erfolgreich war. Da der Authentifizierungsprozess bei PEAP-MSCHAPv2 durch die Verwendung von zwei virtuellen

Servern, sowie der Kommunikation mit dem Domain Controller, komplexer als die EAP-TLS Authentifizierung ist, dauert die Authentifizierung ein wenig länger.

```
time eapol_test -c peap-mschapv2.conf -s testing123 > /dev/null
real    0m0.064s
user    0m0.004s
sys     0m0.004s
```

Bei einer durchschnittlichen Dauer von 64 Millisekunden ergibt sich, dass ein Authentifizierungsserver ungefähr 15 PEAP-MSCHAPv2-Anfragen pro Sekunde bearbeiten kann. Durch den Lastenausgleich erhöht sich dieser Wert auf 30 Anfragen pro Sekunde, was mehr als ausreichend ist.

### 4.2.7. Hinzufügen der Authentikatoren

In der Konfigurationsdatei `raddb/clients.conf` werden die Authentikatoren definiert, die RADIUS-Pakete an die Authentifizierungsserver schicken können. Die Datei enthält mehrere benannte `client`-Sections, die eine IP-Adresse und ein Passwort enthalten.

```
client name {
    ipaddr = 141.71.31.45
    secret = examplepassword
}
```

Es ist möglich, statt einer IP-Adresse auch einen Hostname zu definieren. Allerdings wird in der Dokumentation davon abgeraten, da in diesem Fall der FreeRADIUS-Server beim Starten versucht die Hostnamen aufzulösen. Wenn die Namensauflösung fehlschlägt beendet sich der Server, was nicht akzeptabel ist. Um zusätzlich eine Kommunikation über IPv6 zu ermöglichen, muss eine zweite Section für den Authenticator definiert werden, in der die IPv6-Adresse enthalten ist. [vgl. [Net18j](#)] In dieser Datei werden Einträge für jeden Switch definiert, auf dem 802.1X aktiviert wird. Die Anbindung der Switche an die Authentifizierungsserver wird in [Abschnitt 4.3](#) beschrieben.

## 4.3. Konfiguration der Switche

In diesem Kapitel wird die Anbindung der Switche an die Authentifizierungsserver, sowie die Aktivierung der 802.1X-Authentifizierung auf den Switchports beschrieben. Genauere Informationen, sowie alle Kommandozeilenbefehle, können in [Anhang A](#) nachgelesen werden.

Um einen Switch an die Authentifizierungsserver anzubinden, muss zuerst mit dem Befehl `aaa new-model` die *AAA-Funktion* des Switches aktiviert werden. Danach werden RADIUS-Server-Einträge für beide Authentifizierungsserver angelegt. Beim Hinzufügen eines Authentifizierungsservers muss das Passwort eingegeben werden, das bei der Konfiguration von FreeRADIUS in der Datei `raddb/clients.conf` für diesen Switch vergeben wurde (vgl. [Abschnitt 4.4](#)). Daraufhin wird die in [Abschnitt 3.5.2](#) beschriebene

Authentifizierungsserver-Gruppe angelegt, die Verweise auf die RADIUS-Server-Einträge der beiden Server enthält. Durch das Anlegen der Gruppe ist automatisch die Hochverfügbarkeit des RADIUS-Dienstes für alle Supplicants, die an diesen Switch angeschlossen sind, gewährleistet. In der Gruppe wird mit dem Befehl `load-balance` der Lastenausgleich mit einer Stapelgröße von zehn konfiguriert.

Nun muss die 802.1X-Authentifizierung an den Switchports konfiguriert werden. Wie in [Abschnitt 3.4](#) beschrieben wurde, wird dies nur an den Switchports getan, an denen Geräte angeschlossen sind, die den 802.1X-Standard unterstützen. In den Pool-Räumen, in denen ein zusätzlicher Switch eingesetzt wird, um einige der Pool-PCs anzuschließen, wird die 802.1X-Authentifizierung nur auf den Ports des Switches im Pool-Raum aktiviert, nicht aber auf der Zuleitung in den Serverraum. [Abb. 8](#) zeigt die Konfiguration der Switchports bei diesem Sonderfall.

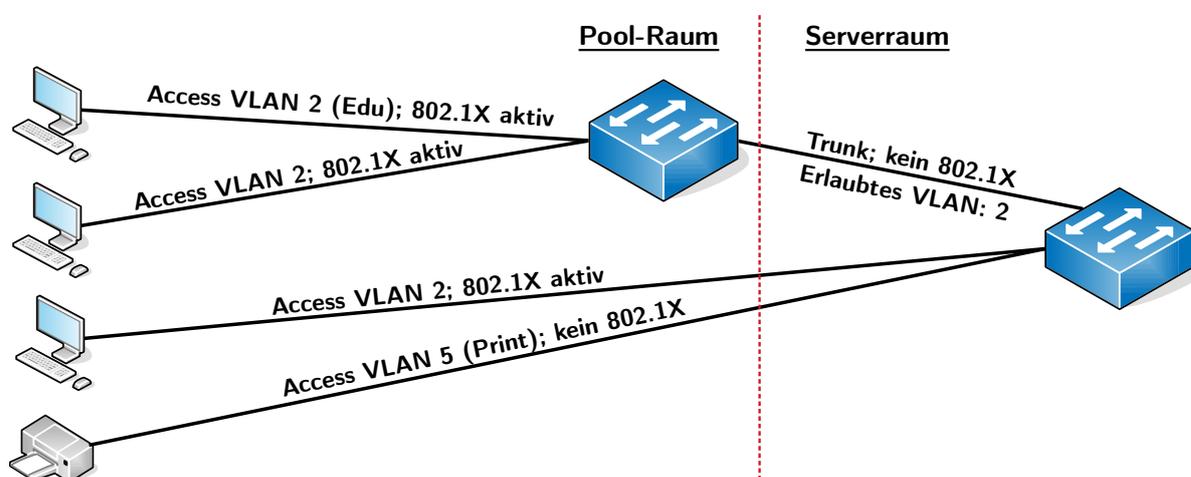


Abbildung 8: Konfiguration der Switchports in Räumen mit zusätzlichem Switch

Da an dem Switch im Pool-Raum nur Geräte angeschlossen sind, die sich im **VLAN** des Edu-Netztes (2) befinden, wird auf dem Trunk-Port, der die beiden Switche miteinander verbindet, nur das **VLAN 2** erlaubt. Würden die erlaubten **VLANs** auf dem Trunk-Port nicht eingeschränkt werden, so wäre es möglich, dass eine unbefugte Person Zugriff auf alle **VLANs** bekommt, die dem Switch im Server-Raum bekannt sind, indem sie das Kabel entfernt, das die beiden Switche verbindet, und für ihr eigenes Gerät benutzt. Da aber auf dem Trunk-Port nur das **VLAN 2** erlaubt ist, könnte diese unbefugte Person nur Zugriff auf dieses **VLAN** erlangen.

#### 4.4. Konfiguration der Supplicants

In diesem Kapitel wird die Konfiguration von 802.1X auf den Pool-PCs und den PCs des Abteilungspersonals beschrieben. Darüberhinaus wird erläutert, wie die Zertifikate auf einem sicheren Weg auf die Pool-PCs verteilt werden können.

### 4.4.1. Pool-PCs

Damit die Pool-PCs sich automatisch über EAP-TLS authentifizieren, muss die Konfiguration der Netzwerkkarte, sowie das CA- und Supplicant-Zertifikat, automatisiert auf die PCs verteilt werden. Die Konfigurationsdatei für die Netzwerkkarte muss in dem Verzeichnis `/etc/NetworkManager/system-connections/` abgelegt werden, damit der *NetworkManager*, der die Netzwerkeinstellungen der Pool-PCs verwaltet, diese einlesen kann. [Code-Beispiel 3](#) zeigt diese Konfigurationsdatei. Die Zertifikate werden in dem Verzeichnis `/root/` abgelegt, da die Accounts der Studierenden auf dieses Verzeichnis nicht zugreifen können. Die Pfade der Zertifikate sind in den Zeilen 14, 15 und 19 der Konfigurationsdatei definiert. Die Datei enthält darüberhinaus auch das Passwort zum Entschlüsseln des privaten Schlüssels des Supplicant-Zertifikats (s. Zeile 20). Die Tatsache, dass dieses Passwort im Klartext in der Datei enthalten ist, beeinträchtigt nicht die Sicherheit, da nur der `root`-Account diese Datei lesen kann.

```
1 [connection]
2 id=Kabelnetzwerkverbindung 1
3 uuid=REPLACEUUID
4 type=ethernet
5 permissions=
6 secondaries=
7
8 [ethernet]
9 duplex=full
10 mac-address-blacklist=
11
12 [802-1x]
13 altsubject-matches=
14 ca-cert=/root/ca.pem
15 client-cert=/root/client.p12
16 eap=tl;
17 identity=hsh
18 phase2-altsubject-matches=
19 private-key=/root/client.p12
20 private-key-password=*****
21
22 [ipv4]
23 dns-search=
24 method=auto
25
26 [ipv6]
27 addr-gen-mode=stable-privacy
28 dns-search=
29 ip6-privacy=0
30 method=auto
```

Code-Beispiel 3: Vorlage für die Netzwerkkonfiguration der Pool-PCs

Das IT-Team der Abteilung Informatik verwendet selbsterstellte Debian-Pakete um Programme, Konfigurationen und andere Dateien auf die Pool-PCs zu verteilen. Dies

ist für die Verteilung der Zertifikate und der Konfiguration keine Option, da die Pakete von allen Studierenden heruntergeladen werden können, wenn sie die entsprechende URL des Webservers kennen, der die Pakete verteilt. Für vertrauliche Dateien werden daher spezielle Pakete verwendet, die bei der Installation den Hostnamen des Pool-PCs an einen Server schicken. Nur wenn der Hostname des Pool-PCs in einer Weißliste auf dem Server enthalten ist, kopiert der Server die vertraulichen Dateien per `scp` auf den Pool-PC. Dieser Ansatz ist sicherer, da Geräte mit nicht freigegebenen Hostnamen die Dateien nicht erhalten können. Wenn ein gültiger Hostname von einem nicht freigegebenen Gerät gesendet wird, kopiert der Server die Dateien trotzdem auf den Pool-PC und nicht auf das nicht freigegebene Gerät. [Code-Beispiel 4](#) zeigt den Entwurf eines Bash-Skripts, das die sichere Übertragung der Dateien implementiert.

Dem Bash-Skript wird als Parameter der Name des Pool-PCs übergeben, der die Konfiguration angefordert hat. Die in [Code-Beispiel 3](#) abgebildete Konfigurationsvorlage und die Zertifikate müssen im gleichen Verzeichnis wie das Bash-Skript liegen. Die Zertifikate dürfen nur für den `root`-Account les- und schreibbar sein. Die Vorlage enthält in der dritten Zeile den Eintrag `uuid=REPLACEUUID`. Damit diese UUID (Beispiel: `59fc7bf3-c317-46d6-a104-395ec58297ad`) nicht auf jedem Pool-PC gleich ist, wird mit dem Programm `uuidgen` eine zufällige UUID erzeugt (s. Zeile 14), die die Zeichenkette `REPLACEUUID` in der Vorlage ersetzt (s. Zeile 25). Die veränderte Vorlage wird in eine temporäre Datei geschrieben, deren Lese- und Schreibrechte auch auf den `root`-Account begrenzt werden. Danach wird diese temporäre Datei über `scp` an die korrekte Position auf dem Pool-PC kopiert (s. Zeile 30–32). Durch die Option `-p` werden Lese- und Schreibrechte der Datei beim kopieren nicht verändert. Die Option `-o StrictHostKeyChecking=yes` bewirkt, dass der Kopiervorgang abgebrochen wird, wenn der SSH-Host-Key des Pool-PCs dem Server nicht bekannt ist. Da alle Host-Keys der Pool-PCs vom IT-Team vorkonfiguriert werden, sind alle legitimen Host-Keys bekannt und diese Option bietet eine zusätzliche Sicherheit.

Ein großes Problem dieses Lösungsansatzes ist, dass die Übertragung der Daten nur funktioniert, wenn an dem Switchport, an dem der Pool-PC angeschlossen ist, keine 802.1X-Authentifizierung erzwungen wird. Die Verteilung der Dateien auf die Pool-PCs ist nach deren Neuinstallation notwendig, die ein Mal pro Semester durchgeführt wird. Für die Neuinstallation starten die Pool-PCs über [PXE](#) ein Bootimage, was ebenfalls mit aktiviertem 802.1X nicht möglich ist. Es ist daher notwendig vor der Neuinstallation der Pool-PCs die 802.1X-Authentifizierung auf den Switchports zu deaktivieren und nach der Installation wieder zu aktivieren, was den Aufwand des gesamten Prozesses stark erhöht. Ein besserer Lösungsansatz für dieses Problem erfordert die Implementierung eines zusätzlichen [VLANs](#) und ist mit umfassenden Änderungen am Netzwerk der Abteilung Informatik verbunden, weshalb dieser Ansatz nur theoretisch betrachtet wird. Auf allen Switchen, an denen Pool-PCs angeschlossen sind, wird ein zusätzliches [VLAN](#) mit dem Namen *Edu-Auth-Fail* implementiert. Die Switches müssen so konfiguriert werden, dass alle Geräte, die sich nicht über 802.1X authentifizieren können, diesem [VLAN](#) zugeordnet werden. Aus diesem [VLAN](#) darf nur ein Zugriff auf alle Server möglich sein, die für die Installation der Pool-PCs benötigt werden. Mit diesem Lösungsansatz bekommen die

```
1 #!/bin/bash
2 #
3 # Dieses Skript kopiert die Zertifikate und Netzwerkkonfiguration fuer 802.1X
4 # ueber einen sicheren Kanal auf den als Parameter uebergebenen Pool-PC
5 #
6 # Autor: Simon Rose
7
8 remote_host=$1
9 whitelist_file=./hostnames.txt
10
11 if [[ -z $remote_host ]]; then
12     echo "Error: Kein Hostname gefunden!"
13     echo "Beispiel: $0 pool123.edu.inform.hs-hannover.de"
14     exit 1
15 elif ! grep -Fxq "$remote_host" $whitelist_file; then
16     echo "Hostname $remote_host nicht freigegeben"
17     exit 1
18 fi
19
20 uuid=$(uuidgen)
21 template_file=./networktemplate
22 tmp_file=./tmp_file-$uuid
23 ca_cert=./ca.pem
24 client_cert=./client.p12
25 remote_config_dir=/etc/NetworkManager/system-connections
26 remote_cert_dir=/root
27 remote_user=root
28 remote_host=$1
29
30 # UUID im Konfigurationstemplate ersetzen und $tmp_file schreiben
31 sed "s/REPLACEUUID/$uuid/" $template_file > $tmp_file
32 # Berechtigungen der Konfigurationen muessen "-rw-----" sein
33 chmod 600 $tmp_file
34
35 # NetworkManager-Konfiguration auf den Pool-PC kopieren
36 scp -i id_rsa -p -o StrictHostKeyChecking=yes \
37     $tmp_file \
38     $remote_user@$remote_host:$remote_config_dir/Kabelnetzwerkverbindung\ 1
39 # Temporäre Konfigurationsdatei entfernen
40 rm $tmp_file
41
42 # CA-Zertifikat und Client-Zertifikat auf den Pool-PC kopieren
43 scp -i id_rsa -p -o StrictHostKeyChecking=yes \
44     $ca_cert $client_cert \
45     $remote_user@$remote_host:$remote_cert_dir/
```

Code-Beispiel 4: Entwurf eines Bash-Skripts zur sicheren Übertragung der Zertifikate und der Netzwerkkonfiguration auf die Pool-PCs

Pool-PCs beim PXE-Start Zugriff auf das VLAN Edu-Auth-Fail, da sie sich nicht über 802.1X authentifizieren können. In diesem VLAN kann die Installation der Pool-PCs und die in diesem Kapitel beschriebene sichere Übertragung der Zertifikate, sowie der 802.1X-Konfiguration, durchgeführt werden. Nach einem Neustart der Pool-PCs können sich diese über 802.1X authentifizieren und bekommen daher Zugriff auf das Edu-VLAN.

### 4.4.2. PCs des Abteilungspersonal

Wie bereits in [Abschnitt 3.2.1](#), soll es den Mitarbeitenden der Abteilung Informatik möglich sein, die 802.1X-Authentifizierung ihrer Geräte selbst zu konfigurieren. Zu diesem Zweck wurde eine Anleitung angefertigt, die die Konfiguration von 802.1X für die Betriebssysteme Windows, macOS und GNU/Linux erläutert. Die Anleitung ist dieser Arbeit angehängt (s. [Anhang B](#)). Unter macOS wird für die Konfiguration ein *802.1X-Profil* benötigt, welches nur mit dem Programm „Apple Configurator 2“ erstellt werden kann. In [Anhang A](#) wird die Erstellung dieses Profils beschrieben. Um die Konfiguration von 802.1X unter macOS zu vereinfachen, wird diese Profildatei zusammen mit der Anleitung zur Verfügung gestellt.

## 5. Fazit

Ziel dieser Arbeit war es, zu untersuchen, in welcher Form der [IEEE-802.1X](#)-Standard im Netzwerk der Abteilung Informatik der Hochschule Hannover implementiert werden kann, um die Netzwerksicherheit zu erhöhen. Es wurde erläutert, wie durch 802.1X sichergestellt wird, dass nur authentifizierte Geräte auf das Netzwerk zugreifen können und wie Supplicant, Authenticator und Authentifizierungsserver über die Protokolle EAP und RADIUS miteinander kommunizieren. Darüberhinaus wurde analysiert für welche Teile des Netzwerks der Abteilung Informatik eine Implementierung von 802.1X sinnvoll ist, welche Geräte authentifiziert werden müssen und in welchen Teilen eine Implementierung eher hinderlich ist. Die Analyse zeigte, dass eine Authentifizierung der Pool-PCs geeignet ist, um die Sicherheit des Edu-Teilnetzes zu gewährleisten, da diese sich in öffentlich zugänglichen Räumen befinden. Des Weiteren zeigte sie, dass 802.1X im Inform-Teilnetz implementiert werden kann, um die Büroräume des Abteilungspersonals abzusichern. Allerdings wurde auch dargestellt, dass in vielen Räumen „Unmanaged-Switches“ zum Einsatz kommen, die die für 802.1X benötigten Funktionen nicht unterstützen.

Daraufhin wurde ein Konzept für den Einsatz von 802.1X in diesen Teilnetzen erarbeitet. Für die Authentifizierung der Pool-PCs wurde die EAP-Methode EAP-TLS gewählt, da diese digitale Zertifikate nutzt, welche eine hohe Sicherheit bieten und sich am besten für die Authentifizierung von Geräten eignen, die von vielen Personen genutzt werden. Für die PCs des Abteilungspersonals wurde die Methode PEAP-MSCHAPv2 gewählt, da diese es ermöglicht, die bestehenden Accounts im Active Directory der Abteilung Informatik zu verwenden. Darüberhinaus bietet diese Methode durch die [TLS](#)-Verschlüsselung ebenfalls ein hohes Maß an Sicherheit. Die Hochverfügbarkeit des RADIUS-Dienstes, ohne den bei der Verwendung von 802.1X für niemanden der Zugriff auf das Netzwerk möglich ist, wird durch die Nutzung einer Authentifizierungsserver-Gruppe auf den Switchen sichergestellt, da dies eine sehr simple und dennoch effektive Lösung ist. Als RADIUS-Server-Software wurde FreeRADIUS gewählt, da dieser Server die erforderlichen EAP-Methoden, sowie die Anbindung an das Active Directory unterstützt und zudem kostenlose und freie Software ist. Abschließend wurde die Realisierbarkeit des Konzepts geprüft, indem zwei Authentifizierungsserver installiert und für die Authentifizierung der Pool-PCs und Active-Directory-Accounts des Abteilungspersonals vorbereitet wurden, was in [Anhang A](#) dokumentiert wurde. Die Authentifizierung der Active-Directory-Accounts wurde mit den Betriebssystemen Windows, macOS und GNU/Linux konfiguriert und ist in [Anhang B](#) in Form einer Anleitung für Endnutzer dokumentiert. Für die Verteilung der Zertifikate auf die Pool-PCs wurde ein sicherer Ansatz vorgestellt, bei dem die sensiblen Daten über eine verschlüsselte Verbindung von einem zentralen Server auf die PCs verteilt werden.

Bei der Prüfung der Realisierbarkeit sind allerdings auch Probleme deutlich geworden. Zum einen ist das Konzept nur umsetzbar, wenn die Unmanaged-Switches durch andere Modelle ersetzt werden, die die benötigten Funktionen unterstützen. Zum anderen ist die Neuinstallation von Pool-PCs und die anschließende Übertragung der Zertifikate zur Konfiguration von 802.1X nur möglich, wenn die 802.1X-Authentifizierung auf den

Switchports vorübergehend deaktiviert wird. In [Abschnitt 4.4.1](#) wurde ein alternativer Lösungsansatz vorgestellt, der dieses Problem durch ein weiteres VLAN löst, das für Geräte genutzt wird, die sich nicht erfolgreich authentifizieren können. Grundsätzlich ist 802.1X aber ein wirksames Mittel zur Stärkung der Sicherheit des Netzwerks der Abteilung Informatik. Das in dieser Arbeit entworfene Konzept ist – abseits der soeben geschilderten Probleme – gut für die Abteilung Informatik geeignet, sollte aber erst nachdem diese Probleme gelöst sind im produktiven Netz umgesetzt werden.

## Glossar

- AES** Der „Advanced Encryption Standard“ ist eine symmetrische Blockchiffre. Zum Zeitpunkt des Schreibens dieser Arbeit existiert keine vollständige Kryptoanalyse. [10](#)
- CHAP** Das „Challenge Handshake Authentication Protocol“ ist ein Authentifizierungsverfahren, bei dem der Server eine Zufallszahl (Challenge) an den Client schickt und dieser von seinem Passwort und der Challenge eine Hashsumme bildet und diese zurückschickt. Anders als bei PAP, wird das Passwort also nicht im Klartext übertragen. [10](#), [11](#), [21](#), [31](#), [32](#)
- DMZ** Eine „Demilitarisierte Zone“ ist ein Computernetzwerk, das Zugriff auf die darin enthaltenen Server aus einem nicht vertrauenswürdigen Netzwerk wie dem Internet zulässt. Der Zugriff ist allerdings durch Firewall geschützt. [18](#)
- HMAC** Ein „hash-based Message Authentication Code“ wird genutzt, um die Authentizität und Integrität einer Nachricht zu gewährleisten. Zur Erzeugung einer HMAC wird eine Hashsumme von der Nachricht und einem Passwort gebildet. Das Passwort muss dem Empfänger und dem Absender der Nachricht bekannt sein. [10](#)
- IANA** Die „Internet Assigned Numbers Authority“ ist eine Organisation, die für die Vergabe von Namen und Nummern im Internet zuständig ist, wie zum Beispiel IP-Adressen. [10](#), [13](#)
- IEEE** Das „Institute of Electrical and Electronics Engineers“ bildet unter anderem Gremien für die Standardisierung von Technologien, wie zum Beispiel WLAN (IEEE 802.11) und IEEE 802.1X. [4](#), [5](#), [10](#), [52](#)
- MAC** Die MAC-Adresse („Media Access Control“) ist die Hardwareadresse eines Netzwerkadapters und identifiziert diesen in einem Netzwerk eindeutig. [4](#), [14](#), [19](#), [20](#), [23](#), [24](#)
- MD5** „Message Digest 5“ ist eine kryptografische Hashfunktion, die 128 Bit lange Hashwerte erzeugt. 2004 wurde die erste Kollision von MD5-Hashwerten erzeugt. [10](#), [14](#), [15](#), [20](#)
- NAS** Ein „Network Attached Storage“ (deutsch: *netzgebundener Speicher*) ist ein einfach zu verwaltender Dateiserver. Die meisten Modelle werden mit spezieller Hard- und Software ausgeliefert, die für die Verwendung als Dateiserver optimiert ist. [23](#)
- NTP** Das „Network Time Protocol“ dient dazu, die Systemzeit von mehreren Computern zu synchronisieren. [19](#)

- PAP** Das „Password Authentication Protocol“ ist ein Authentifizierungsverfahren, bei dem Accountname und Passwort im Klartext zwischen Client und Server übertragen werden. [11](#), [31](#), [32](#)
- PPP** Das „Point-to-Point-Protocol“ ist ein Netzwerkprotokoll zum Verbindungsaufbau über Einwählverbindungen, wie zum Beispiel DSL. [7](#)
- PXE** „Preboot Execution Environment“ ist eine von Intel entwickelte Technologie, die es Clients ermöglicht von einem Bootimage zu starten, welches über das Netzwerk von einem Server bezogen wird. [49](#), [51](#)
- Rainbow Table** Dies ist eine vorberechnete Liste kryptografischer Hashsummen, die zum knacken von Passworthashes verwendet wird. [20](#)
- RFC** „Requests for Comments“ sind von der Internet Engineering Task Force (IETF) herausgegebene Internet Standards. Der erste RFC wurde 1969 veröffentlicht. Zum Zeitpunkt des Schreibens dieser Arbeit wurden über 8000 veröffentlicht. [7–15](#), [19](#), [21](#)
- TLS** „Transport Layer Security“ ist ein Protokoll zur Transportverschlüsselung. Die Vorgängerversionen von TLS heißen SSL („Secure Sockets Layer“). [10](#), [11](#), [21](#), [33](#), [34](#), [42](#), [52](#)
- VLAN** Ein „Virtual LAN“ ist ein logisches Teilnetz innerhalb eines Netzwerks. Durch VLANs lassen sich physisch miteinander verbundene Geräte auf logischer Ebene trennen. [18](#), [23](#), [24](#), [47](#), [49](#), [51](#), [53](#)
- VPN** Mit einem „Virtual Private Network“ wird zwischen getrennten Netzwerken ein (verschlüsselter) Tunnel aufgebaut. Für die Geräte in diesen Netzwerken ist dadurch nicht mehr erkennbar, dass keine direkte physische Verbindung zwischen den Netzwerken besteht. [12](#)
- Wörterbuchangriff** Bei einem Wörterbuchangriff wird versucht ein unbekanntes Passwort mit Hilfe einer Passwortliste zu knacken. [21](#)

## Literatur

- [Abo+04] Bernard Aboba u. a. *Extensible Authentication Protocol (EAP)*. RFC 3748. Juni 2004, S. 3–8, 12–13, 20–25, 29–31, 41. 67 S. DOI: [10.17487/RFC3748](https://doi.org/10.17487/RFC3748). URL: <https://rfc-editor.org/info/rfc3748>.
- [Ara18] Aradial Technologies Ltd. Corp. *Aradial RADIUS Product Overview*. Alpharetta, USA, Jan. 2018, S. 5, 9, 12–13. 29 S. URL: <https://www.aradial.com/Downloads/Aradial-Overview.pdf>.
- [BT07] Florent Bersani und Hannes Tschofenig. *The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method*. RFC 4764. Jan. 2007, S. 4–5, 46, 49–50. 64 S. DOI: [10.17487/RFC4764](https://doi.org/10.17487/RFC4764). URL: <https://rfc-editor.org/info/rfc4764>.
- [BV98] Larry J. Blunk und John R. Vollbrecht. *PPP Extensible Authentication Protocol (EAP)*. RFC 2284. März 1998, S. 11. 15 S. DOI: [10.17487/RFC2284](https://doi.org/10.17487/RFC2284). URL: <https://rfc-editor.org/info/rfc2284>.
- [Cis11] Cisco Systems Inc. *MAC Authentication Bypass Deployment Guide*. San Jose, USA, Mai 2011, S. 18. 23 S. URL: [https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/identity-based-networking-services/config%5C\\_guide\\_c17-663759.pdf](https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/identity-based-networking-services/config%5C_guide_c17-663759.pdf).
- [Cis14] Cisco Systems Inc. *RADIUS Configuration Guide, Cisco IOS Release 15M&T*. San Jose, USA, Jan. 2014, S. 49–56, 114. 192 S. URL: [https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec\\_usr\\_rad/configuration/15-mt/sec-usr-rad-15-mt-book.pdf](https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_usr_rad/configuration/15-mt/sec-usr-rad-15-mt-book.pdf).
- [Cis17] Cisco Systems Inc. *Cisco 110 Series Unmanaged Switches*. San Jose, USA, März 2017, S. 1. 7 S. URL: <https://www.cisco.com/c/en/us/products/collateral/switches/110-series-unmanaged-switches/datasheet-c78-734450.pdf>.
- [Cis18a] Cisco Systems Inc. *Cisco Catalyst 2960-L Series Switches Data Sheet*. San Jose, USA, Okt. 2018, S. 15. 22 S. URL: [https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-2960-l-series-switches/data\\_sheet-c78-737665.pdf](https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-2960-l-series-switches/data_sheet-c78-737665.pdf).
- [Cis18b] Cisco Systems Inc. *Cisco Catalyst 3750-X and 3560-X Series Switches Data Sheet*. San Jose, USA, Juli 2018, S. 30. 46 S. URL: [https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3560-x-series-switches/data\\_sheet\\_c78-584733.pdf](https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3560-x-series-switches/data_sheet_c78-584733.pdf).
- [Cis18c] Cisco Systems Inc. *Cisco Catalyst 3850 Series Switches Data Sheet*. San Jose, USA, Nov. 2018, S. 34. 46 S. URL: [https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3850-series-switches/datasheet\\_c78-720918.pdf](https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3850-series-switches/datasheet_c78-720918.pdf).

- [Cud10] Arran Cudbard-Bell. *Diagram showing protocols involved in wired 802.1X authentication*. 2010. URL: [https://commons.wikimedia.org/wiki/File:802.1X\\_wired\\_protocols.png](https://commons.wikimedia.org/wiki/File:802.1X_wired_protocols.png) (besucht am 08.12.2018).
- [Cud18] Arran Cudbard-Bell. *FreeRADIUS. Active Directory Integration*. 2018. URL: <https://wiki.freeradius.org/guide/freeradius-active-directory-integration-howto> (besucht am 10.12.2018).
- [Doc18] Docker Inc. *Docker Docs. Get Started, Part 1: Orientation and setup*. 2018. URL: <https://docs.docker.com/get-started/> (besucht am 10.12.2018).
- [Dro97] Ralf Droms. *Dynamic Host Configuration Protocol*. RFC 2131. März 1997, S. 2–3. 45 S. DOI: [10.17487/RFC2131](https://doi.org/10.17487/RFC2131). URL: <https://rfc-editor.org/info/rfc2131>.
- [Eur16] Europäische Kommission. *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung)*. 2016. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- [FB08] Paul Funk und Simon Blake-Wilson. *Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TLSv0)*. RFC 5281. Aug. 2008, S. 14, 23, 27–33. 51 S. DOI: [10.17487/RFC5281](https://doi.org/10.17487/RFC5281). URL: <https://rfc-editor.org/info/rfc5281>.
- [Fre18a] FreeRADIUS Server Project and Contributors. *About FreeRADIUS: Security Issues Updates*. 2018. URL: <https://freeradius.org/security/> (besucht am 01.12.2018).
- [Fre18b] FreeRADIUS Server Project and Contributors. *FreeRADIUS. Release Notes*. 2018. URL: <https://freeradius.org/security/> (besucht am 01.12.2018).
- [Int18a] Internet Assigned Numbers Authority (IANA). *Extensible Authentication Protocol (EAP) Registry*. 2018. URL: <https://www.iana.org/assignments/eap-numbers/eap-numbers.xml> (besucht am 11.11.2018).
- [Int18b] Internet Assigned Numbers Authority (IANA). *RADIUS Types*. 2018. URL: <https://www.iana.org/assignments/radius-types/radius-types.xhtml> (besucht am 19.11.2018).
- [Kap18a] Kaplan Bilisim ve Yazilim Ticaret Ltd. *KaplanSoft Web Store*. 2018. URL: <https://www.kaplansoft.com/buy/> (besucht am 01.12.2018).
- [Kap18b] Kaplan Bilisim ve Yazilim Ticaret Ltd. *TekRADIUS - RADIUS Server for Windows - Version 5.4 - Datasheet*. Istanbul, Türkei, Aug. 2018, S. 1–2. 2 S. URL: <https://www.kaplansoft.com/tekradius/Docs/TekRADIUS-Datasheet.pdf>.

- [KPH07] Vivek Kamath, Ashwin Palekar und Ryan Hurst. *Microsoft EAP CHAP Extensions*. Internet-Draft draft-kamath-pppext-eap-mschapv2-02. Work in Progress. Internet Engineering Task Force, Juni 2007, S. 2, 17–18. 26 S. URL: <https://datatracker.ietf.org/doc/html/draft-kamath-pppext-eap-mschapv2-02>.
- [KPW02] Vivek Kamath, Ashwin Palekar und Mark Wodrich. *Microsoft's PEAP version 0 (Implementation in Windows XP SP1)*. Internet-Draft draft-kamath-pppext-peapv0-00. Work in Progress. Internet Engineering Task Force, Okt. 2002, S. 3. 19 S. URL: <https://datatracker.ietf.org/doc/html/draft-kamath-pppext-peapv0-00>.
- [LAN10] LAN/MAN Standards Committee of the IEEE Computer Society. „IEEE Standard for Local and metropolitan area networks—Port-Based Network Access Control“. In: *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)* (Feb. 2010), S. 2, 6, 9. DOI: [10.1109/IEEESTD.2010.5409813](https://doi.org/10.1109/IEEESTD.2010.5409813).
- [Lie18] Oliver Liebel. *Skalierbare Container-Infrastrukturen. Das Handbuch für Administratoren*. 1. Aufl. Rheinwerk Verlag, 2018, S. 247. 1380 S. ISBN: 978-3-8362-6385-6.
- [Lin17] Linux man pages Project. *Linux Programmer's Manual. RANDOM(4)*. Sep. 2017. URL: <http://man7.org/linux/man-pages/man4/random.4.html>.
- [LX12] Fanbao Liu und Tao Xie. „How to Break EAP-MD5“. In: *Proceedings of the 6th IFIP WG 11.2 International Conference on Information Security Theory and Practice: Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems*. WISTP'12. Egham, UK: Springer-Verlag, 2012, S. 49–57. ISBN: 978-3-642-30954-0. DOI: [10.1007/978-3-642-30955-7\\_6](https://doi.org/10.1007/978-3-642-30955-7_6). URL: [https://link.springer.com/chapter/10.1007%5C%2F978-3-642-30955-7\\_6](https://link.springer.com/chapter/10.1007%5C%2F978-3-642-30955-7_6).
- [Mal13] Jouni Malinen. *wpa\_supplicant. Linux WPA/WPA2/IEEE 802.1X Supplicant*. 2013. URL: [https://w1.fi/wpa\\_supplicant/](https://w1.fi/wpa_supplicant/) (besucht am 06. 12. 2018).
- [Mic12] Microsoft Corporation. *The Microsoft Extensible Authentication Protocol-Message Digest 5 (EAP-MD5) implementation is being deprecated from versions of Windows*. März 2012. URL: <https://support.microsoft.com/en-us/help/922574/the-microsoft-extensible-authentication-protocol-message-digest-5-eap> (besucht am 02. 12. 2018).
- [Mic18] Microsoft Corporation. *Windows Desktop Documentation: NP-Allowed-EAP-Type*. 2018. URL: <https://docs.microsoft.com/en-us/windows/desktop/nps/sdo-np-allowed-eap-type> (besucht am 02. 12. 2018).
- [Net14a] Network RADIUS SARL. *The FreeRADIUS Implementation Guide. Chapter 6 - EAP Authentication*. Grenoble, Frankreich, März 2014, S. 2–3. 10 S. URL: <https://networkradius.com/doc/FreeRADIUS-Technical-Guide.pdf>.

- [Net14b] Network RADIUS SARL. *The FreeRADIUS Technical Guide*. Grenoble, Frankreich, März 2014, S. 3, 19. 58 S. URL: <https://networkradius.com/doc/FreeRADIUS-Technical-Guide.pdf>.
- [Net18a] Network RADIUS SARL. *FreeRADIUS Documentation. Virtual Servers*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/sites-available/home.html> (besucht am 09.12.2018).
- [Net18b] Network RADIUS SARL. *FreeRADIUS Documentation. Configuration File Syntax*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/syntax/home.html> (besucht am 08.12.2018).
- [Net18c] Network RADIUS SARL. *FreeRADIUS Documentation. Radiusd*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/radiusd.html> (besucht am 08.12.2018).
- [Net18d] Network RADIUS SARL. *FreeRADIUS Documentation. Virtual Servers: default*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/sites-available/default.html> (besucht am 09.12.2018).
- [Net18e] Network RADIUS SARL. *FreeRADIUS Documentation. Modules: rlm\_mschap*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/mods-available/mschap.html> (besucht am 09.12.2018).
- [Net18f] Network RADIUS SARL. *FreeRADIUS Documentation. Modules: rlm\_pap*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/mods-available/pap.html> (besucht am 10.12.2018).
- [Net18g] Network RADIUS SARL. *FreeRADIUS Documentation. Modules: rlm\_eap*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/mods-available/eap.html> (besucht am 12.12.2018).
- [Net18h] Network RADIUS SARL. *FreeRADIUS Documentation. Modules: rlm\_eap\_tls*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/mods-available/eap/tls.html> (besucht am 12.12.2018).
- [Net18i] Network RADIUS SARL. *FreeRADIUS Documentation. Modules: rlm\_ldap*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/mods-available/ldap.html> (besucht am 13.12.2018).
- [Net18j] Network RADIUS SARL. *FreeRADIUS Documentation. Client Definitions*. 2018. URL: <https://networkradius.com/doc/3.0.10/raddb/clients.html> (besucht am 13.12.2018).
- [NT94] B. Clifford Neuman und Theodore Ts'o. „Kerberos: An authentication service for computer networks“. In: *IEEE Communications magazine* 32.9 (1994), S. 33–38. DOI: [10.1109/35.312841](https://doi.org/10.1109/35.312841). URL: <https://ieeexplore.ieee.org/document/312841>.
- [Ope10] Open1X Project. *Open1X. Home*. 2010. URL: <http://open1x.sourceforge.net/> (besucht am 06.12.2018).

- [Poz08] Sergey Poznyakoff. *GNU Radius Reference Manual*. GNU Press. Boston, USA, Dez. 2008, S. 71–74, 83. 242 S. URL: <https://www.gnu.org/software/radius/manual/radius.pdf>.
- [Res18] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018, S. 26–27. 160 S. DOI: [10.17487/RFC8446](https://doi.org/10.17487/RFC8446). URL: <https://rfc-editor.org/info/rfc8446>.
- [Rig+00] Carl Rigney u. a. *Remote Authentication Dial In User Service (RADIUS)*. RFC 2865. Juni 2000, S. 8, 13–23. 76 S. DOI: [10.17487/RFC2865](https://doi.org/10.17487/RFC2865). URL: <https://rfc-editor.org/info/rfc2865>.
- [Rig00] Carl Rigney. *RADIUS Accounting*. RFC 2866. Juni 2000, S. 4. 28 S. DOI: [10.17487/RFC2866](https://doi.org/10.17487/RFC2866). URL: <https://rfc-editor.org/info/rfc2866>.
- [SAH08] Dan Simon, Bernard Aboba und Ryan Hurst. *The EAP-TLS Authentication Protocol*. RFC 5216. März 2008, S. 2–3. 34 S. DOI: [10.17487/RFC5216](https://doi.org/10.17487/RFC5216). URL: <https://rfc-editor.org/info/rfc5216>.
- [Sam18a] Samba Team. *nmbd (8). NetBIOS name server to provide NetBIOS over IP naming services to clients*. 2018. URL: <https://www.samba.org/samba/docs/current/man-html/nmbd.8.html>.
- [Sam18b] Samba Team. *smbd (8). Server to provide SMB/CIFS services to clients*. 2018. URL: <https://www.samba.org/samba/docs/current/man-html/smbd.8.html>.
- [Sam18c] Samba Team. *Think Samba. What is Samba?* 2018. URL: [https://www.samba.org/samba/what\\_is\\_samba.html](https://www.samba.org/samba/what_is_samba.html) (besucht am 11. 12. 2018).
- [Sam18d] Samba Team. *winbindd (8). Name Service Switch daemon for resolving names from NT servers*. 2018. URL: <https://www.samba.org/samba/docs/current/man-html/winbindd.8.html>.
- [Sim96] William Allen Simpson. *PPP Challenge Handshake Authentication Protocol (CHAP)*. RFC 1994. Aug. 1996, S. 2. 13 S. DOI: [10.17487/RFC1994](https://doi.org/10.17487/RFC1994). URL: <https://rfc-editor.org/info/rfc1994>.
- [SKH18] Dominik Schöner, Arne Koschel und Felix Heine. „Teaching Microservices in the Private Cloud by Example of the eduDScloud“. In: *Service Computation 2018 - The Tenth International Conferences on Advanced Service Computing*. 2018, S. 36–39. ISBN: 978-1-61208-606-4. URL: <https://www.iaria.org/conferences2018/SERVICECOMPUTATION18.html>.
- [Smi18] Aaron Smith. *FreeRADIUS. Certificate Compatibility*. 2018. URL: <https://wiki.freeradius.org/guide/Certificate-Compatibility> (besucht am 12. 12. 2018).

- [SMW99] Bruce Schneier, Mudge und David Wagner. „Cryptanalysis of Microsoft’s PPTP Authentication Extensions (MS-CHAPv2)“. In: *Proceedings of the International Exhibition and Congress on Secure Networking - CQRE (Secure) '99*. Berlin, Heidelberg: Springer-Verlag, 1999, S. 192–203. ISBN: 3-540-66800-4. DOI: [10.1145/288090.288119](https://doi.org/10.1145/288090.288119). URL: <https://dl.acm.org/citation.cfm?doid=288090.288119>.
- [XPe18a] XPerience Technologies. *ClearBox Enterprise RADIUS Server: Main Features*. 2018. URL: <http://xperiencetech.com/products/radius/features-full.asp> (besucht am 01. 12. 2018).
- [XPe18b] XPerience Technologies. *ClearBox Enterprise RADIUS Server: Pricing*. 2018. URL: <http://xperiencetech.com/purchase/pricing.asp> (besucht am 01. 12. 2018).
- [XPe18c] XPerience Technologies. *ClearBox Enterprise RADIUS Server: System Requirements*. 2018. URL: <http://xperiencetech.com/products/radius/requirements.asp> (besucht am 01. 12. 2018).
- [XPe18d] XPerience Technologies. *ClearBox Server Products*. 2018. URL: <http://xperiencetech.com/> (besucht am 01. 12. 2018).
- [Zab18] Andrew Zaborowski. *IWD Documentation. Using IWD with Network Manager*. 2018. URL: <https://iwd.wiki.kernel.org/networkmanager> (besucht am 06. 12. 2018).



**HOCHSCHULE  
HANNOVER**  
UNIVERSITY OF  
APPLIED SCIENCES  
AND ARTS

*Fakultät IV  
Wirtschaft und  
Informatik*

**Fakultät IV  
Abteilung Informatik**

# **Dokumentation**

## **Installations- und Konfigurationshandbuch**

### **Server: RADIUS-01, RADIUS-02**

Hochschule Hannover  
University of Applied Sciences and Arts  
Fakultät IV. Abt. Informatik

Autor: **Simon Rose**

Hannover, den 20. Dezember 2018

## Versionen, Änderungen und Bemerkungen

Datum	Name	Änderung
20. Dezember 2018	Simon Rose	Dokument angelegt

**Anmerkung:** Diese Dokumentation beschreibt die Server **RADIUS-01** und **RADIUS-02**. Beide Server sind nahezu identisch konfiguriert. Auf Unterschiede wird im Text explizit eingegangen, indem die Server als **RADIUS-01** und **RADIUS-02** bezeichnet werden. Wird stattdessen nur von „den RADIUS-Servern“ gesprochen, wird diese Konfiguration auf beiden Servern gleichermaßen vorgenommen.

## Inhaltsverzeichnis

<b>1</b>	<b>Allgemeine Informationen</b>	<b>65</b>
1.1	Server . . . . .	65
1.2	Accounts . . . . .	65
<b>2</b>	<b>Betriebssystemkonfiguration</b>	<b>66</b>
2.1	Installation . . . . .	66
2.2	Netzwerkkonfiguration . . . . .	67
<b>3</b>	<b>Domänenbeitritt</b>	<b>67</b>
3.1	Installation der benötigten Pakete . . . . .	67
3.2	Konfiguration . . . . .	68
3.3	Beitritt . . . . .	68
<b>4</b>	<b>FreeRADIUS-Server</b>	<b>69</b>
4.1	Installation . . . . .	69
4.2	Konfiguration . . . . .	69
4.2.1	Test mit lokalem Account . . . . .	69
4.2.2	Hinzufügen von Clients . . . . .	70
4.2.3	Zertifikatsbasierte Authentifizierung . . . . .	71
4.2.4	Authentifizierung von Active-Directory-Accounts . . . . .	74
4.2.5	Entfernung nicht benötigter Module . . . . .	75
4.2.6	Autostart . . . . .	76
4.3	Administration . . . . .	76
4.3.1	Erneuern von Zertifikaten . . . . .	76
4.3.2	Verwaltung des Dienstes . . . . .	77
4.3.3	Aktualisieren des Servers . . . . .	78
<b>5</b>	<b>Switche</b>	<b>78</b>
<b>6</b>	<b>macOS 802.1X Profil</b>	<b>79</b>

---

# 1 Allgemeine Informationen

## 1.1 Server

Die RADIUS-Server befinden sich im **Inform**-Subnetz und stellen die in Tabelle 1 dargestellten Dienste bereit.

Nr.	Dienst	Beschreibung
1	RADIUS	Authentifizierung und Autorisierung für 802.1X

Tabelle 1: Server RADIUS-01: Bereitgestellte Dienste

Die Möglichkeiten um auf **RADIUS-01** zuzugreifen, sind in der Tabelle 2 aufgelistet.

Nr.	Typ	Adresse
1	IP	141.71.31.41
2	DNS-Name	radius01.inform.hs-hannover.de

Tabelle 2: Server RADIUS-01: Zugriffsmöglichkeiten

Die Möglichkeiten um auf **RADIUS-02** zuzugreifen, sind in der Tabelle 3 aufgelistet.

Nr.	Typ	Adresse
1	IP	141.71.31.42
2	DNS-Name	radius02.inform.hs-hannover.de

Tabelle 3: Server RADIUS-02: Zugriffsmöglichkeiten

## 1.2 Accounts

Um die RADIUS-Server zu administrieren existiert ein Account (vgl. Tabelle 4). Die Administration ist dabei entweder lokal oder über SSH möglich.

Lfd. Nr.	Accountname	Passwort
1	root	*****
Passwörter sind beim IT-Team der Abteilung Informatik erhältlich		

Tabelle 4: RADIUS-Server: Zugangsdaten

## 2 Betriebssystemkonfiguration

### 2.1 Installation

Auf beiden Servern ist das Betriebssystem **Debian** in der **Version 9 (Stretch)** installiert. Als Installationsimage wurde das **Non-Free-Netinstall-Image** verwendet, um die Nachinstallation von proprietären Treibern zu vermeiden. Folgende Einstellungen wurden während der Installation gewählt.

1. Language: **English**
2. Location: **other** → **Europe** → **Germany**
3. Locale: **en\_US.UTF-8**
4. Keymap: **German**
5. Hostname:
  - Server 1: **radius01**
  - Server 2: **radius02**
6. Domain name: **inform.hs-hannover.de**
7. Root password: **\*\*\*\*\***
8. Full name of new user: **srose**
  - Username for new account: **srose**
  - Password for new user: **—**
9. Partition disks: **Guided - use entire Disk and set up LVM**
  - Partitioning Scheme: **All files in one partition**
10. Debian archive mirror: **ftp.de.debian.org**
11. HTTP proxy: *nichts*
12. Choose software to install:
  - **SSH Server**
  - **standard system utilities**

## 2.2 Netzwerkkonfiguration

Die Netzwerkeinstellungen werden in der Datei `/etc/network/interfaces` vorgenommen. Vor der Bearbeitung wird eine Kopie des Originalzustands der Datei angelegt.

```
sudo cp /etc/network/interfaces{,.bak}
sudoedit /etc/network/interfaces
```

Auf dem Server **RADIUS-01** wird das Interface `enp0s8` wie folgt konfiguriert:

```
# The host only network interface
allow-hotplug enp0s8
    iface enp0s8 inet static
        address 141.71.31.41
        netmask 255.255.254.0
        gateway 141.71.31.254
        dns-nameservers 141.71.30.1
```

Auf dem Server **RADIUS-02** wird das Interface `enp0s8` wie folgt konfiguriert:

```
# The host only network interface
allow-hotplug enp0s8
    iface enp0s8 inet static
        address 141.71.31.42
        netmask 255.255.254.0
        gateway 141.71.31.254
        dns-nameservers 141.71.30.1
```

## 3 Domänenbeitritt

Damit *FreeRADIUS* die Accounts im *Active Directory* authentifizieren kann, müssen die RADIUS-Server der Domäne beitreten.

### 3.1 Installation der benötigten Pakete

Das Paket `samba` hat viele optionale Abhängigkeiten, die für unsere Zwecke nicht notwendig sind. Die Option `--no-install-recommends` verhindert die Installation unnötiger Pakete.

```
sudo apt install samba --no-install-recommends
sudo apt install winbind krb5-user samba-dsdb-modules
```

Bei der Installation des Pakets `krb5-user` wird nach dem **Default Kerberos version 5 realm** gefragt. An dieser Stelle wird `FH-H.DE` eingetragen.

### 3.2 Konfiguration

Zuerst wird die Konfiguration von *Samba* bearbeitet.

```
sudo cp /etc/samba/smb.conf{,.bak}
sudoedit /etc/samba/smb.conf
```

Unterhalb des Punktes `[global]` werden folgende Optionen geändert oder eingefügt.

```
[global]
    workgroup = FH-H
    security = ads
    realm = FH-H.DE
    ntlm auth = yes
    dedicated keytab file = /etc/krb5.keytab
    kerberos method = secrets and keytab
```

Danach wird die Konfiguration von *Kerberos* bearbeitet.

```
sudo cp /etc/krb5.conf{,.bak}
sudoedit /etc/krb5.conf
```

Unterhalb der Punkte `[realms]` und `[domain_realm]` werden folgende Optionen ergänzt. Alle anderen Einträge unterhalb dieser Optionen können entfernt werden.

```
[realms]
    FH-H.DE = {
        kdc = dc-i.inform.hs-hannover.de:88
        admin_server = dc-i.inform.hs-hannover.de:749
        default_domain = dc-i.inform.hs-hannover.de
    }

[domain_realm]
    .hs-hannover.de = FH-H.DE
    hs-hannover.de = FH-H.DE
```

Danach werden die Dienste neu gestartet.

```
sudo systemctl restart {nmbd,smbd,winbind}.service
```

### 3.3 Beitritt

Nun kann der Domänenbeitritt folgen.

```
sudo net join -U F4I-Admin
```

Danach kann geprüft werden, ob der *Winbind Daemon* Namen aus dem Active Directory auflösen kann. Eine Meldung über die fehlgeschlagene Klartext-Passwort-Authentifizierung kann ignoriert werden.

```
sudo wbinfo -a rzm-p24%passwort
challenge/response password authentication succeeded
```

---

Wenn dieser Test erfolgreich ist, muss überprüft werden, ob das Programm `ntlm_auth` Accounts aus dem Active Directory authentifizieren kann.

```
ntlm_auth --request-nt-key --domain=FH-H --username=rzm-p24 --password=passwort
NT_STATUS_OK: Success (0x0)
```

Wenn auch dieser Test erfolgreich ist, dann ist der Domänenbeitritt erfolgreich abgeschlossen und der FreeRADIUS-Server kann installiert werden.

## 4 FreeRADIUS-Server

### 4.1 Installation

Das Paket `freeradius` hat ebenfalls viele unnütze optionale Abhängigkeiten. Deshalb wird auch dieses Paket mit der Option `--no-install-recommends` installiert. Nach der Installation wird der RADIUS-Server gestoppt.

```
sudo apt install freeradius freeradius-utils --no-install-recommends
sudo systemctl stop freeradius
```

Um einem Berechtigungsproblem vorzubeugen, wird der User `freerad` der Gruppe `winbindd_priv` hinzugefügt.

```
sudo usermod -aG winbindd_priv freerad
```

Samba stellt im Ordner `/var/lib/samba/winbindd_privileged/` eine *Pipe* für die Interprozesskommunikation bereit. Da nur Mitglieder der Gruppe `winbindd_priv` aus dieser Pipe lesen können, muss der User `freerad` Teil dieser Gruppe sein, um die Ergebnisse der Authentifizierung von Accounts aus dem Active Directory zu lesen.

### 4.2 Konfiguration

Bei der Konfiguration des FreeRADIUS-Servers werden schrittweise neue Funktionen hinzugefügt und getestet. Zum Testen wird der FreeRADIUS-Server im *Debug-Modus* gestartet.

```
sudo freeradius -X
```

#### 4.2.1 Test mit lokalem Account

Um festzustellen, dass es keine grundlegenden Probleme gibt, wird zuerst die Authentifizierung lokaler Account getestet. Dazu wird die Datei `authorize` bearbeitet.

```
sudo cp /etc/freeradius/3.0/mods-config/files/authorize{,.bak}
sudoedit /etc/freeradius/3.0/mods-config/files/authorize
```

An oberster Stelle wird diese Zeile eingefügt:

```
testing Cleartext-Password := "password"
```

Nun kann die Authentifizierung getestet werden.

```
radtest testing password 127.0.0.1 0 testing123
```

`testing123` ist hierbei das *Shared Secret*, mit dem sich RADIUS-Clients beim RADIUS-Server authentifizieren müssen, um Anfragen stellen zu dürfen. Mit *RADIUS-Clients* sind nicht die Computer im Netzwerk gemeint, sondern die Geräte, die über das RADIUS-Protokoll mit dem RADIUS-Server kommunizieren. Dies sind meistens Switches oder Access-Points. In diesem Testfall ist es aber der RADIUS-Server selbst. Das Shared-Secret für 127.0.0.1 ist in der Datei `/etc/freeradius/3.0/clients.conf` festgelegt.

Eine erfolgreiche Authentifizierung erkennt man an der folgenden Ausgabe von `radtest`.

```
Sent Access-Request Id 196 from 0.0.0.0:49104 to 127.0.0.1:1812 length 77
  User-Name = "testing"
  User-Password = "password"
  NAS-IP-Address = 127.0.1.1
  NAS-Port = 0
  Message-Authenticator = 0x00
  Cleartext-Password = "password"
Received Access-Accept Id 196 from 127.0.0.1:1812 to 0.0.0.0:0 length 20
```

Auf der Seite des Servers erkennt man eine erfolgreiche Authentifizierung an folgenden Zeilen in der Debug Ausgabe von FreeRADIUS.

```
Sent Access-Accept Id 12 from 127.0.0.1:1812 to 127.0.0.1:51539 length 0
Finished request
```

Danach kann der Test-Eintrag für den lokalen Account wieder entfernt werden.

```
sudo mv /etc/freeradius/3.0/mods-config/files/authorize{.bak,}
```

### 4.2.2 Hinzufügen von Clients

Nun werden alle Switches in die Liste der autorisierten RADIUS-Clients hinzugefügt. Dazu wird die Datei `clients.conf` bearbeitet.

```
sudo cp /etc/freeradius/3.0/clients.conf{,.bak}
sudoedit /etc/freeradius/3.0/clients.conf
```

Für jeden Switch wird ein Eintrag in dem folgenden Format an das Ende der Datei angehängt.

```
client switchname {
    ipaddr = 141.71.31.45
    secret = switch123
}
```

### 4.2.3 Zertifikatsbasierte Authentifizierung

FreeRADIUS bringt eigene Skripte mit, mit denen selbstsignierte Zertifikate für den RADIUS-Server und die Supplicants erstellt werden. Um Kompatibilitätsprobleme zu vermeiden, sollten diese Skripte verwendet werden. Versuche mit vom IT-Team ausgestellten Zertifikaten sind fehlgeschlagen.

Die Erstellung der Zertifikate findet **ausschließlich** auf dem Server **RADIUS-01** statt. Nach der Erstellung werden die notwendigen Zertifikate und Schlüssel auf den Server **RADIUS-02** kopiert.

Zuerst müssen die drei Dateien `ca.cnf`, `server.cnf` und `client.cnf` auf dem Server **RADIUS-01** bearbeitet werden, die sich im Ordner `/etc/freeradius/3.0/certs/` befinden. Es ist sehr wichtig, dass die Passwörter in den Dateien **keine Sonderzeichen** enthalten, da die Art und Weise, wie die Konfigurationsdateien bei der Erstellung der Zertifikate eingelesen werden sehr, fragil ist.

**ca.cnf:** Dies ist die Konfigurationsdatei für das CA-Zertifikat.

```
sudo cp /etc/freeradius/3.0/certs/ca.cnf{,.bak}
sudoedit /etc/freeradius/3.0/certs/ca.cnf
```

Zeilen die mit `#` beginnen, werden auskommentiert oder gelöscht. Die anderen Zeilen werden wie folgt abgeändert:

```
[CA_default]
#crl_dir           = $dir/crl
#crl               = $dir/crl.pem
default_days      = 3650
#default_crl_days = 30
#crlDistributionPoints = URI:http://www.example.org/example_ca.crl

[req]
#input_password   = whatever
output_password   = *****

[certificate_authority]
countryName       = DE
stateOrProvinceName = Niedersachsen
localityName      = Hannover
organizationName  = Hochschule Hannover
emailAddress      = F4-I-IT-Team@hs-hannover.de
commonName        = HSH 802.1X Certificate Authority

[v3_ca]
#crlDistributionPoints = URI:http://www.example.org/example_ca.crl
```

**server.cnf:** Dies ist die Konfigurationsdatei für das Server-Zertifikat.

```
sudo cp /etc/freeradius/3.0/certs/server.cnf{,.bak}
sudoedit /etc/freeradius/3.0/certs/server.cnf
```

Zeilen die mit # beginnen, werden auskommentiert oder gelöscht. Die anderen Zeilen werden wie folgt abgeändert:

```
[CA_default]
#crl_dir           = $dir/crl
#crl               = $dir/crl.pem
default_days      = 365
#default_crl_days = 30

[req]
#input_password   = whatever
output_password   = *****

[certificate_authority]
countryName       = DE
stateOrProvinceName = Niedersachsen
localityName      = Hannover
organizationName  = Hochschule Hannover
emailAddress      = F4-I-IT-Team@hs-hannover.de
commonName        = HSH 802.1X RADIUS Server
```

**client.cnf:** Dies ist die Konfigurationsdatei für das Supplicant-Zertifikat.

```
sudo cp /etc/freeradius/3.0/certs/client.cnf{,.bak}
sudoedit /etc/freeradius/3.0/certs/client.cnf
```

Zeilen die mit # beginnen werden auskommentiert oder gelöscht. Die anderen Zeilen werden wie folgt abgeändert:

```
[CA_default]
#crl_dir           = $dir/crl
#crl               = $dir/crl.pem
default_days      = 365
#default_crl_days = 30

[req]
#input_password   = whatever
output_password   = *****

[certificate_authority]
countryName       = DE
stateOrProvinceName = Niedersachsen
localityName      = Hannover
organizationName  = Hochschule Hannover
emailAddress      = F4-I-IT-Team@hs-hannover.de
commonName        = HSH 802.1X Client
```

Zur Erstellung der Zertifikate wird ein *Makefile* verwendet. Dieses Makefile muss an einer Stelle an die Syntax von *OpenSSL 1.1* angepasst werden, da sonst die Erstellung der *Diffie-Hellman-Parameter* fehlschlägt.

```
sudo cp /etc/freeradius/3.0/certs/Makefile{,.bak}
sudoedit /etc/freeradius/3.0/certs/Makefile
```

In der Makefile muss die Regel `dh` angepasst werden. Diese befindet sich in der oberen Hälfte der Makefile. Zusätzlich werden die Regeln `revokeserver`, `destroyserver`, `revokeclient` und `destroyclient` am Ende der Makefile angehängt.

Der führende **Tabulator** darf beim Bearbeiten der Regeln nicht vergessen werden. `make` schlägt fehl wenn der Tabulator fehlt oder durch Leerzeichen ersetzt wird.

```
dh:
    openssl dhparam -check -text -5 -out dh $(DH_KEY_SIZE)

#####
#
# Rules for making certificate renewal easier
#
#####
revokeserver:
    openssl ca -revoke server.pem -keyfile ca.key -key $(PASSWORD_CA) -cert ca.pem
    -config ca.cnf

destroyserver:
    rm -f server.crt server.csr server.key server.p12 server.pem

revokeclient:
    openssl ca -revoke client.pem -keyfile ca.key -key $(PASSWORD_CA) -cert ca.pem
    -config ca.cnf

destroyclient:
    rm -f client.crt client.csr client.key client.p12 client.pem $(USER_NAME).pem
```

Danach können die Zertifikate erstellt werden. Dazu muss man sich im Verzeichnis `/etc/freeradius/3.0/certs/` befinden, zu dem nur der `root` User wechseln darf. Danach werden Besitzer und Gruppe der Dateien zu `freerad` geändert, um Berechtigungsprobleme zu vermeiden.

```
sudo -i
cd /etc/freeradius/3.0/certs/
make destroycerts all
chown freerad:freerad *
exit
```

Die Dateien `ca.pem`, `server.crt` und `server.key` werden nun auf den Server **RADIUS-02** kopiert.

```
sudo -i
cd /etc/freeradius/3.0/certs/
make destroycerts dh
cp ~/pfad/zu/{ca.pem,server.{crt,key}} .
chown freerad:freerad *
exit
```

Zuletzt muss der FreeRADIUS-Server noch so konfiguriert werden, dass die gerade erstellten Zertifikate genutzt werden. Dazu wird die Konfigurationsdatei `eap` bearbeitet.

```
sudo cp /etc/freeradius/3.0/mods-available/eap{,.bak}
sudoedit /etc/freeradius/3.0/mods-available/eap
```

Die folgenden Optionen werden in dieser Datei gesetzt. Die Optionen sind über die ganze Datei verteilt und stehen nicht wie hier untereinander.

```
eap {
    default_eap_type = tls
    tls-config tls-common {
        private_key_password = ***** (Server Output PW)
        private_key_file = /etc/freeradius/3.0/certs/server.key
        certificate_file = /etc/freeradius/3.0/certs/server.crt
        ca_file = /etc/freeradius/3.0/certs/ca.pem
    }
}
```

### 4.2.4 Authentifizierung von Active-Directory-Accounts

Die Vorbereitungen für die Authentifizierung von Active-Directory-Accounts wurden bereits in Abschnitt 3 durchgeführt. Damit FreeRADIUS über `ntlm_auth` mit dem Domaincontroller kommunizieren kann, muss lediglich das `mschap`-Modul konfiguriert werden.

```
sudo cp /etc/freeradius/3.0/mods-available/mschap{,.bak}
sudoedit /etc/freeradius/3.0/mods-available/mschap
```

Folgende Optionen werden in der Datei gesetzt.

```
mschap {
    ntlm_auth = "/usr/bin/ntlm_auth --request-nt-key
--require-membership-of='FH-H\F4-I-802.1X-Users'
--username=%{%{Stripped-User-Name}:-%{%{User-Name}:-None}}
--challenge=%{%{mschap:Challenge}:-00} --nt-response=%{%{mschap:NT-Response}:-00}"
}
```

### 4.2.5 Entfernung nicht benötigter Module

Da der RADIUS-Server nur EAP-TLS und PEAP-MSCHAPv2 unterstützen muss, können einige Module aus der Standardkonfiguration entfernt werden.

**default:** Als erstes wird die Datei `default` bearbeitet.

```
sudo cp /etc/freeradius/3.0/sites-available/default{,.bak}
sudoedit /etc/freeradius/3.0/sites-available/default
```

Die folgenden Module werden nicht benötigt und werden daher auskommentiert.

```
authorize {
#   chap
#   digest
#   mschap
#   files
#   pap
}

authenticate {
#   Auth-Type PAP {
#       pap
#   }

#   Auth-Type CHAP {
#       chap
#   }

#   mschap

#   digest
[...]}
}
```

**inner-tunnel:** Danach müssen in der Datei `inner-tunnel` die gleichen Änderungen vorgenommen werden.

```
sudo cp /etc/freeradius/3.0/sites-available/inner-tunnel{,.bak}
sudoedit /etc/freeradius/3.0/sites-available/inner-tunnel
```

Die folgenden Module sind überflüssig und werden auskommentiert.

```
authorize {
#   chap

#   pap
}

authenticate {
#   Auth-Type PAP {
#       pap
#   }

#   Auth-Type CHAP {
#       chap
#   }
}
```

### 4.2.6 Autostart

Nachdem alle Funktionen konfiguriert wurden, muss der FreeRADIUS-Server noch für das automatische Starten konfiguriert werden. Dazu wird der **systemd-Service** aktiviert.

```
sudo systemctl enable freeradius.service --now
```

Die Option `--now` impliziert das Starten des Services.

## 4.3 Administration

### 4.3.1 Erneuern von Zertifikaten

In Abschnitt 4.2.3 wurde die Gültigkeitsdauer der Zertifikate festgelegt:

- RADIUS-Server- und Supplicant-Zertifikat: **Ein Jahr**
- CA-Zertifikat: **Zehn Jahre**

Die Zertifikate müssen in regelmäßigen Abständen erneuert werden, da ansonsten für niemanden der Netzwerkzugriff möglich ist. Zertifikate werden nur auf dem Server **RADIUS-01** erstellt und danach auf den Server **RADIUS-02** kopiert. Zum Erneuern der Zertifikate muss man sich im Ordner `/etc/freeradius/3.0/certs/` befinden, den man nur mit erweiterten Rechten betreten kann.

```
sudo -i
cd /etc/freeradius/3.0/certs/
```

Das weitere Vorgehen unterscheidet sich je nach dem zu erneuernden Zertifikat.

**RADIUS-Server-Zertifikat:** Bevor ein neues Server-Zertifikat angelegt werden kann, muss das alte widerrufen werden, da OpenSSL sonst bei der Erstellung abbricht, da es nicht mehrere valide Zertifikate mit den gleichen Daten geben darf. Beim Widerrufen wird die Seriennummer des Zertifikats ausgegeben (**Revoking Certificate 01.**) Danach muss das Zertifikat mit der passenden Seriennummer gelöscht werden. (`rm 01.pem`)

```
make revokeserver
rm 01.pem
make destroyserver server
chown freerad:freerad *
```

Danach müssen die neu erstellten Dateien `server.key` und `server.crt` auf den Server **RADIUS-02** kopiert werden.

```
sudo -i
cd /etc/freeradius/3.0/certs/
rm server.{key,crt}
cp ~/pfad/zu/server.{key,crt} .
exit
```

**Supplicant-Zertifikat:** Die Erneuerung des Supplicant-Zertifikats funktioniert analog zu der Erneuerung des RADIUS-Server-Zertifikats.

```
make revokeclient
rm 02.pem
make destroyclient client
chown freerad:freerad *
```

Danach müssen die neu erstellten Dateien `client.key` und `.crt` auf die Pool-PCs verteilt werden.

**CA-Zertifikat:** Wenn das CA-Zertifikat erneuert werden muss, müssen natürlich auch die Zertifikate für den RADIUS-Server und die Supplicants erneuert werden. Man könnte diese alle einzeln neu erstellen, aber da das Makefile eine Regel enthält, um alle Zertifikate und Schlüssel zu löschen, ist es einfacher gleich alles neu zu erstellen.

```
make destroycerts all
chown freerad:freerad *
```

Danach werden die Dateien wie in Abschnitt 4.2.3 auf den Server **RADIUS-02** kopiert.

### 4.3.2 Verwaltung des Dienstes

Der FreeRADIUS-Server wird über **systemd** verwaltet.

- Starten: `sudo systemctl start freeradius.service`
- Stoppen: `sudo systemctl stop freeradius.service`

- Neustarten: `sudo systemctl restart freeradius.service`
- Statusüberprüfung: `systemctl status freeradius.service`
- Logs anschauen: `sudo journalctl -u freeradius.service`

### 4.3.3 Aktualisieren des Servers

Da der FreeRADIUS-Server über die Paketverwaltung von Debian installiert wurde, können auch die Sicherheitsupdates über die Paketverwaltung bezogen werden. Nach der Aktualisierung muss der Dienst neu gestartet werden (vgl. Abschnitt 4.3.2).

```
sudo apt update && sudo apt dist-upgrade
sudo systemctl restart freeradius.service
```

## 5 Switche

Dieser Abschnitt beschreibt die Konfiguration der Switche. Auf den Switchen müssen zuerst die beiden RADIUS-Server hinzugefügt werden. Die Passwörter (`key 0 *****)` müssen mit denen, die auf dem FreeRADIUS-Server konfiguriert sind, übereinstimmen.

```
(config)#aaa new-model
(config)#radius server radius01
(config-radius-server)#address ipv4 141.71.31.41 auth-port 1812 acct-port 1813
(config-radius-server)#timeout 2
(config-radius-server)#retransmit 1
(config-radius-server)#key 0 *****)
(config-radius-server)#exit
(config)#radius server radius02
(config-radius-server)#address ipv4 141.71.31.42 auth-port 1812 acct-port 1813
(config-radius-server)#timeout 2
(config-radius-server)#retransmit 1
(config-radius-server)#key 0 *****)
(config-radius-server)#exit
```

Danach muss eine RADIUS-Servergruppe erstellt werden, der die beiden Server hinzugefügt werden. Load-Balancing wird mit einer Batch-Size von zehn konfiguriert.

```
(config)#aaa group server radius radiusservers
(config-sg-radius)#server name radius01
(config-sg-radius)#server name radius02
(config-sg-radius)#load-balance method least-outstanding batch-size 10
(config-sg-radius)#exit
(config)#aaa authentication dot1x default group radius
(config)#dot1x system-auth-control
```

Nun kann 802.1X auf den Switchports aktiviert werden. Geräte, die zu diesem Zeitpunkt an den Switchports angeschlossen sind, werden *sofort* die Verbindung verlieren, wenn

---

sie nicht für 802.1X konfiguriert sind. Die folgenden Befehle müssen im `config-if` oder `config-if-range` Modus ausgeführt werden.

```
(config-if)#dot1x pae authenticator
(config-if)#authentication port-control auto
```

## 6 macOS 802.1X Profil

Für die 802.1X Konfiguration benötigen macOS Clients ein *802.1X-Profil*. Dieses Profil können Clients nur erstellen, wenn sie Zugang zum Internet haben, weswegen eine Profildatei vorbereitet und anschließend verteilt wird.

Die Profildatei wird mit dem Programm **Apple Configurator 2** erstellt, das über iTunes installiert wird.

1. Ablage ⇒ Neues Profil
2. Bereich **Allgemein** bearbeiten.
  - Name: **HSH-802.1x-Auth**
  - ID: *Autogenerated lassen*
  - Organisation: **Hochschule Hannover**
  - Beschreibung: „Dieses Profil dient zur Konfiguration der kabelgebundenen 802.1X-Authentifizierung im Netz der Abteilung Informatik der Hochschule Hannover.“
  - Nachricht zustimmen: *Leer lassen*
  - Sicherheit: **Immer**
  - Profil automatisch entfernen: **Nie**
3. Bereich **WLAN** bearbeiten.
  - Netzwerkname (SSID): **HSH-802.1x-WIRED**
  - Unsichtbares Netzwerk: **Nein**
  - Autom. verbinden: **Nein**
  - Captive Network-Erkennung deaktivieren: **Ja**
  - Proxy-Konfiguration: **Ohne**
  - Sicherheitstyp: **Beliebig (Firmenweit)**
  - **Protokolle:**
    - Akzeptierte EAP-Typen: **PEAP**

- Benutzername: **[auf Gerät festlegen]**
- Passwort bei jedem Verbinden Abfragen: **Nein**
- Passwort: **[auf Gerät festlegen]**
- Identitätszertifikat: **Nicht auswählen**
- Externe Identität: **hsh**
- TLS-Minimumversion: **Ohne**
- TLS-Maximumversion: **Ohne**
- Netzwerktyp: **Standard**
- Fast Lane QoS-Markierung: **QoS-Markierung nicht beschränken**
- **Vertrauen:**
  - Vertrauenswürdige Zertifikate: **Keine auswählen**
  - Vertrauenswürdige Serverzertifikat-Namen: **Keine auswählen**

Abschließend wird die Datei unter dem Namen HSH-802.1x-Auth.mobileconfig gespeichert.

**HOCHSCHULE  
HANNOVER**  
UNIVERSITY OF  
APPLIED SCIENCES  
AND ARTS  
–  
*Fakultät IV  
Wirtschaft und  
Informatik*

# Anleitung

802.1X Authentifizierung im kabelgebundenen Netzwerk der Abteilung Informatik

Version 1.0 — 20. Dezember 2018



# Versionsgeschichte

Ver.	Datum	Name	Änderung
1.0	20. Dezember 2018	Simon Rose	Dokument angelegt

## Inhaltsverzeichnis

<b>1</b>	<b>GNU/Linux</b>	<b>83</b>
<b>2</b>	<b>macOS</b>	<b>86</b>
<b>3</b>	<b>Microsoft Windows</b>	<b>94</b>

---

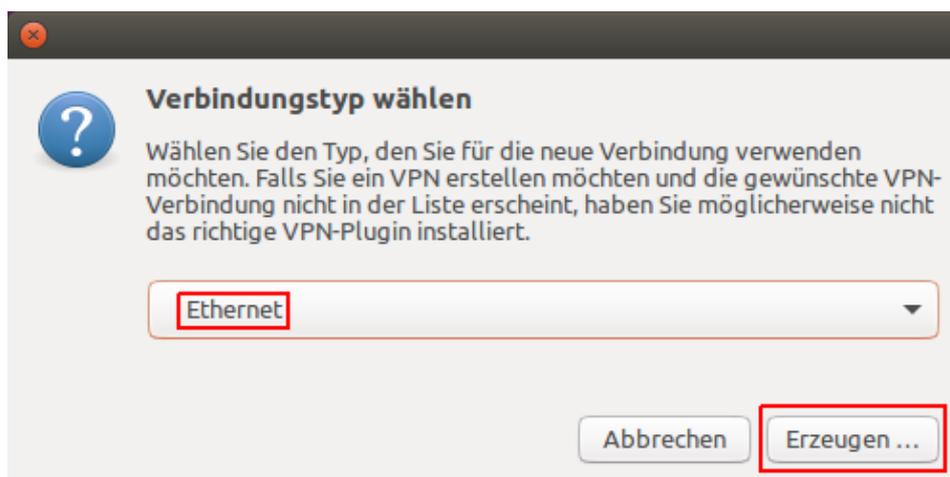
# 1 GNU/Linux

**Schritt 1:** Starten Sie das Programm **Netzwerkverbindungen**. Dies können Sie über das Anwendungsmenü oder die Kommandozeile tun. (nm-connection-editor)

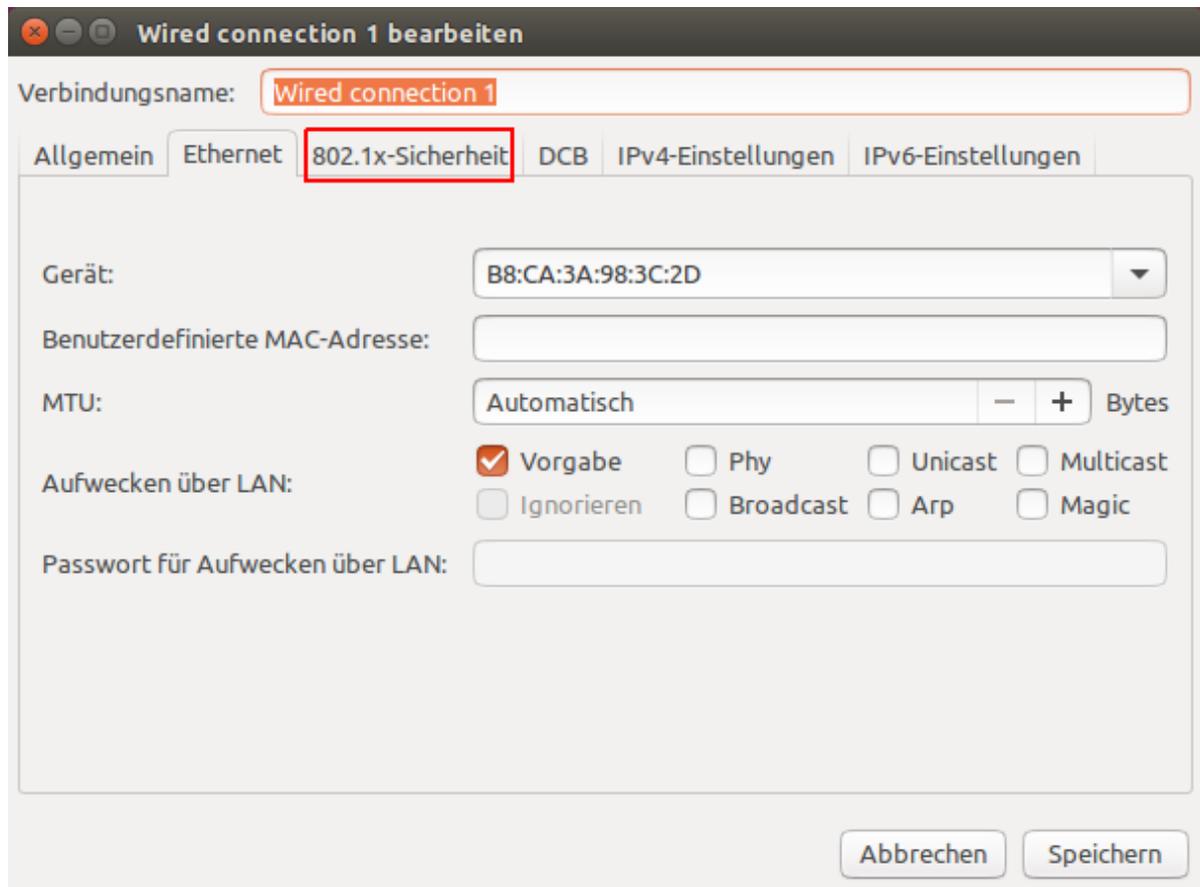
**Schritt 2:** Wählen Sie die Verbindung, die Sie im Hochschulnetz verwenden und klicken Sie auf **Bearbeiten**.



Sollten Sie bisher nur eine Verbindung haben, die Sie in mehreren Netzwerken verwenden, klicken Sie stattdessen auf **Hinzufügen** und wählen Sie den Typ **Ethernet**.

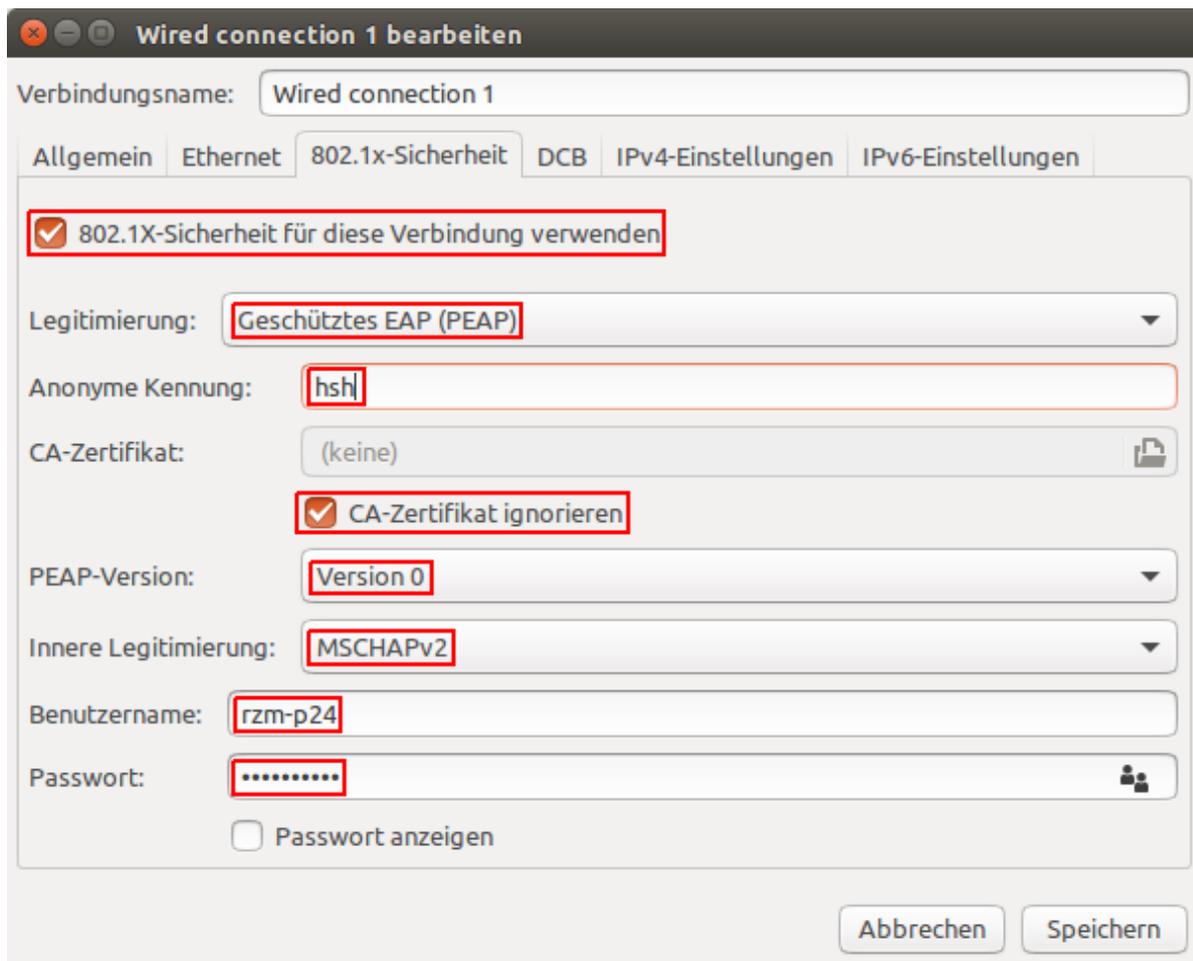


**Schritt 3:** Wählen Sie den Reiter **802.1x-Sicherheit** aus.



**Schritt 4:** Füllen Sie das Formular wie folgt aus:

1. Setzen Sie den Haken bei **802.1X-Sicherheit** für diese Verbindung verwenden.
2. Ändern Sie die Legitimierungsmethode auf **Geschütztes EAP (PEAP)**.
3. Tragen Sie unter „Anonyme Kennung“ **hsh** ein.
4. Setzen Sie den Haken bei **CA-Zertifikat ignorieren**
5. Stellen Sie sicher, dass „PEAP-Version“ auf **Version 0** und „Innere Legitimierung“ auf **MSCHAPv2** gestellt ist.
6. Tragen Sie Ihren Anmeldenamen und Ihr Passwort in die Felder **Benutzername** und **Passwort** ein. Geben Sie keine Domänenendung ein.



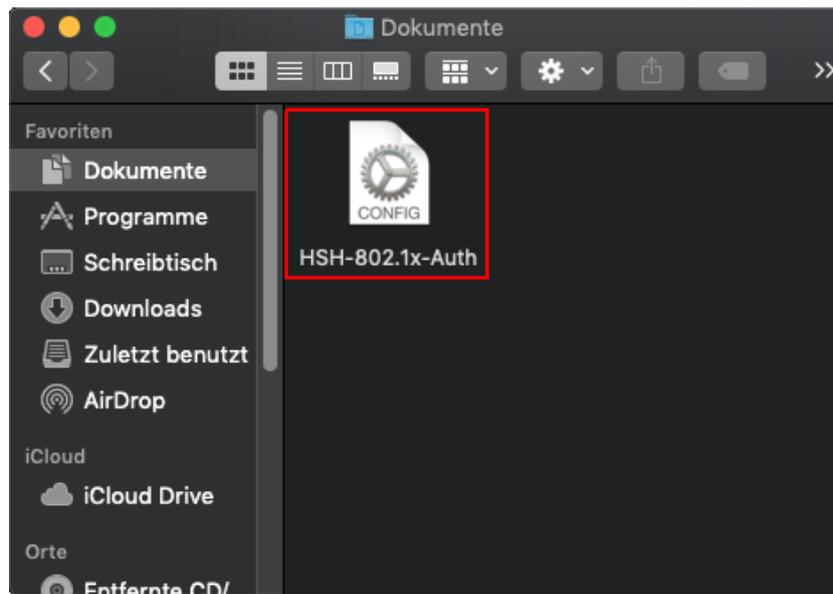
**Schritt 5:** Konfigurieren Sie die restlichen Einstellungen nach Ihren Wünschen (feste oder dynamische IP, etc.).

**Schritt 6:** Klicken Sie auf **Speichern** und verbinden Sie sich mit der bearbeiteten Verbindung.

## 2 macOS

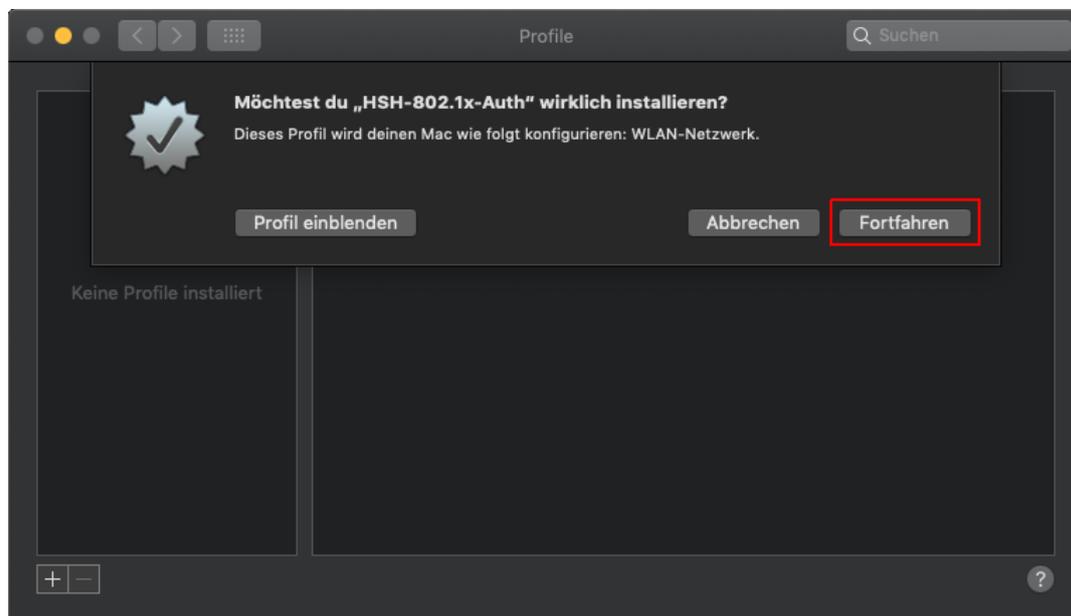
**Schritt 1:** Stellen Sie sicher das Sie die Datei HSH-802.1x-Auth.mobileconfig haben. Falls Ihnen die Datei fehlt, hilft Ihnen das IT-Team gerne weiter.

**Schritt 2:** Öffnen Sie den Ordner, in dem Sie die Profildatei gespeichert haben und klicken Sie doppelt auf die Datei.

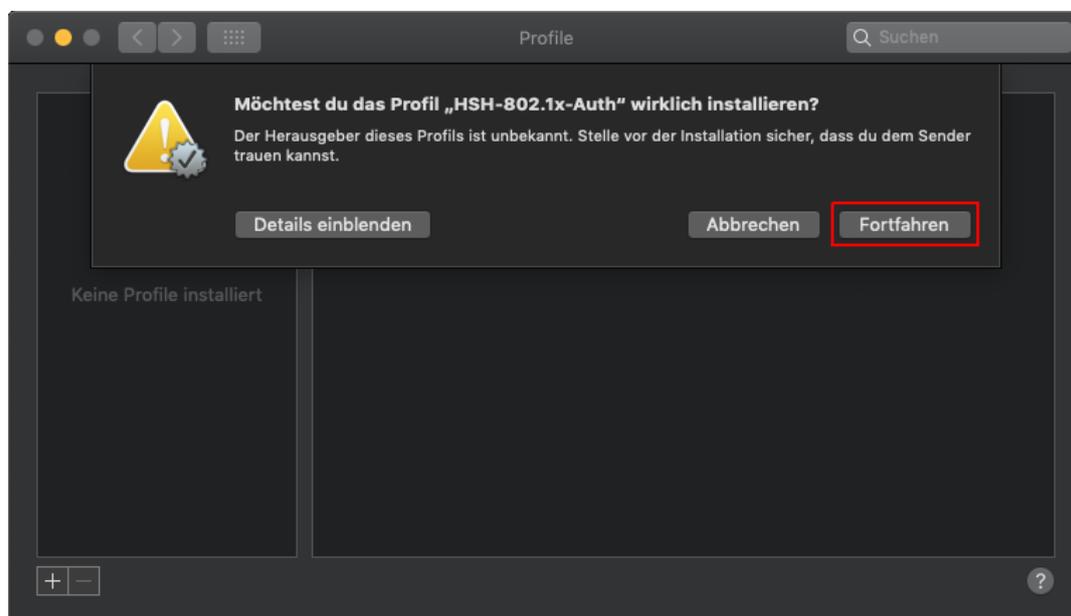


---

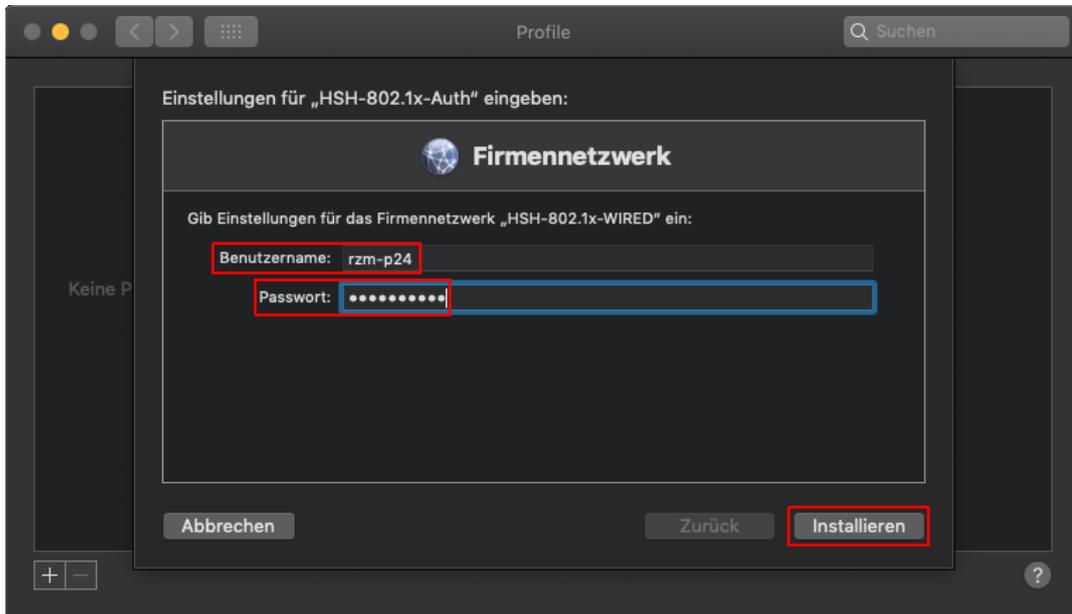
**Schritt 3:** Bestätigen Sie die Installation des Profils **HSH-802.1x-Auth** indem Sie auf **Fortfahren** klicken.



**Schritt 3:** Bestätigen Sie auch die Meldung über den Herausgeber des Profils mit **Fortfahren**.



**Schritt 4:** Tragen Sie in die Felder **Benutzername** und **Passwort** Ihre Anmeldedaten ein.



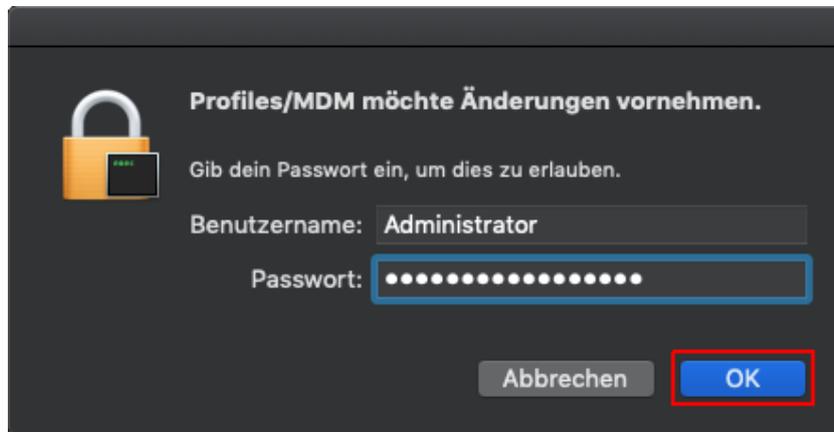
**Schritt 5:** Sollte WLAN auf Ihrem Gerät deaktiviert sein, erscheint diese Meldung. Bestätigen Sie den Dialog indem Sie auf **Installieren** klicken.

*Anmerkung:* Diese Fehlermeldung erscheint, da macOS alle Profile, die 802.1X konfigurieren, als WLAN-Profile interpretiert. Die Profile können aber auch problemlos für kabelgebundene Verbindungen genutzt werden, weshalb die Fehlermeldung ignoriert werden kann.

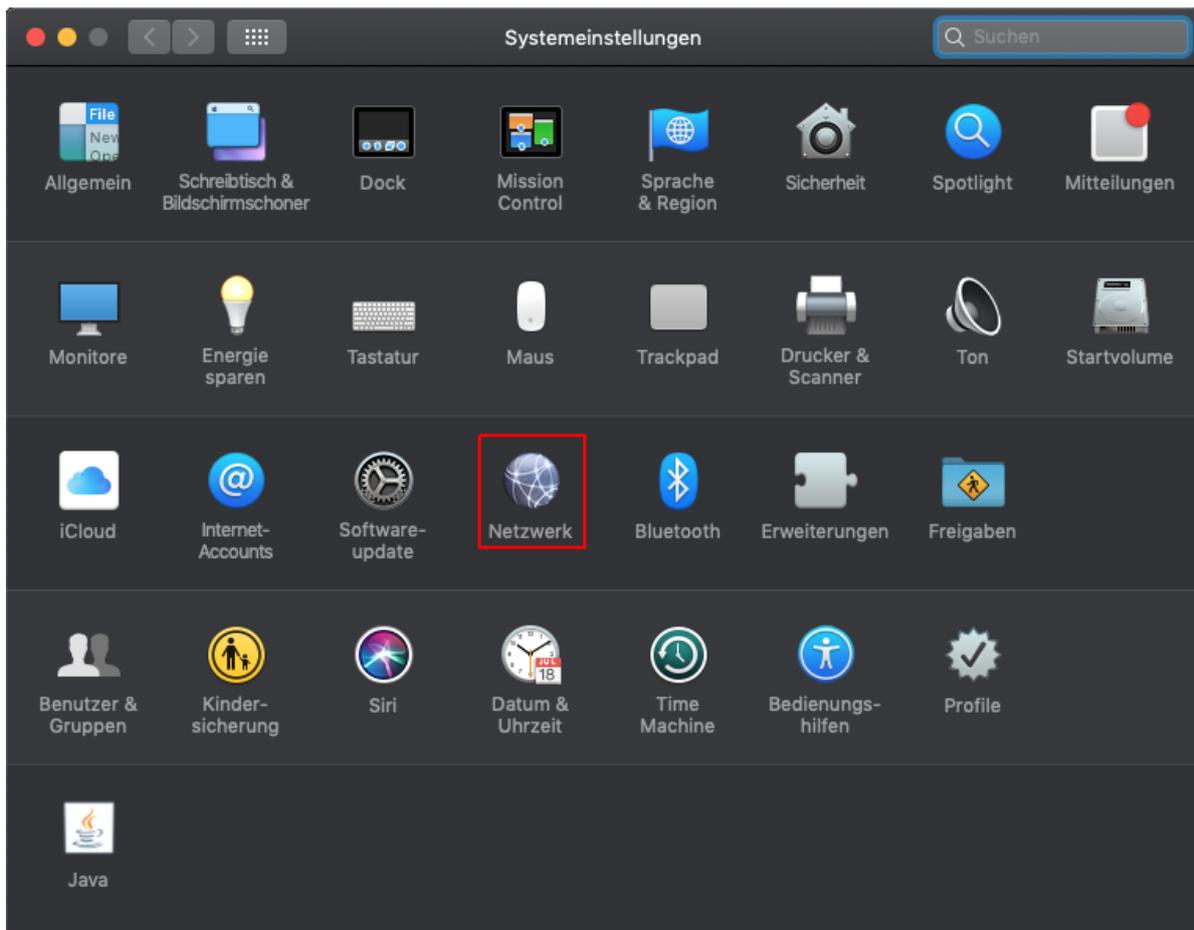


---

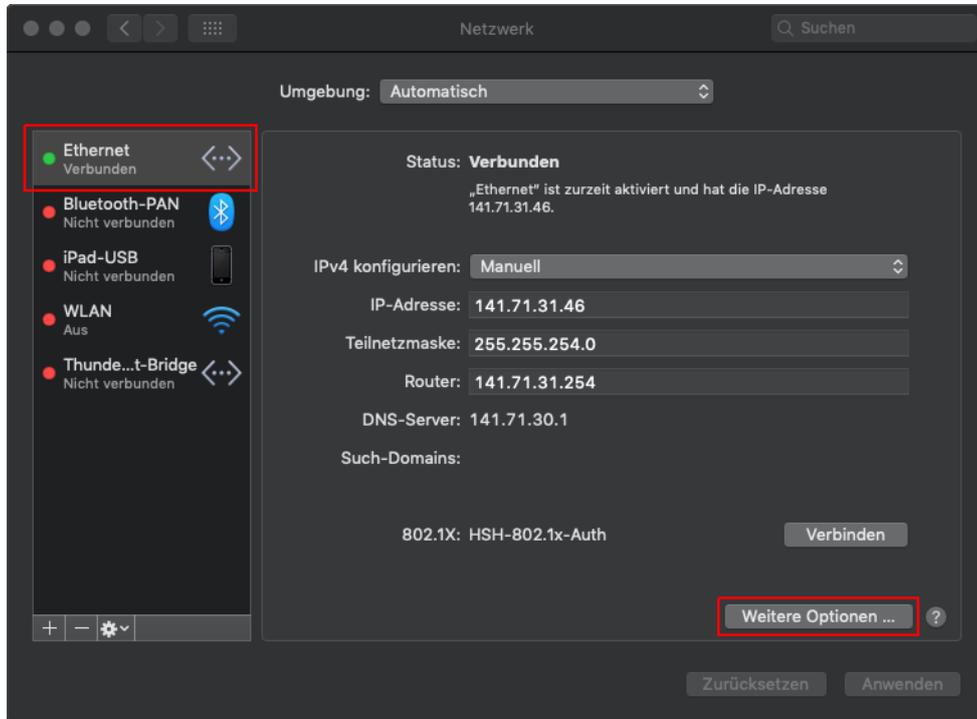
**Schritt 6:** Geben Sie Ihr Passwort ein und klicken Sie auf **OK**.



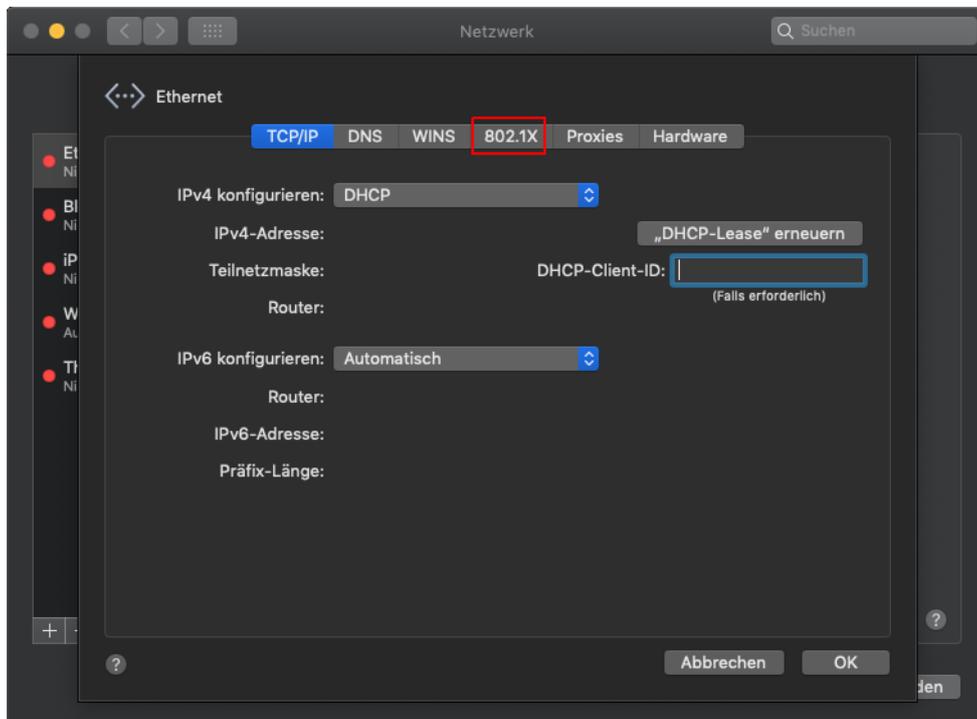
**Schritt 7:** Öffnen Sie die **Systemeinstellungen** und klicken Sie auf **Netzwerk**.



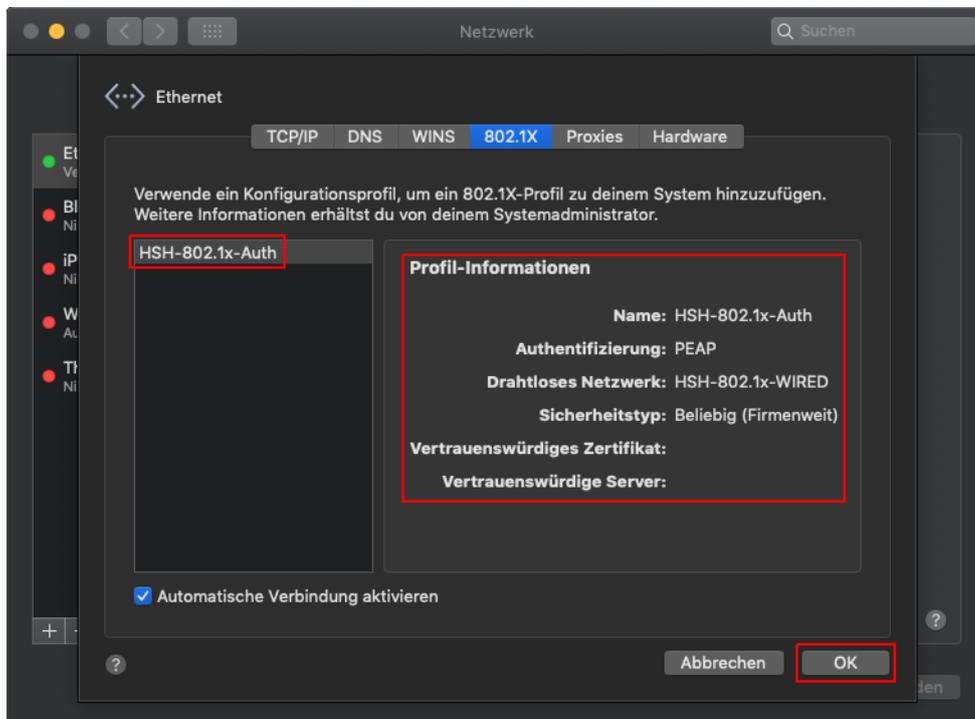
**Schritt 8:** Wählen Sie die korrekte Verbindung aus und klicken Sie auf **Weitere Optionen**.



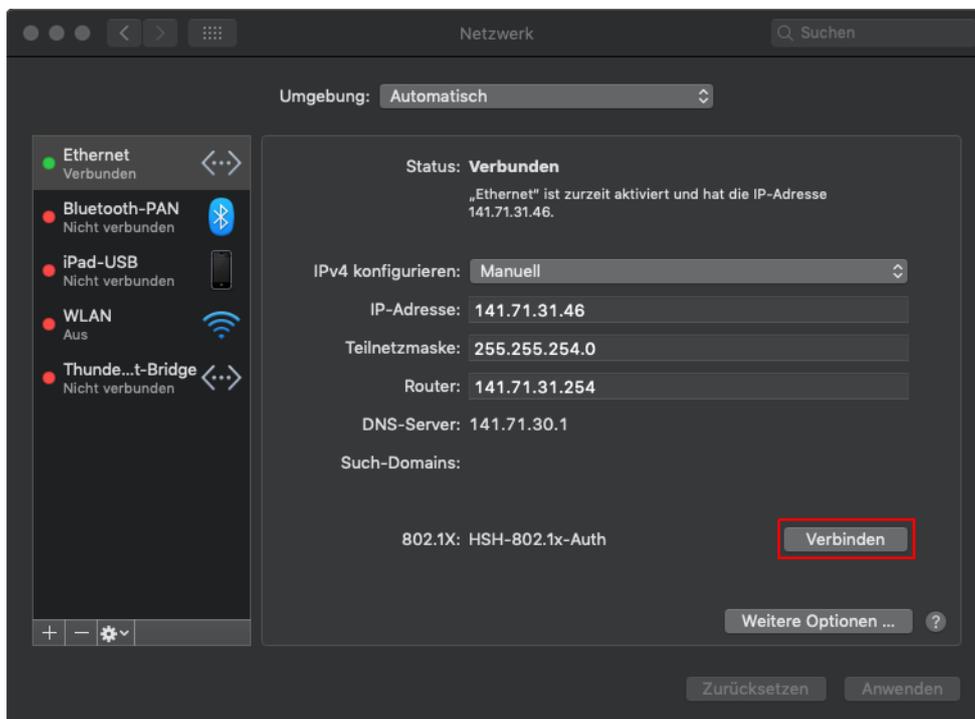
**Schritt 9:** Klicken Sie auf **802.1X**.



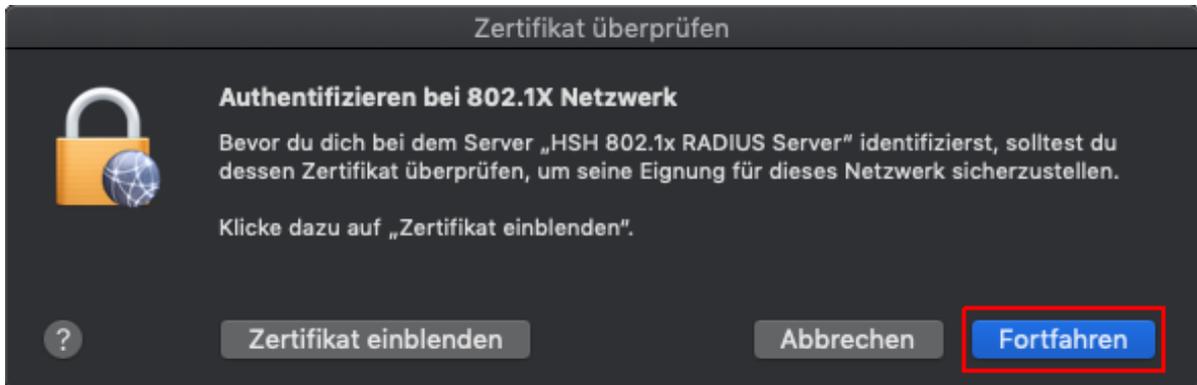
**Schritt 10:** Stellen Sie sicher, dass das Profil **HSH-802.1x-Auth** mit den gezeigten Profil-Informationen ausgewählt ist. Bestätigen Sie durch Drücken von **OK**.



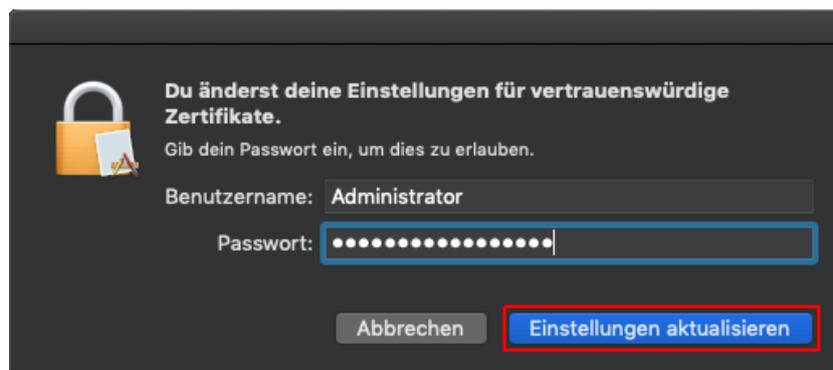
**Schritt 11:** Klicken Sie auf **Verbinden**.



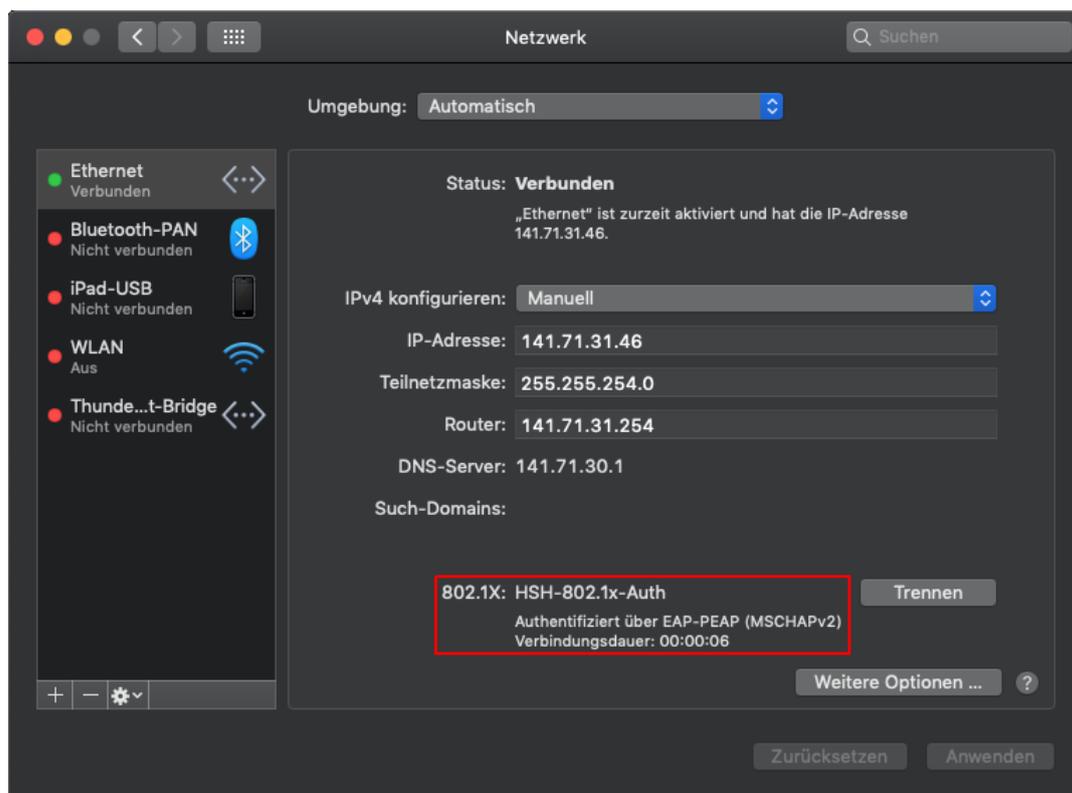
**Schritt 12:** Bestätigen Sie den Dialog „Zertifikat überprüfen“, indem Sie auf **Fortfahren** klicken.



**Schritt 13:** Geben Sie Ihr Passwort ein und klicken Sie auf **Einstellungen aktualisieren**.

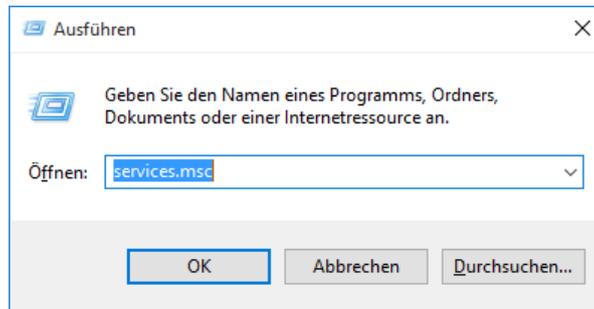


**Schritt 14:** Nach kurzer Zeit findet die Authentifizierung statt. Eine erfolgreiche Verbindung erkennen Sie an dem markierten Bereich.

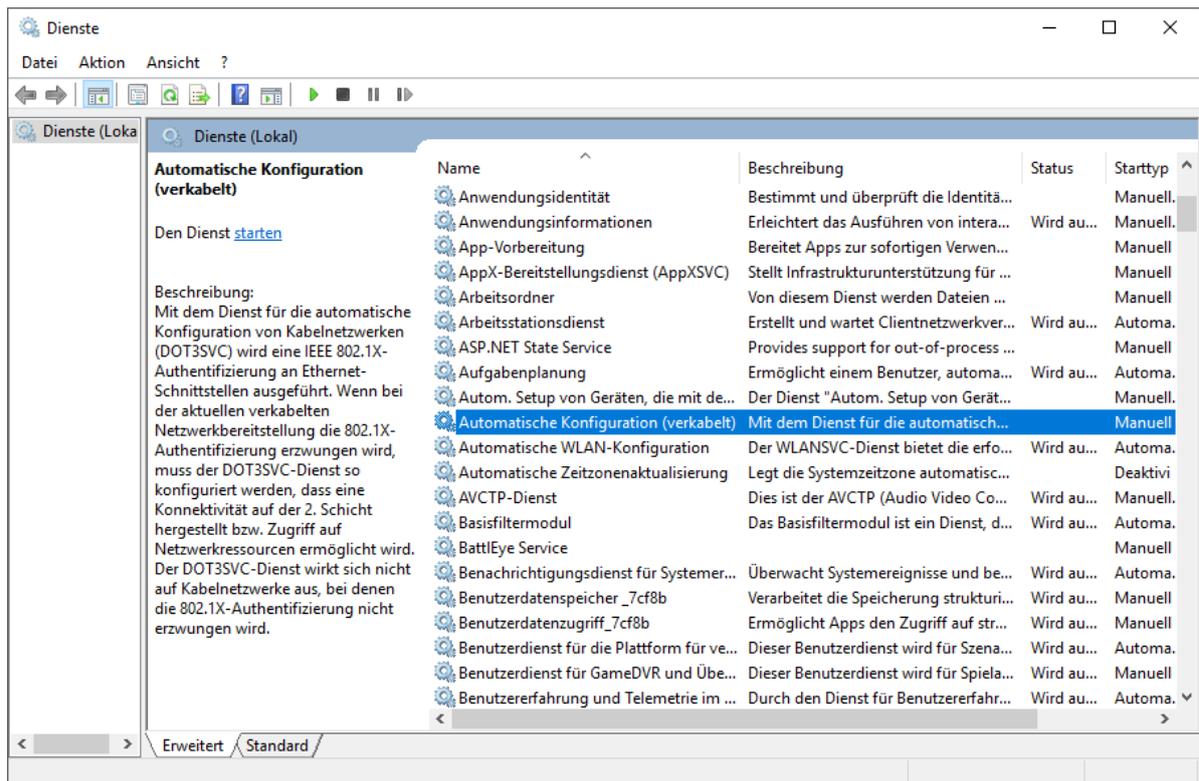


### 3 Microsoft Windows

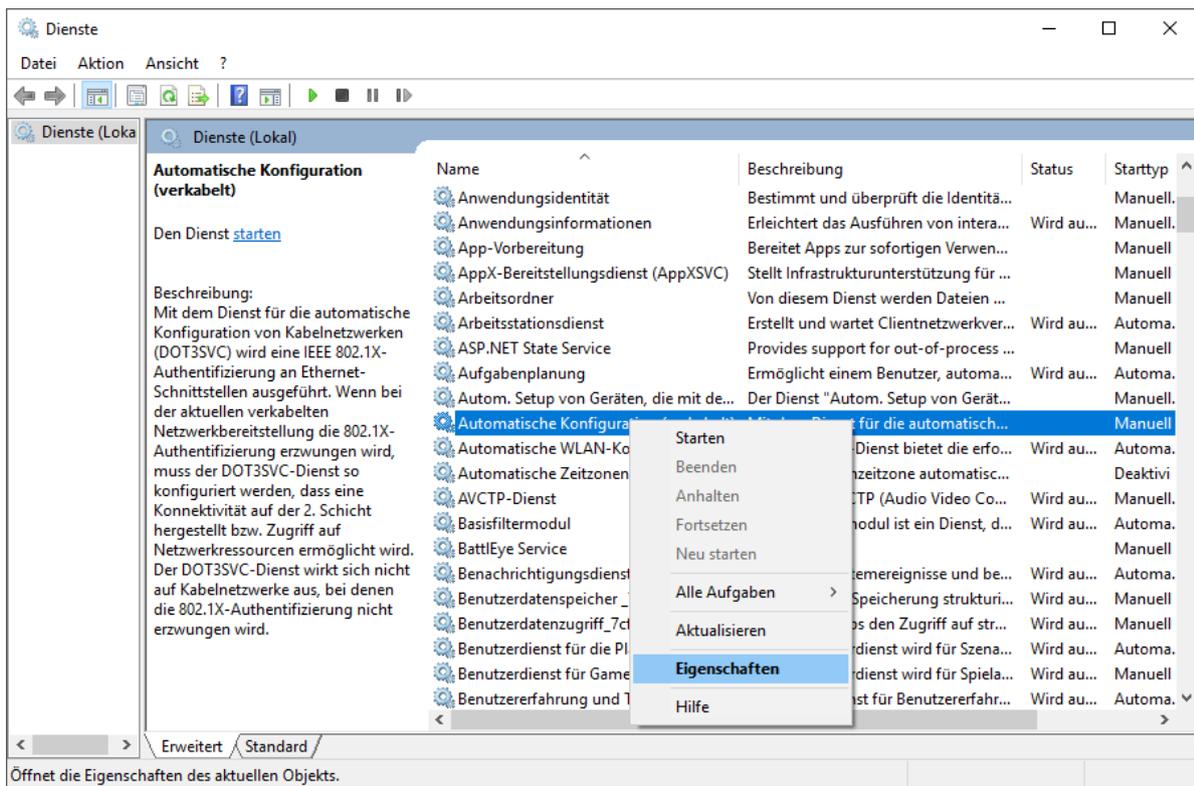
**Schritt 1:** Öffnen Sie das **Ausführen** Fenster durch Drücken von Windows+R und tippen Sie `services.msc` ein. Bestätigen Sie durch Drücken von **OK**.



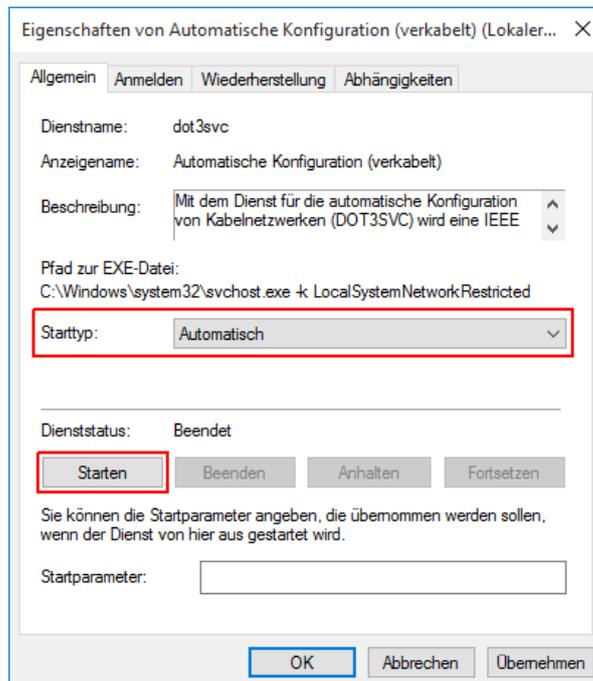
**Schritt 2:** Suchen Sie den Dienst **Automatische Konfiguration (verkabelt)**.



Klicken Sie mit der rechten Maustaste auf den Eintrag und wählen Sie **Eigenschaften** aus.

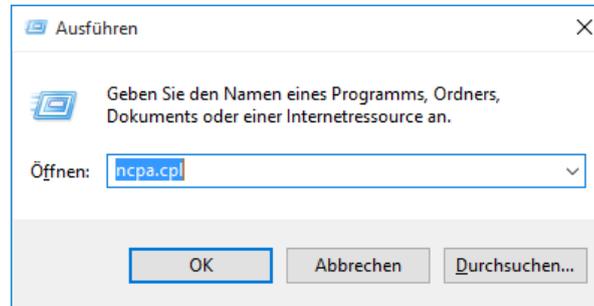


**Schritt 3:** Stellen Sie den „Starttyp“ auf **Automatisch** und starten Sie den Dienst mit einem Klick auf **Starten**.

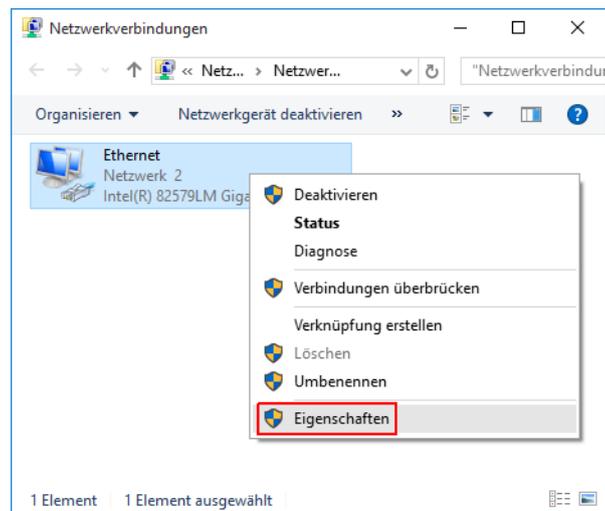


Bestätigen Sie die Eingaben, indem Sie auf **Übernehmen** und auf **OK** klicken.

**Schritt 4:** Öffnen Sie erneut das **Ausführen** Fenster durch Drücken von **Window+R** und tippen Sie `ncpa.cpl` ein. Bestätigen Sie durch Drücken von **OK**.

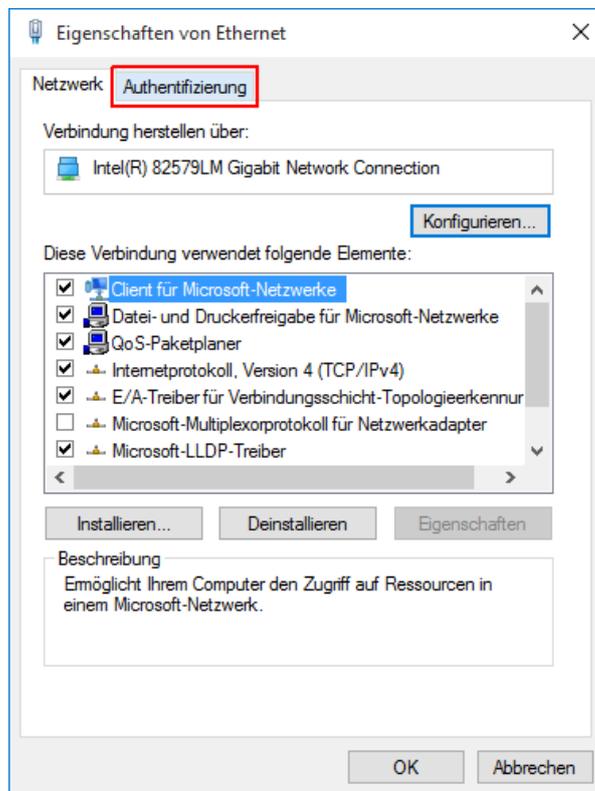


**Schritt 5:** Suchen Sie die korrekte Netzwerkkarte, klicken Sie mit der rechten Maustaste darauf und wählen Sie **Eigenschaften** aus.

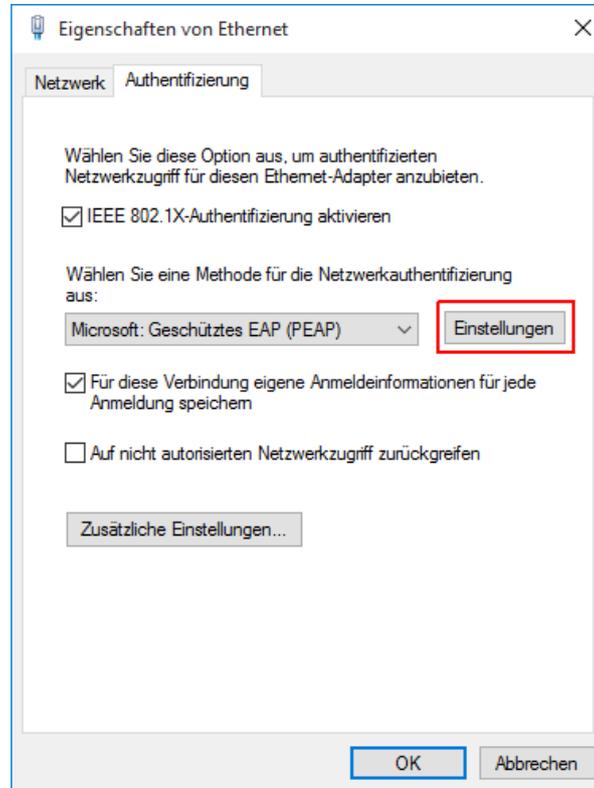


---

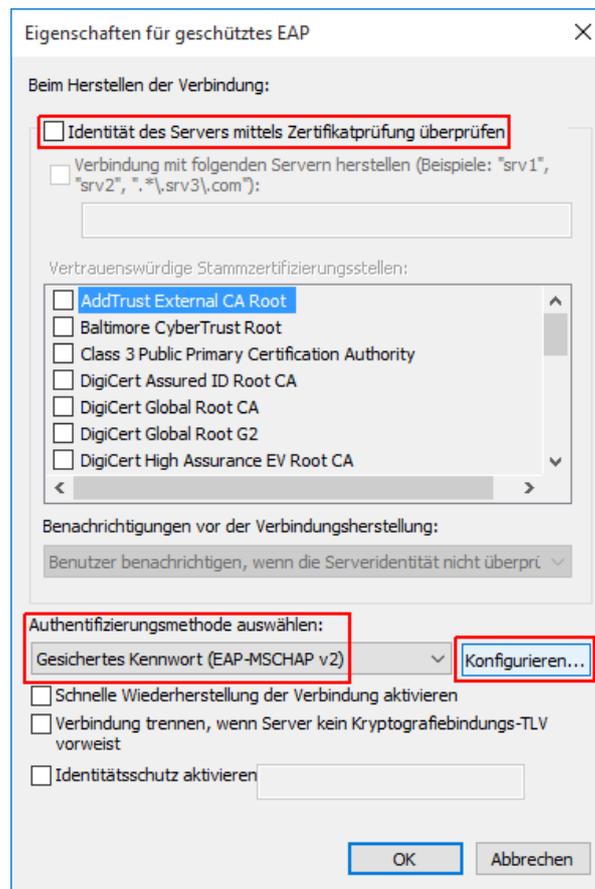
**Schritt 6:** Wählen Sie den Reiter **Authentifizierung** aus.



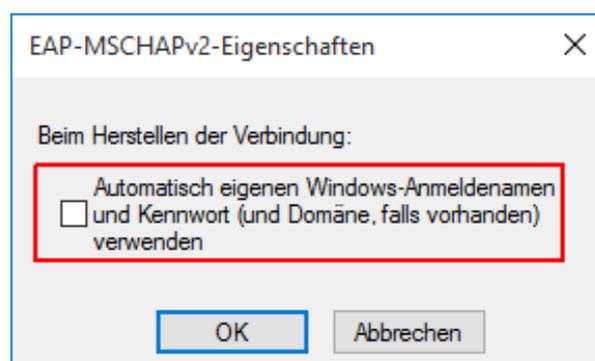
**Schritt 7:** Stellen Sie sicher, dass die Einstellungen des Reiters **Authentifizierung** wie in der folgenden Abbildung gewählt sind. Klicken Sie danach auf **Einstellungen**.



**Schritt 8:** Entfernen Sie den Haken bei **Identität des Servers mittels Zertifikatprüfung überprüfen**. Stellen Sie sicher, dass als „Authentifizierungsmethode“ **Gesichertes Kennwort (EAP-MSCHAP v2)** ausgewählt ist und dass alle Haken darunter entfernt sind. Klicken Sie danach auf **Konfigurieren**.

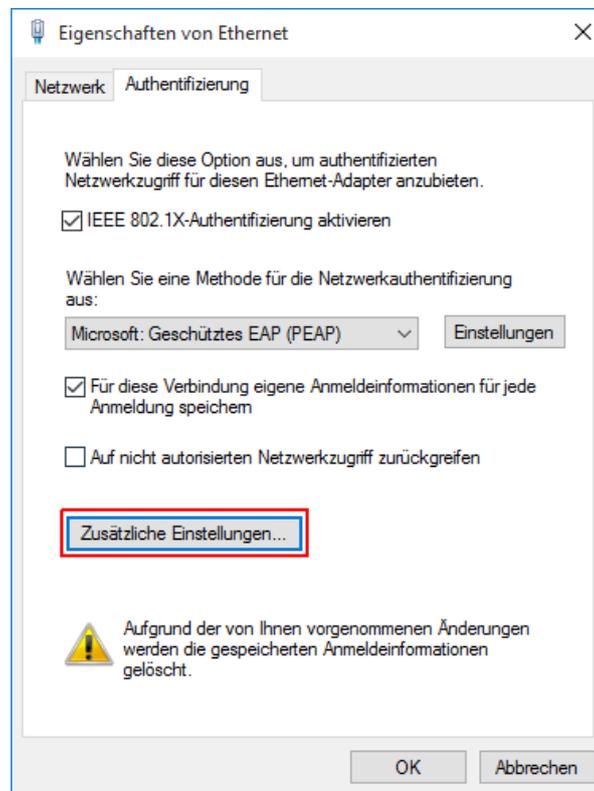


**Schritt 9:** Stellen Sie sicher, dass der Haken bei **Automatisch eigenen Windows-Anmeldenamen und Kennwort verwenden** nicht gesetzt ist. Bestätigen Sie mit einem Klick auf **OK**.

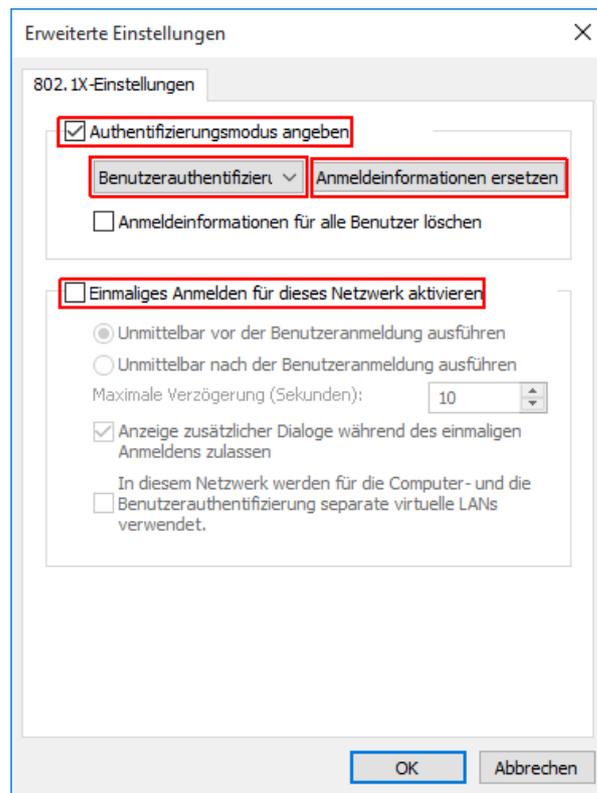


Klicken Sie danach im Fenster „Eigenschaften für geschütztes EAP“ noch einmal auf **OK**.

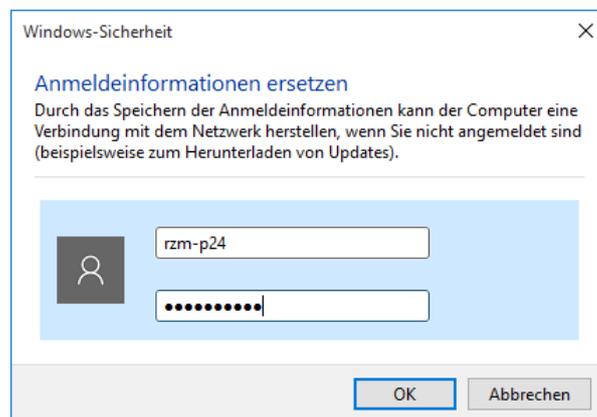
**Schritt 10:** Klicken Sie auf **Zusätzliche Einstellungen**.



**Schritt 11:** Stellen Sie sicher, dass die Option **Authentifizierungsmodus angeben** aktiviert ist und die Option **Einmaliges Anmelden für dieses Netzwerk aktivieren** nicht aktiviert ist. Wählen Sie als „Authentifizierungsmodus“ **Benutzerauthentifizierung** aus. Klicken Sie danach auf **Anmeldeinformationen ersetzen**.



**Schritt 12:** Geben Sie Ihren Anmeldenamen und Ihr Passwort ein. Hängen Sie keine Domänenendung an den Benutzernamen an.



**Schritt 13:** Konfigurieren Sie die restlichen Einstellungen der Netzwerkverbindung nach Ihren Wünschen und schließen Sie alle noch offenen Fenster.

## **C. Datenträger**

### **Inhalt des Datenträgers**

- Digitale Version dieses Dokuments
- Digitale Version der Dokumentation der Authentifizierungsserver
- Digitale Version der Konfigurationsanleitung für die 802.1X-Authentifizierung
- 802.1X-Profil für macOS
- Konfigurationsdateien von FreeRADIUS