INSTITUT FÜR INFORMATIK

DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwicklung einer XML-basierten Authentifizierungsmethode für mobile Netze der vierten Generation

Thomas Robert Hey

Aufgabensteller: Prof. Dr. Hans Jürgen Ohlbach

Betreuer: Benjamin Weyl (BMW Group Forschung und Technik)

Abgabetermin: 31. Mai 2005

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. Mai 2005

…….
(Unterschrift des Kandidaten)

Zusammenfassung

Das Ziel dieser Diplomarbeit ist die Entwicklung einer sicheren und effizienten Authentifizierungsmethode, die für den Einsatz in mobilen Netzen der vierten Generation geeignet ist. Charakteristisch für diese zukünftigen Netze wird ihre Heterogenität sein, die nicht nur durch die vielfältigen Zugangstechnologien, sondern auch durch die hohe Anzahl unterschiedlicher Netzbetreiber und Dienstanbieter zum Ausdruck kommen wird.

Eine zentrale Rolle bei der im Rahmen dieser Arbeit entwickelten Authentifizierungsmethode nimmt die Security Assertion Markup Language ein, die die Möglichkeit bietet, Sicherheitsinformationen über Domänengrenzen hinweg auszutauschen. Dadurch können Nutzer auch in fremden Domänen effizient authentifiziert werden, was in den fragmentierten Umgebungen der vierten Generation eine Grundvoraussetzung für die Nutzung mobiler Dienste sein wird.

Danksagung

An dieser Stelle möchte ich mich ganz herzlich bei allen Menschen bedanken, die direkt oder indirekt am Entstehen dieser Diplomarbeit beteiligt waren.

Mein besonderer Dank gilt Herrn Prof. Dr. Hans Jürgen Ohlbach für die Betreuung dieser Diplomarbeit.

Herrn Benjamin Weyl, meinem Betreuer bei der BMW Forschung und Technik, möchte ich für die außerordentliche Unterstützung, die hilfreichen Anregungen und die motivierende Zusammenarbeit sehr herzlich danken.

Weiterhin danke ich meinen Kollegen von der BMW Forschung und Technik, die mit ihren Beiträgen eine große Hilfe für das Gelingen dieser Diplomarbeit waren.

Darüber hinaus möchte ich mich ganz besonders bei meiner Freundin Natalie, meiner Familie und meinen Freunden bedanken, die mich in dieser Zeit unterstützt haben.

Inhaltsverzeichnis

1. Einleitung 1

1.1 Evolution mobiler Netze 1

1.2 Aufgabenstellung 3

1.3 Aufbau und Gliederung der Arbeit 4

2. Authentifizierungsprotokolle 6

2.1 Begriffsklärung 6

2.2 Das EAP Protokoll 8

2.2.1 Format von EAP Paketen 8

2.2.2 EAP Authentifizierungsmethoden 9

2.2.3 Bewertung und Einsatzmöglichkeiten 13

2.3 Das Diameter Protokoll 14

2.2.3 Das Diameter Basis-Protokoll 16

2.3.2 Diameter Zusatzprotokolle 22

2.3.3 Bewertung des Diameter Protokolls 25

2.4 Das PANA Protokoll 26

2.4.1 PANA Nachrichtenformat 27

2.4.2 Architektur und Einsatzszenario von PANA 28

2.4.3 Bewertung des PANA Protokolls 30

2.5 Zusammenfassung 30

3. XML Sicherheit 31

3.1 Einführung 32

3.2 Die SAML Spezifikation 32

3.2.1 Konzeptioneller Überblick über SAML 32

3.2.2 SAML Assertions 33

3.2.3 Das SAML Protokoll 36

3.3 Single Sign-On Authentifizierung mittels SAML 37

3.3.1 Das Artifact-Profil 38

3.3.2 Das POST-Profil 40

3.4 Absicherung von SAML-Nachrichten 41

3.4.1 Vertraulichkeit 42

3.4.2 Integrität 42

3.4.3 Authentizität 42

3.4.3 Nichtabstreitbarkeit 43

3.5 Zusammenfassung 43

4. Systementwurf **45**

4.1 Anwendungsszenarien ... 45
4.1.1 Neuanmeldung eines Nutzers ... 46
4.1.2 Reauthentifizierung in der Heimatdomäne ... 47
4.1.3 Authentifizierung in einer Partnerdomäne ... 48
4.1.4 Authentifizierung in einer Föderation ... 50

4.2 Anforderungsanalyse ... 52
4.2.1 Generisches Authentifizierungsprotokoll ... 52
4.2.2 Unterstützung unterschiedlicher Zugangstechnologien ... 53
4.2.3 Skalierbarkeit der Authentifizierung ... 53
4.2.4 Effiziente Reauthentifizierung ... 54
4.2.5 Unterstützung der Authentifizierung in Partnerdomänen ... 55
4.2.6 Unterstützung der Authentifizierung in Föderationen ... 55
4.2.7 Sicherer Datentransport ... 56

4.3 Systemarchitektur ... 57

4.4 Kommunikationsabläufe ... 59
4.4.1 Neuanmeldung eines Nutzers ... 60
4.4.2 Reauthentifizierung in der Heimatdomäne ... 62
4.4.3 Authentifizierung in einer föderierten Domäne ... 63

4.5 Zusammenfassung ... 65

5. Implementierung **66**

5.1 Vorhandene Komponenten ... 66
5.1.1 Open Diameter ... 66
5.1.2 SAML-Autorität ... 72

5.2 Implementierte Komponenten ... 74
5.2.1 Implementierung des EAP-SAML Peers und des Authentication Servers ... 75
5.2.2 Zustandsmaschinen des EAP-SAML Peers und Authentication Servers ... 77
5.2.3 Implementierung des Clients und des Authentifizierungsservers ... 81

5.3 Zusammenfassung ... 83

6. Zusammenfassung und Ausblick **84**

A. Anhang **86**

A.1 Auszüge aus dem Quellcode ... 86
A.1.1 EAP-SAML Peer und EAP-SAML Authentication Server ... 86
A.1.2 Client und Authentifizierungsserver ... 92

A.2 Zustandsdiagramme ... 96
A.2.1 Zustandsdiagramm des EAP Peers [43] ... 96
A.2.2 Zustandsdiagramm des EAP Backend Authentication Servers [43] ... 97

Literaturverzeichnis **98**

Abbildungsverzeichnis

Abbildung 1: Netzwerk der vierten Generation [10] 3
Abbildung 2: Format von EAP Paketen 8
Abbildung 3: Typisches Szenario für die Authentifizierung mittels EAP 14
Abbildung 4: Komponenten des Diameter Protokolls 15
Abbildung 5: Diameter Nachrichtenformat 17
Abbildung 6: Diameter Attribut-Wert Paar 18
Abbildung 7: Diameter Architektur 20
Abbildung 8: Nachrichtentransport ohne Ende-zu-Ende Sicherheit 22
Abbildung 9: Authentifizierung über Diameter-EAP 24
Abbildung 10: PANA Nachrichtenformat 27
Abbildung 11: PANA Szenario 29
Abbildung 12: Struktur einer SAML Assertion 34
Abbildung 13: Authentifizierung eines Nutzers mittels SAML 38
Abbildung 14: Authentifizierung gemäß Artifact-Profil 39
Abbildung 15: Authentifizierung gemäß POST-Profil 41
Abbildung 16: Neuanmeldung eines Nutzers 47
Abbildung 17: Authentifizierung in einer Partnerdomäne 49
Abbildung 18: Authentifizierung in einer Föderation 51
Abbildung 19: Systemarchitektur 59
Abbildung 20: Sequenzdiagramm - Neuanmeldung eines Nutzers 60
Abbildung 21: Sequenzdiagramm - Reauthentifizierung in der Heimatdomäne 62
Abbildung 22: Authentifizierung in einer föderierten Domäne 64
Abbildung 23: Struktur einer Diameter-Nachricht Agent [43] 69
Abbildung 24: Architektur des PANA Authentication Agent [43] 70
Abbildung 25: Architektur des PANA Client [43] 72
Abbildung 26: Fragment der SAML-Autorität 73
Abbildung 27: Klassendiagramm des EAP-SAML Peers und EAP-SAML Authentication Servers 76
Abbildung 28: Zustandsdiagramm der EapPeerEAP_SAMLStateMachine 78
Abbildung 29: Zustandsdiagramm der EapAuthEAP_SAMLStateMachine 79

1. Einleitung

In den letzten Jahren waren auf dem Gebiet der Telekommunikation zwei prägende Entwicklungen zu beobachten: zum einen der Siegeszug der Internetprotokolle in praktisch allen Teilbereichen der Netzwerktechnik und zum anderen die immer größere Verbreitung mobiler, drahtloser Kommunikationstechniken. In jüngster Zeit wird zunehmend eine Verbindung der beiden Entwicklungen angestrebt, um dadurch eine mobile Internetkommunikation zu erreichen.

Dieser Ansatz ist für die Automobilindustrie hochinteressant, da er eine Vielzahl neuer Applikationen im Fahrzeug ermöglicht. So bietet die *BMW Group* unter den Namen *BMW Assist* [1] und *BMW Online* [2] eine Reihe von Telematik-Anwendungen an, also Dienste, die online im Fahrzeug verfügbar sind. Mit BMW Assist kann beispielsweise ein Notruf ausgelöst werden, bei dem der Rettungsdienst benachrichtigt und der aktuelle Standort des Fahrzeugs automatisch übermittelt wird. Durch die niedrige Übertragungsrate, die in den aktuellen Mobilfunknetzen zur Verfügung steht, können jedoch höherwertige Dienste wie Videotelefonie oder Audiostreaming nicht angeboten werden. Diese Anwendungen benötigen große Bandbreiten, die erst in einigen Jahren durch den Mobilfunk der vierten Generation unterstützt werden sollen. Der Weg zu diesen Netzen wird im folgenden Abschnitt beschrieben.

1.1 Evolution mobiler Netze

Grundlegende und einschneidende Weiterentwicklungen in der Mobilfunktechnik werden als Generationen bezeichnet. Die erste Mobilfunkgeneration umfasste die analogen A-, B- und C-Netze, die in Deutschland von 1958 bis 2000 betrieben wurden. Analoge Netze hatten mehrere Schwächen, wie zum Beispiel eine geringe Sprachqualität und eine niedrige Kapazität von weniger als einer Million Teilnehmern. Aus diesem Grund wurde in der Europäischen Union Anfang der achtziger Jahre des letzten Jahrhunderts mit der Entwicklung einer neuen Mobilfunktechnologie begonnen. Das Ziel war ein digitales mobiles Telefonsystem, das Teilnehmern eine europaweite Mobilität erlaubte und Sprachdienste anbot, die mit analogen Telefonnetzen kompatibel waren. Die Arbeiten mündeten in den Standard *Global System for Mobile Communications (GSM)* [4], der die zweite Generation des Mobilfunks begründete. GSM wurde in Deutschland und den

meisten europäischen Ländern im Jahr 1992 eingeführt und hat seitdem sämtliche Wachstumsprognosen übertroffen: im Mai 2005 telefonierten über GSM weltweit mehr als 1,3 Milliarden Menschen [4]. GSM ermöglicht neben der Nutzung von Sprachdiensten auch die leitungs- und paketvermittelte Datenübertragung. Allerdings ist die Übertragungsrate mit 9,6 kbit/s viel zu niedrig für moderne mobile Applikationen, wie Videotelefonie oder auch nur das Surfen im Internet. Weiterentwicklungen von GSM wie *GPRS* [5] und *EDGE* [6] konnten die Datenraten nicht wesentlich steigern, weshalb Anfang der neunziger Jahre mit der Arbeit an der dritten Mobilfunkgeneration begonnen wurde. Das Ergebnis war eine Reihe konkurrierender und untereinander inkompatibler Standards, wobei sich Europa im Jahr 1999 für das *Universal Mobile Telecommunications System (UMTS)* [7] entschied. Der kommerzielle Start der dritten Mobilfunkgeneration hat sich aufgrund der hohen Lizenzgebühren und Infrastrukturkosten mehrfach verzögert, so dass in den meisten EU-Ländern UMTS erst im Jahr 2004 in Betrieb genommen wurde. Die Datenrate von UMTS konnte im Vergleich zu GSM erheblich gesteigert werden. So können im Idealfall 1920 kbit/s erreicht werden, unter normalen Umständen sinkt aber die Datenrate auf 384 kbit/s. Dieser Wert liegt zwar immer noch unter dem, was zukünftige Applikationen wie Videostreaming benötigen, allerdings ist die Forschung an der nächsten Generation im vollen Gange.

Im Jahr 2001 wurde von mehreren Mobilfunkunternehmen das *Wireless World Research Forum (WWRF)* [8] gegründet, um die Forschung für die vierte Mobilfunkgeneration zu koordinieren. Ihnen schlossen sich zahlreiche Unternehmen und Universitäten an, so dass das Konsortium mittlerweile über 150 Mitglieder zählt. In dem vom WWRF verfassten *Book of Visions* [9] präsentieren die Autoren ihre Vorstellung von der vierten Mobilfunkgeneration. Darin gehen sie von einem großen heterogenen Netzwerk aus, in dem verschiedene Technologien wie UMTS, WLAN und Bluetooth zu einem Gesamtnetz verschmelzen. Theoretisch sollen zu Beginn Datengeschwindigkeiten von bis zu 10 Mbit/s und in weiteren Ausbaustufen bis zu 100 Mbit/s erreicht werden. Diese Geschwindigkeiten werden aus wirtschaftlichen Gründen nicht im gesamten Netz zur Verfügung stehen, sondern nur in so genannten *Hot Spots*, wo die Nutzerdichte sehr hoch ist. Außerhalb von Ballungsgebieten werden Kunden dagegen UMTS nutzen oder sich ein eigenes lokales kabelloses Netzwerk errichten, das dann über DSL oder Richtfunk mit dem Provider verbunden ist. Die Abbildung 1 zeigt den prinzipiellen Aufbau eines Netzwerks der vierten Generation.

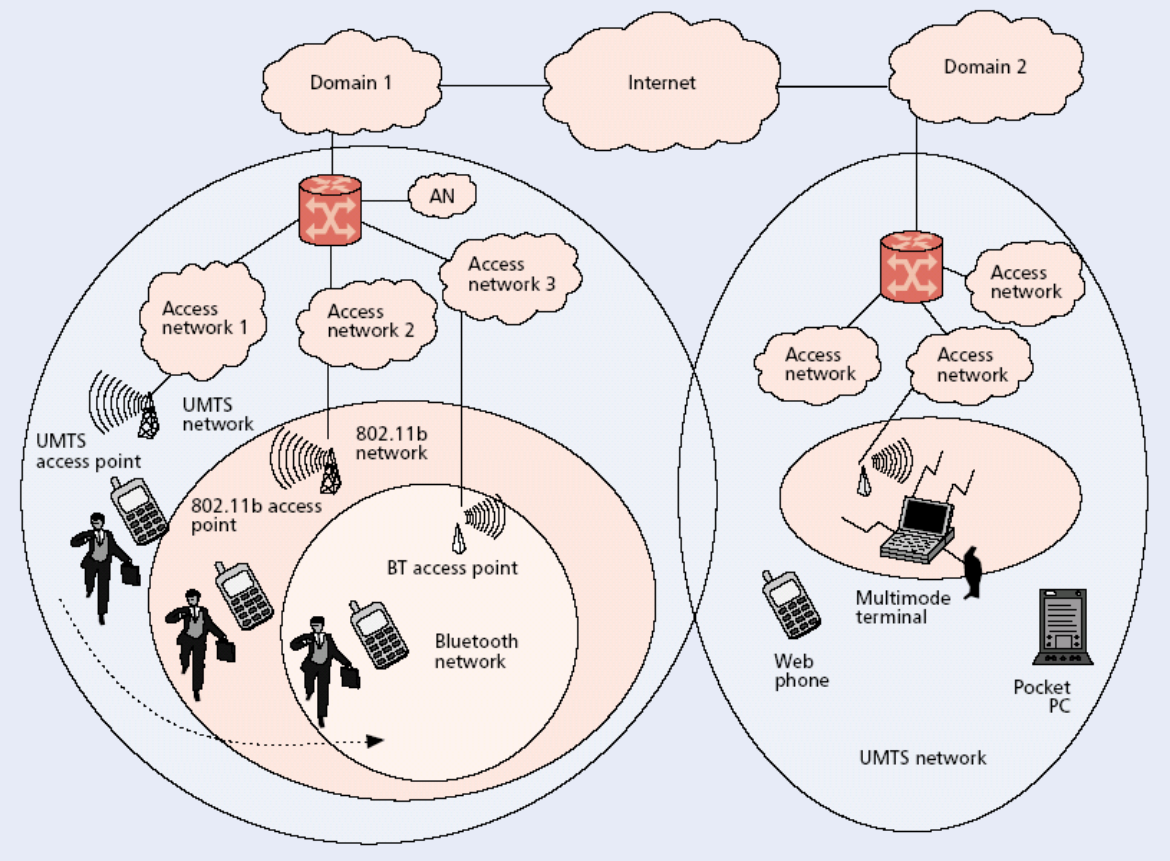


Abbildung 1: Netzwerk der vierten Generation [10]

1.2 Aufgabenstellung

In Netzen der vierten Generation werden häufig Dienste existieren, die benutzerspezifische Informationen und benutzersensible Daten verarbeiten, wie dies beispielsweise bei Online-Banking-Diensten der Fall ist. Aus diesem Grund ist die Authentifizierung der Anwender vor der Nutzung entsprechender Dienste notwendig, um sowohl Benutzer, als auch Dienstanbieter abzusichern.

Das Ziel dieser Diplomarbeit ist die Entwicklung einer sicheren und effizienten Authentifizierungsmethode, die für den Einsatz in mobilen Netzen der vierten Generation geeignet ist.

Charakteristisch für diese Netze wird ihre Heterogenität sein. Es wird ein Nebeneinander von zahlreichen Übertragungstechnologien geben. Die Netze werden von unterschiedlichen Organisationen betrieben werden, so dass der Markt der Netzbetreiber stark fragmentiert sein wird. Manche Netzbetreiber werden sich aus finanziellen Gründen oder aufgrund gesetzlicher Regulierung auf bestimmte Übertragungstechnologien und/oder

auf einzelne geografische Gebiete beschränken müssen. Somit wird ein mobiler Nutzer häufig nicht auf die Dienste seines Netzbetreibers zugreifen können und wird deshalb die Infrastruktur fremder Betreiber nutzen müssen. Um jedoch Zugang zu diesen Diensten zu erlangen, muss der Nutzer vom fremden Betreiber authentifiziert werden. Dabei müssen drei Bedingungen erfüllt sein:

- Der fremde Netzbetreiber muss Nutzer unabhängig von der verwendeten Authentifizierungsmethode authentifizieren können.
- Das Authentifizierungsprotokoll soll über alle eingesetzten Übertragungstechnologien transportiert werden können
- Da der fremde Netzbetreiber über keinerlei Authentifizierungsdaten des Nutzers verfügt, muss es möglich sein, die Authentifizierung an den Heimat-Netzbetreiber zu delegieren.

Sind die oberen Bedingungen erfüllt, kann ein Nutzer beliebig zwischen den Domänen des Netzes wechseln. Dabei fällt jedoch bei jedem Betreten einer Domäne eine neue Authentifizierung an. Ist in diesem Fall eine Interaktion des Nutzers notwendig, führt dies zu häufigen Unterbrechungen in der Dienstnutzung, was einen negativen Einfluss auf die Akzeptanz der Dienste haben wird. Um eine effiziente Authentifizierung zu gewährleisten, muss somit sichergestellt werden, dass der Authentifizierungsvorgang ohne die Interaktion des Nutzers auskommt, und stattdessen direkt zwischen Endgerät und Netzbetreiber durchgeführt wird.

Im Rahmen dieser Arbeit sollen Szenarien aufgezeigt werden, in denen eine derartige Authentifizierungsmethode notwendig ist und darauf basierend eine Architektur entworfen werden, die diese Methode unterstützt. Schließlich sollen die Komponenten dieser Architektur prototypisch implementiert werden.

1.3 Aufbau und Gliederung der Arbeit

Die vorliegende Arbeit umfasst sechs Kapitel, über die ein kurzer Überblick vermittelt werden soll.

Im ersten Kapitel wurden neben einigen Grundlagen zur Mobilfunktechnik die Motivation und das Ziel dieser Diplomarbeit vorgestellt.

Im zweiten Kapitel werden Protokolle eingeführt, die für den Transport der Authentifizierungsdaten in Netzen der vierten Generation eingesetzt werden können. Das

erste Protokoll ist das *Extensible Authentication Protocol (EAP)* [11] , das als einheitliches Protokoll für unterschiedliche Authentifizierungsmethoden benutzt werden kann und somit die Authentifizierung von Subjekten über unterschiedliche Methoden erlaubt. Als nächstes wird das Protokoll *Diameter* [12] vorgestellt, das die Authentifizierungsdaten in heterogenen Netzwerken zwischen dem Netzwerkzugangspunkt und dem Authentifizierungsserver transportieren kann. Schließlich wird das *Protocol for carrying Authentication for Network Access (PANA)* [13] eingeführt, das Authentifizierungsdaten unabhängig von der zugrunde liegenden Netzwerkzugangstechnologie zwischen Endgerät und Netzwerkzugangspunkt transportieren kann.

Das dritte Kapitel beschäftigt sich mit Sicherheitstechnologien, die auf XML basieren. Insbesondere wird auf die XML-basierte *Security Assertion Markup Language (SAML)* [14] eingegangen, die eine Möglichkeit bietet, Sicherheitsinformationen, also auch Authentifizierungsinformationen, zwischen Domänen auszutauschen. Diese Domänen können mit Hilfe von SAML Föderationen bilden, in denen Nutzer ohne eigenes Zutun authentifiziert werden können, was zu einer unterbrechungsfreien Dienstnutzung auch bei Domänenwechsel führt.

Im vierten Kapitel werden Anwendungsszenarien aufgezeigt, die in mobilen Netzen der vierten Generation eintreten können und in denen die Authentifizierungen des Nutzers notwendig ist. Aus den Szenarien werden Anforderungen an die zu entwickelnde Authentifizierungsmethode abgeleitet und eine Systemarchitektur entworfen, die die gestellten Anforderungen erfüllt.

Das fünfte Kapitel widmet sich der Implementierung der Authentifizierungsmethode und ihrer Integration in existierende Software-Komponenten.

Abschließend gibt das sechste Kapitel eine Zusammenfassung der wichtigsten Ergebnisse der Diplomarbeit wieder und endet mit einem Ausblick.

2. Authentifizierungsprotokolle

In diesem Kapitel werden Protokolle vorgestellt, die für den Transport der Authentifizierungsdaten in Netzen der vierten Generation eingesetzt werden können. Nach einer kurzen Einführung wird im Abschnitt 2.2 das EAP Protokoll eingeführt, das die Einkapselung unterschiedlicher Authentifizierungsmethoden übernimmt und so ein generisches Protokoll für den Transport von Authentifizierungsdaten auf der Sicherungsschicht darstellt. Anschließend wird im Abschnitt 2.3 das Diameter Protokoll eingeführt, das Dienste zur Authentifizierung, Autorisierung und Abrechnung anbietet und EAP Pakete innerhalb des drahtgebundenen Abschnitts eines Netzes unabhängig von dessen Infrastruktur transportieren kann. Schließlich wird im Abschnitt 2.4 das PANA-Protokoll vorgestellt, das im drahtlosen Bereich des Netzes eingesetzt wird. Aufgabe dieses Protokolls ist das Heben der EAP Authentifizierungsdaten auf die Netzwerkschicht, um damit eine einheitliche Authentifizierung für alle Zugangsarten zu ermöglichen

2.1 Begriffsklärung

In unserer von der Informationstechnologie durchdrungenen Welt kommt der IT-Sicherheit eine herausragende Bedeutung zu. Sichere Systeme sind eine Grundvoraussetzung für zahlreiche Transaktionen wie E-Commerce und E-Business. Immer häufiger auftretende Angriffe auf IT-Systeme erfordern deshalb wirksame Maßnahmen zur Verhinderung von Missbrauch und unautorisierter Manipulation. Die Authentifizierung dient hierbei als erste Barriere gegen den Zugriff unberechtigter Entitäten.

Unter *Authentifizierung* versteht man die Überprüfung der Identität eines Benutzers, Absenders oder Dienstes mit Hilfe einer vertrauenswürdigen Instanz oder Technik [15]. Der Authentifizierungsvorgang besteht aus zwei Phasen, der Identifizierung und der Verifizierung:

- Während der *Identifizierungsphase* liefert die sich authentifizierende Entität dem Sicherheitssystem einen Identifikator. Die Art des Identifikators hängt dabei von den eingesetzten Sicherheitsmechanismen ab.

- In der anschließenden *Verifizierungsphase* präsentiert die Entität einen Beleg, der die Korrektheit des Identifikators beweisen soll. Das Sicherheitssystem prüft daraufhin ob der Beleg zum Identifikator passt und gewährt gegebenenfalls Zutritt zum System.

Die gebräuchlichste Methode Authentifizierungstechniken zu kategorisieren ist die Unterteilung in biometrische, besitzbasierte und wissensbasierte Verfahren [16]:

- *Biometrie* bezeichnet die Identifikation von Personen anhand einzigartiger Körpermerkmale. Personen können anhand ihrer Fingerabdrücke, ihrer Gesichtsgeometrie, ihrer Stimme oder anderer körpereigenen Merkmalen identifiziert werden. Biometrische Verfahren gelten als sehr sicher [17] sind aber im Allgemeinen auch sehr aufwendig.
- *Besitzbasierte Verfahren* erfordern für die Authentifizierung den Einsatz eines Objekts wie z.B. eines Schlüssels, einer Magnetkarte oder einer Smartcard. Besitzbasierte Verfahren sind weniger aufwendig als biometrische, sie bieten aber auch weniger Sicherheit, da das Objekt gewollt oder ungewollt den Besitzer wechseln kann.
- *Wissensbasierte Verfahren* sind in der Praxis am häufigsten anzutreffen. Die Entität authentifiziert sich beim Sicherheitssystem durch das Vorweisen spezifischen Wissens. Dies kann in Form eines Passworts sein oder einer Antwort auf eine vorher vereinbarte Frage. Wissensbasierte Verfahren sind zwar einfach zu realisieren, aber auch anfällig gegen Attacken. Aus diesem Grund werden sie üblicherweise mittels zusätzlicher Sicherheitsmechanismen wie Verschlüsselung und digitaler Signatur verstärkt.

Zur Realisierung von Authentifizierungsmethoden werden häufig Kombinationen der oben genannten Verfahren verwendet, um auf diese Weise die Sicherheit zu erhöhen und um die Vorteile der unterschiedlichen Techniken auszunutzen. Ein Beispiel hierfür ist die Authentifizierung eines Bankkunden an einem Geldautomaten. Der Kunde beweist seine Identität durch den Besitz einer Chipkarte und zusätzlich durch die Eingabe der PIN. Dieses Beispiel deckt jedoch ein weiteres Problem bei der Authentifizierung auf: da der Kunde sich beim Automaten identifiziert, nicht aber der Automat beim Kunden, spricht man von einer einseitigen Authentifizierung. Der Kunde kann sich nicht sicher sein, dass er mit einem vertrauenswürdigen Gerät interagiert. Im schlimmsten Fall hat er seine geheimen Daten einem betrügerischen Automaten anvertraut und riskiert erheblichen finanziellen Schaden. Um solche Angriffe zu verhindern, ist eine beidseitige Authentifizierung notwendig. Das EAP Protokoll, das im Folgenden vorgestellt werden soll, bietet eine derartige Lösung.

2.2 Das EAP Protokoll

Das Extensible Authentication Protocol (EAP) [11] ist ein relativ neues Rahmenwerk für Authentifizierungsmethoden. Es wurde von der IETF-Gruppe [18] standardisiert, um damit zu verhindern, dass viele verschiedene und inkompatible Authentifizierungsprotokolle für die Sicherungsschicht entwickelt werden. Stattdessen sollte EAP die Einkapselung der unterschiedlichen Authentifizierungsmethoden übernehmen und so ein einheitliches Protokoll für den Transport von Authentifizierungsdaten auf der Sicherungsschicht darstellen.

2.2.1 Format von EAP Paketen

Da EAP nicht für die eigentliche Steuerung der Authentifizierung zuständig ist, sondern diese Aufgabe der eingekapselten Methode überlässt, sind EAP Pakete recht einfach aufgebaut. Die Abbildung 2 zeigt die einzelnen Felder eines EAP Pakets.

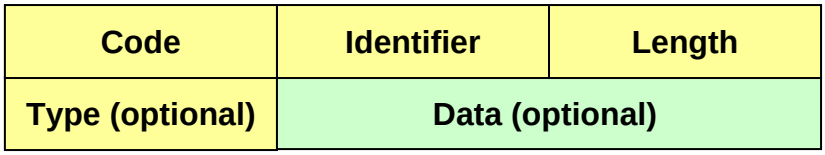


Abbildung 2: Format von EAP Paketen

- Das Feld *Code* bestimmt den Typ des Pakets und gibt somit an, wie die das Datenfeld zu interpretieren ist. Folgende Typen sind definiert: Request, Response, Success und Failure.
- Der *Identifier* erlaubt es, verschiedene Pakete einander zuzuordnen. So muss auf jeden Request ein zugehöriger Response erfolgen, wobei beide Pakete den gleichen Identifier erhalten. Falls ein Paket wiederholt werden muss, wird der Identifier des vorhergehenden Pakets übernommen.
- Im Feld *Length* wird die Länge des gesamten EAP Pakets in Bytes gespeichert. Wird ein Paket empfangen, bei dem die angegebene von der tatsächlichen Länge abweicht, muss dieses Paket abgelehnt werden.
- Die letzten beiden Felder sind optional. Sie werden nur bei Request- und Response-Paketen ausgefüllt.

- Das Feld *Type* gibt an, welche konkrete Authentifizierungsmethode im EAP Paket transportiert wird. Einige Beispiele für Authentifizierungsmethoden werden im Abschnitt 2.2.2 vorgestellt.
- Im Feld *Data* werden schließlich die eigentlichen Authentifizierungsdaten transportiert. Sie variieren abhängig von der verwendeten Authentifizierungsmethode. Das Feld kann in machen Fällen, wie zum Beispiel bei der passwortbasierten Authentifizierung, sehr klein sein. Im Falle einer Authentifizierung mittels Zertifikaten wäre im Gegensatz dazu das Feld um ein Vielfaches größer.

Wie bereits erwähnt, sind in EAP vier Pakettypen definiert. EAP-Requests werden immer vom Authentifizierer dem Nutzer gesendet, der sich anmelden möchte. Der Nutzer antwortet auf jeden EAP-Request mit einem EAP-Response, der den gleichen Identifier trägt. Zum Abschluss einer jeden Authentifizierung sendet der Authentifizierer dem Nutzer eine EAP-Success oder eine EAP-Failure Meldung.

Der Ablauf ist im Allgemeinen der folgende: Jede EAP Sitzung fängt mit einem EAP-Request an, der vom Authentifizierer an den Nutzer gesendet wird. Im Type-Feld kann der Nutzer erkennen welche Authentifizierungsmethode der Authentifizierer starten möchte. Wenn der Nutzer die angebotene Methode unterstützt, sendet er im zweiten Schritt mittels eines EAP-Response erste Authentifizierungsdaten. Falls der Nutzer die angebotene Methode nicht durchführen kann, enthält der EAP-Response eine Anfrage nach einer anderen Methode. Die weitere Abfolge von Nachrichten hängt von der verwendeten Authentifizierungsmethode ab. Schließlich bekommt der Nutzer im letzten Schritt ein EAP-Success oder EAP-Failure, je nachdem ob die Authentifizierung erfolgreich war oder nicht.

2.2.2 EAP Authentifizierungsmethoden

Es wurden in EAP bereits mehrere Authentifizierungsmethoden integriert. Drei der gängigsten Methoden sollen im Folgenden vorgestellt werden.

EAP-Identity

Die Identity Methode wird verwendet, um dem Authentifizierer Informationen über einen Nutzer zu übermitteln. Innerhalb der Identity Methode findet keine echte Authentifizierung statt, vielmehr wird die Identity Methode nur als Vorstufe für eine darauf folgende Authentifizierungsmethode verwendet. Innerhalb der Identity Methode werden Informationen in Klartext transportiert, was sie sehr angreifbar macht. Somit ist oft nicht die wahre Identität des Nutzers in einem Identity-Paket enthalten, sondern lediglich eine

Kennung, aus der der Authentifizierer die benötigten Informationen für dien Initiierung der zweiten Methode ableiten kann. Fast alle Authentifizierungsmethoden in EAP verwenden die Identity Methode für Initialisierungszwecke.

EAP-MD5-Challenge

MD5-Challenge ist eine EAP-Methode, bei der Nutzer sich mit Hilfe von Passwörtern authentifizieren können. Um die Daten vor unrechtmäßigem Zugriff zu schützen, wird das Passwort nicht im Klartext übermittelt, sondern mit dem MD5 Algorithmus (Message Digest Algorithm 5) verschlüsselt. Der MD5 Algorithmus ist eine kryptografische Hashfunktion, die aus einer Zeichenfolge variabler Länge einen 128-Bit-Hashwert erzeugt. Vor jeder MD5-Challenge muss die Identity Methode durchgeführt werden. Dies ist notwendig, da dem Authentifizierer bekannt sein muss, welche Nutzerdaten für die Berechnung des Hashwerts verwendet wurden. Der Authentifizierungsvorgang erfolgt in drei Schritten.

Im ersten Schritt der MD5-Challenge sendet der Authentifizierer dem Nutzer eine zufällige Zeichenfolge (Challenge). Der Nutzer bildet aus der empfangenen Zeichenfolge, seiner Identität und seinem Passwort den 128-Bit Hashwert und übermittelt ihn im zweiten Schritt dem Authentifizierer. Durch die vorher durchgeführte Identity Methode ist dem Authentifizierer bekannt, welche Identität und welches Passwort er zusammen mit der Challenge für die Hashwertbildung verwenden muss und kann seinerseits den Wert errechnen. Sind die beiden Hashwerte gleich, ist die Authentifizierung erfolgreich und dem Nutzer wird eine EAP-Success Meldung übermittelt.

Der Vorteil der MD5-Challenge liegt darin, dass ein Angreifer, der die Kommunikation mithört, nur die Identität, die Challenge und den Hashwert erfährt. Der Hashwert ist dabei für eine spätere Kommunikation wertlos, da er durch die wechselnden Challenges immer verschieden sein wird. Das Passwort mit Hilfe der drei vorhandenen Komponenten zu errechnen ist aber theoretisch unmöglich, da es sich beim MD5-Algorithmus um eine Einweg-Funktion handelt.

Der Problem mit dem man beim Einsatz der MD5-Challenge Methode konfrontiert ist besteht darin, dass es nur theoretisch unmöglich ist das Passwort mit Hilfe der im Klartext übertragenen Identität, Challenge und Hashwerts auszurechnen. Ein Angreifer kann nämlich eine Wörterbuchattacke starten, bei der er wahrscheinliche Passwörter mit der Identität und der Challenge verknüpft und prüft, ob das Ergebnis mit dem abgehörten Hashwert übereinstimmt. Somit ist die Qualität des gesamten Verfahrens von den

benutzten Passwörtern abhängig. Eine derartige Einschränkung der Sicherheit ist in manchen Netzwerken nicht akzeptabel.

EAP-TLS

Eine sehr viel höhere Sicherheit bietet das weit verbreitete EAP-TLS (Transport Layer Security) Protokoll. Es setzt eine installierte *Public Key Infrastructure (PKI)* [19] voraus, die als „Trusted Third Party“, also als vertrauenswürdiger Dritter, fungiert. Die Aufgaben einer PKI umfassen:

- Sichere Erstellung gültiger Schlüssel,
- Überprüfung der Gültigkeit der Identität,
- Ausgabe, Erneuerung und Beendigung von digitalen Zertifikaten,
- Gültigkeitsprüfung von Zertifikaten,
- Verteilung von Zertifikaten und der damit verbundenen Informationen,
- Sichere Verwaltung, d. h. sichere Archivierung und sicheres Wiederfinden von Schlüsseln,
- Generierung von digitalen Signaturen und Zeitstempeln,
- Aufbau und Verwaltung von Vertrauensstellen.

EAP-TLS verwendet für die Authentifizierung digitale Zertifikate, die von der PKI ausgestellt werden. Ein *digitales Zertifikat* verknüpft eine Identität mit ihrer Public-Key und enthält zusätzliche Informationen über die Identität. Die Informationen in einem digitalen Zertifikat werden durch die digitale Signatur der ausstellenden Autorität bestätigt. Nachdem man eine bestimmte Autorität als vertrauenswürdig genug befunden hat, wird sie zur Erstellung eines digitalen Ausweises, dem digitalen Zertifikat, beauftragt. Damit andere Instanzen ein digitales Zertifikat eindeutig einer Identität zuordnen können, muss das digitale Zertifikat genügend Informationen bereitstellen. Ein mögliches Format für ein digitales Zertifikat ist beispielsweise der *X.509 Standard* [20]. Der X.509 Standard definiert den Inhalt und die Anwendung digitaler Zertifikate. Danach enthält ein Zertifikat folgende Merkmale:

- *Inhaber*: Identifikation der Person/Institution, für die das Zertifikat herausgegeben wurde,
- *Öffentlicher Schlüssel*: Public-Key des Inhabers,
- *Aussteller*: Eine vertrauenswürdige Zertifizierungsstelle (Certificate-Authority), die das Zertifikat generiert und signiert hat,
- *Seriennummer*: Die Seriennummer dient der eindeutigen Identifizierung eines Zertifikats, auch wenn ein weiteres Zertifikat mit genau (oder weitgehend) den gleichen Informationen herausgegeben wird,
- *Gültig ab*: Zeitpunkt, ab dem das Zertifikat eingesetzt werden darf,

- *Gültig bis*: Spätester Zeitpunkt, bis zu welchem das Zertifikat eingesetzt werden darf,
- *Schlüsselverwendung*: Beschreibung der zulässigen Anwendungszwecke, für die der Public-Key des Inhabers verwendet werden darf,
- *Digitale Signatur*: Die digitale Signatur der Zertifizierungsstelle gilt als Beweis der Identität des Inhabers. Mit Hilfe des Hashwerts können die Inhalte eines Zertifikates überprüft werden, damit manipulierte Zertifikate ausgeschlossen werden können,
- Das Zertifikat umfasst weitere Attribute, die von bestimmten Applikationen direkt angesprochen werden, z.B. die E-Mail Adresse einer Identität.

Bei der Durchführung der EAP-TLS Methode werden sowohl vom Nutzer als auch vom Server digitalen Zertifikate zum Zwecke der Authentifizierung verwendet. Die in den Zertifikaten enthaltenen öffentlichen Schlüssel werden zur Verschlüsselung des Datenverkehrs und zur Überprüfung der Signatur des Kommunikationspartners eingesetzt. Diese wird mit dem privaten Schlüssel der Entität erstellt und an jedes Paket angefügt, um Manipulationen zu entlarven.

Wie auch EAP-MD5-Challenge, erfordert EAP-TLS die Durchführung von EAP-Identity vor der Authentifizierung. Nach dem erfolgreichen Abschluss der Identity Methode sendet der Authentifizierer dem Nutzer sein Zertifikat. Dieser überprüft ob das Zertifikat von einer vertrauenswürdigen Autorität stammt. Falls das Zertifikat korrekt ist, übermittelt der Nutzer sein eigenes Zertifikat dem Authentifizierer, der es nun seinerseits überprüft. Wenn auch das Nutzerzertifikat erfolgreich validiert wurde, lässt der Authentifizierer dem Nutzer eine EAP-Success Meldung zukommen.

Durch eine beidseitige Authentifizierung wird sichergestellt, dass Nutzer ihre Daten nicht feindlichen Entitäten, die sich als Authentifizierer ausgeben, übermitteln. Somit garantiert die beidseitige Authentifizierung zusammen mit der Verschlüsselung und der Signierung des Datentransports eine extrem hohe Sicherheit. Aus diesem Grund konnte sich EAP-TLS auf breiter Front durchsetzen und wird beispielsweise von allen Microsoft-Betriebsystemen unterstützt.

Die einzige Hürde für den Einsatz von EAP-TLS ist die Notwendigkeit einer PKI. Der Aufbau und die Verwaltung einer PKI ist mit hohen Kosten und erheblichem Aufwand verbunden, was wiederum viele Organisationen dazu zwingt, billigere, dafür aber weniger sichere Authentifizierungsmethoden einzusetzen.

2.2.3 Bewertung und Einsatzmöglichkeiten

Das EAP Protokoll konnte sich im Bereich der Netzwerkauthentifizierung schnell etablieren. Es wird sowohl in drahtgebundenen, als auch in drahtlosen Netzen zum Transport der Authentifizierungsdaten eingesetzt und wird vom IEEE 802.11i Protokoll [21], dem zukünftigen Standard für Sicherheit in WLAN, verwendet.

Der Hauptgrund für den Erfolg des EAP-Protokolls ist seine Erweiterbarkeit. Es kann jede beliebige Authentifizierungsmethode in einem einheitlichen Format transportieren. Dadurch ist es sehr flexibel im Bezug auf die Integration neuer Protokolle, so dass Clients und Server beim Erscheinen einer neuen Methode nicht neu konfiguriert werden müssen. Des Weiteren bietet EAP den Kommunikationspartnern die Möglichkeit, miteinander Authentifizierungsmethoden auszuhandeln. Wenn ein Partner die angebotene Methode nicht unterstützt, kann er eine andere Methode vorschlagen. Dadurch müssen nicht alle Nutzer die gleiche Methode unterstützen. Ein Authentifizierer, der standardmäßig an eine PKI angeschlossen ist, kann so auch Nutzer authentifizieren, die (noch) keinen Zugang zu PKI haben.

EAP wird überwiegend in sehr großen und komplexen Netzwerken eingesetzt. In derartigen Netzen gibt es meist mehr als einen Zugangspunkt, so dass auch mehrere Authentifizierer notwendig wären. Um sicherzustellen, dass sie keinen berechtigten Nutzer ablehnen, müssten sämtliche Nutzerdaten bei jedem Authentifizierer redundant gespeichert werden. Diesen enormen Aufwand gilt es zu vermeiden, weswegen in EAP der Authentifizierer in zwei Komponenten gespaltet werden kann: den Authentication Server und den Passthrough Server.

Der Authentication Server befindet sich im Back-End und ist für die eigentliche Authentifizierung der Nutzer zuständig. Er enthält sämtliche Nutzerdaten und ist gegebenenfalls an eine PKI angeschlossen. Der Passthrough Server ist am Network Access Server positioniert und ist für die Weiterleitung der Nachrichten zwischen Nutzer und Authentication Server zuständig. Am Ende eines Authentifizierungsvorgangs gewährt oder verweigert er den Netzwerkzugang, basierend auf Anweisungen vom Authentication Server.

Ein typisches Szenario für die Authentifizierung mittels EAP in großen Netzwerken zeigt die Abbildung 3. Ein Nutzer, der an einem Notebook angemeldet ist, möchte Zugang zum Netzwerk erlangen. Dafür nimmt er Kontakt mit dem Passthrough Server auf, der die Identity Methode startet und ihn auffordert sich zu identifizieren. Der Nutzer antwortet mit

seiner Identität. Der Passthrough Server leitet anhand dieser Information die Antwort an den zugehörigen Authentication Server weiter. Dieser handelt mit dem Nutzer EAP-TLS als Authentifizierungsmethode aus, fordert bei der PKI die Ausstellung eines Zertifikats an und sendet es dem Nutzer. Der Nutzer überprüft das Zertifikat des Authentication Servers, lässt sich von der PKI auch ein Zertifikat ausstellen und übermittelt es dem Server. Dieser validiert das Zertifikat des Nutzers und entscheidet, dass ihm Zugang zum Netzwerk gewährt werden soll. Er sendet diese Anweisung an den Passthrough Server, dessen Aufgabe es ist, die Anweisung umzusetzen.

Das oben beschriebene Szenario ist nur möglich, wenn für den Transport der EAP Pakete ein Protokoll eingesetzt wird, das Routing-Aufgaben übernehmen kann. Ein solches Protokoll ist Diameter, das im folgenden Kapitel ausführlich behandelt werden soll.

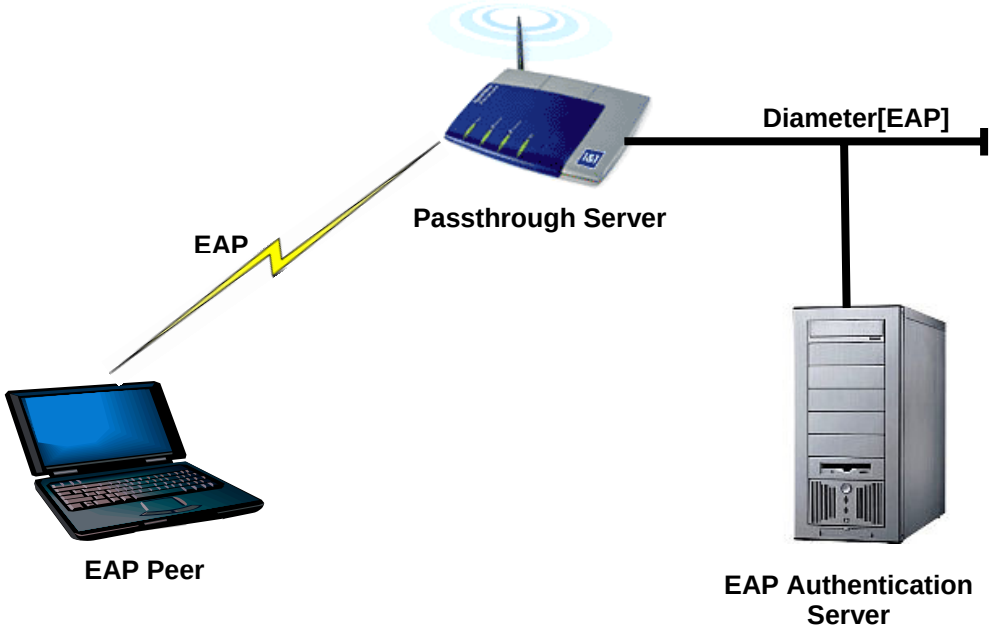


Abbildung 3: Typisches Szenario für die Authentifizierung mittels EAP

2.3 Das Diameter Protokoll

Das im vorigen Abschnitt eingeführte EAP ist ein Protokoll der Sicherungsschicht und damit abhängig von der zugrunde liegenden Netzwerktechnologie. Dies stellt in komplexen Netzwerken, in denen Teilabschnitte unterschiedliche Technologien wie ATM und Ethernet einsetzen können, ein Problem dar. Um die EAP-Authentifizierungsdaten über heterogene Netzwerke zu versenden, ist ein Mechanismus notwendig, der die Kommunikation unabhängig

von der zugrunde liegenden Infrastruktur macht. Diese Aufgabe kann das Diameter Protokoll übernehmen, das im Folgenden vorgestellt wird.

In modernen Netzwerken müssen Server, die den Zugriff auf Netzwerkressourcen kontrollieren, Protokolle unterstützen, mit denen sie überprüfen können, ob ein Nutzer derjenige ist für den er sich ausgibt, ob er die Erlaubnis haben soll bestimmte Dienste zu nutzen und wie viel ihm für die Nutzung dieser Dienste in Rechnung gestellt werden soll. Solche Protokolle, die zur Authentifizierung, Autorisierung und Abrechnung eingesetzt werden, nennt man *AAA-Protokolle* [22].

Das derzeit am weitesten verbreitete AAA-Protokoll ist der *Remote Authentication Dial In User Service (RADIUS)* [23]. Er wurde entwickelt, um AAA-Dienste für Einwahl-Verbindungen zur Verfügung zu stellen. Im Laufe der Zeit konnte RADIUS jedoch der zunehmenden Komplexität der Server und der angebotenen Dienste nicht mehr gerecht werden, weswegen ein neues Protokoll entwickelt werden musste. Im Jahr 2003 wurde bei der IETF-Gruppe als designierter Nachfolger von RADIUS das *Diameter* [24] Protokoll angenommen. Bei der Entwicklung von Diameter wurde großen Wert auf die Rückwärtskompatibilität zu RADIUS gelegt. Dabei hat man aber auch darauf geachtet, dass das Protokoll den zukünftigen Anwendungen des Internets, wie Voice over IP, Fax over IP und Mobile IP genügen würde. Dies wurde durch den modularen Aufbau von Diameter erreicht.

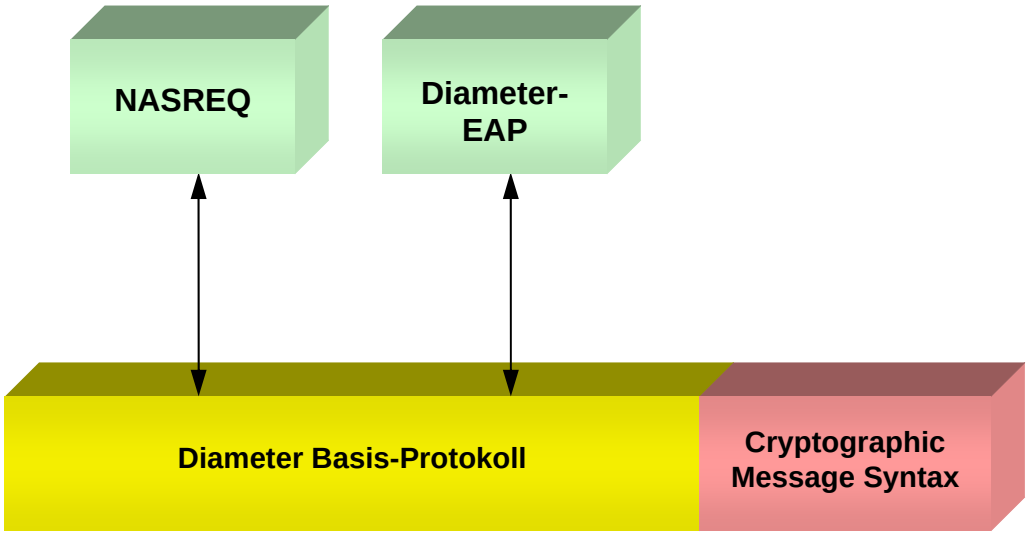


Abbildung 4: Komponenten des Diameter Protokolls

Die einzelnen Komponenten des Standards veranschaulicht die Abbildung 4. Als Fundament für jede Diameter-Implementierung dient das *Diameter Base Protocol* [12] und

das *Cryptographic Message Syntax Modul (CMS)* [25]. Das Basis-Protokoll definiert die Regeln, die festlegen wie der Nachrichtenaustausch zu erfolgen hat, wie Nachrichten zwischen den verschiedenen Diameter-Knoten verteilt werden, und welches Format die Nachrichten haben müssen. Das CMS-Modul stellt die notwendigen Sicherheitsmechanismen wie Verschlüsselung und digitale Signaturen zur Verfügung. Auf diesen beiden Komponenten aufbauend wurden verschiedene Zusatzprotokolle entwickelt. Die wichtigsten wären *Network Access Server REQuirements (NASREQ)* [26] und *Diameter-EAP* [27]. Die einzelnen Protokolle werden im nächsten Abschnitt vorgestellt.

2.2.3 Das Diameter Basis-Protokoll

Das Diameter Basis-Protokoll wurde mit dem Ziel entwickelt, den verschiedenen Anwendungen, die es realisieren, eine sichere und zuverlässige Nachrichtenübertragung zur Verfügung zu stellen. Deswegen ist es notwendig, dass das Protokoll möglichst einfach und somit leicht zu implementieren ist. Um dies zu erreichen, wurde für Diameter ein relativ einfaches Nachrichtenformat gewählt.

Diameter Nachrichtenformat

Eine Diameter Nachricht besteht aus einem *Header* gefolgt von einem oder mehreren *Attribut-Wert Paaren* (vgl. Abbildung 5). Die einzelnen Felder des Header werden im Folgenden beschrieben.

- Das *Versionsfeld* gibt an, ob es sich um eine Diameternachricht handelt oder aufgrund der Rückwärtskompatibilität um ein RADIUS-Paket.
- Das *Message Length* Feld gibt die gesamte Größe der Diameternachricht an. Sowohl die Felder aus dem Header als auch die AVPs werden dabei berücksichtigt.
- Die *Command Flags* legen den Typ der Nachricht fest. Eine Nachricht kann eine Anfrage, eine Antwort oder eine Fehlermeldung sein. Zusätzlich definieren die Command Flags ob eine Nachricht weitergeleitet werden darf oder lokal verarbeitet werden muss und ob die Nachricht zum ersten Mal versendet wird oder ob es sich um eine Wiederholung handelt.
- Der *Command Code* einer Nachricht definiert, welche Aktionen beim Empfang der Nachricht ausgeführt werden müssen. Beispielsweise kann eine Reauthentifizierung, ein Verbindungsabbruch oder das Beenden der aktuellen Sitzung verlangt werden.
- Die *Application-ID* gibt an, für welche Art von Anwendung die Nachricht verwendbar ist. Dabei kann es sich um eine Authentifizierungsanwendung, eine Autorisierungsanwendung oder eine benutzerdefinierte Anwendung handeln.

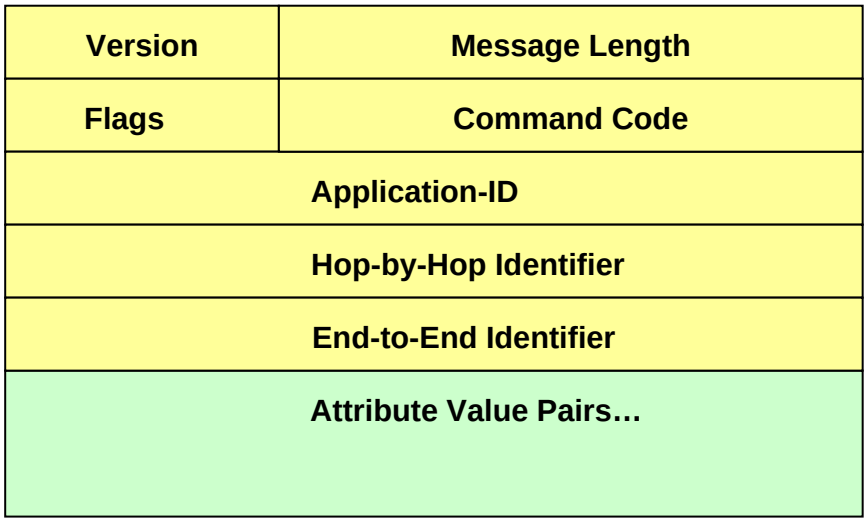


Abbildung 5: Diameter Nachrichtenformat

- Der *Hop-by-Hop-Identifier* ermöglicht es, Antworten auf Anfragen über mehrere Zwischenknoten hinweg and den richtigen Empfänger zu leiten. Dabei wird in jedem Zwischenknoten der Hop-by-Hop-Identifier der Anfrage gespeichert, mit der Adresse des aktuellen Knotens ersetzt und die Anfrage weitergeleitet. Der Empfänger der Anfrage sendet daraufhin seine Antwort an die Adresse, die er aus dem Hop-by-Hop-Identifier ausliest. Die Zwischenknoten ersetzen jeweils die Empfängeradresse der Antwort mit den vorher gespeicherten Daten aus dem Identifier der Anfrage. So wird die Antwort über genau den gleichen Weg geleitet, den auch die Anfrage zurücklegen musste. Dadurch ist jedem Zwischenknoten bekannt, ob auf eine Anfrage auch eine Antwort gesendet wurde.
- Das letzte Feld im Diameter Header ist der *End-to-End-Identifier*. Er wird benötigt, um Duplikate von Nachrichten zu erkennen. Jeder Anfrage wird ein eindeutiger End-to-End-Identifier zugeordnet. Der Identifier wird von den Zwischenknoten nicht verändert und wird vom Empfänger auch für den End-to-End-Identifier der Antwort verwendet. Dadurch ist sicher gestellt, dass der Initiator der Anfrage eine Antwort nur einmal verarbeitet, weil er erkennen, kann falls auf seine Anfrage mehrere Antworten versendet wurden.

Neben dem Header beinhalten Diameter Nachrichten mindestens ein Attribut-Wert Paar. Mit ihrer Hilfe werden zwischen den Knoten Daten für die Authentifizierung, Autorisierung, Abrechnung aber auch Sicherheits- und Konfigurationsinformationen transportiert. Diese Daten werden in Feldern gespeichert, die nun vorgestellt werden sollen.

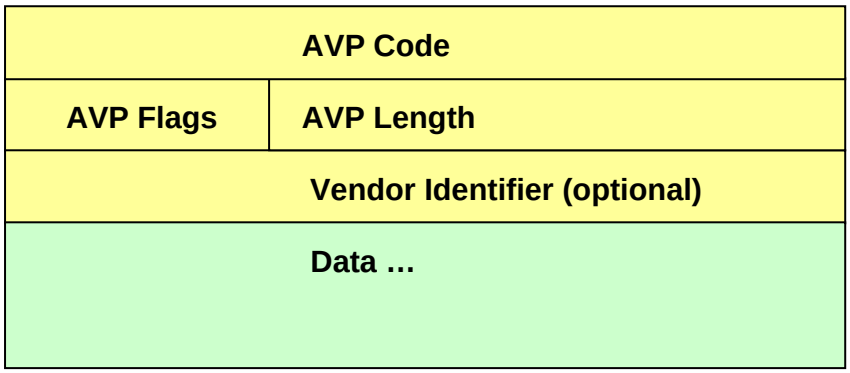


Abbildung 6: Diameter Attribut-Wert Paar

Das erste Feld eines Diameter Attribut-Wert Paares ist der *AVP Code*. Er dient der eindeutigen Identifizierung der einzelnen Attribute.

Durch das Auslesen der *AVP Flags* kann ein Empfänger erkennen, welche Aktionen die empfangenen Attribut-Wert Paare erfordern. So kann beispielsweise mittels dieses Feldes die Notwendigkeit einer Ende-zu-Ende-Verschlüsselung angezeigt werden.

Im Feld *AVP Length* wird die gesamte Länge des Attribut-Wert Paares gespeichert. Wird eine Nachricht empfangen, bei der die angegebene von der tatsächlichen Länge abweicht, muss diese Nachricht abgelehnt werden.

Der *Vendor Identifier* ist ein optionales Feld. Falls eine Organisation zusätzlich zum Diameter Standard Attribut-Wert Paare entwickelt, wird der eindeutige Bezeichner der Organisation im Feld Vendor Identifier der neuen Attribut-Wert Paare hinterlegt.

Das letzte Feld ist das *Datenfeld*, das die Werte der Attribute enthält. Das Format und die Länge des Datenfeldes sind durch den AVP Code und AVP Length festgelegt. Die im Datenfeld gespeicherten Werte dürfen vom Typ Integer64, Unsigned64, Float64 oder OctetString sein. Zusätzlich ist es noch möglich über den Datentyp Grouped eine Sequenz von Attribut-Werte Paaren zu speichern.

Diameter Architektur

In einem Diameter-Netz werden Nachrichten vom Absender meist über mehrere Zwischenknoten zum Empfänger geleitet. Dabei übernehmen die einzelnen Knoten verschiedene Rollen:

Der *Client* ist üblicherweise ein Network Access Server und befindet sich an der Grenze eines Diameter-Netzes. Er gewährt oder verweigert Nutzern den Zugang zum Netzwerk. Die Entscheidungen darüber basiert er auf Informationen, die er von einem Server bekommt.

Der *Server* ist für die Authentifizierung und Autorisierung von Subjekten, die sich bei Clients anmelden, zuständig. Im Falle einer erfolgreichen Authentifizierung und Autorisierung übernimmt der Server auch die Abrechnung der in Anspruch genommenen Dienste.

In Diameter Netzen müssen sich Server und Clients nicht kennen. Die Nachrichten können an so genannte Agenten gesendet werden, deren Aufgabe es ist, die Nachrichten an den Empfänger weiterzuleiten. Durch den Einsatz von Relais-Knoten entfällt die Rekonfiguration, die bei allen Clients durchgeführt werden müsste, falls ein Server dem Netz hinzugefügt wird. Es gibt vier Arten von Agenten:

Relay Agents sind, wie der Name schon sagt, einfache Relais. Sie leiten die Nachrichten basierend auf den im Header enthaltenen Informationen an den nächsten Knoten weiter. Die einzige Änderung, die sie an den Nachrichten vornehmen dürfen, ist das Einfügen von Routing-Informationen.

Proxy Agents sind eine Weiterentwicklung der Relay Agents. Sie routen nicht nur Nachrichten, sondern dürfen sie auch ändern, um netzwerkspezifische Regeln durchzusetzen. Proxy Agents haben jedoch einen großen Nachteil: da sie die Nachrichten überprüfen und eventuell ändern müssen, kann keine Ende-zu-Ende Verschlüsselung oder Signierung verwendet werden.

Der in der Abbildung 7 skizzierte *Redirect Agent* ist eine Art zentrales Adressbuch eines Diameter Netzes. Er kann von Relay Agents oder Clients mit der Weiterleitung von Nachrichten beauftragt werden. Anstatt die Nachricht an den Empfänger weiterzuleiten, sendet der Redirect Agent eine Antwort mit Routing-Informationen an den Absender der Nachricht. Dies ermöglicht dem Absender den Empfänger direkt und ohne Umwege anzusprechen. Redirect Agents werden in Szenarien eingesetzt, in denen die Routing-Konfiguration zentralisiert werden soll. Ein Beispiel hierfür wäre ein Redirect Agent, der sämtliche Adressen innerhalb eines Konsortiums kennt, die Last der Vermittlung aber den Konsortialpartnern überlässt, um Flaschenhälse zu vermeiden.

Die vierte Art von Agenten heißen *Translation Agents* und sorgen dafür, dass Diameter mit anderen Protokollen interagieren kann. Insbesondere wird durch den Einsatz von Translation Agents die Rückwärtskompatibilität zu RADIUS erreicht.

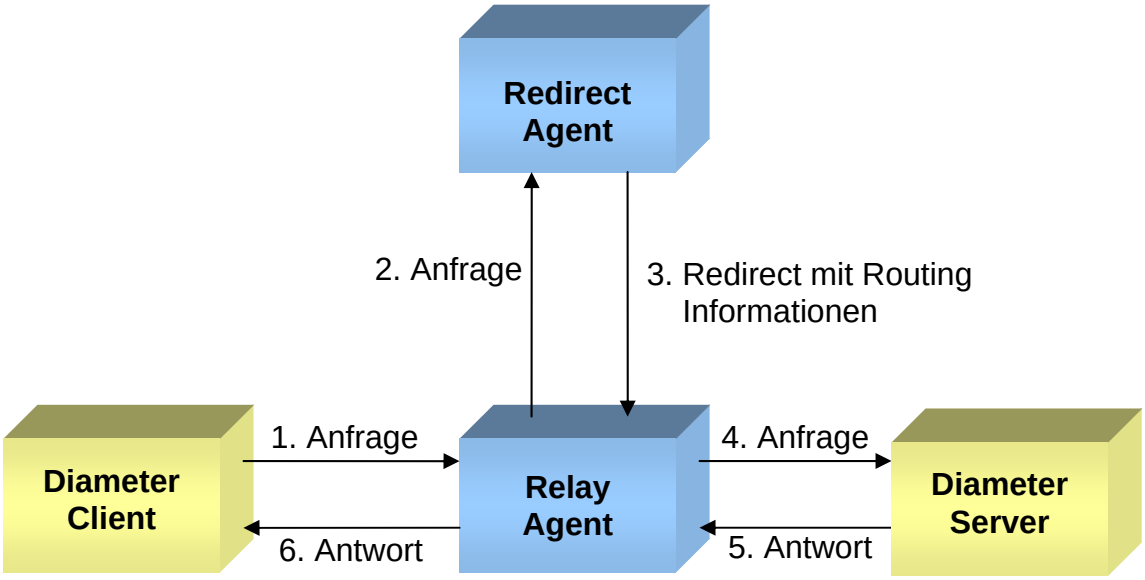


Abbildung 7: Diameter Architektur

Nachrichtentransport in Diameter Netzen

Ältere Protokolle, die für Authentifizierung, Autorisierung und Abrechnung eingesetzt wurden, setzten zum Nachrichtentransport verbindungslose und somit unzuverlässige Protokolle ein. So setzt RADIUS das UDP-Protokoll [28] ein, das auf die Schnelligkeit und nicht auf die Qualität der Verbindung ausgerichtet ist. Diameter verwendet im Gegensatz dazu SCTP [29], ein zuverlässiges und verbindungsorientiertes Protokoll.

SCTP steht für *Stream Control Transmission Protocol* und ist eine Weiterentwicklung des stark verbreiteten TCP Protokolls. SCTP bietet wie auch TCP einen verbindungsorientierten Transport zwischen zwei Endpunkten (Sockets). Es garantiert einen zuverlässigen Datentransfer indem es die Integrität der Daten und ihre Übermittlung in der korrekten Reihenfolge überprüft. Duplikate werden ebenso wie verloren gegangene Pakete erkannt und Gegenmaßnahmen eingeleitet. Auch für die Fluss- und Überlastkontrolle verwendet SCTP ähnliche Algorithmen wie TCP.

Zusätzlich zu den von TCP übernommenen Merkmalen verfügt SCTP über weitere Fähigkeiten wie Multistreaming, Multihoming und Heartbeat-Messaging.

Multistreaming nennt man den gleichzeitigen Austausch von mehreren Datenströmen zwischen zwei Sockets. In jedem Datenstrom werden die Nachrichten in korrekter Reihenfolge versendet und auf Duplikate, beziehungsweise verloren gegangene Pakete überprüft. Kommt es innerhalb eines Datenstroms zu einem Fehler, findet die Fehlerkorrektur nur innerhalb dieses einen Datenstroms statt, währen die Sockets ohne Verzögerung über die restlichen Datenströme kommunizieren können. Somit hält ein Fehler beim Einsatz von SCTP nicht den gesamten Datenverkehr zwischen zwei Endpunkten auf, was einen großen Vorteil gegenüber dem langsamen TCP darstellt.

Eine weitere wichtige Eigenschaft von SCTP ist die Unterstützung von *Multihoming*. Mittels Multihoming können einem Gerät mehrere IP-Adressen zugewiesen werden, die es gleichzeitig benutzen kann. Dadurch können mehrere Netzwerkpfade über verschiedene Zugangstechnologien zwischen zwei Endpunkten aufgebaut werden. Insbesondere ist auch der gleichzeitige Einsatz von mehreren Internetdienstanbietern möglich. Fällt nun ein Dienstanbieter aus, kann die Kommunikation über den nächsten Pfad geleitet werden.

Um die Verfügbarkeit eines Pfades zu überprüfen, werden in regelmäßigen Abständen Heartbeat-Nachrichten über den Pfad gesendet. Werden die Heartbeats innerhalb einer bestimmten Zeit nicht beantwortet, werden der Pfad und die damit verbundene IP-Adresse deaktiviert. Durch die gleichzeitige Verwendung mehrerer Netzwerkpfade kann SCTP im Gegensatz zu TCP einen Weiterbetrieb auch nach einem Netzausfall ermöglichen.

Sicherheit in Diameter-Umgebungen

Eine hohe Sicherheit ist für ein Protokoll, das sensible Daten zur Authentifizierung, Autorisierung und Abrechnung transportiert, eine Notwendigkeit. Aus diesem Grund werden in Diameter zwei Mechanismen eingesetzt: die Absicherung des Datentransports zwischen den einzelnen Knoten und zusätzlich die Ende-zu-Ende Sicherung.

Für die Absicherung des Datentransports zwischen Diameterknoten können zwei Technologien eingesetzt werden: *IPsec* [30] und *TLS* [35]. Beide Protokolle können durch Verschlüsselung und Signierung den Schutz der Vertraulichkeit und der Integrität der Nachrichten gewährleisten. Allerdings gilt dies nur für den Pfad zwischen zwei Knoten, beispielsweise zwischen einem Server und einem Proxy Agent. Der Proxy Agent kann die eingegangene Nachricht entschlüsseln, beliebig ändern und anschließend weiterleiten (Abbildung 8). Dies ist in vielen Fällen nicht erwünscht, da es sich oft um vertrauliche Daten handelt, die nur der Server und der Client lesen und manipulieren dürfen. Um diesem Problem zu begegnen, nutzt Diameter das an das Basis-Protokoll gekoppelte CMS-Modul.

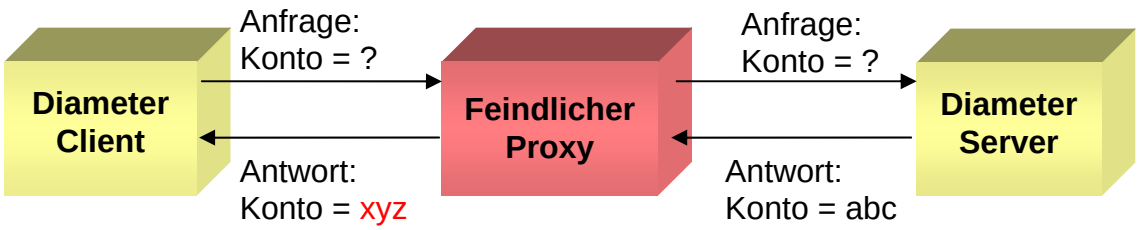


Abbildung 8: Nachrichtentransport ohne Ende-zu-Ende Sicherheit

Die Cryptographic Message Syntax ermöglicht die Ende-zu-Ende Sicherung von Attribut-Wert Paaren. Dies wird durch den Einsatz von Verschlüsselung und digitalen Signaturen erreicht. Die Verschlüsselung garantiert die Vertraulichkeit der verschlüsselten Attribut-Wert Paare, während digitale Signaturen für die Authentifizierung der Knoten und die Zertifizierung der Datenintegrität verwendet werden. Proxy Agents können so nur noch unverschlüsselte Attribut-Wert Paare lesen. Zusätzlich werden alle Änderungen, die sie an signierten Daten vornehmen, vom Empfänger der Daten sofort erkannt.

Dank der eingesetzten Technologien verfügt Diameter über einen hohen Sicherheitsstandard, weswegen Diameter Nachrichten mit geringem Risiko auch über fremde Infrastrukturen geleitet werden können.

2.3.2 Diameter Zusatzprotokolle

Obwohl alle Diameter-Implementierungen das Basis-Protokoll unterstützen müssen, gibt es keine Anwendungen, die ausschließlich auf dem Basis-Protokoll aufbauen. Zwar bietet es die Funktionalität für Authentifizierungs-, Autorisierungs- und Abrechnungsvorgänge, jedoch müssen Anwendungen das Basis-Protokoll stets um ein oder mehrere Zusatzprotokolle erweitern. Im Folgenden sollen die in dieser Arbeit eingesetzten Zusatzprotokolle NASREQ und Diameter-EAP vorgestellt werden.

NASREQ

Das Network Access Server REQuirements Protokoll (NASREQ) wurde entwickelt, um Nutzern von Einwahlnetzen Dienste zur Authentifizierung, Autorisierung und Abrechnung zur Verfügung zu stellen. NASREQ bildet zusammen mit dem Diameter Basis-Protokoll die Nachfolge für das veraltete RADIUS. Um eine möglichst reibungslose Migration zu ermöglichen, übernimmt NASREQ viele Attribute und Command Codes von RADIUS.

Die Aufgaben von Diameter gehen jedoch über die von RADIUS hinaus. Insbesondere hat man sich zum Ziel gesetzt Diameter flexibler zu machen als RADIUS. Dies wurde in erheblichem Maße durch den Einsatz von Network Access Servern erreicht. Die Funktionsweise von Network Access Servern wird in Diameter vom NASREQ Protokoll definiert. Durch ihren Einsatz können Nutzer nicht nur wie mit RADIUS direkt bei ihrem Heimatserver AAA-Dienste konsumieren, sondern auch über einen fremden Network Access Server diese Dienste in Anspruch nehmen. Der Network Access Server muss dabei nur die Identität des Nutzers erfahren, um seine Daten an den richtigen Heimatserver zu leiten. Die Authentifizierung, Autorisierung und Abrechnung der in Anspruch genommenen Dienste werden dabei weiterhin nur zwischen dem Nutzer und dem Heimatserver geregelt. Der Network Access Server hat keinen Zugriff auf die ausgetauschten Daten, sondern fungiert nur als Relay Agent. Falls die Kommunikation zwischen Nutzer und Heimatserver erfolgreich endet, teilt der Heimatserver dem Network Access Server mit, dass er dem Nutzer den angeforderten Dienst gewähren soll.

Zur Erfüllung dieser Aufgaben braucht NASREQ ein Protokoll, über das die Authentifizierungsdaten zwischen Nutzer und Heimatserver transportiert werden können. Es stützt sich dabei auf das bereits vorgestellte EAP Protokoll.

Diameter-EAP

Das in Kapitel 2.2 behandelte Extensible Authentication Protocol wird sehr häufig in Verbindung mit Diameter eingesetzt. Dabei werden zwischen Nutzern, die sich an einem Network Access Server anmelden, und dem Diameter Heimatserver EAP Pakete zum Zwecke der Authentifizierung ausgetauscht. Der Network Access Server nimmt in diesem Fall die Rolle eines Passthrough Server an, während der Diameter Server als Back-End Authentication Server fungiert. Für den Transport der EAP Pakete werden in Diameter-EAP neue Attribut-Wert Paare definiert.

Die Abbildung 9 veranschaulicht den Ablauf einer Diameter-EAP Sitzung.

1. Im ersten Schritt beantragt der Client, an dem der Nutzer angemeldet ist, beim Network Access Server den Start einer EAP Sitzung.
2. Der Network Access Server initiiert die EAP Sitzung zwischen ihm und dem Client.
3. Anschließend sendet der Network Access Server eine Diameter-EAP Anfrage an den Diameter Server, in der er den Start der EAP Kommunikation verlangt.
4. Der Diameter Server sendet eine Diameter-EAP Antwort, in der er die Durchführung einer EAP Methode verlangt. Üblicherweise wird hier die Durchführung der Identity Methode gefordert.

5. Der NAS entpackt die EAP Pakete aus den Attribut-Wert Paaren der Diameter Nachricht und leitet sie im vierten Schritt an den Client weiter.
6. Der Nutzer gibt am Client seine Identität an, worauf der Client die Identität in einer EAP Antwort an den NAS versendet.
7. Der NAS packt die EAP Pakete in Diameter Attribut-Wert Paare und leitet die Daten an den Diameter Server weiter.
8. Der Diameter Server prüft die erhaltene Identität. Falls der Nutzer bekannt ist, wird eine neue EAP Authentifizierungsmethode gestartet, mit der der Nutzer den Beweis seiner Identität antreten kann. Im Beispiel startet der Diameter Server eine Authentifizierung mittels EAP-MD5-Challenge, bei der der Nutzer sich mittels seines Passworts authentifizieren muss.
9. Der NAS leitet die Passwort-Anfrage an den Client weiter.
10. Der Nutzer gibt sein Passwort ein, der Client sendet es in einem EAP Paket an den NAS.

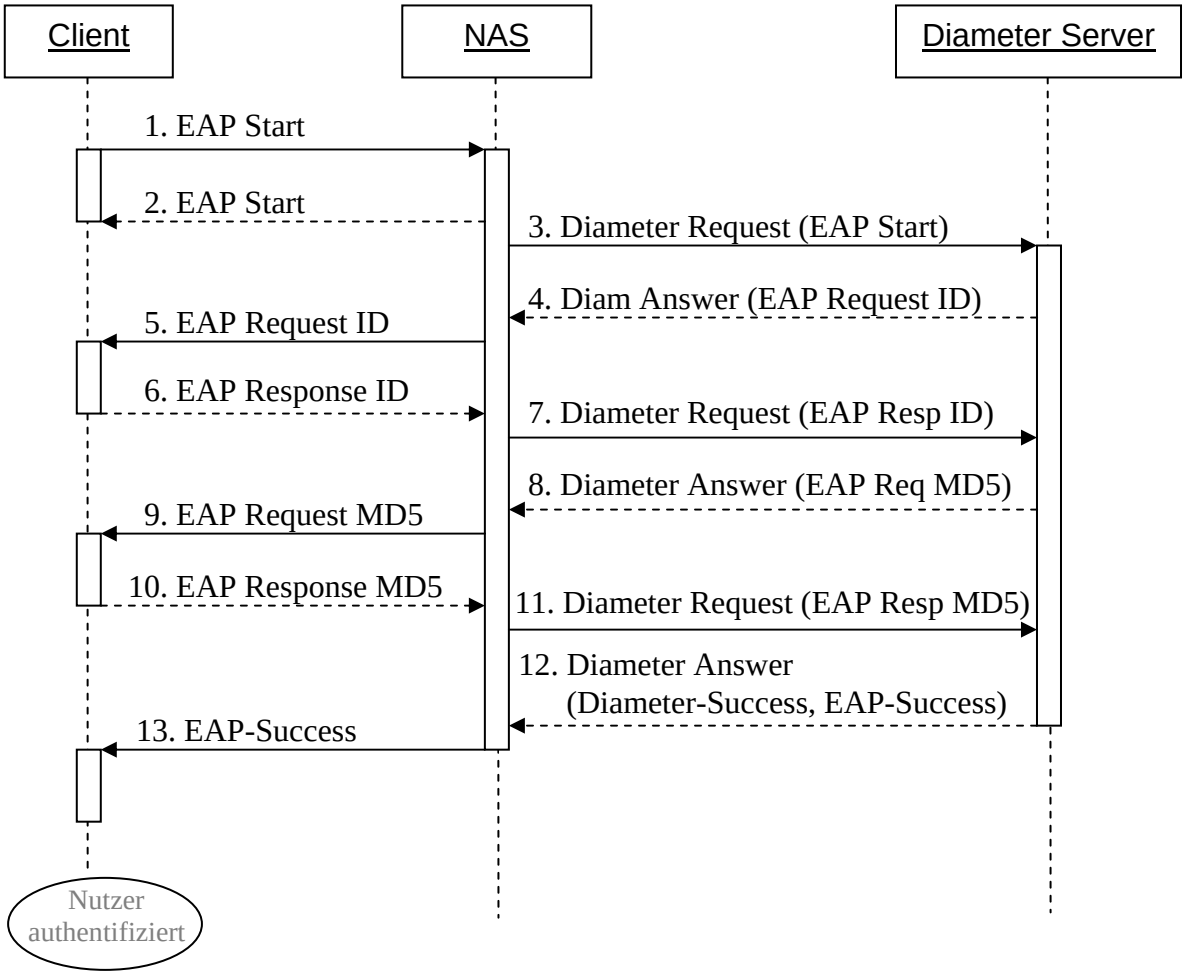


Abbildung 9: Authentifizierung über Diameter-EAP

11. Der NAS verpackt das bekommene EAP Paket in eine Diameter Nachricht und schickt diese an den Diameter Server.
12. Der Diameter Server überprüft das Passwort und versendet gegebenenfalls eine Antwort mit einer Diameter-Success Meldung für den Network Access Server und einer EAP-Success Meldung für den Nutzer.
13. Der Network Access Server gewährt im Falle des Eintreffens einer Diameter-Success Meldung dem Nutzer Zugang zum Netz und sendet ihm die EAP-Success Meldung

Zu beachten ist, dass zwischen dem Client und dem Network Access Server die EAP Pakete nicht über Diameter ausgetauscht werden. Sie können mittels Protokollen der Sicherungsschicht transportiert werden, beispielsweise *PPP* [32] für drahtgebundene Netze oder *IEEE 802.11* [35] für drahtlose Netze. Die Alternative wäre, sie unabhängig von der zugrunde liegenden Übertragungstechnologie über das *PANA*-Protokoll [13] zu transportieren. Da im Rahmen dieser Diplomarbeit für die Übertragung zwischen Client und Network Access Server PANA eingesetzt wurde, soll dieses Protokoll im Abschnitt 2.4 vorgestellt werden.

2.3.3 Bewertung des Diameter Protokolls

Abschließend wird eine Bewertung des Diameter Protokolls durchgeführt. Wie bereits erwähnt, wurde Diameter entwickelt, um das ältere RADIUS Protokoll zu ersetzen. Es sollte zu RADIUS kompatibel, aber auch sicherer und flexibler sein. Dies wurde durch Verbesserungen in den Bereichen Datentransport, Paketvermittlung und Sicherheit erreicht.

Im Bereich Datentransport wird durch den Einsatz von SCTP eine hohe Zuverlässigkeit erreicht. Fehler werden von SCTP erkannt und korrigiert, Datenpakete können somit nicht verloren gehen. Dies ist insbesondere für Abrechnungsdaten äußerst wichtig, da ihr Verlust erhebliche finanzielle Auswirkungen haben kann. Zusätzlich erhöht Multistreaming den Datendurchsatz, wodurch das System schnell auf Wünsche der Nutzer reagieren kann.

Die Paketvermittlung profitiert in erheblichem Maße von den in Diameter eingeführten Agenten. Sie sind über die Hop-by-Hop Identifier in den Diameter Nachrichten ständig darüber informiert, ob eine weitergeleitete Anfrage beantwortet wurde. Falls dies nicht der Fall ist, können sie eigenmächtig die Anfrage erneut versenden, ohne den ursprünglichen Absender zu kontaktieren. Dies hat kürzere Antwortzeiten und eine geringere Netzlast zur Folge. Zu einer hohen Flexibilität innerhalb von Diameter Netzen führt die Definition von Network Access Servern im NASREQ Zusatzprotokoll. Es ermöglicht Nutzern sich auch bei Servern außerhalb ihrer Heimatdomäne anzumelden und Dienste zu konsumieren.

Dadurch können Betreiber sich zu Konsortien zusammenschließen und müssen nicht überall für ihre Kunden Zugangsserver bereitstellen.

Die Sicherheit ist für Diameter eine zentrale Komponente und mit dem Basis-Protokoll eng verknüpft. Das CMS-Modul ermöglicht die Verschlüsselung und Signierung der Attribut-Wert Paare und garantiert dadurch eine Ende-zu-Ende Sicherheit für sämtliche Daten, die nicht für die Vermittlung lesbar bleiben müssen. Beim Transport zwischen den einzelnen Diameter-Knoten werden die Nachrichten zusätzlich vollständig verschlüsselt, wodurch Attacken erheblich erschwert, wenn nicht gar vereitelt werden.

Trotz der zahlreichen Vorteile von Diameter wurde das Protokoll bisher in keinem kommerziellen System implementiert. Das dürfte daran liegen, dass Diameter noch relativ neu ist - es wurde erst im September 2003 als Standard angenommen. Zwar haben an der Entwicklung des Protokolls mehrere große Unternehmen wie Nokia, Cisco, Sun und Ericsson mitgearbeitet, jedoch hat bisher keines dieser Unternehmen Anwendungen basierend auf Diameter entwickelt. Die bisher einzige (partielle) Implementierung des Diameter Protokolls stammt von dem Open Source Projekt *Open Diameter* [34]. Sie wurde im praktischen Teil der vorliegenden Diplomarbeit eingesetzt.

2.4 Das PANA Protokoll

Das im vorigen Abschnitt vorgestellte Diameter-Protokoll ist hervorragend geeignet, um EAP Pakete innerhalb komplexer und heterogener Netze zu transportieren. Allerdings wurde Diameter nur für die Kommunikation zwischen Diameter Clients (typischerweise Network Access Server) und Diameter Servern (Back-End Authentication Server) definiert. Für den Transport der Authentifizierungsdaten zwischen dem Endgerät des Nutzers und dem Network Access Server werden meist Protokolle der Sicherungsschicht wie PPP oder IEEE 802.11 eingesetzt. Protokolle der Sicherungsschicht sind jedoch abhängig von der verwendeten Übertragungstechnologie und somit unflexibel. Falls ein Network Access Server Nutzer bedienen möchte, die UMTS, WLAN und GPRS verwenden, müsste für jede Übertragungstechnologie ein eigenes Authentifizierungsprotokoll implementiert werden. Eine wesentlich einfachere Alternative wäre, alle Authentifizierungsdaten mittels eines Netzwerkprotokolls auf die IP-Ebene zu heben und dadurch die Verwendung eines einzigen Authentifizierungsprotokolls zu ermöglichen. Diese Alternative bietet das PANA Protokoll, das in den nächsten Abschnitten vorgestellt werden soll.

Das *Protocol for carrying Authentication for Network Acces (PANA)* [13] befindet sich noch in einem recht frühen Entwicklungsstadium. Bisher wurden bei der IETF sieben PANA-Drafts eingereicht, wobei Mitte 2005 eine neue Version erscheinen dürfte. PANA wurde mit dem Ziel entwickelt, EAP Pakete zwischen Clients und Network Access Servern auf der Netzwerkschicht zu transportieren. Insofern erfüllt es ähnliche Aufgaben wie Diameter auf dem Abschnitt zwischen Network Access Servern und Authentication Servern. Es war deswegen nahe liegend, das PANA Nachrichtenformat an das von Diameter anzulehnen.

2.4.1 PANA Nachrichtenformat

Wie im Falle von Diameter bestehen PANA Nachrichten aus Headern und Attribut-Wert Paaren. Der PANA Header ist weniger komplex als der von Diameter, da ein Routing über mehrere Knoten hinweg nicht notwendig ist. Der nächste Abschnitt gibt einen groben Überblick über die Felder einer PANA Nachricht.

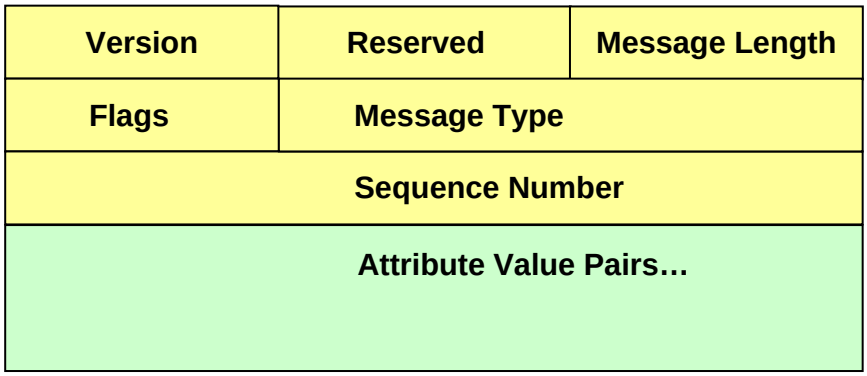


Abbildung 10: PANA Nachrichtenformat

- Das Feld *Version* zeigt das verwendete Protokoll an. Derzeit ist nur der Wert 1 für das PANA Protokoll definiert.

- Das Feld *Reserved* ist für die spätere Verwendung gedacht. Es hat noch keinerlei Funktion und muss immer den Wert 0 annehmen.

- Im Feld *Message Length* wird die Gesamtlänge der Nachricht gespeichert. Falls der Empfänger eine Nachricht bekommt, deren tatsächliche Länge nicht mit dem Wert von *Message Length* übereinstimmt, muss er diese Nachricht verwerfen. PANA ist ein UDP-basiertes Protokoll, somit ist eine Fehlerkorrektur nicht möglich. Der

Empfänger muss deshalb darauf warten, dass ihm nach einer Zeitüberschreitung die Nachricht erneut zugesendet wird.

- Das Feld *Flags* zeigt die Art der Nachricht an. Dabei kann es sich beispielsweise um eine Anfrage (Request) oder um eine Antwort (Response) handeln. Mit Hilfe dieses Feldes kann der Client auch mitteilen, ob er sich in zwei separaten Durchläufen beim Network Access Server und beim Authentifizierungsserver authentifizieren will.

- Im Feld *Message Type* wird der genaue Typ der Nachricht festgelegt. Beispiele für Nachrichtentypen wären PANA-Start-Request, der für den Aufbau der Sitzung verwendet wird und PANA-Auth-Request, dessen Aufgabe ist, EAP Pakete zu transportieren.

- Das Feld *Sequence Number* wird verwendet, um die Reihenfolge der PANA Nachrichten festzulegen. Beim Start einer PANA Sitzung wird das Feld auf 0 gesetzt und bei jeder versendeten Nachricht inkrementiert. Wird eine Nachricht empfangen, die eine andere als die erwartete Sequenznummer trägt, muss diese Nachricht verworfen werden. Dadurch werden Duplikate eliminiert und die Wiederverwendung der Nachrichten durch feindliche Sender verhindert.

- Das Feld *Attribute Value Pairs* transportiert die eigentlichen Authentifizierungsdaten. Da PANA Attribut-Werte Paare den gleichen Aufbau wie die bereits vorgestellten Diameter AVPs haben, soll auf ihre Beschreibung verzichtet werden.

2.4.2 Architektur und Einsatzszenario von PANA

PANA Nachrichten werden zwischen dem Endgerät des Nutzers und dem Network Access Server ausgetauscht. Aus diesem Grund muss sich auf jedem Endgerät, das PANA einsetzt, ein *PANA Client (PaC)* befinden. Ebenso muss jeder Network Access Server einen PANA Server, genannt *PANA Authentication Agent (PAA)*, enthalten.

Typischerweise fungiert PANA als Front-End einer AAA-Infrastruktur, zum Beispiel eines Diameter-Netzes. Die Schnittstelle zwischen PANA und Diameter ist der Network Access Server, der sowohl einen PAA als auch einen Diameter Client enthält. Somit findet die Konvertierung der PANA-EAP Nachrichten in Diameter-EAP Pakete im Network Access Server statt.

Die Abbildung 11 zeigt das Zusammenspiel zwischen PANA und Diameter. Der Nutzer verwendet einen Laptop mit einer UMTS-Karte und einem WLAN-Adapter. Er kann sowohl das UMTS-Netz eines Mobilfunkbetreibers nutzen, als auch dessen WLAN-Infrastruktur. Im Normalfall würde sich der Nutzer je nach gewünschter Zugangsart an verschiedene Network Access Server wenden. PANA vereinheitlicht jedoch den Transport der Authentifizierungsdaten, indem es sie auf die Netzwerkschicht hebt. Dadurch wird die Nutzung eines einzigen Network Access Servers mit beiden Technologien möglich.

Zu Beginn möchte der Nutzer über WLAN Zugang zum Internet erlangen und startet mittels seines PaC einen PANA-EAP Authentifizierungsvorgang. Im Network Access Server, auf dem sich der PAA und Diameter Client befinden, werden alle Nachrichten in das Diameter-EAP Format umgewandelt und an den Diameter Server im Back-End gesendet. Dieser authentifiziert den Nutzer über EAP und weist den Network Access Server an, ihm Zugriff auf das Netzwerk zu gewähren.

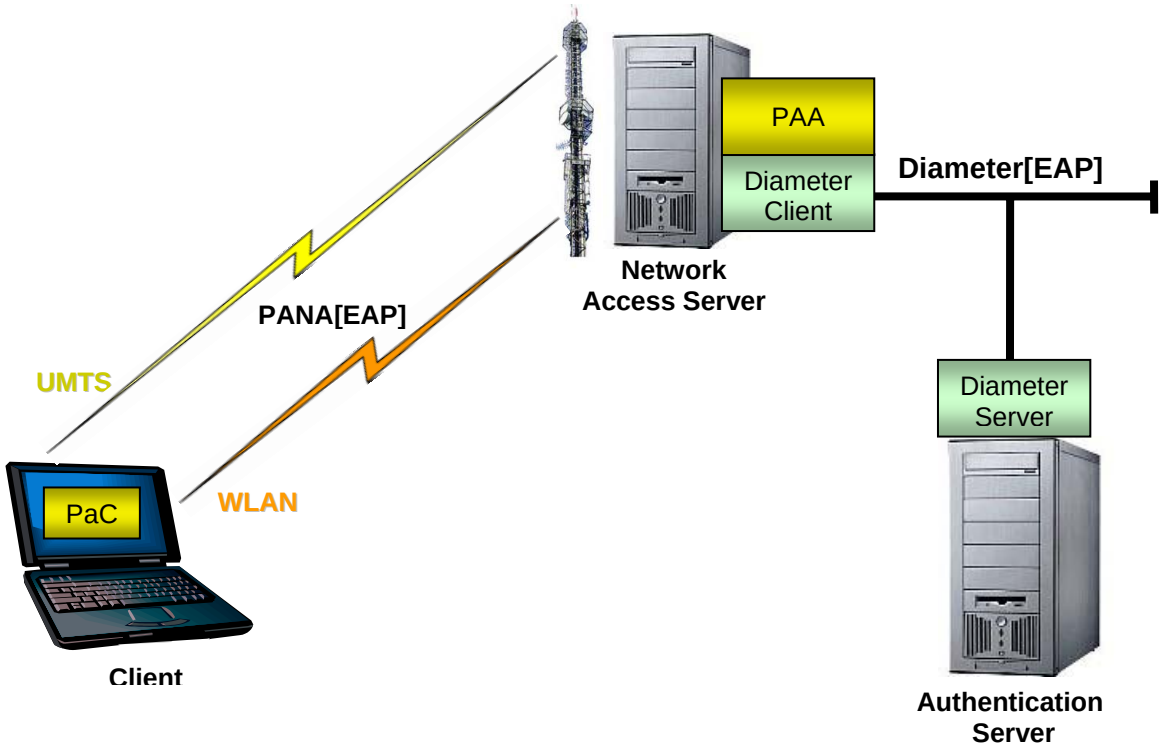


Abbildung 11: PANA Szenario

Im Laufe der Zeit bewegt sich der Nutzer außer Reichweite des WLANs, weswegen er sich entscheidet, über UMTS ins Internet zu gehen. Er baut zum gleichen Network Access Server diesmal eine Verbindung über UMTS auf und authentifiziert sich. Es werden wieder PANA-EAP Nachrichten zwischen PaC und PAA, beziehungsweise Diameter-EAP

Nachrichten zwischen Diameter Client und Diameter Server ausgetauscht. Der Diameter Server führt die Authentifizierung über EAP durch und weist bei erfolgreichem Abschluss den Network Access Server an, den Netzzugang für den Nutzer freizuschalten. Der Nutzer kann somit weiterhin auf das Internet zugreifen, obwohl die ursprüngliche Zugangstechnologie nicht mehr einsatzfähig ist.

2.4.3 Bewertung des PANA Protokolls

In dem vorgestellten Szenario wird die Flexibilität ersichtlich, die der Einsatz von PANA mit sich bringt. Network Access Server können nun Nutzer, die verschiedene Übertragungstechnologien nutzen, einheitlich authentifizieren. Die aufwändige Installation von separaten Authentifizierern für jede Zugangsart entfällt. Dies spart den Netzwerkbetreibern Kosten und ermöglicht eine schnelle und einfache Integration von neuen Technologien, was einen wichtigen Wettbewerbsvorteil darstellt.

Trotz dieser Vorteile dürfte es einige Zeit dauern, bis sich PANA durchsetzen kann. Das Protokoll steckt in einer sehr frühen Entwicklungsphase und es dürfte noch Jahre dauern bis ein Standard ratifiziert wird. Zwar gibt es schon eine PANA Implementierung im Open Diameter Projekt [34], es ist aber davon auszugehen, dass sie noch viele Änderungen erfahren wird. Aus diesem Grund ist PANA für den Forschungsbereich geeignet, ein kommerzieller Einsatz ist zurzeit jedoch nicht zu empfehlen.

2.5 Zusammenfassung

Das zweite Kapitel hat einen Überblick über die Authentifizierungsprotokolle vermittelt, die in dieser Arbeit verwendet werden. Zu Beginn wurde das EAP Protokoll vorgestellt, das die Einkapselung unterschiedlicher Authentifizierungsmethoden übernimmt und so ein generisches Protokoll für den Transport von Authentifizierungsdaten auf der Sicherungsschicht darstellt. Anschließend wurde das Diameter Protokoll eingeführt, das Dienste zur Authentifizierung, Autorisierung und Abrechnung anbietet und EAP Pakete zwischen Network Access Servern und Authentication Servern unabhängig von der Infrastruktur transportieren kann. Im letzten Teil des Kapitels wurde schließlich mit PANA ein Protokoll vorgestellt, das den Transport der EAP-Pakete zwischen Endgerät und Network Access Server übernehmen kann und diese auf die Netzwerkschicht hebt, um damit eine einheitliche Authentifizierung für alle Zugangsarten zu ermöglichen.

3. XML Sicherheit

Das Ziel der vorliegenden Arbeit ist eine Authentifizierungsmethode zu entwickeln, die möglichst generisch und damit für die heterogenen Netze der vierten Generation geeignet ist. Charakteristisch für diese Netze ist neben der Vielzahl unterschiedlicher Zugangstechnologien auch die starke Fragmentierung im Bereich der Netzbetreiber. Ein mobiler Nutzer wird während der Nutzung eines Dienstes mehrfach physische Domänen überschreiten und Netzwerkzugang von verschiedenen Netzbetreibern anfordern. Dafür müsste er sich bei jedem Betreiber erneut authentifizieren, was zu häufigen Unterbrechungen der Dienstnutzung führen würde. Eine Authentifizierung kann auch nur dann zustande kommen, wenn zwischen den Partnern eine Vertrauensbeziehung existiert. Das bedeutet, dass der Nutzer im Vorhinein mit jedem einzelnen Betreiber einen Vertrag abgeschlossen haben muss. Ein derartig aufwendiges Modell kann aber auf keine breite Akzeptanz stoßen. Deswegen ist es wichtig, die häufig anfallenden Authentifizierungsvorgänge ohne die Interaktion des Nutzers durchzuführen. Diese Art von Authentifizierung nennt man *Single Sign-On* [35].

Um Single Sign-On für den Netzwerkzugang zu ermöglichen, müssen Betreiber nutzerbezogene Sicherheitsinformationen austauschen können. Der Nutzer müsste dann nur noch einen einzigen Vertrag abschließen und bräuchte sich bei der Nutzung fremder Netze nicht mehr um die Authentifizierung zu kümmern. Stattdessen würden fremde Netzbetreiber alle benötigten Authentifizierungs- und Autorisierungsinformationen vom Heimatnetzbetreiber beantragen und darauf basierend dem Nutzer den Zugang gewähren oder verwweigern.

Eine Technologie, die das beschriebene Szenario ermöglicht, ist die XML-basierte Security Assertion Markup Language, kurz SAML. Sie definiert eine standardisierte Darstellung für Sicherheitsinformationen und ermöglicht so den Austausch dieser Informationen zwischen den Netzbetreibern. SAML wurde im praktischen Teil der vorliegenden Arbeit eingesetzt und soll deshalb in diesem Kapitel vorgestellt werden.

3.1 Einführung

Die *Security Assertion Markup Language (SAML)* [14] wurde von der *Organization for the Advancement of Structured Information Standards*, kurz *OASIS* [36] entwickelt. OASIS ist ein Konsortium, das Standards für Sicherheit, XML-Konformität, Web Services, und E-Business entwickelt und von zahlreichen namhafte IT-Unternehmen wie IBM, Nokia, RSA, SAP und Microsoft unterstützt wird.

Von der historischen Seite betrachtet, ergab sich folgende Entwicklung, die zum heutigen SAML-Standard führte. Im November 2000 machte das Unternehmen Netegrity bekannt, dass es mit führenden Industrieunternehmen an der Entwicklung eines ersten Standards für sichere E-Commerce-Transaktionen auf Basis der eXtensible Markup Language (XML) arbeiten würde. Diese Kooperation brachte die „Security Services Markup Language“ (S2ML) hervor und bot damit eine Sprache für den Austausch von Authentifizierungs- und Autorisierungsinformationen bei Transaktionen zwischen Unternehmen und Endusern (Mitarbeiter, Kunden) im Rahmen von B2B und B2C. Zeitgleich kündigte die Firma Securant Technologies die Gründung einer offenen Arbeitsgruppe zur Definition eines XML-basierten Standards für Web-Security an. Dieser Standard wurde unter dem Namen „Authentication and Authorization in XML“ (AuthXML) veröffentlicht. Ziel war die Entwicklung eines allgemein akzeptierten Standards, der unternehmens- und plattformübergreifende Authentifizierungs- und Autorisierungsfunktionalität bieten sollte.

Beide Gruppen reichten ihre Spezifikationen beim OASIS Konsortium ein, das aus einer Kombination von S2ML und AuthXML sowie unter Mitwirkung weiterer Partner das SAML Protokoll entwickelte. Im Januar 2001 fand das erste offizielle Meeting zur Standardisierung von SAML statt und im November 2002 wurde SAML in der Version 1.0 veröffentlicht. Zurzeit liegt SAML in der Version 2.0 vor und hat im Vergleich zur ersten Version umfangreiche Erweiterungen erfahren.

3.2 Die SAML Spezifikation

3.2.1 Konzeptioneller Überblick über SAML

SAML ist ein XML-basierter Standard, der die domänenübergreifende Übermittlung von sicherheitsrelevanten personenbezogenen Informationen ermöglicht. Diese Informationen können neben Authentifizierungs- und Autorisierungs- auch Attributsdaten, wie Telefonnummern oder postalische Anschriften sein. Um den Nutzen von SAML zu veranschaulichen, hat OASIS eine Reihe von Nutzfällen erstellt. Der wichtigste Nutzfall

stellt eine Lösung des Single Sign-On Problems dar und wird in [37] ausführlich beschrieben. Hierbei werden zwischen zwei Parteien, als Asserting Party und Relying Party bezeichnet, Daten auf Basis des *Simple Object Access Protocoll (SOAP)* [38] über HTTP ausgetauscht. In SAML 2.0 werden Asserting Party als Identity Provider und Relying Party als Service Provider bezeichnet. Um einheitliche Begriffe zu haben, werden im Folgenden die neuen Namen verwendet. Konzeptionell gliedert sich der SAML in vier Komponenten:

- SAML Assertions
- SAML Protokoll
- SAML Binding
- SAML Profile.

SAML Assertions bilden einen Rahmen, um Informationen, beispielsweise über einen Nutzer, zwischen Identity Provider und Service Provider auszutauschen. Basierend auf dem SAML Protokoll (Request-Response) sendet ein Service Provider eine SAML-Request Nachricht an einen Identity Provider, um Nutzerinformationen anzufordern. Der Identity Provider antwortet mit einem SAML-Response, welcher die angeforderten Informationen innerhalb einer Assertion enthält. Die Request- und Response-Nachrichten selbst werden in SOAP-Envelopes eingebettet (SAML Binding), welche zwischen den beteiligten Providern ausgetauscht werden (SAML Profile). In den folgenden Abschnitten wird auf einige Teilbereiche des SAML-Rahmenwerks im Detail eingegangen.

3.2.2 SAML Assertions

Assertions, also vertrauenswürdige Aussagen über einen Sachverhalt, sind der Kern von SAML. SAML Assertions sind XML-Dokumente, die Informationen über den Authentifizierungsstatus eines Nutzers, seine Eigenschaften und seine Zugangsberechtigungen enthalten können. Dementsprechend gibt es drei Arten von SAML Assertions:

- *Authentifizierungs-Assertions*: bestätigen, dass sich der Nutzer erfolgreich beim System authentifiziert hat,
- *Attribut-Assertions*: bestätigen, dass einem Nutzer bestimmte Attribute (Rollen, Funktionen) oder dynamische Attribute (z.B. Kontostand-Informationen) zugeordnet sind. Attributinformationen spielen bei der Zuweisung von Zugangsberechtigungen eine wichtige Rolle,
- *Autorisierungs-Entscheidungs-Assertions:* enthalten Informationen über die Berechtigungen eines Nutzers, beziehungsweise legen fest, wie auf eine spezifische Ressource zugegriffen werden darf.

Es ist jedoch auch möglich, in einer SAML Assertion gleichzeitig Authentifizierungs-, Attributs- und Autorisierungsinformationen zu hinterlegen. Sie werden in Statements gespeichert, von denen es vier Arten gibt (vgl. Abbildung 12):

- *AuthenticationStatements*, die die Authentifizierung einer Person bestätigen,
- *SubjectStatements*, die die Identität der Person beglaubigen,
- *AttributeStatements*, die bescheinigen, dass dem Subjekt verschiedene Attribute mit bestimmten Werten zugeordnet sind,
- *AuthorizationDecisionStatements*, die Entscheidungen der ausstellenden Autorität über die Erlaubnis des Zugangs auf eine gesicherte Ressource enthalten.

Neben Statements muss jede SAML Assertion folgende Elemente enthalten:

- *MajorVersion* und *MinorVersion*, die die Version des verwendeten SAML Protokolls angeben (aktuell 2.0),
- *Issuer*, der die Identität der ausstellenden Autorität enthält,
- *AssertionID*, in dem der ein eindeutige Bezeichner der Assertion abgelegt wird,
- *IssueInstant*, das den genauen Zeitpunkt der Ausstellung der Assertion angibt.

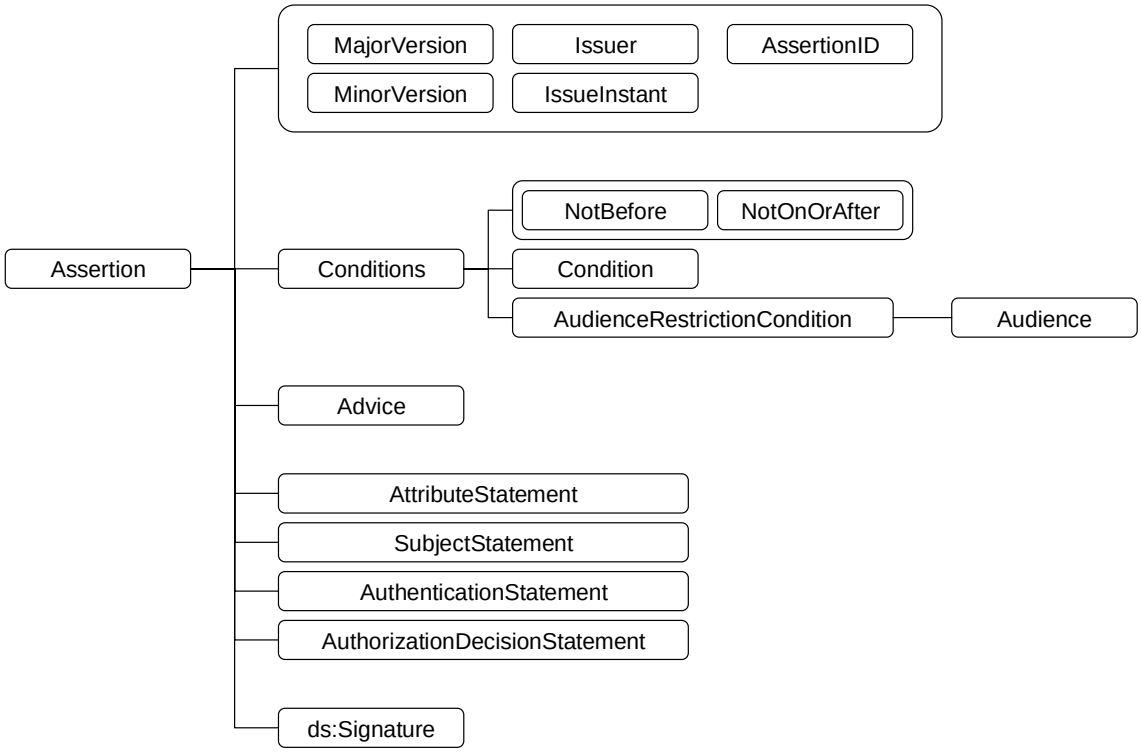


Abbildung 12: Struktur einer SAML Assertion

Zusätzlich können in Assertions folgende optionale Elemente enthalten sein:

- *Conditions*, in denen Bedingungen für die Gültigkeit von Assertion angegeben werden (z.B. Zeitintervall der Gültigkeit),
- *Advice*, in dem Hinweise für die Verarbeitung der Assertion hinterlegt sind,
- *ds:Signature*, zeigt an, dass eine digitale Signatur für den Nachweis der Echtheit der Assertion verwendet wird.

Um überhaupt verwertbar zu sein, müssen SAML Assertions von vertrauenswürdigen Autoritäten ausgestellt werden. Dabei können einzelne Autoritäten bevollmächtigt sein, sowohl Assertions mit Authentifizierungsinformationen, als auch solche mit Autorisierungs- und Attributsinformationen auszugeben. Es ist aber auch möglich, eine Autorität auf einen Bereich einzuschränken und ihr nur die Ausstellung einer einzigen Art von Assertions, beispielsweise Authentifizierungs-Assertions, zu erlauben. Eine solche Assertion wird im folgenden Beispiel vorgestellt:

```
<?xml version="1.0" encoding="UTF-8"?>
<saml:Assertion AssertionID="123456"
      IssueInstant="2005-05-13T17:33:55Z"
      Issuer="BMW AG"
      MajorVersion="2" MinorVersion="0"
      xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
       <saml:Conditions
            NotBefore= "2005-05-13T17:33:55Z"
            NotOnOrAfter="2005-05-13T17:53:55Z">
       </saml:Conditions>
       <saml:AuthenticationStatement
             AuthenticationInstant="2005-05-13T17:33:56Z"
             AuthenticationMethod="urn:oasis:names:tc:SAML:Password">
             <saml:Subject>
                   <saml:NameIdentifier
                      SecurityDomain=www.bmw.de
                      Name="Thomas Hey">
            </saml:Subject>
      </saml:AuthenticationStatement>
</saml:Assertion>
```

Diese Assertion enthält folgende Informationen: Sie wurde am 13. Mai 2005 um 17.33 Uhr UTC-Zeit von der BMW AG ausgestellt und ist nur zwischen 17.33 und 17.53 Uhr gültig.

Die Assertion bestätigt, dass sich das Subjekt namens Thomas Hey aus der Sicherheitsdomäne www.bmw.de mit Hilfe eines Passworts authentifiziert hat. Basierend auf dieser Assertion können Autoritäten, die mit der BMW AG eine Vertrauensbeziehung eingegangen sind, den Nutzer Thomas Hey als authentifiziert ansehen.

3.2.3 Das SAML Protokoll

Das SAML Protokoll definiert, wie SAML Assertions ausgetauscht werden. Jeder Austausch von SAML Informationen basiert auf bestimmten Voraussetzungen, wie einer vertrauenswürdigen Beziehung zwischen dem Identity Provider und dem Service Provider. Des Weiteren müssen sich beide Parteien auf das gleiche Subjekt beziehen, daher müssen die Namen innerhalb einer Sicherheitsdomäne immer eindeutig sein.

Das Protokoll definiert mittels XML-Schema zwei Arten von Nachrichten: *Request* und *Response*. Die wichtigsten Request-Typen sind:

- *AuthenticationQuery*, für die Anfrage nach Authentifizierungsinformationen,
- *AttributeQuery*, für die Frage nach den Eigenschaften des Subjekts,
- *AuthorizationDecisionQuery,* erwartet eine Entscheidung, ob dem Zugriff gewährt werden soll.

Auf jeden Request muss ein Response erfolgen. Für alle Anfragetypen gibt es lediglich einen Response-Typ und dieser enthält stets eine Assertion, die als XML-Element im Response eingefügt ist:

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response ResponseID="456123"
      IssueInstant="2005-05-13T17:36:55Z"
      InResponseTo="456123"
      MajorVersion="2" MinorVersion="0">
       <samlp: Status>
             <samlp:StatusCodeValue="samlp:Success"/>
       </samlp:Status>
       <saml: Assertion ...>
             ..............
       </saml: Assertion>
</samlp: Response>
```

Der nächste Abschnitt zeigt wie diese Nachrichten zusammen mit den vorher definierten Assertions die einmalige Authentifizierung (Single Sign-On) eines Nutzers ermöglichen.

3.3 Single Sign-On Authentifizierung mittels SAML

SAML wurde in dieser Arbeit eingesetzt, um mobilen Nutzern beim Wechsel der Netzbetreiber die wiederholte Anmeldung zu ersparen. Stattdessen soll sich der Nutzer nur ein einziges Mal beim Netzbetreiber seiner Wahl anmelden und danach auch Zugang zu allen anderen Netzen erhalten. Dieser Vorgang heißt Single Sign-On und ist eine der Kernaufgaben von SAML. In nächsten Abschnitt soll deswegen gezeigt werden, wie SAML diese Art von Authentifizierung bewerkstelligt.

SAML verwendet dafür Authentifizierungs-Assertions, die je ein Authentication Statement enthalten müssen. Mit ihnen wird bestätigt, dass ein Subjekt, hier der mobile Nutzer, von einer vertrauenswürdigen Autorität, dem Identity Provider oder einer Drittinstanz, zum angegebenen Zeitpunkt durch ein bestimmtes Verfahren erfolgreich authentifiziert wurde. Ein solches Statement wird im Rahmen der Authentifizierungs-Assertion an den Service Provider übertragen. Die Abbildung 13 auf der nächsten Seite veranschaulicht den Austausch der Authentifizierungsinformationen zwischen dem Identity Provider und dem Service Provider. Wie man erkennen kann, lässt sich die Authentifizierung des Nutzers in zwei Phasen untergliedern:

1. Herkömmliche Authentifizierung beim Identity Provider
2. Authentifizierung mittels einer Assertion beim Service Provider

Im ersten Schritt findet eine Authentifizierung des Nutzers beim Identity Provider statt. Dafür kann beispielsweise ein passwortbasiertes Verfahren eingesetzt werden. Nach erfolgreichem Abschluss der Authentifizierung erstellt der Identity Provider eine Authentifizierungs-Assertion, welche das AuthenticationStatement enthält.

Beim Zugriff auf einen Dienst aus einer anderen Domäne erfolgt dort die Authentifizierung des Nutzers nicht etwa durch eine explizite Anmeldung, sondern mittels der Authentifizierungs-Assertion und des darin enthaltenen AuthenticationStatement. SAML bietet für die Übertragung der Assertion zwei Varianten, die im Folgenden vorgestellt werden sollen.

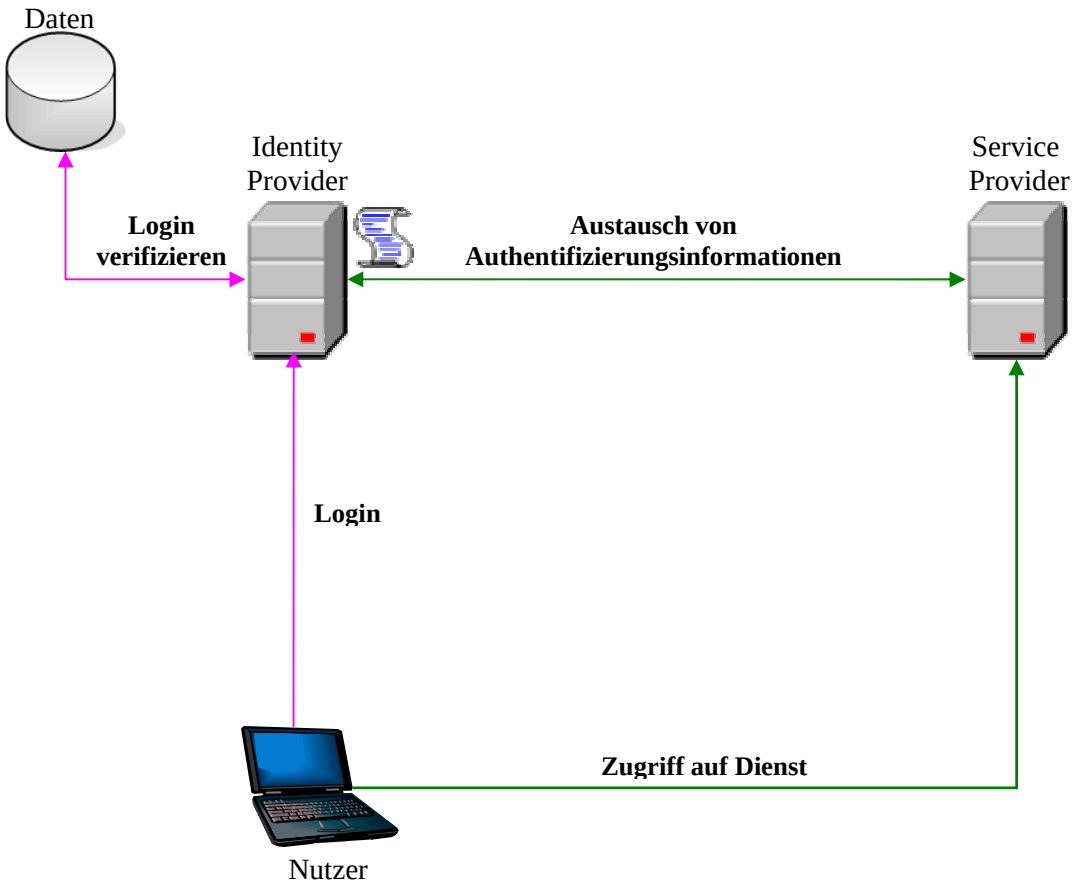


Abbildung 13: Authentifizierung eines Nutzers mittels SAML

3.3.1 Das Artifact-Profil

Im Rahmen des Artifact-Profils wird dem Service Provider eine Referenz auf die Authentifizierungs-Assertion übermittelt, welche als *SAML-Artifact* bezeichnet wird. Nach dem Erhalt einer solchen Referenz muss der Service Provider diese eindeutig auflösen. Innerhalb des Artifacts wird ein SourceID-Element übertragen, um den Ursprung des Artifacts festzuhalten. Jedem Identity Provider ist eine eindeutige SourceID zugeordnet. Für die Konstruktion dieser ID empfiehlt SAML den SHA-1-Hash aus einer Identifikations-URL, welche der Identity Provider selbst wählt (z.B. der Domänenname). Die SAML Spezifikation setzt voraus, dass jeder Service Provider eine Tabelle mit derartigen SourceID-Einträgen besitzt, mittels derer überprüft werden kann, ob das empfangene Artifact von einer vertrauenswürdigen Instanz gesendet wurde. Stimmen Tabelleneintrag und angegebene SourceID nicht überein, darf der Service Provider nicht versuchen die Referenz aufzulösen. In diesem Fall schlägt die Authentifizierung des externen Nutzers fehl. Der Ablauf des Artifact-Profils lässt sich anhand eines Sequenzdiagramms anschaulich darstellen (vgl. Abbildung 14).

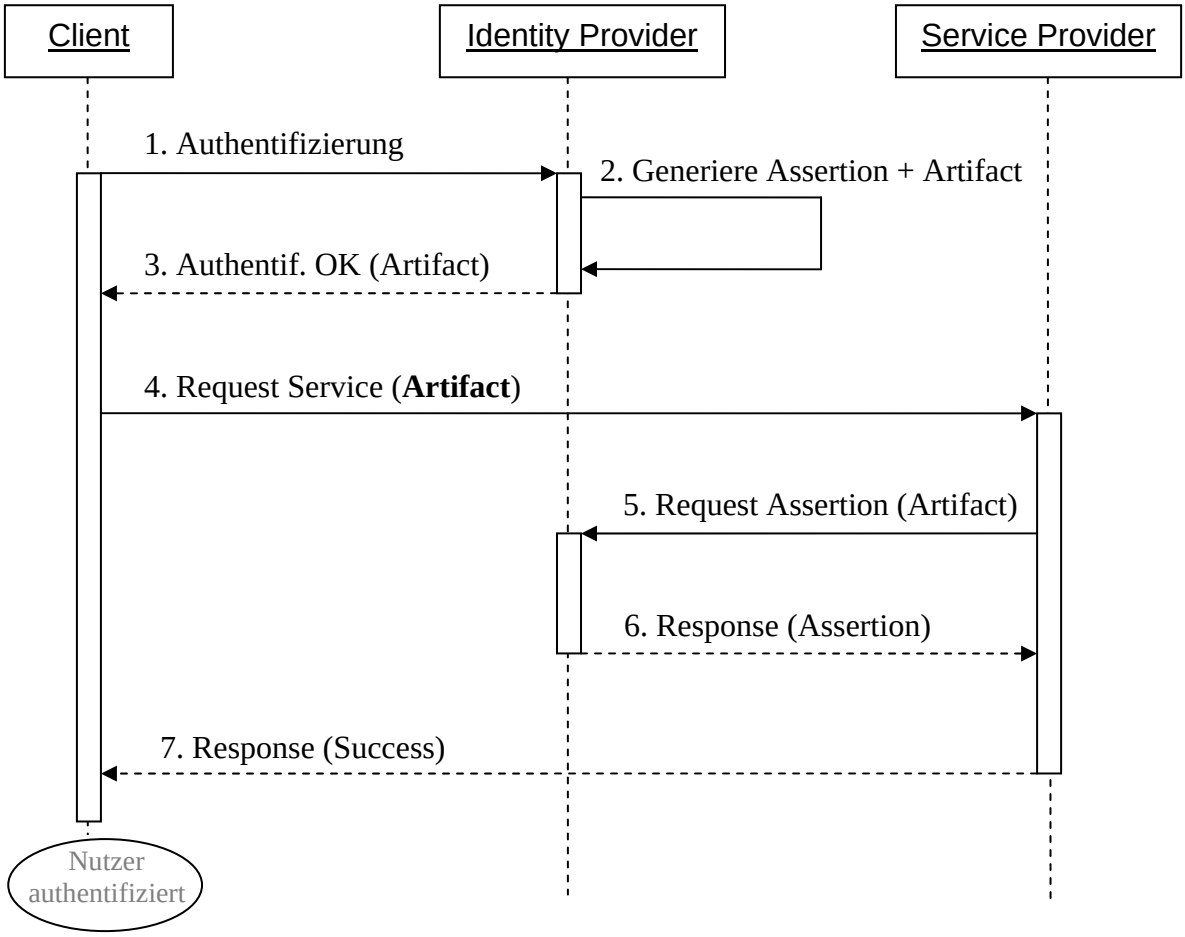


Abbildung 14: Authentifizierung gemäß Artifact-Profil

1. Im ersten Schritt findet die Anmeldung und Authentifizierung des Nutzers beim Identity Provider statt.
2. Bei erfolgreicher Authentifizierung generiert der Identity Provider eine Authentifizierungs-Assertion und ein zugehöriges Artifact.
3. Im dritten Schritt wird dem Client des Nutzers das Artifact übermittelt.
4. Wenn der Nutzer einen Dienst des Service Providers in Anspruch nehmen will, muss er sich nicht erneut anmelden. Stattdessen sendet der Client ohne die Interaktion des Nutzers das Artifact an den Service Provider
5. Um die Referenz (Artifact) auf die Authentifizierungs-Assertion aufzulösen, sendet der Service Provider eine SAML-Request Nachricht an den Identity Provider.
6. Die Antwort des Identity Provider enthält, sofern die angegebenen Referenzen aufgelöst werden konnten, die zugehörige Authentifizierungs-Assertion. Bei negativer Beantwortung des Requests wird ein Statuscode (Fehlercode) übertragen.

7. Im abschließenden Schritt sendet der Service Provider nach Erhalt und Verarbeitung der Authentifizierungs-Assertion dem Client eine Antwort, in der der Zugriff auf den angeforderten Dienst erlaubt oder verweigert wird.

Das Artifact Profil ist durch das geringe Datenvolumen des Artifacts, das zwischen dem Client und den anderen Entitäten ausgetauscht wird, gut geeignet für mobile Netzwerke. In solchen Netzwerken kommuniziert der Client mit dem Identity und Service Provider typischerweise mittels drahtloser Übertragungstechnologien. Diese verfügen über eine wesentlich geringere Bandbreite als die drahtgebundenen Netzwerkabschnitte, die Service Provider und Identity Provider miteinander verbinden und die für den Transport der im Vergleich zum Artifact viel größeren Assertion genutzt werden. Aufgrund dieser positiven Merkmalen wird das Artifact Profil eine herausragende Rolle in der zu entwickelnden Authentifizierungsmethode spielen.

3.3.2 Das POST-Profil

Die zweite Möglichkeit Assertions dem Service Provider zugänglich zu machen wird vom POST-Profil definiert. Im Rahmen des POST-Profils kommt der Nutzer in den Besitz der kompletten Assertion und präsentiert diese für die Authentifizierung beim Service Provider. Dadurch benötigt dieses Profil eine geringere Anzahl von Nachrichten als das Artifact-Profil, allerdings ist das zwischen Client und Service Provider übertragene Datenvolumen um ein Vielfaches höher. Der Ablauf wird in dem folgenden Sequenzdiagramm (vgl. Abbildung 15) dargestellt und anschließend erläutert.

1. Im ersten Schritt muss sich der Nutzer, wie im Artifact-Profil, beim Identity Provider anmelden und authentifizieren.
2. Der Identity Provider erstellt nach einer erfolgreichen Authentifizierung eine Assertion.
3. Dem Client des Nutzers wird nun nicht wie beim Artifact-Profil eine Referenz, sondern die gesamte Authentifizierungs-Assertion übermittelt.
4. Wenn der Nutzer einen Dienst des Service Providers in Anspruch nehmen will, übermittelt der Client die Authentifizierungs-Assertion. Eine Involvierung des Nutzers ist dabei nicht notwendig.
5. Im abschließenden Schritt wird der angeforderte Dienst dem Nutzer zur Verfügung gestellt oder es erfolgt eine Forbidden Statusmeldung.

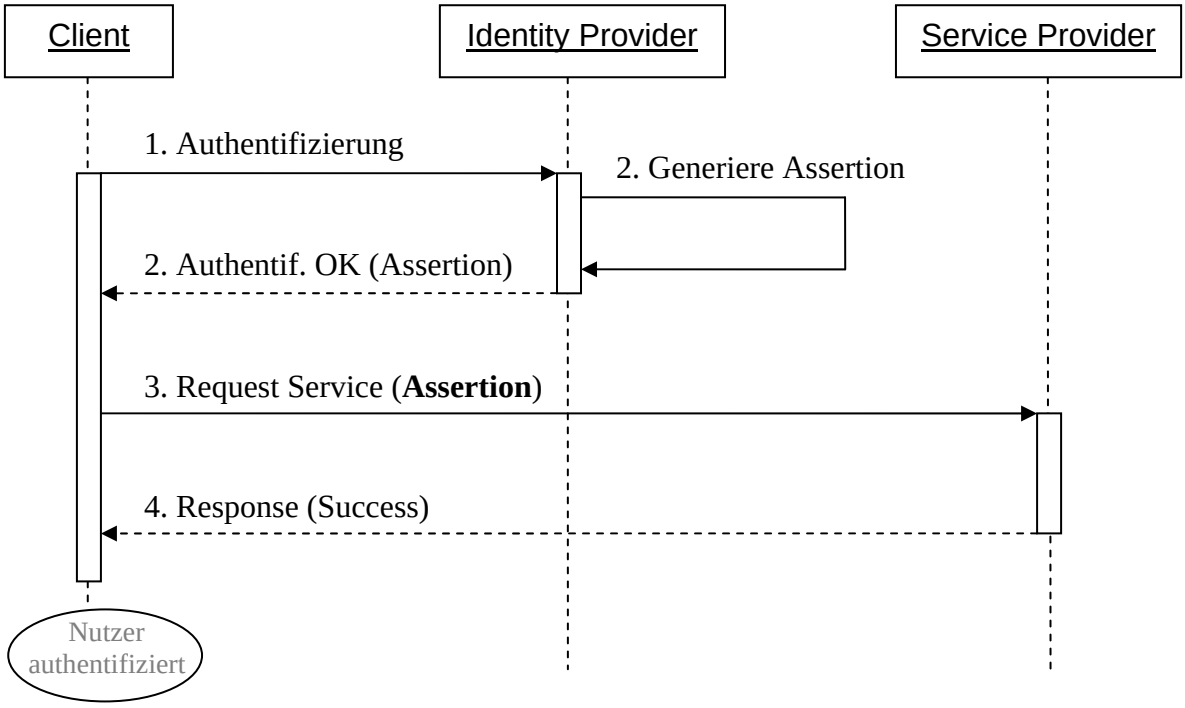


Abbildung 15: Authentifizierung gemäß POST-Profil

Bei einem Authentifizierungsvorgang werden vertrauliche nutzerbezogene Daten über ein unsicheres Medium versendet. In den beiden vorgestellten Profilen werden manche dieser Daten transparent für den Nutzer zwischen verschiedenen Autoritäten ausgetauscht. Dies erhöht das Risiko des Datenmissbrauchs, weswegen SAML über geeignete Mechanismen verfügen muss, um Angriffe zu abzuwehren. Der nächste Abschnitt befasst sich mit den wichtigsten Anforderungen für eine sichere Kommunikation und erläutert wie diese in SAML umgesetzt werden.

3.4 Absicherung von SAML-Nachrichten

Um eine sichere Kommunikation zu realisieren, müssen SAML Nachrichten folgende vier Kriterien erfüllen:

- *Vertraulichkeit*
- *Integrität*
- *Authentizität*
- *Nichtabstreitbarkeit*

In den folgenden Abschnitten wird gezeigt, mit welchen Mitteln SAML die angeführten Anforderungen erfüllt.

3.4.1 Vertraulichkeit

Die Vertraulichkeit der Nachricht und der Daten muss gewährleistet werden, damit nur der wahre Empfänger auf den Inhalt der Nachricht schließen und diesen lesen kann. SAML unterscheidet in diesem Zusammenhang zwischen der Vertraulichkeit, die bei der Übertragung sichergestellt wird, und derjenigen, welche die Nachricht und die darin enthaltenen Informationen betrifft.

Bei der Übertragung erwartet SAML ein sicheres Netzwerkprotokoll, wie das Transport Layer Security Protocol [31] oder das IP Security Protocol [30]. Die Vertraulichkeit der Informationen innerhalb der Nachricht wird mittels XML Encryption [39] gewährleistet, einer Methode, bei der die Daten mit dem öffentlichen Schlüssel des Empfängers verschlüsselt werden. Die Daten können nun nur noch mit dem privaten Schlüssel des Empfängers, der den Schlüssel als Einziger besitzt, entschlüsselt werden.

3.4.2 Integrität

Um eine sichere Kommunikation zu realisieren, muss die Integrität der ausgetauschten Nachrichten gewährleistet werden. Dies bedeutet, dass Nachrichten und die darin enthaltenen Informationen während der Übertragung nicht durch einen Angriff manipuliert werden dürfen, ohne dass der Empfänger in der Lage ist diese Manipulation zu erkennen.

SAML unterscheidet auch hier zwischen der Nachrichtenübertragung und der Nachricht selbst. Für die Übertragung verlangt SAML den Einsatz des Transport Layer Security Protocol oder des IP Security Protocol. Für die Integrität der Daten innerhalb der Nachricht muss XML Signature [40] eingesetzt werden, wodurch die Daten digital signiert werden. Die Signierung der Daten erfolgt mit dem privaten Schlüssel des Absenders, die Überprüfung der Daten wird mit Hilfe des öffentlichen Schlüssels des Absenders durchgeführt. Da nur der Absender in Besitz des privaten Schlüssels ist, kann ein Angreifer nach der Manipulation der Daten diese nicht mit dem richtigen Schlüssel signieren, wodurch die Manipulation beim Empfänger entdeckt wird.

3.4.3 Authentizität

Die Authentizität der SAML Nachrichten muss gewährleistet sein, um den Datenursprung bestimmen zu können. Die fehlende Authentizität der Nachricht kann dazu führen, dass Unbefugte auf IT-Systeme und Daten Zugriff nehmen können und Verursacher von Problemen und Störfällen nicht eindeutig identifiziert werden können. Außerdem wäre es nicht möglich die Herkunft der Daten zu bestimmen.

SAML erlaubt die Authentizität einer Nachricht durch den Einsatz von IPsec oder TLS sicher zu stellen. Befinden sich auf dem Weg zwischen Service Provider und Identity Provider Zwischenstationen, dann reicht diese Absicherung nicht aus. Ein Service Provider muss den Request mit seinem privaten Schlüssel digital signieren, so dass die Authentizität des Requests gewährleistet wird. Diese digitale Signatur muss der Identity Provider mit Hilfe des öffentlichen Schlüssels des Service Providers verifizieren.

3.4.3 Nichtabstreitbarkeit

Der Austausch der Nachrichten muss so erfolgen, dass das Senden der Nachricht und der Daten nicht abgestritten werden kann. Dies ist insbesondere bei der Übermittlung von Authentifizierungs- und Autorisierungsinformationen notwendig. Die Authentizität bildet die Grundlage für die Nichtabstreitbarkeit des Sendens bzw. Empfangs von Nachrichten.

Wie auch die Authentizität, wird die Nichtabstreitbarkeit des Sendens von SAML Nachrichten mittels TLS oder Ipsec ermöglicht. Die Ende-zu-Ende Sicherheit, die notwendig wird, wenn der Nachrichtenaustausch zwischen den Kommunikationspartnern über Zwischenstationen erfolgt, realisiert erneut XML Signature. Eine digitale Signatur vermag nur der Signierende selbst erzeugen und kann daher als Beweis dafür dienen, dass die signierten Informationen von ihm stammen.

3.5 Zusammenfassung

In diesem Kapitel wurde das SAML Protokoll, das eine wichtige Rolle im praktischen Teil der vorliegenden Arbeit spielt, detailliert vorgestellt. Nach einer kurzen Einführung, in der die Entstehungsgeschichte von SAML als Zusammenschluss zweier Protokolle beschrieben wurde, folgte die Vorstellung der SAML Spezifikation und ihrer wichtigsten Bausteine, dem Protokoll und den Assertions. Von den Letzteren wurde insbesondere die Authentifizierungs-Assertion ausführlich behandelt, da sie auch in den im nächsten Kapitel vorzustellenden Szenarien oft zum Einsatz kommen wird. Anschließend wurden zwei SAML Profile vorgestellt, die Single Sign-On ermöglichen. Obwohl die Profile für die Realisierung der einmaligen Anmeldung im Bereich webbasierter Dienste ausgelegt sind, sollen sie im Rahmen dieser Arbeit, mit leichten Abwandlungen, für Single Sign-On beim Netzwerkzugang eingesetzt werden. Im letzten Teil dieses Kapitels wurde eine Sicherheitsanalyse des SAML Protokolls durchgeführt. Es wurde gezeigt, dass SAML mittels kryptografischer Verfahren die Vertraulichkeit, Integrität, Authentizität und Nichtabstreitbarkeit der Nachrichten und der darin transportierten Daten sicherstellen kann.

Dadurch ist SAML dazu geeignet in dieser Arbeit für den Austausch von Authentifizierungsdaten über administrative Domänen hinweg eingesetzt zu werden.

4. Systementwurf

Im Rahmen dieser Arbeit wurden in den beiden vorangehenden Kapiteln Protokolle vorgestellt, die den Netzwerkzugang eines mobilen Nutzers unabhängig von der eingesetzten Zugangstechnologie ermöglichen, seine Authentifizierung über unterschiedliche Methoden erlauben und den Transport dieser Authentifizierungsdaten innerhalb heterogener Infrastrukturen unterstützen. Ferner wurde eine Technologie eingeführt, mittels derer Nutzer sich beim Zugriff auf Dienste fremder Sicherheitsdomänen nicht neu anmelden müssen, sondern durch den Austausch von Sicherheitsinformationen authentifiziert werden, ohne dass eine Nutzerinteraktion nötig wäre.

In diesem Kapitel wird, basierend auf den vorgestellten Technologien, ein Konzept für eine Authentifizierungsmethode erarbeitet, die für den Einsatz in den mobilen Netzen der vierten Generation geeignet ist. Bei der Methode wird es sich um eine Erweiterung des EAP-Protokolls handeln, dem Elemente aus SAML hinzugefügt werden, um Single Sign-On für den Netzwerkzugang zu ermöglichen. Der Name der neuen Authentifizierungsmethode wird deswegen *EAP-SAML* lauten. Im ersten Teil des Kapitels werden einige Szenarien vorgestellt, die diese Authentifizierungsmethode benötigen. Im zweiten Abschnitt werden aus den Szenarien Anforderungen abgeleitet und nach deren Analyse eine Systemarchitektur aufgestellt. Abschließend werden die Interaktionen zwischen den definierten Systemkomponenten aufgezeigt, wie sie innerhalb der beschriebenen Szenarien stattfinden können.

4.1 Anwendungsszenarien

Mobile Netze der vierten Generation werden in erster Linie durch ihre Heterogenität charakterisiert sein. Es wird ein Nebeneinander von zahlreichen Übertragungstechnologien, wie UMTS, GPRS, WLAN und Bluetooth geben. Diese Netze werden von unterschiedlichen Organisationen betrieben werden, so dass der Markt der Netzbetreiber stark fragmentiert sein wird. Manche Netzbetreiber werden sich aus finanziellen Gründen oder aufgrund gesetzlicher Regulierung auf bestimmte Übertragungstechnologien und/oder auf einzelne geografische Gebiete beschränken müssen. Somit wird ein Kunde, der ein Fahrzeug nutzt, bedingt durch seine hohe Mobilität

häufig nicht auf die Dienste seines Netzbetreibers zugreifen können und wird deshalb die Infrastruktur fremder Betreiber nutzen müssen. Im Folgenden sollen Szenarien aufgezeigt werden, die die Authentifizierung des Nutzers in verschiedenen Umgebungen betrachten.

4.1.1 Neuanmeldung eines Nutzers

Die Neuanmeldung eines mobilen Nutzers ist relativ einfach. Die an der Neuanmeldung beteiligten Komponenten sind das Endgerät des Nutzers, der Network Access Server, der dem Nutzer während des Anmeldevorgangs einen eingeschränkten Netzwerkzugang gewährt, der Authentifizierungsserver und eine an den Authentifizierungsserver angebundene Komponente, die die notwendigen Authentifizierungsinformationen enthält. Es wird angenommen, dass zwischen dem Nutzer und dem Netzbetreiber ein Vertrauensverhältnis besteht, beispielsweise durch den Abschluss eines Vertrags. Dadurch verfügt der Nutzer über die für die Authentifizierung notwendigen Zugangsdaten. Für den Authentifizierungsvorgang bei der Neuanmeldung sind alle Authentifizierungsmethoden geeignet. An dieser Stelle sollen zwei Varianten vorgestellt werden.
Die am einfachsten zu realisierende Variante ist das passwortbasierte Verfahren, bei dem der Nutzer ein nur ihm und dem Authentifizierungsserver bekanntes Passwort vorweisen muss (vgl. Abbildung 16).

Der Nutzer übermittelt seine Kennung und das zugehörige Passwort an den Authentifizierungsserver, dieser vergleicht sie mit den Informationen, die in der Datenbank gespeichert sind und schließt die Authentifizierung gegebenenfalls erfolgreich ab. Passwortbasierte Verfahren sind leider anfällig für Wörterbuch-Attacken und Wiederholungsangriffen, weswegen sie immer seltener zum Einsatz kommen.

Ist eine höhere Sicherheit und eine beidseitige Authentifizierung erforderlich, kann die Authentifizierung mittels Zertifikaten eingesetzt werden. Voraussetzung dafür ist jedoch eine bereits installierte *Public Key Infrastructure (PKI)* [20], die als „Trusted Third Party“, also als vertrauenswürdiger Dritter fungiert. Aufgabe der PKI ist die Erzeugung, Verwaltung und Verteilung von Private/Public-Key Schlüsselpaaren und die Erstellung von digitalen Zertifikaten, die Identitäten mit diesen Schlüsseln verknüpfen. Eine Entität, die an eine PKI angeschlossen ist, kann eine andere Entität, die sich mit einem von der PKI ausgestellten Zertifikat anmeldet, erfolgreich authentifizieren. Ein Nutzer und ein Server, die an die gleiche PKI angeschlossen sind, können sich somit mittels digitaler Zertifikate gegenseitig authentifizieren.

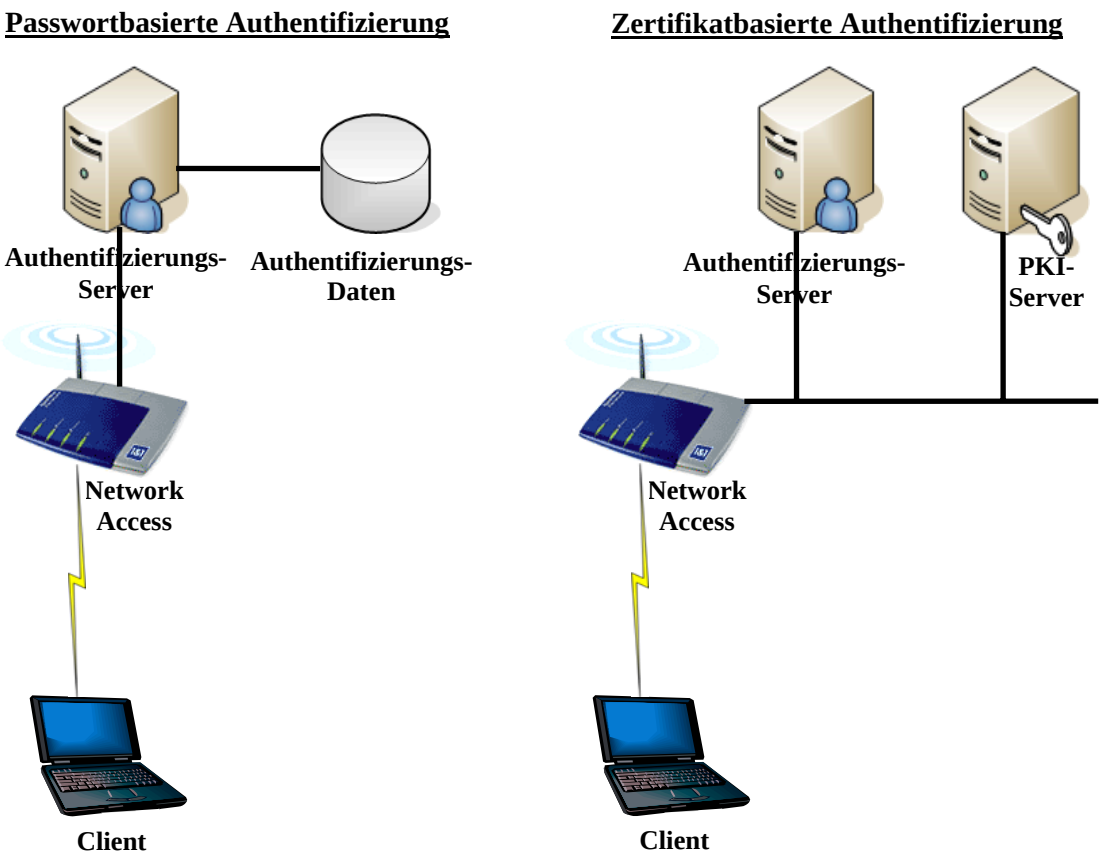


Abbildung 16: Neuanmeldung eines Nutzers

Da der Datenverkehr mit dem aus dem Zertifikat stammenden öffentlichen Schlüssel des jeweiligen Kommunikationspartners verschlüsselt ist, bietet dieses Verfahren eine sehr hohe Sicherheit. Von Nachteil sind jedoch der hohe Aufwand und die damit verbundenen Kosten, die eine PKI verursacht. Der Authentifizierungsvorgang läuft folgendermaßen ab: Der Authentifizierungsserver beantragt vom PKI-Server ein Zertifikat und sendet es an den Client. Dieser prüft das Server-Zertifikat, beantragt seinerseits ein Zertifikat vom PKI-Server und übermittelt es an den Authentifizierungsserver. Nach der erfolgreichen Überprüfung des Client-Zertifikats, ist der Nutzer authentifiziert.

4.1.2 Reauthentifizierung in der Heimatdomäne

In einem mobilen Netzwerk ist es aus Sicherheitsgründen notwendig, dass sich Nutzer periodisch oder beim Übertritt in einen anderen administrativen Bereich des Netzes neu authentifizieren. Das ist beim heute verbreiteten GSM-System [3] für den Nutzer nicht weiter störend, da er bei dieser Art von Authentifizierung nicht involviert wird und von dem Vorgang keinerlei Kenntnis erhält. Stattdessen wird die Authentifizierung zwischen dem Authentifizierungsserver des Betreibers und der im Mobiltelefon integrierten SIM-Karte durchgeführt. Diese Lösung bedeutet jedoch eine starke Einschränkung der

verwendbaren Geräte, da jedes Endgerät über eine SIM-Karte verfügen muss. Außerdem ist das Gerät durch die SIM-Karte an eine einzige Identität gebunden und kann nicht von anderen Identitäten auf eigene Rechnung benutzt werden.

In Netzen der vierten Generation werden aber nicht nur Mobiltelefone, sondern auch Endgeräte zum Einsatz kommen, die über keine SIM-Karten verfügen und von verschiedenen Nutzern verwendet werden sollen. So ist es vorstellbar, dass zum Beispiel in einem Fahrzeug biometrische Verfahren zum Einsatz kommen, über die sich Nutzer authentifizieren können. Ebenso ist es möglich, dass manche Kunden die Erfassung ihrer biometrischen Daten ablehnen und sich stattdessen über wissensbasierte Verfahren authentifizieren möchten. In diesen Fällen gibt es keinen allgemeingültigen Berechtigungsnachweis, wie ihn die SIM-Karte in GSM Netzen darstellt, und die Reauthentifizierung eines Nutzers müsste auf eine gewöhnliche Authentifizierung wie im Abschnitt 4.1.1 zurückgeführt werden. Das würde einer Neuanmeldung des Nutzers gleichkommen, was aufgrund des hohen Aufwands für den Nutzer nicht praktikabel ist und durch eine anwenderfreundlichere Variante ersetzt werden muss.

4.1.3 Authentifizierung in einer Partnerdomäne

Wie am Anfang dieses Kapitels bemerkt, werden in Netzen der vierten Generation Nutzer ihre Heimatdomäne oft verlassen und Dienste in fremden Domänen anfordern. Das stellt eine Herausforderung dar, da sich Nutzer und Betreiber nicht kennen und somit auch keine Verträge bestehen, die eine Abrechnung zwischen ihnen ermöglichen würde. Heutzutage taucht dieses Problem bei Mobilfunknutzern auf, die ihr Mobiltelefon im Ausland einsetzen wollen. Voraussetzung dafür ist, dass zwischen dem Betreiber des Heimatnetzes und dem ausländischen Betreiber eine Partnerschaft besteht, die zum Beispiel durch einen Roaming-Vertrag geregelt wurde. Bei der Authentifizierung des Nutzers in der Partnerdomäne muss der Partner auf das Authentifizierungsmodul des Heimatnetzbetreibers zugreifen können, um nutzerbezogene Daten zu erlangen. Die Herausgabe solch vertraulicher Daten ist in der Regel problematisch, weswegen das Telefonieren im Ausland noch lange nicht überall möglich ist.

In der Abbildung 17 wird ein Szenario veranschaulicht, in dem ein mobiler Nutzer einen Domänenwechsel von seiner Heimatdomäne zu einer Partnerdomäne vollzieht. Die beteiligten Entitäten sind der an einem Endgerät angemeldete Nutzer, die Network Access Server der Heimatdomäne und der Partnerdomäne, die beiden Authentifizierungsserver und die Datenbanken, die die jeweiligen Authentifizierungsdaten beinhalten. Beim Domänenwechsel entfernt sich der Nutzer so weit von der Heimatdomäne, dass die Verbindung nicht mehr aufrechterhalten werden kann. Er muss bei einem anderem

Betreiber Netzwerkzugang anfordern. Der Betreiber der neuen Domäne verlangt die Informationen über die Identität des Nutzers und die zugehörigen Authentifizierungsdaten. Der Nutzer übermittelt beides, allerdings stellt der Betreiber anhand der Nutzer-Identität fest, dass er die Authentifizierung nicht selbst durchführen kann und sendet stattdessen die Daten an die Heimatdomäne des Nutzers, mit der er bereits eine Vertrauensbeziehung gehabt haben muss. Der Authentifizierungsserver der Heimatdomäne muss nun über einen sicheren Tunnel mit dem Nutzer kommunizieren können, um ihn zu authentifizieren. Der Betreiber der Partner-Domäne darf die ausgetauschten Daten nicht lesen oder ändern, seine Rolle beschränkt sich auf die Weiterleitung der Nachrichten. Beim erfolgreichen Abschluss des Authentifizierungsvorganges sendet der Authentifizierungsserver der Heimatdomäne eine entsprechende Meldung an den Partner-Authentifizierungsserver. Daraufhin wird dem Nutzer der Zugang zum Netzwerk gewährt.

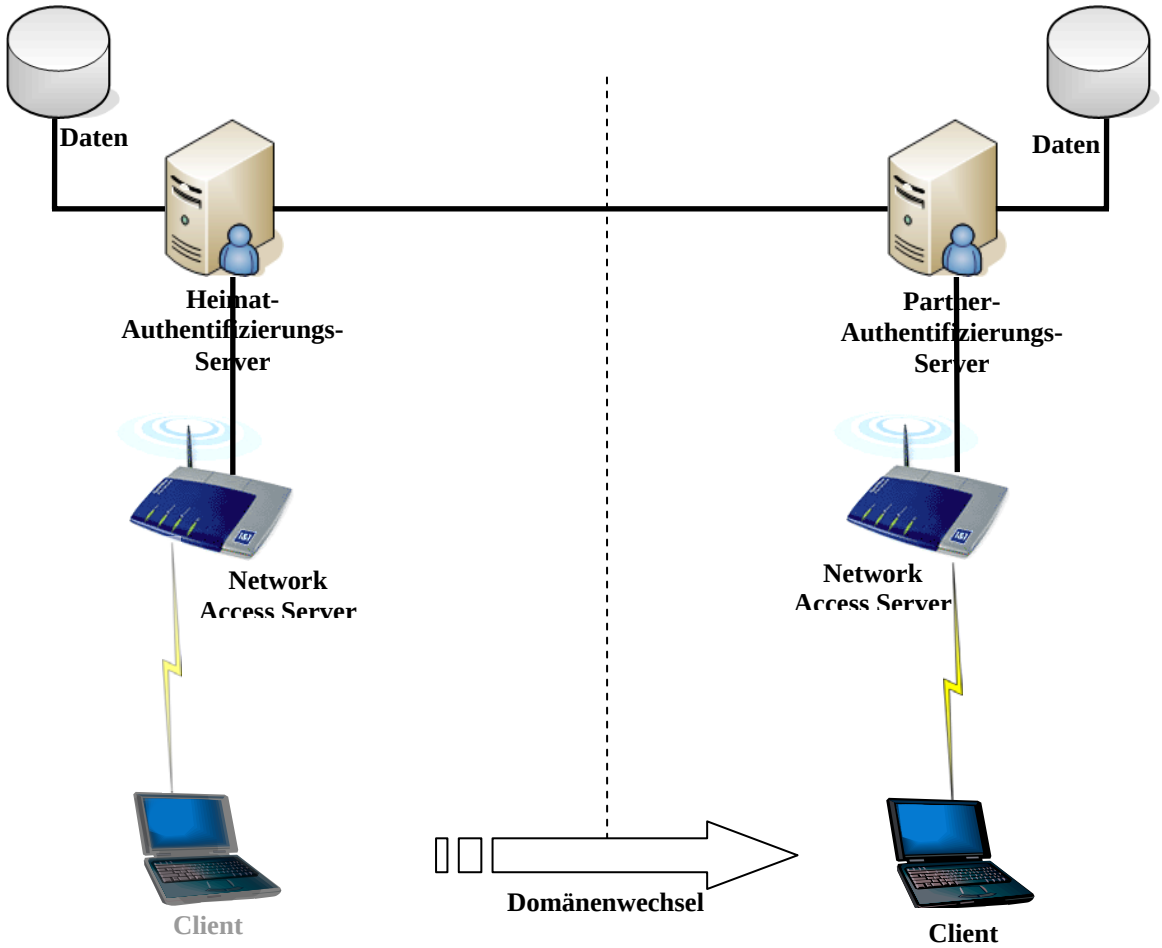


Abbildung 17: Authentifizierung in einer Partnerdomäne

4.1.4 Authentifizierung in einer Föderation

Partnerschaften, die auf bilateralen Verträgen zwischen Betreibern basieren, ermöglichen es heutzutage vielen Mobilfunknutzern über Grenzen hinweg erreichbar zu sein. Bei jeweils einer Handvoll Betreibern in jedem Land ist der dafür praktizierte Verwaltungsaufwand noch vertretbar. In Zukunft jedoch, wenn sogar in jeder Region eine Vielzahl von Netzbetreibern ihre Dienste anbieten werden, sind bilaterale Vertragsabschlüsse zwischen allen Betreibern nicht mehr praktikabel. Dennoch muss Nutzern auch in Netzen der vierten Generation unabhängig von ihrem Aufenthaltsort die Nutzung von Diensten ermöglicht werden.

In diesem Abschnitt soll ein Szenario vorgestellt werden, in dem ein Nutzer sich Zugang zu einem Netz verschaffen will, mit dessen Betreiber sein eigener Heimatnetzbetreiber keinerlei Abmachungen getroffen hat. Um das fehlende Vertrauen zwischen den beiden Betreibern zu kompensieren, wird in diesem Szenario ein vertrauenswürdiger Dritter benötigt. Seine Aufgabe besteht darin, mit möglichst vielen Betreibern Vertrauensbeziehungen einzugehen und so eine Föderation von Domänen zu erschaffen. Der vertrauenswürdige Dritte soll deswegen von nun an *Föderierungs-Manager* genannt werden. Der größte Vorteil eines Föderierungs-Managers ist der wesentlich geringere Verwaltungsaufwand, den er ermöglicht. Da nun die Domänen nicht mehr paarweise ihre Abmachungen treffen müssen, sondern jede nur noch mit dem Föderierungs-Manager interagieren muss, müssen bei n Domänen nicht mehr $n(n-1)$ sondern nur noch n Verträge abgeschlossen werden. Des Weiteren können Föderationen leicht erweitert werden, da ein neuer Betreiber nicht mehr mit allen anderen Betreibern Verträge abschließen muss. Schließlich ist es in einer Föderation viel leichter eventuelle Regeländerungen umzusetzen, was die Flexibilität dieses Konstrukts weiter erhöht.

Der Einsatz eines Föderierungs-Managers wirft allerdings ein Problem auf: Betreibern fehlt die Kontrolle darüber wer Mitglied in der Föderation wird und somit zur Interaktion mit allen anderen Domänen befähigt ist. Die Betreiber sollen nicht darauf vertrauen müssen, dass nur ihnen genehme Organisationen in die Föderation aufgenommen werden. Sie müssen darauf Einfluss nehmen können, wie sie unter welchen Umständen und mit welchen Föderationsmitgliedern zusammenarbeiten wollen. Im Laufe dieses Kapitels soll gezeigt werden, wie dieses Problem gelöst werden kann.

Die Abbildung 18 zeigt die Authentifizierung eines Nutzers in einer fremden Domäne, die mit seiner Heimatdomäne föderiert ist. Die beteiligten Komponenten sind die Authentifizierungsserver der beiden Domänen, ihre Authentifizierungsdatenbanken, die

Network Access Server der beiden Netze, der Nutzer und als oberste und zentrale Instanz der Föderierungs-Manager. Der Authentifizierungsvorgang läuft folgendermaßen ab: der Authentifizierungsserver der fremden Domäne verlangt die Authentifizierung des Nutzers. Dieser sendet seine Identitätsinformationen an den Server, welcher daraufhin feststellt, dass der Nutzer einer anderen Domäne angehört und deshalb seine Daten an den Föderierungs-Manager weiterleitet. Der Föderierungs-Manager findet den Authentifizierungsserver der Heimatdomäne und sendet ihm die Identitätsdaten. Zwischen dem Authentifizierungsserver und dem Nutzer wird ein sicherer Tunnel aufgebaut und es erfolgt die Authentifizierung. Weder der Föderierungs-Manager noch der fremde Betreiber dürfen die Daten lesen, sie sind nur für ihre Weiterleitung zuständig. Beim erfolgreichen Abschluss der Authentifizierung sendet der Heimatserver eine entsprechende Meldung an den Föderierungs-Manager, der sie dem fremden Netzbetreiber übermittelt. Der Nutzer ist somit erfolgreich authentifiziert und darf auf die Netzwerkressourcen des fremden Netzes zugreifen.

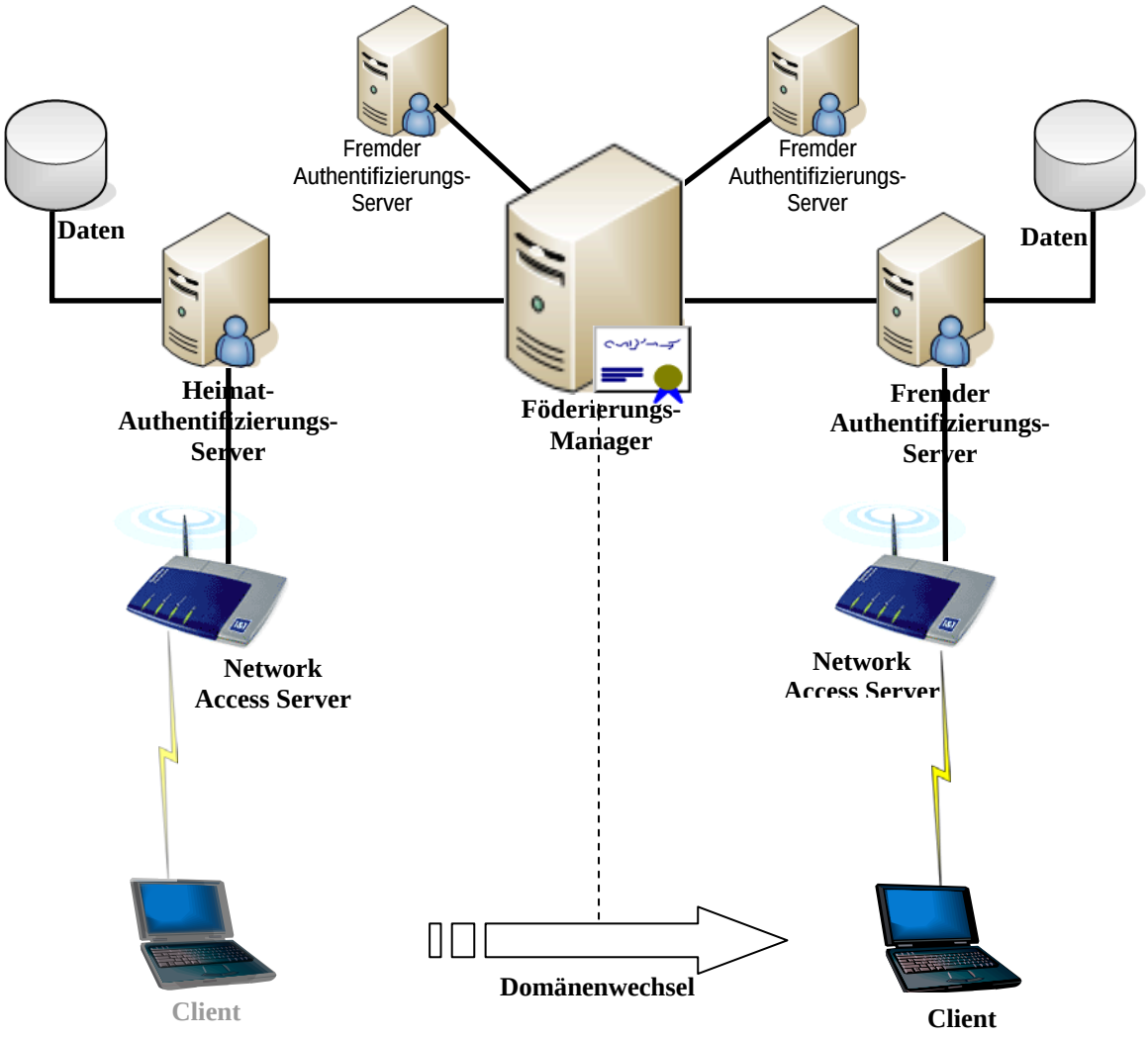


Abbildung 18: Authentifizierung in einer Föderation

Das letzte Szenario ist das komplexeste, das in dieser Arbeit behandelt wird und stellt damit auch die höchsten Ansprüche an das zu realisierende System. Im nächsten Abschnitt soll eine Anforderungsanalyse durchgeführt und Möglichkeiten, diese Anforderungen zu erfüllen, präsentiert werden.

4.2 Anforderungsanalyse

Das Ziel dieser Arbeit ist, für die in Kapitel 4.1 vorgestellten Szenarien möglichst sichere, aber auch effiziente Lösungen zu erarbeiten. Aus den Szenarien lassen sich eine Reihe von Anforderungen ableiten, die im Folgenden betrachtet werden sollen.

4.2.1 Generisches Authentifizierungsprotokoll

Eine Forderung, die bereits aus dem ersten Szenario resultiert, ist, dass die Kommunikationspartner vielfältige Authentifizierungsmethoden unterstützen müssen. Network Access Server müssen also so konfiguriert werden, dass sie jede Methode, die ein Nutzer für die Authentifizierung verwenden möchte, unterstützen. Dies beschränkt sich nicht auf jene Methoden, die Nutzer der eigenen Domäne verwenden dürfen. Es müssen auch Nutzer bedient werden können, die aus einer fremden Domäne stammen und sich über bisher unbekannte Verfahren authentifizieren wollen. Es ist also eine Technologie notwendig, die es den Network Access Servern erlaubt, unabhängig von der eingesetzten Authentifizierungsmethode und dem Datenformat die Authentifizierungsdaten korrekt entgegenzunehmen und an den Authentifizierungsserver im Back-End weiterzuleiten. Ferner ist es notwendig, dass die eingesetzte Technologie flexibel in Bezug auf die Anzahl und Reihenfolge der ausgetauschten Nachrichten ist.

Diese Anforderungen erfüllt das in Kapitel 2.2 eingeführte *EAP-Protokoll*, das jede beliebige Authentifizierungsmethode in einer einheitlichen Form transportiert. Dadurch können auch solche Nutzer Zugang zum Netzwerk erlangen, die aus einer anderen Domäne stammen und eine Authentifizierungsmethode verwenden, die von der aktuellen Domäne gar nicht unterstützt wird. Der Network Access Server kann die in EAP Requests und Responses verpackten Authentifizierungsdaten so lange zwischen Nutzer und Back-End weiterleiten, bis er eine EAP-Success Meldung erhält und den Nutzer freischalten kann. Aufgrund seiner hohen Flexibilität soll das EAP-Protokoll im Rahmen dieser Arbeit für die Einkapselung der Authentifizierungsdaten und deren Transport zwischen Nutzer und Authentifizierer eingesetzt werden.

4.2.2 Unterstützung unterschiedlicher Zugangstechnologien

Die betrachteten Szenarien spielen sich allesamt in Netzen der vierten Generation ab. Diese werden durch die Vielfältigkeit der verwendeten Zugangstechnologien charakterisiert. Ein Nutzer kann sich somit über UMTS, WLAN oder GPRS Zugang zum Netzwerk verschaffen. Seine Authentifizierungsinformationen könnte er ebenfalls über UMTS, WLAN oder GPRS an den Network Access Server übermitteln, was diesen zwingen würde, für jede Übertragungstechnologie eine separate EAP-Methode zu implementieren. Einen derart großen Aufwand gilt es zu vermeiden, weswegen ein Protokoll verwendet werden soll, das eine einheitliche Authentifizierung für alle Übertragungstechnologien ermöglicht.

Das einzige Protokoll, das diese Anforderung erfüllt, ist das sich noch im Entwicklungsstadium befindende *PANA-Protokoll*, das im 2. Kapitel eingeführt wurde. PANA ist ein Netzwerkprotokoll, das für den Transport von EAP-Paketen auf der IP-Ebene zuständig ist. Dadurch ist der Authentifizierungsvorgang unabhängig von der verwendeten Zugangstechnologie, was eine einheitliche Authentifizierung ermöglicht. In dieser Arbeit soll PANA auf dem Abschnitt zwischen Nutzer und Network Access Server eingesetzt werden.

4.2.3 Skalierbarkeit der Authentifizierung

Bereits im ersten Szenario ist erkennbar, dass die EAP-Pakete nicht nur vom Nutzer zum Network Access Server transportiert werden, sondern zum Authentifizierungsserver im Back-End weitergeleitet werden müssen. Nur in sehr kleinen Domänen kann ein Network Access Server gleichzeitig als Authentifizierungsserver fungieren. Damit eine Domäne skalierbar ist, müssen die beiden Entitäten entkoppelt werden. Dadurch können der Domäne weitere Network Access Server hinzugefügt werden, ohne dass das einen Einfluss auf den Authentifizierungsvorgang haben würde. Für die Übertragung der Daten zwischen den Network Access Servern und dem Authentifizierungsserver muss ein so genanntes AAA-Protokoll eingesetzt werden, das Authentifizierungs-, Autorisierung-, und Abrechnungsdaten transportieren kann.

In der vorliegenden Arbeit wird für diesen Zweck das in Kapitel 2.3 vorgestellte *Diameter-Protokoll* verwendet. Diameter ist ein Protokoll der Netzwerkschicht, das AAA-Daten und damit auch EAP-Pakete unabhängig von der zugrunde liegenden Infrastruktur transportieren kann. Es ermöglicht nicht nur den Transport von EAP-Paketen innerhalb einer Domäne, sondern erlaubt auch die Weiterleitung dieser Pakete an

Authentifizierungsserver anderer Domänen. Dadurch ist die Skalierbarkeit des Systems sichergestellt, da bei Bedarf weitere Domänen hinzugefügt werden können.

4.2.4 Effiziente Reauthentifizierung

Im zweiten Szenario wurde auf die Notwendigkeit der periodischen Reauthentifizierung hingewiesen. Das Problem, das bei einem Endgerät ohne Sim-Karte anfällt, besteht darin, dass die Reauthentifizierung auf eine gewöhnliche Authentifizierung zurückgeführt wird. Das würde einer Neuanmeldung des Nutzers gleichkommen, was aufgrund des hohen Aufwands nicht praktikabel ist. Stattdessen muss eine Möglichkeit gefunden werden, die Reauthentifizierung automatisch erfolgen zu lassen. Zusätzlich sollen für eine Reauthentifizierung weniger Kommunikationsrunden zwischen Endgerät und Netzwerk anfallen als bei einer Neuanmeldung, um so den gesamten Vorgang zu beschleunigen.

Dies kann mit dem Einsatz von *SAML* (vgl. Kapitel 3) erreicht werden. Die erste Authentifizierung des Nutzers erfolgt über eine herkömmliche Authentifizierungsmethode. Nach dem erfolgreichen Abschluss des Vorgangs erstellt eine an den Authentifizierungsserver angeschlossene SAML-Autorität eine Authentifizierungs-Assertion und eine darauf verweisende Referenz namens Artifact (vgl. Artifact Profil in Kapitel 3.3.1). Das Artifact wird über den Authentifizierungsserver dem Nutzer übermittelt und auf dessen Endgerät gespeichert. Wenn eine Reauthentifizierung anfällt, präsentiert das Endgerät das Artifact, ohne dass eine Interaktion seitens des Nutzers notwendig wird. Der Authentifizierungsserver leitet das Artifact an die SAML-Autorität und verlangt die zugehörige Assertion. Die SAML-Autorität übermittelt die Assertion dem Authentifizierungsserver, der nun je nach Inhalt die Reauthentifizierung des Nutzers erfolgreich oder erfolglos abschließt.

Diese Methode ist effizient, da sie den Nutzer für die Reauthentifizierung nicht mehr benötigt. Das hat Vorteile für den Nutzer, weil ihm die lästige wiederholte Anmeldung erspart wird. Darüber hinaus ist die Dauer der Reauthentifizierung und die damit verbundene Unterbrechung des Dienstes sehr viel kürzer, da nur noch Maschinen miteinander kommunizieren und die bremsende Nutzerinteraktion vermieden wird. Schließlich wird durch den Einsatz dieser Methode das Netzwerk entlastet, da für eine Reauthentifizierung nur noch eine Kommunikationsrunde zwischen Endgerät und Netzwerk anfällt.

4.2.5 Unterstützung der Authentifizierung in Partnerdomänen

Das dritte Szenario behandelt den Fall, dass ein mobiler Nutzer seine Heimatdomäne verlässt und Zugang zum Netzwerk einer anderen Domäne anfordert. Da die besuchte Domäne keinerlei Authentifizierungsdaten über den Nutzer besitzt, ist eine Kommunikation zwischen den Authentifizierungsservern der beiden Domänen notwendig. Im angesprochenen Szenario wurde angenommen, dass zwischen den beiden Domänen Verträge existieren, sie sich also kennen.

Die beiden Authentifizierungsserver haben somit die notwendigen Routing-Informationen um die Partnerdomäne zu erreichen, und sie verfügen über die notwendigen Schlüssel um einen sicheren Datentransfer zu realisieren. In diesem Fall kann *Diameter* in Verbindung mit *SAML* eingesetzt werden. Diameter kann für die Kommunikation zwischen den Domänen eingesetzt werden und ist für den Transport von Authentifizierungsinformationen zuständig. Diese liegen im SAML-Format vor und werden gemäß dem Artifact Profil zwischen dem Authentifizierungsserver der Partnerdomäne und der SAML-Autorität der Heimatdomäne ausgetauscht.

4.2.6 Unterstützung der Authentifizierung in Föderationen

Im vierten Szenario wird der Einschätzung Rechnung getragen, dass aufgrund der hohen Anzahl von Netzbetreibern bilaterale Verträge zwischen ihnen eher die Ausnahme sein werden. Stattdessen werden vertrauenswürdige Dritte, genannt Föderations-Manager, mit dem Aufbau von Föderationen beauftragt. Die Föderations-Manager speichern die Routing- und Verschlüsselungsinformationen der einzelnen Domänen und können so Verbindungen zwischen Domänen herstellen, die davor keinerlei Kontakt miteinander hatten. Dafür müssen Föderations-Manager die Fähigkeit besitzen Authentifizierungsdaten zu verarbeiten und an die richtigen Authentifizierungsserver weiterzuleiten.

Das *Diameter-Protokoll* definiert für diese Aufgabe den Proxy Agent. Proxy Agents sind nicht nur für die simple Vermittlung der Nachrichten zuständig, sondern dürfen sie auch abändern, um Regeln durchzusetzen. Diese Regeln können bei der Aufnahme einer Domäne in die Föderation definiert werden, und können festlegen, unter welchen Umständen die neue Domäne mit welchen Föderationsmitgliedern zusammenarbeiten möchte. Dadurch kann die Domäne Einfluss auf die Auswahl ihrer Kommunikationspartner nehmen.

Das *SAML-Protokoll* wird weiterhin für den Austausch der Authentifizierungsdaten zwischen den Domänen eingesetzt. Der Austausch erfolgt erneut über das Artifact Profil,

bei dem der Nutzer in der fremden Domäne ein Artifact präsentiert, und der Authentifizierungsserver durch den Föderierungs-Manager sich von der SAML-Autorität der Heimatdomäne die zugehörige Assertion verschafft.

4.2.7 Sicherer Datentransport

Bei einem Authentifizierungsvorgang werden vertrauliche Daten über ein unsicheres Medium versendet. In den vorgestellten Szenarien passieren Authentifizierungsdaten auf ihrem Weg vom Nutzer zum Authentifizierungsserver mehrere Knoten. Um sie vor eventuellen feindlichen Knoten zu schützen, muss die Ende-zu-Ende Sicherheit für diese Daten gewährleistet werden. Zusätzlich muss auch eine Hop-zu-Hop Sicherheit realisiert werden, die den Schutz jener Daten sicherstellt, die nur von einem Knoten zum nächsten übermittelt werden.

Die Hop-zu-Hop Sicherheit wird durch den Einsatz von *Diameter* erreicht. Für die Absicherung des Datentransports zwischen den Diameter-Knoten wird das IPsec–Protokoll eingesetzt, das die Verschlüsselung und digitale Signierung des Datenverkehrs erlaubt. Dadurch kann die Vertraulichkeit und Integrität der Daten sichergestellt werden. Zusätzlich wird die Authentizität der Kommunikationspartner und die Nichtabstreitbarkeit der versendeten Nachrichten garantiert.

Das an das Diameter Basis-Protokoll gekoppelte CMS-Modul (vgl. Kapitel 2.3.1) unterstützt auch die Errichtung von Sicherheitsbeziehungen zwischen entfernten Knoten, die über keine direkte Kommunikationsmöglichkeit verfügen. Dadurch kann innerhalb von Diameter-Netzen eine Ende-zu-Ende Sicherheit realisiert werden.

Um eine vollständige Ende-zu-Ende Sicherheit zu ermöglichen, müssen die *EAP*-Pakete, die für den Transport der Authentifizierungsdaten zwischen Nutzer und Authentifizierer zuständig sind, verschlüsselt und digital signiert werden. Die Verschlüsselungsdaten können über verschiedene Wege ausgetauscht werden. Denkbar wäre die Nutzung einer PKI für die Verteilung öffentlicher und privater Schlüssel oder die Festlegung von Schlüsseln bei der ersten Authentifizierung. Im Folgenden wird angenommen, dass ein asynchroner Verschlüsselungsalgorithmus verwendet wird und die Kommunikationspartner den jeweiligen öffentlichen Schlüssel der Gegenseite kennen. Der Nutzer muss dann sein Artifact erst mit seinem privaten Schlüssel signieren, damit die Bedingung der Nichtabstreitbarkeit erfüllt ist. Danach wird das signierte Artifact mit dem öffentlichen Schlüssel des Authentifizierungsservers verschlüsselt, um sie vor Abhörattacken zu schützen. Schließlich werden die Daten erneut mit dem privaten Schlüssel des Nutzers signiert, damit der Empfänger eine eventuelle Manipulation der

Daten erkennen kann. Die signierte und verschlüsselte Nachricht bildet zusammen mit der Adresse des Nutzers den so genannten IdToken, der in Kapitel 5 näher erklärt wird. Da die Adresse unverschlüsselt ist, kann der IdToken über PANA und Diameter an den Authentifizierungsserver der Heimatdomäne übermittelt werden. Auch wenn die Authentifizierungsdaten über einen feindlichen Zwischenknoten geleitet werden sollten, wäre es diesem nicht möglich die Nachricht zu entschlüsseln, da er den privaten Schlüssels des Empfängers nicht besitzt. Ebenso kann er keine Manipulationen an der Nachricht durchführen, da er, um diese zu vertuschen, den privaten Schlüssel des Absenders bräuchte. Dadurch ist die Vertraulichkeit, Integrität und Nichtabstreitbarkeit der Nachrichten, die zwischen Nutzer und Authentifizierungsserver ausgetauscht werden, sichergestellt und die Authentizität der Kommunikationspartner gewährleistet.

In den vorigen Abschnitten wurden aus den zu realisierenden Szenarien Anforderungen abgeleitet und Möglichkeiten aufgezeigt, diese mittels Technologien zu lösen, die im zweiten und dritten Kapitel der Diplomarbeit eingeführt wurden. Die angesprochenen Technologien werden bei der Entwicklung der Systemarchitektur, die im nächsten Abschnitt erfolgt, eine zentrale Rolle spielen.

4.3 Systemarchitektur

In diesem Teil der Diplomarbeit wird die Architektur des zu realisierenden Systems vorgestellt. Es werden die Komponenten eingeführt, die bei einer Authentifizierung über EAP-SAML beteiligt sind, ihr Aufbau erklärt und ihre Aufgaben erläutert.

Client

Der Client ist ein Endpunkt des Systems. Nutzer brauchen den Client, um Zugriff auf das Netzwerk und seine Dienste zu erlangen. Dafür ist eine Authentifizierung des Nutzers über das EAP-Protokoll notwendig, weswegen der Client einen EAP-Peer beinhalten muss. Das Endgerät, auf dem der Client läuft, ist mobil und kann unterschiedliche Übertragungstechnologien unterstützen. Die EAP-Pakete sollen deswegen über das PANA-Protokoll transportiert werden, was voraussetzt, dass sich ein PANA Client (PaC) auf dem Endgerät befindet.

Network Access Server

Die Aufgabe des Network Access Server ist Nutzern nach der Authentifizierung den Netzwerkzugang zur Verfügung stellen. Weil unterschiedliche Übertragungstechnologien unterstützt werden müssen und die Authentifizierungsdaten über das PANA-Protokoll

transportiert werden, beinhaltet jeder Network Access Server einen PANA Authentication Agent (PAA). Die EAP-Pakete müssen über Diameter zum Authentifizierungsserver im Back-End weitergeleitet werden, weswegen eine weitere Komponente notwendig ist: der Diameter Client.

Authentifizierungsserver

Der Diameter Client im Network Access Server kommuniziert mit einem Diameter Server, der Teil des Authentifizierungsservers ist. Dabei werden EAP-Pakete ausgetauscht, die für die Authentifizierung der Nutzer notwendig sind. Die Authentifizierung übernimmt der ebenfalls im Authentifizierungsserver enthaltene EAP-Authentication-Server. Dieser muss unterschiedliche EAP-Methoden unterstützen und die Möglichkeit bieten Methoden auszuhandeln, falls ein Nutzer die voreingestellte Methode nicht unterstützt.

SAML-Autorität

Die SAML-Autorität kommt bei der Verwendung der EAP-SAML Methode zum Einsatz. Sie wird vom Authentifizierungsserver bei der Neuanmeldung eines Nutzers mit der Erstellung einer Assertion und des zugehörigen Artifacts beauftragt. Das Artifact wird an den Authentifizierungsserver übermittelt, der es seinerseits an den Nutzer weiterleitet. Präsentiert der Nutzer zu einem späteren Zeitpunkt das Artifact dem Authentifizierungsserver, fordert dieser die zugehörige Assertion von der SAML-Autorität an.

Föderierungs-Manager

Die letzte Komponente ist der Föderierungs-Manager. Sie ist für den Aufbau und die Verwaltung von Föderationen zuständig. Der Föderierungs-Manager muss Diameter-Nachrichten zwischen Authentifizierungsservern unterschiedlicher Domänen vermitteln und setzt dafür einen Diameter Proxy Agent ein. Der Diameter Proxy Agent ist so definiert, dass er nicht nur für die Weiterleitung von Diameter-Nachrichten zuständig ist, sondern diese Nachrichten auch ändern oder filtern kann, um Richtlinien umzusetzen. Dadurch können Mitglieder auf die Auswahl ihrer Kommunikationspartner Einfluss nehmen.

Die Abbildung 19 veranschaulicht die Architektur des Systems und den Aufbau der Komponenten. Ebenso ist ersichtlich, welche Protokolle zwischen den einzelnen Komponenten zum Einsatz kommen. Der Übersichtlichkeit halber wurde der Heimat-Authentifizierungsserver, der im Falle einer Authentifizierung eines Nutzers in einer fremden Domäne kontaktiert werden muss, nur angedeutet.

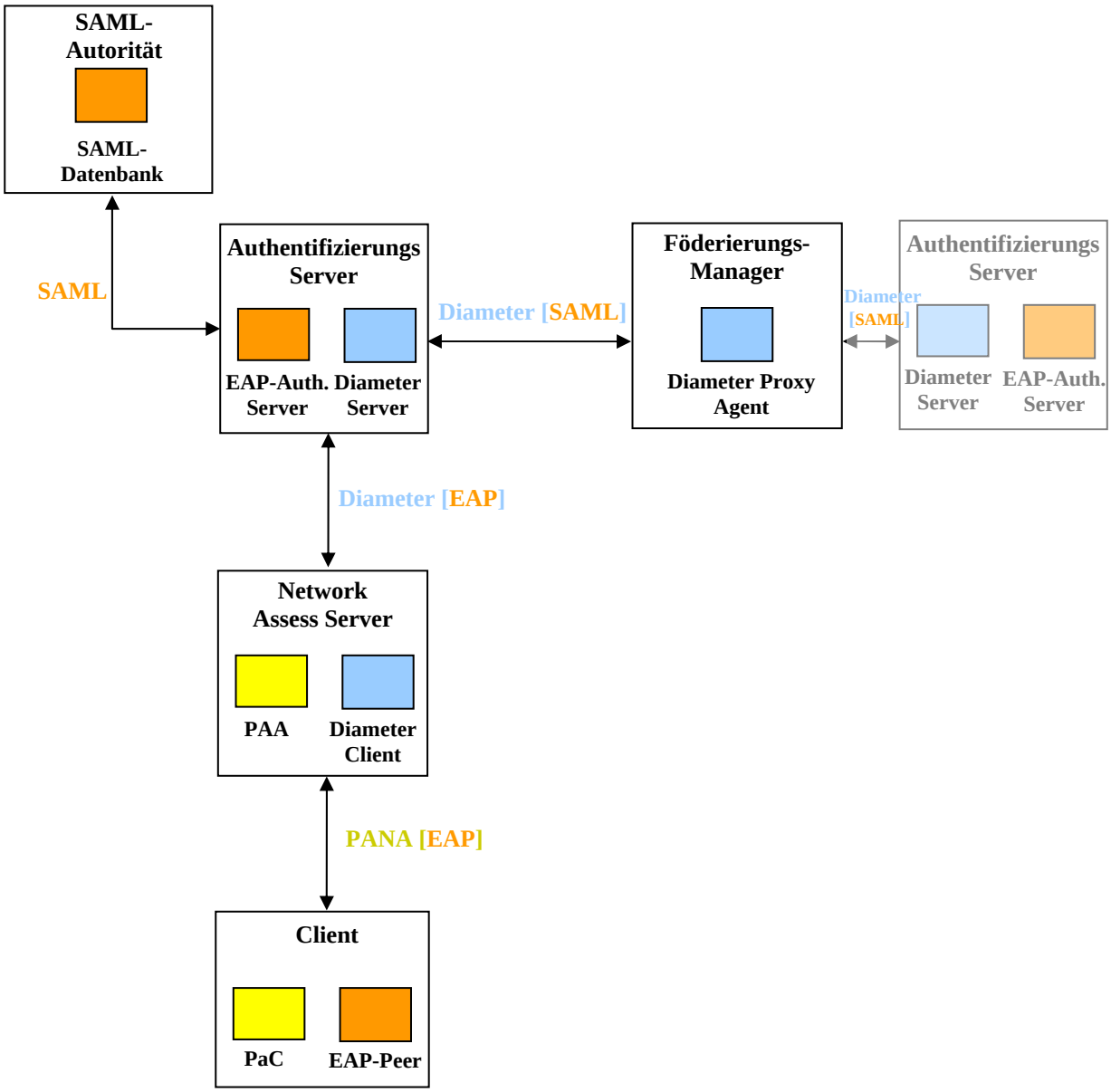


Abbildung 19: Systemarchitektur

4.4 Kommunikationsabläufe

Bei der Durchführung einer Authentifizierung kollaborieren je nach Situation alle oder nur ein Teil der im vorigen Abschnitt eingeführten Systemkomponenten. Die Abläufe dieser Kollaboration sollen in den nächsten Abschnitten anhand von Sequenzdiagrammen detailliert vorgestellt werden.

4.4.1 Neuanmeldung eines Nutzers

Wenn sich ein Nutzer bei einer Neuanmeldung zum ersten Mal in seiner Heimatdomäne authentifiziert, existiert noch kein Artifact und keine Assertion, die die Authentifizierung erleichtern könnten. Stattdessen muss sich der Nutzer mittels einer herkömmlichen Methode authentifizieren. Im folgenden Beispiel verwendet er die in Kapitel 2.2.2 eingeführten Methoden EAP-Identity und EAP-TLS.

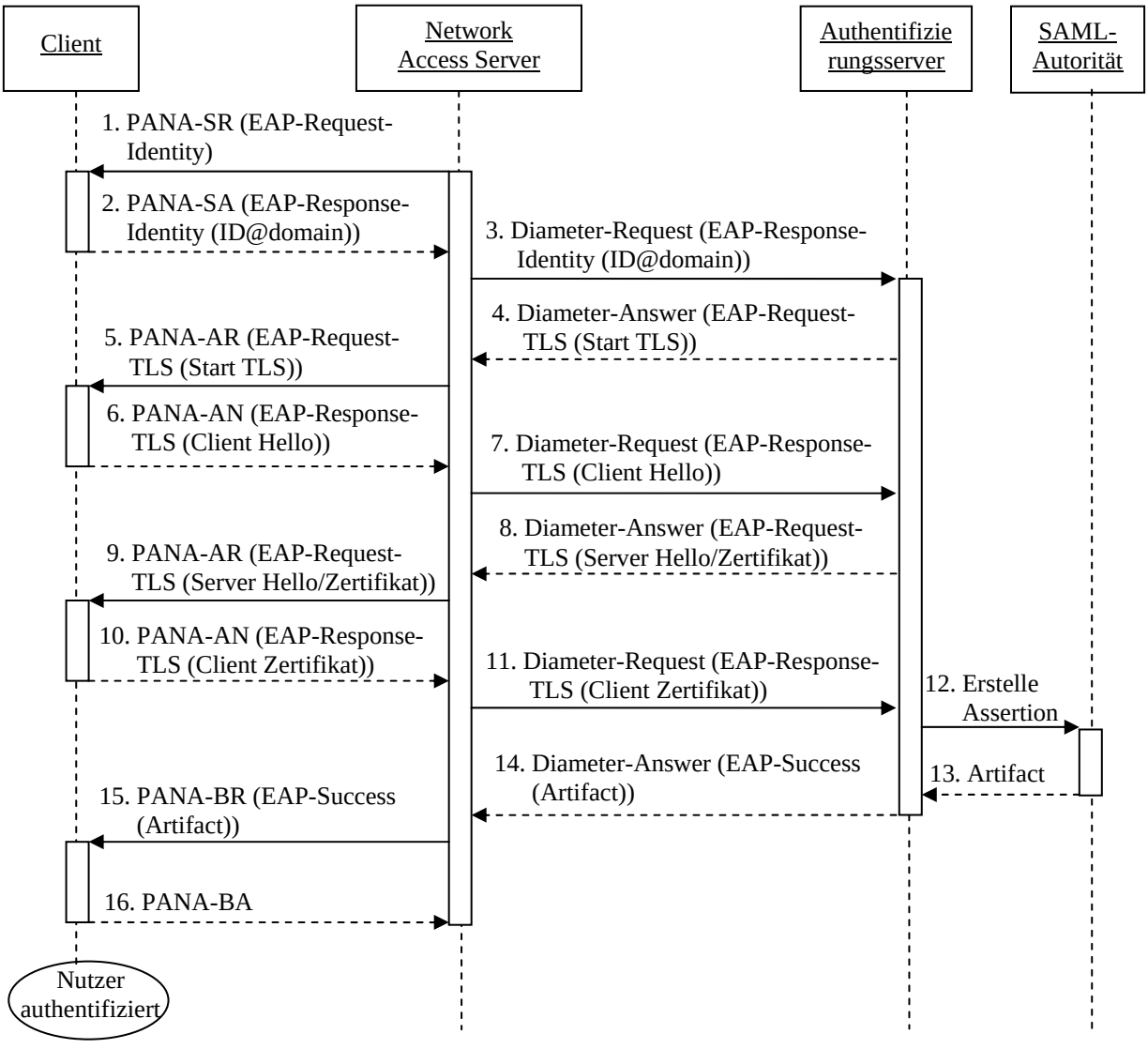


Abbildung 20: Sequenzdiagramm - Neuanmeldung eines Nutzers

1. Die Kommunikation zwischen Netzbetreiber und Nutzer wird durch einen PANA-Start-Request gestartet, der vom PAA des Network Access Server an den PaC des Clients gesendet wird. Darin enthalten ist eine Aufforderung, sich über EAP-Identity zu identifizieren.

2. Der Client verpackt die vom EAP-Peer ausgestellten Identitätsinformationen in eine PANA-Start-Answer und übermittelt die Antwort an den PAA.
3. Die EAP-Identity Pakete werden im Network Access Server mittels des PAA und des Diameter Clients von PANA auf Diameter portiert und innerhalb eines Diameter-Requests an den Diameter Server im Authentifizierungsserver geleitet.
4. Der EAP-Authentication Server im Authentifizierungsserver analysiert die Identitätsinformationen, leitet daraus ab, dass mit dem Nutzer EAP-TLS als bevorzugte Authentifizierungsmethode vereinbart wurde, und sendet eine EAP-TLS Initialisierungsnachricht, die in einem Diameter-Answer verpackt ist.
5. Diese Nachricht wird im Network Access Server von Diameter auf PANA portiert und innerhalb eines PANA-Authentication-Request an den Client weitergeleitet.
6. Der EAP-Peer im Client erkennt, dass der Back-End-Server eine Authentifizierung über EAP-TLS wünscht und sendet sein Einverständnis mit der Methode. Diese wird in ein PANA-Authentication-Answer eingepackt.
7. Das EAP Paket wird mittels PAA und Diameter Client an das Back-End weitergeleitet.
8. Der EAP-Authentication Server verpackt das Server-Zertifikat in ein EAP-TLS Paket und sendet es mittels des Diameter Servers an den Network Access Server
9. Der Network Access Server ist erneut für die Portierung und Weiterleitung des EAP Pakets zuständig.
10. Der EAP-Peer im Client überprüft das Server-Zertifikat und versendet dann sein eigenes Zertifikat in einer EAP Nachricht.
11. Das Client-Zertifikat wir über PANA und Diameter an den Authentifizierungsserver geleitet.
12. Der EAP-Authentication Server überprüft das Zertifikat und beauftragt die SAML-Autorität mit der Erstellung eines (Assertion, Artifact) Paares.
13. Die SAML-Autorität generiert eine Authentifizierungs-Assertion und ein Artifact für den Nutzer und sendet Letzteres an den Authentifizierungsserver.
14. Der Authentifizierungsserver sendet das Artifact zusammen mit einer Erfolgsmeldung über Diameter an den Network Access Server.
15. Der Network Access Server übermittelt das Artifact innerhalb eines PANA-Bind-Requests an den Client und signalisiert damit, dass er für den Netzzugang freigeschaltet wird.
16. Der Client speichert das Artifact für die spätere Verwendung und akzeptiert die vom Network Access Server angebotene Verbindung mit einem PANA-Bind-Answer.

4.4.2 Reauthentifizierung in der Heimatdomäne

In einem mobilen Netzwerk ist es aus Sicherheitsgründen notwendig, dass sich Nutzer periodisch oder beim Übertritt in einen anderen administrativen Bereich des Netzes neu authentifizieren. Da der Nutzer bereits angemeldet ist, kann davon ausgegangen werden, dass auf seinem Endgerät ein Artifact gespeichert ist. Durch dieses Artifact, das in einem verschlüsselten und signierten IdToken eingekapselt ist, kann dem Nutzer die aufwendige Neuanmeldung erspart werden, was auch einen positiven Einfluss auf die Geschwindigkeit des Authentifizierungsvorgangs hat. Das Sequenzdiagramm in der Abbildung 21 zeigt den Ablauf der mit Hilfe des Artifacts optimierten Reauthentifizierung. Es ist offensichtlich, dass die Kommunikation sehr viel kürzer ist als bei einer kompletten Neuanmeldung, die ohne den Einsatz von SAML anfallen würde.

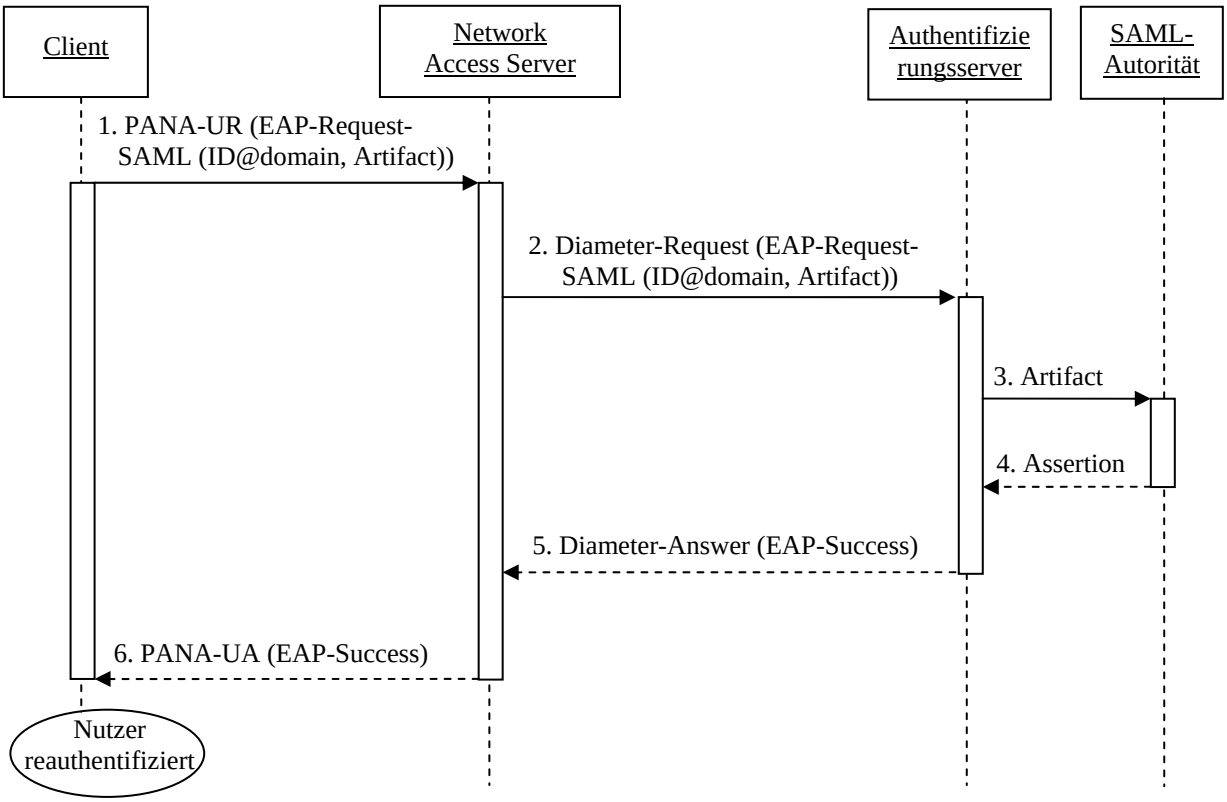


Abbildung 21: Sequenzdiagramm - Reauthentifizierung in der Heimatdomäne

1. Wenn ein Client feststellt, dass er innerhalb eines Netzes die Funkzellen gewechselt hat, muss er das dem Betreiber melden, um weiter erreichbar zu sein. Dafür sendet er ein PANA-Update-Request, in dem er ein EAP-SAML Paket einfügt, das seine Identitätsinformationen und sein SAML Artifact enthält.
2. Der Network Access Server portiert mittels PAA und Diameter Client das EAP-SAML Paket auf Diameter und sendet es an den Authentifizierungsserver.

3. Der EAP-Authentication Server im Back-End sendet das Artifact an die SAML-Autorität und verlangt die zugehörige Assertion.
4. Die SAML-Autorität überprüft, ob eine derartige Assertion existiert und übermittelt diese gegebenenfalls an den Authentifizierungsserver.
5. Der Authentifizierungsserver überprüft die Assertion auf ihre Gültigkeit und leitet dann eine Erfolgsmeldung an den Network Access Server.
6. Der Network Access Server leitet daraufhin die Erfolgsmeldung über ein PANA-Update-Answer an den Client weiter.

4.4.3 Authentifizierung in einer föderierten Domäne

Die Authentifizierung eines Nutzers in einer Partnerdomäne ist ein Spezialfall der Authentifizierung in einer föderierten Domäne, da die Authentifizierungsserver direkt und ohne die Vermittlung des Föderierungs-Manager miteinander kommunizieren können. Im Folgenden wird deshalb nur die Interaktion der beteiligten Komponenten bei einer Authentifizierung in einer föderierten Domäne dargestellt.

Wenn ein Nutzer sich in einer Domäne authentifizieren will, die zur gleichen Föderation wie die Heimatdomäne gehört, müssen seine Authentifizierungsdaten zur Heimatdomäne geleitet werden. Für die Vermittlung der Nachrichten zwischen den Domänen ist der Föderationsmanager zuständig. Das Sequenzdiagramm in Abbildung 22 zeigt die Interaktionen zwischen den bei der Authentifizierung beteiligten Komponenten.

1. Wenn ein Nutzer eine fremde Domäne betritt, wird die Kommunikation zwischen dem Netzbetreiber und dem Endgerät des Nutzers durch einen PANA-Start-Request initiiert. Der Netzbetreiber fordert darin den Client auf dem Endgerät auf, sich zu identifizieren.
2. Der Client antwortet mit einem PANA-Start-Answer, in dem er ein EAP-SAML Paket verpackt. In dem EAP-SAML Paket sind nicht nur die Identitätsinformationen des Nutzers verpackt, sondern auch seine Authentifizierungsdaten in Form eines SAML Artifacts.
3. Der Network Access Server portiert das EAP-SAML Paket von PANA auf Diameter und sendet es an den Authentifizierungsserver der fremden Domäne.
4. Dieser stellt anhand der Identitätsinformationen fest, dass der Nutzer zu einer anderen Domäne gehört und sendet die Identitätsinformationen und das Artifact über das Diameter-Protokoll an den Föderierungs-Manager.
5. Der Föderierungs-Manager fungiert als Diameter Proxy Agent, der die Authentifizierungsserver aller Domänen innerhalb der Föderation kennt. Er leitet die Diameter-Nachricht mit den Identitätsinformationen und dem Artifact an den Authentifizierungsserver der Heimatdomäne weiter.

6. Der Authentifizierungsserver sendet das Artifact an die SAML-Autorität und fordert die zugehörige Assertion an.
7. Die SAML-Autorität überprüft, ob das Artifact korrekt ist und ob eine Assertion dazu existiert. Sie sendet dann die Assertion an den Authentifizierungsserver.

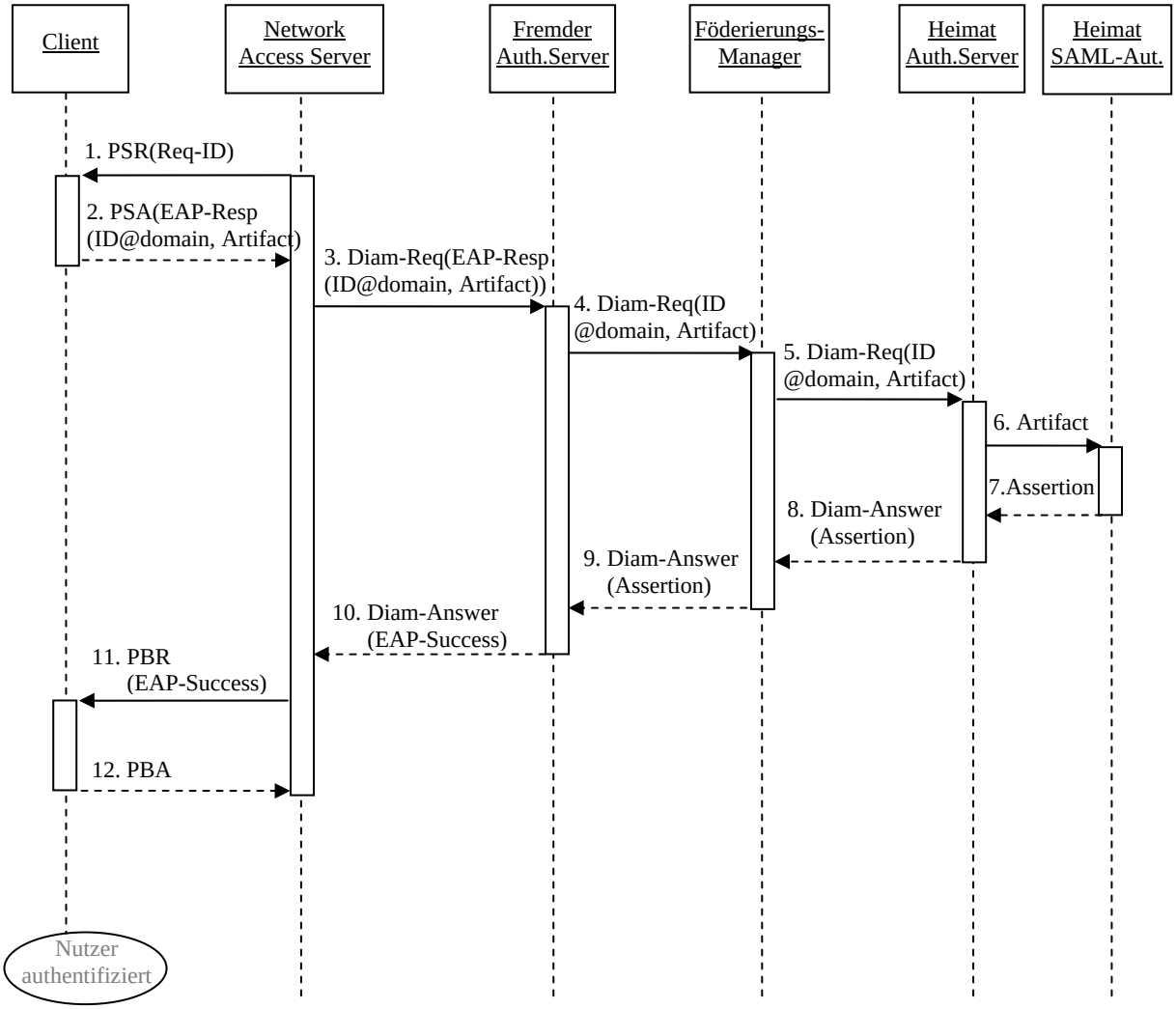


Abbildung 22: Authentifizierung in einer föderierten Domäne

8. Dieser überprüft die Assertion auf ihre Gültigkeit und sendet sie über das Diameter-Protokoll an den Föderierungsmanager.
9. Der Föderierungsmanager leitet daraufhin die Diameter-Nachricht an den fremden Authentifizierungsserver weiter.
10. Der Authentifizierungsserver kann nun basierend auf der Assertion eine Entscheidung treffen, ob der Client erfolgreich authentifiziert werden soll. Falls die Authentifizierung erfolgreich ist, wird eine Erfolgsmeldung an den Network Access Server gesendet.

11. Dieser leitet die Erfolgsmeldung über ein PANA-Bind-Request weiter, wodurch er dem Client signalisiert, dass er erfolgreich authentifiziert wurde und Zugang zum Netz erhalten kann.
12. Der Client akzeptiert die vom Network Access Server angebotene Verbindung, indem er mit einem PANA-Bind-Answer antwortet.

Es wird erneut deutlich, dass die Verwendung von EAP-SAML zur Authentifizierung eines Nutzers große Vorteile hat. Der Kommunikationsaufwand wird drastisch reduziert. Eine herkömmliche Authentifizierungsmethode wie EAP-TLS benötigt allein für die Authentifizierung in der Heimatdomäne mehr Nachrichten als eine SAML-basierte Authentifizierung in einer Föderation. In welchem Bereich die Zeitersparnis liegt, die durch den Einsatz von EAP-SAML erzielt wird, muss durch zukünftige Vergleichstests ermittelt werden.

4.5 Zusammenfassung

Im abgeschlossenen Kapitel wurde ein Konzept für eine Authentifizierungsmethode entwickelt, die für den Einsatz in Mobilenfunknetzen der vierten Generation geeignet ist. Es wurden Szenarien vorgestellt, in denen für die Erlangung des Netzwerkzugangs eine Authentifizierung notwendig ist. Daraus wurden Anforderungen abgeleitet, die eine Authentifizierungsmethode in solchen Umgebungen erfüllen muss. Die Anforderungen verlangen, dass die Authentifizierungsdaten über unterschiedliche Zugangstechnologien transportiert werden können, und dass der Einsatz der Methode auch in großen, heterogenen Netzwerken sichergestellt ist. Zusätzlich muss durch Single Sign-On eine effiziente Authentifizierung sowohl in der Heimatdomäne des Nutzers, als auch in fremden Domänen realisiert werden. Schließlich muss auch der sichere Transport der Authentifizierungsdaten gewährleistet werden. Basierend auf diesen Vorgaben wurde ein Protokoll namens EAP-SAML entwickelt, das zusammen mit den Protokollen PANA und Diameter die gestellten Anforderungen erfüllt. Es wurden die Komponenten beschrieben, die für die Durchführung von EAP-SAML notwendig sind und es wurde mit Hilfe von Interaktionsdiagrammen veranschaulicht, wie diese Komponenten bei einem EAP-SAML Authentifizierungsvorgang interagieren. Das nächste Kapitel befasst sich mit der Realisierung der angesprochenen Systemkomponenten und Interaktionsdiagrammen.

5. Implementierung

Das fünfte Kapitel befasst sich mit dem praktischen Teil der Diplomarbeit. Dieser bestand darin, eine Authentifizierungsmethode zu implementieren, die die im vierten Kapitel entworfenen Konzepte unterstützt. Die Implementierung erfolgte in C++ auf einer Linux-Plattform. Ausschlaggebend für die Entscheidung C++ zu verwenden war die Objektorientierung der Sprache und die höhere Geschwindigkeit im Vergleich zu anderen objektorientierten Programmiersprachen wie Java. Im Gegensatz zu Java ist C++ jedoch plattformabhängig, was die Einsatzmöglichkeiten der Software stark einschränkt. Um dennoch die Portierung der Software auf andere Betriebssysteme zu ermöglichen, wurden Bibliotheken der *Adaptive Communication Environment*, kurz *ACE* [41], benutzt. ACE ist ein objektorientiertes Framework, das eine Menge von C++ Klassen und Funktionen bereitstellt, die die Entwicklung nebenläufiger, ereignisgesteuerter, plattformübergreifender Kommunikationssoftware ermöglicht. Neben ACE wurden im Rahmen der Implementierung zwei weitere Ressourcen verwendet: das *Open Diameter Framework* [43] von SourceForge.net und die *SAML-Autorität* der BMW Group. Beide werden im ersten Teil des Kapitels ausführlich behandelt. Die selbst entwickelten Komponenten, wie das EAP-SAML Protokoll mit dem EAP-SAML Peer und dem EAP-SAML Authentication Server sowie der auf dem mobilen Endgerät laufende Client und der sich im Back-End befindliche Authentifizierungsserver, werden im zweiten Teil des Kapitels vorgestellt.

5.1 Vorhandene Komponenten

Im Rahmen dieser Arbeit musste in erheblichem Maße auf bereits vorhandene Software zurückgegriffen werden. So wurde, um die im zweiten Kapitel vorgestellten Authentifizierungsprotokolle PANA und Diameter einsetzen zu können, das Open Diameter Framework verwendet. Im nächsten Abschnitt werden die für diese Arbeit relevanten Teile des Frameworks vorgestellt.

5.1.1 Open Diameter

Das *Open Diameter Projekt* [43] wurde im Januar 2002 von David Frascone (SUN) und Yoshihiro Ohba (Toshiba) auf SourceForge.net [42], einem Managementsystem für

gemeinschaftliche Softwareentwicklungen, gegründet. Das Ziel des Projekts ist die Implementierung des Diameter Basis-Protokolls und weiterer Zusatzprotokolle. Bis zum Mai 2005 wurden folgende Protokolle implementiert:

- *libdiameter*: ist eine Bibliothek, die den Kern des Diameter Basis-Protokolls implementiert. Hier werden Diameter Clients, Server, Proxy und Redirect Agents mit ihren Zustandsmaschinen definiert.
- *libdiamparser*: diese Bibliothek übernimmt das Message Parsing der Diameter-Nachrichten und implementiert den entsprechenden Teil des Basis-Protokolls.
- *libdiameternasreq:* realisiert das im Kapitel 2.3.2 vorgestellte NASREQ-Zusatzprotokoll, das unter anderem die Rückwärtskompatibilität zum weit verbreiteten RADIUS-Prokoll gewährleistet.
- *libdiametereap:* definiert eine Programmierschnittstelle zum EAP-Protokoll, indem es das Diameter-EAP-Zusatzprotokoll realisiert.
- *libeap:* enthält eine Sammlung von EAP-Methoden, wie EAP-TLS und EAP-MD5.
- *libdiametermip4:* implementiert das Mobile-IPv4-Zusatzprotokoll als Front-End für Diameter in mobilen Netzen, das jedoch mit IPv6, dem Standard in mobilen Netzen der vierten Generation, nicht einsetzbar ist.
- *libpana:* eine Bibliothek, die das PANA-Protokoll als Front-End für Diameter-Infrastrukturen implementiert. PANA ist sowohl in IPv4 als auch in IPv6 Netzen verwendbar.

Alle angeführten Bibliotheken unterliegen der GNU General Public License [44] und sind damit frei verfügbar. Im Folgenden werden Komponenten der Bibliotheken libdiameter, libdiamparser und libpana vorgestellt.

Diameter

Die Diameter Programmierschnittstelle ist eine sitzungsbasierte API, bei der Diameter-Sitzungen mit Hilfe von C++ Klassen realisiert werden. Diese Klassen stellen die Funktionalitäten zur Verfügung, die notwendig sind um Nachrichten zu versenden und zu empfangen und Ereignisse zu verarbeiten. Es gibt zwei Typen von Sitzungs-Klassen (Session Classes): Clients und Server. Der größte Unterschied zwischen diesen Klassen liegt in der Art, wie sie instanziiert werden. Anwendungen, die auf Client-Sitzungen basieren, sind selbst für die Erstellung und Verwaltung der Instanzen dieser Sitzungen zuständig. Dagegen übernimmt bei Anwendungen, die auf Server-Klassen basieren, das Framework die Erstellung und Entfernung der Instanzen von Server-Sitzungen. Um auf eine Instanz einer Server-Sitzung zugreifen zu können, müssen Anwendungen sich bei so genannten Session Factories registrieren. Trifft eine neue Diameter-Anfrage ein, generiert die Session Factory für diese Anfrage eine neue Server-Sitzung. Hat die Anfrage dagegen

eine Session ID, die der Factory bekannt ist, wird die Anfrage an die entsprechende Server-Sitzung weitergeleitet.

Sowohl Client- als auch Server-Klassen übernehmen ausschließlich das Sitzungsmanagement. Für den Aufbau der Verbindung und das Routing der Nachrichten sind die Diameter Anwendungs-Klassen (Application Classes) zuständig. Client-Objekte werden an diese Anwendungen über ihren Konstruktor gebunden, während bei Server-Instanzen dies über die bereits eingeführten Session Factories geschieht. Jede Instanz einer Application Class repräsentiert eine eigenständige Diameter Entität. Da Programme mehrere Instanzen generieren können, bedeutet das, dass sie auch über mehrere Diameter Entitäten gleichzeitig verfügen können. Dies ist notwendig, weil ein Diameter Knoten oft nicht nur eine Authentifizierungsanwendung verwendet, sondern zusätzlich über weitere Anwendungen verfügen muss, die die Autorisierung und Abrechnung übernehmen.

Die Kommunikation zwischen Diameter Entitäten geschieht über Diameter-Requests und Diameter-Answers. Für die Komposition und Dekomposition der Nachrichten wird ein Diameter Message Parser verwendet. Die benötigten Informationen wie Typen von Attribut-Wert-Paare (AVPs) oder Befehl-Codes lädt der Message Parser während der Initialisierungsphase aus so genannten Dictionary-Dateien. Das sind XML-Dokumente, die mit Hilfe von Xerces [45] verarbeitet werden. Die aus den Nachrichten ausgelesenen Informationen werden vom Parser in Container abgelegt, die in Listen zusammengefasst werden. Wird ein Diameter-Request erstellt, so werden die einzelnen AVPs in *AAAAvpContainer* gespeichert. Diese enthalten neben dem AVP-Namen ein oder mehrere Objekte vom Typ *AAAAvpContainerEntry*, in denen die Werte des AVP gespeichert werden. Dabei kann in einem *AAAAvpContainerEntry* durchaus ein Zeiger auf eine neue Container-Liste hinterlegt sein, was die Verschachtelung von AVPs ermöglicht. Für die Verwaltung der AVP-Werte sind Instanzen der Klasse *AAAAvpContainerEntryManager* zuständig, die AVPs als Ganzes werden von *AAAAvpContainerManager*-Objekten verwaltet.

Die Abbildung 23 zeigt die Struktur einer Diameter-Nachricht. Ein Objekt des Typs *AAAAvpContainerList* stellt eine einfach verkettete Liste von *AAAAvpContainer*-Instanzen her. Die erste Container-Instanz enthält ein Attribut-Wert-Paar des Typs „Origin Host“ mit dem Wert „aaa1.domain1“. Beim zweiten Container handelt es sich bereits um eine Gruppe von AVPs. Der Name der Gruppe ist „Proxy-Info“ und sie enthält zwei AVPs: einmal das Attribut „Proxy-Host“ mit dem Wert „proxy.mydomain“ und zusätzlich das Attribut „Proxy-State“ mit dem Wert „abcde“.

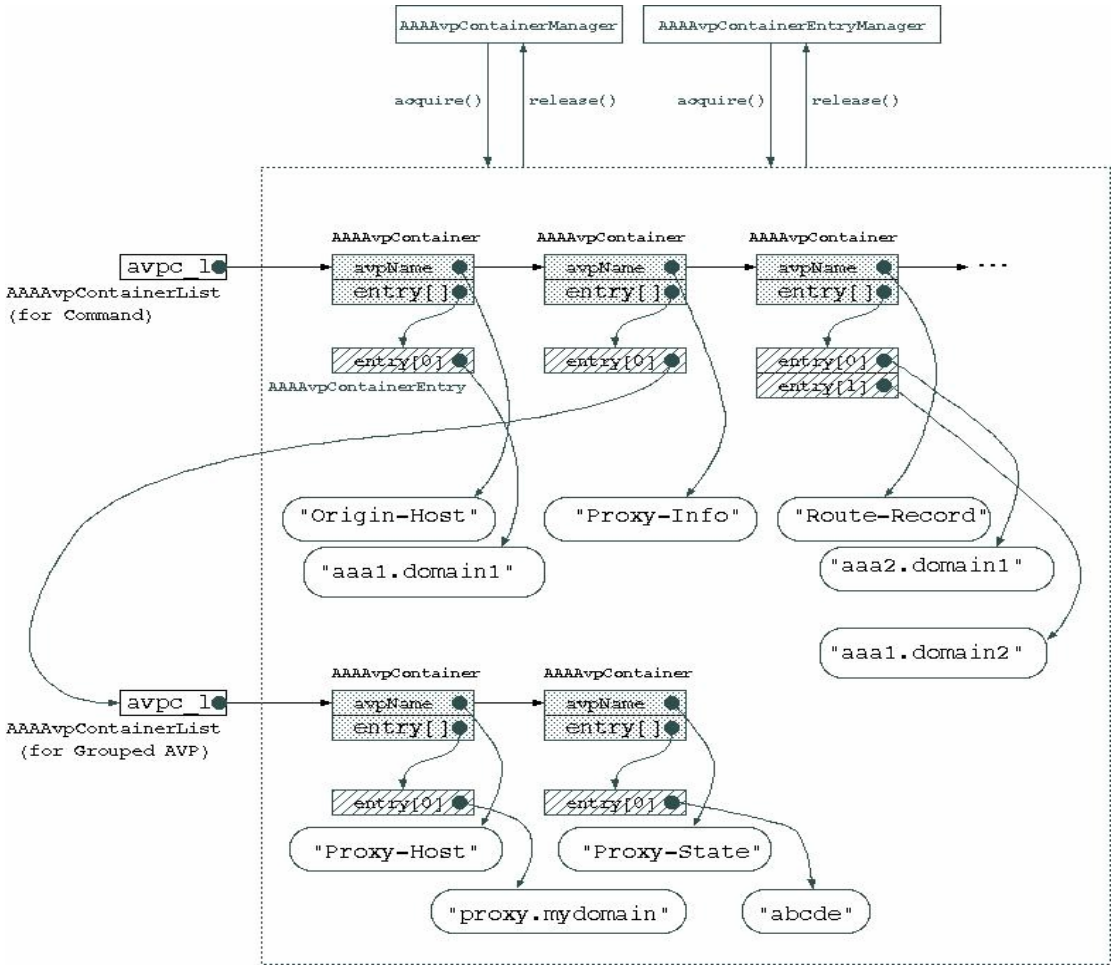


Abbildung 23: Struktur einer Diameter-Nachricht Agent [43]

Schließlich zeigt der dritte Container, dass es auch möglich ist, einem AVP mehrere Werte zuzuweisen. Insgesamt lautet die Information, die in der Nachricht gespeichert ist: Diese Nachricht stammt vom Host „aaa1.domain1“ und wurde über den sich im Zustand „abcde“ befindlichen Proxy „proxy.mydomain“ über die Knoten „aaa2.domain1“ und „aaa1.domain2“ geleitet.

Der Message Parser ist eine eigenständige Bibliothek und wird von allen Protokollen, die in Open Diameter implementiert wurden, benutzt. Das PANA-Protokoll, das im nächsten Abschnitt vorgestellt wird, kann dank des ähnlichen Aufbaus seiner Nachrichten ebenfalls den Diameter Message Parser verwenden.

PANA

Im Rahmen des Open Diameter Projektes wurde PANA als Front-End für die Diameter-Infrastruktur implementiert. PANA hat die Aufgabe, EAP Pakete vom mobilen Endgerät zum Network Access Server und damit zum Diameter Client zu leiten.

Die API ermöglicht die Erstellung von zwei Instanzen: dem PANA Client und dem PANA Authentication Agent. Diese müssen die vordefinierten Schnittstellen *PANA_ClientEventInterface* und *PANA_PaaEventInterface* implementieren. Während PANA Clients von den Anwendungen selbst generiert werden können, müssen PANA Authentication Agents von einer Factory erstellt werden und dem Client als Template-Parameter eines *PANA_PacSessionFactoryAdapter*<…> übergeben werden.

Die Architektur des PANA Authentication Agent wird in der Abbildung 24 veranschaulicht. Ein PAA besteht aus drei Modulen: dem Core Module, dem Transport Module und dem Session Module.

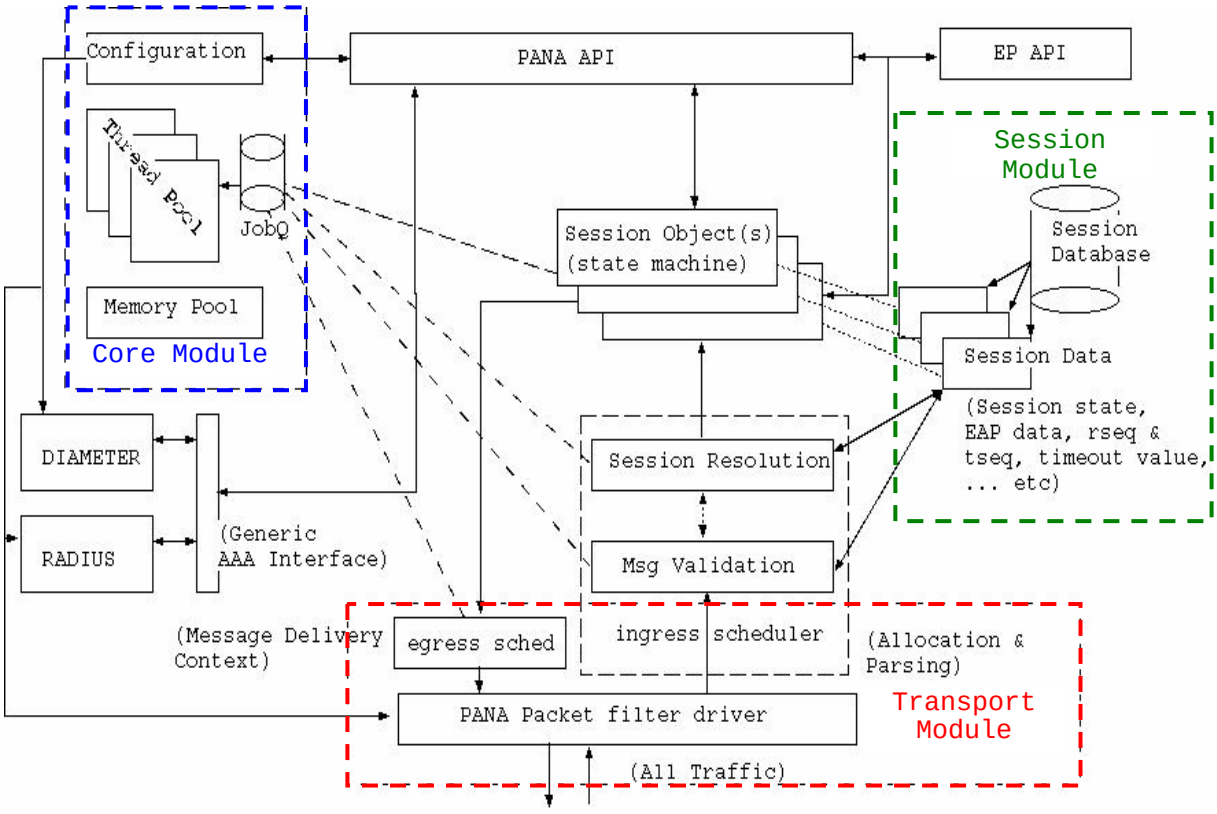


Abbildung 24: Architektur des PANA Authentication Agent [43]

Das Core Module beinhaltet Objekte für die Prozessverarbeitung, de nicht unmittelbar mit den Protokoll-Eigenschaften zusammenhängen. Die erste Aufgabe des Moduls ist die Initialisierung des PANA Authentication Agents. Es ist zuständig für die Verarbeitung und die Verwaltung von Konfigurationsdaten, die in XML-Dokumenten gespeichert werden. Das Core Modul ist so konzipiert, dass es die Abarbeitung einer hohen Anzahl paralleler Anfragen ermöglicht. Dafür sind alle Dienste und Funktionen des PANA Authentication Agents in Jobs unterteilt. Die Jobs werden von den im Thread Pool wartenden Anfragen nebenläufig ausgeführt, wobei es für Jobs unterschiedliche Prioritäten geben kann. Um

einen effizienten Kontextwechsel bei Systemaufrufen oder Prozesswechseln zu ermöglichen, wird ein Memory Pool verwendet.

Das Transport Module besteht aus einem Packet Filter, einem Ingress Scheduler und einem Egress Scheduler. EAP-Pakete, die den PAA erreichen, werden vom Packet Filter nach einer Validitätsprüfung an den Ingress Scheduler übermittelt. Die Authentifizierungsdaten im EAP-Paket, die anfangs in unformatierter Form vorliegen, werden in eine Job-Warteschleife abgelegt und vom nächsten freien Thread übernommen. Dieser übernimmt das Parsen der Daten und die Weiterleitung an das Session Module. Für die Kommunikation in die entgegengesetzte Richtung ist der Egress Scheduler zuständig. Er übernimmt formatierte Daten vom Session Module, verpackt sie in EAP-Pakete und sendet sie an den Packet Filter. Die Pakete werden schließlich an einen Diameter Client oder einen PANA Client weitergeleitet.

Das Session Module empfängt sämtliche Authentifizierungsdaten, die den PAA über das Transport Module erreichen. Die Daten, die bereits in formatierter Form vorliegen, werden mit Hilfe der in der Session Database gespeicherten Sitzungsinformationen der Instanz der zugehörigen Sitzung, einem Session Object, zugeordnet.

Der PANA Client hat einen ähnlichen Aufbau wie der PANA Authentication Agent. Er besteht aus den gleichen drei Modulen, dem Core Module, dem Transport Module und dem Session Module (vgl. Abbildung 25).

Die Funktionsweise der Module ist denen des PAA sehr ähnlich, weshalb nur auf die wenigen Unterschiede eingegangen wird. Da der PaC keine direkte Verbindung zu einem Diameter Client hat, ist die Schnittstelle zu einem AAA-Protokoll nicht mehr notwendig. Ein weiterer Unterschied zwischen den Architekturen ist das einfachere Session Module des PaC. Clients können höchstens eine Verbindung gleichzeitig zu Agents aufbauen, weswegen sie auch nur eine einzige Sitzung haben. Aus diesem Grund ist eine Datenbank mit Sitzungsinformationen nicht mehr notwendig. Da sich jedoch das PANA Protokoll noch in einem frühen Entwicklungsstadium befindet und es Überlegungen gibt, Endgeräten die gleichzeitige Nutzung mehrerer Netze zu erlauben, kann es in einer zukünftigen Version doch noch zu einer Unterstützung mehrerer Sitzungen kommen.

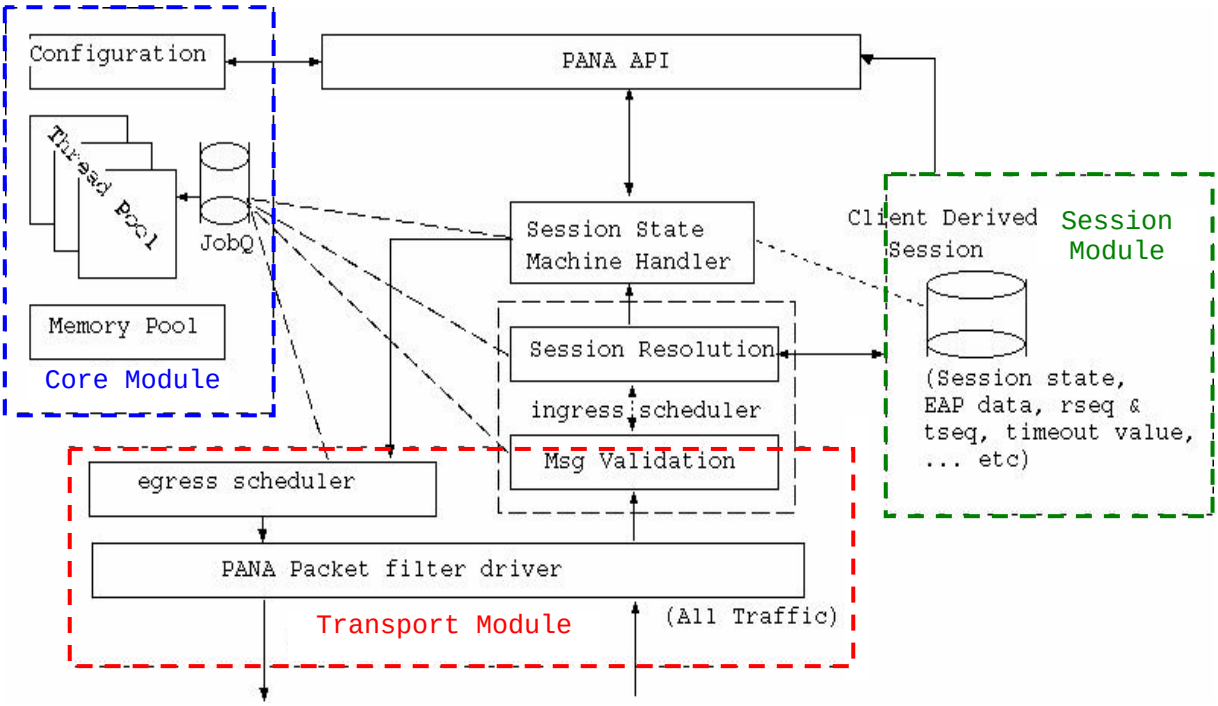


Abbildung 25: Architektur des PANA Client [43]

5.1.2 SAML-Autorität

Für die Authentifizierung über EAP-SAML ist eine Komponente notwendig, die Assertions und Artifacts generieren und verwalten kann. Bei der BMW Group wurde für diesen Zweck eine SAML-Autorität implementiert, die auf *OpenSAML* [46] basiert, einer Sammlung von Open-Source-Bibliotheken des *Internet2-Konsortiums* [47]. Die SAML Autorität ermöglicht die Erstellung, die Speicherung und die Übermittlung von SAML Nachrichten. In der Abbildung 26 sind die wichtigsten Klassen der SAML-Autorität und ihre wichtigsten Funktionen dargestellt.

Die zentrale Komponente der SAML-Autorität ist die Klasse *SAMLAuthorityService*. Sie fungiert als Schnittstelle zu allen Entitäten außerhalb der Autorität und übernimmt die Erstellung von Assertions und Artifacts. Die Entitäten rufen dafür die Funktion *createAssertionForRegistrationID* auf, die ihrerseits statische Funktionen der Klasse *AssertionFactory* verwendet. Soll beispielsweise eine Authentifizierungs-Assertion generiert werden, muss erst mit *createSubject* ein Identifikator für den zu authentifizierenden Nutzer erstellt werden. Daraufhin wird für diesen Nutzer mit der Funktion *createAuthenticationStatement* ein Statement generiert, in dem die Authentifizierungsmethode und der Zeitpunkt der Authentifizierung hinterlegt sind. Im nächsten Schritt werden die erstellten Elemente mit Hilfe der Funktion *buildAssertion* in

eine Authentifizierungs-Assertion zusammengefasst. *Schließlich* wird mit *createArtifact* eine Referenz auf diese Assertion generiert.

Abbildung 26: Fragment der SAML-Autorität

Assertions werden innerhalb der SAML-Autorität in einer SQL-Datenbank abgelegt. Die dafür verwendete Klasse heißt *SAMLAuthorityStorage* und wurde gemäß dem Singleton-Pattern [48] realisiert. Das bedeutet, dass es nur eine einzige Instanz dieser Klasse geben kann, auf die mit *getInstance* zugegriffen wird. In der Datenbank werden Assertions mit Hilfe der Funktion storeRegistrationAssertion gespeichert. Der Zugriff auf gespeicherte Elemente erfolgt über *Artifacts*, die der Funktion *findRegistrationAssertionByArtifact* übergeben werden.

Da die SAML-Autorität vertrauliche Informationen verwaltet, ist es wichtig diese Informationen möglichst gut zu schützen. Artifacts die zum Zwecke der Authentifizierung eingesetzt werden, verfügen gewöhnlich über keine Schutzmechanismen, sondern werden im Klartext versendet. Die SAML-Autorität behebt das Problem durch die Verschlüsselung und Signierung des Artifacts. Diese Aufgabe übernimmt die *IdTokenFactory*, die Artifacts in so genannte IdTokens einkapselt. Ein IdToken ist ein Byte-Array, der aus folgenden Bestandteilen aufgebaut wird:

- Einem Artifact, das für die Authentifizierung gemäß dem Artifact-Profil verwendet wird,
- Einer Sequenznummer, die gegen Wiederholungsattacken schützen soll und außerdem sicherstellt, dass Nachrichten nicht doppelt verarbeitet werden,
- Einer Zufallszahl, die gewährleistet, dass ein IdToken sich wesentlich von seinen Vorgängern und Nachfolgern unterscheidet, und somit keine Rückschlüsse auf den verwendeten Schlüssel erlaubt,
- Einer Signatur, die die Integrität, Authentizität und Nichtabstreitbarkeit des IdTokens garantiert,
- Der Identität des entsprechenden Nutzers. Diese hat die Form ID@Domain und wird für die Vermittlung des IdTokens im Netzwerk verwendet.

Die Erstellung des IdTokens erfolgt über die Funktion *createIdToken*, die das Artifact, die Sequenznummer und die Zufallszahl in einem Block verschlüsselt, den verschlüsselten Byte-Array signiert, und schließlich die Identität anhängt. Die Identität bleibt im Klartext, da sie für das Routing der Nachricht verwendet werden muss. Erhält die SAML-Autorität ein IdToken, so muss dieses überprüft und das Artifact extrahiert werden. Für diese Aufgabe stellt die Klasse *IdTokenFactory* die Funktionen *verifySignature* beziehungsweise *getArtifact* zur Verfügung. Erst nach der Entschlüsselung des Artifacts kann aus der *SAMLAuthorityStorage* die zugehörige Assertion eingeholt werden.

Mit Hilfe der vorgestellten Klassen kann somit eine Single Sign-On Authentifizierung mittels SAML durchgeführt werden. Die SAML-Autorität der BMW Group verfügt über weitere Komponenten und Funktionalitäten, die mit dem Thema der Arbeit nicht unmittelbar zusammenhängen und deswegen hier nicht erörtert werden. Es ist jedoch sehr wahrscheinlich, dass es zu einem späteren Zeitpunkt zu einer engeren Integration der beiden Konzepte kommt, weshalb im Rahmen des Ausblicks im letzten Kapitel einige dieser Funktionalitäten vorgestellt werden sollen.

5.2 Implementierte Komponenten

Das Open Diameter Projekt bietet mit seinen Bibliotheken zahlreiche Möglichkeiten einen Nutzer zu authentifizieren. Charakteristisch für diese Methoden ist jedoch die fehlende Unterstützung von Single Sign-On. Die im Rahmen dieser Diplomarbeit entwickelte Authentifizierungsmethode EAP-SAML erweitert Open Diameter um diese Funktionalität, der vor allem in Netzen der vierten Generation eine große Bedeutung zukommen wird.

EAP-SAML wurde, aufbauend auf den Open Diameter Bibliotheken, als eigenständige Authentifizierungsmethode implementiert. Dafür wurden folgende Komponenten realisiert:

- Es wurde eine Erweiterung des EAP-Protokolls mit dem entsprechenden EAP-SAML-Peer und EAP-SAML-Authentication-Server definiert.
- Für beide Entitäten wurden Zustandsmaschinen mit Zustandstabellen und Aktionsklassen implementiert.
- Da die EAP-Pakete zwischen Peer und Server über PANA und Diameter transportiert werden, musste ein Parser geschrieben werden, der die Daten in alle Richtungen konvertieren kann.
- Schließlich wurden der EAP-SAML-Peer und der EAP-SAML-Authentication-Server mit einem PANA Client beziehungsweise mit einem Diameter Server integriert.

Zu beachten ist, dass für den Network Access Server kein EAP-SAML-Passthrough-Server implementiert werden musste. Grund hierfür ist, dass der Aufbau der EAP-SAML-Nachricht es dem standardmäßig implementierten EAP-Identity-Passthrough erlaubt, die Nachrichten korrekt an das Back-End weiterzuleiten.

5.2.1 Implementierung des EAP-SAML Peers und des Authentication Servers

Das implementierte Protokoll basiert auf der Diameter-EAP Programmierschnittstelle des Open Diameter Frameworks. Eine zentrale Anforderung dieser Schnittstelle ist, dass bei neu implementierten Protokollen eine Trennung zwischen der Grundfunktionalität, die allen EAP-Methoden gemein ist, und der spezifischen Funktionalität der Methode gewährleistet wird. Dafür werden separate Zustandsmaschinen für die generischen Funktionen beziehungsweise für die spezifischen Funktionen erstellt. Die Zustandsmaschinen sind in den Klassen *EapSwitchStateMachine* und *EapMethodStateMachine* definiert, die beide die Klasse *EapStateMachine* erweitern.

Es wurden zwei Arten von *EapSwitchStateMachine*-Klassen implementiert: *EAP_SAMLPeerSwitchStateMachine,* die eine Zustandsmaschine für den EAP-SAML-Peer definiert und *EAP_SAMLAuthSwitchStateMachine,* die die Zustandsmaschine des Authentication Servers implementiert. Diese Klassen enthalten:

- Funktionen zum Versenden und Empfangen von Diameter-EAP beziehungsweise PANA-EAP Nachrichten,
- Abstrakte Callback-Funktionen, die in den Anwendungen definiert sind und aufgerufen werden, wenn die Zustandsmaschine ein eintreffendes Ereignis der Anwendung weiterleiten muss,

- Eine Authentifizierungs-Policy, die Richtlinien für den Ablauf des Authentifizierungsvorgangs über EAP-SAML festlegt. Eine Policy kann anordnen, dass mehrere Authentifizierungsvorgänge nacheinander durchgeführt werden sollen,
- Eine oder mehrere Subklassen von *EapMethodStateMachine*, in der die methodenspezifischen Funktionen enthalten sind. Definiert die Policy den Einsatz von mehreren aufeinander folgenden Authentifizierungsmethoden, so sind auch mehrere *EapMethodStateMachines* notwendig.

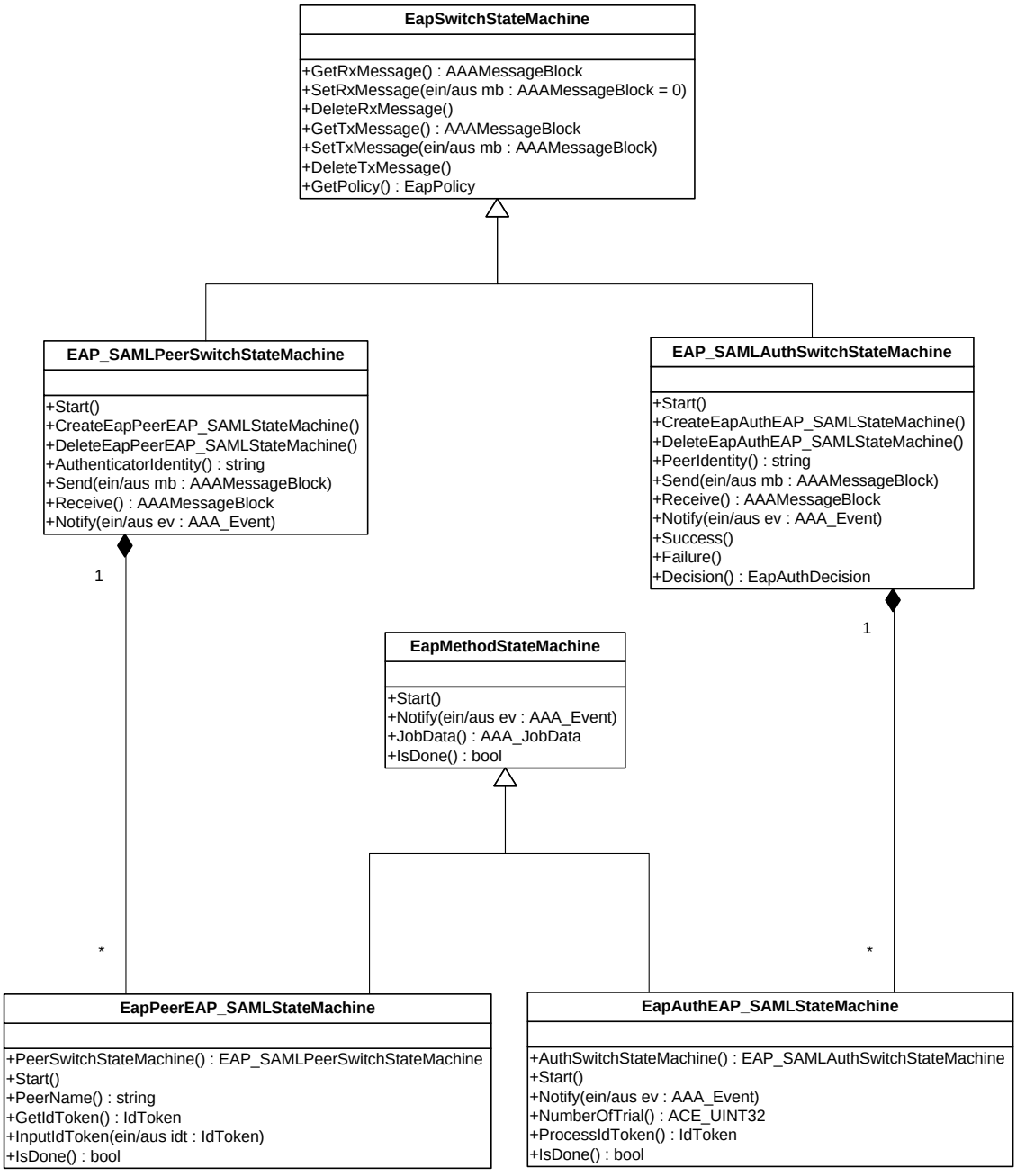


Abbildung 27: Klassendiagramm des EAP-SAML Peers und EAP-SAML Authentication Servers

Die Klasse *EapMethodStateMachine* wird von zwei Subklassen erweitert: der Klasse *EapAuthEAP_SAMLStateMachine*, die die EAP-SAML-spezifische Zustandsmaschine des Authentication Servers implementiert und der Klasse *EapPeerEAP_SAMLStateMachine*, die die Zustandsmaschine des Peers definiert. Diese Klassen sind als Attribute in *EapSwitchStateMachines* enthalten und werden von Letzteren instanziiert, sobald die Authentifizierungs-Policy die Durchführung der EAP-SAML-Methode verlangt. Diese Klassen enthalten:

- Funktionen zur Steuerung des Authentifizierungsvorgangs,
- Funktionen zur Verarbeitung der Authentifizierungsdaten,
- Einen Zeiger auf die *EapSwitchStateMachine*, die die Instanz generiert hat.

5.2.2 Zustandsmaschinen des EAP-SAML Peers und Authentication Servers

Die Funktionsweise von Zustandsmaschinen wird in dieser Arbeit mittels Zustandsübergangstabellen definiert. Jede Zustandsübergangstabelle ist eine Subklasse von *AAA_StateTable* und verfügt über eine Reihe von Zuständen, Ereignissen und Aktionen, wobei Letztere über Funktionsobjekten definiert werden. Funktionsobjekte sind Objekte, die genauso aufgerufen werden können wie Funktionen. Der Funktionsoperator () ist mit der Operatorfunktion *operator()* überladen. Obwohl sie sich wie Funktionen verhalten, verfügen Funktionsobjekte dennoch über alle Eigenschaften von Objekten, was in einer komplexen Umgebung wie der von EAP-SAML große Vorteile bringt.

Eine Zustandsübergangstabelle, die *AAA_StateTable* erweitert, verfügt über einen oder mehrere Einträge, die folgende Attribute besitzen müssen:

- *aktueller Zustand:* ist der Zustand, der eingenommen wurde bevor ein bestimmtes Ereignis eingetroffen ist,
- *Ereignis:* repräsentiert ein Ereignis, das die Zustandsmaschine im aktuellen Zustand empfangen und verarbeiten kann,
- *nächster Zustand:* ist der Zustand, in die die Zustandsmaschine nach dem Empfang des Ereignisses übergeht,
- *Aktion:* verweist auf das Funktionsobjekt, dessen Operatorfunktion aufgerufen werden soll, wenn die Zustandsmaschine ein Ereignis akzeptiert hat.

Zum besseren Verständnis der Struktur von Zustandsübergangstabelle und insbesondere der Funktionsobjekte kann der Quellcode im Anhang A.1.1 hilfreich sein. Dort sind die methodenspezifischen Zustandsmaschinen von EAP-SAML aufgeführt, die generischen SwitchStateMachines wurden aufgrund ihrer sehr hohen Komplexität nicht angehängt.

Im folgenden Abschnitt werden ebenfalls nur die methodenspezifischen Zustandsmaschinen der von *EapMethodStateMachine* abgeleiteten Klassen *EapPeerEAP_SAMLStateMachine* und *EapAuthEAP_SAMLStateMachine* behandelt. Diese Maschinen sind eingebettet in die sehr viel komplexeren Zustandsmaschinen der Klassen *EAP_SAMLPeerSwitchStateMachine* und *EAP_SAMLAuthSwitchStateMachine*, deren Zustandsdiagramme im Anhang A.2 einsehbar sind.

Die methodenspezifische Zustandsmaschine des EAP-SAML Peers, definiert in der Klasse *EapPeerEAP_SAMLStateMachine*, veranschaulicht die Abbildung 28.

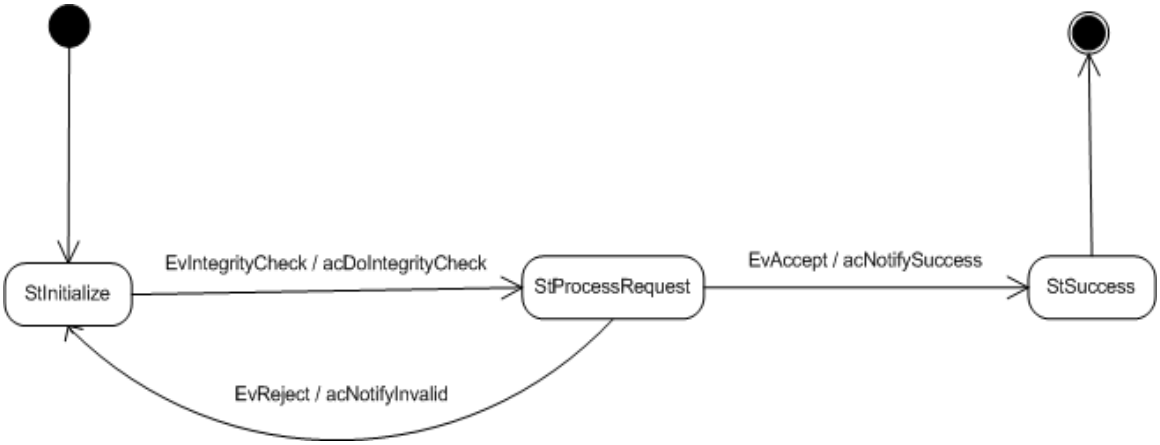


Abbildung 28: Zustandsdiagramm der EapPeerEAP_SAMLStateMachine

Wird dem Peer vom Authentication Server oder vom Passthrough Server EAP-SAML als Authentifizierungsmethode angeboten, so startet die SwitchStatMachine des Peers die Zustandsmaschine für EAP-SAML. Der Peer kann während des EAP-SAML Authentifizierungsvorgangs drei Zustände einnehmen.

- Der erste Zustand heißt *StInitialize* und stellt den Startpunkt für die Authentifizierung dar. Falls die SwitchStateMachine ein EAP-SAML Paket bekommt, verlangt sie dessen Überprüfung durch die *EapPeerEAP_SAMLStateMachine*, indem sie eine *EvIntegrityCheck*-Nachricht versendet. Die Zustandsmaschine ruft daraufhin die Operatorfunktion des Funktionsobjekts *acDoIntegrityCheck* auf, die das EAP-SAML Paket mit Hilfe des EAP-SAML Parsers konvertiert.
- Die Zustandsmaschine wechselt in den Zustand *StProcessRequest* und wartet auf das Ergebnis des Parsers. Schlägt die Konvertierung fehl, tritt das Ereignis *EvReject* ein und die Maschine kehrt zum Zustand *StInitialize* zurück. Der Authentication Server wird über das Funktionsobjekts *acNotifyInvalid* darüber informiert, dass seine Nachricht ungültig war und er somit einen neuen Versuch

starten muss. Ist jedoch die Konvertierung erfolgreich ist, tritt das Ereignis *EvAccept* ein, und der Server wird über das Funktionsobjekt *acNotifySuccess* benachrichtigt, dass seine Nachricht akzeptiert wurde.

- Wurde die Verarbeitung der EAP-SAML Nachricht erfolgreich abgeschlossen, so geht die Zustandsmaschine in den *StSuccess*-Zustand über.

Zu beachten ist, dass der Peer keinen Failure-Zustand besitzt. Das liegt daran, dass nur der Server entscheiden können soll, wann eine Authentifizierung abgebrochen wird. Allein der Grund, dass eine vom Server versendete Nachricht nicht richtig verarbeitet werden konnte, sollte nicht zum Abbruch führen. Stattdessen sollte der Server aufgefordert werden, die Nachricht zu wiederholen. Diesem obliegt die Entscheidung, ob er den Vorgang fortsetzen will oder nicht.

Die methodenspezifische Zustandsmaschine des Authentication Servers, die in der Klasse *EapAuthEAP_SAMLStateMachine* definiert ist, veranschaulicht das Zustandsdiagramm in der Abbildung 29.

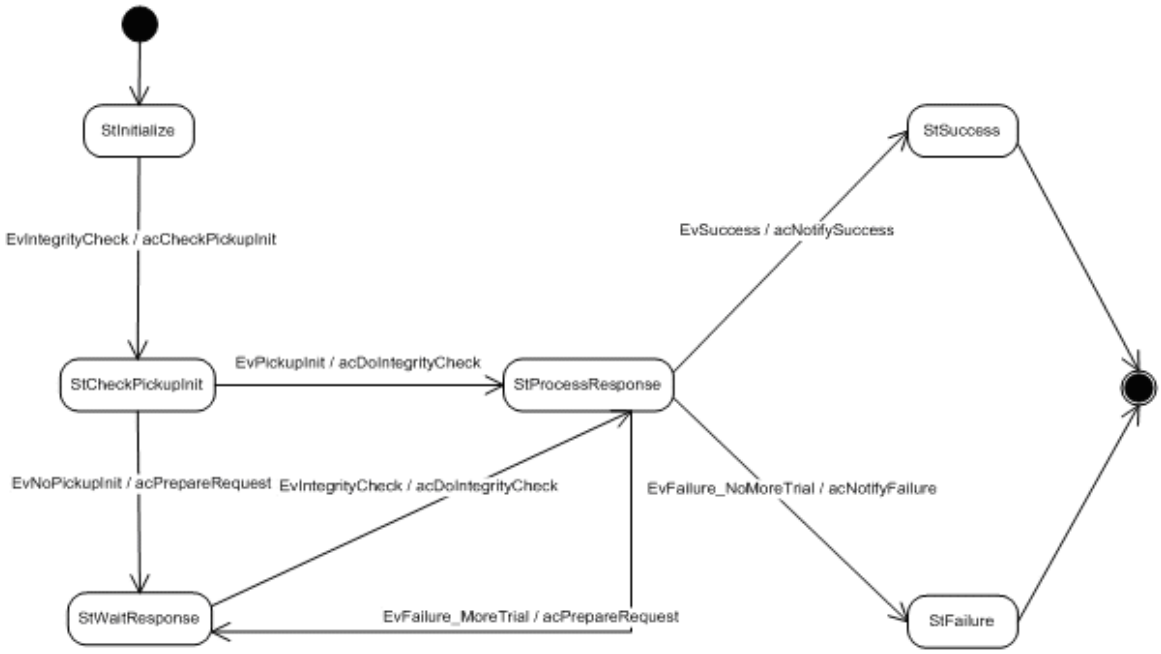


Abbildung 29: Zustandsdiagramm der EapAuthEAP_SAMLStateMachine

Die Zustandsmaschine des Servers ist um einiges komplexer als die des Peers und soll auch detailliert behandelt werden. Die *EapAuthEAP_SAMLStateMachine* ist ebenso wie im Falle des Peers in einer SwitchStateMachine integriert. Diese kann mehrere Authentifizierungsmethoden unterstützen und somit mehrere methodenspezifische Zustandsmaschinen besitzen. Ähnlich wie beim Peer wird die

EapAuthEAP_SAMLStateMachine aktiviert, wenn die Authentifizierung eines Nutzers über EAP-SAML laufen soll. Die EAP-SAML Zustandsmaschine verfügt über folgende Zustände:

- Der erste Zustand heißt *StInitialize* und ist der Startpunkt einer Authentifizierung mittels EAP-SAML. In diesem Zustand kann das Ereignis *EvIntegrityCheck* verarbeitet werden. Falls das Ereignis eintritt, wird die Operatorfunktion des Funktionsobjekts *acCheckPickupInit* aufgerufen. Diese Funktion überprüft, ob die aktuelle Sitzung eine durch den Passthrough Server initiierte Nutzer-Authentifizierung unterstützt. Insbesondere muss festgestellt werden, ob eine Antwort vom Nutzer an den Passthrough Server von diesem an den Authentication Server weitergeleitet werden kann, ohne dass der Authentication Server davor eine Anfrage gestellt hätte.
- Während der Überprüfung der Merkmale des Authentication Servers befindet sich die Maschine im Zustand *StCheckPickupInit*. Wird festgestellt, dass der Authentication Server eine vom Passthrough gestartete Authentifizierung fortsetzen kann, tritt das Ereignis *EvPickupInit* ein, und die Operatorfunktion des Funktionsobjekts *acDoIntegrityCheck* wird aufgerufen. Diese überprüft mittels des EAP-SAML Parsers, ob das empfangene Paket fehlerfrei ist, versucht es aus dem Byte-Format in ein IdToken umzuwandeln und weiter zu verarbeiten. Die Maschine würde daraufhin in den Zustand *StProcessResponse* übergehen. Kann eine vom Passthrough gestartete Authentifizierung jedoch nicht fortgesetzt werden, tritt das Ereignis *EvNoPickupInit* und das Funktionsobjekt *acPrepareRequest* wird verwendet. Dabei wird ein EAP-SAML Request erstellt und über den Passthrough Server an den Nutzer versendet. Die Maschine würde dann in den Zustand *StWaitResponse* übergehen.
- Im Zustand *StWaitResponse* verbleibt der Server so lange, bis eine Antwort auf die mittels *acPrepareRequest* verschickte Anfrage eintrifft. Dies wird über das Ereignis *EvIntegrityCheck* signalisiert und veranlasst die Überprüfung der Nachricht mit dem Funktionsobjekt *acDoIntegrityCheck*. Der Server wechselt damit in den Zustand *StProcessResponse*.
- Der Server nimmt den Zustand *StProcessResponse* ein, so lange das empfangene EAP-SAML Paket vom Funktionsobjekt *acDoIntegrityCheck* verarbeitet wird. Wird die Überprüfung erfolgreich abgeschlossen, so tritt das Ereignis *EvSuccess* ein und das Funktionsobjekt *acNotifySuccess* sendet eine Erfolgsmeldung an die übergeordnete SwitchStateMachine, die ihrerseits diese Meldung an Passthrough Server und Nutzer weiterleitet. Endet die Operatormethode von *acDoIntegrityCheck* mit einem Fehler, muss überprüft werden, ob weitere Versuche erlaubt sind. Falls ja, tritt das Ereignis *EvFailure_MoreTrial* ein, der

Server versendet mit *acPrepareRequest* eine neue Anfrage und geht in den Zustand *StWaitResponse* über. Falls keine weiteren Versuche erlaubt sind, tritt das Ereignis *EvFailure_NoMoreTrial* ein, der Nutzer und der Passthrough Server werden mittels *acNotifyFailure* über den Abbruch informiert und der Server wechselt in den Zustand *StFailure* über.

- Der Zustand *StSuccess* wird erreicht, wenn der Nutzer sich über EAP-SAML erfolgreich authentifizieren konnte.
- Der Server geht in den Zustand *StFailure* über, wenn ein Nutzer sich nach wiederholten Versuchen (in der Implementierung: 3 Versuche) nicht authentifizieren konnte, entweder weil seine EAP-SAML Pakete nicht geparst werden konnten oder weil die enthaltenen Authentifizierungsdaten falsch waren.

In diesem Abschnitt wurden nur die methodenspezifischen Zustandsmaschinen des Peers und des Authentication Servers vorgestellt. Der Passthrough Server dagegen wurde nicht behandelt, da er keine spezielle Zustandsmaschine für EAP-SAML enthält. Das Format der EAP-SAML Pakete wurde so entwickelt, dass sie auch von dem standardmäßig implementierten Passthrough Server verarbeitet und weitergeleitet werden können. Aus diesem Grund werden auch im nächsten Abschnitt nur der Client und der Authentifizierungsserver vorgestellt.

5.2.3 Implementierung des Clients und des Authentifizierungsservers

Bei der Realisierung des Clients und des Authentifizierungsservers mussten mehrere Technologien miteinander verbunden werden. Sowohl der auf dem mobilen Endgerät befindliche Client als auch der im Back-End einer Diameter-Infrastruktur stehende Server implementieren das EAP-SAML Protokoll. Dabei enthält der Letztere einen EAP-SAML Authentication Server, der Client enthält einen EAP-SAML Peer.

Da der mobile Client sich jedoch über unterschiedliche Übertragungstechnologien beim Network Access Server des Netzbetreibers anmelden können soll, muss die Kommunikation zwischen diesen beiden Entitäten über das PANA Protokoll laufen. Der Client beinhaltet dafür einen PANA Client, wogegen der Network Access Server einen PANA Authentication Agent enthält.

Innerhalb des Netzwerks des Netzbetreibers müssen die EAP-SAML Pakete über ein AAA-Protokoll, in diesem Fall über Diameter, laufen. Die Portierung der EAP-SAML Pakete von PANA auf Diameter findet im Network Access Server statt. Dieser beinhaltet dafür einen Diameter Client und vermittelt die EAP-SAML Pakete an den entsprechenden Diameter Server weiter. Der Diameter Server läuft innerhalb eines

Authentifizierungsservers, der an einem beliebigen Ort im Back-End steht. Mit Hilfe des ebenfalls im Authentifizierungsserver enthaltenen EAP-SAML Authentication Servers, kann schließlich die Authentifizierung des Nutzers durchgeführt werden. Im Folgenden soll die Struktur des Clients und des Authentifizierungsservers näher betrachtet werden.

Die Hauptkomponente des Clients ist die Klasse *PeerApplication*. Diese implementiert die Schnittstelle *PANA_ClientEventInterface* und instanziiert die Klasse *PANA_PacSession*. Dadurch verfügt die Anwendung *PeerApplication* über die benötigte Funktionalität eines PANA Client. Für die Unterstützung von EAP-SAML instanziiert *PeerApplication* die im Abschnitt 5.2.1 vorgestellte Klasse *EAP_SAMLPeerSwitchStateMachine.* Wie gezeigt wurde, definiert diese Klasse den generischen Teil einer EAP-Zustandsmaschine. Für die methodenspezifischen Funktionen wird zusätzlich die Klasse *EapPeerEAP_SAMLStateMachine* integriert. In der *main* Funktion werden schließlich die Verbindungsinformationen aus einer Konfigurationsdatei ausgelesen und der PANA Client zusammen mit den EAP-SAML Zustandsmaschinen gestartet.

Der Authentifizierungsserver hat einen ähnlichen Aufbau wie der Client. Die Hauptkomponente heißt hier *DiameterEapServerSession*. Server Sessions können jedoch im Gegensatz zu PANA Clients nicht von der Anwendung instanziiert werden, sondern werden vom Framework erstellt. Dafür wird die Klasse *AAAServerSessionClassFactory* verwendet, dem die *DiameterEapServerSession* als Template-Parameter übergeben wird. Es gibt erneut zwei Zustandsmaschinen für die EAP-Methode. Die Klasse *EAP_SAMLAuthSwitchStateMachine* (vgl. 5.2.1) definiert die für alle EAP-Methoden gültige Zustandsmaschine, während die methodenspezifische Zustandsmaschine für EAP-SAML in der Klasse *EapAuthEAP_SAMLStateMachine* definiert ist. In der *main* Funktion des Authentifizierungsservers werden die Initialisierungsdaten aus einer Konfigurationsdatei ausgelesen und der Diameter Server zusammen mit den Zustandsmaschinen des EAP Authentication Servers gestartet.

Abschließend soll noch einmal darauf hingewiesen werden, dass sowohl der Client als auch der Server mehrere Authentifizierungsmethoden unterstützen können. Es ist zum Beispiel notwendig, dass vor der allerersten Authentifizierung über EAP-SAML das Artifact an den Client übermittelt wird. Dies kann nur geschehen, wenn der Client sich beim ersten Mal mittels einer anderen Methode, beispielsweise EAP-TLS authentifiziert, um sein Artifact zu bekommen. Wie das konkret umgesetzt wird, zeigt der Quellcode im Anhang A.1.2.

5.3 Zusammenfassung

Im fünften Kapitel dieser Diplomarbeit wurde die praktische Umsetzung der im vierten Kapitel entworfenen Konzepte dargestellt. Im ersten Teil wurden die zwei wichtigsten Software-Komponenten, auf denen die Implementierung aufbaut, vorgestellt. Die erste Komponente ist die bei der BMW Group entwickelte SAML-Autorität, die für die Erstellung und Verwaltung der in Form von SAML Assertions und Artifacts vorliegenden Authentifizierungsdaten benötigt wird. Die zweite Software ist das Open Diameter Framework, das unter anderem die für diese Arbeit unverzichtbaren Protokolle Diameter, NASREQ und PANA implementiert.
Im zweiten Teil des Kapitels wurden einige wichtige Komponenten vorgestellt, die im Rahmen dieser Arbeit implementiert wurden. Herzstück dieser Komponenten sind der EAP-SAML Peer und EAP-SAML Authentication Server. Beide wurden detailliert beschrieben und ihre jeweiligen Zustandsmaschinen ausführlich erklärt. Abschließend wurde gezeigt, wie die EAP-SAML Komponenten in den Client, der einen PANA Client enthält, beziehungsweise in den Authentifizierungsserver, der einen Diameter Server beinhaltet, integriert wurden.

6. Zusammenfassung und Ausblick

In Rahmen dieser Diplomarbeit wurde mit EAP-SAML eine Authentifizierungsmethode entwickelt, die den komplexen Anforderungen (vgl. Kapitel 4.2) mobiler Netze der vierten Generation genügt. Durch die Verwendung der Protokolle EAP, PANA und Diameter, kann der Transport der Authentifizierungsdaten in den heterogenen Infrastrukturen dieser Netze gewährleistet werden. Mit dem Einsatz von SAML, das den Austausch von Sicherheitsinformationen zwischen Domänen ermöglicht, können Föderationen aufgebaut werden, innerhalb derer Nutzer ohne eigenes Zutun authentifiziert werden, was zu einer unterbrechungsfreien Dienstnutzung auch bei Domänenwechsel führt.

Föderationen werden von Föderations-Managern verwaltet, die in dieser Arbeit als Diameter Proxy Agents entworfen wurden. In dieser Form haben sie jedoch eine sehr begrenzte Funktionalität, da sie nur für die Vermittlung der Nachrichten anhand von vorher festgelegten, also statischen, Regeln zuständig sind. Eine Erweiterung dieses Konzepts wäre mit Hilfe der XML-basierten Sprache *Extensible Access Control Markup Language (XACML)* [49] möglich. XACML ist ebenso wie SAML eine Spezifikation des OASIS Konsortiums und wird für die Regeldefinition und Zugangskontrolle verwendet. Mit XACML könnte dadurch ein Föderationsmanager dynamisch darüber entscheiden, ob ein Nutzer Zugang zu einem bestimmten Netzbereich, also einer Domäne, erhalten soll oder nicht.

Ein weiterer Bereich, in dem das gegenwärtige Konzept erweitert werden sollte, ist die Unterstützung mehrerer Identitäten je Nutzer. Derzeit werden unterschiedliche Identitäten, die Nutzer in verschiedenen Domänen haben können, nicht zentral verwaltet, so dass eine Zuordnung zwischen diesen Identitäten nicht möglich ist. Eine einheitliche Identität für alle Domänen kann andererseits aus Datenschutzgründen problematisch sein. Eine Lösung für dieses Problem bieten Weyl et al. mit dem Konzept der *Virtual Idenitity* [50]. Virtuelle Identitäten ermöglichen es einem Nutzer für verschiedene Domänen unterschiedliche, selbst definierte Identitäten zu nutzen. Dadurch bleibt fremden Domänen die wahre Identität des Nutzers verborgen. Die einzige Instanz, die eine Zuordnung zwischen der wahren Identität, der *Registration Identity*, und der virtuellen Identität durchführen kann, ist die Heimatdomäne selbst. Fremde Domänen werden sich so zwecks Authentifizierung

des Nutzers und Abrechnung der Dienste stets an die Heimatdomäne wenden müssen, so dass die Identitätsdaten geschützt bleiben.

Zum Schluss wird eine weitere Verbesserungsmöglichkeit aufgezeigt, durch die die Sicherheit der entwickelten Authentifizierungsmethode zusätzlich erhöht werden kann. Wie in Kapitel 4.2 dargelegt wurde, verfügt EAP-SAML sowohl über eine Ende-zu-Ende Sicherheit als auch über eine Hop-zu-Hop Sicherheit. Allerdings fehlt der Methode die Unterstützung für die beidseitige Authentifizierung, mit der sichergestellt wird, dass Clients nur mit solchen Servern vertrauliche Daten austauschen, die sich im Vornherein bei ihnen authentifiziert haben. Ohne eine beidseitige Authentifizierung kann sich ein Angreifer als Authentifizierungsserver ausgeben und vom Nutzer die Durchführung einer Authentifizierung verlangen. Auch wenn dieser die Informationen in verschlüsselter Form übermittelt, könnte der Angreifer nach einer sehr hohen Anzahl von Authentifizierungsvorgängen die Nachricht entschlüsseln. Dies ist ein bekanntes Problem des GSM-Standards und soll in zukünftigen Netzen vermieden werden. Deswegen muss sichergestellt werden, dass Nutzer sich nur bei solchen Entitäten authentifizieren, die sich im Vorfeld bei ihnen authentifiziert haben. Dafür kann der Nutzer ein Zertifikat des Servers speichern und verlangen, dass der Server sich beispielsweise über EAP-TLS authentifiziert. Im Rahmen dieser Authentifizierung kann ein Hashwert an den Nutzer gesendet werden, der durch die Anwendung einer Hashfunktion auf das Artifact des Nutzers generiert wird. Bei einer Reauthentifizierung würde im ersten Schritt der Authentifizierungsserver diesen Hashwert präsentieren, um das Vertrauen des Nutzers zu erlangen. Erst dann würde der Nutzer sein Artifact übermitteln. Ein Angreifer kann sich somit nicht mehr als Authentifizierungsserver ausgeben, da er den Hashwert des Artifacts nicht besitzen kann. Zwar kann in EAP-SAML mit dem Einsatz des IdTokens, der durch eine sich permanent ändernde Zufallszahl keine Rückschlüsse auf die Daten und die Beschaffenheit des verwendeten Schlüssels zulassen soll, eine recht hohe Sicherheit auch gegen den beschriebenen Angriff erreicht werden. Eine Erweiterung von EAP-SAML um die beidseitige Authentifizierung ist dennoch mit Blick auf die höheren Sicherheitsrisiken in den fragmentierten Netzwerken der vierten Generation empfehlenswert.

A. Anhang

A.1 Auszüge aus dem Quellcode

Im Folgenden werden wichtige Fragmente des Quellcodes gezeigt. Im ersten Abschnitt werden Teile des EAP-SAML Peers und des EAP-SAML Authentication Servers angeführt

A.1.1 EAP-SAML Peer und EAP-SAML Authentication Server

Der folgende Quellcode zeigt die Implementierung der Zustandsübergangstabelle für den EAP-SAML Peer und der dabei eingesetzten Funktionsobjekte (Action Classes).

```
#include <string>
#include <ace/Singleton.h>
#include <ace/OS_String.h>
#include <ace/Message_Block.h>
#include <ace/Thread.h>
#include "eap.hxx"
#include "eap_method_registrar.hxx"
#include "eap_saml_parser.hxx"
#include "eap_parser.hxx"
#include "eap_log.hxx"
#include "eap_saml.hxx"

[…]

// Peer Action class.
class EapPeerEAP_SAMLAction : public
AAA_Action<EapPeerEAP_SAMLStateMachine>
{
 public:
  virtual void operator()(EapPeerEAP_SAMLStateMachine&) {}
 protected:
  EapPeerEAP_SAMLAction() {}
  virtual ~EapPeerEAP_SAMLAction() {}
};

// State table used by EapPeerEAP_SAMLStateMachine.
class EapPeerEAP_SAMLStateTable_S :
  public AAA_StateTable<EapPeerEAP_SAMLStateMachine>

{
  friend class ACE_Singleton<EapPeerEAP_SAMLStateTable_S,
                     ACE_Recursive_Thread_Mutex>;

private:
  // defined as a leaf class
  class AcDoIntegrityCheck : public EapPeerEAP_SAMLAction
  {
    void operator()(EapPeerEAP_SAMLStateMachine &msm)
    {
```

```
      EapPeerSwitchStateMachine &ssm = msm.PeerSwitchStateMachine();

      // Dequeue the message from the processing queue.
       AAAMessageBlock *msg = ssm.GetRxMessage();

      // Check the Artifact
      IDToken id;
      EapResponseEAP_SAMLParser parser;
      parser.setAppData(&id);
      parser.setRawData(msg);
      try { parser.parseRawToApp(); }
      catch (...)
      {
       EAP_LOG(LM_ERROR, "PeerEAP_SAML: Parse error.\n");
       msm.Event(EvSgReject);
       return;
       }

       ssm.AuthenticatorIdentity() = id.getArtifact();

      EapPeerEAP_SAMLStateMachine::ProcessIdentityResult result =
                msm.ProcessIdentity(ssm.AuthenticatorIdentity());

      if (result==EapPeerEAP_SAMLStateMachine::Success)
      {
      msm.Event(EvSgAccept);
          return;
      }
       msm.Event(EvSgReject);
      }
  };

  class AcNotifySuccess : public EapPeerEAP_SAMLAction
  {
    void operator()(EapPeerEAP_SAMLStateMachine &msm)
    {
      //EAP_LOG(LM_DEBUG, "PeerEAP_SAMLStateTable: AcNotifySuccess
1.\n");
      EapPeerSwitchStateMachine &ssm = msm.PeerSwitchStateMachine();
      IDToken id;

      AAAMessageBlock *msg = AAAMessageBlock::Acquire(5);

      EapResponseEAP_SAMLParser parser;
      parser.setAppData(&id);
      parser.setRawData(msg);
       parser.parseRawToApp();

      // Set the message to the session.
      ssm.SetTxMessage(msg);

      ssm.Decision() = EapPeerSwitchStateMachine::UNCOND_SUCC;
       //EAP_LOG(LM_DEBUG, "PeerEAP_SAMLStateTable: AcNotifySuccess
1.\n");

      msm.IsDone() = true;
      ssm.Event(EapPeerSwitchStateMachine::EvSgValidReq);
    }
  };

  class AcNotifyInvalid : public EapPeerEAP_SAMLAction
```

```
  {
    void operator()(EapPeerEAP_SAMLStateMachine &msm)
    {
      EapPeerSwitchStateMachine &ssm = msm.PeerSwitchStateMachine();
      ssm.Event(EapPeerSwitchStateMachine::EvSgInvalidReq);
    }
  };

  enum {
    EvSgAccept,
    EvSgReject,
  };
  enum state {
    StInitialize,
    StProcessRequest,
    StSuccess
  };

  AcDoIntegrityCheck acDoIntegrityCheck;
  AcNotifySuccess acNotifySuccess;
  AcNotifyInvalid acNotifyInvalid;

  EapPeerEAP_SAMLStateTable_S()
  {
    AddStateTableEntry(StInitialize,
                EapMethodStateMachine::EvSgIntegrityCheck,
                StProcessRequest, acDoIntegrityCheck);
    AddStateTableEntry(StProcessRequest, EvSgReject,
                StInitialize, acNotifyInvalid);
    AddStateTableEntry(StProcessRequest, EvSgAccept,
                StSuccess, acNotifySuccess);
    AddWildcardStateTableEntry(StInitialize, StInitialize);
    AddWildcardStateTableEntry(StProcessRequest, StProcessRequest);
    AddWildcardStateTableEntry(StSuccess, StSuccess);

    InitialState(StInitialize);
  } // leaf class
  ~EapPeerEAP_SAMLStateTable_S() {}
};
[…]
```

Folgender Sourcecode zeigt, wie ein EAP-SAML Peer mittels einer *EapSwitchStateMachine* und einer *EapPeerEAP_SAMLStateMachine* mit der Zustandsübegangstabelle *EapPeerEAP_SAMLStateTable* erstellt wird.

```
[…]
typedef ACE_Singleton<EapPeerEAP_SAMLStateTable_S,
ACE_Recursive_Thread_Mutex>
EapPeerEAP_SAMLStateTable;

EapPeerEAP_SAMLStateMachine::EapPeerEAP_SAMLStateMachine
(EapSwitchStateMachine &s)
  : EapMethodStateMachine(s),
    EapStateMachine<EapPeerEAP_SAMLStateMachine>
  (*this, *EapPeerEAP_SAMLStateTable::instance(), s.Reactor(), s,
"EAP_SAML(peer)"), nTrial(0){}
[…]
```

Der folgende Quellcode zeigt die Implementierung der Zustandsübergangstabelle für den EAP-SAML Authentication Server und der dabei eingesetzten Funktionsobjekte (Action Classes).

```
#include <string>
#include <ace/Singleton.h>
#include <ace/OS_String.h>
#include <ace/Message_Block.h>
#include <ace/Thread.h>
#include "eap.hxx"
#include "eap_method_registrar.hxx"
#include "eap_saml_parser.hxx"
#include "eap_parser.hxx"
#include "eap_log.hxx"
#include "eap_saml.hxx"

[…]
// Action class.
class EapAuthEAP_SAMLAction : public
AAA_Action<EapAuthEAP_SAMLStateMachine>
{
 public:
  virtual void operator()(EapAuthEAP_SAMLStateMachine&) {}
 protected:
  EapAuthEAP_SAMLAction() {}
  virtual ~EapAuthEAP_SAMLAction() {}
};


// State table used by EapAuthEAP_SAMLStateMachine.
class EapAuthEAP_SAMLStateTable_S :
  public AAA_StateTable<EapAuthEAP_SAMLStateMachine>

{
  friend class ACE_Singleton<EapAuthEAP_SAMLStateTable_S,
                     ACE_Recursive_Thread_Mutex>;

private:

  class AcCheckPickUpInit : public EapAuthEAP_SAMLAction
  {
    void operator()(EapAuthEAP_SAMLStateMachine &msm)
    {
      EapAuthSwitchStateMachine &ssm = msm.AuthSwitchStateMachine();
      //EAP_LOG(LM_DEBUG, "AuthEAP_SAMLStateTable: Check pickup
init.\n");

      if (ssm.IsEapBackend() && !ssm.NeedInitialRequestToSend())
     msm.Event(EvSgPickUpInit);
      else
     msm.Event(EvSgNoPickUpInit);
    }
  };

  class AcDoIntegrityCheck  : public EapAuthEAP_SAMLAction
  {
    void operator()(EapAuthEAP_SAMLStateMachine &msm)
    {
      EapAuthSwitchStateMachine &ssm = msm.AuthSwitchStateMachine();
```

```
    EAP_LOG(LM_DEBUG, "AuthEAP_SAMLStateMachine: Process Message\n");

    // Dequeue the message from the processing queue.
    AAAMessageBlock *msg = ssm.GetRxMessage();

    // Check the Artifact
    IDToken id;
    EapResponseEAP_SAMLParser parser;
    parser.setAppData(&id);
    parser.setRawData(msg);
    try { parser.parseRawToApp(); }
    catch (...) {
   if (msm.NumberOfTrial()++ < msm.maxTrial) {
     msm.Event(EvSgFailure_MoreTrial);
   }
   else {
     msm.Event(EvSgFailure_NoMoreTrial);
   }
   return;
    }

    ssm.PeerIdentity() = id.getArtifact();

    EapAuthEAP_SAMLStateMachine::ProcessIdentityResult result =
   msm.ProcessIdentity(ssm.PeerIdentity());
    if (result==EapAuthEAP_SAMLStateMachine::Success)
    {
     msm.Event(EvSgSuccess);
       return;
    }
    else if (msm.NumberOfTrial()++ < msm.maxTrial)
    {
        msm.Event(EvSgFailure_MoreTrial);
    }
        else
        {
             msm.Event(EvSgFailure_NoMoreTrial);
        }
  }
};

class AcNotifySuccess : public EapAuthEAP_SAMLAction
{
  void operator()(EapAuthEAP_SAMLStateMachine &msm)
  {
    EapAuthSwitchStateMachine &ssm = msm.AuthSwitchStateMachine();

    ssm.Policy().Update(EapContinuedPolicyElement::PolicyOnSuccess);
    msm.IsDone() = true;
    ssm.Notify(EapAuthSwitchStateMachine::EvSgValidResp);
    EAP_LOG(LM_DEBUG, "AuthEAP_SAMLStateMachine: Success\n");

  }
};

class AcNotifyFailure : public EapAuthEAP_SAMLAction
{
  void operator()(EapAuthEAP_SAMLStateMachine &msm)
  {
    EAP_LOG(LM_DEBUG, "AuthEAP_SAMLStateMachine: Notify Failure.\n");
```

```
      EapAuthSwitchStateMachine &ssm = msm.AuthSwitchStateMachine();
      ssm.Policy().Update(EapContinuedPolicyElement::PolicyOnFailure);
      msm.IsDone() = true;
      ssm.Notify(EapAuthSwitchStateMachine::EvSgInvalidResp);
    }
  };

  class AcPrepareRequest : public EapAuthEAP_SAMLAction
  {
    void operator()(EapAuthEAP_SAMLStateMachine &msm)
    {
      //EAP_LOG(LM_DEBUG, "AuthEAP_SAMLStateMachine:
AcPrepareRequest.\n");
      EapAuthSwitchStateMachine &ssm = msm.AuthSwitchStateMachine();
      AAAMessageBlock *msg = AAAMessageBlock::Acquire(5);
      EapRequest req(EapType(12));

      // Use parser to set Type field.
      EapRequestParser parser;
      parser.setAppData(&req);
      parser.setRawData(msg);
      parser.parseAppToRaw();

      ssm.SetTxMessage(msg);
      ssm.Notify(EapAuthSwitchStateMachine::EvSgValidResp);
    }
  };

  enum {
    EvSgSuccess,
    EvSgFailure_MoreTrial,
    EvSgFailure_NoMoreTrial,
    EvSgPickUpInit,
    EvSgNoPickUpInit,
  };

  enum state {
    StInitialize,
    StCheckPickUpInit,
    StWaitResponse,
    StProcessResponse,
    StSuccess,
    StFailure
  };

  AcCheckPickUpInit acCheckPickUpInit;
  AcDoIntegrityCheck acDoIntegrityCheck;
  AcNotifySuccess acNotifySuccess;
  AcNotifyFailure acNotifyFailure;
  AcPrepareRequest acPrepareRequest;


  EapAuthEAP_SAMLStateTable_S()
  {
    AddStateTableEntry(StInitialize,
                EapMethodStateMachine::EvSgIntegrityCheck,
                StCheckPickUpInit, acCheckPickUpInit);
    AddStateTableEntry(StCheckPickUpInit, EvSgPickUpInit,
                StProcessResponse, acDoIntegrityCheck);
    AddStateTableEntry(StCheckPickUpInit, EvSgNoPickUpInit,
                StWaitResponse, acPrepareRequest);
```

```
    AddStateTableEntry(StWaitResponse,
                EapMethodStateMachine::EvSgIntegrityCheck,
                StProcessResponse, acDoIntegrityCheck);
    AddStateTableEntry(StProcessResponse, EvSgSuccess,
                StSuccess, acNotifySuccess);
    AddStateTableEntry(StProcessResponse, EvSgFailure_MoreTrial,
                StWaitResponse, acPrepareRequest);
    AddStateTableEntry(StProcessResponse, EvSgFailure_NoMoreTrial,
                StFailure, acNotifyFailure);
    AddWildcardStateTableEntry(StSuccess, StSuccess);
    AddWildcardStateTableEntry(StFailure, StFailure);
    InitialState(StInitialize);
  }
  ~EapAuthEAP_SAMLStateTable_S() {}
};
[…]
```

Folgender Sourcecode zeigt, wie ein EAP-SAML Authentication Server mittels einer *EapSwitchStateMachine* und einer *EapAuthEAP_SAMLStateMachine* mit der Zustandsübegangstabelle *EapAuthEAP_SAMLStateTable* erstellt wird.

```
[…]
typedef ACE_Singleton<EapAuthEAP_SAMLStateTable_S,
ACE_Recursive_Thread_Mutex>
EapAuthEAP_SAMLStateTable;

const ACE_UINT32 EapAuthEAP_SAMLStateMachine::maxTrial = 3;

EapAuthEAP_SAMLStateMachine::EapAuthEAP_SAMLStateMachine
(EapSwitchStateMachine &s)
  : EapMethodStateMachine(s),
    EapStateMachine<EapAuthEAP_SAMLStateMachine>
  (*this, *EapAuthEAP_SAMLStateTable::instance(), s.Reactor(), s,
"EAP_SAML(authenticator)"),
    nTrial(0){}
[…]
```

A.1.2 Client und Authentifizierungsserver

Folgender Quellcode zeigt ein Fragment des sich auf dem mobilen Endgerät befindlichen Client, der neben einem PANA Client einen EAP-SAML Peer enthält.

```
#include <iostream>
#include <ace/OS.h>
#include <ace/Signal.h>
#include <ace/Event_Handler.h>
#include <ace/Get_Opt.h>
#include <boost/shared_ptr.hpp>
#include "eap.hxx"
#include "eap_peerfsm.hxx"
#include "eap_authfsm.hxx"
#include "eap_policy.hxx"
#include "eap_method_registrar.hxx"
#include "eap_fsm.hxx"
#include "eap_log.hxx"
#include "eap_saml.hxx"
#include "pana_client_fsm.h"
```

```
[…]
class PeerApplication : public AAA_JobData,
                        public PANA_ClientEventInterface
{
 public:
  PeerApplication(PANA_Node &n, ACE_Semaphore &sem) :
    pacSession(n, *this),
    handle(EapJobHandle
    (AAA_GroupedJob::Create(n.Task().Job(),this, ,peer"))),
     eap(boost::shared_ptr<MyPeerSwitchStateMachine>
     (new MyPeerSwitchStateMachine(*n.Task().reactor(), handle))),
     semaphore(sem),
      samlMethod(EapContinuedPolicyElement(EapType(12)))
  {
    eap->Policy().InitialPolicyElement(&samlMethod);
  }
  virtual ~PeerApplication() { }

  MyPeerSwitchStateMachine& Eap() { return *eap; }

  ACE_Semaphore& Semaphore() { return semaphore; }

  void EapStart() {
     eap->Stop();
     eap->Start();
  }
  void EapRequest(AAAMessageBlock *request,
                  PANA_PINFO provider,
                  const PANA_CfgProviderInfo *pInfo) {
     eap->Receive(request);
  }
  void EapRequest(AAAMessageBlock *request,
                  ACE_UINT32 resultCode,
                  ACE_UINT32 pcap,
                  PANA_DeviceIdContainer &epId) {
     eap->Receive(request);

     PANA_DeviceIdIterator i = epId.begin();
     for (;i != epId.end(); i++) {
          PANA_DeviceId *id = (*i);

          char buf[256];
          ACE_INET_Addr addr;
          PANA_DeviceIdConverter::FormatToAddr(*id, addr);
          addr.addr_to_string(buf, sizeof(buf));

          std::cout << "EP device id: " << id->type();
          std::cout << " value: ";
          std::cout << buf << std::endl;
     }
  }
[…]
 private:
  PANA_PacSession pacSession;
  EapJobHandle handle;
  boost::shared_ptr<MyPeerSwitchStateMachine> eap;
  ACE_Semaphore &semaphore;

EapContinuedPolicyElement samlMethod;
};
[…]
```

Folgender Quellcode zeigt ein Fragment des sich im Back-End befindlichen Authentifizierungsservers, der neben einem Diameter Server einen EAP-SAML Authentication Server enthält.

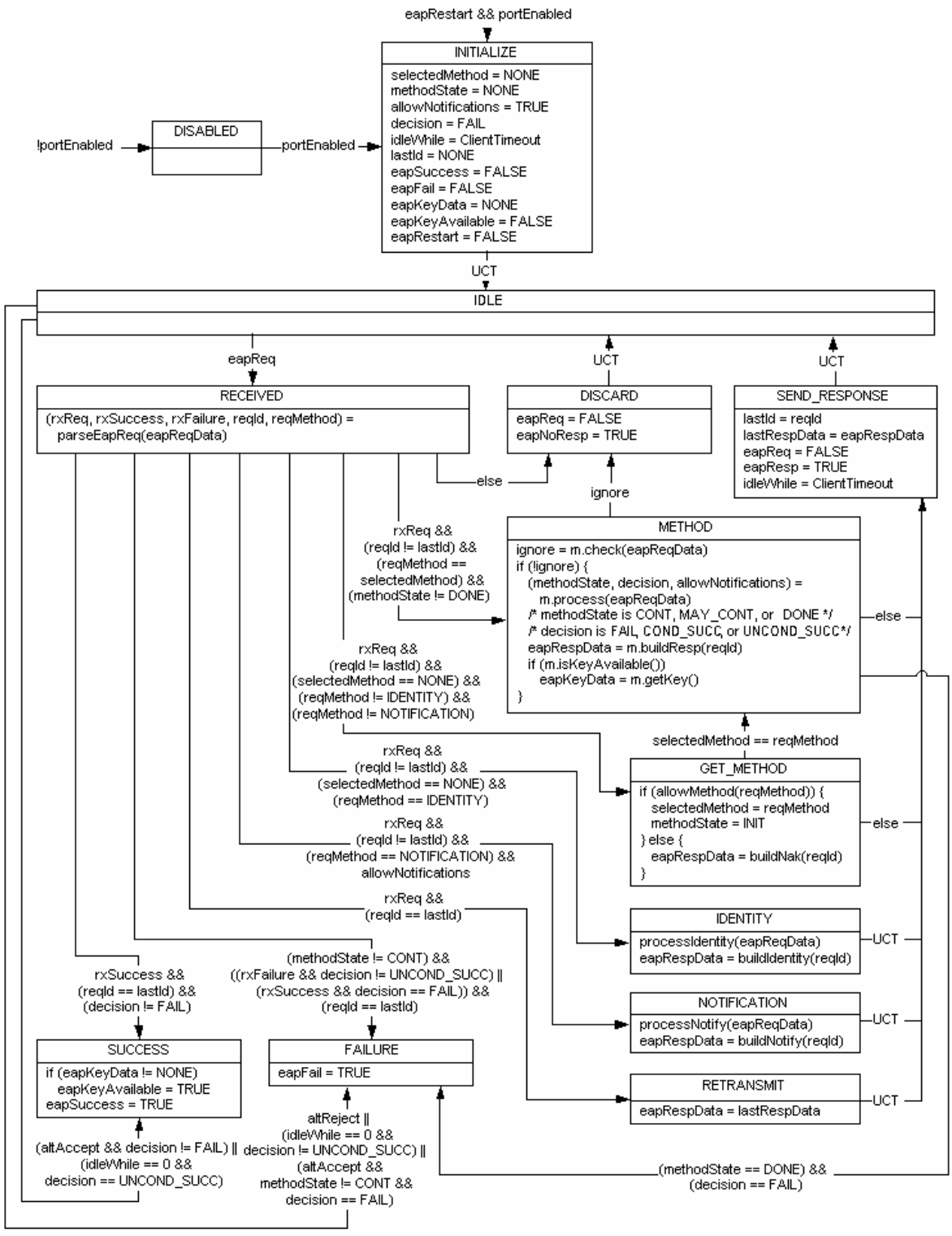
```
#include <iostream>
#include <ace/Log_Msg.h>
#include <ace/OS.h>
#include "diameter_api.h"
#include "diameter_eap_server_session.hxx"
#include "eap.hxx"
#include "eap_peerfsm.hxx"
#include "eap_authfsm.hxx"
#include "eap_identity.hxx"
#include "eap_policy.hxx"
#include "eap_method_registrar.hxx"
#include "eap_fsm.hxx"
#include "eap_log.hxx"
#include "eap_saml.hxx"

[…]
class MyDiameterEapServerSession : public DiameterEapServerSession,
                        public AAA_JobData
{
 public:
  MyDiameterEapServerSession(AAAApplicationCore& appCore,
                     diameter_unsigned32_t appId=EapApplicationId)
    : DiameterEapServerSession(appCore, appId),
      handle(EapJobHandle
          (AAA_GroupedJob::Create(appCore.GetTask().Job(),
                           this, "backend"))),
      eap(boost::shared_ptr<MyBackendAuthSwitchStateMachine>
       (new MyBackendAuthSwitchStateMachine
        (*appCore.GetTask().reactor(), handle))),
       //identityMethod(EapContinuedPolicyElement(EapType(1))),
       samlMethod(EapContinuedPolicyElement(EapType(12))),
      initial(true) //,userNameValidate(true)
  {

    //This is needed in case EAP-TLS should be performed before EAP-SAML.
    //EAP-TLS is needed to deliver the artifact during the very first
    //authentication method
    eap->Policy().InitialPolicyElement(&tlsMethod);
    identityMethod.AddContinuedPolicyElement
    (&samlMethod, EapContinuedPolicyElement::PolicyOnSuccess);

    eap->Policy().InitialPolicyElement(&samlMethod);


    // Allows the StPickupInit-State of the FSM
    eap->NeedInitialRequestToSend(false);

    this->Start();
  }

  void Start() throw(AAA_Error)
  {
    DiameterEapServerSession::Start();
  }
```

```
  /// This virtual function is called when an EAP server session is
  /// aborted due to enqueue failure of a job or an event inside
  /// Diametger EAP server state machine.
  void Abort()
  {
    std::cout << "Diameter EAP server session aborted." << std::endl;
    DiameterEapServerSession::Stop();
    Eap().Stop();
  }

  MyBackendAuthSwitchStateMachine& Eap() { return *eap; }


 private:
  EapJobHandle handle;
  boost::shared_ptr<MyBackendAuthSwitchStateMachine> eap;
  EapContinuedPolicyElement samlMethod;
  EapContinuedPolicyElement tlsMethod;
  bool initial;
};
[…]
```

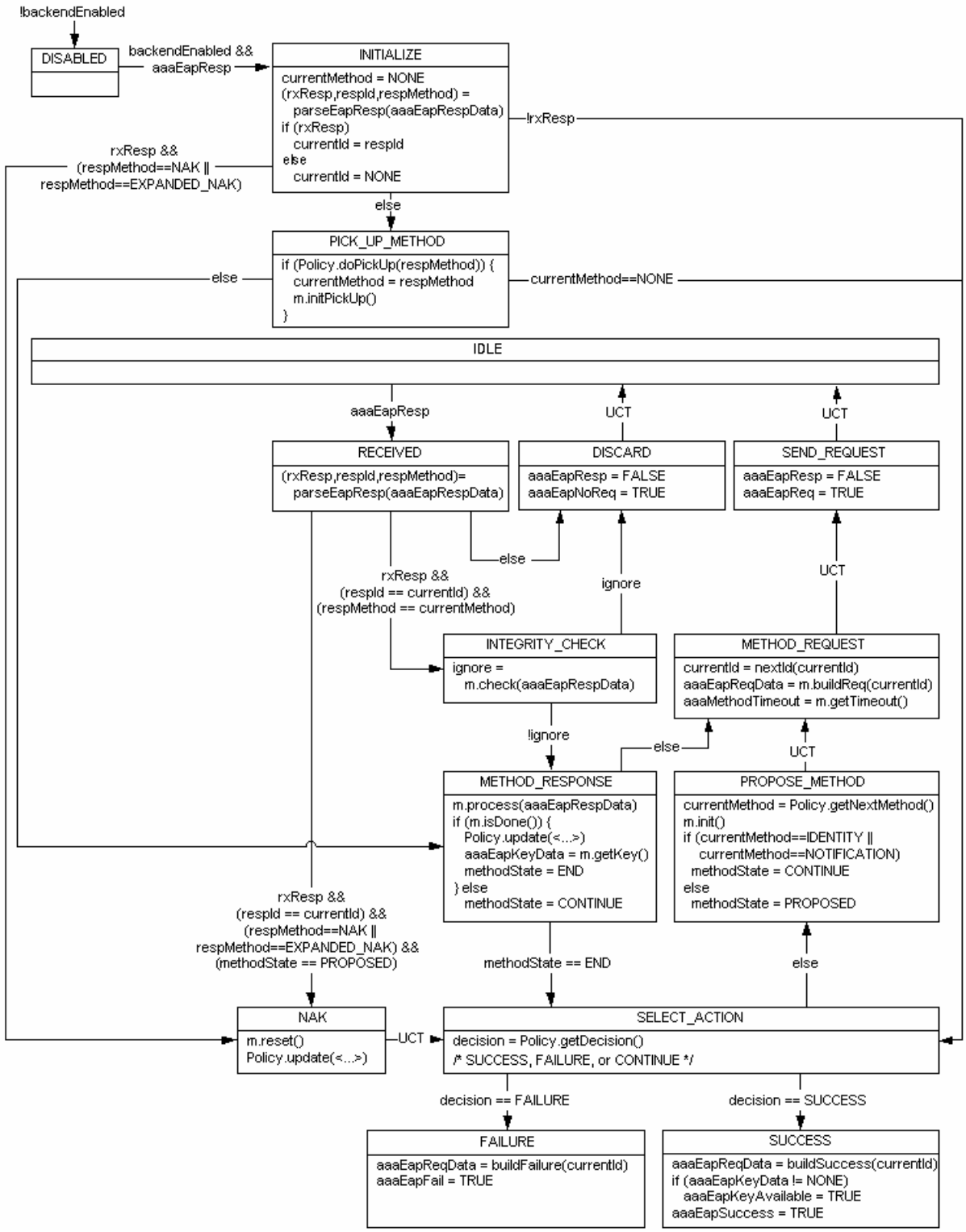
A.2 Zustandsdiagramme

Die folgenden Zustandsdiagramme beschreiben die Zustandsmaschinen eines EAP Peer und eines EAP Authentication Server. Die Zustandsmaschine der EAP-SAML Methode läuft eingebettet in der EAP Zustandsmaschine ab.

A.2.1 Zustandsdiagramm des EAP Peers [43]



A.2.2 Zustandsdiagramm des EAP Backend Authentication Servers [43]



Literaturverzeichnis

[1] BMW Assist, http://www.bmw-telematik.de/bmw_assist.html.

[2] BMW Online, http://www.bmw-telematik.de/bmw_online.html.

[3] Global System for Mobile Communications, http://www.gsmworld.com/technology/gsm.shtml.

[4] GSM Association, http://www.gsmworld.com/index.shtml.

[5] General Packet Radio Service, http://www.gsmworld.com/technology/gprs/index.shtml.

[6] Enhanced Data rates for GSM Evolution, http://www.gsmworld.com/technology/edge/index.shtml.

[7] Universal Mobile Telecommunications System, http://www.3gpp.org/ftp/Specs/html-info/22100.htm.

[8] Wireless World Research Forum, http://www.wireless-world-research.org/.

[9] Wireless World Research Forum: “Book of Visions”, http://www.wireless-world-research.org/general_information/Bookofvisions/BoV1.0/BoV/BoV2001v1.1B.pdf, 2001.

[10] Misra, A., Watson T., Dutta A., McAuley-Sajal, A.: “IDMP-Based Fast Handoffs and Paging un IP-Based 4G Mobile Networks”, IEEE Communication Magazine, Seiten 138-148, 2002.

[11] L. Blunk, J. Vollbrecht, , J. Carlson, H. Levkowetz "Extensible Authentication Protocol (EAP)", IETF RFC 3748, http://www.ietf.org/rfc/rfc2284.txt, 2004.

[12] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: “Diameter Base Protocol”, http://www.ietf.org/rfc/rfc3588.txt, 2003.

[13] Forsberg, D., Ohba, D., Patil, B., Tschofenig, H., Yegin, A.: “Protocol for carrying Authentication for Network Access”, http://www.ietf.org/internet-drafts/draft-ietf-pana-pana-08.txt, 2005.

[14] Security Assertion Markup Language, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#samlv20.

[15] Universität Dortmund, Hochschulrechenzentrum: “I&K Glossar”, http://www.hrz.uni-dortmund.de/docs/Glossar.html, 2005.

[16] University of Colorado at Boulder: “Authentication services”, http://spot.colorado.edu/~kjk/i2/core/Authentication.htm, 2005.

[17] Bundesamt für Sicherheit in der Informationstechnik: “Grundsätzliche Funktionsweise biometrischer Verfahren”, http://www.bsi.de/fachthem/biometrie/einfuehrung.htm, 2005.

[18] The Internet Engineering Task Force, http://www.ietf.org/.

[19] Nash A., Duane W., Joseph C., Brink D.: “PKI – E-Security implementieren”, mitp-Verlag, Bonn, 2002.

[20] Housley, R., Ford, W., Polk, W., Solo, D.: “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, http://www.ietf.org/rfc/rfc2459.txt, 1999.

[21] IEEE 802.11i Protocol: http://standards.ieee.org/getieee802/download/802.11i-2004.pdf, 2004.

[22] De Laat, C., Gross, G., Gommans, L., Vollbrecht, J., Spence, D.: “Generic AAA Architecture”, http://www.ietf.org/rfc/rfc2903.txt, 2000.

[23] Rigney, C., Willens, S., Rubens, A., Simpson, W.: “Remote Authentication Dial In User Service”, http://www.ietf.org/rfc/rfc2865.txt, 2000.

[24] Diameter Framework, http://www.diameter.org/.

[25] Calhoun, P., Farrell, S., Bulley, W.: “Diameter CMS Security Application”, http://www.diameter.org/drafts/latest/draft-ietf-aaa-diameter-cms-sec-04.txt, 2002.

[26] Calhoun, P., Zorn, G., Spence, D., Mitton, D.: “Diameter Network Access Server Application”, http://www.diameter.org/drafts/latest/draft-ietf-aaa-diameter-nasreq-11.txt, 2003.

[27] Eronen, P., Hiller, T., Zorn, G.: “Diameter Extensible Authentication Protocol Application”, http://www.ietf.org/internet-drafts/draft-ietf-aaa-eap-10.txt, 2004.

[28] Postel, J.: “User Datagram Protocol”, http://www.ietf.org/rfc/rfc0768.txt, 1980.

[29] Stewart, R., Xie, Q., Morneault, K, Sharp, C., Schwarzbauer, H., Zhang, L.: “Stream Control Transmission Protocol”, http://www.ietf.org/rfc/rfc2960.txt, 2000.

[30] IP Security Protocol, http://de.wikipedia.org/wiki/IPsec.

[31] Dierks, T., Allen, C.: “Transport Layer Security”, http://www.ietf.org/rfc/rfc2246.txt, 1999.

[32] Simpson, W.: Point-to-Point Protocol, http://www.ietf.org/rfc/rfc1661.txt, 1994.

[33] IEEE 802.11 http://standards.ieee.org/getieee802/802.11.html.

[34] Open Diameter, http://www.opendiameter.org/.

[35] Open Group: “Single Sign-On”, http://www.opengroup.org/security/sso/.

[36] Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/.

[37] Hughes, J., Maler, E.: “Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1”, http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf, 2004.

[38] Simple Object Access Protocol, http://www.w3.org/TR/SOAP/.

[39] XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/.

[40] XML Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/.

[41] Schmidt, D.: “Adaptive Communication Environment”, http://www.cs.wustl.edu/~schmidt/ACE.html, 2004.

[42] SourceForge.net, http://sourceforge.net/index.php.

[43] Open Diameter Documentation Page, http://diameter.sourceforge.net/.

[44] GNU General Public License, http://www.gnu.de/gpl-ger.html.

[45] Xerces C++ Parser, http://xml.apache.org/xerces-c/.

[46] OpenSAML 1.0.1 - an Open Source Security Assertion Markup Language implementation, http://www.opensaml.org/.

[47] Internet2 Consortium, http://www.internet2.org/.

[48] Gamma, E., Helms, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Verlag, 1995.

[49] Extensible Access Control Markup Language, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[50] Weyl, B., Gómez Skarmeta, A., Marin Lopez, R., Mishra, P., Hauser, C., Ziemek, H.: “Protecting Privacy of Identities in Federated Operator Environments” wird veröffentlicht in: Proceedings of IST Mobile & Wireless Communications Summit, Dresden, 2005.

[51] Weyl, B.: "Single Sign-On and Web Services Security for an Open Telematics Market", Technische Universität München, 2003.

[52] Weyl, B., Vögel, H.-J., Michel, H.-U.: “Integrated Authentication for Telematic Services and Beyond-3G Access Infrastructures using SAML”, in: Proceedings of IST Mobile & Wireless Communications Summit, , Lyon, Seiten 212-217, 2004.

[53] Vögel, H.-J.: "Aspects of Personalization and Security in an Open Telematics Services Market", in: Proceedings of ITS 2003, Intelligent Transport Systems and Services, Madrid, 2003.

[54] Weyl, B., Vögel, H.-J.: “SAML and XACML: XML-Technologies for Security and Service Authorization in beyond 3G Personalized Communication”, 2004.

[55] Weyl, B., Vögel, H.-J.: “SAML-based Network and Service Access Control in Beyond 3G Infrastructures”, BMW Internal Work Report, 2004.

[56] DAIDALOS Project, http://www.ist-daidalos.org/.

[57] DAIDALOS Deliverable 341:“A4C Framework Design Specification”, 2004.

[58] Eckert, C.: “IT-Sicherheit - Konzepte, Verfahren, Protokolle” Oldenbourg Verlag, München, 2003.