Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik
Lehrstuhl für Systemarchitektur

Betreuer: Dr. Wolf Müller
Gutachter: Prof. Dr. Jens-Peter Redlich
Prof. Dr. Ernst-Günter Giessmann

Diplomarbeit

Benutzerkonten-Verwaltung mit dem neuen Personalausweis

Mathias Jeschke

Berlin, den 1. Juni 2011

Mathias Jeschke
Bahnhofstraße 31
D-15378 Hennickendorf
E-Mail: mj@majes.de

Digitale Ausgabe: http://mj.majes.de/diplomarbeit.pdf
Vorliegende Revision: 161
Satz: Autorensatz mit LaTeXund dem KOMA-Script-Paket

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Heutzutage werden Geschäftsprozesse und Dienstleistungen zunehmend über das Internet abgewickelt. Dabei spielt die Identifizierung unbekannter Geschäftspartner und Benutzer sowie die Authentifizierung bzw. Wiedererkennung bekannter Geschäftspartner und Benutzer eine wichtige Rolle.

Mit dem neuen Personalausweis (nPA) gibt es erstmals in Deutschland ein elektronisches Identitätsdokument, dass zum einen einer breiten Anwenderschaft zur Verfügung steht[1] und zum anderen durch eine hoheitliche und damit sehr vertrauenswürdige Institution – der Bundesrepublik Deutschland – ausgestellt wird. Mit der eID-Funktion des nPA wird somit einerseits die Identifizierung im elektronischen Handel (E-Commerce) und elektronischen Verwaltung (E-Governement) ermöglicht und andererseits die eindeutige Wiedererkennung eines Ausweisinhabers gewährleistet.

Bei der Nutzung von Benutzername und Passwort kommt es häufig vor, dass ein Benutzer das Passwort (oder sogar den Benutzernamen) vergisst. Dies gilt umso mehr, wenn ein bestimmter Dienst selten genutzt wird. Das führt bei der Administration des Diensteanbieters zu enormen Supportaufwänden, um Benutzern neue Zugangsdaten zuzuweisen.

In dieser Arbeit wird ein System vorgestellt, das die Verwaltung von herkömmlichen Benutzerkonten (engl. Accounts) mit Benutzername und Passwort und dem neuen Personalausweis so verknüpft, dass der Ausweisinhaber bei Vergessen der Zugangsdaten eigenständig neue Zugangsdaten erstellen kann. Weiterhin wird untersucht, in welchem Umfang die autonome Erzeugung von neuen Accounts möglich ist.

1.2. Zielstellung

Die vorliegende Arbeit soll aufzeigen, inwiefern der neue Personalausweis die Verwaltung von Benutzerkonten bei einer Organisation unterstützen kann. Es ist zu prüfen, welche Möglichkeiten der Anbindung an einen eID-Service bestehen, um die Daten und Funktionen von Testausweisen im Rahmen des offenen Anwendungstests des neuen Personalausweises nutzen zu können.

In einem Software-Prototyp sollen typische Anwendungsfälle realisiert und demonstriert werden. Dabei spielen Fragen nach der Sicherheit des zu entwerfenden Benutzerverwaltungsdienstes eine wichtige Rolle.

[1] Den bisherigen „alten“ Personalausweis besitzen mehr als 60 Mio. Bundesbürger, vgl.: http://bmi.bund.de/SharedDocs/FAQs/DE/Themen/Sicherheit/epersonalausweis_faq_chip.html

1.3. Aufbau der Arbeit

In Kapitel 2 werden mögliche Authentifikationsverfahren vorgestellt. Es wird dargestellt, welche Probleme im Zusammenhang mit Passwortverfahren auftreten und welche Lösungen existieren. Auf alternative Verfahren wird hingewiesen.

Kapitel 3 widmet sich dem neuen Personalausweis und untersucht die damit möglichen eID-Szenarien zur Nutzung des Ausweises.

Darauf aufbauend wird in Kapitel 4 ein Dienst zur Benutzerverwaltung modelliert, wie er hauptsächlich im universitären Umfeld, aber auch in großen Organisationen zum Einsatz kommen könnte.

Schließlich gibt Kapitel 5 einen Überblick über die praktische Realisierung eines Prototyps und skizziert die Struktur des Dienstes aus Softwaresicht.

Im letzten Kapitel wird die Arbeit durch eine Zusammenfassung und einen Ausblick abgeschlossen.

2. Authentifikationsverfahren

Trotz enormer Fortschritte auf dem Gebiet der IT-Sicherheit und den damit verfügbaren, sehr sicheren Verfahren ist das Passwort in vielen Fällen auch heutzutage immer noch *der* Schlüssel zu virtuellen Welten, wie z. B. dem Internet und den dort angebotenen elektronischen Diensten.

In diesem Kapitel werden zunächst existierende Authentifikationsverfahren vorgestellt, insbesondere solche, die auf einem Passwort basieren. Anschließend werden Möglichkeiten erläutert, wie ein Passwort zurückgesetzt werden kann.

2.1. Begriffe

Zur Betrachtung und Abgrenzung verschiedener Authentifikationsverfahren ist es notwendig, zunächst einige Begriffe einzuführen, da diese in der Literatur nicht durchgängig in gleicher Weise verwendet werden.

2.1.1. IT-System

Ein IT-System (im Weiteren einfach: „System“) ist eine technische Einrichtung, die für ein Subjekt Daten speichert und verarbeitet. Im Fokus dieses Kapitels stehen dabei (Teil-)Systeme die sicherstellen, dass ausschließlich berechtigte, also authentifizierte Subjekte eine Datenverarbeitung veranlassen können.

2.1.2. Schutzziele

In einem System gewährleistet die **Vertraulichkeit**, dass keine unautorisierte Informationsgewinnung möglich ist, d. h., es ist sichergestellt, dass sensible Daten und Nachrichten unkenntlich gemacht bzw. verborgen werden. In der Praxis wird dies meist durch Verschlüsselung der Daten erreicht, vgl. [Eck08b], S. 8.

Die **Integrität** (auch Unverfälschtheit genannt) garantiert, dass eine unzulässige Veränderung von Daten erkannt werden kann.

Die Überprüfbarkeit der Echtheit und Glaubwürdigkeit von Subjekten und Objekten anhand von charakteristischen Eigenschaften eines Systems wird als **Authentizität** (des Subjekts bzw. Objekts) bezeichnet. Konkret erfolgt diese Überprüfung bei Subjekten (und deren eindeutiger Identität) durch die *Authentifikation*. Die Authentizität von Objekten dagegen wird durch einen Urheber- bzw. Ursprungsnachweis gewährleistet, was vorrangig bei verteilten Systemen notwendig ist und mit kryptographischen Verfahren umgesetzt wird ([Eck08b], S. 7). Im Vergleich zur Authentizität von Objekten (in Form von Nachrichten) ermöglicht die **Verbindlichkeit**, dass nicht nur der eigentliche Empfänger, sondern auch Dritte die Unverfälschtheit und Urheberschaft feststellen können ([BNS05], S. 2).

Neben den drei „klassischen“ Schutzzielen Vertraulichkeit, Integrität und Authentizität, die im Allgemeinen mit sicheren Systemen assoziiert werden, spielen oftmals noch weitere Ziele eine nicht zu unterschätzende Rolle. Dies gilt insbesondere hinsichtlich des zunehmenden Einsatzes von Online-Diensten. Die **Verfügbarkeit** gibt an, mit welcher Wahrscheinlichkeit ein Dienst für die Benutzer erreichbar ist. Beispielsweise darf ein Dienst mit einer Verfügbarkeit von 99% im Jahr für maximal 87,6 Stunden nicht erreichbar sein.

Anonymität und Pseudonymität geben an, ob sich eine Identität gar nicht bzw. nur eingeschränkt zurückverfolgen lässt. Diese Schutzziele sind meist direkt aus Gesetzen und Bestimmungen, wie dem Datenschutz, abgeleitet.

In konkreten Systemen werden einzelne Schutzziele oftmals geeignet kombiniert, um die spezifischen Schwächen zu verringern und ihre Stärken zu vereinen. So hilft eine vertrauliche Kommunikation durch Verschlüsselung der Daten nur gegen *passive Angreifer*. Aktive Angreifer dagegen können mittels eines *Man-in-the-Middle-Angriffs* (MITM) durch separate Kanäle eine vertrauliche (weil verschlüsselte) Kommunikation vorgaukeln. Daher wird häufig eine gegenseitige Authentifikation beider Kommunikationspartner angestrebt.[1]

2.1.3. Digitale Identität

Für das Verständnis, wie Authentifikationsverfahren funktionieren, ist der Begriff der Digitalen Identität von zentraler Bedeutung. Eine Digitale Identität ist eine Menge von Daten, die ein Subjekt (oder Entität) abstrakt beschreiben. Die Herausforderung dabei liegt in der Auswahl der Identifikatoren – jenen Daten bzw. Attributen, die eine Identität von anderen *eindeutig* unterscheidbar machen.[2] Subjekte repräsentieren wiederum Akteure der realen Welt, also Personen, Computer, Prozesse, usw.

Durch biometrische Verfahren lassen sich Entitäten eindeutig (innerhalb bestimmter Toleranzen) bestimmen, die Unterscheidungsmerkmale werden Entifikatoren genannt. Der Unterschied von Identifikatoren und Entifikatoren wird durch die Betrachtung von Anonymen bzw. Pseudonymen deutlich: Entifikatoren ermöglichen, bis auf Fehlerfälle, immer eine Zuordnung zu einer Entität; aus Identifikatoren lässt sich eine Entität gar nicht oder nur mittelbar ableiten. Ein technisches Beispiel zeigt, dass dies nicht nur für Personen, sondern für alle Entitäten gilt: Ein moderner Computer besitzt in der Regel eine Ethernet-Netzwerkkarte mit einer MAC-Adresse, die diesen eindeutig von anderen unterscheidet. Zusätzlich ist einem Computer für die Internet-Kommunikation mindestens eine IP-Adresse zugewiesen (Identifikator). Im Gegensatz zur MAC-Adresse lässt sich eine IP-Adresse – entsprechend der Umgebung oder des Zwecks – ohne Weiteres austauschen, z. B. beim Wechseln von einem Netzwerk in ein anderes.[3]

[1] Natürlich scheitert dieser MITM-Angriff, wenn beide Kommunikationspartner Schlüssel über einen weiteren, unabhängigen (und authentifizierten) Kanal austauschen können. Heutzutage kommen aber oftmals Protokolle zur Schlüsselvereinbarung bzw. Schlüsselaustausch zum Einsatz, die nur gegen passive Angreifer schützen. Ein Vertreter dieser Protokolle heißt Diffie-Hellman-Schlüsselvereinbarung, vgl. [BNS05], Kapitel 19.

[2] In [Win05], S. 9 werden diese Daten nach Attributen, Vorlieben (engl. preferences) und Merkmalen (engl. traits) unterschieden. Merkmale sind in diesem Kontext als Identifikatoren besonders geeignet.

[3] Dieses Beispiel zeigt auch deutlich: Ein Entifikator allein kann unter Umständen nicht zur Bestimmung der Entität ausreichen – so lässt sich die MAC-Adresse mancher Computer, ebenso wie der Fingerabdruck einer Person manipulieren.

In IT-Systemen dienen (digitale) Identitäten heutzutage als Grundlage vieler Prozesse. Mit ihnen lässt sich die Zuordnung von Daten, die Abrechnung von genutzten Diensten und die Zugriffskontrolle von Objekten innerhalb des Systems sinnvoll umsetzen.

Digitale Identitäten werden in Form von *Benutzerkonten* Gegenstand weiterer Betrachtungen in dieser Arbeit sein.

Gleichwohl ist es möglich und sogar ausdrücklich erwünscht, dass eine Entität mehrere (digitale) Identitäten besitzen kann: Die Person, die Zugang zum Benutzerkonto „Administrator“ bzw. „root“ hat, soll dieses aus Sicherheitsgründen nur für administrative Tätigkeiten nutzen. Benutzeraufgaben (z. B. das Lesen von E-Mails) sollen mittels eines so genannten nichtprivilegierten Benutzerkontos ausgeübt werden. Hierbei spricht man auch von den *Rollen* einer Person.

Das folgende Zitat von Mario Grobholz[4] beschreibt den Zusammenhang zwischen Identitäten und Personen:

> *„Der Begriff ‚Identität‘ zielt in erster Linie auf die Unterschiede zwischen Personen ab. Aber auch die Gemeinsamkeiten mit anderen werden berücksichtigt. In der Regel wird darunter die einzigartige Kombination von persönlichen und unverwechselbaren Eigenschaften des Individuums verstanden. Dazu gehören unter anderem der Name, das Geschlecht und der Beruf. Dadurch lassen sich Personen voneinander unterscheiden. Identität ist immer eng verknüpft mit Identifizierung, dem Ausdruck der eigenen Persönlichkeit, Gruppenzugehörigkeiten und Selbstpräsentation. Wer weiß, wie stark sich in Zukunft unseres digitales Erscheinungsbild, unsere Online-Reputation, auf unser Leben auswirkt und auch den herkömmlichen Identitätsbegriff obsolet macht bzw. ihn erweitert.“*

2.1.4. Authentifikation

Mit Authentifikation (oder Authentifizierung) wird der *Nachweis* einer Identitätsbehauptung bezeichnet, die Nachweise als *Credentials* oder *Authentifikatoren*. Daneben wird in dieser Arbeit synonym der Begriff der *Authentisierung* benutzt. Die Existenz der beiden Begriffe liegt in den beiden Sichtweisen auf den Prozess begründet: Ein Benutzer kann sich *authentisieren* oder aber durch ein System *authentifiziert* werden. Man unterscheidet zwischen *einseitiger* und *gegenseitiger* bzw. wechselseitiger Authentifikation. Der Begriff *Identifikation* (oder Identifizierung) bezeichnet in diesem Zusammenhang den Vorgang, bei dem ein Subjekt eben jene Identitätsbehauptung aufstellt ([Sch07], S. 338).[5]

Das den Beweis anstrebende Subjekt wird dabei als *Prover*, die prüfende Instanz als *Verifier* bezeichnet. Im Zusammenhang mit der Authentifizierung auf Basis von Dokumenten bzw. Zertifikaten (in der realen Welt auch Ausweise genannt) spielt eine dritte Instanz eine bedeutende Rolle: Der *Issuer* stellt die notwendigen Dokumente aus. Er muss ein besonderes Vertrauensverhältnis zum Verifier haben – beispielsweise durch seine Reputation, Autorität oder aufgrund einer marktbeherrschenden Stellung.[6]

[4] Quelle: http://blog.myonid.de/2007/11/digitale-identitat-i-ein-neues-selbstverstandnis/.

[5] Umgangssprachlich wird mit „Identifizierung“ oftmals eigentlich eine Authentifikation gemeint, z. B. beim Nachweis der eigenen Identität mittels eines Ausweises. In diesem Zusammenhang ist deshalb der eher formale Begriff „Identitätsfeststellung“ zutreffender. Der Ausweis ist dabei ein Authentifikator, der auf den Faktoren *Besitz* (das physische „Token“) und *Biometrie* (z. B. durch ein Passbild) beruht.

[6] So legen Browserhersteller fest, welche Root-CAs in der Standardeinstellung als vertrauenswürdig gelten.

In einem (Ausweis-)Dokument werden die charakteristischen Identitätsattribute des Provers, welche demzufolge vom Issuer sorgfältig zu prüfen sind, an eine Menge von Authentifikatoren (z. B. ein Lichtbild, Fingerabdrücke oder einen öffentlichen Schlüssel) gebunden. Wichtig ist dabei, dass der Verifier in der Lage sein muss, die Authentizität des Dokuments zu prüfen. In Ausweisdokumenten geschieht das durch Sicherheitsmerkmale; in Zertifikaten durch eine digitale Signatur, vgl. [Mü10], S. 46ff.

Die Art und Weise zur Erbringung des Identitätsnachweises, wird durch Authentifikationstechniken charakterisiert. Diese beruhen auf Faktoren dieser drei Kategorien:

- Wissen (etwas, das jemand weiß),
- Besitz (etwas, das jemand besitzt),
- Biometrie (etwas, das jemand ist).

Zur Umsetzung eines Authentifikationsverfahrens werden oftmals mehrere Techniken eingesetzt, um die einzelnen spezifischen Vorteile zu kombinieren. So wird beim Verfahren „Authentifikation mittels Smartcard“ zum einen ausgenutzt, dass sich eine Smartcard mit den darauf gespeicherten Daten gar nicht oder wenn dann nur sehr schwer kopieren lässt. Dadurch wird sichergestellt, dass nur derjenige die Identität nachweisen kann, der im Besitz der dazu gehörenden Smartcard ist. Zum anderen wird durch geheimes Wissen, welches nur der Inhaber der Smartcard haben sollte (die Personal Identification Number - PIN), ein Missbrauch durch Entwenden der Karte erschwert.

Anforderungen an Authentifikationsverfahren

Bei der Bewertung von Authentifikationsverfahren helfen folgende Anforderungen (vgl. [BNS05], S. 217f.):

- *„Durchführbarkeit: Ein berechtigter Prover muss einem Verifier gegenüber in der Lage sein, seine Identität nachzuweisen.*
- *Keine Impersonation: Kein Teilnehmer eines Systems darf sich als ein anderer ausgeben können.*
- *Unübertragbarkeit: Ein Verifier darf nicht in der Lage sein, sich als ein anderer Teilnehmer auszugeben.“*

2.1.5. Autorisierung

Ein legitimer Benutzer wird nach der Authentifikation bestimmte Dienste nutzen wollen. Die Autorisierung beantwortet die Frage: „Ist der konkrete Benutzer dazu berechtigt?“
Die Umsetzung erfolgt meist durch ein Regelwerk, der *Access Control List* (ACL).

2.2. Authentifikationsverfahren – Ein Überblick

2.2.1. Wissenbasierte Verfahren

Diese Klasse von Verfahren basiert auf einem Wissen, das zwischen dem Benutzer und dem Systembetreiber *vereinbart* wurde. Es wird dabei einerseits unterschieden, ob bei der

Authentifikation das Wissen als solches übertragen wird (und dabei eventuell von einem Angreifer abgefangen und dann für eigene Authentifikationen genutzt werden kann), und andererseits, ob es notwendig ist, das Wissen auf Seiten des Systems (im „Klartext“) abzuspeichern, oder es ausreicht, Ableitungen davon vorzuhalten. Wichtige Vertreter sind:

- passwortbasierte Verfahren,
- Challenge-Response-Verfahren,
- Zero-Knowledge-Verfahren.

Aufgrund der hohen Verbreitung in der Praxis und den damit begründeten Thesen dieser Arbeit liegt im Folgenden der Schwerpunkt bei diesen Verfahren.

2.2.2. Besitzbasierte Verfahren

Diese Verfahren zeichnen sich dadurch aus, dass der Prover etwas besitzt, zu dem andere Subjekte keinen Zugriff haben. Wichtige Vertreter sind:

- cookiebasierte Verfahren (in Webanwendungen),
- schlüssel- bzw. zertfikatbasierte Verfahren,
- tokenbasierte Verfahren (z. B. Dokumente, Ausweise, Smartcards).

Aufgrund der losen Bindung eines Gegenstands an den Eigentümer werden besitzbasierte häufig mit wissenbasierten Verfahren kombiniert, d. h., das Objekt wird durch eine zusätzliche Wissenskomponente an den Eigentümer gebunden und damit das Objekt vor Fremdnutzung geschützt.

Bewertungskriterien besitzbasierter Verfahren sind insbesondere:

- die Kopierbarkeit,
- die Verfügbarkeit,
- die Interoperabilität

des betrachteten Authentifikationsobjekts.

Beispielsweise hat die Verwendung von Bankkarten mit (kopierbarem) Magnetstreifen gezeigt, dass diese Technologie trotz eines weiteren Authentifikations-Faktors (der PIN) eine Kompromittierung der Identität des Bankkunden durch Skimming-Angriffe begünstigt. In der Folge werden, zumindest in Deutschland, vermehrt Bankkarten mit Chip ausgegeben, es handelt sich somit um (nicht kopierbare) Smartcards.[7]

Die Verfügbarkeit und die Interoperabilität eines Objekts spielen vor allem in mobilen Szenarien eine wichtige Rolle: Im Gegensatz zu einem Passwort, welches der Prover immer bei sich „trägt“, muss dieser bei besitzbasierten Verfahren daran denken, das notwendige

[7] Nichtsdestotrotz verfügen solche Bankkarten – aus Kompatibilitätsgründen – in der Regel noch über einen Magnetstreifen, der sich natürlich trotzdem kopieren lässt. Daher fordern Strafverfolger die Abschaffung des Magnetstreifens: [BKA11].

Objekt mitzuführen und – sofern er nicht eigene Geräte benutzt – darauf hoffen, dass Hilfseinrichtungen, wie z. B. ein Kartenlesegerät und dazugehörige Software, überall verfügbar sind. Vor allem auf Smartphones (Mobiltelefone mit erweitertem Funktionsumfang) ist die Interoperabilität und somit die Durchführbarkeit der Authentifikation fraglich, da oftmals diese Hilfseinrichtungen auf PC-Szenarien ausgelegt sind und auf Smartphones nicht funktionieren.

Aus krytographischer Sicht handelt es sich bei Passwörtern (Wissen) und Cookies bzw. symmetrischen Schlüsseln (Besitz) um dasselbe – es ist geteiltes „Wissen". Für die Praxis ist der Unterschied dennoch bedeutsam, da sich der Benutzer letztere nicht merken kann und somit auf Datenträger angewiesen ist, auf denen das Geheimnis gespeichert wird. Diese Datenträger stellen dann den „Besitz" dar.

2.2.3. Biometrische Verfahren

Biometrische Verfahren basieren auf der Erfassung und dem Vergleich bekannter Merkmale einer Entität. Die Authentifikation beruht also auf der (zuverlässigen) Bestimmung einer Entität hinter der zu prüfenden Identität. Voraussetzung dafür ist die Existenz von charakteristischen, unveränderlichen Merkmalen für eine Entität. Typische Merkmale, die benutzt werden, sind:

- ein Lichtbild / Passfoto,
- Fingerabdrücke,
- Eigenschaften der Iris einer Person,
- Charakteristika der Sprache,
- Handschrift,
- Hardwareeigenschaften eines Computers (z. B. Seriennummern, Adressen).

Trotz dieser vielversprechenden Techniken sind einige Probleme offenkundig:

1. Es gibt Entitäten, die nicht alle Merkmale in ausreichender Qualität besitzen.[8] Daraus folgt, dass eventuell die Anforderung an die Durchführbarkeit verletzt ist.
2. Die Unveränderlichkeit eines Merkmals stellt nach dessen Kompromittierung ein Problem dar.[9]
3. Die inhärenten Toleranzen dieser Verfahren sperren entweder legitime Benutzer aus oder begünstigen Angriffe auf das System durch illegitime Benutzer.
4. Es ist keine Pseudonymität oder Anonymität möglich.

[8] Ausreichende Qualität bezieht sich auf die typischen Toleranzen der Erfassungskomponente. So haben beispielsweise manche Personen Fingerabdrücke, die nicht stark genug ausgeprägt sind, um technisch erfasst zu werden.

[9] Der ehemalige Bundesinnenminister Wolfgang Schäuble dürfte nach der Veröffentlichung von Attrappen seiner Fingerabdrücke durch den Chaos Computer Club Probleme haben, entsprechende Authentifikationssysteme zu benutzen. Siehe [CCC08].

2.3. Passwortbasierte Authentifikationsverfahren

Bei einer passwortbasierten Authentifikation geht es darum, den Nachweis einer Identität darauf zurückzuführen, dass das der Prover die Kenntnis eines *gemeinsamen Geheimnisses* glaubhaft macht. Somit handelt es sich um wissenbasierte Verfahren.

Der praktische Ablauf lässt sich gut am Beispiel „Login an einem Computer“ beschreiben: Der Benutzer (Prover) gibt am System (z. B. einer Computer-Anwendung) zunächst seine systemeindeutige Benutzerkennung bzw. seinen Loginnamen ein. Hierbei handelt es sich um die Identitätsbehauptung. Anschließend wird das zugehörige Passwort – der Beweis – abgefragt[10] und die Kombination von Loginname und Passwort durch das System überprüft. Bei Erfolg ist der Benutzer am System angemeldet bzw. eine Transaktion wird ausgeführt.

2.3.1. Prinzipielle Probleme und Gegenmaßnahmen

Kopierbarkeit

Das inhärente Problem von Passwörtern ist deren Kopierbarkeit. So kann ein Angreifer von jedem Ort aus (an dem ein Zugang zum System möglich ist) sich unter der Identität des Opfers ausgeben – wenn er nur das Passwort kennt.[11]

Der Begriff „Identitätsdiebstahl“ der damit zusammen benutzt wird, ist problematisch, wenn nicht sogar irreführend, weil durch den Begriff „Diebstahl“ ein Verlust des Passworts bzw. der Identität assoziiert wird. Da der legitime Benutzer jedoch weiterhin im Besitz des Passworts ist, wird durch diesen Begriff eine „trügerische Sicherheit“ vermittelt.

Allgemein gilt: Durch häufige Nutzung eines Passworts nimmt die Wahrscheinlichkeit für dessen Kompromittierung zu. Daher sind in vielen Authentifikationssystemen Verfahren implementiert, die Richtlinien zum regelmäßigen Ändern eines Passworts überwachen. Umstritten dabei ist allerdings, in welchen Intervallen ein Passwortwechsel sinnvoll ist, da mit der Häufigkeit von notwendigen Passwortänderungen die Akzeptanz des Benutzers für diesen Prozess, aber auch die Qualität der benutzten Passwörter abnimmt.

Erratbarkeit

Ein großes Problem stellt die Wahl des Passworts dar. Wie bei allen kryptographischen Systemen ist es prinzipiell möglich, den Schlüsselraum – also alle Kombinationen theoretisch gültiger Passwörter – durchzuprobieren. Beim naiven Ansatz werden alle Zeichen des zugrunde liegenden Passwort-Alphabets (in der Regel ASCII) ausgehend von Passwörtern mit minimaler bis maximaler Länge getestet. Diese Klasse von Angriffen wird unter dem Begriff *Brute-Force-Attacke* (engl.: „rohe Gewalt“) zusammengefasst. Man unterscheidet hierbei zwischen Online- und Offline-Angriffen. Bei Online-Angriffen besteht aus Sicht des Passwortverfikationssystems die Möglichkeit, Einfluss auf die Menge bzw. den Durchsatz (in Versuchen pro Zeiteinheit) der Passwortprüfungen zu nehmen, da die Prüfung auf

[10] In vielen Anwendungen gibt der Benutzer auch beides zusammen ein und veranlasst die Passwortprüfung durch das System. Dies ist vor allem bei verteilten Systemen sinnvoll, da die Anzahl der Kommunikationsschritte reduziert wird.

[11] Genau genommen muss auch der Loginname bekannt sein. Dieser ist aber oftmals in (frei zugänglichen) Benutzerverzeichnissen einsehbar oder aus der E-Mail-Adresse des „Opfers“ ableitbar.

einer vom Angreifer unkontrollierten Komponente stattfindet.[12] Offline-Angriffe dagegen finden vollständig unter der Kontrolle des Angreifers statt und sind somit nur durch die verfügbare Rechenleistung limitiert. Dabei werden beispielsweise Transformationen von Passwörtern gegen eine Liste bekannter Transformationen vergleichen. Die Anwendung einer krytographischen Hashfunktion auf das Klartextpasswort stellt eine solche Transformationsvorschrift dar. Es ist zu erwarten, dass diese Form der Angriffe in Zukunft zunehmen wird, da es mithilfe spezieller Hardware wie bestimmten Grafikkarten (vgl. [Bv10]), aber auch Cloud-Computing-Diensten (vgl. [EC2a] und [EC2b]) immer einfacher wird, ein bestimmtes transformiertes Passwort „zu knacken“, d. h., ein Urbild zu finden. Als Abhilfe bleibt dem Benutzer nur, hinreichend lange Passwörter zu wählen, die einen Angriff unwirtschaftlich machen und mit kostengünstiger Technik zu lange dauern, vgl. [FS09], Abschnitt 2.1.

Um Online-Angriffe abzumildern oder gar zu vermeiden, wird häufig nach einer Anzahl von Fehlversuchen (z. B. nach 3) die nächste Prüfung für eine bestimmte Zeitdauer (Timeout) zurückgewiesen. In manchen Szenarien wird darüber hinaus auch das Benutzerkonto komplett für weitere Versuche gesperrt und muss erst über einen anderen Mechanismus entsperrt werden. Diese Maßnahme ist jedoch genau abzuwägen, da einerseits der Support-Aufwand bei vergessenen Passwörtern steigt und andererseits das Passwortverfikationssystem somit anfällig für Sabotage (Denial-of-Service-Angriffe) wird, vgl. [FS09], Abschnitt 2.2.

Um den Suchraum – unabhängig von der Art des Angriffs – zu verringern, bedienen sich Angreifer oftmals so genannter Wörterbücher und Regeln. Dadurch wird der theoretisch mögliche Suchraum erheblich eingeschränkt und die praktische Suche enorm erleichtert. Eine solche Regel lautet z. B. „Verwende nur Kleinbuchstaben, da viele Benutzer Passwörter mit ausschließlich kleinen Buchstaben verwenden“, oder: „Benutze Wörter, die auf einer QWERTY-Tastatur in einer Reihe liegen.“

Viele solcher Regeln machen es sich zunutze, dass Computer-Benutzer schlechte Passwörter auswählen. In der Folge vergeben viele Betreiber zufällig erzeugte Kennwörter, die nicht vom Benutzer geändert werden können, oder unterwerfen bei Passwortänderungen das neue Passwort strikten Qualitätskriterien. Als Maß für die Qualität eines Passworts wird die Entropie benutzt: *„Die Entropie misst den Informationsgehalt, den man im Durchschnitt erhält, wenn man die Quelle* [Anm. einer Nachricht] *beobachtet. Sie ist damit ein Maß für die Unbestimmtheit der Quelle. Die Entropie erlaubt Aussagen über die Unbestimmtheit der erzeugten Nachrichten, falls die Quelle nicht beobachtet werden kann*“ ([Eck08b], S. 285).

Für den Informationsgehalt I der Zeichen z_i Alphabets $Z = z_1, z_2, \ldots, z_n$ gilt

$$I(z_i) = \log_2\left(\frac{1}{p_i}\right) = -\log_2(p_i) \quad \text{in Bit pro Zeichen,}$$

wobei p_i die Wahrscheinlichkeit des Auftretens von z_i ist. Die Entropie von Z definiert man wie folgt:

$$H(Z) = -\sum_{i=i}^{n} p_i \cdot \log_2(p_i).$$

Wenn man annimmt, dass die Zeichen von Z gleichverteilt sind, gilt $p_i = \frac{1}{n}$ für $i = 1, \ldots, n$

[12] Sollte der Angreifer diese Komponente bereits kontrollieren, ist eine Brute-Force-Attacke nicht mehr nötig.

und somit

$$H(Z) = -\sum_{i=i}^{n} p_i \cdot \log_2(p_i) = -\sum_{i=i}^{n} \frac{1}{n} \cdot \log_2\left(\frac{1}{n}\right) = -\log_2\left(\frac{1}{n}\right) = \log_2(n).$$

(vgl. [Eck08b], Abschnitt 7.4.2).

Beispiel: Ein gleichverteiltes Alphabet bestehend aus Klein- und Großbuchstaben sowie den Ziffern 0 bis 9 (insgesamt 62 Zeichen) entspricht einer Entropie von ca. 6 Bit/Zeichen ($H(Z) = \log_2(62)$). Für ein Passwort, in dem alle Zeichen voneinander stochastisch unabhängig sind, ergibt sich für eine Komplexitätsanforderung von 64 Bit, dass das Passwort mindestens 11 Zeichen lang sein muss.

Verwenden die Benutzer sogar ein Datum als Passwort (dies ist häufig bei sechs- oder achtstelligen Passwörtern der Fall), braucht ein Angreifer nur noch 36500 Möglichkeiten durchzuprobieren, um Erfolg zu haben, wenn man einen Zeitraum von 100 Jahren berücksichtigt (365 Tage · 100), vgl. [FS09], Abschnitt 2.3.

Als Resultat kann man dem Betreiber eines IT-Systems nur anraten, selbst aktuelle Werkzeuge zum Passwortbrechen auf die eigene Benutzerdatenbank anzuwenden und Benutzerkonten mit schwachen Passwörtern nach einer entsprechenden Mitteilung an den Benutzer zu sperren.

Social Engineering

Eine andere Art von Problemen stellen Social-Engineering-Angriffe dar. Diese Angriffe zielen nicht gegen das Authentifikationssystem als solches, sondern es wird versucht, das Passwort direkt vom Benutzer zu erfahren. Die klassische Variante ist die, dass der Angreifer bei einem Benutzer anruft und sich als Administrator oder andere Autorität ausgibt und nach dem Passwort des Benutzers fragt, um sich am System „zu Wartungszwecken“ unter dessen Identität anmelden zu können. Alternativ werden solche *Phishing-Angriffe* per E-Mail von Kriminellen häufig genutzt, um Zugangs- und Transaktionskennwörter (TAN) für Online-Banking-Konten abzugreifen.

Viele Passwörter und Nutzungshäufigkeit

Mit zunehmender Zahl von Identitäten eines Benutzers treten zwei Probleme auf:

1. Es wird das gleiche Passwort für unterschiedliche Identitäten benutzt. Wie bereits erläutert, nimmt in der Folge die Kompromittierungs-Wahrscheinlichkeit zu – schlimmer noch: Ein Angreifer hat bei erfolgreicher Kompromittierung Zugriff auf viele (unterschiedliche) Identitäten. Beispielsweise betreiben Kriminelle „bösartige Dienste“ (engl. malicious service) im Internet oder manipulieren Geldautomaten, um das Passwort bzw. die PIN eines Benutzers zu erlangen.

2. Selbst wenn sich Benutzer ersterem Problem bewusst sind, führt dies dazu, dass Passwörter selten genutzter Identitäten vergessen werden und in der Folge Passwort-Rücksetzverfahren nötig sind.

Deshalb haben nahezu alle Webbrowser-Hersteller Möglichkeiten zur Verknüpfung und Hinterlegung von Benutzername-Passwort-Paaren zu Websites realisiert. Eine Site wird im

Allgemeinen anhand des Rechnernamens in der Adresse (URL) eines Dienstes identifiziert. Um einen unbefugten Zugriff auf diese Credentials zu verhindern, bedienen sich die Webbrowser den Zugriffsmechanismen des Betriebssystems („Dateirechte“) und eines (optionalen) *Master-Passworts*.

Als Alternative existieren Erweiterungen für Webbrowser bzw. herunterladbare Applets, aber auch Standalone-Anwendungen, die ausschließlich mit einem Masterpasswort arbeiten und während einer Sitzung auf Grundlage von Regeln das eigentliche (kryptische) Kontopasswort errechnen. Beispiele sind der „PasswortSitter“[13], „KeePass“[14] und „Password Safe“[15]. Um eine Mehrfachverwendung des berechneten Passworts zu vermeiden, geht in die Berechnung (z. B. durch Anwendung einer Hashfunktion) ein Site-spezifisches Datum mit ein.

Der Vorteil dabei ist, dass die Software auf der Seite des Verifiers nicht angepasst werden muss – das Authentifikationsverfahren ist weiterhin passwortbasiert. Diese Browserfunktionen haben auch einen Nachteil: Sie lassen sich nicht ohne die notwendige Software nutzen oder man ist sogar zur Benutzung des betreffenden Kontos an einen Standort bzw. Computer gebunden oder man muss Synchronisierungssoftware (z. B. Xmarks[16]) benutzen.

In [Eik11] wird ein Verfahren vorgestellt, welches die Ableitung des Zugangs- aus einem Masterpasswort aufgreift, die Berechnung aber dem Benutzer überlässt. Ob die Qualität dieser „generierten“ Passwörter ausreicht, darf bezweifelt werden, da oftmals die Länge des resultierenden Kontopassworts beschränkt und die Site-Kennung nicht als geheim einzustufen ist.

Alle Verfahren, die nur auf einem Masterpasswort (und einer Site-Kennung) beruhen, sind vor die gleiche Herausforderung gestellt: Wie kann man das resultierende Passwort eines einzelnen Kontos ändern, ohne dazu das Masterpasswort ändern zu müssen? Bei einer Änderung des Masterpassworts müssten sonst auch alle anderen Konten simultan geändert werden. Die Nutzung eines zusätzlichen Selektors (z. B. ein Seed-Wert) führt aber leider dazu, dass dieses irgendwo gespeichert werden muss. Praktisch wird das nur auf Benutzerseite möglich sein.[17] Zudem funktioniert die Nutzung zusätzlichen Wissens nur bei Systemen mit Datenbank und schließt manuelle Verfahren wie das in beschriebene [Eik11] aus.

2.4. Alternative Authentifikationsverfahren

2.4.1. Einmalpasswörter

Gegen die Bedrohung durch Kompromittierung von Passwörtern hilft eine (zunächst simpel klingende) Maßnahme: Statt immer dasselbe Passwort zu verwenden, wird bei jeder Authentifikation ein neues, nur für *diese* Authentifikation gültiges Passwort benutzt – ein so genanntes Einmalpasswort. Somit kann sich ein Benutzer auch in potentiell unsicheren

[13] http://www.passwordsitter.de/.
[14] http://keepass.info/.
[15] http://passwordsafe.sourceforge.net/.
[16] http://www.xmarks.com/.
[17] Andererseits müssten *alle* Betreiber von Diensten mit Passwort-Authentifikation ihre Anmeldedialoge anpassen.

Umgebungen, beispielsweise Internetcafes, authentifizieren.[18]

Die Verwendung von Einmalpasswörtern dürfte jedem Online-Banking-Kunden vertraut sein: Zum Abschluss einer Transaktion (z. B. einer Überweisung) ist es nötig, eine Transaktionsnummer (TAN) einzugeben. Die TAN ist ein solches Einmalpasswort. Wie kann ein solches Verfahren umgesetzt werden?

Beim naheliegenden Ansatz vereinbaren Verifier und Prover eine Liste von gültigen Einmalpasswörtern mit optionalem Index (zur Adressierung eines bestimmten Passworts) – analog zur TAN-Liste beim Online-Banking.

Das S/Key-Verfahren hingegen basiert auf einer krytographischen Hashfunktion und einem Geheimnis, das aber nur der Prover kennen muss. Zusätzlich wird ein *Seed* benutzt, um mehrere unterschiedliche Identitäten mit diesem Verfahren bei gleichem Geheimnis sicher authentifizieren zu können. Der Seed entspricht von der Funktion her der Site-Kennung bei den oben beschriebenen Passwort-Managern. Die Sicherheit des Verfahrens beruht auf der Annahme, dass es einfach ist, den Wert einer solchen Hashfunktion zu berechnen, es dagegen aber praktisch unmöglich ist, ein Urbild zu diesem Wert zu finden.

Initialisierung

Vor der eigentlichen Authentifikation (und nach n Authentifikationen) muss eine Initialisierungsphase zwischen Verifier und Prover absolviert werden. Dabei berechnet der Prover mit seinem Geheimnis g und dem Seed s das Ergebnis der n-maligen Anwendung einer Hashfunktion h:

$$p_n = h^{(n)}(g, s) = h(p_{n-1}),$$
$$p_1 = h(g, s).$$

Das Ergebnis p_n muss anschließend über einen sicheren Kanal an den Verifier übertragen werden.

Authentifikation

Um sich gegenüber dem Verifier zu authentifizieren, berechnet der Prover, nachdem er die Werte für k und dem Seed vom Verifier erhalten hat, durch k-malige Anwendung der Hashfunktion den Wert:

$$p_k = h^{(k)}(g, s).$$

Der Wert k entspricht dabei dem Dekrement von k des vorherigen Authentifikationsdurchlaufs. Um das Einmalpasswort p_k zu prüfen, braucht der Verifier nur einmal die (vereinbarte) Hashfunktion auf p_k anzuwenden:

$$p_{k+1} = h(p_k) \stackrel{?}{=} p'_{k+1}.$$

Stimmt das Ergebnis mit dem gespeicherten Wert p'_{k+1} (aus dem vorherigen Authentifikationsdurchlauf) überein, ist das Einmalpasswort korrekt und p_k wird als neuer Referenzwert, $k - 1$ als neuer Index gespeichert.

[18] Dabei ist aber zu beachten, dass Einmalpasswörter in der Tat nur gegen Kompromittierung des Passworts schützen. Wird ein Einmalpasswort in einer unsicheren Umgebung zur Authentifikation einer Sitzung benutzt, besteht weiterhin die Gefahr, dass die gesamte Sitzung „gestohlen“ wird. Es sollten daher nur unkritische Anwendungen in solchen Umgebungen eingesetzt werden.

Angriffe

Durch aktives Eingreifen in die Kommunikation mittels Man-in-the-Middle-Angriff kann ein Angreifer die Weitergabe des Einmalpassworts p_k vom Prover an den Verifier unterdrücken und für eine eigene Authentifikation benutzen. Der legitime Prover würde dies am Abbruch der ursprünglichen Transaktion und dem Scheitern einer erneuten Authentifikation mit p_k merken, da dann p_{k-1} das gültige Passwort wäre.

Sollte der Prover jedoch im Unklaren über das nächste k sein und diese Information dem Authentifikationsprotokoll entnehmen, kann der Angreifer gegenüber dem Prover behaupten, $k-1$ sei der aktuelle Index, woraufhin der Angreifer p_{k-1} erhielte. Durch Anwendung der Hashfunktion auf p_{k-1} kann der Angreifer $p_k = h(p_{k-1})$ errechnen und mit p_k eine Transaktion unter der Identität des Provers ausführen. Anschließend würde er die legitime Transaktion mit p_{k-1} weiterleiten. Das Opfer könnte dies nur durch eine verzögerte Ausführungsdauer der Transaktion bemerken, was bei Best-Effort-Systemen wie Online-Diensten praktisch nicht erkennbar ist.

Als Maßnahme ist daher *vor* der Eingabe des Einmalpassworts zwingend der Verifier zu authentifizieren, um diese Art der Angriffe zu verhindern.

2.4.2. Challenge-Response-Verfahren

Eine Verallgemeinerung des Einmalpasswort-Prinzips stellen Challenge-Response-Verfahren dar.[19] Diese Verfahren ermöglichen den Nachweis eines Wissens, ohne dieses dabei offenlegen zu müssen – und sind somit auch zur passwortbasierten Authentifikation geeignet. Wie es der Name andeutet, erfolgt die Authentifikation durch den Austausch einer Folge von Frage-Antwort-Paaren. Formal bedeutet dies, dass sowohl der Prover – nennen wir ihn „Alice“ – als auch der Verfier – nennen wir ihn „Bob“ – eine Funktion, basierend auf dem Geheimnis s_A bzw. s_B, auf einen Eingabewert N (die *Nonce*) anwenden. Der Wert von N, den Bob bestimmt hat, kann problemlos über einen unsicheren Kanal zu Alice übertragen werden. Den zu berechnenden Funktionswert bezeichnen wir mit r_A bzw. r_B.

Nach der Berechnung von r_A überträgt Alice das Ergebnis zu Bob, der r_A mit dem Ergebnis seiner Berechnung r_B vergleicht:

$$f(s_A, N) = r_A \stackrel{?}{=} r_B = f(s_B, N).$$

Sofern Alice nicht die Möglichkeit hatte, den Wert von r_B zu erraten, ist sichergestellt, dass Alice im Besitz von $s_A = s_B$ ist.

Angriffe

Eine Bedrohung – unabhängig von der Wahl der Funktion f – sind Replay-Angriffe. Dabei merkt sich der Angreifer sowohl die Challenges von Bob als auch die (korrekten) Responses von Alice. Wählt nun Bob bei einer späteren Authentifikation die gleiche Challenge, die er zuvor schon einmal an Alice gesendet hat, kann der Angreifer sofort die Response liefern. Es ist jene, die Alice im früheren Durchlauf berechnet hatte. Abhilfe schafft in

[19] Bei Einmalpasswörtern haben der Seed und die Iterationszahl die Funktion der Challenge, das Einmalpasswort die Funktion der Response.

diesem Fall z. B. die zusätzliche Verwendung einer monoton steigenden Sequenznummer (beispielsweise die Systemzeit) bei der Bestimmung der Challenge.

Sofern die Übertragung von Challenge und Response über einen unverschlüsselten Kanal erfolgt und für f eine Verschlüsselungsfunktion benutzt wird, kann ein Angreifer prinzipiell einen Known-Plaintext-Angriff durchführen, um den Schlüssel zu ermitteln, mit dem die Response verschlüsselt wurde.

Bei der Verwendung von Challenge-Response-Verfahren zur Authentifikation eines Kommunikationspartners über einen verschlüsselten Kanal (z. B. mit Diffie-Hellman-Schlüsselvereinbarung) ist die Bedrohung durch Man-in-the-Middle-Angriffe zu berücksichtigen. Zur Abwehr ist die Berechnung der Response r so zu konstruieren, dass kanalspezifische Parameter mit einfließen. Damit ist das Paar aus Challenge und Response an genau diesen Kanal gebunden. Andernfalls könnte der Angreifer die Challenge und die Response vom verschlüsselten Kanal zwischen Alice und dem Angreifer in einen anderen zwischen dem Angreifer und Bob weiterleiten, ohne dass Alice oder Bob dies bemerken würden (engl. Relaying).

Wahl der Funktion f

Wie bereits angedeutet beruht die Sicherheit des Verfahrens auf der Annahme, dass der Prover (Alice) bzw. ein Angreifer nicht in Lage ist, die Response zu erraten.

Insofern spielt die Wahl der Funktion eine wichtige Rolle. Wählte man als Funktion beispielsweise die Entschlüsselung einer vom Verifier (Bob) verschlüsselten Challenge, so würde die Response aus dem Klartext (vor der Verschlüsselung beim Verifier) bestehen. Bei ungeschickter Wahl der Challenge oder eines schlechten Zufallszahlengenerators beim Verifier könnte ein Angreifer das Ergebnis unter Umständen voraussagen, ohne tatsächlich die Challenge entschlüsseln zu können (vgl. [BNS05], S. 223).

Als Schlussfolgerung sollte man Challenge-Response-Verfahren nach Möglichkeit so konstruieren, dass der Prover zur Berechnung der Response entweder eine Verschlüsselung bzw. Signatur durchführen muss oder das als Response vom Verifier ein Message-Authentication-Code (MAC) über die Challenge erwartet wird.

2.5. Passwortrücksetzverfahren

Mit dem Einsatz von passwortbasierten Authentifikationsverfahren geht die Frage nach der Rücksetzbarkeit eines Passworts einher. Die Notwendigkeit ergibt sich üblicherweise dadurch, dass sich ein Benutzer nicht mehr an sein Kennwort erinnern kann. Andere Gründe sind beispielsweise die Sperrung eines Accounts nach mehrfacher Falscheingabe, aber auch gemeinsam genutzte Accounts (Rollen) nach dem Ausscheiden eines Mitarbeiters.

Beim Zurücksetzen eines Passwort steht der Systembetreiber prinzipiell vor dem gleichen Problem wie beim Login eines Benutzers: Wie kann die Identitätsbehauptung des Benutzers überprüft werden? So reicht es definitiv nicht aus, bei der Benutzerbetreuung (engl. Helpdesk) anzurufen und sich ein neues Passwort durchsagen zu lassen. Rücksetzmechanismen für Passwörter kann man grundlegend in zwei Klassen einteilen: zentrale und dezentrale Mechanismen. Dezentrale Mechanismen wirken im Hoheitsbereich des Benutzers – in der Regel als Teil eines Zwei-Faktor-Authentifikationssystems. Zentrale Mechanismen

wirken beim Betreiber des Systems. Dabei werden in der Regel Datenbankinhalte verändert.

2.5.1. Die PUK – Super-Passwort für dezentrale Entsperrung

Dezentrale Mechanismen sind dadurch gekennzeichnet, dass der Benutzer selbst ein neues Passwort vergeben kann. Der Nachteil besteht u. U. in einer großen Latenz, da das benötigte „Rücksetzpasswort“ eventuell nicht unmittelbar verfügbar ist, z. B. weil es als Brief zu Hause abgeheftet ist, der Benutzer es aber von unterwegs bräuchte.

2.5.2. Zentrale Maßnahmen

Bei den zentralen Mechanismen besteht die Aufgabe darin, den Systembetreiber davon zu überzeugen, ein neues Passwort für ein angegebenes Benutzerkonto zu setzen. Hierbei ist wiederum eine Form der Authentifikation notwendig.

Persönliches Erscheinen

Die Authentifikation kann beispielsweise dadurch erfolgen, dass der betroffene Benutzer bei der Benutzerbetreuung oder einem Administrator persönlich erscheint und sich mit entsprechenden Ausweisen (z. B. Personal-, Studenten- und/oder Mitarbeiterausweis) als Berechtigter zu erkennen gibt. Der Mitarbeiter trägt anschließend das neue Passwort, aufgrund seiner (größeren) Befugnisse, in das Passwortverwaltungssystem bzw. die Benutzerdatenbank ein. Das Verfahren ist zwar sehr sicher, hat aber erhebliche Nachteile:

1. Es funktioniert nur zu den Geschäfts- bzw. Öffnungszeiten des Supports.
2. Der Benutzer muss u. U. eine räumliche Distanz überwinden, was das Verfahren sehr teuer bis praktisch unmöglich machen kann.
3. Es existieren Dienste, bei denen dieses Verfahren – meist aufgrund der hohen Kosten – nicht vorgesehen ist. Dies betrifft vor allem Dienste im Internet.

Alternativen

Neben der Rücksetz-Methode „Persönliches Erscheinen“ existieren daher Methoden, die rechnergestützt über das Internet funktionieren. Hierbei initiiert der Benutzer den Rücksetzprozess über eine festgelegte Schnittstelle, in der Regel eine Webapplikation. Zur Authentifikation werden zwei Optionen genutzt:

1. Der Benutzer authentifiziert die Rücksetzanfrage über ein „Super-Passwort“ oder durch die Beantwortung von (vorab festgelegten) Frage-Antwort-Paaren.
2. Der Benutzer erhält ein neues Passwort über einen weiteren sicheren Kanal.[20] In der Praxis ist dies oftmals eine E-Mail an die hinterlegte Adresse des Benutzerkontos.

[20] Alternativ erhält der Benutzer eine (einmal gültige) URL zu einer Webapplikation, in der er ein neues Passwort selbst festlegen kann.

Beide Verfahren sind problembehaftet. Ersteres ist anfällig für Social-Engineering-Angriffe, da eventuell nicht nur der betroffene Benutzer den „Mädchennamen der Mutter“ oder andere persönliche Dinge kennt. Andererseits ist es problematisch, wenn sich der Benutzer eigene Fragen ausdenkt, weil er im Ernstfall neben dem Passwort selbst auch die Antworten auf die Fragen vergessen hat – etwa wegen der geringen Nutzungshäufigkeit. Oder um es anders auszudrücken: Es bestehen die gleichen Probleme wie bei Login-Passwörtern, vor allem aber ist die Qualität (Erratbarkeit) des Super-Passworts selten befriedigend.

Die Zustellung eines Credentials über einen sicheren Kanal hat ebenfalls Tücken, ist aber sehr weit verbreitet. Allerdings lässt sich ein E-Mail-Zugang schlecht über eine „Rücksetz-E-Mail“ entsperren.

3. Die digitale Identität mit dem neuen Personalausweis

Personalausweise erfüllten schon immer eine wichtige Funktion: den Nachweis der Identität des Inhabers durch die ausstellende Behörde. Im Rahmen der Identitätsprüfung einer Person, der Authentisierung, untersucht der *Verifier* als derjenige, dem ein Ausweis vorgelegt wird, dessen Sicherheitsmerkmale, um die Echtheit des Ausweises festzustellen.

Zum Schutz gegen eine Fremdnutzung erfolgt anhand von Referenzmerkmalen des Inhabers eine Zuordnung von Ausweis zu Inhaber.[1] Klassischerweise wird eine Sichtprüfung des Lichtbilds auf dem Ausweis durchgeführt, zunehmend werden aber auch andere biometrische Merkmale, z. B. Fingerabdrücke, genutzt ([Sch09], S. 7).[2] Stimmen die im Ausweis genannten mit den Merkmalen des Inhabers überein, kann angenommen werden, dass die Identitätsattribute im Ausweis (Name, Anschrift, etc.) dem Ausweisbesitzer entsprechen.[3]

Offenbar funktioniert diese manuelle Ausweisprüfung nur, wenn Verifier und Prover zur selben Zeit am selben Ort sind. So eigenen sich klassische Personalausweise gut zur Personenkontrolle durch hoheitliche Organe und im nicht-elektronischen Geschäftsverkehr, beispielsweise zum Altersnachweis für den Erwerb bestimmter Waren.

Mit zunehmender Verbreitung des elektronischen Handels (E-Commerce) und des damit ebenfalls einhergehenden Online-Betrugs, aber auch aufgrund gesetzlicher Anforderungen spielen Mechanismen zur entfernten Authentisierung eines Geschäftspartners eine immer wichtigere Rolle.

Mit dem Postident-Verfahren [PID11] besteht eine Möglichkeit zur Authentisierung unter Zuhilfenahme des Personalausweises. Hierbei wird durch einen „Stellvertreter“ (den Mitarbeiter der Deutschen Post) ein Äquivalent zur Ausweisprüfung im nicht-elektronischen Geschäftsverkehr geschaffen. Konkret werden bestimmte Ausweisdaten von einem Postmitarbeiter in ein Formular eingetragen, das vom Vertragspartner der Deutschen Post (z. B. eine Bank oder ein Mobilfunkanbieter) bereitgestellt wird. Durch die Unterschrift des Kunden (Ausweisinhaber) und des Postmitarbeiters wird die Korrektheit der Daten im Formular bestätigt.

Leider ist neben den Kosten einer Postident-Authentisierung vor allem die sehr hohe Latenz einer Authentisierung, im Vergleich zu anderen E-Commerce-Prozessen, als eines der K.-o.-Kriterien für viele elektronische Anwendungen anzusehen.

[1] Streng genommen steht erst *nach* der Verifikation fest, ob es sich bei demjenigen um den Ausweisinhaber handelt, der den Ausweis vorlegt. Stattdessen wird der Begriff *Prover* verwendet.

[2] Ein Blick in die Historie zeigt, dass eine Prüfung auch ohne Passbild möglich ist. So wurden vor dessen Einführung Anfang des 20. Jahrhunderts andere Merkmale wie Haar- und Augenfarbe, Statur oder sogar Form des Bartwuchses in frühe Ausweisdokumente aufgenommen. vgl. [Sch09], S. 12f.

[3] Aus diesem Grund erfordert die Prüfung von Berechtigungsnachweisen ohne solche Merkmale, oftmals ebenfalls als „Ausweis“ bezeichnet, dass ein (Identitäts-)Ausweis mit den nötigen Merkmalen hinzugezogen und die Übereinstimmung der Daten auf beiden Dokumenten (Name, . . .) geprüft wird.

Andere Anwendungen mit geringeren juristischen Anforderungen greifen deshalb für einen Identitätsnachweis zu Fotokopien von Personalausweisen. Das Problem ist aber, dass dabei die Sicherheitsmerkmale des Ausweises nicht mehr oder nur noch eingeschränkt prüfbar sind, so dass in der Folge der Nachweis nur bedingt gewährleistet ist.

In diesem Kapitel wird zunächst die Technik des neuen Personalausweises (nPA) betrachtet und die Abläufe bei der eID-Funktion beschrieben.

Im zweiten Teil wird untersucht, welche Szenarien und Topologien bei der Umsetzung von eID-Diensten für den neuen Personalausweis prinzipiell möglich sind, und es werden die Vor- und Nachteile dargestellt.

3.1. Der neue Personalausweis

Nachdem seit Oktober 2005 der Reisepass in der Bundesrepublik Deutschland durch eine neue Generation von Dokumenten mit Funkchip (RFID) abgelöst wurde, folgte alsbald auch für den Personalausweis der Beschluss, auf eine neue Version umzusteigen. Seit November 2010 soll der neue, elektronische Personalausweis (Kurzform: zunächst ePA, später dann nPA) nicht nur die Fälschungssicherheit für hoheitliche Zwecke verbessern, sondern im Gegensatz zum ePass (dem elektronischen Reisepass) auch weitere Funktionen für privatwirtschaftliche Anwendungen mitbringen (vgl. [BKMN08]).

3.1.1. Authentisierung versus Signatur

In bisherigen Geschäftsvorfällen spielen zwei Prozesse zwischen den Parteien eine besondere Rolle, die beim Entwurf des neuen Personalausweises im Fokus standen: die *Authentisierung* und die *Signaturerstellung*, vgl. [BKMN10].

Durch eine (gegenseitige) Authentisierung weisen die Beteiligten (z. B. Verkäufer und Käufer) ihre jeweilige Identität nach. Dabei handelt es sich jedoch um eine *Momentaufnahme* für beide Beteiligte – ein Nachweis gegenüber Dritten wird nicht erbracht. Eine Authentisierung ist vor allem für jene Verträge wichtig, bei denen eine Partei in Vorleistung tritt. Im Online-Handel ist das oft der Regelfall.[4] Anders als im Ladengeschäft – durch deren Betreten der potentielle Käufer den Ladeninhaber implizit authentifiziert – liefert entweder der Verkäufer zunächst die Ware und erhält im Anschluss den Kaufbetrag (Rechnungsstellung) oder umgekehrt: Der Käufer zahlt per Vorauskasse und erhält dann die Ware.

[4] Im Hotelgewerbe wird dies auch für nicht-elektronische Verträge deutlich: Ein Gast erhält erst nach Vorlage eines Ausweises den Zimmerschlüssel, da Übernachtungs- und andere anfallende Kosten in der Regel erst beim Verlassen des Hotels beglichen werden.

Auch um das Problem der fehlenden Identitätssicherheit zu lösen, haben sich Treuhanddienste etabliert, die als „vertrauenswürdige Dritte“ zwischen beiden Geschäftspartnern vermitteln.[5]

Bei Verträgen in Schriftform reicht eine Authentisierung, die die Identität der Vertragspartner sicherstellt, nicht aus; es bedarf einer *expliziten Willensbekundung*, dass die Inhalte des Vertrags akzeptiert werden. Durch eine eigenhändige Unterschrift oder eine qualifizierte elektronische Signatur geben die Vertragspartner eine solche auch nachweislich für Dritte (z. B. für einen Richter) ab.

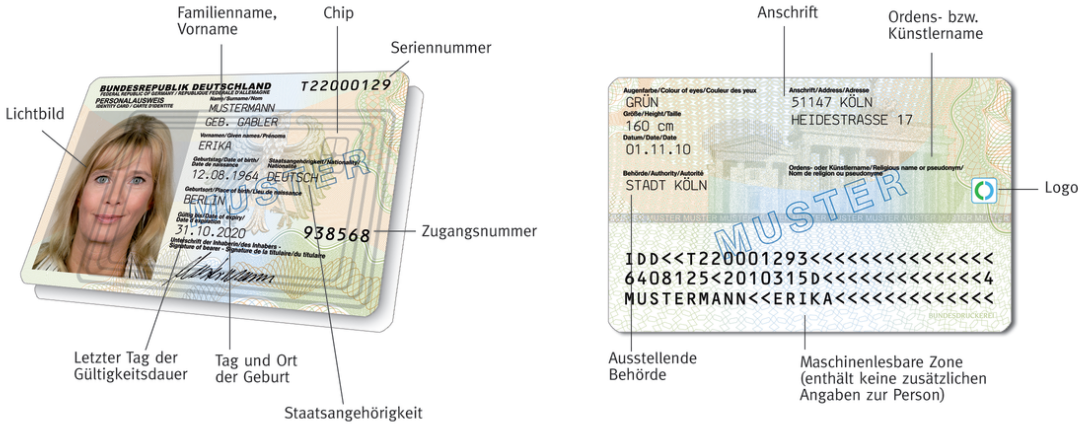

Abbildung 3.1.: Aufbau eines neuen Personalausweises, Quelle: Personalausweisportal

.

3.1.2. Physische Sicht

Beim neuen Personalausweis (nPA) handelt es sich um eine *kontaktlose Smartcard* nach [ISO 14443] (RFID), also um eine standardisierte Karte aus Polycarbonat mit eingebettetem Schaltkreis, der drahtlos kommunizieren kann und seine Versorgungsenergie per Induktion aus einem elektromagnetischen Feld bezieht, das vom RFID-Lesegerät erzeugt wird. Die Karte hat die Größe td-1 nach [ICA08] und entspricht von den Abmessungen her somit dem „Scheckkarten-Format“ ID-1.

Vorrangig dient die Karte weiterhin als visueller Ausweis, d. h., dass die bekannten Daten des Ausweisinhabers auch von einem nPA ablesbar und optische Sicherheitsmerkmale vorhanden sind. Zudem ist die von der International Civil Aviation Organization (ICAO) spezifizierte maschinenlesbare Zone (MRZ) auf der Rückseite aufgedruckt, so dass auch der neue Personalausweis als Reisedokument (MRTD) in vielen Ländern genutzt werden kann.[6]

[5] So bietet ein großes Internetauktionshaus an, den Kaufbetrag einer Auktionsware in Empfang zu nehmen und diesen erst dann an den Verkäufer auszuzahlen, wenn die Ware ordnungsgemäß beim Käufer eingetroffen ist.

[6] http://mrtd.icao.int/.

3.1.3. Logische Sicht: Anwendungen

Analog zu anderen etablierten Smartcard-Systemen stellt der nPA mehrere Anwendungen bereit, die über eine Middleware ausgewählt werden können:

- Die **ePass-Anwendung** erlaubt den Zugriff auf authentische Identitätsdaten. Das sind neben den aufgedruckten Personendaten (bis auf die Anschrift) biometrische Daten (eine digitale Form des Lichtbilds und optionale Fingerabdrücke) des Ausweisinhabers. Zur Nutzung bedarf es eines hoheitlichen Lesegerätes. Diese Anwendung ist auf allen neuen Personalausweisen aktiviert.

- Die **eID-Anwendung**, auch Online-Authentisierung, ermöglicht für *zertifizierte Diensteanbieter* den Zugriff auf personen- und ausweisbezogene Daten über ein Datennetzwerk. Ein Zugriff auf biometrische Daten ist ausdrücklich nicht gestattet. Diese Anwendung wird die zentrale Rolle in dieser Arbeit spielen. Sie ist optional und kann auf Wunsch des Inhabers – gegen Gebühr – im Bürgeramt aktiviert und deaktiviert werden.

- Mit der **Signatur-Anwendung** erhält der Ausweisinhaber die Möglichkeit, die qualifizierte Signatur mit seinem nPA zu nutzen. Voraussetzung dafür ist ein entsprechendes Signaturzertifikat, das über ein Trustcenter bezogen werden muss und nicht vom Aussteller des Ausweises bereit gestellt wird. Der neue Personalausweis ist somit für die Erzeugung qualifizierter Signaturen *vorbereitet*. Diese Anwendung kann auf Wunsch des Inhabers durch Setzen der Signatur-PIN, Erzeugen eines Schlüsselpaares und Installieren eines Signaturzertifikats aktiviert werden, vgl. [TR-03127], S. 34.

Die logische Kommunikation erfolgt nach dem Standard ISO 7816 mittels APDUs.

3.1.4. Zugriff auf die Ausweisdaten

Für den Zugriff auf die hoheitlichen und nicht-hoheitlichen Daten und Funktionen (Altersverifikation, Wohnortabfrage, Pseudonymfunktion) wird ein sogenanntes *Terminal* benötigt. Nach [TR-03127], S. 21ff. gibt es folgende Terminal-Typen:

- hoheitliches Inspektionssystem (zur Polizei- und Grenzkontrolle),

- hoheitliches Authentisierungsterminal (für Änderungen der eID-Daten im Bürgeramt),

- nicht-hoheitliches Authentisierungsterminal (zum Onlinezugriff über das Internet durch Diensteanbieter),

- Signaturterminal (für qualifizierte elektronische Signaturen – QES),

- nicht-authentisiertes Terminal (z. B. die AusweisApp zur Änderung der eID-PIN).

3.1.5. Berechtigungszertifikate

Alle Terminals mit Ausnahme des nicht-authentisierten Terminals, egal ob hoheitlich oder nicht-hoheitlich, benötigen für den Zugriff auf einen nPA ein Berechtigungszertifikat, das von der separaten Public-Key-Infrastruktur mit eigener Zertifizierungsstelle, der CVCA (Country Verifying Certificate Authority, auch „Berechtigungs-CA“ genannt) abgeleitet wurde. Dabei handelt es sich um CV-Zertifikate (Card Verifiable), die für die Prüfung durch Smartcards optimiert wurden. Ein Berechtigungszertifikat beinhaltet unter anderem diese Angaben:

- Referenz auf den Diensteanbieter (CHR – Certificate Holder Reference),
- Referenz auf die DVCA (CAR – Certificate Authority Reference),
- öffentlicher Schlüssel (engl. public key) des Diensteanbieters,
- Zugriffsrechte (CHAT – Certificate Holder Authorisation Template),
- Gültigkeitszeitraum,
- Erweiterungen (vgl. [TR-03127], S. 29f.) wie
  - Angaben zum Diensteanbieter (subjectName, Anschrift, E-Mail-Adresse, Zweck der Anfrage),
  - Name der ausstellenden Berechtigungs-CA (issuerName),
  - Angaben zur zuständigen Datenschutzbehörde,
  - Hashwerte der TLS-Zertifikate des Diensteanbieters (commCertificates).

3.2. Die eID-Anwendung

Mit dem neuen Personalausweis und der eID-Anwendung stellt die Bundesrepublik Deutschland eine sehr fortschrittliche Technologie für die Online-Authentisierung zur Verfügung, die auch hohen Ansprüchen an den Datenschutz gerecht wird.

Anders als bei eID-Projekten in anderen Ländern werden dabei die Daten nicht durch den Aussteller des ID-Dokuments signiert, sondern können im Rahmen der eID-Anwendung über einen sicheren Kanal ausgelesen werden, vgl. [Kub11]. Zudem müssen Organisationen, die Daten aus einem Ausweis lesen wollen, ihre Berechtigung über ein Zertifikat nachweisen. Durch Bindung der Daten an den sicheren Kanal kann der Auslesende Manipulationen ausschließen. Die Authentizität der Daten ist durch eingeschränkte Schreibrechte gesichert, die nur für den Ausweishersteller und entsprechende Terminals mit separaten PKI-Strukturen gewährt werden, wobei auch nur wenige Attribute wie z.B. die Adresse überhaupt geändert werden können.

Neben den aufgedruckten (sichtbaren) Daten eines Ausweisinhabers wie Vorname, Name, Geburtstag und -ort sowie die Anschrift erlaubt der Ausweis – eine entsprechende Berechtigung vorausgesetzt – die Nutzung weiterer Funktionen, die mit dem bisherigen „alten“ Personalausweis nicht möglich waren und aus Sicht des Datenschutz zu begrüßen sind, die:

- Wohnortabfrage, d. h., ein Diensteanbieter kann einen Teilschlüssel gegen die auf dem Ausweis gespeicherte Wohnort-ID – einer leicht modifizierten Variante des Amtlichen Gemeindeschlüssels (AGS) – vergleichen lassen,
- Altersverifikation,
- Pseudonymfunktion (auch Restricted Identification).

So kann ein Online-Dienst beispielsweise prüfen, ob ein Nutzer bereits das 18. Lebensjahr vollendet hat, erhält aber nicht das genaue Geburtsdatum, was eventuelle eine Wiedererkennung des Nutzers ermöglichen würde.

Bei Nutzung der eID-Anwendung eines neuen Personalausweises spielen folgende Entitäten eine zentrale Rolle:

- der Ausweis selbst, bzw. der eingebrachte Chip darin,
- der Ausweisinhaber bzw. der eID-Nutzer,
- der Diensteanbieter – er möchte die Daten auslesen,
- die Zertifizierungsinstanz.

3.2.1. Komponenten

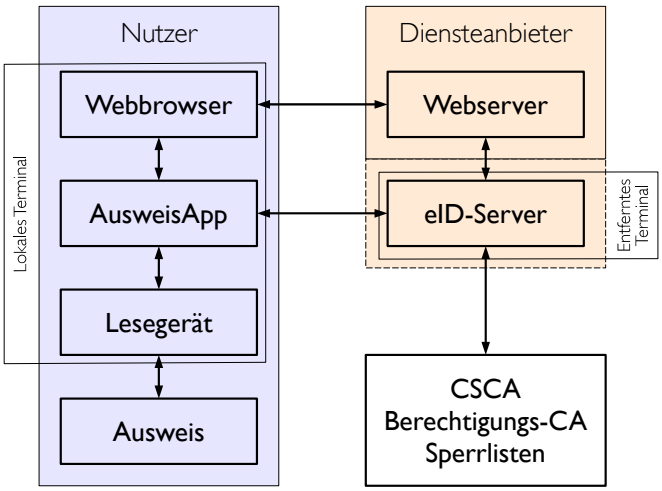

Abbildung 3.2.: Kommunikationsbeziehungen bei der eID-Anwendung (vereinfacht, modifiziert aus [TR-03127]).

Für die Kommunikation mit dem **neuen Personalausweis** benötigt der **Diensteanbieter** eine Software-Komponente, die die im Folgenden beschriebenen Schritte im Protokollablauf umsetzt, die notwendigen Daten entsprechend formatiert und ggf. Fehlersituationen verarbeitet und dem Nutzer in geeigneter Form eine Rückmeldung gibt. Diese Komponente wird als **eID-Server** bezeichnet und stellt das **Remote-Terminal** dar. Remote-Terminal und

das unten besprochene Lokale Terminal bilden zusammen das nicht-hoheitliche Authentisierungsterminal. Der eID-Server kann entweder direkt vom Diensteanbieter oder aber von externen Dienstleistern betrieben werden. Externe Dienstleister werden in diesem Rahmen als *eID-Provider* bezeichnet und erbringen im Auftrag des Mandanten (dem Diensteanbieter) den eID-Service. Soll ein eID-Provider zum Einsatz kommen, sind datenschutzrechtliche Auswirkungen zu berücksichtigen, da dieser unter Umständen – je nach Modellierung (siehe Abschnitt 3.3) – Zugriff auf alle ausgelesenen Daten eines Diensteanbieters hat.

Die Kommunikation mit dem Nutzer wird über das **Lokale Terminal** abgewickelt. Bei Webanwendungen, auf die die eID-Funktion primär abzielt, handelt es sich um *verteilte Anwendungen*. So läuft ein Teil der Gesamtanwendung auf der Nutzerseite im **Webbrowser** und der andere Teil auf der Seite des Diensteanbieters. Da sich Personalausweis und **Lesegerät** auf der Seite des Nutzers befinden, wird eine Teilkomponente benötigt, die einerseits die Kommunikation zwischen Lesegerät/Ausweis[7] und entfernter Anwendung beim Diensteanbieter herstellt und andererseits eine PIN-Eingabe bei Nutzung einfacher Lesegeräte, so genannten Basislesern, ermöglicht.

Um die Anbindung und Interoperabilität von verschiedenen eID-Providern zu gewährleisten, und dies sogar für andere Smartcard-Typen (wie der elektronischen Gesundheitskarte, eGK), wurde mit der **eCard-API** eine Schnittstelle für diese Kommunikation vom BSI spezifiziert [TR-03112]. Mit der **AusweisApp** (vormals „Bürgerclient“) wird eine Software angeboten, die auf Nutzerseite eben diese Schnittstelle realisiert. Sie liefert zudem die notwendigen Erweiterungen (engl. Plugins) für weitverbreitete Webbrowser mit, um eine lokale Kommunikation mit der AusweisApp zu ermöglichen. Die Trennung von Browser und eCard-API-Software hat den Vorteil, dass gerade nicht für jeden Browser am Markt ein Plugin mit der vollständigen eCard-API implementiert werden muss.

3.2.2. Ablauf einer eID-Sitzung

Vor dem eigentlichen Auslesen der Daten mittels eID-Funktion initiiert der Nutzer zunächst eine *Sitzung* (engl. Session) beim IT-System des Diensteanbieters und bereitet somit die eID-Nutzung vor. Das kann beispielsweise über das Auswählen von Gütern in einem Online-Shop (Füllen eines Warenkorbs) oder durch das Ausfüllen und Absenden eines Formulars in einer eGovernment-Anwendung erfolgen. Denkbar ist auch das Auswählen einer Zigarettenmarke an einem entsprechenden Automaten als ein Beispiel der Initiierung einer Sitzung außerhalb des Online-Handels.

Im Anschluss soll der neue (elektronische) Personalausweis vom Diensteanbieter ausgelesen bzw. eine der drei genannten eID-Funktionen genutzt werden, um den Nutzer und somit die in der Sitzung eingegebenen Daten zu authentisieren bzw. eine Berechtigung zu prüfen („Ist der Nutzer volljährig, um auf jugendgefährdende Inhalte zugreifen zu dürfen?“, „Wohnt der Nutzer in Musterstadt?“, „Hat der Nutzer diesen Dienst bereits in der Vergangenheit genutzt?“).

Dazu erteilt die Anwendung beim Diensteanbieter (mittels Browser-Plugin) an das lokale Terminal die Anweisung, eine Verbindung zum eID-Server mittels eCard-API aufzubauen. Die Middleware/AusweisApp prüft hierbei die Bindung des Dienstes mit den Einträgen im Berechtigungszertifikat des Diensteanbieters (z. B. durch einen Fingerabdruck

[7] Die Kommunikation zwischen Lesegerät und Betriebssystem ist durch die PC/SC-Schnittstelle standardisiert.

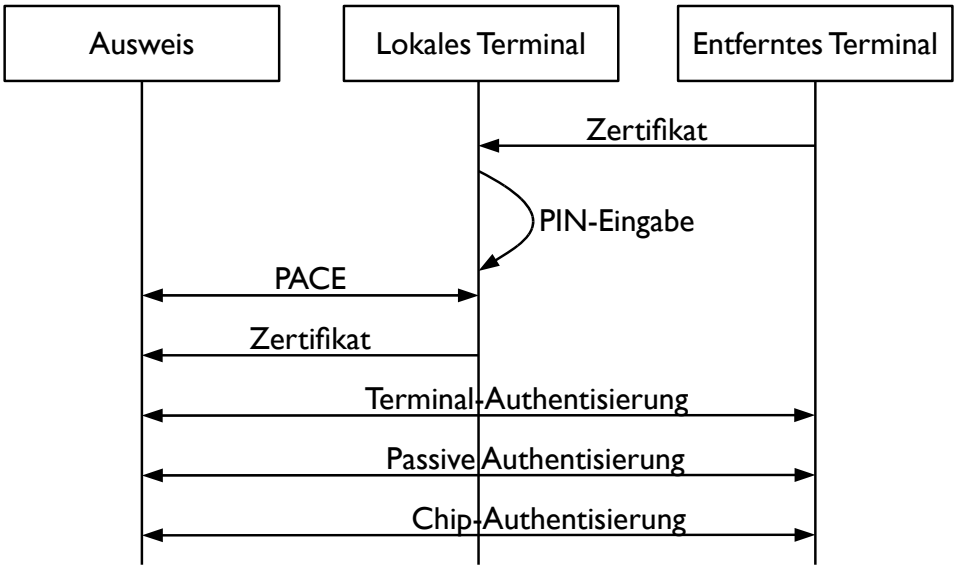

Abbildung 3.3.: Zugriffskontrolle und Authentisierung während einer eID-Sitzung, modifiziert aus [BKMN08].

des SSL-Zertifikats der Webanwendung). Der eID-Server initiiert daraufhin den Auslesevorgang durch Übermittlung eines Steuerkommandos (CHAT). Nach Überprüfung, ob ein Lesegerät und ein Personalausweis erkannt wurde, werden dem Nutzer Daten über den Diensteanbieter aus dem Berechtigungszertifikat, das bei der Initiierung übermittelt wurde, angezeigt. Nach [TR-03127] sind dies mindestens:

- Name, Anschrift und E-Mail-Adresse des Diensteanbieters,
- Internetadresse (URL) des Diensteanbieters und der Zweck der Datenübermittlung,
- Hinweis auf die für den Diensteanbieter zuständige Datenschutzbehörde,
- erwünschte Zugriffsrechte (Menge der abzufragenden Berechtigungen),
- Gültigkeitszeitraum des Zertifikates.

Somit ist der erste Teil der gegenseitigen Authentisierung (Identitätsbehauptung des Diensteanbieters) gewährleistet. Nachdem der Nutzer diese Informationen und die auszulesenden Attribute zur Kenntnis genommen hat, bestätigt er seine Einwilligung zum Fortfahren des Ausleseprozesses durch Eingabe seiner eID-PIN.[8] Dabei ist es dem Nutzer gegebenenfalls möglich, einzelne Attribute bzw. Kategorien abzuwählen – diese werden dann vom nPA-Chip nicht an den Diensteanbieter übermittelt. Es ist aber fraglich, ob es in vielen

[8] Der Ausweisinhaber erhält nach Beantragung des nPA eine Transport-PIN zusammen mit einer PUK (zum Zurücksetzen des Fehlbedienungszählers). Die Transport-PIN ermöglicht das Setzen der benutzerdefinierten eID-PIN.

Szenarien möglich sein wird, nur eine Teilmenge der angeforderten Daten zu übermitteln, da eine „Übererhebung“ von personenbezogenen Daten schon wegen des Grundsatzes der Datensparsamkeit auszuschließen ist, oder andernfalls der Diensteanbieter – aufgrund des Fehlens wichtiger Daten – seinen Dienst nicht vollumfänglich anbieten können wird.[9]

Auf Basis der eID-PIN wird nun mittels **Password Authenticated Connection Establishment** (PACE) ein vertraulicher, authentisierter Kanal zwischen lokalem Terminal und dem nPA-Chip über die RFID-Schnittstelle aufgebaut. Anschließend bestätigt der Diensteanbieter bzw. eID-Server durch die **Terminalauthentisierung** (TA) seine Identitäts- und Berechtigungsbehauptung und prüft im Rahmen der **Passiven Authentisierung** (PA) und der **Chip-Authentisierung** (CA) die Echtheit und Gültigkeit des nPA-Chips. Nach Abschluss der CA verfügen sowohl das Remote-Terminal als auch der nPA-Chip, über Sitzungsschlüssel, auf deren Grundlage der eID-Server über einen Ende-zu-Ende-Kanal mittels Secure-Messaging Zugriff auf die zugesicherten Daten erhält, vgl. [TR-03127] Kapitel 4.

Bei Unterbrechung der Stromversorgung des Chips („Entfernen des Ausweises vom Lesegerät“) wird der Chip in seinen Ausgangszustand versetzt und zum Auslesen von Daten muss der beschriebene Protokollablauf gemäß GAP – *General Authentication Procedure* (PACE, TA, PA, CA, etc.) – erneut durchgeführt werden.

3.2.3. Sicherheit und Datenschutz

Bei der Kommunikation eines Diensteanbieters mit einem nPA im Rahmen der eID-Anwendung sind die Schutzziele Vertraulichkeit, Integrität, Authentizität der Kommunikationspartner, aber auch Anonymität (bzgl. der Wohnortabfrage und Alterverifikation) und der Pseudonymität (bzgl. der Restricted ID) anzustreben. Ein weiteres wichtiges Ziel beim Entwurf des neuen Personalausweises ist die Nutzerorientierung (engl. user centred design): Eine Datenübermittlung erfolgt nur mit Zustimmung des Inhabers. Daneben spielt auch die Möglichkeit einen bestimmten Ausweis jederzeit sperren zu können, eine wichtige Rolle, vgl. [BKMN10], S. 295f.

Für die Bewertung der Sicherheit und der Einhaltung des Datenschutz gilt es die verwendeten Protokolle abzuklopfen. Dabei startet die Analyse bei der Funkschnittstelle und geht zu den Protokollen der GAP über.

RFID- und ISO-7816-Schnittstelle

Entsprechend dem korrekten Verhalten eines RFID-Tags benötigt auch der nPA-Chip einen *Unique Identifier* zur Adressierung im Feld des Lesegeräts. Um eine Identifizierung des Nutzers schon auf RFID-Ebene zu verhindern (Tracking) „wird der Unique Identifier (UID, [ISO 14443] Typ A) bzw. der Pseudo-Unique PICC Identifier (PUPI, [ISO 14443] Typ B) des Chips bei jeder Aktivierung des Chips zufällig erzeugt“ ([TR-03127], S. 8).

Beim Zugriff auf die logische Smartcard-Sicht (nach ISO 7816) lassen sich nur statische Daten auslesen, die für eine Zuordnung zu einem spezifischen Ausweis und damit zu einem Nutzer nicht ausreichen. Ebenso wurden die Domainparameter für PACE (siehe

[9] Aus diesem Grund kann der eID-Server eine Liste von „für den Geschäftsvorgang notwendigen“ Datengruppen und eID-Funktionen über die eCard-API an die AusweisApp schicken, so dass der Nutzer bereits vor der PIN-Eingabe erfährt, dass ein Abwählen bestimmter Daten nicht möglich ist.

nächster Abschnitt) für alle bisherigen neuen Personalausweise universell vergeben. Gegebenenfalls wird es in Zukunft unterschiedliche Ausweisgenerationen geben, deren Domainparameter dann auch für eine Vielzahl von Dokumenten zum Einsatz kommen werden.

Alle anderen Funktionen lassen sich erst nach einer Authentisierung mittels PACE nutzen. Ein Ausspähen „im Vorbeigehen“ wird somit unterbunden.

PACE

Zur Etablierung eines vertraulichen, integritätsgeschützten und authentisierten Kanals über die per se abhörbare Luftschnittstelle wurde das PACE-Protokoll [TR-03110] entwickelt. Im Gegensatz zum Protokoll Basic Access Control (BAC), das beim ePass zum Einsatz kommt, erlaubt PACE die Authentisierung und Etablierung kryptographisch starken Schlüsselmaterials auf Basis von kurzen Passwörtern (der 6-stelligen eID-PIN und CAN bzw. einer 5-stelligen Transport-PIN); beim ePass wird hingegen ein 56-Bit langer Access-Key auf Grundlage der maschinenlesbaren Zone (MRZ) benutzt,[10] vgl. [Eck08a] S. 16. Aus Kompatibilitätsgründen ist zudem die Nutzung der MRZ als Passwort für PACE möglich.

Um ein systematisches Ausprobieren der kurzen PINs und somit einen unzulässigen Zugriff auf den Ausweis-Chip zu verhindern, kommt einerseits ein Fehlbedienungszähler für die eID-PIN zur Anwendung. Andererseits erlaubt weder die (auf dem Ausweis aufgedruckte) CAN noch die MRZ eine Nutzung der eID-Anwendung. Nach zwei Fehlversuchen bei der Verifikation der eID-PIN wird die Anwendung gesperrt, wodurch es möglich ist, einen nPA-Chip (vorübergehend) unbrauchbar zu machen. Ein dritter Versuch lässt sich erst nach einem (erfolgreichen) PACE-Durchlauf mit der CAN als Passwort freischalten. Schlägt auch der dritte Versuch mit der eID-PIN fehl, lässt sich der nPA erst nach korrekter Eingabe des 10-stelligen PIN Unblocking Key (PUK) wieder für die eID nutzen, vgl. [TR-03127], Abschnitt 3.3, S. 16ff.

Terminal Authentication

Im Zuge der Terminal Authentication (TA) erfüllt das Terminal (bzw. Diensteanbieter bei der eID-Anwendung) zwei Aufgaben:

1. die Bereitstellung einer Zertifikatskette, die der nPA-Chip zur Verifikation des Berechtigungszertifikats benötigt,

2. den Nachweis der angekündigten Leseberechtigungen.

Auf Grundlage des im nPA-Chip bei der Auslieferung gespeicherten CVCA-Zertifikats liefert das Remote Terminal alle notwendigen Zertifikate, die zwischen der CVCA und dem Berechtigungszertifikat in der Hierarchie der CV-Zertifikate stehen. Diese Abhängigkeit erlaubt es dem Chip, die Echtheit und Unverfälschtheit – ausgehend von der CVCA – Schritt für Schritt bis zum Berechtigungszertifikat des Diensteanbieters zu überprüfen, siehe auch Abbildung 3.4.

[10] Die MRZ kann z. B. bei Grenzkontrollen mittels eines optischen Lesegeräts gelesen werden, wofür ein physischer Zugriff auf den Pass bzw. Ausweis notwendig ist.

Mittels Challenge-Response-Verfahren (siehe Abschnitt 2.4.2) überprüft der Chip auf dem nPA zudem, ob das Remote-Terminal Zugriff auf den zum Berechtigungszertifikat passenden privaten Schlüssel hat. Dieser ist in einem HSM abgelegt, so dass ein Kopieren des Schlüssels nicht ermöglicht wird. Bei der Prüfung wird vom Chip eine Zufallszahl zur Signierung an das Remote Terminal gesendet und das Ergebnis mittels des öffentlichen Schlüssel (engl. public key) aus dem Berechtigungszertifikat verifiziert. Außerdem wird die Gültigkeit des CV-Zertifikats abgeschätzt: Liegt das Ablaufdatum vor dem im Chip gespeicherten Zeit-Referenzwert ist das Zertifikat definitiv ungültig.

Passive Authentication

Um die Integrität eines nPA-Chips zu prüfen, werden zwei Verfahren benutzt. Die Passive Authentication prüft die Signatur der auf dem Ausweis abgelegten (und lesbaren) Daten, insbesondere den öffentlichen Schlüssel für die Chip Authentication. Die Signatur hierfür wird vom Produzenten der Personalausweise erstellt – dem Document Signer (DS). Er erhält hierfür ein entsprechendes Zertifikat von der zuständigen Autorität des ausstellenden Landes, der CSCA. (In Deutschland ist dies das BSI, der DS ist die Bundesdruckerei.)

Chip Authentication

Bei der Chip Authentication (CA) hingegen wird die Authentizität des nPA-Chips überprüft. Dies ist dahingehend wichtig, da nur echte Ausweise mit echtem Chip die zugesicherten Eigenschaften, insbesondere die Authentizität der gespeicherten (nicht signierten Daten), garantieren. Analog zur Terminal Authentication erfolgt der Nachweis mittels Challenge-Response-Protokoll auf Basis des oben genannten öffentlichen Chip-Authentication-Schlüssels und eines geheimen Schlüssels, der im Chip gespeichert ist, aber zur Wahrung der Anonymität einiger eID-Funktionen für größere Chargen von Ausweisen identisch ist. Nach Durchlauf der CA haben der nPA-Chip und das (Remote-)Terminal ebenfalls zwei Schlüssel K_{ENC} und K_{MAC} als Geheimnisse für die ab diesem Punkt nur noch mögliche Secure-Messaging-Kommunikation etabliert.

Secure Messaging

Die Bindung der auszulesenden Daten an den authentisierten Kanal, aber auch Vertraulichkeit und Integrität der Datenübertragung werden mit den etablierten Verfahren AES für symmetrische Verschlüsselung und Message Authentication Code (MAC) zum Integritätsschutz erreicht.

PKI für Berechtigungszertifikate

Entsprechend der in Abschnitt 3.1.4 genannten Terminal-Typen gibt es für jeden Anwendungsbereich eigene *Document Verifier*, die die jeweiligen Berechtigungszertifikate ausstellen (vgl. [TR-03127], S. 28):

- *Qualitätssicherung beim Ausweishersteller;*
- *Anwendungen in den Ausweisbehörden – Änderungsdienst/Visualisierung;*

- *Berechtigungs-CAs für eBusiness/eGovernment-Diensteanbieter;*
- *hoheitliches Kontrollwesen – Polizei und Grenzkontrolle;*
- *Zertifizierung von Signaturterminals – bestätigte Signaturterminals erhalten ein Zertifikat als Nachweis der Bestätigung.*

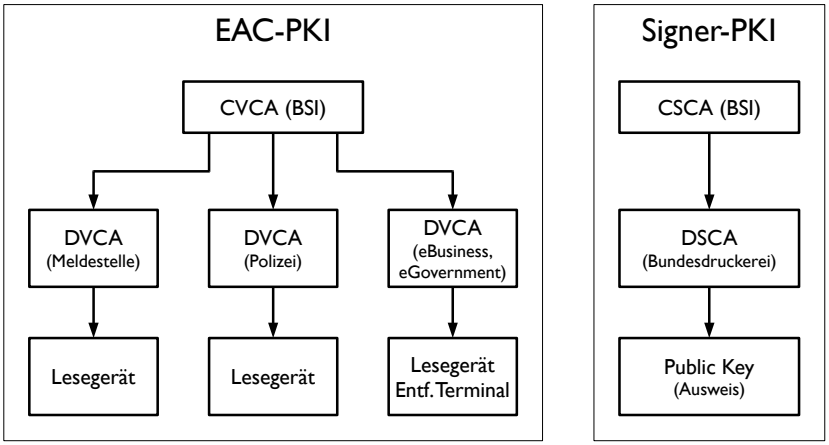

Abbildung 3.4.: Public-Key-Infrastrukturen für den neuen Personalausweises, modifiziert aus [BKMN08].

Da der nPA-Chip keine globale Uhr besitzt, ist es ihm nicht möglich die genaue Gültigkeit des Zertifikats eines Terminals anhand einer Rückrufliste (CRL) zu prüfen.[11] Deshalb gelten diese Zertifikate nur für einen kurzen Zeitraum (z. B. drei Tage) und als Anhaltspunkt für das aktuelle Datum setzt der nPA-Chip einen internen Speicher auf das Ausstellungsdatum des letzten (als gültig verifizierten) Zertifikats.[12] Bestätigte Signaturterminals stellen eine Ausnahme dar, da hier die Zertifikate/Schlüssel „in Hardware gegossen werden". Zur Aktualisierung dieses „Referenzwertes" (z. B. nach langer Nichtnutzung der eID-Anwendung) ist die Möglichkeit vorgesehen, einen Online-Zeitaktualisierungsdienst zu kontaktieren, der keine weiteren Zugriffsrechte auf die Ausweisdaten hat und nur genau diesem Zweck dient.

3.3. eID-Szenarien

Nachdem der potentielle Diensteanbieter mit dem Ablauf einer eID-Sitzung vertraut ist und sich entschieden hat, ein Berechtigungszertifikat zu beantragen, bleibt die Frage: *Welche Möglichkeiten zum Betrieb eines eID-Servers gibt es und welche Maßnahmen sind zu ergreifen?*

Im Weiteren wird davon ausgegangen, dass die Dienste in Form von Webanwendungen erbracht werden. Andere mögliche Szenarien – wie der oft genannte „Zigarettenautomat"

[11] Eine Rückrufliste ohne Zeitstempel wäre ohne Wirkung, da der illegitime Diensteanbieter eine ältere Version der Liste – ohne das gesperrte Zertifikat – bereitstellen könnte.

[12] Ausgenommen Zertifikate von nicht-hoheitliche Terminals.

– Andere mögliche Szenarien – wie der oft genannte „Zigarettenautomat“ – sind nicht Teil der Betrachtungen.

Zunächst ist zu erwähnen, dass es technisch gesehen keines eID-Servers bedarf. Die Funktionalität könnte ebenso gut – unter Berücksichtigung der notwendigen Schnittstellen – von der obligatorischen Webanwendung realisiert werden. Da es allerdings unwirtschaftlich ist, eine solche komplexe Anwendung für jeden Diensteanbieter neu zu programmieren, und zudem eine Zertifizierung notwendig ist, wurde im Rahmen der [TR-03130] diese Komponente spezifiziert. Das Integratorenhandbuch der Bundesdruckerei in [BDr10] beschreibt die Infrastruktur, Installation und Konfiguration des eID-Service, wie er im (offenen) Anwendungstest zur Verfügung stand.

Bevor die einzelnen Ausprägungen der eID-Server analysiert werden, gilt es, die Schnittstellen eines solchen, aber auch die der Gesamtanwendung („den Dienst“) genauer unter die Lupe zu nehmen. Für eine Bewertung der Szenarien müssen zudem die Anforderungen an einen eID-Einsatz definiert werden.

3.3.1. Komponenten

Im Folgenden wird der Fokus auf der Ebene des Diensteanbieters liegen und die Existenz der Komponente „eID-Server“ vorausgesetzt.

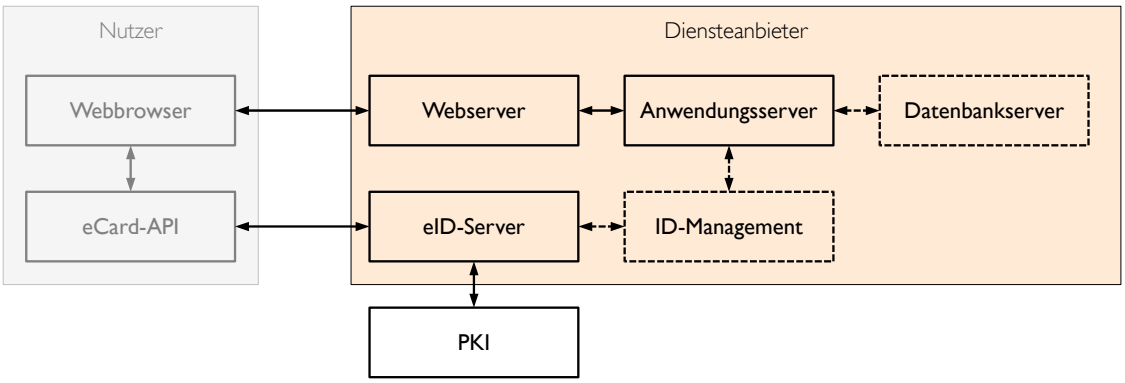

Abbildung 3.5.: Komponenten beim Diensteanbieter (Quelle: [TR-03130]).

Zudem wird es üblicherweise weitere Komponenten geben:

- ein Webserver ist die zentrale Komponente des Diensteanbieters – er bildet die primäre Schnittstelle zum Nutzer, hierüber wird die Webanwendung (die den eigentlichen Dienst darstellt) angesprochen;
- ein Anwendungsserver bildet oftmals die Laufzeitumgebung für die Webanwendung bzw. für Teile davon;
- nahezu obligatorisch ist ein Datenbankserver, der als zentrale Komponente die Daten der Webanwendung und für angeschlossene Prozesse bereitstellt (Prinzip der „Trennung von Form und Inhalt“);

- eine ID-Management-Komponente ist sinnvoll, wenn die Webanwendung über mehrere Authentisierungsquellen verfügt und die eID-Anwendung als weitere Quelle genutzt werden soll (vgl. [TR-03130], Abschnitt 2.4.1);
- die PKI-Komponente deutet den Zugriff auf notwendige Zertifikate, die während der eID-Sitzung benötigt werden (u. a. das Berechtigungszertifikat, CV-Zertifikate und Sperrlisten).

Diese Komponenten können (beispielsweise zur Erhöhung der Verfügbarkeit) auch mehrfach vorgehalten werden.

3.3.2. eID-Server

Schnittstellen

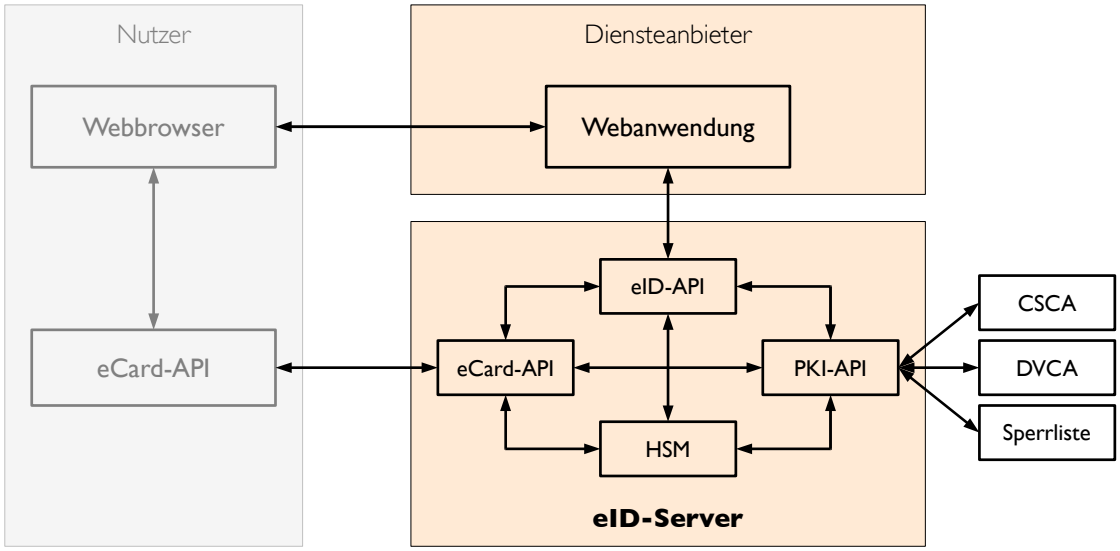

Abbildung 3.6.: Schnittstellen eines eID-Servers (modifiziert aus [TR-03130]).

Der eID-Server muss eine Reihe von Schnittstellen zur Erfüllung des eID-Service bedienen, Abbildung 3.6 zeigt das Zusammenwirken der Module an den Schnittstellen. Zunächst müssen über eine Management-Schnittstelle bestimmte Parameter, wie Zugangsdaten für angeschlossene Dienste, aber auch Schlüsselmaterial für die Authentisierung der Mandanten, sofern vorhanden, angelegt werden.

Über eine eID-API kann er dann mit dem Web- bzw. Anwendungsserver des Diensteanbieters kommunizieren. Dabei sind gemäß [TR-03130], Kapitel 3, signierte XML-Datenstrukturen zu benutzen, die per WSDL spezifiziert wurden. Um die Vertraulichkeit der Daten über diese Schnittstelle zu gewährleisten, ist eine Transportverschlüsselung (z. B. per SSL/TLS oder VPN-Techniken) zu benutzen.

Mittels eCard-API kommuniziert der eID-Server mit der Client-Instanz der eCard-API, in der Regel die AusweisApp.

Letztlich beschafft der eID-Server benötigte Zertifikate (von der DVCA und CSCA) und Sperrlisten über die PKI-API von den definierten Diensten, vgl. [TR-03130], Abschnitt 2.2.3.

Aufgaben

Die Komponente eID-Server kapselt zahlreiche Funktionen, unter anderem um diese Aufgaben zu erfüllen, vgl. [VfB11], Abschnitt 2.3:

- **Auslesen der Ausweisdaten und Bereitstellung** für den Diensteanbieter über geeignete Schnittstellen: Dies ist der primäre Grund zum Betrieb des eID-Server.
- Optional eine **Mandantenverwaltung**. Sofern der eID-Server nicht nur für einen Diensteanbieter agiert, ist es notwendig, Zugangsdaten für die Abholung und Aktualisierung der Berechtigungszertifikate (der Diensteanbieter) abzulegen. Weiterhin muss kryptografisches Schlüsselmaterial für die Kommunikation zwischen Diensteanbieter (Webanwendung) und eID-Server (eID-Schnittstelle) eingerichtet und ggf. aktualisiert werden können.
- **Betrieb eines HSM und Zertifikatspeichers**: Zur sicheren Aufbewahrung von kryptografischen Schlüsseln muss der Betreiber des eID-Servers ein Hardware-Sicherheits-Modul (HSM) bereithalten. Die darin gespeicherten (geheimen) Schlüssel können nicht ausgelesen werden, ein Zugriff erfolgt nur über Funktionen zur Verschlüsselung bzw. Signierung einer zu übergebenen Nachricht. (Typisch ist das Signieren einer kryptografischen Prüfsumme mit einem asymmetrischen Verfahren auf Grundlage eines im HSM gespeicherten privaten Schlüssels.)

  Der Zertifikatspeicher bewahrt das bzw. die Berechtigungszertifikat(e) für die Diensteanbieter, aber auch notwendige CV-Zertifikate der CAs zur Generierung von Zertifikatsketten auf. Ein Zertifikatspeicher erleichtert das Vorhalten (Caching) der notwendigen Zertifikate, die zu aktualisieren sind und somit nicht erst während der eID-Sitzung heruntergeladen werden müssen.
- **Aktualisierung von Berechtigungszertifikaten und Sperrlisten**: Wie bereits dargelegt haben Berechtigungszertifikate für Diensteanbieter nur eine Laufzeit von drei Tagen. Daher ist es erforderlich, Automatismen zu haben, die die Aktualisierung dieses Zertifikats effizient ermöglichen.

  Daneben bedarf es auch einer regelmäßigen Erneuerung der Ausweis-Sperrlisten, um als ungültig gekennzeichnete Ausweise von der Nutzung der eID-Anwendung auszusperren. Diese Sperrlisten sind an den Terminal-Sektor gebunden und somit Diensteanbieter-spezifisch, vgl. [TR-03127], S. 32.
- **Bereitstellung von CV-Zertifikaten**: Da das Gültigkeits- bzw. Ablaufdatum nicht nur für Berechtigungszertifikate, sondern auch für DVCA-Zertifikate (der Berechtigungs-CA) gilt und diese geringer als die Gültigkeitsdauer des Personalausweises ist (6 bzw. 10 Jahre), muss der nPA-Chip vor einer Prüfung des Berechtigungszertifikats im Zuge der Terminal Authentication (TA) erst die – aus seiner Sicht – neuen CA-Zertifikate einspielen. Für die Bereitstellung ist ausschließlich der Diensteanbieter (bzw. das Terminal bei hoheitlichen Anwendungen) zuständig.

3.3.3. Anforderungen an die eID-Infrastruktur

Generell müssen die beteiligten Server entsprechend dem Stand der Technik gesichert sein. Die IT-Grundschutz-Kataloge vom BSI (früher: IT-Grundschutzhandbuch) [ITGS09] behandeln entsprechende Gefährdungen und Maßnahmen.

Zur Integration eines eID-Servers in die Infrastruktur des Diensteanbieter gibt das BSI in [TR-03130], Abschnitt 2.3, eine Reihe von Anforderungen vor, die bei Bewertung eines konkreten Szenarios zu berücksichtigen sind.

Da in der Regel personenbezogene Daten zwischen dem Webbrowser und dem Webserver für die Erbringung des Dienstes übertragen werden, ist diese Kommunikation per SSL/TLS (HTTPS) zu sichern. Bei der Ausstellung eines Berechtigungszertifikat wird dafür ein Hashwert des SSL-Zertifikats (X.509) berücksichtigt.

In jedem Fall muss eine Zuordnung der Kanäle zwischen Webbrowser und Webserver sowie zwischen eCard-API-Client (AusweisApp) und eCard-API-Server (eID-Server) vorgenommen werden, um eine missbräuchliche Nutzung zu verhindern. Das notwendige gemeinsame, generierte Geheimnis (PSK) wird über den HTTPS-Kanal zum eCard-API-Client und zum eCard-API-Server übertragen, vgl. [TR-03130], S. 15. Müssen die Daten zwischen Diensteanbieter und eID-Server über offene Netze übertragen werden (entfernter Betrieb des eID-Servers), sind die Daten zu verschlüsseln und zu signieren (ebd.).

Zusätzlich sollten noch diese Anforderungen gestellt werden:

- Keine Signierung der gelesenen Ausweisdaten bzw. wenn dann nur mit Mandantenspezifischen Signaturschlüsseln. Andernfalls könnte der Diensteanbieter analog zu [TR-03127], S. 13, einen „kryptographischen Echtheitsnachweis an Dritte weitergeben“, da der eID-Provider die Signatur nicht abstreiten bzw. jeder diese Signatur prüfen könnte. Dies ist insbesondere hinsichtlich der Tatsache zu berücksichtigen, dass es (momentan) nur wenige eID-Provider gibt.

- Es muss (z.B. durch ein Audit) geprüft werden, dass vom eID-Provider keine Authentisierungsprofile erstellt werden (können), vgl. [TR-03130], Abschnitt 2.5:
  „Die Identitätsdaten werden technisch vom nPA vorgehalten und bereitgestellt und nicht vom eID-Service. Der eID-Service ist daher auch dann kein Identity Provider, wenn er entfernt und ggf. durch andere betrieben wird.“

- Datensparsamkeit: Bei Auslagerung zu externen eID-Providern ist sicherzustellen, dass diese nur Zugriff auf Daten erhalten, die für die Erbringung des eID-Service unbedingt notwendig sind.

3.3.4. Szenario I: Lokaler eID-Server

Im einfachsten Fall wird der eID-Server im Hoheitsbereich des Diensteanbieters betrieben. Dafür muss der Diensteanbieter ein Hardware-Sicherheits-Modul (HSM) gemäß [CPeID], S. 52, vorhalten, in dem die zum Berechtigungszertifikat passenden privaten Schlüssel sicher gespeichert sind. Im Rahmen der Terminal-Authentication wird vom nPA-Chip geprüft, ob der Diensteanbieter Zugriff auf eben diesen Schlüssel hat und im Erfolgsfall Zugriff auf die zugesicherten Daten des Ausweises gewährt. Der Fall, dass der Diensteanbieter

den eID-Server örtlich getrennt von der Webanwendung betreibt (vgl. [VfB11], Abschnitt 2.4.2), unterscheidet sich technisch wenig von diesem Szenario und wird nicht gesondert betrachtet.

Da für viele Diensteanbieter das Vorhalten dieses HSM und entsprechender Server-Hardware und -Software inkl. Lizenzen unwirtschaftlich ist, besteht eine Möglichkeit darin, diesen „Dienst“ zu einem eID-Provider auszulagern.

3.3.5. Szenario II: Entfernter eID-Server mit Vollzugriff

Stellvertretend für den Diensteanbieter führt ein entfernter eID-Server in dessen Auftrag das beschriebene Protokoll zum Auslesen der Daten aus. Nachdem der Diensteanbieter einen Vertrag mit einem eID-Provider geschlossen und dieser die notwendigen Zugangsdaten und Schlüssel sowie das Berechtigungszertifikat für den Mandanten angelegt hat, kann der Diensteanbieter die Abfrage von Ausweisdaten über die eID-Schnittstelle auslösen (nachdem der Nutzer eine Sitzung beim Dienst initiiert hat).

Über den Kanal zwischen dem Webbrowser beim Nutzer und dem Webserver beim Diensteanbieter wird die Middleware angewiesen, eine eCard-API-Verbindung aufzubauen. Bei der AusweisApp erfolgt die technische Realisierung über ein Webbrowser-Plugin.

Hierüber erfolgt nun der Auslesevorgang wie in Abschnitt 3.2.2 erläutert (Anzeige der Anbieter- und Datenschutzinformationen, PIN-Eingabe, PACE, TA, CA, SM).

Sofern das Auslesen der gewünschten Daten erfolgreich war, verpackt der eID-Server diese gemäß der oben beschriebenen WSDL-Definition und signiert das Ergebnis per XML-Signatur mit dem Mandantenspezifischen Signierschlüssel. In Fehlerfällen wird stattdessen eine XML-Nachricht mit der Fehlermeldung an die Webanwendung zurückgegeben.

Da die Webanwendung nicht wissen kann, wann die eID-Transaktion abgeschlossen ist, erfolgt periodisch eine Status-Anfrage (getResultRequest) über die eID-API, bis die Anfrage erfolgreich ist. In Abbildung 3.7 ist der komplette Ablauf einer eID-Anfrage skizziert.

3.3.6. Anwendungstest: eID-Service der Bundesdruckerei

Für den in dieser Arbeit beschriebenen Benutzerverwaltungsdienst wurde der im Rahmen des offenen Anwendungstests bereit gestellte eID-Service der Bundesdruckerei genutzt. Hierbei verhält sich der Ablauf der eID-Kommunikation etwas anders als in Szenario II beschrieben. Für das Auslesen des Personalausweises leitet die Webanwendung beim Diensteanbieter bereits auf Ebene des Anwendungsprotokolls (HTTP) den Webbrowser zu einem Webserver beim eID-Provider (die Bundesdruckerei) weiter und gibt damit die Kontrolle über den Programmablauf der Webanwendung an den eID-Provider ab. Die Webanwendung beim eID-Provider wiederum leitet dann die Initiierung der eCard-API-Sitzung ein, analog zur Beschreibung oben. Der Transport der eID-Daten zwischen Diensteanbieter und eID-Provider erfolgt mittels SAML-Nachrichten über die HTTPS-Verbindungen zwischen Diensteanbieter und Nutzer sowie zwischen Nutzer und eID-Provider. Das Routing entspricht dem SAML-Profil *Web Browser SSO Profile* sowie dem *Redirect/POST-Binding*, d. h., die Anfrage vom Diensteanbieter zum Auslesen wird via HTTP-Redirect an den Nutzer und dann an den eID-Provider übertragen, vgl. [OAS08], Abschnitt 5.1.2. Die Antwort des eID-Providers wird über SAML-formatierte Daten per HTTP-POST übermittelt. Das vom

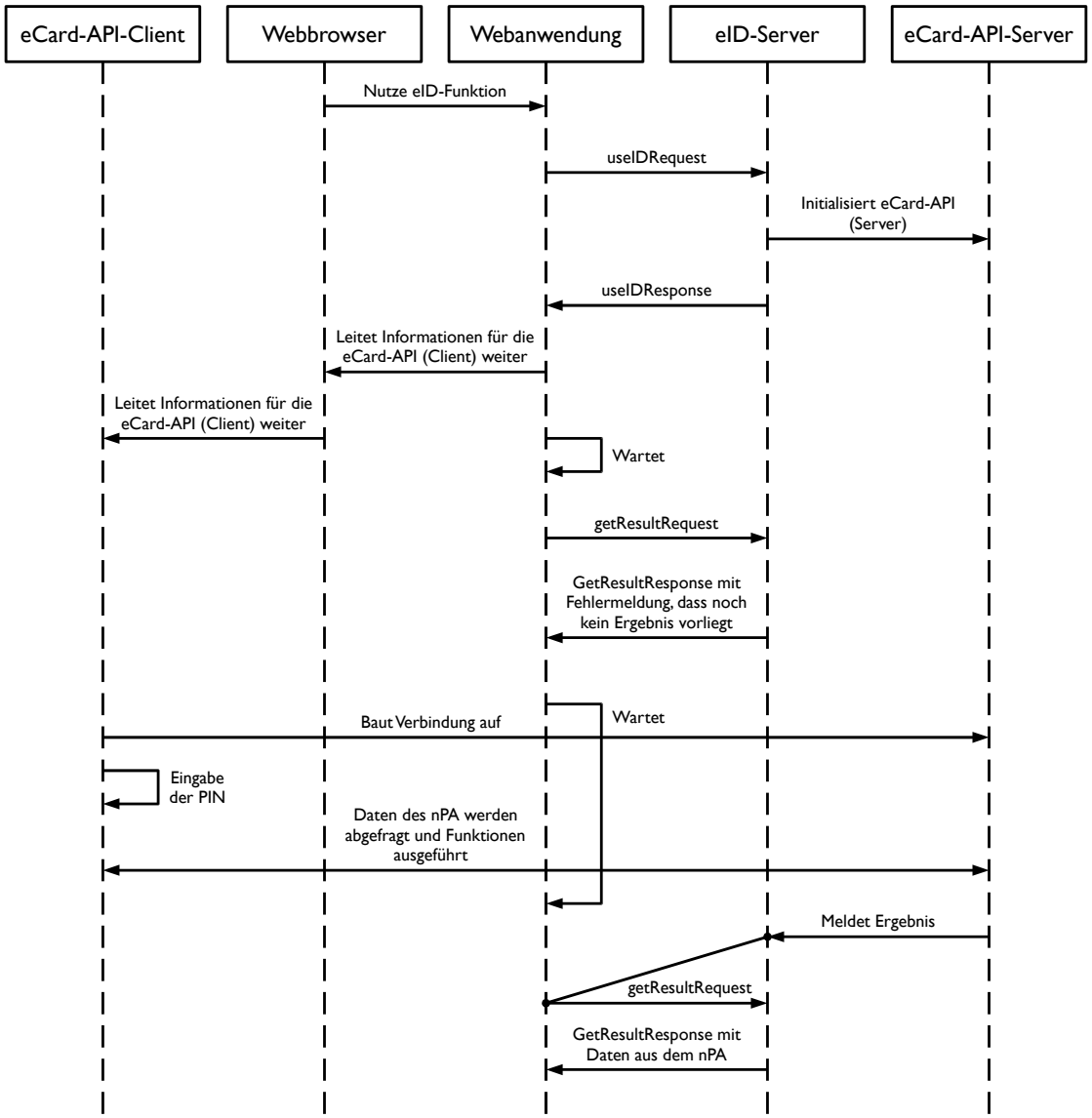

Abbildung 3.7.: Sequenzdiagramm: Funktionaler Ablauf einer eID-Anfrage (Quelle: [TR-03130]).

eID-Provider vorbereitete Formular wird (automatisch) vom Nutzer an eine vordefinierte Adresse der Diensteanwendung (Callback-URL) beim Diensteanbieter abgeschickt.

Für die Authentisierung der Anfrage signiert der Diensteanbieter den SAML-Request mit einem bei der Einrichtung des Mandantenkontos vereinbarten Signierschlüssel. Die Antwort (SAML-Response) wird vom eID-Provider mit einem Mandantenspezifischen Signierschlüssel signiert und mit dem öffentlichen Schlüssel des Mandanten verschlüsselt. So ist sichergestellt, dass außer dem Diensteanbieter niemand die personenbezogenen Daten lesen kann und Manipulationen an den SAML-Nachrichten durch den Diensteanbieter er-

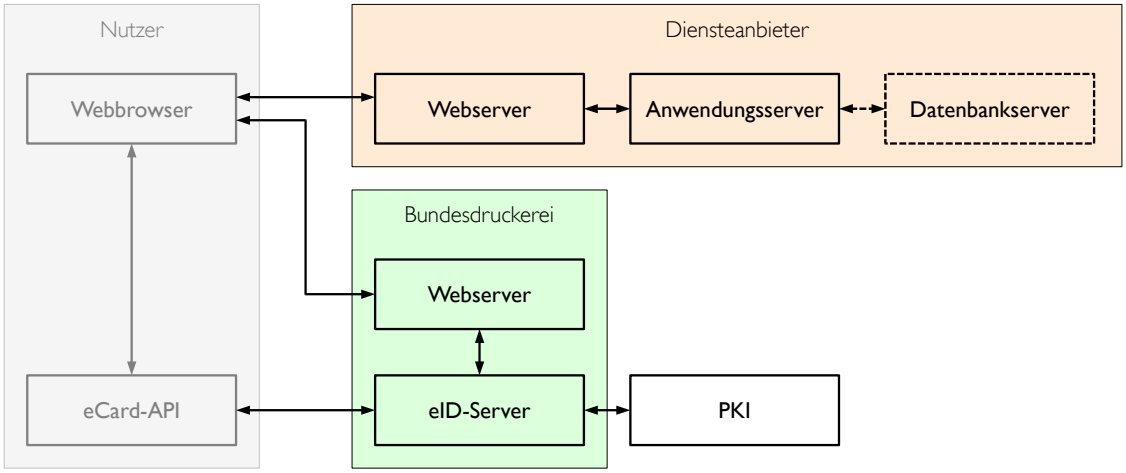

Abbildung 3.8.: Einbindung des eID-Server der Bundesdruckerei beim Anwendungstest.

kannt werden. Der Einsatz von Mandantenspezifischen Signierschlüsseln ermöglicht es, dass der eID-Provider eine konkrete Signatur abstreiten kann und diese somit für einen Dritten wertlos ist, da der Diensteanbieter selbst diese Signatur erzeugt haben könnte.

Sicherheitsmaßnahmen

In der Technischen Richtlinie zum eID-Server [TR-03130] wird im Anhang A ebenfalls auf die Eigenschaften eines eID-Server mit SAML-basierter eID-Schnittstelle eingegangen. Die Aussagen beziehen sich dabei auf das Web Browser SSO Profile ([OAS05a], Abschnitt 4.1), das wie erwähnt auch beim eID-Service der Bundesdruckerei benutzt wird.

Es werden die folgenden Maßnahmen vorgeschrieben, um die Kommunikation zwischen eID-Server und Webanwendung beim Diensteanbieter abzusichern:

- **Absicherung der Transportebene mittels SSL/TLS.**
  Dies ist in diesem Szenario durch den geforderten Einsatz von HTTPS als Kommunikationsprotokoll zwischen Benutzer und Diensteanbieter sowie zwischen Benutzer und eID-Provider gewährleistet.

- **XML-Signatur der SAML-Nachrichten.**
  Um die Authentizität der eID-Anfrage und -Antwort zu gewährleisten, werden die entsprechenden SAML-Nachrichten signiert. Die Signatur erfolgt über das vollständige XML-Dokument, das dem SAML-Protokoll zugrunde liegt.

- **XML-Verschlüsselung in den SAML-Nachrichten.**
  Die Verschlüsselung des SAML-Request stellt sicher, dass neben dem Absender der Nachricht jeweils nur der Adressat Kenntnis vom Inhalt bekommen kann. Aufgrund der Transportsicherung durch die TLS-Kanäle könnten die SAML-Nachrichten prinzipiell nur an den Endpunkten wirksam angegriffen werden. Gleiches gilt für die XML-Signatur.

- **Bindung der drei Kommunikationskanäle einer Sitzung aneinander.**
  Um die Gültigkeit der im Zuge der eID-Sitzung ausgetauschten SAML-Nachrichten auf genau diese Sitzung zu beschränken, ist es erforderlich die drei Kanäle
  - Benutzer ↔ Diensteanbieter,
  - Benutzer ↔ Webserver beim eID-Provider und
  - eCard-API-Client ↔ eCard-API-Server

  miteinander zu verknüpfen. Hierfür wird jeweils ein gemeinsames Geheimnis (engl. preshared key – PSK) verwendet, so dass sich kein Dritter per Man-in-the-Middle-Angriff in die Kommunikation einklinken kann.

- **Verwendung von Zeitstempeln in den SAML-Nachrichten.**
  Als Schutz vor Replay- und Denial-of-Service-Angriffen, also dem Wiedereinspielen von bereits gesendeten (und eventuell verarbeiteten) Nachrichten bzw. dem Überlasten des Dienstes beim Diensteanbieter oder eID-Provider, werden Zeitstempel innerhalb der SAML-Nachrichten verwendet. Ist der Zeitstempel einer empfangenen Nachricht außerhalb des erwarteten Zeitfensters, wird die Anfrage bzw. Antwort als ungültig verworfen. Dadurch ist es allerdings zwingend erforderlich, dass der Diensteanbieter und der eID-Server die gleiche (globale) Uhrzeit benutzen. Hierzu existieren aber beispielsweise mit dem *Network Time Protocol* (NTP) etablierte Mechanismen, die diese Anforderung leicht realisieren können.
  Replay-Angriffe funktionieren somit nur innerhalb dieses kurzen Zeitfensters.
  Für Denial-of-Service-Angriffe müsste der Angreifer – sofern sich der Dienst die Kombination (Zeitstempel, Signaturersteller) merkt – jeweils „frische“, korrekt signierte SAML-Nachrichten erzeugen, was einen solchen Angriff aber zu teuer werden lässt.

Durch die Kombination der Maßnahmen ist sicher gestellt, dass es zumindest auf Ebene der SAML-Nachrichten nicht möglich ist, einen Identitätsnachweis (eine sogenannte *Assertion*) zu „stehlen“ und in einer anderen Sitzung als den eigenen auszugeben. Einerseits müsste der Angreifer in den Besitz der Assertion kommen (z.B. durch einen Man-in-the-Middle-Angriff oder Trojaner auf dem Rechner des Opfers). Andererseits würde die Assertion nichts nützen, da diese durch die Kanalbindung an den ursprünglichen Kanal zwischen Diensteanbieter und Opfer an das Geheimnis (PSK) gebunden ist. Der Angreifer – in der Rolle eines Nutzers – kann in einer unabhängigen Sitzung das Geheimnis nicht zu seinen Gunsten beeinflussen, da dieses typischerweise vom Diensteanbieter generiert wird. Es ist aber zu berücksichtigen, dass ein Identitätsmissbrauch auf anderer Ebene der eID-Nutzung möglich ist, so wie es in [MO10] gezeigt wurde.

Wie in [OAS05b] neben den soeben genannten Gefährdungen beschrieben, muss sich der Betreiber des Dienstes, aber auch der des eID-Servers im Klaren sein, dass die Sicherheit der SAML-Kommunikation und der aufgestellten Identitätsbehauptungen von neuralgischen Punkten abhängen. Zum einen kann eine SAML-Assertion offensichtlich nur so aussagekräftig sein wie das Resultat ihrer Erfassung. Die Assertion hängt also unmittelbar von der eID-Anwendung auf der Seite des eID-Providers ab. Zum anderen ist ebenfalls ein Augenmerk auf die Anforderungen der benutzen Sicherheitsprotokolle zu richten. So sind die kryptographischen Schlüssel über einen *sicheren* Kanal auszutauschen und sicher bei den beteiligten Parteien zu speichern. Schließlich ist für die Durchführung der Protokolle ein

kryptographisch sicherer Zufallszahlengenerator erforderlich, da sonst eventuell benutzte Ephemeralschlüssel vorhersagbar wären.

Als Anmerkung wird in [TR-03130], Anhang A 2.2, von der Verwendung des Redirect-Bindings abgeraten, wie es für die Übermittlung des SAML-Request während des Anwendungstests eingesetzt wurde.[13] Da dabei die SAML-Daten und die Signatur in der URL kodiert werden und die maximale Größe der URL von der Implementierung des Webbrowsers abhängt, kann nicht sicher gestellt werden, dass die Übermittlung des SAML-Request zu Fehlern führt. Andererseits muss dabei auch berücksichtigt werden, dass für das Nutzungsszenario nur sehr wenige Webbrowser in Frage kommen, für die ein AusweisApp-Plugin bereit steht. Alle Browser, die im Rahmen der Implementierung zusammen mit dem eID-Service der Bundesdruckerei getestet wurden, zeigten hier keine Anfälligkeiten – die Größe des SAML-Requests ist zudem beschränkt, was eingabeabhängige Laufzeitfehler ausschließt. Dennoch sollte das Problem bei der Portierung auf andere Geräte (beispielsweise Smartphones) im Zusammenhang mit SAML-basierten eID-Servern berücksichtigt werden.

3.3.7. Szenario III: Entfernter eID-Server mit eingeschränktem Zugriff

Das in Abschnitt 3.3.5 beschriebene Szenario hat ein inhärentes Problem: Es laufen sehr viele personenbezogene Daten bei wenigen eID-Providern an wenigen zentralen Stellen auf, so dass prinzipiell Bewegungsprofile der Form „Nutzer *A* authentisiert sich beim Online-Shop *B*“ oder auch „Nutzer *C* fragt Daten von Behörde *D* ab“ erstellt werden könnten. Aufgrund von Datenschutzbestimmungen und Zertifizierungen soll dies zwar unterbunden werden, dennoch wecken solche „Daten-Knotenpunkte“ Begehrlichkeiten.

Mit dem folgenden Szenario wird eine Möglichkeit aufgezeigt, wie dieses Problem datenschutzfreundlich gelöst werden kann. Dabei handelt es sich aber um eine Skizze, die in keinem der bekannten eID-Server implementiert ist. Notwendig ist dafür eine weitere Komponente beim Diensteanbieter: der eCard-API-Proxy.

Zunächst wird wie im oben beschrieben Ablauf eine Sitzung mit der Webanwendung initiiert. An der Stelle des eCard-API-Verbindungsaufbaus wird dem eCard-API-Client (AusweisApp) statt der Adresse des eID-Servers die des eCard-API-Proxys mitgeteilt. Dieser Proxy kann den gesamten Protokollfluss der eCard-API-Kommunikation nachvollziehen und leitet nun alle eCard-API-Pakete an die eCard-API des eID-Servers weiter, insbesondere die für den Nachweis des Zugriff auf den privaten Schlüssel zum Berechtigungszertifikat. Nach erfolgreichem Durchlauf von TA und CA sowie der Sperrlistenprüfung ist der eID-Server nun im Besitz der Ephemeral-Schlüssel für die Secure-Messaging-Kommunikation mit dem Ausweis. Diese Schlüssel sind flüchtig und gelten nur für genau diese Sitzung zum Auslesen der erlaubten Daten aus dem nPA. Der eCard-Proxy beim Diensteanbieter würde nun aber – im Gegensatz zur eCard-Kommunikation nach Szenario II – keine Secure-Messaging Anfragen zulassen, sondern stattdessen die Ephemeral-Schlüssel über ein geeignetes Protokoll beim eID-Server abfragen und selbstständig den nPA-Chip unter Nutzung der Schlüssel per Secure-Messaging via eCard-API befragen. Hierfür benötigt der eCard-API-Proxy zudem Kenntnisse über den Aufbau der Datenstrukturen, die

[13] In neueren Versionen der Testinfrastruktur der Bundesdruckerei wird hingegen, wie in der [TR-03130] empfohlen, nur noch das POST-Binding benutzt und die Daten im SAML-Request sind zudem für den eID-Server verschlüsselt.

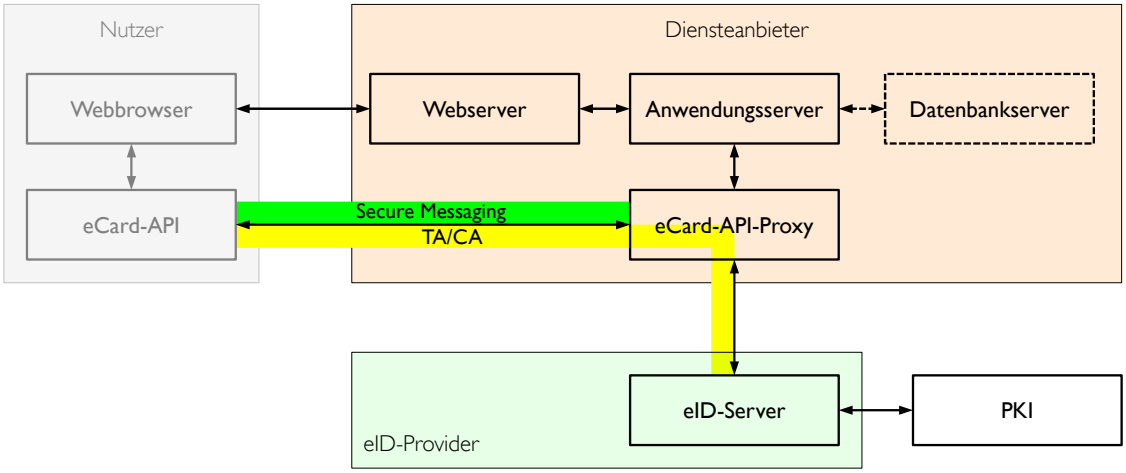

Abbildung 3.9.: Szenario mit eCard-API-Proxy.

vom eCard-API-Client zurückgeliefert werden, zur weiteren Verarbeitung bzw. Bereitstellung für die Webanwendung. Somit würde sichergestellt, dass nur der (dezentrale) Diensteanbieter die für ihn bestimmten Daten „zu Gesicht bekommt“.

3.3.8. Fazit

Die vorgestellten Szenarien und Abläufe zeigen, dass die eID-Schnittstellen und die eCard-API flexibel genug sind, den Betrieb eines notwendigen eID-Servers an spezialisierte eID-Provider auszulagern.

Vor allem der Betrieb vorgeschriebener Komponenten, wie eines HSM, stellt für viele Diensteanbieter (z.B. Online-Shops) eine Hürde dar, wenn die Anwendung in einem Rechenzentrum betrieben wird. Dort ist in der Regel die Installation von kundenspezifischer Hardware nicht möglich.

Die Spezifikation des eID-Servers durch das BSI hat den Vorteil, dass entsprechende Bibliotheken zur Anbindung eines „TR-03130-konformen“ eID-Service für den neuen Personalausweis existieren werden und die Anbindung an einen Diensteanbieter erheblich vereinfacht wird.

Das Fallbeispiel „eID-Server der Bundesdruckerei“ zeigt einen anderen Ansatz: die Nutzung eines standardisierten Protokolls (SAML) zum Transport der Identitätsdaten über ein unsicheres Medium (Nutzer-Rechner im Internet) und – bis auf Aufgaben zur Mandantenverwaltung – keine direkte Kommunikation zwischen Diensteanbieter und eID-Provider. In diesem Szenario wird erheblich Last von der Webanwendung beim Diensteanbieter genommen, da ständige Anfragen (Polling) nach dem Status der eID-Sitzung nicht notwendig sind. Stattdessen wird ein ereignisgesteuerter Ansatz per Callback realisiert.

Bei der Spezifikation des eID-Servers hat das BSI aber leider auch die Chance vertan, zentrale Stellen zu vermeiden, an denen Daten ausgelesen und prinzipiell technisch nachverfolgt werden können, auch wenn dies durch Policies verboten ist.

4. Entwurf eines Benutzerverwaltungsdienstes

Nachdem im vorherigen Kapitel der neue Personalausweis und eID-Infrastrukturen vorgestellt wurden, wird nun der Entwurf eines Benutzerverwaltungsdienstes, der auf diesen Komponenten aufbaut, beschrieben.

4.1. Zielstellung

Viele bisherige Systeme zur Verwaltung von Benutzerkonten (engl. Accounts) haben einen entscheidenden Nachteil, der in der Praxis zu hohen Supportkosten oder Sicherheitsrisiken führt: Es fehlt ein starker Authentifikator, auf dessen Grundlage das eigene Benutzerkonto eigenständig verwaltet werden kann. Einerseits kann ein Benutzerkonto ausschließlich durch autorisiertes Personal (den *Help Desk*) verwaltet werden, das dafür aber Zugriff auf viele Accounts hat. Anderseits existieren benutzerzentrische Lösungen, die es beispielsweise erlauben, ein neues Passwort zu erhalten. Dazu wird entweder das Problem verlagert, indem ein authentisierter Kanal (z. B. über ein Einmalpasswort per E-Mail) vorausgesetzt wird, oder es werden schwache Authentifikatoren zum Zugriff auf das Benutzerkonto eingesetzt. Alle genannten Lösungen sind unbefriedigend.

In dieser Arbeit wird ein Dienst vorgestellt, der die Pseudonymfunktion des neuen Personalausweises als starken Zwei-Faktor-Authentifikator benutzt. Auf dieser Grundlage kann der Benutzer ein entscheidendes Attribut des eigenen Accounts verwalten – das Passwort für den alltäglichen (Rechner-)Login.

Zudem soll untersucht werden, ob sogar neue Benutzerkonten mit Hilfe des Dienstes erzeugt werden können.

Dafür soll ein Benutzerverwaltungsdienst prototypisch für das Fallbeispiel *Benutzerkontenverwaltung am Institut für Informatik der Humboldt-Universität zu Berlin (HUB)* entwickelt werden, der die prinzipielle Machbarkeit darlegen und Probleme verdeutlichen soll. Auf die Implementierungsdetails wird im anschließenden Kapitel eingegangen.

4.2. Fragestellungen

- Welche Voraussetzungen müssen erfüllt sein, um die komplette Verwaltung von Benutzerkonten auf Basis des neuen Personalausweises zu ermöglichen?
  Die komplette Verwaltung umfasst die Erzeugung eines Accounts sowie das Ändern, Verlängern und Löschen bestehender Accounts.
- Wie kann ein neuer Personalausweis mit einem bestehenden Account verknüpft werden, wenn der bislang verknüpfte Ausweis verloren wird oder abgelaufen ist?
- Welche Maßnahmen sind zu ergreifen, um die zu speichernden personenbezogenen Daten vor Missbrauch zu schützen?

4.3. Identititätsmanagement

Der Begriff *Identititätsmanagement* (IdM – engl. identity management), teilweise auch *Identity and Access Management* (IAM), bezeichnet eine Reihe von Maßnahmen die notwendig sind, um die Benutzer eines IT-System zu identifizieren und sicherzustellen, dass diese nur unbedingt erforderliche Zugriffe auf das System und seine Ressourcen erhalten, vgl. [MA07], S. 10.

Im einfachsten Fall sind dies Maßnahmen zum

- Anlegen,
- Ändern und
- Löschen

von Benutzerkonten. Für die Verwaltung eines autarken Systems reichen diese Maßnahmen oftmals aus. Sobald jedoch die Interaktion mit anderen Systemen notwendig ist oder viele solcher Systeme parallel betrieben werden, bedarf es einer Lösung, die Abläufe abstrakter fasst und für eine Synchronisierung der beteiligten System sorgt.

Umfangreichere Identitätsmanagement-Systeme erfüllen daher eine Reihe von Aufgaben, die im Zusammenhang mit Nutzung von IT-Systemen durch ihre Benutzer anstehen. In [MA07] wird ein Ebenenmodell zur Beschreibung der Identitätsmanagement-Themen präsentiert, das die Ebenen *Personendaten*, *Ressourcen*, *Autorisierung* und *Authentisierung* benutzt. Diese Ebenen werden durch die Abbildung auf *Richtlinien*, *Prozesse* und *Technik* verfeinert, siehe Abbildung 4.1.

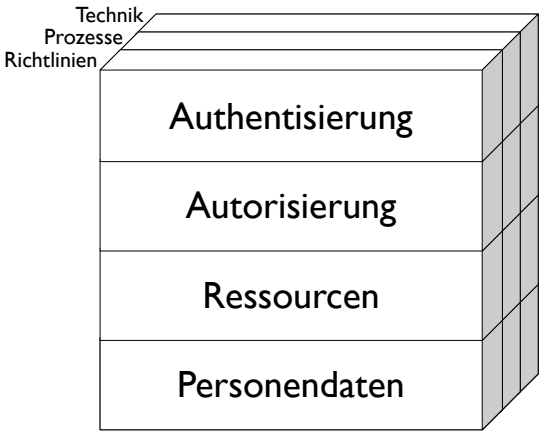

Abbildung 4.1.: Ein Ebenen-Modell für das Identitätsmanagement (Quelle: [MA07])

Zur Veranschaulichung soll das Modell anhand des Ist-Zustands der Benutzerverwaltung am Institut für Informatik der Humboldt-Universität zu Berlin erläutert werden.

4.3.1. Ist-Zustand

Die Personendaten-Ebene bildet die Basis aller Ebenen. Hier werden Daten der Personen gesammelt und für die IT-Systeme der Organisation bereit gestellt. In Firmen werden die

Daten von der Personalabteilung erfasst, aktualisiert und in geeigneter Form (z. B. als Datenbank) anderen Abteilungen übergeben. Die Universitätsverwaltung betreibt ebenfalls eine solche Datenbank, die aber primär für Belange des Studiums gedacht ist (Immatrikulation, Rückmeldung, Rechnungsstellung, Versand von Informationsschreiben, etc.). Aus Datenschutzgründen bekommt die Rechnerbetriebsgruppe[1] des Instituts für Informatik nur eine Liste von Matrikelnummern der immatrikulierten Studenten mit Informatik als Hauptfach, auf deren Grundlage sie eigene Personendaten zur Erfüllung ihrer Aufgaben erhebt und überwacht. Beispielsweise muss am Anfang eines jeden Semesters geprüft werden, welche Studenten die Universität verlassen haben und welche Benutzerkonten somit zunächst zu deaktivieren und nach einer bestimmten Frist zu löschen sind. Die neuen Studenten am Institut erfassen ihre Personendaten selbst. Dazu geben sie zur Beantragung eines Institut-Accounts die benötigten Daten in ein Webformular ein. Ein typischer Verwaltungs-Prozess (nach [MA07] ein „operationaler Prozess“) auf dieser Ebene ist das Ändern des Attributs „Nachname“ einer Person, z. B. nach einer Eheschließung. Darauf aufbauend werden dann u.U. Prozesse in den anderen Ebenen beeinflusst. So existiert für jeden Benutzer ein E-Mail-Alias[2] der Form „vorname.nachname@informatik.hu-berlin.de“, der ebenfalls zu berücksichtigen ist.

Die zweite Ebene befasst sich mit der Verwaltung der Ressourcen, die sich in Systeme und Daten aufteilen. Unter anderem werden hier die Benutzerkonten (inklusive Benutzerkennzeichen) erzeugt, die neue Studenten über das Webformular beantragen. Benutzerkonten sind „Träger“ von Berechtigungen für den Zugriff auf die Ressourcen und haben am Institut jedoch zunächst nur eine zeitlich befristete Gültigkeit, bis der Student seine Identität bei einem Mitarbeiter der Rechnerbetriebsgruppe hat bestätigen lassen. Dazu ist es erforderlich, dass der Student mittels Ausweisdokument persönlich vor Ort erscheint. Ebenfalls bei der Erzeugung wird dem Account eine primäre *Benutzergruppe* zugewiesen, die als Hilfskonstrukt (bei der Autorisierung in Ebene 3) dienen kann. Üblicherweise ist dies die Gruppe der Studienanfänger eines Jahrgangs. Benutzergruppen fassen Eigenschaften von Benutzerkonten zusammen und erleichtern die Zuweisung von Berechtigungen, da diese nicht für jeden einzelnen Benutzer, sondern stellvertretend für die Gruppe erfolgen kann. Der Vorteil von Identitätsmanagement-Systemen zeigt sich, wenn mehrere unterschiedliche Kontensysteme betrieben werden. So muss der Administrator ohne ein IdM bei Änderungen daran denken, alle Systeme zu aktualisieren, und ggf. „von Hand“ die benötigten Accounts anlegen, ändern oder löschen. Bei Nutzung eines IdM werden diese Prozesse dagegen in einem abstrakten Modell definiert. Änderungen können dann automatisch erfolgen.

In der Autorisierungs-Ebene werden Berechtigungen für den Zugriff auf die Ressourcen (in Ebene 2) erteilt. Abstrakt kann der Administrator dafür ein Rollen-Modell benutzen. „Eine Rolle ist eine Sammlung von Berechtigungen, die notwendig sind, um eine bestimmte Aufgabe zu erfüllen“ ([MA07], S. 16). Entscheidend ist dabei, dass ein Benutzer immer nur eine Rolle zur Erfüllung der Aufgabe erhält und bei einer neuen Aufgabe die alte Rolle und somit die Berechtigungen abgibt, um eine neue Rolle zu erhalten. Praktisch kann dies

[1] Die Rechnerbetriebsgruppe (RBG) sorgt – wie der Name es andeutet – für den reibunglosen Betrieb des Rechnerbetriebs am Institut. Analog zu einem „kleinen Rechenzentrum“ stellt sie Dienste für die Studenten und Mitarbeiter bereit und koordiniert die Anbindung von IT-Systemen an den Lehrstühlen.

[2] Eine E-Mail an die „Alias-Adresse“ des Benutzers wird an dasselbe Postfach des Benutzers zugestellt wie bei einer E-Mail an die primäre E-Mail-Adresse.

durch die Nutzung unterschiedlicher digitaler Identitäten (aus Ebene 2) umgesetzt werden. Dies spiegelt sich beispielsweise in der Richtlinie wider, dass Administratoren einen privilegierten Account nur zur Systemadministration nutzen sollen. Das Rollenmodell ermöglicht zudem durch die abstrakte Beschreibung der Berechtigungen eine Harmonisierung der systemspezifischen Abbildungen (z. B. durch Benutzerkonten und Gruppen) unterschiedlicher Systeme und eine transparente Überprüfung der Rollen zu Berechtigungen. Beispielsweise kann durch eine Abbildung der Rolle „Student“ in das entsprechende Modell der institutseigenen Veranstaltungsverwaltungssoftware „GOYA“[3] sicherstellen, dass kein Student die GOYA-Rolle „Dozent“ mit zusätzlichen Berechtigungen erhält.

Die Authentisierungs-Ebene schließlich legt fest, auf welche Art und Weise ein Benutzer des Systems Zugriff erlangen kann. Generell bedarf es für die Nutzung von IT-Systemen des Instituts eines Benutzerkontos, da hiermit die Anerkennung der Benutzungsordnung und eine Zuordnung von Daten und Berechtigungen zu einem Benutzer sichergestellt ist. Eine andere Richtlinie und Technik dieser Modellebene betreffen den Webserver des Instituts: So ist der Zugriff auf bestimmte Teilbereiche (engl. locations) von außerhalb des Universitätsnetzes passwortgeschützt. In einem weiteren Beispiel besagt eine Richtlinie, dass Schlüsselkarten für die elektronische Schließanlage nur an Studenten mit Informatik-Account ausgegeben werden, die „[...] mindestens ein Semester erfolgreich an der HU studiert haben [...].“[4] Dies zeigt zudem die Abhängigkeit zur dritten Ebene (Autorisierung): Die Studenten ab dem zweiten Semester kann man als Rolle „Studenten mit Schlüsselkarten-Berechtigung“ auffassen.

4.3.2. Speicherung der Personendaten

Ein Fokus dieses Kapitels liegt auf der Speicherung von Personendaten beim Identitätsmanagement. Allgemein werden diese Speichersysteme unter dem Begriff *Repository* zusammengefasst. Sehr weit verbreitete Technologien dieser Kategorie sind *Verzeichnisdienste* und *Relationale Datenbanksysteme*.

Verzeichnisdienste stellen strukturierte Daten an zentraler Stelle in einer Organisation über ein Netzwerk zur Verfügung. Das zugrunde liegende Verzeichnis kann im einfachsten Fall als Liste realisiert sein, z. B. ein Telefon-Verzeichnis bzw. Telefonbuch. Hierarchische Verzeichnisse hingegen erlauben aufgrund ihrer Baumstruktur, die als *Directory Information Tree* (DIT) bezeichnet wird, die Abbildung von Beziehungen und Abhängigkeiten.

Das *Lightweight Directory Access Protocol* (LDAP) ist das heutzutage vorrangig verwendete Protokoll für den Zugriff auf ein X.500-konformes Verzeichnis und wurde in [RFC4511] standardisiert. Am Institut für Informatik sind es ebenfalls LDAP-Server, die für die Verwaltung von Daten für Benutzerkonten eingesetzt werden. Darauf wird in Abschnitt 4.6.3 nochmals eingegangen. Ein LDAP-Server ermöglich dabei folgende Operationen, vgl. [RFC4511], Kapitel 4:

- bind/unbind – Verbindungsaufbau und -abbau zu einem LDAP-Server (inkl. Authentifikation);
- search – Suchen nach einem Objekt gemäß eines angegebenen Suchfilters;

[3] https://goya3.informatik.hu-berlin.de/goyacs.
[4] Vgl. http://www2.informatik.hu-berlin.de/rbg/skarte.shtml.

- add/delete/modify – Hinzufügen, Löschen und Ändern der Attribute eines Objekts;
- compare – Vergleich eines Attribut-/Wert-Paares mit den korrespondierenden Attributen des im DIT gespeicherten Objekts (DN);
- abandon – Abbruch einer Anfrage, die zuvor ausgeführt, aber noch nicht beendet wurde;
- modify RDN – ermöglicht Verschieben von Objekten (Knoten) und Teilbäumen eines DIT.

In einem Verzeichnis nach X.500 können die Objekte über einen absoluten Namen, den *Distinguished Name* (DN), oder über den (zum Elternobjekt) relativen Namen *Relative Distinguished Name* (RDN) adressiert werden. Die Daten im Verzeichnis werden auf Basis sogenannter Schemata strukturiert. Ein Schema definiert unter anderem die Datentypen von Attributen und die Verkettung von Attributtypen in Objektklassen. Analog zur objektorientierten Programmierung besitzen alle Instanzen einer solchen Klasse die definierten Attribute, die verpflichtend oder optional sein können, und es können Klassen im Zuge der Vererbung voneinander abgeleitet sein. (Zum Beispiel ist der Nachname einer Person zwingend, die Telefonnummer hingegen nicht.)

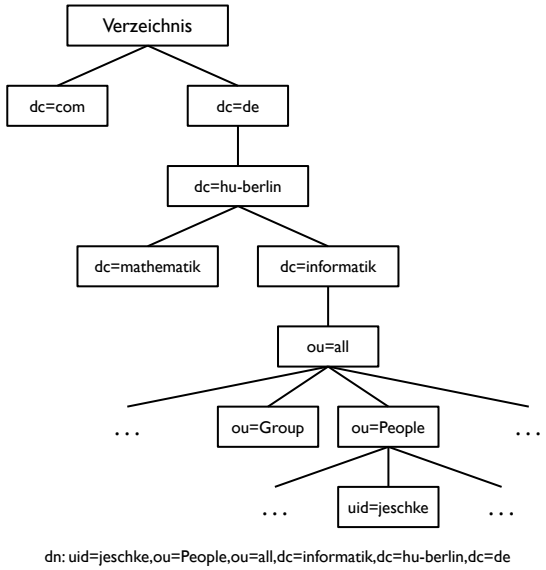

Abbildung 4.2.: Benutzer-Objekt, das den Autor im LDAP-Verzeichnis des Instituts repräsentiert

Weiterhin ist es für Attributtypen möglich, das Verhalten bzgl. eines Suchschlüssels bei Suchabfragen („EQUALITY“) festzulegen. Die EQUALITY-Regel „telephoneNumberMatch“ ermöglicht den Vergleich von Telefonnummern auch dann, wenn diese optische Gruppierungszeichen wie „-“ und Leerzeichen enthalten, der Referenzwert aber nicht und umgekehrt. Verzeichnisdienste sind für den lesenden Zugriff optimiert.

Im Gegensatz dazu sind Daten in einem relationalen Datenbanksystem sehr viel einfacher strukturiert. Die Objekte der Relation (auch Datensätze genannt) sind Tupel, die aus einer Kombination von Attributwerten (die Attributnamen zugeordnet sind) bestehen. Anschaulich beschreibt eine Relation somit eine Tabelle in der Datenbank. Attribute sind in der Regel typisiert (Integer, String, Boolean, etc.). Diese einfache Form zusammen mit einer Abfragesprache, wie der Structured Query Language (SQL), die es ermöglicht, Daten flexibel zu verknüpfen (über die Operationen SELECT, JOIN, UNION, Subqueries und Aggregationen), macht Relationale Datenbanksysteme attraktiv für Anwendungen, in denen große Mengen gleichartiger Daten anfallen. So werden diese Systeme gern dann zur Personen- und Account-Datenverwaltung eingesetzt, wenn das Datenbanksystem sowieso bereits vorhanden ist und nur wenige Anwendungen auf den Datenbestand auf-

setzen. LDAP-Verzeichnisse zielen vor allem auf solche Szenarien ab, in denen viele unterschiedliche Systeme Zugriff auf Benutzerdaten benötigen. Dies lässt sich beispielsweise daran ablesen, dass fast alle Produkte, die eine Authentifizierung durchführen, eine LDAP-Schnittstelle bereitstellen.

Diese Arbeit beschränkt sich im Weiteren auf die Betrachtung der Verwaltung von Personendaten und darauf aufbauend den Benutzerkonten (Ebene 1 und 2 im vorgestellten Modell). Insbesondere wird nicht auf die Verwaltung von Zugriffsrechten eingegangen. Im Prototyp wird die Funktion des Identitätsmanagements durch die betrachtete Webanwendung umgesetzt.

Hat ein Benutzer sein Passwort für den „Infomatik-Account“ vergessen, bleibt ihm bislang nur der Weg zur Rechnerbetriebsgruppe, die nach Vorlage von Studentenausweis und einem Dokument mit Lichtbild (Pass, Personalausweis oder Führerschein) ein neues Passwort vergibt. Der Computer- und Medienservice (CMS – vormals Rechenzentrum) der Humboldt-Universität hat für dieses Problem eine andere Lösung vorgesehen. Nach der Immatrikulation bzw. Rückmeldung erhalten alle Studenten zusammen mit dem Studierendenausweis die sogenannte *Studienbuchseite*, auf der eine personalisierte 4-stellige PIN aufgedruckt ist. Diese kann – da über den unabhängigen Kanal „Briefzustellung“ übermittelt – zum Setzen eines neuen Passworts über eine dafür eingerichtete Webanwendung des CMS benutzt werden. Weiterhin steht aber auch die Nutzung des Help Desks zur Verfügung. Die „PIN“ entspricht funktional somit einem *Passwort Unblocking Key* (PUK).

4.4. Prozesse

Aus der Zielstellung ergeben sich unmittelbar drei Prozesse, die der Dienst zur Realisierung umsetzen muss. Diese sollen im Folgenden umrissen werden.

4.4.1. Zurücksetzen des Account-Passworts

Die zentrale Funktion des Dienstes, die von der eID-Anwendung des neuen Personalausweises am meisten profitiert und einen echten Mehrwert für den Benutzer bringt, ist das Zurücksetzen des Account-Passworts.[5]

Hat ein Benutzer sein Passwort vergessen, ist es abgelaufen bzw. wurde das Passwort wegen mehrfacher Falscheingabe gesperrt (je nach Richtlinie der Organisation, die den Account bereitstellt), kann der Benutzer seinen Account durch eine Authentisierung mit dem nPA und Wahl eines neuen Passworts reaktivieren. Dazu wird ausschließlich die Pseudonymfunktion der eID-Anwendung zum Auslesen der Restricted ID (rID) des „ausgesperrten“ Benutzers genutzt. Nachdem diese (pro Nutzer und Dienst) eindeutige Kennung erfolgreich bestimmt werden konnte, sucht der Dienst in seinem Benutzer-Repository (Verzeichnis, Datenbank, etc.) nach dem Benutzerkonto, dem diese Kennung zu einem früheren Zeitpunkt zugeordnet wurde. War die Suche erfolgreich, setzt der Dienst das neue Passwort für den wiedererkannten Benutzer.

[5] Die (automatische) Erfassung der personenbezogenen Daten aus einem Personalausweis ist aus organisatorischen und Sicherheitsgründen zu begrüßen, würde aber mehr dem Rechnerbetrieb denn dem Benutzer helfen.

Der Vorteil liegt darin, dass dies zum einen auch entfernt (über das Internet) funktioniert und zum anderen ein starker Zwei-Faktor-Authentifikator – der neue Personalausweis und die eID-PIN – zum Einsatz kommt. Somit muss der Benutzer nicht mehr persönlich erscheinen. Ferner kann der „ausgesperrte“ Benutzer sein Passwort auch außerhalb der Bürozeiten des Help Desks zurücksetzen.

Diese Funktion setzt voraus, dass zuvor bereits eine Zuordnung von nPA zu Account durchgeführt wurde.

4.4.2. Verknüpfen eines bestehenden Accounts mit einem neuen Ausweis

Aus dem ersten Prozess ergibt sich implizit ein weiterer: Es sollte möglich sein, einem bestehenden Account einen bislang nicht im System registrierten nPA zuzuweisen. Dies ist bei jenen Benutzerkonten notwendig, die vor der Ausstellung von neuen Personalausweisen (am 1. November 2010) erzeugt wurden bzw. deren Inhaber bislang nicht über einen nPA verfügen und einen solchen zukünftig nutzen möchten. Auch ist das Verfahren bei Verlust oder nach Ablauf eines nPA und für die Bekanntmachung der Restricted ID (rID) des neuen Ausweises notwendig. Zur Erinnerung: Die rID ist pro *Ausweis* und Dienst eindeutig, nicht pro Ausweis-Inhaber und Dienst.

Die Forderung nach einem starken Zwei-Faktor-Authentifikator – der gerade ein Vorteil gegenüber anderen Lösungen, wie die in Abschnitt 4.3.2 beschriebene PIN-Nutzung beim CMS, ist – führt an dieser Stelle zu einem Problem: Wie kann mit den zur Verfügung stehenden Mitteln (gültiges Passwort, PIN der Studienbuchseite, etc.) die rID des neuen Ausweises hinterlegt werden, ohne damit das Sicherheitsniveau des nPA als Authentifikator zu gefährden und auf das Niveau der anderen Mittel zu reduzieren.

Ein Bedrohungsszenario soll das Problem veranschaulichen. Angenommen die Registrierung eines nPA und die Zuordnung zu einem bestehenden Account wäre nur von der Kenntnis des Account-Passworts abhängig („Login mit dem Passwort am Benutzerverwaltungsdienst“). Unabhängig davon, ob dieser Account bereits einem nPA zugeordnet ist oder nicht, könnte ein Angreifer, der ebenfalls im Besitz des Passworts ist, einen in seinem Besitz befindlichen nPA[6] mit dem Account des Opfers verknüpfen. Eine Rückmeldung dieser Transaktion, beispielsweise per E-Mail, hälfe wenig, da der Angreifer diese unterdrücken könnte. Wenn nun das Opfer entsprechend der Passwortrichtlinien das Account-Passwort in den vorgeschrieben Intervallen ändert, berührt das die rID im Benutzer-Repository nicht. Der Angreifer könnte jederzeit ein eigenes Passwort mit der Zurücksetzfunktion des Dienstes etablieren, solange das Opfer nicht den eigenen Personalausweis mit dem Account verknüpft. Da das Passwort des Angreifers jedoch voraussichtlich vom originalen Passwort des Opfers abweicht, würde der Angriff beim nächsten Login des Opfers auffallen. Es ist aber davon auszugehen, dass dann ein Missbrauch der Identität bereits stattgefunden hat.

Alternativ könnte nach einem Login mit dem Account-Passwort als Transaktionslegitimation ein Einmalpasswort bzw. eine TAN zur Autorisierung des Benutzers für diese Funktion des Dienstes zum Einsatz kommen. Auf diese Art würde eine Zwei-Faktor-Authentifizierung (Wissen und Wissen) benutzt, die durch das Einmalpasswort ein fast so

[6] Hierfür könnte der Angreifer theoretisch sogar seinen eigenen nPA benutzen. Da nur die Restricted ID vom Dienst gelesen wird, wäre die Identität des Angreifers hinter dem Pseudonym „geschützt“.

hohes Niveau wie Besitz und Wissen hätte: Ein Kopieren und Benutzen des Einmalpasswortes würde durch den legitimen Nutzer – analog zum Abhandenkommen des Besitzes – erkannt werden, wenn dieser versucht, seinen nPA zu registrieren. Das Einmalpasswort sollte nur persönlich nach entsprechender Identifizierung durch einen Mitarbeiter des Help Desks übergeben werden, z. B. im Zuge der oben beschriebenen Accountfreischaltung am Institut. Möglicherweise reicht es aber auch aus, einen Vergleich von Vorname und Nachname des Ausweisinhabers mit denen des Benutzers zusammen mit einer Passwortprüfung durchzuführen, um einen Angriff einzudämmen.

4.4.3. Anlegen eines neuen Accounts

Eine wünschenswerte Funktion eines Benutzerverwaltungsdienstes ist das eigenständige Anlegen eines Accounts. Wichtig dabei ist – analog zur oben beschriebenen Webanwendung des Instituts für Informatik, dass der Benutzer mit dem Account zunächst nur eine *Basisrolle* bekommt (vgl. [MA07], S. 46), d. h., er erhält automatisch und initial nur die Rechte, die prinzipiell jedem Benutzer des IT-Systems zugesichert werden. Eine solche Basisrolle könnte beispielsweise „Student“ oder „Angestellter“ lauten. Weitergehende Berechtigungen müssen auf herkömmlichem Wege durch die zuständigen Verantwortlichen zugewiesen werden.

Durch den Dienst muss zudem sichergestellt werden, dass nach einer Zuweisung *nPA* $\rightarrow$ *Benutzerkonto* keine weitere Zuweisung möglich ist, um einen Missbrauch durch Anlegen beliebig vieler Accounts zu verhindern.

Im Gegensatz zu den anderen Funktionen werden hierfür – die Erteilung eines entsprechenden Berechtigungszertifikats vorausgesetzt – neben der rID auch andere personenbezogene Daten ausgelesen. Zur Nutzung von einigen Diensten (z.B. E-Mail) und zur Erzeugung eines Benutzerkennzeichens ist in jedem Fall der Vor- und Zuname des neuen Benutzers notwendig. Zur Ahndung von Verstößen gegen die Benutzerordnung[7] und wegen möglicher Auskunftsersuchen von Sicherheitsbehörden[8] muss zudem die Anschrift erhoben werden. Darüber hinaus müssen „Studierende anderer Fachrichtungen, die Lehrveranstaltungen [...] besuchen“ die Erfordernis eines Benutzerkennzeichens nachweisen.[9] Dazu ist einem (Web-)Formular das Haupt- und Nebenfach sowie ggf. der Dozent der Veranstaltung anzugeben, für deren Besuch ein Informatik-Account vorausgesetzt wird.

Grundsätzlich besteht für die Realisierung dieser Funktion ein Problem: Wie kann sichergestellt werden, dass die Zuordnung *nPA* $\rightarrow$ *Neues Benutzerkonto* nur für Angehörige der Organisation möglich ist? Hierfür stehen unter anderem folgende Maßnahmen zur Verfügung:

1. Die Beschränkung der Verfügbarkeit des Dienstes auf einen Standort bzw. das lokale Netzwerk. Diese Variante empfiehlt sich tendenziell für Firmen, da hier die neuen Mitarbeiter in der Regel vor Ort sind. Im universitären Umfeld könnte dies jedoch den

[7] Gemäß Benutzungsordnung des Instituts für Informatik der HU Berlin, §3, Absatz 6.

[8] Das Institut bzw. die Universität tritt als Internetdienstanbieter (engl. internet service provider, ISP) auf und muss gemäß §§ 113 TKG, Abs. 1 in Verbindung mit §§ 111 TKG den Namen und die Anschrift des Benutzers sowie das „Datum des Vertragsbeginns“ erheben, speichern und den im Gesetz genannten „zuständigen Stellen auf deren Verlangen unverzüglich Auskünfte über die [...] erhobenen Daten [...] erteilen.“

[9] Vgl. `http://www2.informatik.hu-berlin.de/rbg/account.shtml`.

erwünschten Effekt (Entlastung der Benutzerbetreuung) verhindern, da neue Studenten solche administrativen Aufgaben noch vor Studienbeginn erledigen werden und dafür ein Zugriff von zu Hause wünschenswert ist.

2. Die Ausweisdaten werden mit „Stammdaten“ abgeglichen. Existiert eine Person im Repository ohne Benutzerkonto und stimmen die Daten aus der Immatrikulation bei Studenten bzw. der Einstellung bei Mitarbeitern (Name, Vorname, Geburtsdatum, Anschrift) mit den Ausweisdaten überein, wird der Account angelegt und mit dem nPA verknüpft. Dazu bedarf es aber intelligenter Matching-Methoden, wie dem Fuzzy-Matching oder einer Menge von regulären Ausdrücken, um eine Übereinstimmung auch bei kleinen Abweichungen zu ermöglichen.

3. Die Wiedererkennung eines Angehörigen mittels eines zuvor vereinbarten Geheimnisses. So könnte, wie in den vorgestellten Anwendungen beim Informatik-Institut und CMS, eine Zuordnung anhand der Matrikelnummer oder PIN erfolgen.

4. Ableitung eines neuen nPA-basierten Accounts von einem bestehenden Account einer anderen Abteilung der Organisation. Beispielsweise kann ein neuer Informatik-Account auch nach Verifizierung der Credentials eines CMS-Accounts angelegt werden.[10] Die Bindung des Benutzers an die Organisation ist in dem Fall schon bei der anderen Abteilung erfolgt.

5. Alternativ könnte bei einer *elektronischen Immatrikulation* auf Basis des nPA die Restricted ID des Studenten in der Studenten-Datenbank hinterlegt werden:
Ein elektronisches (Web-)Formular wird mit den personenbezogenen Daten der eID-Anwendung ausgefüllt (Name, Vorname, Anschrift, Geburtsdatum)[11], und zusätzliche Daten können durch den zukünftigen Studenten ergänzt werden. Im Anschluss druckt dieser eine Bestätigung aus, welche eine eindeutige Nummer zur späteren Zuordnung enthält, und schickt diese unterschrieben mit den notwendigen Anlagen zum Immatrikulationsbüro. Einrichtungsspezifische Accounts könnten dann auf Basis dieser rID/Stammdaten angelegt werden.

4.5. Entwurfsentscheidungen

Um aus den funktionellen Anforderungen, die wie dargestellt noch Optionen anbieten, einen Prototyp erstellen zu können, werden folgende Entscheidungen für den Entwurf der Architektur und die spätere Implementierung getroffen:

- Da der eID-Service der Bundesdruckerei zusammen mit der AusweisApp für den Zugriff auf einen nPA genutzt werden soll, wird der Benutzerverwaltungsdienst als Webanwendung umgesetzt. Die eID-Kommunikation erfolgt wie im eID-Szenario II vorgestellt, siehe auch: Abschnitt 3.3.5.

[10] Diese Variante wird zur Vereinfachung der Prozesse zukünftig ausschließlich zur Erzeugung von Informatik-Accounts verwendet werden.

[11] Diese Daten werden bei einer Immatrikulation generell erhoben, so dass keine datenschutzrechtlichen Bedenken zu erwarten sind.

- Trotz dieser Festlegung bzgl. des eID-Providers soll die Tatsache berücksichtigt werden, dass der Betreiber später eventuell einen anderen eID-Provider nutzen möchte, der sich in der eID-Schnittstelle von jener der Bundesdruckerei unterscheidet. Das heißt, die Software soll prinzipiell ohne große Anpassungen unterschiedliche eID-Server anbinden können.

- Im Prototyp soll der eigentlichen Webanwendung ein Webserver vorgeschaltet[12] werden, der eine flexible Handhabung der Anfragen (engl. request) und eine Zugriffsteuerung ermöglicht. Der Webserver soll somit den Funktionsumfang der HTTP-Schnittstelle des Anwendungsservers erweitern. Beispielsweise kann es nötig sein, dass einzelne URLs auf der Seite des Diensteanbieters umgeschrieben werden müssen. Da eine solche Anpassung nur innerhalb des SSL/TLS-Kanals erfolgen kann, muss der Webserver ebenfalls die Terminierung der HTTPS-Verbindung übernehmen. Zwischen Webserver und Anwendungsserver kann je nach Absicherung des Kommunikationskanals zwischen beiden Komponenten eine ungesicherte HTTP-Verbindung benutzt werden, beispielsweise weil beide Server auf dem gleichen Host ausgeführt werden.

- Die bestehende Benutzerverwaltung soll durch den in dieser Arbeit beschriebenen Dienst zunächst erweitert werden.

  Dazu ist es nötig, das gleiche Repository (ein LDAP-Verzeichnis) zur Speicherung der Benutzerdaten anzubinden. Für den Prototyp kommt ein eigener LDAP-Server zum Einsatz, der die gleichen Schemata und Aufbau wie das reale System benutzt. Der „Test-DIT“ enthält nur den Teilbaum des realen DITs, der die Personendaten betrifft (dn: ou=People, ...). So kann zudem sichergestellt werden, dass eventuelle Probleme der Abbildung von den eID-Daten auf die LDAP-Struktur spätestens bei der Implementierung erkannt werden können.

- Zur Vereinfachung der Implementierung kann nur ein Benutzerkonto (das „primäre“) mit dem nPA verknüpft werden. Andere Implementierungen könnten über die Abfrage des Loginnamens mehrere Accounts einem nPA zuordnen. In der Praxis dürfte diese Anforderung aber nur für wenige Benutzer, oftmals Angehörige des IT-Supports (zur Realisierung unterschiedlicher Rollen), relevant sein.

- Um die Komplexität handhabbar zu gestalten, soll die Software in Module aufgeteilt werden. Dies ermöglicht – bei Definition von geeigneten Schnittstellen – einen späteren Austausch von einzelnen Modulen, z. B. um den Dienst an geänderte Randbedingungen anzupassen.

4.5.1. Umsetzung der Funktionen im Prototyp

Aus den gezeigten Alternativen werden für den Prototyp folgende Festlegungen getroffen:

1. Für die Erstellung eines Accounts wird eine gültige, nicht zugeordnete Matrikelnummer abgefragt und geprüft, ob diese auf einer Referenzliste und zudem nicht auf einer

[12] Diese Technik wird als *Proxy* bzw. *ReverseProxy* bezeichnet.

schwarzen Liste enthalten ist. Nach der Erstellung des Accounts wird diese Matrikelnummer in die schwarze Liste eingetragen. Im Zuge dessen wird zudem ein Passwort für den neuen Account abgefragt. Der Benutzer erhält im Anschluss ein automatisch generiertes Benutzerkennzeichen.

Bei Unstimmigkeiten (Matrikelnummer nicht gültig, Passwort unterschiedlich, etc.), ebenso wie bei einer bereits registrierten Restricted ID erhält der Benutzer eine Fehlermeldung.

2. Zur Verknüpfung eines neuen Personalausweises mit einem Account soll es zu Demonstrationszwecken – trotz der in Abschnitt 4.4.2 angesprochenen Bedenken – für den Prototyp ausreichen, wenn sich der Benutzer mit seinen Credentials authentisiert und anschließend die Restricted ID seines nPA auslesen lässt.

   Ist die rID schon einmal (auf diesen oder einen anderen Account) zugeordnet worden, erhält der Benutzer eine Fehlermeldung, siehe Screenshot A.10 im Anhang.

4.6. Architektur

Für die Komposition des Dienstes werden eine Reihe von Subsystemen benötigt. Ein Teil der Systeme wurde bereits bei den eID-Szenarien in Abschnitt 3.3 beschrieben.

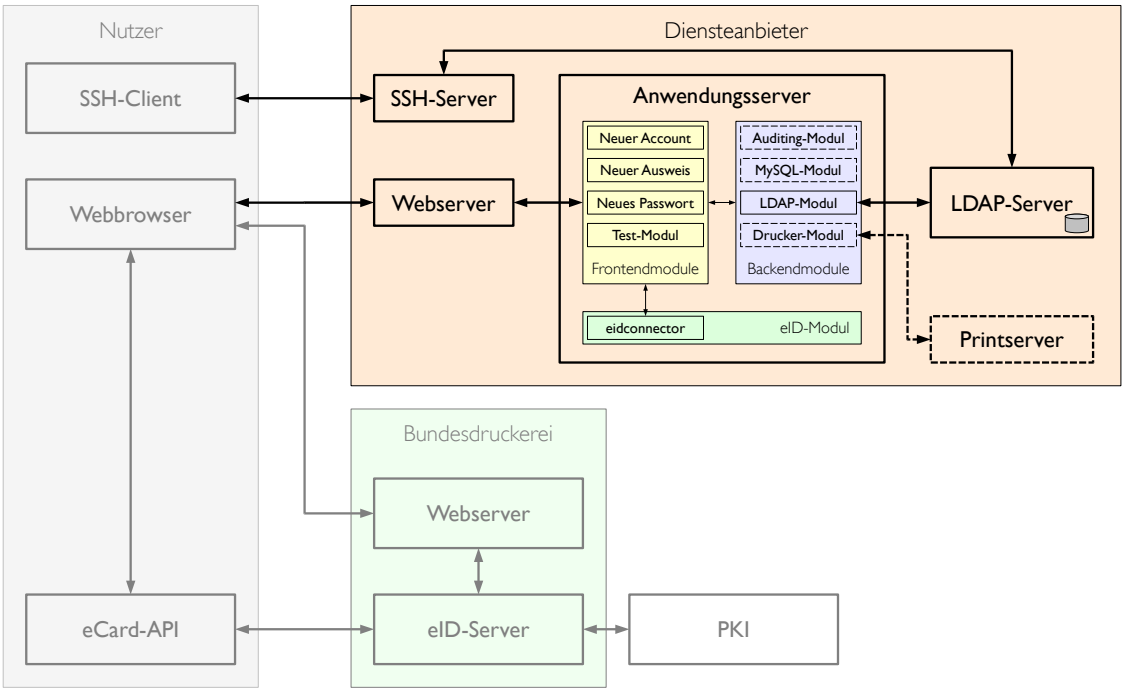

Abbildung 4.3.: Komponenten des Benutzerverwaltungsdienstes

4.6.1. Webserver

Als Schnittstelle zum Benutzer terminiert der Webserver den HTTPS-Kanal zwischen Diensteanbieter und Webbrowser auf der Clientseite. Viele Applikationsserver bringen auch einen eigenen Webserver zur Nutzer-Diensteanbieter-Kommunikation mit. Sofern diese HTTP-Schnittstelle des Anwendungsservers alle vom Diensteanbieter benötigten Funktionen abdeckt, kann die Komponente *Webserver* auch entfallen.

4.6.2. Anwendungsserver

Der Anwendungsserver (oder Applikationsserver) führt die *Dienstanwendung* aus und stellt dafür die Ausführungsumgebung zur Verfügung, z. B. benötigte Interpreter und Software-Bibliotheken, aber auch ein Modul, das per HTTP mit der „Außenwelt“ kommuniziert. Er verbindet sich ferner mit dem eID-Service, um die Ausweisdaten des Benutzers zu erhalten und auf dieser Grundlage den weiteren Ablauf einer Sitzung der Webanwendung zu steuern. Hierfür werden zwei Einsprung-Adressen für die Anwendung eingerichtet:

1. Die Hauptseite der Anwendung: `https://server.example.com/account/`.
2. Die Callback-URL über die der eID-Server die Daten zurückliefert: `https://server.example.com/account/verarbeiteAusweisdaten.py`.

Der Systemadministrator konfiguriert auf diesem Server die notwendigen Schlüssel (Verschlüsselung und Signatur) für die Kommunikation mit dem eID-Server.

Es empfiehlt sich diese Komponente aufgrund der hohen Komplexität weiter zu verfeinern. Eine sinnvolle Modularisierung wird im Abschnitt 4.6.6 präsentiert.

4.6.3. Verzeichnisdienst

Wie in Abschnitt 4.3.2 geschildert, kommt am Institut für Informatik ein LDAP-basierter Verzeichnisdienst zum Einsatz. Dieser soll auch beim Dienst als Repository für die Benutzerdaten und Credentials dienen. Darüber hinaus gibt es zwei Möglichkeiten, wie die für die beschriebenen Prozesse notwendige Restricted ID im Verzeichnis abgelegt werden kann. Im einfachsten Fall benutzt man ein bislang nicht verwendetes Feld der Objektklasse des Standard-Schemas. Nahezu alle LDAP-Implementierungen verwenden zur Abbildung von Personendaten die Objektklassen *person*und darauf aufbauend *inetOrgPerson*. Bei Letzterer böte sich scheinbar das Attribut *jpegPhoto* an, welches generell Binärdaten wie die Restricted ID unterstützt. Ungünstigerweise lässt sich nach diesem Attribut nicht suchen, was aber für den behandelten Dienst notwendig ist. Ein anderes Feld, das eine Suche ermöglicht und – zumindest am Institut für Informatik – nicht benutzt wird, ist das Feld *description*. Da es sich hierbei um ein Attribut für Zeichenketten handelt, muss die Restricted ID mittels Base64[13] kodiert bzw. dekodiert werden. Da das Attribut die EQUALITY „caseIgnoreMatch“

[13] Base64 beschreibt eine Möglichkeit, beliebige Binärdaten (Octets) mit druckbaren Zeichen darzustellen. Dabei werden jeweils drei ursprüngliche Zeichen (zu 8-Bit) in vier druckbare Zeichen (zu 6 Bit), also insgesamt 24 Bit, umgewandelt. Der Name entspricht der Zahlenbasis des Zeichenvorats von $2^6 = 64$ Zeichen. Als Zeichen des Base64-Alphabets werden die großen und kleinen Zeichen des lateinischen Alphabets (52 Zeichen) zzgl. der Ziffern 0 bis 9 sowie „+“ und „/“ benutzt. Das Gleichheitszeichen „=“ dient zum Auffüllen (Padding), wenn gilt: Länge(Binärdaten) mod $3 \neq 0$.

besitzt, ist zudem das Ergebnis durch die Anwendung mit dem Suchfilter zu vergleichen, auch wenn eine Übereinstimmung trotz nicht berücksichtigter Groß-/Kleinschreibung des Base64-Strings unwahrscheinlich erscheinen mag.

Der empfohlene Weg zur Speicherung der Restricted ID ist jedoch das Installieren eines passenden Schemas für die Restricted ID. Alle anderen eID-Daten haben bereits eine Entsprechung in den Standard-Schemata.

4.6.4. eID-Server

Aufgrund des erhebliches Aufwands zum Betrieb eines eigenen eID-Server soll ein eID-Service zum Einsatz kommen. Somit gehört der eID-Server im Falle des Prototyps genau genommen nicht zu den Komponenten des Dienstes. Ein eigener eID-Server wäre aber prinzipiell möglich. Da dieser Dienst noch vor der Ausstellung neuer Personalausweise konzipiert und entwickelt wurde, kamen im Rahmen des offenen Anwendungstests die Dienste der Bundesdruckerei mit eID-Server, „Pseudo-Berechtigungs-PKI“ und den zugehörigen Testausweisen zum Einsatz. Für die Entwicklung von Dienst-Anwendungen stellt die Bundesdruckerei immer noch eine ähnliche Test-Infrastruktur bereit. Es ist daher zu erwarten, dass andere eID-Provider ähnliche Möglichkeiten anbieten werden. Alternativ kann auch „gegen ein Produktiv-System entwickelt werden“, was allerdings von vornherein ein offizielles Berechtigungszertifikat und echte Personalausweise voraussetzt.

4.6.5. Testlogin-System

Für die Demonstration des Prototypen kommt ein zusätzlicher Server, ein SSH-Server, zum Einsatz, der wie die Systeme im operativen Betrieb einen LDAP-Server zur Ermittlung von benötigten Benutzerdaten abfragt. Hierzu zählen unter anderem die Information, ob es sich bei einem Benutzer-Objekt im Verzeichnis um ein Account handelt (posixAccount) und die dafür elementaren Daten, z. B. die numerischen Kennungen für Benutzer und primäre Gruppe (uid und gid), aber auch Informationen zum Heimatverzeichnis und der Login-Shell. Außerdem muss der Testlogin-Server für eine Prüfung des Passworts des Benutzers sorgen. Dazu wird aber nicht die transformierte Form (Hash) des Passworts vom LDAP-Server abgefragt und gegen die Eingabe geprüft, sondern es wird eine `bind`-Operation mit dem korrespondierenden Benutzerobjekt und dem eingegebenen Passwort durchgeführt. So kann auf einfache Weise sichergestellt werden, dass niemand außer dem LDAP-Server Zugriff auf die – wenn auch transformierten – Passwörter erhält.

Im operativen Betrieb wird man in der Regel eine Prüfanwendung statt eines separaten Servers benutzen, der die äquivalenten Abfragen beim LDAP-Server durchführt und das Ergebnis in geeigneter Weise aufbereitet.

4.6.6. Anwendungsmodule

Für die Verfeinerung der Anwendung werden drei Arten von Modulen benutzt: *Frontend-Module* bilden die besprochenen Funktionen auf eine Benutzerschnittstelle ab und steuern den Aufruf der für die Realisierung notwendigen Methoden aus den anderen Modul-Typen. Über eigene URLs lassen sich die jeweiligen Frontendmodule und dementsprechend die einzelnen Funktionen direkt adressieren. Somit kann die Funktionalität des Dien-

stes auch leicht erweitert werden, beispielsweise für die Ausstellung von „WLAN-Zertifikaten“[14] nach einer Authentisierung mit der rID-Funktion. *Backend-Module* liefern Funktionen, die von Frontendmodulen benötigt werden, und führen Aktionen zum Abschluss einer eID-Sitzung aus. Ein Repository-Modul bietet deshalb eine Funktion zum Auffinden eines Accounts anhand einer Benutzerkennung oder einer rID. Das Ändern eines Passworts muss vom Modul in eine entsprechende Operation des Repository („ldapmodify()“) umgesetzt werden. Ein *eID-Modul* kapselt die komplexe Kommunikation mit dem eID-Server. Dazu müssen eventuell providerspezifische Softwarebibliotheken eingebunden und die enthaltenen Methoden in geeigneter Weise aufgerufen werden.

Frontend-Module

Entsprechend der funktionellen Anforderung besitzt die Dienstanwendung die drei Frontendmodule *Neuen Account erstellen*, *Neuen Ausweis verknüpfen* und *Neues Passwort setzen*. Darüber hinaus soll ein weiteres Frontend-Modul zum Testen der eID-Anbindung umgesetzt werden. Somit können Fehler bezüglich dieser Anbindung ausgeschlossen und ausgelesenen Daten gesichtet werden. Dies wird sich vor allem bei der Implementierung als hilfreich erweisen.

Backend-Module

Das wichtigste Backend-Modul ist das *LDAP-Modul* (bzw. ein SQL-Modul, wenn eine SQL-Datenbank als Benutzer-Repository eingesetzt wird). Daneben sind ebenfalls Module wie ein Auditing-Modul oder Druck-Modul sinnvoll, die Daten für die Überwachung eines geordneten Betrieb bereitstellen bzw. im Falle einer Account-Erstellung einen Ausdruck der Benutzerdaten auf Papier, in Analogie zum aktuellen Verfahren, bewerkstelligen.

eID-Modul

Gemäß der benutzten eID-Schnittstelle der Bundesdruckerei über SAML-Nachrichten setzt das eID-Modul (für die Bundesdruckerei) die beiden folgenden Teile der Kommunikation um:

1. Die Erzeugung des (signierten und verschlüsselten) SAML-Requests mit den vom Frontend-Modul geforderten Datengruppen und Funktionen. (z. B. „Lese Vorname, Nachname, Anschrift und Restricted ID aus dem Ausweis des Benutzers!“)

2. Die Verarbeitung der SAML-Response vom eID-Server. Dazu muss die Nachricht entschlüsselt und die Signatur (vom eID-Provider) geprüft werden. Anschließend sind die personenbezogenen Daten in geeigneter Form (beispielsweise als Instanz der Klasse „Person“) der Callback-Methode des Frontendmoduls zu übergeben.

[14] Der Standard IEEE 802.1X erlaubt eine Authentifizierung in Ethernet-Netzwerken, insbesondere die eines WLAN-Clients (Station) unter Nutzung eines X.509-Zertifikats. Der Nachweis über den Besitz des zugehörigen privaten Schlüssels wird mittels Challenge-Response erbracht.

4.7. Datenschutzrechtliche Betrachtungen

Bei der Untersuchung der Fragestellung bzgl. des Datenschutzes der personenbezogenen Daten sollen drei Aspekte erörtert werden:

1. Welche Daten werden von einem Benutzer erhoben und gespeichert?
2. Aus welchem Grund werden die Daten erhoben?
3. Wer kann auf welche Daten zugreifen?

4.7.1. Betroffene Datengruppen

Grundsätzlich muss man bei der Betrachtung, welche Daten vom vorgestellten Dienst bearbeitet werden, zwischen *Stammdaten* und *Bewegungsdaten* unterscheiden.

Stammdaten sind jene, die einen Benutzer unmittelbar von anderen unterscheidbar machen und die sich nur selten bis gar nicht ändern. Im Gegensatz dazu können Bewegungsdaten zunächst einmal nur einer Gruppe von Personen zugeordnet werden und sind erst bei Verknüpfung mit anderen (Stamm-)Daten einem Individuum zuzuschreiben (Beispiel: Accountname $\rightarrow$ Name und Anschrift).

Dennoch bergen Bewegungsdaten eine gewisse Brisanz in sich, da sie Rückschlüsse auf Präferenzen und Eigenschaften von Benutzern zulassen können, die die Intimsphäre der Personen betreffen und somit rechtlich besonders geschützt sind.

Zu den vom Dienst betroffenen Stammdaten gehören:

- Nachname des Benutzers,
- Vorname des Benutzers,
- Anschrift des Benutzers,
- Restricted ID (rID) bzgl. des Dienstes und des Personalausweises des Benutzers.

Auf folgende Bewegungsdaten kann der Dienst zugreifen:

- IP-Adresse des Rechners bzw. Anschlusses der Benutzers (Zugriff auf die Webanwendung),
- Daten, die im Zuge eines Auditings der Anwendung erhoben werden.

Außerhalb des Zugriffs des Diensteanbieters fallen zudem Bewegungsdaten der eID-Sitzung beim eID-Provider an. Diese werden in dieser Arbeit nicht weiter betrachtet. Da das Auditing als optionales Modul angesehen wird, kann durch den Dienst der Umfang der Datenerhebung nicht abgeschätzt werden. (Prinzipiell können natürlich alle erhobenen eID-Daten und Nutzereingaben sowie Netzwerkdaten etc. betroffen sein.)

4.7.2. Datenerhebung und -speicherung

Wie bereits erwähnt müssen aufgrund von rechtlichen Bestimmungen der Nachname, der Vorname und die Anschrift des Benutzers erhoben und gespeichert werden. In der bisherigen Benutzerverwaltung des Instituts für Informatik wird aber nur der Nachname und Vorname elektronisch im LDAP-Server gespeichert. So ist im Falle einer Sicherheitslücke oder Fehlkonfiguration einer der LDAP-Server sicher gestellt, dass Stammdaten, die für den Rechnerbetrieb nicht unmittelbar benötigt werden, nicht in falsche Hände gelangen. Die Anschrift wird nur in Papierform mit entsprechendem physischem Zugriffsschutz und Zugriffsrechten für die Rechnerbetriebsgruppe vorgehalten. Im Zuge einer Umstellung auf den hier vorgeschlagenen Dienst sollte diese Richtlinie beibehalten werden. Beim (automatischen) Anlegen eines Accounts könnte die Anschrift (zusammen mit den anderen Stammdaten) auf einem Drucker ausgegeben werden.

Die Restricted ID wird im Interesse des Benutzers (zur Wiedererkennung) und nur durch seine Mitwirkung sowie durch den Einsatz des Berechtigungszertifikats (und dem korrespondierenden privaten Schlüssel) erhoben und ist daher als unkritisch einzustufen. Sie sollte aus Sicherheitsgründen aber nicht für alle Benutzer im LDAP-Verzeichnis lesbar sein (entsprechend der Richtlinie bzgl. transformierter Passwörter).

4.7.3. Datenzugriff

Auf den Nachnamen und Vornamen der Benutzer haben alle anderen Benutzer am Institut Zugriff, da diese für viele Dienste benötigt werden.

Auf die restlichen Daten darf nur die Rechnerbetriebsgruppe Zugriff haben, was durch entsprechende Datenschutzrichtlinien gewährleistet ist. Hier ist es allerdings ausreichend, wenn zwei Personen (hinsichtlich Krankheits- und Urlaubsvertretung) auf die Adressdaten zugreifen können – beispielsweise zur Erfüllung der Auskunftspflicht gegenüber staatlichen Behörden (siehe 4.4.3).

4.8. Sicherheit der Dienstanwendung

Für die Betrachtung der Sicherheitsgefährdungen und den daraus abzuleitenden Sicherheitsmaßnahmen ist es sinnvoll, sich an den Komponenten der Dienstanwendung zu orientieren. Bei der Anbindung an den eID-Service (der Bundesdruckerei) gelten die gleichen Gefährdungen und Maßnahmen, wie sie im Abschnitt 3.3.6 dargelegt wurden.

Beim LDAP-Server sind zwei Aspekte besonders zu berücksichtigen:

1. der Zugriff auf die Objekte des DIT und deren Attribute,

2. die Verfügbarkeit des Verzeichnisdienstes.

Da bereits LDAP-Server am Institut für Informatik betrieben werden und zudem zahlreiche Literatur existiert (z. B. [LU09], Abschnitt 3.7), wie man den Zugriff auf ein DIT regelt, empfiehlt es sich, die bestehenden Regeln zu übernehmen. Dennoch sollten die Betrachtungen zum Datenschutz in Abschnitt 4.7 berücksichtigt werden und in das Regelwerk einfließen.

Sehr wichtig für den Betrieb der von den LDAP-Servern abhängigen Dienste ist die Verfügbarkeit des Verzeichnisdienstes. Auch hierfür existieren mit der *Replikation* und der *Partitionierung* bewährte Methoden, um eine Skalierbarkeit des Verzeichnisdienstes zu erreichen. Die Replikation sorgt dafür, dass eine Kopie des DIT auf separaten LDAP-Servern zur Verfügung steht, die Partitionierung hingegen teilt einen DIT auf mehrere LDAP-Server auf, wobei jeder Server für den jeweiligen Teilbaum verantwortlich ist. Beide Verfahren können kombiniert werden.

Das Testlogin-System ist nur zu Demonstrationszwecken Teil des Dienstes und soll, wie bereits dargelegt, nicht im Produktivbetrieb zum Einsatz kommen. Andere (produktive) UNIX-Server, die sich ähnlich wie der Testserver verhalten (Einbindung der LDAP-Server zur Accountabfrage, Login-Möglichkeit per SSH, etc.), sind gemäß den üblichen Maßnahmen für Internet- und Intranetserver abzusichern. Dazu zählt unter anderem das Deaktivieren nicht benötigter Dienste auf dem Server und das regelmäßige Einspielen von Software-Updates, um das Ausnutzen bekannter Schwachstellen zu verhindern. Zusätzlich können die Prüfsummen (Fingerprints) der öffentlichen Schlüssel der SSH-Hostkeys[15] in den DNS-Einträgen des SSH-Servers (DNS-Record-Typ: „SSHFP“) hinterlegt werden, die ein SSH-Client abfragt, um Man-in-the-Middle-Angriffe zu erschweren, vgl. [RFC4255].

Da auf dem Webserver die TLS-Kanäle zwischen Diensteanbieter und Benutzer terminiert werden, ist an dieser Stelle auch der Zugriff auf den privaten Schlüssel zum verwendeten X.509-Zertifikat notwendig, das zur Authentisierung des Webservers gegenüber dem Benutzer zum Einsatz kommt. Dieser private Schlüssel ist am besten in einem HSM zu speichern oder sofern nicht vorhanden mit den restriktiven Zugriffsrechten im Dateisystem abzulegen (Leserecht für den Webserver-Prozess, sonst keinerlei Zugriffsmöglichkeiten). Dennoch ist es möglich, dass ein solcher Schlüssel kompromittiert wird, weshalb der Schlüssel nach Ablaufzeit des X.509-Zertifikats bzw. Bekanntwerden einer Kompromittierung des Servers bzw. Schlüssels auf einem sicheren System neu zu erzeugen ist.

Zu guter Letzt ist die Webanwendung potentiell erheblichen Angriffen ausgesetzt. Die Angriffe können sich auf der einen Seite gegen die Frontendmodule, auf der anderen Seite gegen die Callback-Schnittstelle richten. Gegen Angriffe der Frontendmodule sind in der Software der Diensteanwendung Eingabeprüfungen von Formulardaten bzw. POST-Daten vorzunehmen und ungültige Eingaben zurückzuweisen. Die Callback-Schnittstelle empfängt SAML-Responses vom Benutzer, die vom eID-Server erzeugt werden. Dabei sind, wie in Abschnitt 3.3.6 beschrieben, die Signatur, die Kanalbindung (PSK) und die Gültigkeit (anhand des Zeitstempels) der Response zu prüfen.

Da durch die Webanwendung direkt Accounts der Organisation modifiziert werden können, ist ein Auditing der im Backend ausgeführten Aktionen (LDAP-Zugriffe, etc.) empfohlen und kann leicht als zusätzliches Backend-Modul oder als Proxy vor den LDAP-Zugriffen implementiert werden.

[15] Der SSH-Hostkey dient analog zum Schlüsselpaar bei TLS-Verbindungen im SSH-Protokoll zur Authentisierung des Servers gegenüber dem Client. Da sich hierfür bislang keine PKI-Strukturen etabliert haben, wird dem Benutzer zur (initialen) Verifikation der Fingerprint des öffentlichen Schlüssels angezeigt. Nach der Bestätigung durch den Benutzer wird der öffentliche Schlüssel auf dem lokalen Rechner in der Datei `known_hosts` gespeichert und in zukünftigen Sitzungen automatisch geprüft.

5. Implementierung

Dieses Kapitel beschreibt Werkzeuge und Gesichtspunkte, die für die Implementierung des im vorangegangen Kapitel präsentierten Benutzerverwaltungsdienstes relevant waren. Um die Auswahl der Software-Komponenten nachvollziehen zu können, werden zunächst die Entscheidungskriterien erläutert und im Anschluss einzelne Komponenten beschrieben.

5.1. Abhängigkeiten und Auswahlkriterien

Für die Implementierung der Software, die später den Dienst realisieren soll, hat man anfangs scheinbar alle Freiheiten bezüglich der zu verwendenden Laufzeitumgebung und somit der möglichen Programmiersprache. Ein weiterer wichtiger Punkt ist die Verfügbarkeit umfangreicher Softwarebibliotheken, da die Anwendungsentwicklung auch unter ökonomischen Gesichtspunkten effizient erfolgen sollte.

Zwei Entscheidungen aus der Entwurfphase schränken diese Freiheit jedoch erheblich ein: das *webbasierte Einsatzszenario der Anwendung* und die *Anbindung an den eID-Service*. Der notwendige Betrieb einer Webanwendung wird heutzutage durch das Angebot an zahlreichen Anwendungsservern dahin gehend vereinfacht, dass typische Aufgaben (z. B. die Sitzungsverwaltung bezüglich eines verbundenen Nutzers) direkt vom Anwendungsserver umgesetzt werden und der Programmierer somit seinen Fokus mehr auf die Realisierung der Kernfunktionalität richten kann. Zudem liefert der eID-Provider möglicherweise eine Softwarebibliothek, die zur Anbindung an den eID-Service zu benutzen ist. Auch wenn es mit der [TR-03130] eine Standardisierung für eID-Server und die verwendeten Schnittstellen gibt, sind entsprechende, frei verfügbare Bibliotheken bislang nicht erhältlich.

Wie bereits angemerkt wurde, soll für die Realisierung des Prototyps der eID-Service der Bundesdruckerei angebunden werden. Die Bundesdruckerei stellt hierzu ein Servlet namens *eidconnector* zusammen mit ausführlicher Dokumentation [BDr10] zur Verfügung. Ein Servlet ist eine spezielle Java-Klasse, die für die Belange einer Webanwendung passende Schnittstellen implementiert. Dazu gehören beispielsweise Methoden zur Verarbeitung der HTTP-Methoden GET, POST, etc.

Als Ausführungsumgebung wird dazu ein sogenannter *Servlet-Container* benötigt, der unter anderem die Klassen instanziiert und den Instanzen Verbindungen zuweist, Fehlerzustände abfängt, entsprechend formatiert und sich, wie angesprochen, um die Sitzungsverwaltung kümmert. Der Servlet-Container stellt hierbei den Anwendungsserver dar. Die eigentliche Laufzeitumgebung ist dabei die *Java Runtime Environment* (JRE), die aus der *Java Virtual Machine* (JVM) und der Java-Standard-Klassenbibliothek besteht.

Durch diese Abhängigkeiten kann die Implementierung nur in einer Programmiersprache erfolgen, für die es einen Compiler bzw. Interpreter gibt, der Zielcode für die JVM erzeugt. Dazu zählen, neben Java, beispielsweise die Sprachen: Groovy, JRuby, Scala, Jython. Die Implementierung des Prototyps erfolgte in *Jython*.

Jython verbindet die weit verbreitete Sprache *Python* und ihrer ebenfalls sehr umfangreichen Standardbibliothek mit der „Java-Welt“. Python ist als Programmiersprache, vor allem aufgrund der *Dynamischen Typisierung*, aber auch wegen der einfachen Notation von Listen- und Zeichenkettenoperationen besonders für die Erstellung von Prototypen geeignet. Der Jython-Interpreter erzeugt für den Jython-Programmcode, der nahezu vollständig der Python-Syntax entspricht, Java-Bytecode, der direkt auf der JVM ausgeführt wird. Der Vorteil gegenüber der Standard-Laufzeitumgebung für Python, *CPython* genannt, liegt darin, dass einerseits Python-Code überall dort ausgeführt werden kann, wo Zugriff auf eine JVM besteht bzw. keine CPython-Version existiert. Andererseits – und dies ist der entscheidende Grund für die Verwendung in dieser Implementierung – hat ein Jython-Programm Zugriff auf sämtliche Schnittstellen der JVM, wie dies beispielsweise auch für ein Java-Programm der Fall ist. Insbesondere können Klassenmethoden aus Softwarebibliotheken Dritter benutzt werden, die nur in compilierter Form als Bytecode verfügbar sind.

Ferner soll für die Implementierung des Prototypen (nach Möglichkeit) nur freie Software benutzt werden, um allen Interessierten die Möglichkeit zu geben, einen gleichartigen Dienst auf Basis der vorgestellten Dienstanwendung testen und einsetzen zu können.

5.2. Komponenten des Dienstes und Werkzeuge

Nachfolgend soll das Setup der benutzten Rechner-Instanzen kurz vorgestellt werden, wie es für die Demonstration des Prototyps benutzt wird.

5.2.1. Virtuelle Maschinen zum Aufbau der Demonstrationsumgebung

Die im Entwurf genannten Server Anwendungsserver, Webserver, LDAP-Server und der SSH-Server für Testlogins wurden als Dämonen in zwei virtuellen Maschinen (VM), basierend auf dem Debian Linux-Betriebssystem, aufgesetzt. Virtuelle Maschinen sind heutzutage ein etablierter Weg, Test-Systeme (wie diesen Prototyp) ohne die Notwendigkeit echte physische Maschinen betreiben zu müssen. Dabei werden aufgrund der wenigen zu erwartenden Anfragen an den Dienst keine große Anforderungen an die Virtualisierungsumgebung gestellt. Lediglich für die VM mit den Serverdiensten sollte wegen der Laufzeitumgebung des Anwendungsservers ausreichend Arbeitsspeicher (RAM) zur Verfügung stehen. Anschließend werden die Rahmendaten eines beispielhaften Nutzer-PCs und die Werkzeuge eines Entwickler-PCs aufgezählt.

5.2.2. VM I – Server-Dämonen

Der „Hauptserver“ beheimatet bis auf das Testlogin-SSH-System alle benötigten Dämonen zum Betrieb des Dienstes, d. h., sofern keine Test-Logins demonstriert werden sollen, reicht dieser Server aus. Als Nachweis der Backend-Aktionen könnte auch ein LDAP-Browser zum Betrachten der angelegten und modifizierten LDAP-Objekte dienen. Für die Software-Installation wurden, sofern möglich, Pakete der Linux-Distribution benutzt. Dies hat den Vorteil, dass über ein zentrales Kommando auf jedem Server Software-Updates automatisch und entsprechend den ausgewählten Paketen installiert werden können, ohne dass der Systembetreuer die Software hinsichtlich der Veröffentlichung von Sicherheits-

lücken überwachen muss. Bei den Linux-Distributionen (und auch bei anderen Betriebssystemen) wird diese Aufgabe von sogenannten Paket-Maintainern übernommen, die jeweils *ihr* Software-Paket im Auge behalten und ggf. Aktualisierungen für das Paket bereitstellen.

Anwendungsserver

Der zentrale Dämon neben dem Webserver ist der Servlet-Container Tomcat. Er bildet die Ausführungsumgebung der Dienstanwendung, lädt die installierten Servlet-Klassen in den Hauptspeicher und bindet sie entsprechend der Konfiguration unter einem URL-Pfad des Tomcat-Webservers ein. Für das Deployment von Webanwendungen existiert das Verzeichnis `webapps`, in das der Administrator die Installation vornimmt (in der konkreten Debian-Installation nach `/var/lib/tomcat5.5/webapps`). Die Anwendung kann entweder in Form eines *Web Archive* (`.war`-Datei) oder als eigenständiges Unterverzeichnis abgelegt werden. Im ersten Fall extrahiert der Tomcat-Server die Archivdatei automatisch in ein gleichnamiges Unterverzeichnis und löscht dieses bei Entfernen der `.war`-Datei wieder. Der Name des Unterverzeichnis bildet ebenfalls den URL-Pfad, unter dem die Webanwendung erreichbar ist. Die Datei `WEB-INF/web.xml` innerhalb des Unterverzeichnisses enthält die Konfiguration der Webanwendung.

Softwarebibliotheken, die für Webanwendungen benötigt werden, können entweder unterhalb des globalen Verzeichnisses `shared` oder des lokalen Verzeichnisses der Anwendung `WEB-INF/lib` abgelegt werden. Üblicherweise werden dafür *Java Archives* (`.jar`-Dateien) benutzt, die Java-Klassenbibliotheken mit Bytecode enthalten. Auf diesem Weg wurden der Jython-Interpreter `jython.jar` und dessen Standardbibliothek sowie die eID-Anbindung `eidconnector.jar` unter `/var/lib/tomcat5.5/webapps/account/WEB-INF/lib` eingerichtet.

Webserver

Als Webserver an der Benutzerschnittstelle wird der Apache-Webserver eingesetzt. Die Standardmethode zur Prozesskommunikation mit dem Tomcat-Server ist die Verwendung des Apache-Plugins *mod_jk*, das wegen einer Besonderheit allerdings hier nicht eingesetzt wird. Abschnitt 5.3.5 am Ende dieses Kapitels geht auf die Besonderheit ein.

LDAP-Server

Um die Anbindung des LDAP-Verzeichnisses nicht mit dem Produktivsystem des Instituts für Informatik testen zu müssen, wird ein (minimaler) OpenLDAP-Server für dem Prototyp betrieben. Dabei werden nur die Standard-Schemata

- `core.schema`,
- `cosine.schema`,
- `nis.schema`,
- `inetorgperson.schema`,
- `samba.schema` aus dem Paket `samba-doc`

geladen.[1] Um die grundlegende Verzeichnisstruktur in den (noch leeren) DIT zu laden, wurde die LDIF-Datei B.1 in Anhang B, zusammen mit dem Programm `ldapadd` benutzt.

SSH-Server

Der „Hauptserver“ erhält für die Konfiguration der Dämonen und für das Debugging einen SSH-Zugang, der vom Entwickler bzw. Administrator des Dienstes genutzt wird.

5.2.3. VM II – Testlogin-System

Die Funktionsweise der Accounterstellung mit dem Prototypen lässt sich durch einen Testlogin am Ende des Erstellungsprozesses gut veranschaulichen. Hierzu wurde eine weitere virtuelle Maschine mit Linux eingerichtet, die ihre Accountinformationen (bis auf statische Systemaccounts) aus dem LDAP-Verzeichnis im Backend bezieht. Per SSH-Sitzung loggt sich der neue Benutzer unter Eingabe des generierten Kennzeichens (Benutzername) und dem selbst gewählten Passwort auf dem Testsystem ein.

LDAP-Anbindung

Dabei wird vom **SSH-Server** zunächst geprüft, ob ein Account mit dem angegeben Namen existiert. Diese Namensauflösung wird über den *Name Service Switch* (NSS oder nsswitch) abgewickelt, die für unterschiedlichste Namensräume möglich ist. Gängige Namensräume, die über diesen Dienst abgefragt werden, sind unter anderem Rechner-, Benutzer- und Gruppennamen. Da der nsswitch über die Standard-C-Bibliothek eingebunden ist, werden die gleichen Informationen für alle Anwendungen bereitgestellt, die sich entsprechend des POSIX-Standards verhalten. Vor allem reicht es somit aus, spezielle Backends, wie ein LDAP-Verzeichnis, an zentraler Stelle zu konfigurieren: in der Datei `/etc/nsswitch.conf`. Die Parameter für das LDAP-Modul werden unter Debian in der Datei `libnss-ldap.conf` definiert.

Zur Überprüfung des Benutzerpassworts wird die PAM-Bibliothek[2] benutzt, in der vorab der LDAP-Server als Authentifikationsmechanismus über ein spezielles Backend-Modul konfiguriert wurde. Dieses Modul baut eine Verbindung zum LDAP-Server auf und versucht eine `bind`-Operation mit den vom Benutzer eingegebenen Credentials. Ist die Operation erfolgreich gewesen, gilt das auch für die Authentifikation des Moduls.[3] Je nach PAM-Konfiguration ist damit die gesamte Authentifikation positiv (`sufficient`) oder es werden für den Gesamtstatus noch weitere Module abgefragt (`required`). Im Fehlerfall, wenn das eingegebene Passwort falsch ist, hängt dieser Status ebenfalls von der PAM-Konfiguration

[1] Letzteres ist notwendig, da die Vorlage des (realen) Test-Benutzers, wie im Produktiv-Betrieb, entsprechende Attribute zur Abbildung der Windows-Nutzer-Eigenschaften besitzt. Zur Erinnerung: Das Test-DIT soll von der Struktur her dem Produktiv-DIT entsprechen, um ggf. Fehler der LDAP-Verzeichnisstruktur bei der Implementierung erkennen zu können.

[2] Die *Pluggable Authentication Modules* (PAM) bilden eine Abstraktionsschicht für Anwendungsprogramme auf UNIX-Systemen, die generische Mechanismen zur Authentifikation eines Benutzers anbieten. Das System lässt sich über Module erweitern.

[3] Sofern die `bind`-Option *simple* konfiguriert wurde (Standardeinstellung), werden die Passwörter im Klartext übertragen. Daher ist die Verwendung von TLS zur Transportsicherung (auch als LDAPS bezeichnet) oder alternativen `bind`-Optionen, wie sie die SASL-Bibliothek ermöglicht, dringend empfohlen.

ab. So gibt es PAM-Module, deren Fehlschlag nicht oder nur bedingt relevant den Gesamtstatus sind (`optional` bzw. `sufficient`). In [vJ06], S. 9ff., ist das Verfahren ausführlich dargestellt.

Am Ende des fehlerfreien Loginprozesses erhält der Benutzer seine eingestellte Login-Shell und kann mit den erteilten Rechten auf dem SSH-Server arbeiten.

Dieser Ablauf verdeutlicht auch die Notwendigkeit der beiden Mechanismen nsswitch und PAM. Während der nsswitch die Accountdaten bereitstellt, führt PAM ausschließlich die Authentifikation und, davon abhängig, zusätzliche Aktionen durch – beispielsweise das Erstellen eines Homeverzeichnisses für einen Nutzer, der sich erstmals am System einloggt. Ohne PAM müssten die (transformierten) Passwörter, wie beim (veralteten) *Network Information System* (NIS), vom LDAP-Server für die angeschlossenen UNIX-Rechner exportiert werden, die damit eine (lokale) Passwortprüfung durchführen würden. Die daraus resultierenden Probleme wurden in Abschnitt 2.3.1 beschrieben.

5.2.4. Testclient

Der Testclient dient als Demonstrationsplattform und simuliert einen (fiktiven) Benutzer, der den Dienst zur Verwaltung seines Accounts aufruft. Als notwendige Software sind daher (in einer virtuellen Maschine mit Windows XP) die **AusweisApp**, zwei **Webbrowser** (Mozilla Firefox und Microsoft Internet Explorer), ein **Kartenlesegerät** (Omnikey CardMan 5321) und ein **SSH-Client** zur Verbindung mit dem Testlogin-System installiert worden. Darüber hinaus wird mindestens ein Test-Ausweis benötigt – im Rahmen des Anwendungstests zum neuen Personalausweis standen bis zum 30.10.2010 Ausweise aus der „Mustermann-Familie“ zur Verfügung.

5.2.5. Entwickler-PC

Der Entwickler-PC benötigt neben dem obligatorischen **Editor** (oder einer Entwicklungsumgebung) ebenfalls einen Webbrowser (hier: Mozilla Firefox) und einen SSH-Client, um Installations-, Konfigurations- und Debug-Aufgaben auf den Servern durchführen zu können. Im Gegensatz zum Testclient ist hier nicht die Installation der AusweisApp und eines Kartenlesegerätes erforderlich. Eine eID-Sitzung kann funktional auch mit einem *Testdienst* des eID-Service durchgespielt werden, der keines Ausweises bedarf und den statischen Datensatz eines fiktiven Ausweises über die eID-Schnittstelle liefert.

Für den Webbrowser Firefox existiert die Erweiterung (Add-on) namens **HttpFox**, mit der sich der Verlauf einer Websitzung, also die HTTP-Aufrufe inklusive der benutzten Parameter und ausgetauschten Daten, nachvollziehen lässt.

Zusätzlich kann für den Zugriff auf das LDAP-Verzeichnis ein LDAP-Browser benutzt werden, um direkt Manipulationen am DIT vorzunehmen bzw. Backend-Operationen überwachen zu können. Sowohl das Add-on HttpFox als auch der Browser LdapAdmin[4] haben sich bei der Entwicklung der Webanwendung bewährt.

[4] `http://ldapadmin.sourceforge.net/`.

5.3. Die Dienstanwendung

Im Mittelpunkt der Betrachtung steht die Dienstanwendung, deren modularer Aufbau die Komplexität beherrschbar macht und Flexibilität bezüglich zusätzlicher Funktionen bringt. Neue Module können später leicht integriert werden.

5.3.1. Verzeichnisstruktur

Für eine bessere Übersicht werden die verwalteten Dateien thematisch in Unterverzeichnissen gruppiert. Die folgende Tabelle gibt einen Überblick:

Pfad	Beschreibung
`backend`	Backend-Module zum Zugriff auf Subsysteme (z. B. LDAP)
`config`	Konfiguration des Dienstes
`eid`	Modul zur Anbindung des eID-Servers
`keys`	Speicherort für Schlüsseldateien
`lib`	Gemeinsam genutzte Klassen und Methoden
`modules`	Frontend-Module der Dienstes
`static`	Statische Inhalte für Frontend-Module (z. B. Bilder, Stylesheets)
`templates`	Vorlagen für Frontend-Module (Trennung von Inhalt & Form)
`index.jsp`	Einstiegsseite des Servlets (Session-Initiierung)
`index.py`	Startseite des Dienstes (Modul-Auswahl)

5.3.2. Module

Frontend-Module werden im Verzeichnis `modules` abgelegt. Die Startapplikation `index.py` importiert das Meta-Objekt der Frontend-Module.[5] Aus den Beschreibungen des Meta-Objekts erzeugt die Startapplikation wiederum eine Liste von Verknüpfungen, jeweils aus dem Dateinamen und der Beschreibung eines Moduls (Meta-Attribut `description`). Beim Aufruf der so definierten URL eines Frontend-Moduls instantiiert der Servlet-Container aus der Modul-Datei ein Servlet mithilfe des Jython-Interpreters. Im Anhang B.2 ist einerseits die angepasste Konfigurationsdatei der Tomcat-Anwendung `web.xml` und andererseits der grundlegende Aufbau eines Frontend-Moduls `modules/example.py` angegeben.

Backend-Module hingegen werden durch die Frontend-Module mittels `import`-Anweisung eingebunden und nicht als Servlet angelegt. Neue Module diesen Typs können durch Kopieren in das Verzeichnis `backend` installiert werden und müssen entsprechend im gewünschten Frontend-Modul eingebunden werden.

Das **eID-Modul** – im Grunde auch ein Backend-Modul – hat eine Sonderrolle in der Dienstanwendung. Durch ein eigenes Unterverzeichnis und wegen des Import-Mechanismus von Python/Jython ist es möglich, die konkrete Implementierung der eID-Schnittstelle austauschen zu können, da zunächst versucht wird, die Datei `__init__.py` im Verzeichnis `eid` zu laden. Diese Datei kann dann auf die gewünschte eID-Implementierung durch einen erneuten Import der Modul-Bezeichner (`from bdr import eid`) verweisen, ohne dass ein

[5] Die Module in diesem Verzeichnis werden an der Dateiendung `.py` identifiziert.

Frontend-Modul angepasst werden muss. Dazu bedarf es aber einer genauen Schnittstellen-Definition der Methoden, die vom eID-Modul bereitgestellt werden. Durch den Aufruf der Methode `sendRequest()` des eID-Moduls wird die eigentliche eID-Sitzung gestartet. Im Falle des Bundesdruckerei-eID-Services wird also ein SAML-Request an den eID-Server geschickt usw. Nachdem der Personalausweis gelesen wurde, erfolgt durch den Webbrowser des Nutzers der (automatische) Aufruf der Callback-URL (auch *assertionConsumerUrl*). In dieser Anwendung wird dazu ein Servlet auf Basis von `verarbeiteAusweisdaten.py` eingerichtet, das zunächst (anhand eines *Session-Cookies*) prüft, ob die Sitzung der Webanwendung gültig ist,[6] und lässt dann bei erfolgreicher Prüfung die SAML-Response, durch Aufruf der Methode `handleResponse()` des eid-Moduls, auswerten. Abschließend wird die `callback()`-Methode des ursprünglichen Frontend-Moduls zur Verarbeitung der Ausweisdaten gerufen. Welche Methode konkret zu rufen ist, ermittelt das Servlet anhand der in einer Session-Variablen hinterlegten Referenz.

5.3.3. eID-Anbindung

Eigens um die Implementierung von eID-Anbindungen zu erleichtern und Probleme auf Clientseite auszuklammern, bietet der eID-Service der Bundesdruckerei zwei Dienstausprägungen an:

1. Der Testservice (Sandbox) liefert einen statischen Datensatz, ohne einen nPA auslesen zu müssen.

2. Der Liveservice (Referenzsystem) ermöglichte während des Anwendungstests zum neuen Personalausweis das Auslesen von Testausweisen. Eine ähnliche Funktion bietet die Bundesdruckerei auch nach dem Ablauf des Anwendungstests für spezielle Ausweismuster („Test Online-Ausweisfunktion“) an. Der operative Betrieb zum Auslesen von „echten“ Personalausweisen (inklusive der nötigen PKI-Strukturen) wird hingegen durch einen separaten eID-Server (Wirksystem) angeboten.

Mit dem Testservice lassen sich zudem Fehlersituationen simulieren. Nach Anpassung der Dienst-URL des eID-Servers (*eIdServiceDestination*) liefert der Dienst pro eID-Sitzung eine andere Fehlermeldung, so dass die eigene Anwendung auf die korrekte Fehlerbehandlung hin untersucht werden kann.

5.3.4. Konfiguration

Das Verzeichnis `config` beinhaltet Konfigurationsdateien für die Module der Anwendung. Mittels der Python-Klasse `ConfigParser` aus der Standardbibliothek[7] ist es leicht möglich, anwendungsspezifische Parameter zu definieren und in den Modulen zu verwenden. Die Syntax der Konfigurationsdateien ist an die unter Windows gebräuchlichen INI-Dateien angelehnt. Auf diese Art werden vom Prototyp vor allem Einstellungen für den Zugriff auf das LDAP-Verzeichnis (siehe `ldap.cfg` im Anhang B.2) und auf den eID-Server (siehe `testserver.cfg` und `liveservice.cfg` im Anhang B.2) festgelegt.

[6] Alternativ kann ein Nutzer auch auf Basis eines SAML-Attributs wiedererkannt werden, sofern der Webbrowser keine Cookies übermittelt.

[7] `http://docs.python.org/library/configparser.html`.

Typische Anpassungen

Nach der Installation der Dienstanwendung sind typischerweise folgende Anpassungen der Konfiguration vorzunehmen:

- Es ist Schlüsselmaterial für den Zugriff auf den eID-Service gemäß den Angaben des eID-Providers zu erzeugen und im Verzeichnis `keys` abzulegen, vgl. [BDr10], Abschnitt 4.2.
- Die Zugangsdaten zum LDAP-Server sind in der Datei `config/ldap.cfg` anzugeben. Nur so können Objekte im LDAP-Verzeichnis durch den Dienst geändert werden.
- Bei Nutzung des Bundesdruckerei-eID-Service sind in der Datei `liveservice.cfg` folgende Parameter anzupassen:
  - serviceProviderIssuerUrl – die Einstiegsseite (Mandantenkennung) des Dienstes,
  - assertionConsumerUrl – die Adresse die nach dem Auslesen des Ausweises vom Client aufgerufen wird (Callback-URL),
  - Angaben zum Schlüsselmaterial (Dateinamen, Zugriffskennwörter auf private Schlüssel).

5.3.5. Besonderheiten bei der Implementierung

Webserver zum Umschreiben der URLs

Während des Anwendungstests bis zur Veröffentlichung von Version 1.5 des eidconnectors war es nicht möglich, die Callback-URL des eigenen Dienstes zur Laufzeit zu konfigurieren und (über den SAML-Request) dem eID-Server mitzuteilen.[8] Beim Liveservice wurde daher die *AssertionConsumerUrl* statisch für den Mandanten in der Datenbank des eID-Servers hinterlegt und während der Erzeugung der SAML-Response integriert. Das Problem an dieser Implementierung stellte sich wie folgt dar: Der Testservice schickte die SAML-Response immer an die vordefinierte URL:
`https://localhost:1443/Example-ServiceProvider/saml/SAMLAssertionConsumer`.
Dies erschwerte zum einen die Demonstration, weil zunächst ein TCP-Tunnel vom lokalen Rechner (Port 1443) zum Anwendungsserver, beispielsweise per SSH, aufgebaut werden musste. Zum anderen musste aufgrund der in Abschnitt 5.2.2 beschriebenen Erzeugung der URL-Pfade das Callback-Servlet der eigenen Webanwendung in der Testphase stets dem Schema `Example-ServiceProvider/saml/SAMLAssertionConsumer` entsprechen.

Als Lösung dieses Problems wurde deshalb der Apache-Webserver dem Tomcat-Anwendungsserver vorgeschaltet, der per Erweiterungsmodul `mod_proxy` die Callback-URL bei Nutzung des Testservice so umschreibt, dass es für den Tomcat-Server so aussieht, als wurde die selbst definierte URL des Benutzerverwaltungsdienstes aufgerufen.[9]

[8] Nur der eID-Server kann aufgrund des Kommunikationsablaufs bestimmen, an welche URL die SAML-Response durch den Nutzer geschickt wird.

[9] Da diese Umschreibung jedoch erst beim Dienst stattfand, musste der (Test-)Nutzer dennoch wie beschrieben einen Tunnel zum Anwendungsserver aufbauen. Erst seit Version 1.5 des eidconnectors ist eine „reguläre“ Nutzung des Testservice mit der eigenen Anwendung möglich.

LDAP-Anbindung mit Jython

Im Gegensatz zu anderen Funktionen, für die man sich bei Python-Modulen der Standardbibliothek bedienen kann, ist der Zugriff auf LDAP-Server etwas schwieriger zu realisieren. Für die CPython-Laufzeitumgebung existiert dafür das separate Modul `python-ldap`[10], das aber nicht für Jython zur Verfügung steht. An dieser Stelle wird deshalb das *Java Naming and Directory Interface* (JNDI) benutzt, welches eine LDAP-Unterstützung ermöglicht und mit der Java-Standardbibliothek ausgeliefert wird.

Das Backend-Modul `ldap.py` kapselt diese Funktionalität in der Klasse `ldapConnection`. Die Verbindungsparameter werden dazu aus der Konfigurationsdatei `ldap.cfg` gelesen.

5.3.6. Quelltexte

Die Quelltextdateien der Dienstanwendung befinden sich auf der CD-ROM, die den gedruckten Exemplaren dieser Arbeit beiliegt.

[10] http://www.python-ldap.org/.

6. Fazit

6.1. Zusammenfassung

In dieser Arbeit wurde untersucht, inwieweit die Verwaltung von Benutzerkonten einer Organisation, insbesondere das Zurücksetzen vergessener und gesperrter Passwörter, durch die Verwendung der eID-Anwendung des neuen Personalausweises (nPA) erleichtert und eigenständig durch den Benutzer durchgeführt werden kann. Darüber hinaus wurde geprüft, ob analog dazu neue Benutzerkonten und die Verknüpfung von nPA und Benutzerkonto möglich ist. Ein Prototyp eines Dienstes, der diese Aufgaben assistiert, demonstriert die praktische Umsetzbarkeit. Für die Benutzung von Diensten und Testausweisen wurde auf die Infrastruktur und Mittel im Rahmen des offenen Anwendungstests zum neuen Personalausweis zugegriffen.

Für die Untersuchung der Passwortverwaltung wurden zunächst die Probleme von passwortbasierten Authentifikationsverfahren aufgezeigt und Alternativen vorstellt. Die Möglichkeiten zum Zurücksetzen von Passwörtern bildeten die Vorlage für eines der Kernpunkte der Arbeit: das Zurücksetzen eines Benutzerpasswortes mithilfe des nPA.

Anschließend wurden Technik und Protokolle rund um den neuen Personalausweis mit Fokus auf die für den Inhaber (aus meiner Sicht) attraktivste Anwendung betrachtet: die eID-Anwendung, auch Online-Authentisierung genannt. Für viele Diensteanbieter ist die Verfügbarkeit von eID-Servern und deren Anbindung an bestehende Dienste – hier in Form von Webanwendungen – essentiell. eID-Services erlauben es, auf Basis der neuartigen eID-Anwendung die auf dem Ausweis aufgedruckten Personendaten in digitaler Form authentisch auszulesen. Die Voraussetzungen zum Auslesen durch einen Diensteanbieter, unter anderem die Erteilung eines Berechtigungszertifikats, werden erläutert. Schließlich werden unterschiedliche Szenarien vorgestellt, wie ein Diensteanbieter die Infrastruktur rund um das Herzstück, den eID-Server, prinzipiell organisieren kann.

Im Zentrum der Arbeit steht ein Verwaltungsdienst für Benutzerkonten auf Grundlage des neuen Personalausweises. Er ermöglicht es seinem Inhaber, administrative Aufgaben bezüglich seines Benutzerkontos wahrzunehmen. Dazu zählen das „Zurücksetzen des Passworts“, das „Verknüpfen eines neuen Ausweises mit einem bestehenden Benutzerkonto“ und (eingeschränkt) das „Anlegen eines neuen Benutzerkontos unter Nutzung der eID-Anwendung des Personalausweises“. Hierbei werden Probleme aufgelistet, die hauptsächlich die initiale Verknüpfung von Ausweis und Benutzerkonto betreffen.

Abschließend werden Hinweise und Empfehlungen hinsichtlich der Implementierung eines solchen Dienstes am Beispiel des für diese Arbeit entwickelten Prototyps abgeleitet.

6.2. Ausblick

Die Arbeit hat gezeigt, dass die Hauptaufgabe „Zurücksetzen eines Passworts“ mit Hilfe des neuen Personalausweises effektiv umgesetzt werden kann, sofern die Anbindung an einen eID-Server vollzogen und der Ausweis initial im IT-System der Organisation erfasst und mit dem entsprechenden Benutzerkonto (Useraccount) verknüpft wurde.

Das grundsätzliche Problem der Bindung eines Ausweisinhabers und zukünftigen Benutzers der IT-Systeme an eine Organisation bei der Erstellung eines Benutzerkontos stellt vor allem größere und nahezu anonyme Einrichtungen, wie eine Universität, vor offene Fragen. So dürfte es kritisch sein, die Verknüpfung eines Ausweises mit einem soeben erzeugten Benutzerkonto nur anhand des Vor- und Nachnamens vorzunehmen. In der Regel stehen bei einzelnen Unterabteilungen der Organisation, bei denen ein Account benötigt wird, aus Datenschutzgründen keine weiteren Abgleichmerkmale, beispielsweise das Geburtsdatum, zur Verfügung. Auch hinsichtlich der beschriebenen Probleme der Verknüpfung eines bestehenden Accounts mit einem Ausweis wäre daher der Einsatz des neuen Personalausweises bereits im Rahmen einer „Online-Einschreibung“[1] wünschenswert.

Als weiterer Punkt sind die Kosten für den Betrieb der eID-Infrastruktur zu berücksichtigen, die das Budget der meisten Institute sprengen dürfte. Deshalb würde ein solcher Dienst eher universitätsübergreifend betrieben werden. Dabei gilt es aber zu beachten, dass die ausgelesenen Daten nicht weitergegeben werden dürfen, so dass es eines separaten Pseudonymdienstes bedarf. Ein solcher zentraler Wiedererkennungsdienst basierend auf der eID-Anwendung des neuen Personalausweises, genauer der Restricted ID bzw. Pseudonymfunktion, wird in [MRJ11] vorgestellt. In der Praxis wäre der Dienst aufgrund der Komplexität und Kosten aber noch oberhalb einer großen Organisation, im universitären Umfeld wünschenswerterweise beim DFN, angesiedelt. Das Problem der Weitergabe der Daten wird dadurch gelöst, dass nur eine Wiedererkennung, aber keine Bereitstellung von Personendaten möglich ist. Dabei darf aber auch nicht die (ausgelesene) Restricted ID direkt weitergegeben werden, weil diese für alle angeschlossenen Hochschulen beim selben Ausweis identisch wäre und der Benutzer innerhalb des Pseudonymdienstes nachverfolgbar wäre. Dazu verwaltet der Pseudonymdienst eine separate Kennung, die durch die Restricted ID eines bekannten nPA referenziert wird. Somit genügt der Dienst den (strengen) Datenschutzanforderungen an den neuen Personalausweis und angeschlossenen Einrichtungen ist es möglich, einen Ausweisinhaber zweifelsfrei wiederzuerkennen. Hiermit wäre dann eine kostengünstige Zwei-Faktor-Authentisierung mit dem nPA für alle im DFN organisierten Forschungseinrichtungen realisierbar.

[1] Eine Online-Einschreibung in Form einer Webanwendung haben offenbar viele deutsche Hochschulen eingeführt, so dass eine Anpassung dieser Anwendung durchaus möglich erscheint.

A. Screenshots der Webanwendung

A.1. Startseite: Modulauswahl

Abbildung A.1.: Modulauswahl.

A.2. Test-Modul

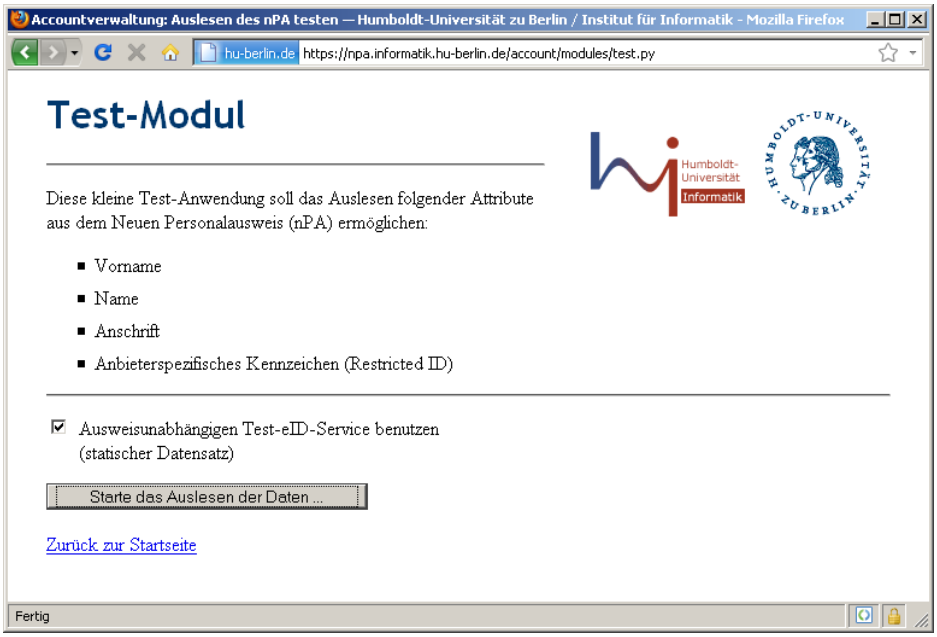

Abbildung A.2.: Daten auslesen starten.

Abbildung A.3.: Anzeige der „Ausweisdaten“.

A.3. Ein neues Passwort setzen

Abbildung A.4.: Eingabe des neuen Passworts.

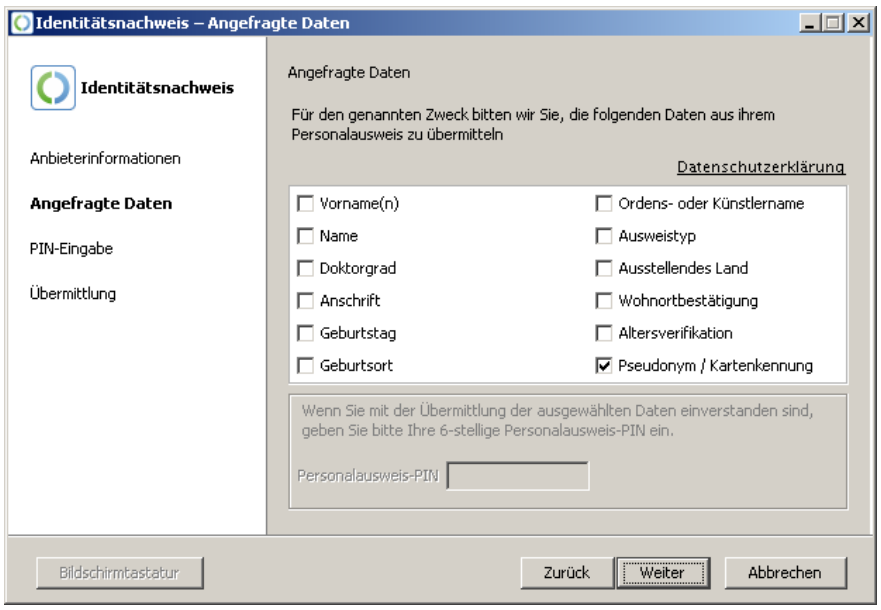

Abbildung A.5.: Bestätigung der Datenübermittlung.

Abbildung A.6.: Status der Transaktion und Anzeige des Nutzerkennzeichens.

A.4. Einen neuen Account beantragen.

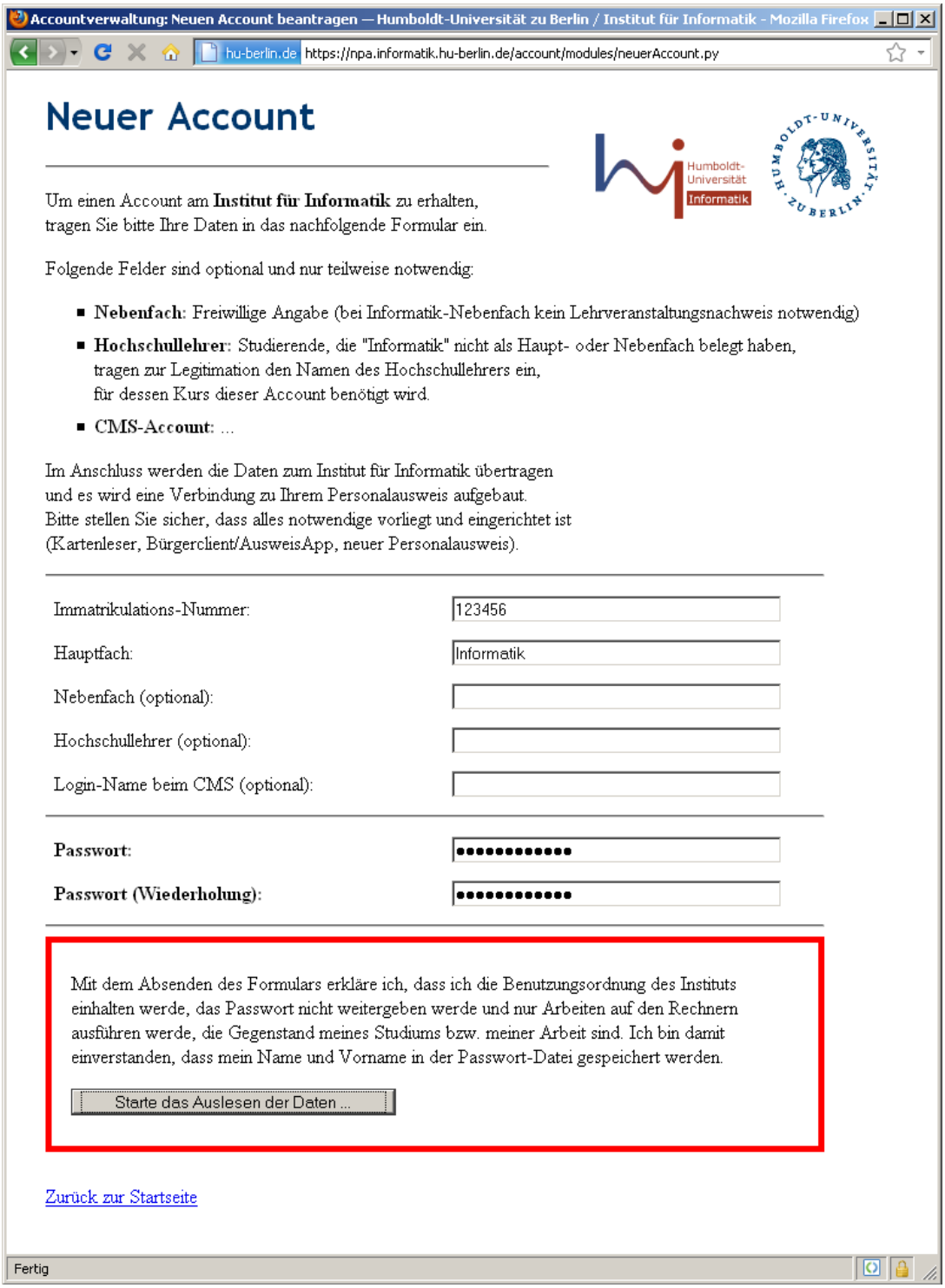

Abbildung A.7.: Ausfüllen des Formulars.

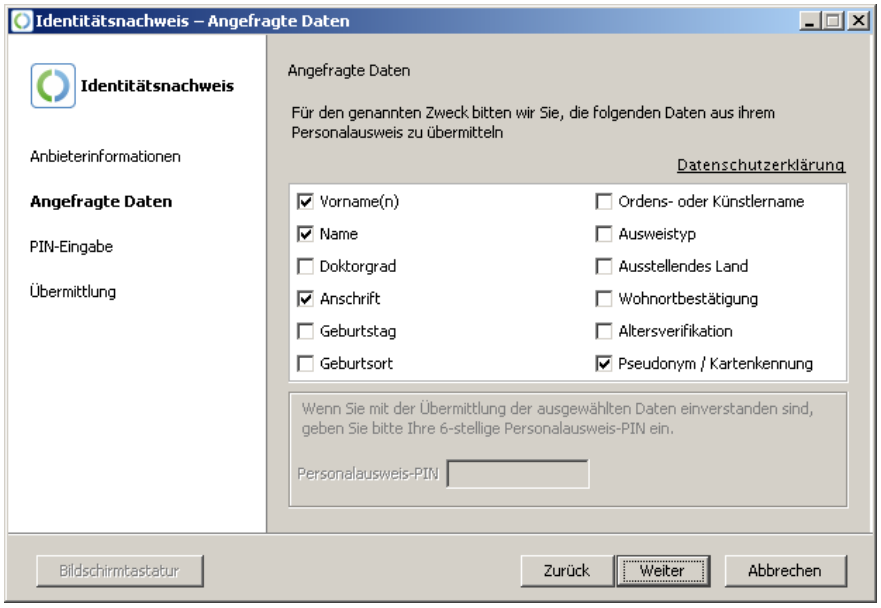

Abbildung A.8.: Bestätigung der Datenübermittlung.

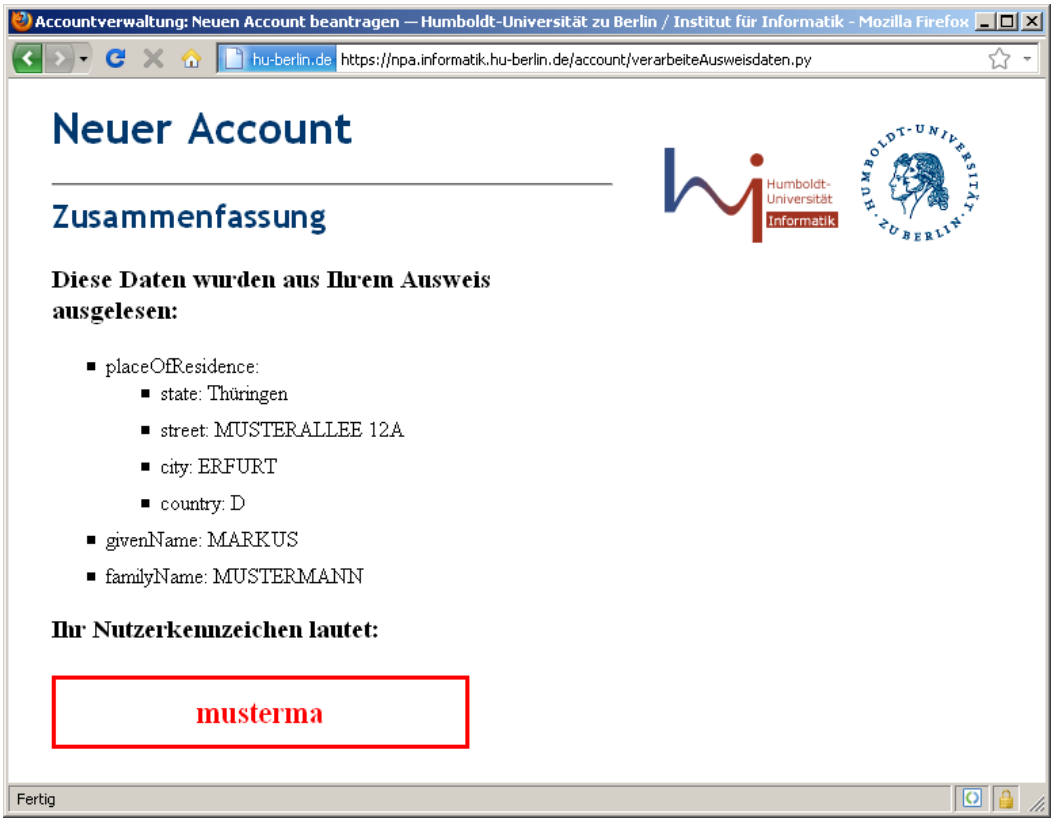

Abbildung A.9.: Status der Transaktion und Anzeige des Nutzerkennzeichens.

A.5. Fehlermeldung

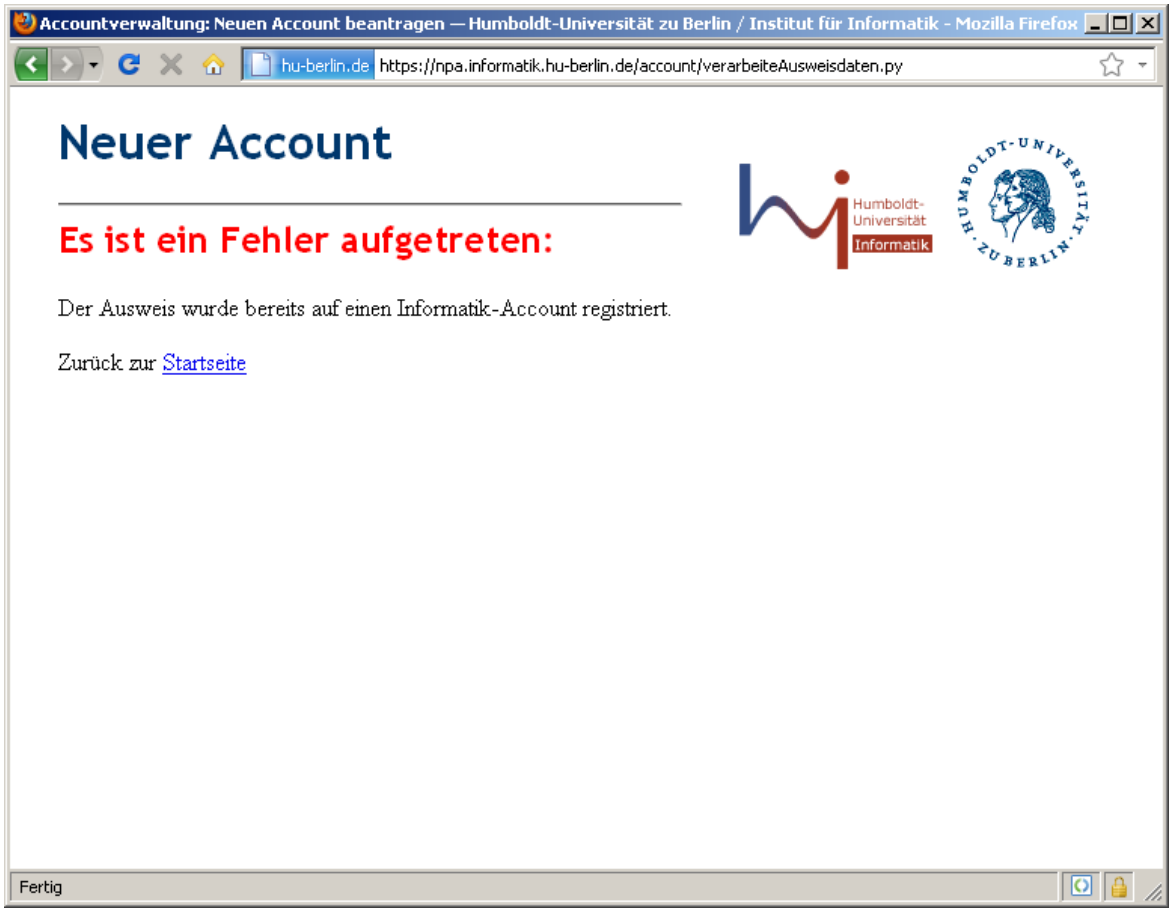

Abbildung A.10.: Der Ausweis wurde bereits mit einem Benutzerkonto verknüpft.

B. Konfigurationsdateien

B.1. LDAP

Listing B.1: Initiale LDAP-Einträge: `informatik.ldif`.

```
dn: dc=informatik,dc=hu-berlin,dc=de
objectClass: top
objectClass: dcObject
objectClass: organization
dc: informatik
o: Institut üfr Informatik

dn: ou=all,dc=informatik,dc=hu-berlin,dc=de
objectClass: top
objectClass: organizationalUnit
ou: all

dn: ou=People,ou=all,dc=informatik,dc=hu-berlin,dc=de
objectClass: top
objectClass: organizationalUnit
ou: People
description: Benutzer-Objekte

dn: uid=student,ou=People,ou=all,dc=informatik,dc=hu-berlin,dc=de
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: sambaSamAccount
uid: student
shadowLastChange: 1234567890
shadowFlag: 0
sambaPwdLastSet: 1234567890
sambaAcctFlags: [U          ]
sambaSID: S-1-5-21-1234567890-1234567890-123456789-12345
givenName: Test
sn: Student
cn: Test Student
uidNumber: 10001
gidNumber: 20001
gecos: Test Student, Informatik, CMS-1234
homeDirectory: /home/student
loginShell: /bin/bash
mail: test.student@informatik.hu-berlin.de
businessCategory: Informatik, CMS-1234
```

Listing B.2: OpenLDAP-Client-Bibliothek: `/etc/ldap/ldap.conf`.

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE            ou=all,dc=informatik,dc=hu-berlin,dc=de
URI             ldaps://ldap.example.com/
TLS_CACERT      /etc/ssl/certs/ldap.example.com.pem
```

Listing B.3: NSS-LDAP-Modul: `/etc/libnss-ldap.conf`.

```
# @(#)$Id: ldap.conf,v 2.48 2008/07/03 02:30:29 lukeh Exp $
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#
# PADL Software
# http://www.padl.com
#

# Your LDAP server. Must be resolvable without using LDAP.
# Multiple hosts may be specified, each separated by a
# space. How long nss_ldap takes to failover depends on
# whether your LDAP client library supports configurable
# network or connect timeouts (see bind_timelimit).
#host 127.0.0.1

#[...]

# Another way to specify your LDAP server is to provide an
uri ldaps://ldap.example.com/
```

Listing B.4: PAM-LDAP-Modul: `/etc/pam_ldap.conf`.

```
uri ldaps://ldap.example.com/
ldap_version 3
pam_password crypt
```

Listing B.5: PAM-Konfiguration: `/etc/pam.d/common-auth`.

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    sufficient      pam_ldap.so debug
auth    required        pam_unix.so nullok_secure
```

B.2. Benutzerverwaltungsdienst

Listing B.6: Konfiguration `config/testservice.cfg` für das eID-Modul.

```
[DEFAULT]
keyLocation = keys

[general]
# Url des eID-Service
eIdServiceDestination = https://test.eid-service.de/epa
serviceProviderIssuerUrl = https://localhost:1443/Example-ServiceProvider
assertionConsumerUrl = https://npa.informatik.hu-berlin.de/account/
    verarbeiteAusweisdaten.py
keyType = PEM

[PEM-KEYS]
signatureKeyPassword = TopSecretPrivateKeyPassPhrase
# Name der Datei mit dem privaten Signaturschluessel
signatureKeyLocation = %(keyLocation)s/testservice/MySignPrivateKey.pem
# Passwort zur Verwendung des privaten Verschluesselungsschluessels
encryptionKeyPassword = TopSecretPrivateKeyPassPhrase
# Name der Datei mit dem privaten Verschluesselungsschluessel
encryptionKeyLocation = %(keyLocation)s/testservice/MyEncrPrivateKey.pem

[eIdServiceCerts]
eIdServiceSigCertificate = %(keyLocation)s/eIdServiceSign.cer
eIdServiceEncCertificate = %(keyLocation)s/eIdServiceEncr.cer
```

Listing B.7: Konfiguration `config/liveservice.cfg` für das eID-Modul.

```
[DEFAULT]
keyLocation = keys

[general]
# Url des eID-Service
eIdServiceDestination = https://live.eid-service.de/epa
serviceProviderIssuerUrl = https://npa.informatik.hu-berlin.de/account/
assertionConsumerUrl = https://npa.informatik.hu-berlin.de/account/
    verarbeiteAusweisdaten.py
keyType = PEM

[PEM-KEYS]
signatureKeyPassword = TopSecretPrivateKeyPassPhrase
# Name der Datei mit dem privaten Signaturschluessel
signatureKeyLocation = %(keyLocation)s/liveservice/MySignPrivateKey.pem
# Passwort zur Verwendung des privaten Verschluesselungsschluessels
encryptionKeyPassword = %(signatureKeyPassword)s
# Name der Datei mit dem privaten Verschluesselungsschluessel
encryptionKeyLocation = %(keyLocation)s/liveservice/MyEncrPrivateKey.pem

[eIdServiceCerts]
eIdServiceSigCertificate = %(keyLocation)s/eIdServiceSign.cer
eIdServiceEncCertificate = %(keyLocation)s/eIdServiceEncr.cer
```

Listing B.8: Aufbau eines Beispiel-Frontend-Moduls `modules/example.py`.

```
import ...

# Modul-Beschreibung fuer Einstiegsseite der Accountverwaltung
class Meta:
  name = 'FrontendModul'
  title = 'Titel im Browserfenster'
  description = 'Modul-Beschreibung auf der Startseite'
  # Position des Modul-Eintrags auf der Startseite
  ordering = 1

class example(AccountModule):
  def init(self):
    # Hier erfolgt die Initialisierung des Frontendmoduls,
    # u.a. die Einrichtung der eID-Schnittstelle.
    ...

  def doGet(self, response, request):
    # Hier wird der abgebildete Prozess des Moduls gestartet,
    # i.d.R. durch Praesentation eines HTML-Formulars.
    # (Es hat sich bewaehrt hierfuer eine Template-Software zu verwenden.)
    ...

  def doPost(self, response, request):
    # Die Eingabe-Daten des Formulars aus doGet() werden hier verifiziert
    # und die Uebergabe der "Ablaufkontrolle" an die eID-Komponente
        vorbereitet.
    ...

  def callback(self):
    # Diese Methode wird vom "Ruecksprung-Programm" aufgerufen und
        ermoeglicht
    # somit eine Weiterverarbeitung des Ausweisdaten durch dieses Modul.
    # Es wird in dieser Arbeit das Ruecksprungprogramm "
        verarbeiteAusweisdaten.py"
    # genutzt, welches die Modul-Auswahl anhand einer Sitzungsvariablen
        trifft.
    ...
```

Listing B.9: Konfiguration `config/ldap.cfg` für das LDAP-Backend-Modul.

```
[LDAP]
serverUrl = ldap://localhost/
baseDn = ou=nPAPeople,ou=all,dc=informatik,dc=hu-berlin,dc=de
bindDn = cn=admin,dc=informatik,dc=hu-berlin,dc=de
bindPw = secret
```

Listing B.10: Tomcat-Konfiguration `web.xml` der Webanwendung.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/
       xml/ns/javaee/web-app_2_5.xsd">

        <display-name>Example-ServiceProvider</display-name>
  <servlet>
    <servlet-name>PyServlet</servlet-name>
    <servlet-class>org.python.util.PyServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>PyServlet</servlet-name>
    <url-pattern>*.py</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Abbildungsverzeichnis

Abkürzungsverzeichnis

ACL	Access Control List
AES	Advanced Encryption Standard
AGS	Amtlicher Gemeindeschlüssel
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BAC	Basic Access Control
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Certificate Authority
CA	Chip Authentication
CAN	Card Access Number
CHAT	CHAT Certificate Holder Authorisation Template
CMS	Computer- und Medienservice
CRL	Certificate Revocation List
CSCA	Country Signing Certificate Authority
CVCA	Country Validating Certificate Authority
DIT	Directory Information Tree
DFN	Deutsches Forschungsnetz
DN	Distinguished Name
DNS	Domain Name System
DS	Document Signer
DV	Document Verifier
EAC	Extended Access Control
eID	Elektronische Identität
ePA	Elektronischer Personalausweis
HSM	Hardware-Sicherheitsmodul
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICAO	International Civil Aviation Organization
IdM	Identitätsmanagement
IP	Internet Protocol
IT	Informationstechnik
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol

MAC . . . . . Message Authentication Code
MITM . . . . . Man In The Middle
MRTD . . . . . Machine Readable Travel Document
MRZ . . . . . Machine Readable Zone
nPA . . . . . Neuer Personalausweis
PACE . . . . . Password Authenticated Connection Establishment
PAM . . . . . Pluggable Authentication Modules
PIN . . . . . Persönliche Identifikationsnummer
PKI . . . . . Public Key Infrastructure
POSIX . . . . . Portable Operating System Interface
PSK . . . . . Pre-shared key
PUK . . . . . PIN Unblocking Key
QES . . . . . Qualifizierte Elektronische Signatur
RAM . . . . . Random Access Memory
RFID . . . . . Radio Frequency Identification
rID . . . . . Restricted Identification
SAML . . . . . Security Assertion Markup Language
SSH . . . . . Secure Shell
SSL . . . . . Secure Sockets Layer
SQL . . . . . Structured Query Language
TA . . . . . Terminal Authentication
TAN . . . . . Transaktionsnummer
TCP . . . . . Transmission Control Protocol
TKG . . . . . Telekommunikationsgesetz
TLS . . . . . Transport Layer Security
UID . . . . . Unique Identifier
URL . . . . . Uniform Resource Locator
VM . . . . . Virtuelle Maschine
VPN . . . . . Virtual Private Network
WLAN . . . . Wireless Local Area Network
WSDL . . . . . Web Services Description Language
XML . . . . . Extensible Markup Language

Literaturverzeichnis

[BDr10] BUNDESDRUCKEREI: *eID-Service Integrationshandbuch für Diensteanbieter*. Version 2.0.1. 2010.

[BKA11] HEISE ONLINE: *Schutz vor Skimming: BKA fordert magnetstreifenlose EC-Karten*. 2011. http://www.heise.de/-1162245.

[BKMN08] BENDER, Jens; KÜGLER, Dennis; MARGRAF, Marian; NAUMANN, Ingo: Sicherheitsmechanismen für kontaktlose Chips im deutschen elektronischen Personalausweis. In: *Datenschutz und Datensicherheit - DuD* 32 (2008), S. 173–177. http://dx.doi.org/10.1007/s11623-008-0026-7. – ISSN 1614–0702. – 10.1007/s11623-008-0026-7.

[BKMN10] BENDER, Jens; KÜGLER, Dennis; MARGRAF, Marian; NAUMANN, Ingo: Das Sperrmanagement im neuen deutschen Personalausweis. In: *Datenschutz und Datensicherheit - DuD* 34 (2010), S. 295–298. http://dx.doi.org/10.1007/s11623-010-0090-7. – ISSN 1614–0702. – 10.1007/s11623-010-0090-7.

[BNS05] BEUTELSPACHER, Albrecht; NEUMANN, Heike B.; SCHWARZPAUL, Thomas: *Kryptografie in Theorie und Praxis: mathematische Grundlagen für elektronisches Geld, Internetsicherheit und Mobilfunk*. Vieweg, 2005. – ISBN 9783528031688.

[Bv10] BAKKER, Marcus; VAN DER JAGT, Roel: *GPU-based password cracking* / University of Amsterdam. 2010. – Forschungsbericht. http://staff.science.uva.nl/~delaat/sne-2009-2010/p34/report.pdf.

[CCC08] *Chaos Computer Club konkretisiert Biometrie-Debatte an Schäubles Fingerabdruck*. 2008. http://www.ccc.de/updates/2008/schaubles-finger.

[CPeID] BSI: *Certificate Policy für die eID-Anwendung des ePA*. Version 1.26. 2010. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ElekAusweise/CVCA/Certificate_Policy.pdf.

[EC2a] HEISE ONLINE: *Preiswert Schlüssel knacken in der Cloud*. 2009. http://heise.de/-848574.

[EC2b] HEISE ONLINE: *GPUs knacken Passwörter in der Cloud*. 2010. http://heise.de/-1138949.

[Eck08a] ECKERT, Claudia: *Elektronische Reise- und Ausweisdokumente*. 2008.

[Eck08b] ECKERT, Claudia: *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. 5., überarbeitete Auflage. Oldenbourg, 2008. – ISBN 9783486582703.

[Eik11] EIKENBERG, Ronald: Sesam, öffne Dich nicht: Sicherheit von Passwörtern in Theorie und Praxis. In: *c't magazin für computer technik* (2011), Nr. 2, S. 150–152.

[FS09] FOX, Dirk; SCHAEFER, Frank: Passwörter -— fünf Mythen und fünf Versäumnisse. In: *Datenschutz und Datensicherheit - DuD* 33 (2009), S. 425–429. http://dx.doi.org/10.1007/s11623-009-0109-0. – ISSN 1614–0702. – 10.1007/s11623-009-0109-0.

[ICA08] ICAO: *Doc 9303 - Machine Readable Travel Documents - MRtds with Machine Readable Data Stored in Optical Character Recognition Format*. Version 3. International Civil Aviation Organisation, 2008.

[ISO 14443] ISO: *Identification cards – Contactless integrated circuit(s) cards – Proximity cards*. International Organization for Standardization, 2000.

[ITGS09] BSI: *IT-Grundschutz-Kataloge*. 11. Ergänzungslieferung. 2009. https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/kataloge.html.

[Kub11] KUBICEK, Herbert: Akzeptanzprobleme sicherer elektronischer Identitäten. In: *Datenschutz und Datensicherheit - DuD* 35 (2011), S. 43–47. http://dx.doi.org/10.1007/s11623-011-0012-3. – ISSN 1614–0702. – 10.1007/s11623-011-0012-3.

[LU09] LIEBEL, Oliver ; UNGAR, John M.: *OpenLDAP 2.4: Aktuell zur Version 2.4, Schemata, Services, Tools, SSL, TLS, ACLs, Samba, Kerberos...* Galileo Computing, 2009. – ISBN 9783836211987.

[MA07] MEZLER-ANDELBERG, Christian: *Identity Management – eine Einführung: Grundlagen, Technik, wirtschaftlicher Nutzen*. dpunkt, 2007. – ISBN 9783898644389.

[MO10] MORGNER, Frank; OEPEN, Dominik: *Die gesamte Technik ist sicher – Besitz und Wissen: Relay-Angriffe auf den neuen Personalausweis* / Humboldt-Universität zu Berlin. 2010. – Forschungsbericht. http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2010-13/SAR-PR-2010-13.pdf.

[MRJ11] MÜLLER, Wolf; REDLICH, Jens-Peter; JESCHKE, Mathias: Auth^2(nPA) – Starke Authentifizierung mit nPA für jedermann. In: *Datenschutz und Datensicherheit - DuD* 35 (2011). – Noch nicht veröffentlicht.

[Mü10] MÜLLER, Wolf: Documentless Proof of Identity. In: *Handbook of eID Security: Concepts, Practical Experiences, Technologies*. Publicis Publishing, 2010. – ISBN 9783895783791.

[OAS05a] OASIS (Hrsg.): *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.

[OAS05b] OASIS (Hrsg.): *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf.

[OAS08] OASIS (Hrsg.): *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. OASIS, 2008. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.pdf.

[PID11] *Identitätsprüfung – denn sicher ist sicher*. 2011. http://www.deutschepost.de/dpag?xmlFile=1015469.

[RFC4255] SCHLYTER, J.; GRIFFIN, W.: *Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints*. RFC 4255 (Proposed Standard). Januar 2006 (Request for Comments). http://www.ietf.org/rfc/rfc4255.txt.

[RFC4511] SERMERSHEIM, J.: *Lightweight Directory Access Protocol (LDAP): The Protocol*. RFC 4511 (Proposed Standard). Juni 2006 (Request for Comments). http://www.ietf.org/rfc/rfc4511.txt.

[Sch07] SCHMEH, Klaus: *Kryptografie: Verfahren, Protokolle, Infrastrukturen*. 3., überarbeitete und erweiterte Auflage. dpunkt, 2007. – ISBN 9783898644358.

[Sch09] SCHMEH, Klaus: *Elektronische Ausweisdokumente: Grundlagen und Praxisbeispiele*. Hanser, 2009. – ISBN 9783446419186.

[TR-03110] BSI: *Advanced Security Mechanisms for Machine Readable Travel Documents*. Version 2.03. Technical Guideline TR-03110. 2010. https://www.bsi.bund.de/cae/servlet/contentblob/532066/publicationFile/44802/TR-03110_v203_pdf.pdf.

[TR-03112] BSI: *eCard-API-Framework*. Version 1.1. Technical Guideline TR-03112. 2009. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html.

[TR-03127] BSI: *Architektur elektronischer Personalausweis und elektronischer Aufenthaltstitel*. Version 1.13. Technische Richtlinie TR-03127. 2010. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03127/BSI-TR-03127_pdf.pdf.

[TR-03130] BSI: *eID-Server*. Version 1.4.1. Technische Richtlinie TR-03130. 2010. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03130/TR-03130_TR-eID-Server_V1_4_pdf.pdf.

[VfB11] VERGABESTELLE FÜR BERECHTIGUNGSZERTIFIKATE: *Technische und organisatorische Anforderungen zur Nutzung von Berechtigungszertifikaten*. Version 1.1. 2011. http://www.personalausweisportal.de/SharedDocs/Downloads/DE/richtlinie_vfb_berechtigungen.pdf.

[vJ06] VON HAGEN, Bill; JONES, Brian K.: *100 neue Linux Server Hacks*. O'Reilly, 2006. – ISBN 9783897214613.

[Win05] WINDLEY, Philip J.: *Digital Identity*. O'Reilly, 2005. – ISBN 9780596008789.

Alle URLs wurden zuletzt am 31.05.2011 geprüft.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Berlin, den 1. Juni 2011

Unterschrift

Einverständniserklärung

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Instituts für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Berlin, den 1. Juni 2011

Unterschrift