INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Implementierung eines Debit-Abrechnungssystems für Jura-Kaffeevollautomaten

Leon Busse

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Implementierung eines Debit-Abrechnungssystems für Jura-Kaffeevollautomaten

Leon Busse

Aufgabensteller:	Prof. Dr. Helmut Reiser
Betreuer:	Dr. David Schmitz Helmut Tröbs
Abgabetermin:	01. November 2017

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

München, den 27. Oktober 2017

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Im Rahmen dieser Arbeit wird ein zentrales Debit-Abrechnungssystem für Kaffeemaschinen am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften entworfen und implementiert. Anhand dieser Problemstellung wird die Machbarkeit sicherer vernetzter Systeme von Haushaltsgeräten auf Basis von Mikrocontrollern demonstriert. Den Mitarbeitern soll der Kaffeebezug mithilfe bereits vorhandener RFID-Transponder des Gebäudezugangssystems ermöglicht werden. Zusätzlich werden umfangreiche Verwaltungsmöglichkeiten über ein webbasiertes Administrationstool bereitgestellt. Die Informationssicherheit des Systems wird durch bewährte sowie eigens dafür entwickelte Kommunikationsprotokolle gewährleistet. Auf etwaige Schwachstellen wird eingegangen und passende Lösungsvorschläge werden präsentiert. Die erfolgreiche Implementierung eines Prototypen wird detailliert beschrieben.

Inhaltsverzeichnis

1 Einleitung

Im Rahmen dieser Bachelorarbeit wird ein flexibles und sicheres Abrechnungssystem für die Kaffeevollautomaten des Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) entworfen und umgesetzt. Ziel ist es, die momentan manuell erfolgende Abrechnung durch einen vollständig automatisierten Vorgang mit umfangreichen Möglichkeiten zur Verwaltung und Anpassung zu ersetzen.

1.1 Motivation

Die manuelle Verwaltung und Abrechnung von kleinen Leistungen für Angestellte im Berufsalltag kann aufwendig und zeitraubend sein. Kaffeeautomaten, Snackbars und andere solcher Einrichtungen in kleinem Maßstab basieren zudem oft allein auf dem Vertrauen der Mitarbeiter untereinander. So sind Strichlisten oder Kleingeldboxen zwar eine schnelle Lösung und einfach zu nutzen, doch es kann schnell zu Chaos und Inkonsistenzen bei der Abrechnung kommen.

Wünschenswert ist ein Abrechnungssystem, das unkompliziert zu verwalten ist, Verbindlichkeit auf Seite der Nutzer sowie der des Betreibers schafft und dennoch so leicht wie eine Strichliste zu bedienen ist. Um ein kompliziertes Rechnungsverfahren zu vermeiden, soll es auf Guthaben-Konten basieren, auf die vor dem Kaffeebezug einzuzahlen ist. Ein solches Abrechnungssystem für das Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) zu entwerfen und umzusetzen, wird das Ziel dieser Arbeit sein. Zunächst geht es um die Einbindung von Kaffeevollautomaten der Firma Jura. Doch das Ergebnis dieser Arbeit soll flexibel und erweiterbar genug sein, um Kaffeemaschinen aller Art mit wenig Aufwand in das System zu integrieren.

Eine weitere Motivation für diese Arbeit ist die zunehmende Vernetzung unseres Alltags und die damit einhergehende Kommunikationsflut zwischen unterschiedlichsten Geräten. Vom Web-Server über den PC bis hin zum ins Thermometer integrierten Mikrocontroller können all diese Geräte miteinander kommunizieren Ein großes Problem ist derzeit jedoch die Vernachlässigung der Sicherheit in diesem Bereich. Es steht fest, dass (hinreichend) sichere Kommunikation zwischen modernen Computern und Servern über das Internet möglich ist. Doch der aktuelle Trend beim Thema Internet of Things geht in Richtung kleiner, kostengünstiger Mikrocontroller, deren Rechenleistung oft für moderne Verschlüsselungsstandards zu gering ist. Das in dieser Arbeit entworfene System soll ein Beispiel für ein sicheres, effizientes Abrechnungssystem sein, das auf Client-Seite mit der Rechenleistung kleinster Mikrocontroller auskommt.

1.2 Stand der Technik

Projekte wie *Sharepresso* aus dem Computermagazin c't [Sie16] oder das Forschungsprojekt *Bedienung und Verwaltung von Haushaltsgeräten mittels NFC am Beispiel eines Kaffeevoll-*

automaten von der Hochschule für Technik und Wirtschaft Berlin [Roh12] haben schon gute Arbeit auf dem Gebiet der Abrechnungssysteme für den Bürobedarf geleistet. Die von ihnen implementierten Systeme erfüllen jedoch nicht ganz die Anforderungen dieser Arbeit. So ist das *Sharepresso*-System eine rein lokale Lösung, d.h. alle Daten werden auf dem verwendeten Mikrocontroller gespeichert. Somit ist die Einbindung weiterer Kaffeemaschinen zur zentralen Verwaltung nicht möglich. Die in der Arbeit *Bedienung und Verwaltung von Haushaltsgeräten mittels NFC am Beispiel eines Kaffeevollautomaten* präsentierte Lösung verlässt sich auf das herstellerspezifische Jura Multi-Drop-Bus-Interface und Manipulationen der Kaffeemaschine selbst, wie das Anlöten von Kabeln an die Knöpfe der Maschine, was den Anforderungen an Kosteneffizienz und Flexibilität nicht gerecht wird.

In einer weiteren Arbeit, namentlich *Entwicklung eines web-basierten Verwaltungstools zur Protokollierung und Zugangssteuerung von Industriegeräten* [Fal15] der Leopold-Franzens-Universität Innsbruck, wird ein umfangreiches zentrales Abrechnungssystem mit komfortablen Möglichkeiten zur Bedienung und Verwaltung erarbeitet. Als Client wird jedoch der Minicomputer Raspberry Pi genutzt, was in einem hochsprachlichen Kommunikationsprotokoll mit dem Server (HTTPS, JSON) resultiert und auch hier wird Gebrauch von einer kommerziellen Schnittstelle zur Kaffeemaschine gemacht.

1.3 Problembeschreibung

Es gilt ein sicheres, erweiterbares, zentrales Debit-Abrechnungssystem für Kaffeevollautomaten beliebiger Hersteller zu entwerfen, das kostengünstige Mikrocontroller als Client-Plattform nutzt. Es soll die Abrechnung aller bereitgestellten Leistungen automatisieren, indem es die Guthaben-Konten der Nutzer verwaltet und, wenn nötig, auch die Ausgabe eines Produkts blockieren kann. Nutzer sollen ihre persönlichen RFID-Transponder für den Gebäudezugang auch zur Identifikation gegenüber dem Abrechnungssystem nutzen können. Dadurch kann eine Anmeldung per Passwort oder die Einführung zusätzlicher Identifizierungsmerkmale der Nutzer vermieden werden. Zudem sollen die Nutzer über den Prozess der Produktausgabe an der Kaffeemaschine informiert werden. Der Mikrocontroller muss also auch Feedback z.B. über einen Bildschirm geben können. Welche anderen Voraussetzungen das Abrechnungssystem darüber hinaus erfüllen sollte, wird im Rahmen dieser Arbeit erforscht, bevor ein Systementwurf erarbeitet wird.

Auf die Sicherheit des Systems wird besonderer Wert gelegt. Eine erfolgreiche Manipulation des Abrechnungssystems kann finanzielle Schäden für die Nutzer und den Betreiber des Systems bedeuten. Deswegen müssen alle Systemkomponenten, die über unsichere Kanäle kommunizieren, entsprechende Maßnahmen zur Verschlüsselung und Authentifizierung durchführen. Auf welche Weise dies mit den limitierten Ressourcen eines Mikrocontrollers sichergestellt werden kann, wird im Verlauf dieser Arbeit betrachtet.

1.4 Vorgehensweise

Die Realisierung eines Abrechnungssystem ist eine komplexe Aufgabe und muss deswegen einem strukturierten Vorgehensmodell folgen. Die Schritte, die in dieser Arbeit durchlaufen werden sind:

- Problembeschreibung

- Formale Definition der Anforderungen
- Entwurf des Abrechnungssystems
- Realisierung
- Evaluierung

Die Problembeschreibung ist in diesem Kapitel, insbesondere Abschnitt 1.3, erfolgt. Darauf folgt zunächst in Kapitel 2 die Beschreibung der relevanten theoretischen Grundlagen und Technologien, die zum Verständnis der folgenden Kapitel notwendig sind. Dazu gehört der Hardwarestandard RFID sowie Kenntnisse der IT-Sicherheit, insbesondere gängige kryptographische Methoden. In Kapitel 3 werden Use Cases beschrieben und die Anforderungen an das Abrechnungssystem formal festgehalten, sowie die aktuelle Situation am Leibniz-Rechenzentrum dargelegt. Darauf folgt eine Übersicht von existierenden Abrechnungssystemen, die einen ähnlichen Ansatz verfolgen, in Kapitel 4. Sie werden unter den zuvor definierten Anforderungen evaluiert und passende Teillösungen werden herausgearbeitet. Kapitel 5 befasst sich mit dem Entwurf des Abrechnungssystems. Die Architektur des Systems wird erarbeitet und verschiedene Systemkomponenten definiert, deren Aufgabenbereiche festgelegt werden. In Kapitel 6 wird die Implementierung des Prototyps geschildert. Genaue Software-Architektur und Datenmodelle des Abrechnungssystems, sowie die Präsentation der Kommunikationsprotokolle gehören dazu. Darauf folgt die Evaluierung des Prototyps gemäß den formalen Anforderungen aus Kapitel 3. Abschließend folgt die Zusammenfassung der erzielten Ergebnisse, Lösungsvorschläge für eventuell bestehende Probleme und Vorschläge für Erweiterungen des Systems im Fazit.

2 Grundlagen

Dieses Kapitel gibt einen Überblick über die theoretischen Grundlagen und Technologien, die für das Verständnis dieser Arbeit notwendig sind. Zunächst werden Grundlagen der IT-Sicherheit und der Kryptographie, insbesondere im Bereich der symmetrische Verschlüsselung, präsentiert. Darauf folgt eine kurze Erklärung des Hardware-Standard *RFID*, der in den folgenden Kapiteln von großer Relevanz ist.

2.1 Kryptographie

Dieser Abschnitt befasst sich mit einigen grundlegenden Kenntnissen aus dem Bereich der Kryptographie. Diese werden im weiteren Verlauf dieser Arbeit notwendig, um die sichere Kommunikation zwischen den Systemkomponenten zu verstehen. Nach einer kurzen Vorstellung der Ziele, die es mit kryptographischen Mitteln zu erreichen gilt, werden zunächst symmetrische Verschlüsselungsmethoden und darauf kryptographische Hash-Funktionen als Mittel zur Authentifizierung vorgestellt. Zuletzt wird ein Verfahren erklärt, das diese Methoden kombiniert: *Authenticated Encryption with Associated Data.*

2.1.1 Schutzziele

Um die Wirksamkeit und Nützlichkeit kryptographischer Maßnahmen in der Informationssicherheit beurteilen zu können, müssen zunächst die gewünschten Eigenschaften eines sicheren Systems definiert werden. Im Allgemeinen spricht man bei IT-Systemen von den sogenannten Schutzzielen:

- **Vertraulichkeit/Confidentiality.** Eine Nachricht soll nur von autorisierten Entitäten gelesen werden können.
- **Integrität/Integrity.** Eine empfangene Nachricht soll nachweisbar unverändert sein.
- **Verfügbarkeit/Availability.** Ein informationstechnischer Dienst soll autorisierten Entitäten jederzeit zugänglich sein.
- **Authentizität/Authenticity.** Bei Kontakt soll ein Kommunikationspartner zweifelsfrei identifiziert werden können.
- **Verbindlichkeit/Non-repudiation.** Eine gesendete Nachricht soll jederzeit zweifelsfrei einem bestimmten Absender zugeordnet werden können.

In bestimmten Situationen können noch andere Ziele definiert werden, wie zum Beispiel die Anonymität im Internet. [Spi11, S. 14-18]

2.1.2 Symmetrische Verschlüsselung

Schnelle Kommunikation ist heute eine unerlässliche Voraussetzung in fast allen Bereichen des Lebens. Sie wird ermöglicht durch verschiedene Systeme und Technologien, vor allem durch das Internet. Viele dieser Kommunikationskanäle, und im Speziellen das Internet, sind jedoch nicht sicher. Das bedeutet, die oben erläuterten Schutzziele werden nicht erreicht. Das Abhören von Nachrichten, Identitätsfälschung oder die Leugnung von Nachrichten ist möglich.

One Time Pad

Soll die Vertraulichkeit einer geheimen Nachricht sichergestellt werden, obwohl diese über einen unsicheren Kommunikationskanal gesendet wird, so ist die Verschlüsselung der Nachricht notwendig. Die simpelste Variante ist zunächst über einen sicheren Kommunikationskanal einen geheimen Schlüssel mit dem Kommunikationspartner auszumachen. Im Falle digitaler Nachrichten sind Schlüssel und Nachricht als Binärzahlen darstellbar und die Verschlüsselung kann, im einfachsten Szenario, durch XOR-Operation von Nachricht und Schlüssel erfolgen. Vorausgesetzt der Schlüssel ist von derselben Länge wie die Nachricht, können so Nachrichten sicher verschlüsselt werden. Die verschlüsselte Version der Nachricht (Ciphertext) kann durch dasselbe Verfahren vom Empfänger wieder entschlüsselt werden (um den Klartext zu erhalten). Diese Vorgehensweise wird als One-Time-Pad bezeichnet, da der Schlüssel nur einmal genutzt werden kann, ohne Sicherheitslücken zu öffnen. [Wil08, S. 67-68] Verfahren wie diese, bei denen beide Parteien denselben Schlüssel nutzen, nennt man symmetrische Verschlüsselung. [Buc16, S. 74-75]

Blockchiffren

In der Praxis sind jedoch kürzere Schlüssel gefragt, die über längere Zeiträume verwendet werden können. Moderne Blockchiffren (wie z.B. der *Advanced Encryption Standard*, kurz AES) erfüllen diese Voraussetzungen. Mit Schlüssellängen zwischen 128 und 256 Bit bieten sie so starke Verschlüsselung, dass die Entschlüsselung ohne Kenntnis des Schlüssels unvertretbar lange Zeit in Anspruch nehmen würde. Unvertretbar bedeutet hier, der gesuchte Schlüssel wäre zum Zeitpunkt der Berechnung nicht mehr gültig, bzw. damit verschlüsselte Daten nicht mehr sicherheitsrelevant.

Um diesen Grad an Sicherheit zu erreichen, wird der Klartext in Blöcke konstanter Länge geteilt. Jeder einzelne wird dann über mehrere Runden verschiedenen Byte-Substitutionen und linearen Abbildungen unterzogen und zusätzlich mit Rundenschlüsseln verknüpft (XOR). Die Rundenschlüssel werden aus dem Master-Schlüssel abgeleitet. Aufeinanderfolgende Blöcke können zusätzlich miteinander verknüpft werden, um Ciphertext-Abschnitte kontextabhängig und somit noch robuster gegenüber Angreifern zu machen (*Cipher Block Chaining Modus*). Fig. 2.1 zeigt den schematischen Ablauf des Rijndael Algorithmus, der unter dem Namen *Advanced Encryption Standard* standardisiert wurde. [Wil08, S. 69-73]

Stromchiffren

Alternativ kann unabhängig von den zu verschlüsselnden Daten ein Schlüsselstrom erzeugt werden, also eine Folge von pseudo-zufälligen Bits, die durch XOR-Operation mit den Daten verknüpft wird. Mithilfe des symmetrischen Schlüssels kann dieser Strom auf Empfängerseite

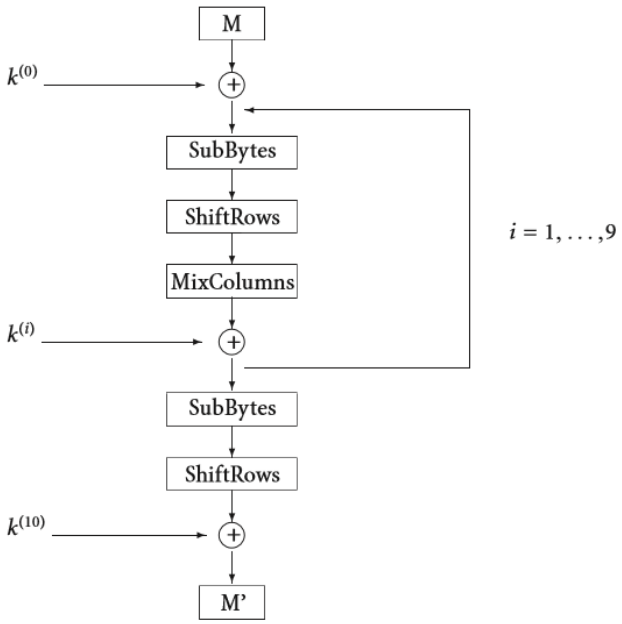

Abbildung 2.1: Schematischer Ablauf des Rijndael Algorithmus (Advanced Encryption Standard). Klartext M, Ciphertext M', Rundenschlüssel k^i. [Wil08, S. 70]

reproduziert werden, um durch erneutes XOR den Klartext zu erzeugen. Das bietet die Vorteile, dass die Ver- und Entschlüsselungsprozesse parallelisierbar sind und der Schlüsselstrom schon im Voraus berechnet werden kann. Darüber hinaus sind Stromchiffren im Allgemeinen effizientere Algorithmen, die oft gut für Systeme mit begrenzten Ressourcen geeignet sind. [vT11, S. 1263-1265]

Initialization Vector

Ein *Initialization Vector* (IV), oder auch *Nonce* (Number used Once), ist eine pseudo-zufällige Zahl, die bei dem Ver- bzw. Entschlüsselungsvorgang einer Nachricht miteinbezogen wird, um die kryptographischen Eigenschaften der Verschlüsselung zu verbessern. Sie kann zum Beispiel vom Absender generiert und zusammen mit der Nachricht verschickt werden, um die Entschlüsselung zu ermöglichen. Der Vorteil hierbei ist, dass auch im Falle identischer Nachrichten nie derselbe Ciphertext generiert wird. Ein solcher Fall würde einem Angreifer wertvolle Informationen über die Nachrichten geben.

2.1.3 Integrität und Authentizität

Über die Vertraulichkeit einer Nachricht hinaus ist es wichtig, die Integrität derselben sicherzustellen. Einer erhaltenen Nachricht ist nicht zu trauen, wenn man nicht nachweisen kann, dass sie nicht bei der Übertragung verändert wurde. Genauso muss der Ursprung der Nachricht überprüfbar sein, um Identitätsfälschung zu verhindern. Hash-Funktionen bilden die Grundlage für sogenannte *Message Authentication Codes* (MAC), die Integrität und Authentizität einer Nachricht garantieren.

Kryptographische Hash-Funktionen

Eine Hash-Funktion h bezeichnet eine mathematische Abbildung von einem Bitstring beliebiger Länge auf einen Bitstring von festgelegter Länge n, also

$$h : \{0,1\}^* \to \{0,1\}^n, \quad n \in \mathbb{N}.$$

Sie sind also nie injektiv, was zu zwei Bitstrings a, b führen kann, deren Funktionswerte identisch sind, also

$$x \neq x' \wedge h(x) = h(x') \quad x, x' \in \{0,1\}^*.$$

Das Paar (x, x') nennt man Kollision. Eine Hash-Funktion ist schwach kollisionsresistent, wenn man zu einem gegebenen x in vertretbarer Zeit kein anderes x' finden kann, für das gilt $h(x) = h(x')$. Stark kollisionsresistent sind solche Hash-Funktionen, bei denen der vorige Satz selbst bei frei wählbaren x und x' gilt. [Buc16, S. 233-236]

In der Praxis finden die Hash-Funktionen *SHA-2* und *SHA-3* (Secure Hash Algorithm 2 und 3) am meisten Anwendung. Sie wurden von der amerikanischen Standadisierungsbehörde NIST standadisiert und gelten heute als sicher. [Buc16, S. 239-240]

Message Authentication Codes

Um sowohl Integrität als auch Authentizität einer Nachricht zu gewährleisten, fügt man dieser einen Message Authentication Code (MAC) bei. Eine Möglichkeit zur Erzeugung eines MAC ist die Verwendung bestimmter Hash-Funktionen, die sowohl die Nachricht selbst, als auch einen kryptogrphischen Schlüssel als Argument verwenden. Den erzeugten Code bezeichnet man als *Keyed-Hash Message Authentication Code* (HMAC). Entsprechende Generationsfunktionen können einfach aus bekannten kryptographischen Hash-Funktionen konstruiert werden: Man korrigiert die Länge des Schlüssels K durch Anfügen von Nullen (falls zu kurz) oder durch Anwendung der gewählten Hash-Funktion h (falls zu lang), sodass er die gewünschte Länge des MACs n hat. Dann wird der MAC durch

$$h((K \oplus opad) \mid\mid h(K \oplus ipad \mid\mid M))$$

erzeugt, wobei *opad* und *ipad* bestimmte statische Bitstrings sind. Diese Konstruktion ist nötig, um auch mit nicht kollisionsresistenten Hash-Funktionen sichere MACs erzeugen zu können. Alternativ kann man eine MAC auch auf Grundlage einer Blockchiffre wie AES erzeugen, indem man nur den letzten verschlüsselten Block als MAC verwendet. Dazu muss die Blockchiffre im Cipher Block Chaining Modus betrieben werden, damit eine Änderung der Nachricht an beliebiger Stelle eine Änderung des MAC zur Folge hat.

Möchte man nun die Integrität und Authentizität einer mit MAC versehenen Nachricht überprüfen, ist lediglich die erneute Errechnung des MAC mithilfe des geheimen Schlüssels K und der Vergleich mit dem übermittelten MAC notwendig, um Manipulationen zu erkennen. Möchte ein Angreifer die Nachricht modifizieren, ohne bemerkt zu werden, muss er in der Lage sein, auch den MAC zu ändern. Um einen passenden MAC zu generieren, müsste er den geheimen Schlüssel K kennen. Die Integrität ist also gewährleistet, da die Nachricht nicht unbemerkt geändert werden kann. Die Authentizität ist gewährleistet, da nur Entitäten im Besitz des Schlüssels K die Nachricht verfasst haben können. [Buc16, S. 241-243]

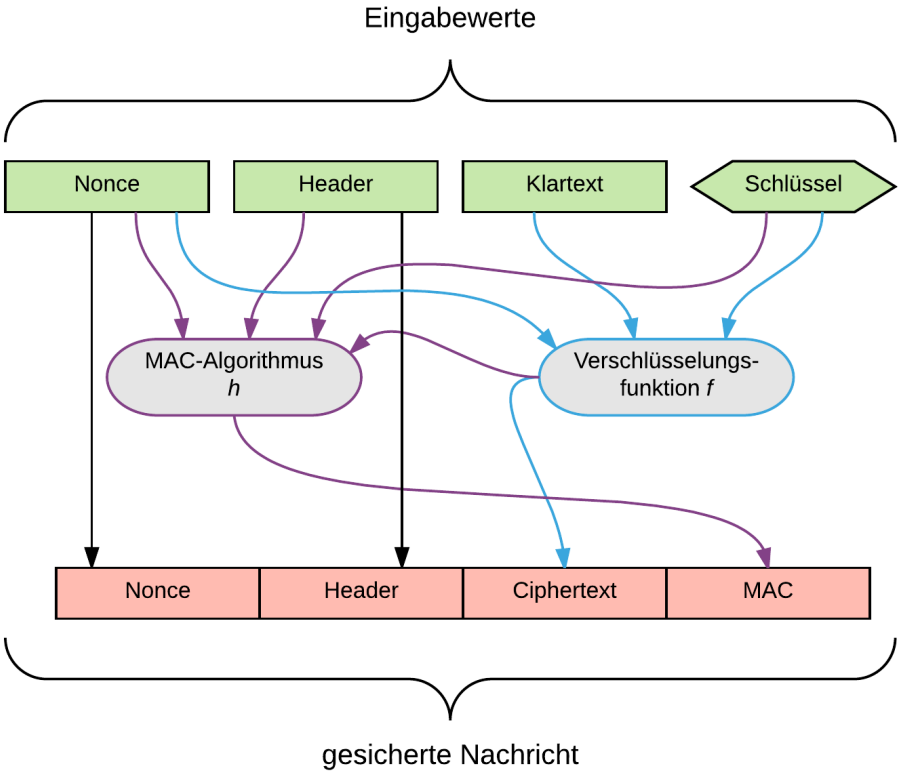

Abbildung 2.2: Schema der Generic Composition eines *Authenticated Encryption with Associated Data*-Verfahrens.

Authenticated Encryption with Associated Data

In vielen Situationen ist es wichtig, neben verschlüsselten Daten auch unverschlüsselte Informationen in derselben Nachricht zu verschicken. So zum Beispiel bei der gesicherten Kommunikation im Internet: Während die Nutzdaten selbst nur vom Empfänger einsehbar sein sollen, müssen Metadaten wie Herkunfts- und Zieladresse offen liegen. Schließlich wird auf dem Weg zum Empfänger die Nachricht unzählige Male weitervermittelt. Nur mithilfe der Zieladresse kann die Nachricht korrekt zum Empfänger geleitet werden. Einfaches Hinzufügen dieser Informationen zur mit MAC gesicherten Nachricht ist keine Lösung, da so keine Prüfung der Integrität möglich ist. Die zusätzlichen Informationen müssen also in den Sicherungsprozess einbezogen werden.

Ein Ansatz dazu ist die *Generic Composition*, die Komposition eines Verschlüsselungs- und eines Authentifizierungsalgorithmus. Die Algorithmen selbst sind dabei austauschbar. Oft kommt dabei die folgende Methode zum Einsatz:

1. Der Klartext P wird mit Schlüssel K, Nonce N und dem Verschlüsselungsalgorithmus f zum Ciphertext C verschlüsselt.

2. Das Authentication Tag T wird mit Authentifizierungsalgorithmus h und Schlüssel K durch $h_K(N \mid\mid A \mid\mid C)$ berechnet, wobei A die zusätzlichen, nicht-verschlüsselten Daten (Header) darstellt.

3. Die sichere Nachricht M setzt sich schließlich so zusammen: $M = N \,||A\,||\,C\,||\,T$

In Abb. 2.2 ist die Zusammensetzung graphisch dargestellt. Es kann demnach zum Beispiel AES im *Cipher Block Chaining Modus* als Verschlüsselungsalgorithmus und eine HMAC-Konstruktion mit SHA-3 als Authentifizierungsalgorithmus in einem AEAD-Verfahren angewandt werden. Andere Methoden zur Konstruktion eines AEAD-Verfahens sind *EAX* und *GCM* [Rog02, S. 15].

Die Kapselung des gesamten AEAD-Verfahrens in Software-Bibliotheken trägt in der Praxis dazu bei, Fehler bei der Implementierung sicherer Kommunikation zu vermeiden. Denn schon kleine Details können große Auswirkungen auf die Sicherheit kryptographischer Methoden haben und nicht jeder Entwickler ist ein Experte auf diesem Gebiet.

2.2 RFID

Radio Frequency Identification (RFID) bezeichnet die elektronische Identitätsfindung über Funk. Dazu werden Objekte mit sogenannten Transpondern ausgestattet, die sich jederzeit per Radiowellen gegenüber RFID-Lesegeräten identifizieren können. Die Technik findet häufig Anwendung für Produktverfolgung im Prozessmanagement und als Authentifizierungsmethode zum Beispiel für den Gebäudezutritt. Der Standard RFID ist sehr allgemein formuliert und wird von anderen Technologien, wie zum Beispiel *Near Field Communication* (NFC), weiter spezifiziert.

Da RFID-Transponder sehr klein sind (< 1 cm) und dazu günstig produziert werden können, ist es möglich, selbst Massenware damit auszustatten. RFID-Transponder ersetzen in einigen Bereichen bereits Barcodes, da die Identifizierung so ohne Sichtkontakt erfolgen kann [HPP07, S. 2]. Abhängig von der Implementierung und den verwendeten Frequenzen kann die Identifizierung auch über größere Distanzen erfolgen (15m bei 2,45GHz [BO04, S. 29]). Somit findet die Technologie auch bei der Erfassung und Verfolgung von Fahrzeugen Anwendung, zum Beispiel zur Mautabrechnung bei LKW.

2.2.1 Funktionsweise

Ein RFID-System besteht aus drei Komponenten: dem Transponder, dem RFID-Lesegerät und einem Controller. Vom Transponder übermittelte Daten werden vom Lesegerät erfasst und an einen Controller weitergeleitet. Controller können Computer oder Mikroprozessoren sein, die die Auswertung der Daten übernehmen.

Ein Transponder besteht aus einer Antenne, einem Datenspeicher und einem digitalen Schaltkreis. Sobald er den Lesebereich eines RFID-Lesegeräts betritt, wird er mittels Radiosignal dazu aufgefordert, sich zu identifizieren. Der digitale Schaltkreis, bei komplexeren Anwendungen ein Mikrocontroller, registriert das Signal und antwortet mit den identitätsgebenden Daten, die sich auf dem Speicher befinden.

In der Regel haben RFID-Transponder keine eigene Energiequelle (z.B. Batterie), da sie in der Lage sind, ausreichend Energie aus dem elektromagnetischen Feld des RFID-Lesegeräts abzuführen, das auch zur Kommunikation genutzt wird. Solche Transponder nennt man passiv. Haben sie dagegen eine eigene Batterie oder sonstige Energiequelle, werden sie als aktive Transponder bezeichnet. Diese können zum Beispiel eine höhere Übertragungsreichweite oder komplexere Elektronik haben.

Der Speicher des Transponders ist oft Read Only Memory (ROM). Er ist also nur einmal bei der Herstellung mit wichtigen Daten beschreibbar. Dazu zählt die einzigartige Identifikationsnummer (ID) und domänenspezifische Daten, wie Produktionsdatum, Fabriknummer u.v.m. Bei der Verwendung von wiederbeschreibbaren Speichermedien (EEPROM oder Flash Memory) dagegen können neben den permanenten Daten auch andere Informationen temporär auf dem Transponder abgelegt werden. [HPP07]

Die zur Datenübermittlung genutzten Radiofrequenzen unterscheiden sich teilweise zwischen verschiedenen Ländern, als bewährt gelten jedoch 125 - 134 kHz und 13,56 MHz. Darüber hinaus werden noch weitere Frequenzen im ultrahochfrequenten Radiobereich, sowie im Mikrowellenbereich genutzt: 868 MHz, 915 MHz, 2,45 GHz und 5,8 GHz. [BO04, S. 29-30]

2.2.2 Sicherheit

Da die RFID-Technologie auch in sicherheitskritischen Systemen genutzt wird, kommen verschiedene Sicherheitsmaßnahmen zum Einsatz. Eine simple Maßnahme, um die eindeutige Identifizierung eines RFID-Tranponders zu ermöglichen, ist die international einheitliche Vergabe von ID-Nummern. Dies geschieht in Form des Electronic Product Code (EPC), ein spezieller Standard für die Identifizierung physischer Objekte.

Da die Authentifizierung die Hauptaufgabe eines RFID-Transponders ist, müssen Maßnahmen zur Vermeidung von Identitätsfälschung zum Einsatz kommen. Im Falle von unverschlüsselt übertragenen ID-Nummern an das Lesegerät könnte ein Angreifer diese problemlos mithören, oder sogar sich selbst als berechtigtes Lesegerät ausgeben.

Mithilfe von Challenge-Response-Verfahren kann Identitätsfälschung vermieden werden. Dazu sendet das Lesegerät einen Zeitstempel oder eine Zufallszahl an den Transponder, der diese dann verschlüsselt zurückschickt. Wenn der genutzte symmetrische Schlüssel nur authorisierten Parteien bekannt ist, wird der Transponder dadurch authentifiziert. Das Lesegerät (oder der Controller) kann die verschlüsselte Zahl leicht selbst entschlüsseln und die Gleichheit überprüfen. Das genaue Verfahren ist in der Norm ISO/IEC 9798-1 beschrieben. Auch die Authentifizierung des Lesegeräts gegenüber dem Transponder kann so erfolgen. Allerdings verfügen nur bestimmte Transponder-Modelle über diese kryptographischen Möglichkeiten. [BO04, S. 47-50]

3 Anforderungen

In diesem Kapitel werden die Anforderungen an das Abrechnungssystem gesammelt und formal festgehalten. Diese sollen die Evaluierung existierender Systeme hinsichtlich der Nutzbarkeit am LRZ ermöglichen. Darüber hinaus dienen die formalen Anforderungen als Leitfaden für das Design eines neuen Abrechnungssystems und schließlich als Basis für die Bewertung der Implementierung. Zunächst werden jedoch die voraussichtlichen Use Cases geschildert, um die Interaktionen zwischen Nutzer und System zu identifizieren. Darauf aufbauend können die funktionalen Anforderungen leicht erarbeitet werden. Zudem werden die Anforderungen an die Informationssicherheit des Systems und zuletzt noch weitere, allgemeine Anforderungen definiert.

3.1 Situation am Leibniz-Rechenzentrum

Das Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften ist der IT-Dienstleister für die Münchner Universitäten und Hochschulen. Zudem umfasst es ein Supercomputing-Zentrum (SuperMUC) und eine Vielzahl anderer wissenschaftlicher Einrichtungen. Es wurde 1962 gegründet und hat sich seitdem zu einem der größten Rechenzentren Europas entwickelt.

Jeder Mitarbeiter am LRZ besitzt einen persönlichen elektronischen Schlüssel, der ihm Zugang zu allen Bereichen ermöglicht, zu denen er befugt ist. Diese Schlüssel basieren auf der RFID-Technologie und sind somit auch für andere Zwecke einsetzbar. Der genaue Vorgang der sicheren Identifikation gegenüber einer gegebenen Tür (siehe *Challenge-Response-Verfahren* in Abschnitt 2.2.2) ist jedoch proprietär. Ebenso sind die geheimen Schlüssel der einzelnen Transponder nicht vom Hersteller in Erfahrung zu bringen und natürlich auch nicht auslesbar. Lediglich der generelle *Unique Identifier* (UID) kann mithilfe handelsüblicher RFID-Lesegeräte ausgelesen werden.

3.2 Use Cases

Im Folgenden werden die wichtigsten Anwendungsfälle des Systems geschildert. Sie orientieren sich an den intuitiv wichtigen Interaktionen, die mit einem solchen System zu erwarten sind. Die Anwendungsfälle beziehen sich auf die Aktionen zwei verschiedener Gruppen von Akteuren: Die Administratoren des Systems und die Nutzer. Während die Nutzer nur die grundlegenden Funktionen des Abrechnungssystems in Anspruch nehmen, erhalten die Administratoren Berechtigung zur Verwaltung des Abrechnungssystems und somit Zugriff auf erweiterte Funktionen.

Die folgenden Use Cases betrachten zwei getrennte Subsysteme: Einerseits die Kaffeemaschine inklusive ihrer lokalen Steuerungs-Hardware wie Mikrocontroller, RFID-Lesegerät und Bildschirm (die *Kaffeestation*) und andererseits die Webseite zur Verwaltung des Abrechnungssystems (das *Administrationstool*).

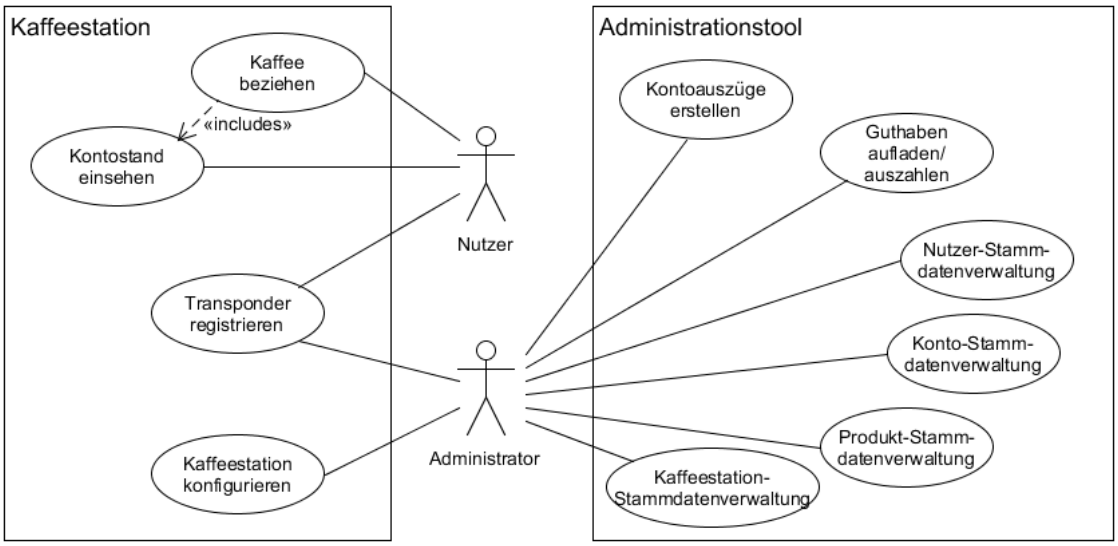

Abbildung 3.1: Use-Case-Diagramm des Abrechnungssystems.

Use Case 1: Kontostand einsehen

Beschreibung: Ein Nutzer möchte das Guthaben eines seiner Konten einsehen. Dazu scannt er den entsprechenden Transponder am RFID-Lesegerät einer Kaffeestation. Das aktuelle Guthaben des verknüpften Kontos erscheint auf dem zugehörigen Bildschirm.
Akteure: Nutzer
Einstieg: Ein Nutzer möchte das Guthaben eines seiner Konten einsehen.
Vorbedingungen:

- Der Nutzer ist registriert, besitzt ein aktives Konto und einen verknüpften Transponder.

Nachbedingungen:

- Der Nutzer wurde über sein aktuelles Guthaben informiert.

Normaler Ablauf:

1. Der Nutzer scannt seinen Transponder am RFID-Lesegerät einer Kaffeestation.
2. Der Bildschirm an der Kaffeemaschine zeigt den aktuellen Kontostand des mit dem Transponder verknüpften Kontos an.
3. Nach einigen Sekunden verschwindet das Guthaben vom Bildschirm und die Kaffeestation ist bereit für die nächste Interaktion.

Ausnahmefälle:

2a. Der Transponder kann keinem Konto zugeordnet werden. Der Bildschirm zeigt für einige Sekunden eine entsprechende Meldung an und geht zurück in den Ausgangszustand.

Use Case 2: Kaffee beziehen

Beschreibung: Ein Nutzer des Abrechnungssystems möchte einen Kaffee beziehen. Dazu scannt er seinen Transponder am RFID-Lesegerät einer Kaffeestation. Daraufhin wird ihm sein Guthaben angezeigt. Nun gibt er seine Bestellung direkt an der Kaffeemaschine ein. Der entsprechende Betrag wird von seinem Konto abgebucht und der Kaffee wird ausgegeben.
Akteure: Nutzer
Einstieg: Ein Nutzer möchte einen Kaffee kaufen.
Vorbedingungen:

- Der Nutzer ist registriert, besitzt ein aktives Konto und einen verknüpften Transponder.
- Das Guthaben auf seinem Konto reicht für die Bestellung aus.

Nachbedingungen:

- Der Nutzer hat den gewünschten Kaffee erhalten.
- Der Preis des Kaffees wurde vom Konto des Nutzers abgebucht.
- Eine permanent gespeicherte Transaktion dokumentiert den Vorgang in einem Transaktionslog.

Normaler Ablauf:

1. Der Nutzer scannt seinen Transponder am RFID-Lesegerät einer Kaffeestation.
2. Der Bildschirm an der Kaffeestation zeigt den aktuellen Kontostand des mit dem Transponder verknüpften Kontos an.
3. Der Nutzer gibt seine Bestellung an der Kaffeemaschine ein.
4. Bei ausreichendem Guthaben wird der gewünschte Kaffee ausgegeben.
5. Die Transaktion wird gespeichert und der zu zahlende Betrag vom Konto des Nutzers abgebucht.
6. Die Kaffeestation geht zurück in den Ausgangszustand und ist bereit für die nächste Interaktion.

Ausnahmefälle:

4a. Das Guthaben des mit dem Transponder verknüpften Kontos reicht nicht aus. Die Transaktion wird abgebrochen und gespeichert und eine Meldung auf dem Bildschirm informiert den Nutzer über den Vorgang.

4b. Der Kaffee konnte nicht ausgegeben werden. Das kann durch fehlendes Wasser oder Bohnen in der Kaffeemaschine verursacht werden oder durch einen technischen Defekt. Dann wird die Transaktion abgebrochen und gespeichert. Der Nutzer wird über den Bildschirm über den Abbruch informiert und gegebenenfalls dazu aufgefordert, Wasser oder Bohnen nachzufüllen.

Use Case 3: Transponder registrieren

Beschreibung: Ein neuer Transponder soll ins Abrechnungssystem aufgenommen werden, damit er zum Kaffeebezug genutzt werden kann. Die UID eines Transponders ist in der Regel jedoch nicht bekannt und kann nur durch das Auslesen des Transponders in Erfahrung gebracht werden. Deswegen sollen Transponder direkt durch Einscannen an einer Kaffeestation im Abrechnungssystem registriert werden können. Dazu scannt der Administrator zunächst einen Master-Transponder an einer Kaffeestation, wodurch ein Registrierungsmodus eingeleitet wird. Danach kann der neue Transponder eingescannt und somit registriert werden.

Akteure: Administrator
Einstieg: Ein neuer Transponder soll ins Abrechnungssystem aufgenommen werden, damit er zum Kaffeebezug genutzt werden kann.
Vorbedingungen:

- Der Administrator hat einen Master-Transponder bei sich.

Nachbedingungen:

- Der neue Transponder ist im System registriert und kann mit einem Konto verknüpft werden.

Normaler Ablauf:

1. Der Administrator scannt den Master-Transponder an einer Kaffeestation. Der Registrierungsmodus wird eingeleitet und eine entsprechende Meldung erscheint auf dem Bildschirm.
2. Der Administrator scannt den neuen Transponder an derselben Kaffeestation.
3. Die UID des Transponders wird angezeigt und die Registrierung wird durchgeführt.
4. Die Registrierung des Transponders wird von der Kaffeestation bestätigt.

Ausnahmefälle:

4a. Die UID (*Universal Identifier*) des Transponders wird bereits im System verwendet. Dann ist der Transponder entweder bereits registriert oder ein anderer Transponder nutzt dieselbe UID, was jedoch extrem unwahrscheinlich ist[1]. Die Kaffeestation zeigt eine entsprechende Fehlermeldung an und kehrt in den Ausgangszustand zurück.

Use Case 4: Kaffeestation konfigurieren

Beschreibung: Die Konfiguration einer Kaffeestation ist nötig, wenn eine neue Station in das Abrechnungssystem eingebunden werden soll oder der geheime Schlüssel, der zur Kommunikation verwendet wird, gewechselt werden soll. Dazu müssen zunächst die technischen Informationen auf den Mikrocontroller gespielt werden und danach die entsprechende Software für den regulären Betrieb.
Akteure: Administrator
Einstieg: Die Konfiguration einer neuen Kaffeestation soll erfolgen oder die einer alten geändert werden.
Vorbedingungen:

- Die Kaffeestation wurde im Administrationstool registriert und die technischen Daten, insbesondere der geheime Schlüssel, liegen vor.
- Der Administrator verfügt über die benötigte Hardware und Software zur Modifizierung des Mikrocontrollers.

Nachbedingungen:

- Der Mikrocontroller hat die gewünschte Konfiguration und ist einsatzbereit.

[1] UIDs sind zwar nicht global einzigartig, aber einzigartig hinsichtlich des Herstellers. Somit ist das Auftreten zwei identischer UIDs bei einer Länge von vier Byte und nur einer Handvoll verschiedener Hersteller extrem gering.

Normaler Ablauf:

1. Der Administrator schließt seine Hardware an den Mikrocontroller an.
2. Er führt die Programme zur Konfiguration des Mikrocontrollers aus.
3. Er spielt die Software für den regulären Betrieb auf den Mikrocontroller.
4. Der Mikrocontroller wird an die zusätzliche Hardware (Bildschirm, RFID-Lesegerät), die Kaffeemaschine und das lokale Netz angeschlossen und neu gestartet. Falls die entsprechende Konfiguration im Administrationstool vorliegt, ist die Kaffeestation nun einsatzbereit.

Use Case 5: Guthaben aufladen/auszahlen

Beschreibung: Ein Nutzer möchte das Guthaben eines seiner Konten aufladen oder ausgezahlt bekommen. Der Nutzer kontaktiert einen Administrator persönlich und der Geldbetrag wird ausgetauscht. Dann führt der Administrator die Aufladung/Auszahlung im Administrationstool durch.
Akteure: Administrator, Nutzer
Einstieg: Ein Nutzer hat einen Administrator darüber informiert, dass er das Guthaben eines seiner Konten aufladen/abheben möchte.
Vorbedingungen:

- Der Administrator ist als solcher registriert und besitzt somit die nötigen Berechtigungen.
- Der Nutzer ist registriert und besitzt ein aktives Konto.

Nachbedingungen:

- Das Guthaben des gewünschten Kontos wurde um den angegebenen Betrag erhöht oder verringert.
- Eine permanent gespeicherte Transaktion dokumentiert den Vorgang in einem Transaktionslog.

Normaler Ablauf:

1. Der Geldbetrag wird zwischen Nutzer und Administrator in bar ausgetauscht.
2. Der Administrator meldet sich im Administrationstool an und navigiert auf eine entsprechende Seite.
3. Der Administrator gibt den gewünschten Betrag und optional einen Betreff an und bestätigt die Transaktion.
4. Die Transaktion wird ausgeführt und in einem Transaktionslog permanent gespeichert. Das Administrationstool bestätigt den Abschluss der Transaktion.

Use Case 6: Kontoauszüge erstellen

Beschreibung: Der Administrator des Abrechnungssystems möchte die Nutzer regelmäßig über ihre Aktivität informieren. Dazu erstellt er im Administrationstool Kontoauszüge, die alle relevanten Informationen zu den Nutzerkonten in einem definierten Zeitraum enthalten.
Akteure: Administrator
Einstieg: Ein Administrator möchte einen Kontoauszug zu einem bestimmten Konto des Abrechnungssystems erstellen.
Vorbedingungen:

- Der Administrator ist als solcher registriert und besitzt somit die nötigen Berechtigungen.

Nachbedingungen:
- Der Administrator ist im Besitz eines Kontoauszugs des gewünschten Kontos.

Normaler Ablauf:
1. Der Administrator meldet sich im Administrationstool an und navigiert auf eine entsprechende Seite.
2. Der Administrator gibt das Konto sowie den gewünschten Zeitraum an.
3. Der Kontoauszug wird dem Administrator zum Download bereitgestellt.

Use Case 7: Nutzer-Stammdatenverwaltung

Beschreibung: Ein Administrator möchte die Stammdaten eines Nutzers ändern, ein neues Nutzerprofil anlegen oder ein altes löschen. Informationen wie Vorname, Nachname und E-Mailadresse können modifiziert werden.
Akteure: Administrator
Vorbedingungen:
- Der Administrator ist als solcher registriert und besitzt somit die nötigen Berechtigungen.

Nachbedingungen:
- Ein Nutzerprofil wurde angelegt, geändert oder entfernt.

Normaler Ablauf:
1. Der Administrator meldet sich im Administrationstool an und navigiert auf eine entsprechende Seite.
2. Der Administrator wählt die gewünschte Aktion: Anlegen, Ändern oder Entfernen eines Nutzerprofils.
3. Der Administrator gibt ggf. weitere Informationen zur auszuführenden Aktion an und bestätigt diese. Beim Anlegen eines Nutzerprofils müssen Vor- und Nachname angegeben werden, bei der Änderung eines Profils die neuen Informationen.
4. Der Administrator erhält eine Rückmeldung über die ausgeführte Aktion vom Administrationstool.

Use Case 8: Konto-Stammdatenverwaltung

Beschreibung: Ein Administrator möchte die Stammdaten eines Kontos ändern, ein neues Konto anlegen oder ein altes löschen. Informationen wie Kontoname, Besitzer und Guthaben können modifiziert werden.
Akteure: Administrator
Vorbedingungen:
- Der Administrator ist als solcher registriert und besitzt somit die nötigen Berechtigungen.

Nachbedingungen:
- Ein Konto wurde angelegt, geändert oder entfernt.

Normaler Ablauf:

1. Der Administrator meldet sich im Administrationstool an und navigiert auf eine entsprechende Seite.
2. Der Administrator wählt die gewünschte Aktion: Anlegen, Ändern oder Entfernen eines Kontos.
3. Der Administrator gibt ggf. weitere Informationen zur auszuführenden Aktion an und bestätigt diese.
4. Der Administrator erhält eine Rückmeldung über die ausgeführte Aktion vom Administrationstool.

Use Case 9: Produkt-Stammdatenverwaltung

Beschreibung: Ein Administrator möchte die Stammdaten eines Produkts ändern, ein neues Produkt anlegen oder ein altes löschen. Informationen wie Produktname und Preis können modifiziert werden.
Akteure: Administrator
Vorbedingungen:

- Der Administrator ist als solcher registriert und besitzt somit die nötigen Berechtigungen.

Nachbedingungen:

- Ein Produkt wurde angelegt, geändert oder entfernt.

Normaler Ablauf:

1. Der Administrator meldet sich im Administrationstool an und navigiert auf eine entsprechende Seite.
2. Der Administrator wählt die gewünschte Aktion: Anlegen, Ändern oder Entfernen eines Produkts.
3. Der Administrator gibt ggf. weitere Informationen zur auszuführenden Aktion an und bestätigt diese.
4. Der Administrator erhält eine Rückmeldung über die ausgeführte Aktion vom Administrationstool.

Use Case 10: Kaffeestation-Stammdatenverwaltung

Beschreibung: Ein Administrator möchte die Stammdaten einer Kaffeestation ändern, eine neue Kaffeestation anlegen oder eine alte löschen. Informationen wie Stationsname und Ort können modifiziert werden.
Akteure: Administrator
Vorbedingungen:

- Der Administrator ist als solcher registriert und besitzt somit die nötigen Berechtigungen.

Nachbedingungen:

- Eine Kaffeestation wurde angelegt, geändert oder entfernt.

Normaler Ablauf:

1. Der Administrator meldet sich im Administrationstool an und navigiert auf eine entsprechende Seite.
2. Der Administrator wählt die gewünschte Aktion: Anlegen, Ändern oder Entfernen einer Kaffeestation.
3. Der Administrator gibt ggf. weitere Informationen zur auszuführenden Aktion an und bestätigt diese.
4. Der Administrator erhält eine Rückmeldung über die ausgeführte Aktion vom Administrationstool.

3.3 Funktionale Anforderungen

Auf der Grundlage der zuvor geschilderten Use Cases werden nun die konkreten funktionalen Anforderungen an das Abrechnungssystem erarbeitet. Diese spezifizieren die erwünschten Eigenschaften und Funktionen, die das System erbringen muss.

FA1: Guthabeneinsicht an Kaffeestation

Kaffeestationen müssen dazu in der Lage sein, zu jedem registrierten Transponder das Guthaben des assoziierten Kontos abzurufen und dem Nutzer zu präsentieren (siehe Use Case 1).

FA2: Kaffeebezug an Kaffeestation

Nutzer sollen mithilfe ihres registrierten Transponders an jeder beliebigen Kaffeestation Kaffee beziehen können, sofern ihr Guthaben ausreicht (siehe Use Case 2).

FA3: Kontrolle über Kaffeeausgabe

Das Abrechnungssystem muss die Funktionen der Kaffeemaschinen soweit kontrollieren, dass der Bezug von Kaffee nur über den durch das System definierten Ablauf möglich ist. Das heißt, alle unautorisierten Kaffeebezüge müssen blockiert werden können (siehe Use Case 2).

FA4: Kaffeestationen geben Rückmeldung an Nutzer

Die Kaffeestationen sollen dazu in der Lage sein, die Nutzer über die aktuelle Situation und über die Ergebnisse einer Aktion zu informieren. Dazu werden Meldungen wie „*Verbindung zum Server wird aufgebaut.*“oder „*Produktkauf abgeschlossen.*“über einen Bildschirm angezeigt (siehe Use Cases 1, 2 und 3).

FA5: Registrierung eines Transponders an Kaffeestation

Transponder können direkt durch Einscannen an einer Kaffeestation im Abrechnungssystem registriert werden (siehe Use Case 3).

FA6: Verlässliche Kontostände

Die Kontostände spiegeln zu jeder Zeit das wirkliche Guthaben eines Kontos wieder und werden laufend aktualisiert. Kontostände können nur bis auf ein definiertes Minimum abfallen, bevor weitere Abbuchungen abgelehnt werden (siehe Use Cases 3 und 6).

FA7: Buchführung über Transaktionen

Jede Transaktion (also Gutschrift oder Abbuchung) muss verlässlich und permanent dokumentiert werden (siehe Use-Cases 2 und 5). Anhand eines so entstehenden Transaktionslogs müssen alle aktuellen Kontostände korrekt rekonstruiert werden können.

FA8: Anlegen, Modifizieren und Entfernen der Stammdaten

Das Abrechnungssystem muss Möglichkeiten zur Einsicht und Änderungen sämtlicher Stammdaten, wie Nutzerprofile, Konten, Transponder und Kaffeestationen, bieten. Zusätzliche Funktionen sollen die Registrierung neuer Nutzer, Konten etc. so einfach wie möglich gestalten (siehe Use-Cases 7-10).

FA9: Kontoauszüge erstellen

Zu jedem Konto sollen Kontoauszüge erstellt werden können, die alle Transaktionen in einem definierten Zeitraum enthalten. Das schafft Transparenz für die Nutzer hinsichtlich ihrer Aktivität und der Entwicklung ihres Guthabens (siehe Use-Case 6).

FA10: Archivierung der Transaktionsdaten

Die Transaktionsdaten sollen exportiert werden können, um sie an anderer Stelle langfristig aufzubewahren.

FA11: Rechnungsschnitt

Es soll jederzeit ein Rechnungsschnitt angesetzt werden können, d.h. alle Transaktionen und alten Stammdaten können ab einem definierten Zeitpunkt rückwirkend gelöscht werden, ohne das es zu Inkonsistenzen der Datenbank oder zu anderen Problemen führt.

FA12: Integration beliebig vieler Kaffeemaschinen

Das Abrechnungssystem kann eine hinreichend große Zahl an räumlich verteilten Kaffeemaschinen einbinden und zentral verwalten.

FA13: Statusabfragen der Kaffeestationen

Das Administrationstool sollte jederzeit aktuelle Informationen zu allen Kaffeestationen bereitstellen können. Darunter aktueller Status und letzte Transaktion.

ID	Name der Anforderung
FA1	Guthabeneinsicht an Kaffeestation
FA2	Kaffeebezug an Kaffeestation
FA3	Kontrolle über Kaffeeausgabe
FA4	Kaffeestationen geben Rückmeldung an Nutzer
FA5	Registrierung eines Transponders an Kaffeestation
FA6	Verlässliche Kontostände
FA7	Buchführung über Transaktionen
FA8	Anlegen, Modifizieren und Entfernen der Stammdaten
FA9	Kontoauszüge erstellen
FA10	Archivierung der Transaktionsdaten
FA11	Rechnungsschnitt
FA12	Integration beliebig vieler Kaffeemaschinen
FA13	Statusabfragen der Kaffeestationen

Tabelle 3.1: Funktionale Anforderungen an das Abrechnungssystem.

3.4 Informationssicherheit

Wie in Abschnitt 2.1.1 dargelegt, spricht man bei der Informationssicherheit von sogenannten Schutzzielen, die es zu erreichen gilt. Grundsätzlich soll das Abrechnungssystem natürlich die Ziele Vertraulichkeit, Integrität, Authentizität, Verbindlichkeit und Verfügbarkeit sicherstellen. Im Folgenden werden die Anforderungen noch einmal konkretisiert.

3.4.1 Annahmen

Die genannten Schutzziele sind unter folgenden Bedingungen zu wahren:

- **Angreifer können die Kommunikation der Komponenten manipulieren.** Informationssicherheit soll auch in dem Fall gegeben sein, dass ein Angreifer vollkommene Freiheit hat, was das Belauschen, Blocken oder Ändern der Nachrichten zwischen den Systemkomponenten betrifft.

- **Angreifer haben keinen physischen Zugriff auf die Systemkomponenten.** Falls ein Angreifer physischen Zugriff auf Systemkomponenten erlangt, ist die Gewährleistung der Informationssicherheit in der Regel nicht möglich. Denn dann können sicherheitsrelevante Daten wie geheime Schlüssel direkt von der Hardware gelesen werden. Beispielsweise lässt sich der permanente Speicher eines Mikrocontrollers mit entsprechender Ausrüstung leicht auslesen.

3.4.2 Angriffsmodelle

Im Folgenden werden einige Methoden vorgestellt, die häufig genutzt werden, um in Computersysteme einzudringen. Das Abrechnungssystem muss diesen ausnahmslos standhalten. Alle Angriffsmodelle beziehen sich auf eine laufende verschlüsselte Kommunikation zwischen den zwei Parteien *Alice* und *Bob* und den Angreifer *Eve*.

Chosen Plaintext Attack

Bei einer *Chosen Plaintext Attack* wird davon ausgegangen, dass *Eve* die Möglichkeit hat, Ciphertexte beliebiger Nachrichten unter dem aktuellen Schlüssel zu erlangen. Das Ziel ist es dann, mithilfe der daraus gewonnen Informationen Rückschlüsse auf diesen Schlüssel oder die Klartexte der zwischen *Alice* und *Bob* ausgetauschten Nachrichten zu ziehen. Ein solcher Angriff hat also das Ziel, den genutzten Verschlüsselungsalgorithmus selbst zu brechen. In der Praxis wird das verhindert, in dem jede Nachricht eine zufällige Komponente erhält, die *Nonce*. Da sie bewirkt, dass selbst aus identischen Nachrichten unterschiedliche Ciphertexte hervorgehen, kann ein Angreifer sehr viel weniger Informationen durch die Analyse des Nachrichtenverkehrs erhalten. [vT11, S. 205-206]

Man in the Middle Attack

Eine *Man in the Middle Attack* basiert auf der Notwendigkeit eines Schlüsselaustauschs bei asymmetrischen Verschlüsselungsverfahren. Möchten *Alice* und *Bob* kommunizieren, müssen sie zunächst den öffentlichen Schlüssel des jeweils anderen kennen, um sicher verschlüsselte und signierte Nachrichten zu verschicken können. Indem *Eve* sich *Alice* gegenüber als *Bob* ausgibt und andersherum, kann sie sich als Bindeglied der Kommunikation zwischen den beiden etablieren und erlangt so die volle Kontrolle über den Kommunikationskanal. Wenn *Alice* eine Nachricht an *Eve* schickt, in dem Glauben sie gehe an *Bob*, kann *Eve* diese beliebig verändern und weiterleiten oder direkt anstelle von *Bob* antworten. Vermieden werden kann diese Methode, indem *Alice* und *Bob* ihre öffentlichen Schlüssel über einen sicheren Kanal austauschen oder diese von einer vertrauenswürdigen Instanz beziehen (wie einer *Certificate Authority*). [vT11, S. 759]

Replay Attack

Bei einer *Replay Attack* benötigt *Eve* die Fähigkeit, Nachrichten abzufangen, zu ändern, zu verzögern oder auch zu duplizieren. Sie versucht durch das erneute oder verzögerte Verschicken mitgehörter Nachrichten das Verhalten von *Alice* und *Bob* zu beeinflussen. Angenommen *Alice* und *Bob* sichern ihre Kommunikation durch ein *AEAD*-Verfahren und *Alice* verschickt Anweisungen an *Bob*, die dieser immer ausführt, wenn die Nachricht korrekt ist (MAC ist gültig). Dann kann *Eve* eine zuvor verschickte Nachricht einfach erneut an *Bob* schicken, der die Fälschung ohne zusätzliche Maßnahmen nicht bemerkt. So können zusätzliche Aktionen herbeigeführt oder deren Reihenfolge verändert werden. Schutz gegen diesen Angriff bietet die Nutzung von Sitzungen und Nachrichtenzählern. Voraussetzung für die Akzeptanz ist dann, dass der Zähler einer empfangenen Nachricht größer als der der vorherigen Nachrichten ist und zur aktuellen Sitzung gehört. [vT11, S. 1042]

3.4.3 Anforderungen an die Informationssicherheit

Die Anforderungen an die Informationssicherheit des Abrechnungssystems basieren auf den zuvor beschriebenen Annahmen und Angriffsmodellen. Sie bilden die Grundlage für den Entwurf eines sicheren Systems.

SA1: Kommunikation mit Kaffeestationen kann nicht abgehört werden

Die Verschlüsselung des Kommunikationskanals zwischen Kaffeestationen und Server ist so zu wählen, dass ein Informationsgewinn durch Abhören der Nachrichten praktisch unmöglich wird. Konkret soll die Kommunikation also hinsichtlich Chosen Plaintext Attacken sicher sein.

SA2: In Kommunikation mit Kaffeestationen kann nicht eingegriffen werden

Es darf für einen Angreifer nicht möglich sein, in irgendeiner Weise mit einer Kaffeestation oder dem Server zu kommunizieren, ohne dass diese es bemerken und solche Nachrichten sofort verwerfen.

SA3: Zugriff auf Nutzerfunktionen wird nur durch registrierte Transponder gewährt

Es ist nicht möglich, auf Nutzerfunktionen wie den Bezug eines Kaffees oder das Einsehen eines Kontostands zuzugreifen, ohne in Besitz des entsprechenden Transponders zu sein. Insbesondere können Transponder nicht dupliziert oder simuliert werden.

SA4: Zugriff auf das Administrationstool ist den Administratoren vorbehalten

Nur registrierte Administratoren können auf das Administrationstool zugreifen und die Verwaltungsfunktionen nutzen.

SA5: Kommunikation mit Administrationstool kann nicht abgehört werden

Die Verschlüsselung des Kommunikationskanals zwischen Administrationstool und Server ist so zu wählen, dass ein Informationsgewinn durch Abhören der Nachrichten praktisch unmöglich wird. Konkret soll die Kommunikation also hinsichtlich Chosen Plaintext Attacken sicher sein.

SA6: In Kommunikation mit Administrationstool kann nicht eingegriffen werden

Es darf für einen Angreifer nicht möglich sein, in irgendeiner Weise mit dem Administrationstool oder dem Server zu kommunizieren, ohne dass diese es bemerken und solche Nachrichten sofort verwerfen.

3.5 Allgemeine Anforderungen

Gewisse Eigenschaften gelten als allgemein vorteilhaft für jede Art von Computersystem. Im Folgenden sind diejenigen Eigenschaften erklärt, die für das geplante Abrechnungssystem am wichtigsten sind.

ID	Name der Anforderung
SA1	Kommunikation mit Kaffeestationen kann nicht abgehört werden
SA2	In Kommunikation mit Kaffeestationen kann nicht eingegriffen werden
SA3	Zugriff auf Nutzerfunktionen wird nur durch registrierte Transponder gewährt
SA4	Zugriff auf das Administrationstool ist den Administratoren vorbehalten
SA5	Kommunikation mit Administrationstool kann nicht abgehört werden
SA6	In Kommunikation mit Administrationstool kann nicht eingegriffen werden

Tabelle 3.2: Anforderungen an die Informationssicherheit des Abrechnungssystems.

AA1: Benutzerfreundlichkeit

Da viele Personen das Abrechnungssystem regelmäßig nutzen werden, ist die Benutzerfreundlichkeit in diesem Fall nicht zu vernachlässigen. Dazu gehört einerseits ein intuitiv verständliches User-Interface am Kaffeeautomaten sowie Rückmeldungen zu Nutzereingaben, andererseits auch effektive Möglichkeiten zur Administration des Systems.

AA2: Robustheit

Ein robustes System zeichnet sich durch Widerstandsfähigkeit gegenüber fehlerhaften Inputs, internen Fehlfunktionen, oder gar teilweisen Ausfällen eigener Komponenten aus. Sie ist extrem wichtig für das Abrechnungssystem, weil ein Ausfall zu Zeit- und Geldverlusten seitens der Nutzer führen kann und außerdem der Service (den es eigentlich vereinfachen soll) dadurch komplett blockiert werden könnte.

AA3: Skalierbarkeit

Skalierbarkeit bezeichnet die Möglichkeit zur Leistungssteigerung eines Computersystems durch Bereitstellung zusätzlicher Ressourcen zu einem beliebigem Zeitpunkt. Zwar ist zur Zeit der Entwicklung des Abrechnungssystems die Anwendung auf einen Kaffeevollautomaten begrenzt, doch zukünftig sollen noch beliebig viele weitere Kaffeemaschinen angebunden werden. Deswegen ist die Skalierbarkeit des Abrechnungssystems eine zentrale Anforderung.

Allerdings muss die gesamte Systemarchitektur darauf abgestimmt sein. So muss der Abrechnungsserver entweder leistungsstark genug sein, um eine Großzahl an Anfragen in der Sekunde zu verarbeiten, oder entsprechende Möglichkeiten zur künftigen Erweiterung bieten (gleiches gilt für die Datenbank). Genauso müssen Systemparameter einen Anstieg der Nutzer- sowie Kaffeemaschinenanzahl zulassen (z.B. der Definitionsbereich von IDs).

AA4: Modularität

Modularität ist eine Anforderung, die an fast jede Art von Softwaresystem gestellt wird. Ein modular aufgebautes System kapselt nicht nur die Komplexität einzelner Aufgabengebiete, was den Überblick enorm vereinfacht, sondern kapselt genauso die Auswirkungen von Fehlern

innerhalb eines Moduls. Wenn jede Komponente des Systems seine Inputs und Outputs kontrolliert, sind weitreichende Systemausfälle zu verhindern.

Außerdem vereinfacht Modularität die zukünftige Erweiterung des Systems. Zum Beispiel werden künftige Kaffeemaschinen nicht zwangsläufig vom selben Hersteller erworben. Also muss die Schnittstelle zur Kaffeemaschine austauschbar sein. Darüber hinaus können die Kontostände des Systems für andere Anwendungen zugänglich gemacht werden, wenn die Schnittstelle zum Server allgemein gestaltet wird.

AA5: Kosteneffizienz

Bei der Planung und Umsetzung eines Systems muss immer die Verhältnismäßigkeit einer Lösung betrachtet werden. So können die meisten Probleme durch teuerste Hardware und übertriebene Systemarchitektur gelöst werden, doch dann steigen auch die Kosten ins Unermessliche. Um effiziente Lösungen zu erarbeiten ist es unerlässlich, auch den Preis derselben im Auge zu behalten. Für das Abrechnungssystem dieser Arbeit bedeutet das, die Kosten sollten nur einen Bruchteil derer eines käuflichen, professionellen Systems ausmachen.

ID	Name der Anforderung
AA1	Benutzerfreundlichkeit
AA2	Robustheit
AA3	Skalierbarkeit
AA4	Modularität
AA5	Kosteneffizienz

Tabelle 3.3: Allgemeine Anforderungen an das Abrechnungssystem.

4 Vergleichbare Systeme

Dieses Kapitel befasst sich mit bereits realisierten Lösungen zur Abrechnung von Serviceleistungen im Allgemeinen und der Abrechnung von Kaffee im Speziellen. Dazu werden drei Projekte betrachtet, die interessante Entwürfe von Abrechnungssystemen präsentieren.

4.1 Sharepresso

Das Computermagazin *c't* beschreibt im Artikel *Sharespresso - NFC-Bezahlsystem für Kaffeevollautomaten* vom Januar 2016 [Sie16] die Implementierung eines Abrechnungssystems auf einem Arduino Uno (siehe Arduino in Abschnitt 5.3.1). Es nutzt ein RFID-Lesegerät, um Nutzer zu identifizieren, und einen 16x2-Zeichen-Bildschirm für Systemmeldungen. Die genutzte Kaffeemaschine ist ein Jura X7 Kaffeevollautomat.

Das komplette Abrechnungssystem läuft auf einem Mikrocontroller, der direkt an eine Kaffeemaschine angeschlossen ist. Persistente Daten, wie Nutzer-IDs und Guthaben, werden auf dem integrierten EEPROM abgelegt. Das anschließen weiterer Kaffeemaschinen an das System ist nicht möglich.

Bewertung

Obwohl das *Sharespresso*-System auf nur eine Kaffeemaschine ausgelegt ist, und somit der funktionalen Anforderung FA12 nicht nachkommt (siehe Abschnitt 3.3), bietet es einige interessante Teillösungen für das Abrechnungssystem dieser Arbeit. So demonstriert es die Machbarkeit, einen Jura-Kaffeevollautomaten von einem Mikrocontroller aus fernzusteuern. Darüber hinaus stellt es den Arduino Uno als eine geeignete Plattform dar.

4.2 Bedienung und Verwaltung von Haushaltsgeräten

In dem Forschungsprojekt *Bedienung und Verwaltung von Haushaltsgeräten mittels NFC am Beispiel eines Kaffeevollautomaten* von der Hochschule für Technik und Wirtschaft Berlin [Roh12] wird ein Abrechnungssystem für Haushaltssysteme implementiert, das vor allem die Kompatibilität mit Kaffeemaschinen beliebiger Hersteller zum Ziel hat. Auch dieses System setzt auf einen Arduino für die Kommunikation mit der Kaffeemaschine und NFC (ein spezieller RFID Standard) als Identifikationsmethode. Ein zentraler Server übernimmt hier die Abrechnungsfunktionalität, die Kommunikation mit dem Client erfolgt jedoch unverschlüsselt. Die große Kompatibilität zu verschiedenen Kaffeemaschinen dieses Abrechnungssystems kommt dadurch zu Stande, dass keine integrierte Schnittstelle genutzt wird, sondern die Kommunikation direkt über an die Kaffeemaschine angelötete Kabel abläuft. Die Kabel sind im Gehäuse mit den Kontakten der Knöpfe verbunden, wodurch manuelle Knopfdrücke simuliert werden können.

Bewertung

Dieses System demonstriert die zentrale Verwaltung mehrerer Kaffeemaschinen durch einen Applikationsserver. Für die Implementierung des Servers wurde das Web-Framework *Django* genutzt. Es vereinfacht die Entwicklung eines Applikationsservers durch das Bereitstellen eines *Object Relational Mapper* (siehe *Django* in Abschnitt 5.3.2), eines URL-Parsers und vielen weiteren Hilfen enorm. Die Hardware-Manipulation an der Kaffeemaschine hingegen ist für das Abrechnungssystem dieser Arbeit uninteressant und die unverschlüsselte Kommunikation der Systemkomponenten verstößt direkt gegen die zuvor definierten Sicherheitsanforderungen (siehe FA1, FA2, FA5, FA6).

4.3 Protokollierung und Zugangssteuerung von Industriegeräten

Die Bachelorarbeit *Entwicklung eines web-basierten Verwaltungstools zur Protokollierung und Zugangssteuerung von Industriegeräten* von der Leopold-Franzens-Universität Innsbruck [Fal15] stellt ein umfangreiches Abrechnungssystem vor, das Server-seitig auf den JavaScript Technologien NodeJS und AngularJS und Client-seitig auf einem Raspberry Pi basiert. Die Client-Server-Kommunikation kann somit über Protokolle wie HTTPS und JSON ablaufen, die relativ hohe Ansprüche an die Rechenleistung des Client stellen. Kompatibilität zu verschiedenen Kaffeemaschinen wird durch ein Mini-Coffee-Interface erreicht, das jedoch mehrere Hundert Euro kostet[1].

Bewertung

Das System aus dieser Arbeit bietet einen sehr großen Funktionsumfang und komfortable Verwaltungsmöglichkeiten. Diese stützen sich jedoch teilweise auf die Möglichkeiten der Raspberry Pi Clients. Einen vergleichbaren Funktionsumfang und ein ebenbürtiges Sicherheitsniveau auf Basis von Mikrocontrollern als Clients zu erreichen, ist das Ziel dieser Arbeit. Die große Kompatibilität, die das Mini-Coffee-Interface ermöglicht, soll jedoch durch andere, kosteneffizientere Mittel erreicht werden (siehe AA5).

[1] Jura Online-Store: http://www.jurastore.at/de_at/jura-schanktool-silexia-mini-coffee-interface-seriell.html

5 Entwurf

Dieses Kapitel befasst sich mit dem Entwurf des Abrechnungssystems auf konzeptioneller Ebene. Es wird auf die Architektur, die Kommunikationsmethoden und auf die gewählten Technologien eingegangen. Die konkrete Implementierung dieses Entwurfs wird in Kapitel 6 dargelegt.

Auf einen Überblick über die Architektur des Abrechnungssystems folgt die Betrachtung der Kommunikationssicherheit des Systems. Dazu wird für jeden Kommunikationskanal eine passende Methode zur Sicherung präsentiert und auf deren Vor- und Nachteile eingegangen. Als nächstes werden die Systemkomponenten vorgestellt und die jeweiligen Aufgaben definiert. Darauf folgt eine Untersuchung einiger zentraler Abläufe, die aus dem Verhalten des Systems hervorgehen. Zuletzt werden noch passende Technologien für die einzelnen Systemkomponenten präsentiert.

5.1 Systemarchitektur

Die Realisierung eines jeden Computersystems erfordert zunächst den Entwurf eines Modells, das sich auf eine Architektur festlegt und allen Systemkomponenten klar definierte Aufgaben zuweist. Auch relevante Sicherheitsfragen müssen im Rahmen dieses Modells beantwortet werden. Dieser Abschnitt präsentiert ein solches Modell für das zu realisierende Abrechnungssystem.

5.1.1 Überblick

In den vorigen Kapiteln ist klar geworden, dass das Abrechnungssystem im Kern auf einem zentralen Server implementiert werden soll. Das Einbinden einer Vielzahl an räumlich verteilten Kaffeemaschinen kann nur so sinnvoll realisiert werden [FA12]. Das impliziert eine klassische Client-Server-Architektur, wobei zwei verschiedene Arten von Clients zum Einsatz kommen. Zum einen fungieren die Mikrocontroller, die jeweils eine Kaffeemaschine steuern, als Client (ab sofort *Kaffee-Clients* genannt), zum anderen soll der Administrator über ein Online-Administrationstool das System verwalten können (ab sofort *Webclient*), um den Anforderungen FA8 - FA11 gerecht zu werden. Der zentrale Server, im Folgenden *Backend* genannt, besteht selbst aus mehreren Komponenten: einem Webserver, der die eingehende Kommunikation kontrolliert und reguliert, einem Applikationsserver, der die eigentliche Geschäftslogik beinhaltet, und einem Datenbankserver. Die Systemarchitektur ist in Abb. 5.1 visualisiert.

Wie zuvor erklärt, können Mikrocontroller, im Gegensatz zu leistungsstärkeren vollständigen Computern, die für das HTTPS-Protokoll notwendigen Verschlüsselungsalgorithmen nicht bewältigen. Die Kommunikation mit diesen wird stattdessen über HTTP ablaufen, wobei der Inhalt durch effizientere Verschlüsselungsverfahren vor Angreifern geschützt wird. Diese Verfahren müssen den Anforderungen an die Informationssicherheit nachkommen (siehe Abschnitt 3.3). Die Kommunikation mit dem Webclient kann dagegen problemlos über

HTTPS gesichert werden. Der Applikationsserver muss also zwei verschiedene Schnittstellen bereitstellen, um sowohl mit den Mikrocontrollern, als auch mit gewöhnlichen Browsern kommunizieren zu können. Für die Kommunikation mit dem Webclient wird das *Web-Interface* zuständig sein und die Kommunikation mit den Mikrocontrollern wird über das *Controller-Interface* erfolgen. Um dem Prinzip der Modularität nachzukommen, wird die Abrechnungsfunktionalität, d.h. die Kontoführung und alle Verwaltungsmaßnahmen, die die Datenbank manipulieren, in einer separaten Systemkomponente gekapselt. Die unmittelbare Abrechnungsfunktionalität wird also unabhängig von externen Technologien und Kommunikationsmethoden implementiert und ist somit leicht in beliebige Abläufe einzubinden. Im Folgenden wird diese zentrale Systemkomponente *Abrechnungskern* genannt. Diese drei Komponenten sind Teil derselben Serverapplikation und bilden zusammen den *Applikationsserver*.

5.1.2 Kommunikationssicherheit

Die Sicherheit des Systems, im Sinne der Informationssicherheit (*Security*, nicht *Safety*), steht im Mittelpunkt dieser Arbeit. Da ein Großteil der Kommunikation von Systemkomponenten untereinander über ein unsicheres Medium abläuft, muss die Sicherheit an diesen Stellen durch Authentifizierungs- und Identifizierungsverfahren, sowie Verschlüsselung garantiert werden. In diesem Abschnitt werden Maßnahmen zur Sicherung des Systems für jeden Kommunikationskanal gewählt.

Webclient - Backend

Die Kommunikation zwischen dem Browser, der den Webclient darstellt, und dem Backend ist leicht durch die Nutzung des HTTPS-Protokolls zu sichern. Die Identifizierung und Authentifizierung des Backend wird über den TLS-Handshake sichergestellt, der auf das von einer Certificate-Authority signierte Server-Zertifikat zurückgreift [TR01, S. 31-34]. In umgekehrter Richtung muss der Nutzer des Webclients (der Administrator) seine Authentizität auch gegenüber dem Backend beweisen. Dazu wird ein Passwort-System auf Seite des Backend implementiert. Der Nutzer kann sich dann durch Angabe von Nutzername und Passwort anmelden und erhält für eine Sitzung Zugriff auf die Administrationsfunktionen. Somit werden die Sicherheitsanforderungen SA4 und SA6 erfüllt und insbesondere ist dieser Kommunikationskanal gegen *Man-in-the-Middle-Attacken* geschützt.

Im Rahmen der Authentifizierung wird auch ein Sitzungsschlüssel ausgetauscht, der zur symmetrischen Verschlüsselung der weiteren Kommunikation genutzt wird. Somit ist auch die Vertraulichkeit der Kommunikation [SA5] garantiert. Die verwendeten Verschlüsselungsmethoden sind robust gegenüber *Chosen-Plaintext-Attacken* und verwenden Nachrichtenzähler und *MACs*, um *Replay Attacken* zu verhindern. [TR01, S. 78]

Kaffee-Client - Backend

Wie zuvor erwähnt, ist eine Implementierung des HTTPS-Protokolls auf einem Mikrocontroller nicht praktikabel. Die Speicherkapazität für den Programmcode liegt hier typischerweise bei weit weniger als einem Megabyte, eine minimale TLS-Implementierung ist um ein Vielfaches größer.

Alternativ ist der Entwurf eines eigenen asymmetrischen Verfahrens denkbar, das RSA (oder ähnliche Algorithmen) in periodischen Abständen nutzt, um symmetrische Schlüssel zwischen Kaffee-Client und Backend auszutauschen. Mit diesen symmetrischen Schlüsseln

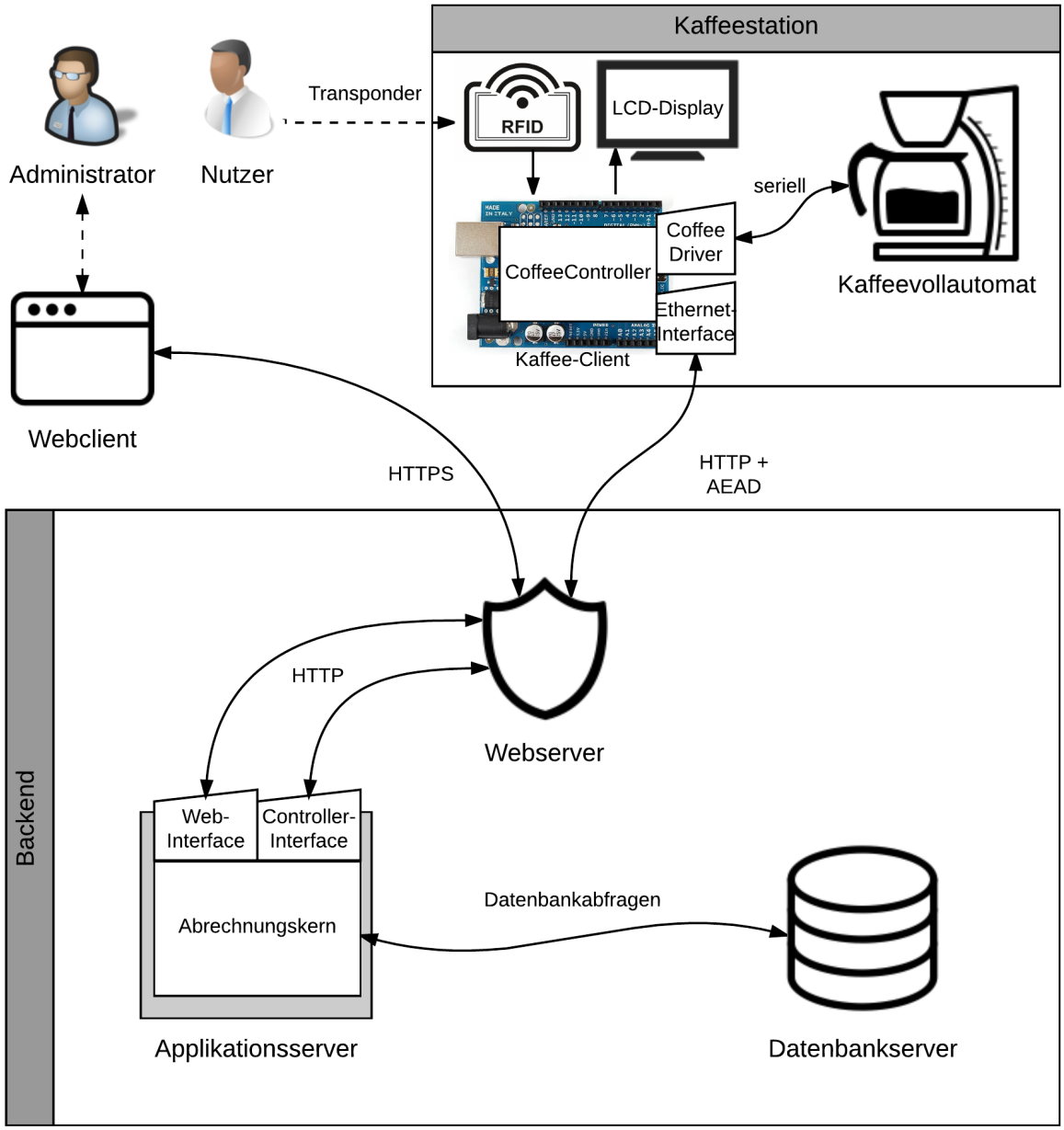

Abbildung 5.1: Die Architektur des gesamten Abrechnungssystems.

kann dann eine begrenzte Zeit lang kommuniziert werden. Voraussetzung ist der einmalige manuelle Austausch der öffentlichen Schlüssel von Kaffee-Client und Backend, was im Rahmen des Abrechnungssystems keinen zu großen Aufwand bedeutet. Die Machbarkeit dieser Methode des Schlüsselaustauschs mit Mikrocontrollern wurde schon teilweise demonstriert[1]. Fraglich ist jedoch, ob die volle Implementierung eines solchen Protokolls ausreichend Programmspeicher und dynamischen Speicher für die anderen Aufgaben des Mikrocontrollers lassen (u.a. Koordinierung von Kaffeemaschine, Bildschirm und RFID-Lesegerät). Auch die Rechenzeit einer einzelnen Verschlüsselung auf dem Controller im Bereich von zehn Sekunden ist bedenklich.

Hinsichtlich der Speicher- und Leistungsanforderungen sind symmetrische Verfahren, die sich auf einen langfristigen geheimen Schlüssel (*Master-Schlüssel*) verlassen, sehr viel anspruchsloser. Es genügt die Implementierung eines einzigen Verschlüsselungsalgorithmus, der Sitzungsschlüssel für den Nutzdatenverkehr und den Master-Schlüssel für die Etablierung neuer Sitzungsschlüssel nutzt. Wendet man hier ein AEAD-Verfahren an, wie in Abschnitt 2.1.3 beschrieben, dann wird sichere Kommunikation auch mit den begrenzten Möglichkeiten eines Mikrocontrollers erreicht.

Die in der Praxis für AEAD-Verfahren verwendeten Verschlüsselungsalgorithmen sind robust gegenüber *Chosen-Plaintext-Attacken* (mehr dazu in Abschnitt 6.1.2). Durch Verwendung von Nachrichtenzählern und Sitzungsschlüsseln werden auch *Replay Attacken* unmöglich. Die Sicherheitsanforderungen SA1 und SA2 werden also vollständig erfüllt, solange der Master-Schlüssel nur den befugten Parteien bekannt ist.

RFID-Transponder - Kaffee-Client

Im allgemeinen Fall ist es möglich, die Kommunikation zwischen einem modernen[2] Transponder und einem Kaffee-Client völlig sicher einzurichten. Jeder Transponder besitzt einen geheimen Master-Schlüssel. Ist dieser auch dem Kaffee-Client bekannt, können sich Transponder und Lesegerät in einem gegenseitigen Challenge-Response-Verfahren authentifizieren (siehe Abbschnitt 2.2.2). Im konkreten Fall des LRZ ist das jedoch problematisch, da die geheimen Master-Schlüssel der Transponder nur dem Zugangskontrollsystem des Gebäudes bekannt sind. Dieses ist proprietär und nicht ohne Weiteres erweiterbar.

Somit kann zwar eine Identifizierung des Transponders gegenüber dem Lesegerät durch den generell auslesbaren *Universal Identifier* erfolgen, umgekehrt jedoch nicht, und es findet keinerlei Authentifizierung statt. Somit entsteht an dieser Stelle eine gravierende Sicherheitslücke, die in Zukunft durch Integration des Abrechnungssystems mit dem Zugangskontrollsystem geschlossen werden könnte, oder alternativ durch die Verwendung unabhängiger Transponder für den Kaffeebezug. Das liegt jedoch nicht im Umfang dieser Arbeit.

Die Sicherheitslücke könnte ausgenutzt werden, indem ein Angreifer in Besitz eines RFID-Lesegeräts unbemerkt den *Universal Identifier* eines registrierten Transponders ausliest. Diesen kann er dann auf einen beschreibbaren Transponder übermitteln und so eine Kopie des rechtmäßig registrierten Transponders herstellen. Alle Funktionen des ursprünglichen Transponders stehen dem Angreifer dann offen. Die Sicherheitsanforderung SA3 wird somit nicht

[1] Beispielimplementierung eines RSA-basierten Schlüsselaustauschs, allerdings ohne Authentifizierung des Servers gegenüber dem Mikrocontroller in [Ard14].

[2] Ältere Transponder, wie der MIFARE Classic, nutzen unsichere Verschlüsselungsmethoden, die praktisch keine Sicherheit gegen Angreifer bieten. Neuere Modelle nutzen Algorithmen wie 3DES oder AES-128, die als sicher gelten. [NXP17]

erfüllt.

5.1.3 Systemkomponenten

Jede Komponente des Abrechnungssystems hat genau definierte Aufgabenbereiche, die sich nicht überschneiden. So wird die Modularität und die Erweiterbarkeit des System sichergestellt. Im Folgenden werden die Aufgaben der wichtigsten Systemkomponenten noch einmal erläutert.

Abrechnungskern

Der Abrechnungskern ist das zentrale Modul des Applikationsservers im Abrechnungssystem. Er implementiert die Verwaltung der Stammdaten sowie die Kontoführung, also das Durchführen von Transaktionen und ihre Sicherung. Seine Funktionalität wird den Interface-Modulen über eine API bereitgestellt. Er besitzt den exklusiven Zugriff auf die Datenbank, um Konflikte mit Zugriffen anderer Systemkomponenten zu vermeiden.

Um Nachvollziehbarkeit und Reproduzierbarkeit der Kontostände zu garantieren, wird jede Transaktion (im Sinne von Überweisung) in einem Transaktionslog festgehalten. Auch bei Abbruch oder nachträglicher Löschung einer Transaktion bleibt sie im Log, damit die Nachvollziehbarkeit immer gegeben ist [FA7]. Zusätzlich werden sämtliche internen Abläufe protokolliert, um beim Auftreten von Inkonsistenzen deren Entstehung nachvollziehen zu können.

Web-Interface

Die Hauptaufgabe des Web-Interface ist das Bereitstellen einer Weboberfläche, über die das Abrechnungssystem verwaltet werden kann. Dazu gehört einerseits die Einsicht und Änderung der Stammdaten, also alle Daten, die sich auf Nutzer, Konten, Transponder und Produkte beziehen. Andererseits werden weitere Funktionen wie das Erstellen von Kontoauszügen oder die Archivierung alter Daten bereitgestellt (siehe FA8- FA11). Alle Daten werden dazu über die Abrechnungskern-API bezogen.

Über die Präsentation der Daten hinaus ist es jedoch auch für die Authentifizierung des berechtigten Nutzers (des Administrators) gegenüber dem System verantwortlich. Das geschieht über eine Benutzername-Passwort-Kombination. Die Verschlüsselung und Authentifizierung über TLS muss diese Komponente nicht implementieren, da das vom Webserver übernommen wird.

Controller-Interface

Das Controller-Interface ist Teil des Applikationsservers und vermittelt zwischen dem Abrechnungskern und den Kaffee-Clients. Es ist sowohl für Ver-/Entschlüsselung und Authentizitätsprüfung als auch für die (De-)Codierung aller ausgetauschten Nachrichten zuständig. Ersteres wird über die Anwendung eines *Authenticated Encryption with Associated Data*-Verfahrens (siehe Abschnitt 2.1.3) erreicht. Die geheimen Schlüssel der Kaffee-Clients werden vom Controller-Interface in einer eigenen (logisch getrennten) Datenbank verwaltet.

Kaffee-Client

Der Kaffee-Client ist der Mikrocontroller, der eine Kaffeestation kontrolliert. Er steuert die Kaffeemaschine, nimmt Input vom RFID-Lesegerät an, schickt Output an einen Bildschirm und kommuniziert mit dem Backend über das lokale LRZ-Netz. Die Software des Kaffee-Clients wird im Folgenden *CoffeeController* und dessen Schnittstelle zum Internet *Ethernet-Interface* genannt (siehe Abbildung 5.1).

Die Kommunikation mit der Kaffeemaschine erfolgt über eine serielle Schnittstelle und ist im jeweiligen Protokoll des Herstellers codiert. Um Kompatibilität zu Kaffeemaschinen verschiedener Hersteller zu bieten, werden die allgemeinen Befehle, die vom Controller verwendet werden, von einem eigenen Modul, dem *CoffeeDriver*, zu herstellerspezifischen Befehlen übersetzt. Auf Nachrichten vom Lesegerät reagiert der Controller nur, wenn diese erwartet werden (also wenn ein Transponder gescannt werden soll). Der Bildschirm muss lediglich wenige Zeilen Text anzeigen können, um dem Nutzer die Systemausgaben zu präsentieren. Dieser Text wird vom Controller übertragen, es erfolgt keine Kommunikation in umgekehrter Richtung. Er kommuniziert mit dem Backend über HTTP, wobei der Nachrichteninhalt durch ein *Authenticated Encryption with Associated Data*-Verfahren verschlüsselt und authentifiziert wird.

Der Mikrocontroller muss also über ausreichend digitale Pins verfügen, um mit Lesegerät, Bildschirm und Kaffeemaschine zu kommunizieren, internetfähig sein und darüber hinaus ausreichend Rechen- und Speicherkapazität für symmetrische Verschlüsselungsverfahren haben.

Webserver

Die Kommunikation zwischen Clients und Applikationsserver wird über einen Webserver abgewickelt. Dieser kann statische Ressourcen direkt an Clients verteilen und liefert dynamische Elemente mithilfe des Applikationsservers. Außerdem ist er maßgeblich für die Systemsicherheit verantwortlich, da er als Schnittstelle zum Abrechnungskern vom gesamten lokalen Netzwerk aus erreicht werden kann und somit ein Angriffsziel darstellt. Korrekt konfiguriert bieten moderne Webserver jedoch Sicherheit vor unbefugten Manipulationen. Der Webserver ist auch für den Einsatz des *Transport Layer Security*-Protokolls (TLS) zwischen Webclient und Backend zuständig und verwaltet die benötigten Zertifikate. Client-Anfragen, die eine dynamische Antwort erfordern, werden im HTTP-Protokoll an den Applikationsserver weitergeleitet. Dort wird eine HTTP-Antwort generiert, der Webserver ergänzt die TLS-Schicht und schickt sie zurück zum Webclient. Somit ist die Kommunikation über das unsichere Netzwerk geschützt.

5.2 Systemabläufe

Dieser Abschnitt befasst sich mit den technischen Abläufen des Abrechnungssystems. Es werden der Transaktionsprozess, der Produktbezug, die Statusabfrage der Kaffeestationen und die Registrierung neuer Transponder an einer Kaffeestation erläutert.

5.2.1 Transaktionsprozess

Jede Buchung im Abrechnungssystem wird in Form einer Transaktion festgehalten. Diese wird in der Datenbank gespeichert und umfasst alle relevanten Daten zur Buchung, u.a. das

betroffene Konto, der Betrag und der Zeitpunkt der Buchung.

Guthaben-Aufladungen durch den Administrator können risikofrei in einem Schritt durchgeführt werden, da hier keine Einschränkungen bestehen. Die meisten Transaktionen, namentlich die Produktkäufe, werden jedoch von den Kaffee-Clients ausgelöst. Sie kommunizieren einerseits mit dem Abrechnungskern über ein unsicheres Netz, wodurch Nachrichten verloren gehen können oder die ganze Verbindung abbrechen kann. Andererseits mit einer Kaffeemaschine, die in blockierte Zustände geraten kann, wenn zum Beispiel Wasser oder Bohnen fehlen, eine Reinigung vonnöten ist oder schlicht ein Hardware-Defekt auftritt. Störungen bei der Produktausgabe und somit beim Transaktionsprozess sind also durchaus möglich.

Um solche Störungsfälle zu erkennen und den laufenden Transaktionsprozess abbrechen zu können, wird er in zwei Schritte geteilt. Sobald ein Nutzer ein Produkt bestellt, wird die Transaktion durch eine entsprechende Server-Anfrage zunächst angemeldet. Ist das Guthaben ausreichend, wird dies vom Abrechnungskern erlaubt. Daraufhin gibt der Kaffee-Client das Signal zur Produktausgabe an die Kaffeemaschine weiter. Kommt es zu keiner Störung, wird die Ausgabe durch eine weitere Nachricht an den Abrechnungskern bestätigt. Erst jetzt wird die Transaktion vollendet und Guthaben vom Konto abgebucht. Dieser geteilte Prozess ermöglicht es auch, gleichzeitige Produktbezüge von einem Konto zu verhindern. Diese sind unerwünscht, da sie dazu führen könnten, dass ein Konto überzogen wird. Eine Transaktionsanmeldung wird also nicht bestätigt, falls es kurze Zeit zuvor (ca. die Dauer eines Produktbezugs) bereits eine solche für dasselbe Konto gab. Somit wird die Anforderung FA6 berücksichtigt und ungewollte Kontoüberziehungen werden verhindert.

5.2.2 Produktbezug

Die zentralen Anforderungen an das Abrechnungssystem aus Sicht der Nutzer sind diejenigen, die an die Kaffeestationen gerichtet sind (siehe FA1-FA4). Zunächst muss ein Nutzer sein Guthaben durch Scannen seines Transponders an einer Kaffeestation einsehen können [FA1]. Dazu schickt der Kaffee-Client eine entsprechende Server-Anfrage, die über den Webserver an das Controller-Interface des Applikationsservers weitergeleitet wird, dass wiederum die Applikationskern-API nutzt, um das Guthaben des richtigen Kontos von der Datenbank abzurufen. Die Daten werden vom Controller-Interface verschlüsselt und zurückgesendet. Daraufhin hat der Nutzer die Möglichkeit, ein beliebiges Produkt an der Kaffeemaschine zu bestellen [FA2]. Eine Bestellung wird an den Kaffee-Client weitergeleitet und es folgt eine weitere Server-Anfrage, diesmal der Beginn einer Transaktion, die auf demselben Weg wie die vorherige bearbeitet wird. An dieser Stelle kann der Abrechnungskern den Produktbezug abbrechen, falls das Guthaben des Nutzer nicht ausreicht [FA3]. Die Bestätigung oder das Abbruchsignal werden zurück zum Kaffee-Client geschickt. Dieser gibt dem Nutzer nun über den Bildschirm Rückmeldung, ob das Produkt ausgegeben werden kann [FA4]. Ist das der Fall, löst ein entsprechendes Signal die Produktausgabe in der Kaffeemaschine aus. Zuletzt schickt der Kaffee-Client eine weitere Nachricht zum Server, der diesen über den Erfolg der Produktausgabe informiert.

5.2.3 Statusabfrage

Aus Anforderung FA13 geht hervor, dass der aktuelle Status eines Kaffee-Clients regelmäßig an das Backend übertragen werden muss, um im Administrationstool abrufbar zu sein. Dazu

sendet der Kaffee-Client in regelmäßigen Abständen Nachrichten mit einem kurzen Statuscode an das Backend. Eine Anfrage des Servers an die Kaffee-Clients kommt nicht in Frage, da so das ansonsten strikt eingehaltene Kommunikationsmuster gebrochen würde, bei dem jede Interaktion von den Clients ausgeht. Für Anfragen des Servers an die Kaffee-Clients wären statische IP-Adressen der Clients nötig, was so vermieden werden kann.

5.2.4 Registrierung eines RFID-Transponders

Wie bereits in Use Case 3 (siehe Abschnitt 3.2) beschrieben, erfordert das Anlegen eines RFID-Transponders in der Datenbank hauptsächlich sein *Unique Identifier* (UID). Dieser könnte durch einen Scan am Lesegerät in Erfahrung gebracht werden und dann im Administrationstool angegeben werden. Diese Prozedur erfordert jedoch die manuelle Eingabe des UID, was kein wünschenswerter Schritt ist. Um das zu umgehen, soll der Kaffee-Client durch das Scannen eines bestimmten Master-Transponders in einen Registrierungszustand versetzt werden können. In diesem schickt er die nächste gescannte UID an das Backend, um ein Transponder-Modell in der Datenbank anzulegen, anstatt einen Check-In auszulösen [FA5]. Die Freischaltung des Transponders und die Verknüpfung mit einem Konto kann dann im Administrationstool vorgenommen werden.

5.3 Technologien

Die für das Abrechnungssystem verwendeten Technologien und Software-Frameworks werden im Folgenden präsentiert. Bei der Auswahl wurde besonders auf Effizienz, gute Dokumentation sowie langfristigen Support geachtet.

5.3.1 Arduino

Die Arduino-Plattform umfasst sowohl Open-Source-Hardware als auch Software, was eine kostengünstige Entwicklung kleiner Elektronikprojekte ermöglicht. Während die Hardware in den USA unter dem eigentlichen Namen Arduino vertrieben wird, tragen alle international verkauften Produkte wegen Streitigkeiten um die Markenrechte den Namen Genuino. [Kön15]

Die sogenannten *Boards* sind im wesentlichen Mikrocontroller (meist Atmel AVR) mit vorinstalliertem Bootloader und integrierter USB-Schnittstelle. Dies ermöglicht das Aufspielen eigener Programme ohne zusätzliches Programmiergerät, was die Zugänglichkeit der Plattform erhöht und den Lernprozess beschleunigt. Deswegen gilt die Arduino-Plattform als einsteigerfreundlich und auch für Nutzer aus nicht-technischen Bereichen geeignet. [Ard17b] Abgesehen von den Mikrocontrollern werden auch viele Komponenten, die deren Funktionalität erweitern, angeboten (sogenannte *Shields*). Einbindung in Netzwerke über Ethernet oder WLAN, Anbindung an das Mobilfunknetz oder der Betrieb externer Motoren kann mit den Shields ermöglicht werden. Zudem verfügt jedes Board über eine Vielzahl an Pins, an die andere Hardware, wie LCD-Bildschirme, RFID-Lesegeräte, Relays oder Sensoren, angeschlossen werden können. Neben den über die Plattform selbst vertriebenen Geräten ist durch die umfangreiche Quelloffenheit auch anderen Herstellern möglich, kompatible Mikroprozessoren und Erweiterungen zu verkaufen. Sie sind oft günstiger, tragen aber nicht den Namen Arduino und unterliegen somit nicht denselben Qualitätsansprüchen. [Ard17a]

Die Arduino-Boards werden in C/C++ programmiert, wobei die Nutzung der Entwicklungsumgebung Arduino IDE einige Vereinfachungen wie die Integration von Hardware-

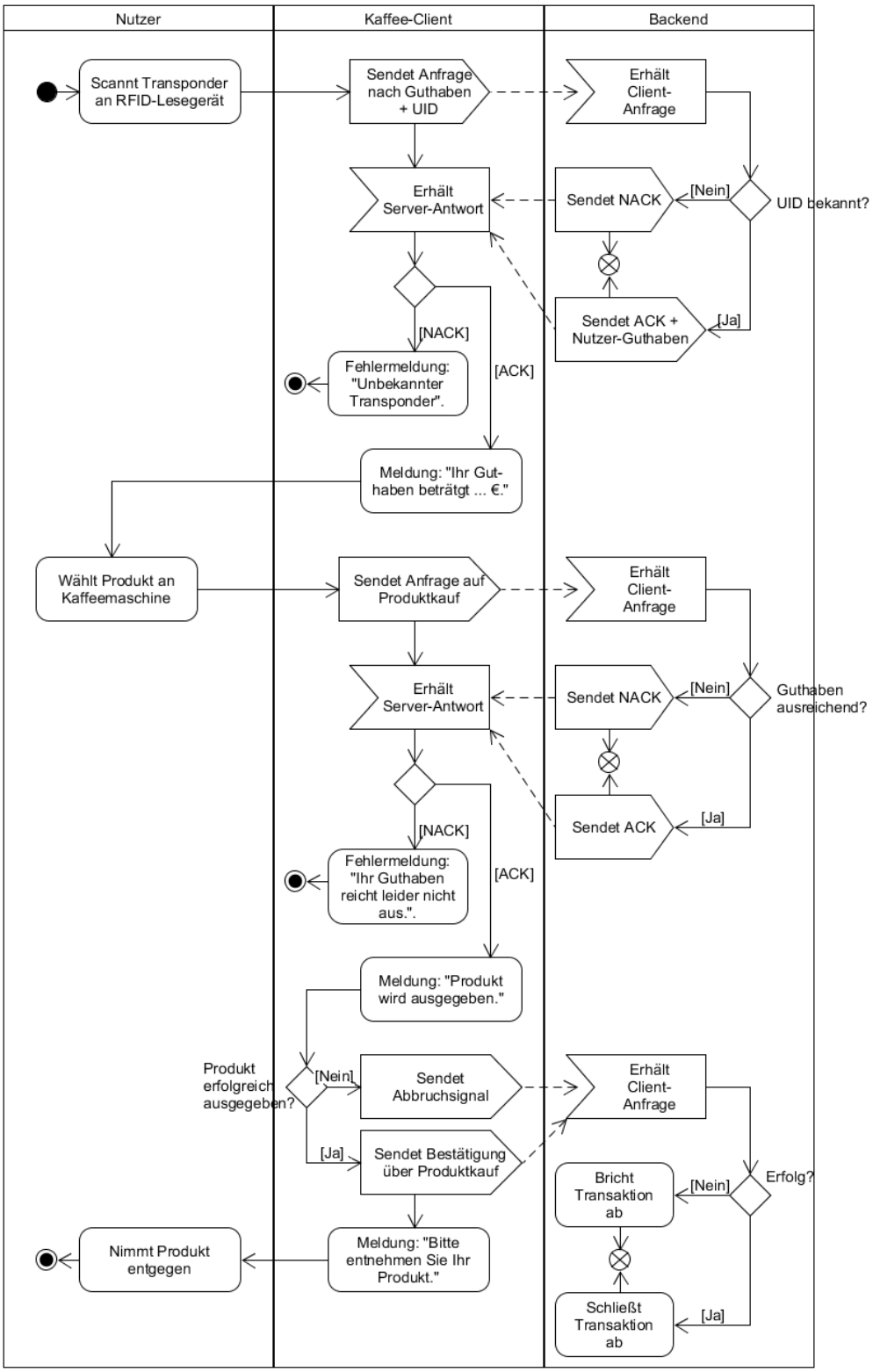

Abbildung 5.2: Ablauf eines Produktbezugs an einer Kaffeestation aus technischer Sicht.

Bibliotheken, die automatische Generierung der Header-Dateien und die Definition praktischer Datentypen bietet. Zudem wird die Programmstruktur durch die Methoden *init*, einmalig aufgerufen beim Start, und *loop*, der eigentlichen Programmschleife, vorgegeben. Die Programme können alternativ auch in Standard-C/C++ entwickelt und der Build-Prozess mit Open-Source-Tools durchgeführt werden. Die Plattform ist über Jahre immer weiter gewachsen und hat eine große Entwicklergemeinschaft aufgebaut, wodurch auch das Angebot an Open-Source-Software-Bibliotheken enorm groß ist. Somit stehen die Chancen für langfristigen Support der Plattform gut.

5.3.2 Django

Django ist ein Python-Framework, dass für die schnelle und dynamische Entwicklung großer sowie kleiner Internetanwendungen gedacht ist. Es baut auf dem Model-View-Template-Architekturmuster auf (eine art MVC-Pattern). Durch objektrelationale Abbildung (ORM) werden Objekte der Webanwendung automatisiert auf relationale Tabellen der Datenbank abgebildet, was die Verwaltung persistenter Daten enorm vereinfacht. Außerdem stellt es eine Template-Sprache bereit, mit der HTML-Seiten dynamisch generiert werden und auch Vererbung, wie bei der objektorientierten Programmierung, ermöglicht. Hinzu kommen einige Funktionen, die die allermeisten Webanwendungen benötigen, wie das Sammeln statischer Dateien an einem zentralen Ort für den Webserver und ein eigenes Authentifizierungssystem, das leicht konfiguriert werden kann. [Dja17]

Innerhalb eines Django-Projekts können mehrere sogenannte Django-Apps angelegt werden, die durch getrennte Python-Packages realisiert werden. Jeder App wird in der Regel ein eigener URL-Pfad zugewiesen und ihre Datenmodelle sind konzeptionell getrennt, auch wenn sie auf demselben Datenbankserver liegen.

5.3.3 Apache HTTP Server

Der Apache HTTP Server ist ein Multi-Plattform-Webserver (Unix, Windows), der entwickelt wurde, um sichere, effiziente und erweiterbare HTTP-Services über das Internet anbieten zu können. Entwickelt wird er von der Apache Software Foundation und ist unter der Apache License, Version 2.0 frei verfügbar. Schon seit 1996 ist er der populärste Webserver.

Durch zusätzliche Module verfügt der Apache HTTP Server über Funktionen wie eine URL-Rewrite-Engine, TLS-Unterstützung und Schnittstellen für verschiedene Programmiersprachen, u.a Perl, Python und PHP. [Apa17]

6 Realisierung

In diesem Kapitel wird die Implementierung des zuvor entworfenen Abrechnungssystems geschildert. Zunächst wird auf die Umsetzung der Kommunikation zwischen Kaffee-Clients und Backend eingegangen, wobei Strukturierung, Verschlüsselung und Codierung der Nachrichten erklärt wird. Darauf folgt die Implementierung des Backends inklusive Webserver, Applikationsserver und Datenbank. Die Organisation der Daten, zusätzliche Sicherheitsmechanismen und die Besonderheiten der einzelnen Plattformen und Technologien werden dargelegt. Das im Rahmen dieser Arbeit realisierte System ist jedoch nur ein Prototyp, der zwar einen Großteil der Anforderungen erfüllt, an einigen Stellen aber noch nicht vollständig ist.

6.1 Kommunikation zwischen Kaffee-Clients und Backend

Das Kommunikationsprotokoll zwischen Kaffee-Clients und Backend folgt dem klassischen Client-Server-Modell. Jegliche Kommunikation geht also von den Kaffee-Clients aus, auf jede Anfrage folgt genau eine Antwort vom Backend. Ein Szenario, in dem das Backend Informationen von den Kaffee-Clients anfordert, ist nicht vorgesehen. Somit sind keine statischen IP-Adressen für die Kaffee-Clients notwendig.

Es wird das HTTP-Protokoll genutzt, um die Implementierung auf Serverseite zu vereinfachen, denn so nimmt das Django-Framework dem Entwickler viel Arbeit ab. Damit keine Informationen über die Nachrichten unverschlüsselt übertragen werden, befinden sich keine relevanten Informationen in den unsicheren HTTP-Headern. Da auch die URL der Anfrage eine wichtige Informationen darstellt, sie könnte zum Beispiel über die Art der Serveranfrage Aufschluss geben, erfolgen alle Anfragen auf eine einheitliche URL. Die folgenden Abschnitte beschreiben die Struktur, die Verschlüsselung und Authentifizierung, sowie die Codierung der Nachrichten zwischen Absender und Empfänger.

6.1.1 Struktur

Typische Protokolle für die Client-Server-Kommunikation sind textbasiert, wie z.B. *XML* (Extensible Markup Language) oder *JSON* (JavaScript Object Notation). Sie sind eine gute Wahl für die Strukturierung von Nachrichten zwischen leistungsstärkeren Maschinen, für die Implementierung auf einem Mikrocontroller sind sie jedoch zu ineffizient. Die Nachrichten zwischen Kaffee-Client und Backend sind Byte-orientiert, d.h. es wird kein für Menschen lesbarer Text übertragen. Das erste Byte einer Nachrichtist ist der sogenannte *Message Code*. Aus ihm geht der Zweck der Nachricht hervor und mit dieser Information kann der Empfänger sie korrekt interpretieren. Empfängt der Server also beispielsweise eine Nachricht mit dem Message Code für eine Check-In-Anfrage, weiß er auch, dass die folgenden vier Bytes die UID des Transponders enthalten. Für jeden Message Code ist die Struktur der entsprechenden Nachricht genau definiert. In den Tabellen 6.1 und 6.2 sind alle Nachrichtentypen samt Message Codes und ihrer Bedeutung aufgeführt. Mit Ausnahme der Server-Antwort beim

Schlüsselaustausch ist keine Nachricht größer als 16 Bytes. Bei der Nutzung von XML oder JSON müsste eine Vielfaches dieser Datenmenge übertragen werden.

Message Code	**Nutzdaten und Länge in Byte**
Schlüsselaustausch	`Schlüssel-Nonce 8`
Check-In	`UID 4`
Karte registrieren	`UID 4`
Produktkauf	`UID 4 || Produkt-Code 1`
Kaufbestätigung	`UID 4 || Produkt-Code 1 || Transaktions-ID 4`
Kaufabbruch	`UID 4 || Produkt-Code 1 || Transaktions-ID 4`
Alive Signal	-

Tabelle 6.1: Client-Anfragen und ihre Struktur.

Message Code	**Nutzdaten und Länge in Byte**
Schlüssela. ACK	`Schlüssel-Nonce 8 || Temporärer Schlüssel 24`
Check-In Master	`UID 4`
Check-In ACK	`UID 4 || Kontostand 2`
Check-In NACK	`UID 4`
Produktkauf ACK	`UID 4 || Produkt-Code 1 || Transaktions-ID 4`
Produktkauf NACK	`UID 4 || Produkt-Code 1`

Tabelle 6.2: Server-Antworten und ihre Struktur.

6.1.2 Verschlüsselung und Authentifizierung

Die Sicherung der Kommunikation mit den Kaffee-Clients erfolgt über *Authenticated Encryption with Associated Data* (siehe Abschnitt 2.1.3), was einen gemeinsamen geheimen Schlüssel der Kommunikationspartner erfordert (Client-Master-Schlüssel). Dieser ist langfristig gültig und muss sicher auf den Geräten verwahrt werden. Zusätzlich wird ein temporär Schlüssel ausgetauscht, der für die AEAD-Anwendung auf alle regulären Nachrichten genutzt wird. Der Client-Master-Schlüssel wird ausschließlich für den Austausch der temporären Schlüssel verwendet. Die Vorteile dieser Methode werden im folgenden Abschnitt beschrieben.

Schlüsselaustausch und Nachrichtenzähler

Es gibt mehrere Gründe, aus denen die Nutzung des langfristigen Master-Schlüssels für die Sicherung des Nachrichtenaustauschs allein nicht ratsam ist. Erstens, führt der Verlust eines

Schlüssels zur Offenlegung aller damit gesicherten Nachrichten und ermöglicht Angreifern, sich mithilfe desselben als authentischer Kommunikationspartner auszugeben. Handelt es sich um den Master-Schlüssel, so muss dieser aufwendig ersetzt werden. Wurde lediglich ein temporärer Schlüssel kompromittiert, ist der Austausch eines neuen für die Kommunikationspartner in der Regel unkompliziert.

Des weiteren existieren bestimmte Angriffsmodelle, die durch die Nutzung eines Sitzungsschlüssels einfach verhindert werden können. Eine Replay-Attacke bezeichnet den Versuch eines Angreifers, eine zuvor aufgezeichnete Nachricht zu einem von ihm bestimmten Zeitpunkt erneut zu verschicken. Angenommen das Abrechnungssystem nutzt nur den Master-Schlüssel und ein Angreifer *Mallory* belauscht den Kommunikationsverkehr zwischen einem Kaffee-Client *Alice* und dem Backend *Bob*. Ein Nutzer *Nick* bestellt einen Kaffee bei *Alice*, was zu einigen Nachrichten zwischen *Alice* und *Bob* führt. Verschickt *Mallory* die Nachrichten von *Alice* in derselben Reihenfolge zu einem späteren Zeitpunkt erneut an *Bob*, werden diese als weitere Nachrichten von *Alice* akzeptiert. *Mallory* kann also die zuletzt von *Nick* getätigten Transaktionen beliebig oft wiederholen, ohne dass *Bob* es mitbekommt und so *Nicks* Konto unrechtmäßig belasten.

Die simpelste Maßnahme, diese Attacke zu verhindern, ist die Einführung eines globalen Nachrichtenzählers. Er ist Teil des Headers und wird nach jeder Nachricht inkrementiert. Da der Nachrichten-Header durch das AEAD-Verfahren gesichert wird, ist eine Manipulation nicht möglich. Eine solche Lösung auf Mikrocontrollern zu implementieren, ist jedoch problematisch. Der globale Zähler muss persistent gespeichert werden, damit beide Kommunikationspartner auch nach Neustart des Systems noch auf demselben Stand sind. Die meisten Mikrocontroller verfügen dafür über EEPROM (*Electrically Erasable Programmable Read-Only Memory*), der jedoch nur eine begrenzte Anzahl von Schreibzyklen überlebt, oft nur ca. 100.000. Da nach jeder Nachricht auf den EEPROM geschrieben werden muss, würden die meisten EEPROMs bei dieser Vorgehensweise nicht einmal ein Jahr überleben:

$$5\ \frac{\text{Nachrichten}}{\text{Kaffeebezug}} \cdot 100\ \frac{\text{Kaffeebezüge}}{\text{Tag}} \cdot 365\ \frac{\text{Tage}}{\text{Jahr}} = 182.500\ \frac{\text{Nachrichten}}{\text{Jahr}}$$

Es gibt Methoden, die Lebensdauer des EEPROM durch Verteilung der Last zu verlängern, doch eine simplere Lösung für das Problem ist die Nutzung eines temporären Nachrichtenzählers in Verbindung mit einem temporären Schlüssel.

Ein temporärer Nachrichtenzähler zählt nur die Nachrichten, die mit dem aktuellen Schlüssel verschickt wurden. Wird das System neu gestartet und der Zähler geht somit verloren, kann einfach ein neuer Schlüssel mithilfe des Client-Master-Schlüssels etabliert werden und der Zähler kann von Null anfangen. Nun kann dieselbe Attacke natürlich auf den Austausch des temporären Schlüssels angewandt werden. Zeichnet *Mallory* einen solchen Austausch auf und wiederholt ihn, kann *Bob* dies nicht erkennen und teilt *Alice* einen neuen temporären Schlüssel zu. Die weitere Kommunikation wird dadurch unterbrochen, da *Alice* und *Bob* unterschiedliche Schlüssel verwenden und ihre Nachrichten gegenseitig nicht mehr authentifizieren können. Somit wird das Schutzziel der Verfügbarkeit (siehe Abschnitt 2.1.1) nicht mehr erreicht. Da bei dieser Betrachtung aber ohnehin davon ausgegangen wird, dass *Mallory* die Kommunikation der beiden Parteien unterbrechen und modifizieren kann (siehe Abschnitt 3.4.1), ist das Schutzziel der Verfügbarkeit unmöglich zu garantieren. In der Praxis sollte *Alice* nach einigen vergeblichen Versuchen *Bob* zu erreichen, einfach einen neuen Schlüsselaustausch anstoßen.

Ablauf eines Schlüsselaustauschs

Der Austausch eines temporären Schlüssels zwischen CoffeeController und Controller-Interface geht immer vom Client aus, also dem CoffeeController. Dieser generiert zunächst zwei Zufallszahlen: Eine Nonce für die Sicherung der Nachricht mittels des AEAD-Verfahrens, und eine weitere sogenannte *Key Nonce*. Diese wird später als Beweis dienen, dass die Server-Antwort einen neuen temporären Schlüssel enthält. Die Server-Anfrage enthält nur den entsprechenden Message Code für den Schlüsselaustausch und die Key Nonce. Sie wird per AEAD-Verfahren und Client-Master-Schlüssel gesichert und verschlüsselt.

Das Server-seitige Controller-Interface überprüft und entschlüsselt die Anfrage entsprechend und generiert selbst zwei weitere Zufallszahlen. Eine ist wiederum für die Sicherung der Server-Antwort vorgesehen, die andere ist der neue temporäre Schlüssel. Die Server-Antwort enthält neben diesem und dem entsprechenden Message Code auch die Key Nonce aus der Anfrage des Clients. Das Controller-Interface ersetzt an diese Stelle auch den temporären Schlüssel in der Datenbank, Kommunikation mittels des alten Schlüssels ist nun also nicht mehr möglich. Die Server-Antwort wird durch den Client-Master-Schlüssel gesichert übertragen und vom CoffeeController überprüft und entschlüsselt. Jetzt muss dieser nur noch die Key Nonce aus der Server-Antwort mit der zuvor selbst generierten Key Nonce vergleichen. Stimmen sie nicht überein, ist die erhaltene Nachricht wahrscheinlich Teil einer Replay-Attacke und muss verworfen werden. Andernfalls wird der erhaltene temporäre Schlüssel akzeptiert und die weitere Kommunikation mit diesem gesichert. Der gesamte Ablauf eines solchen Austauschs ist in Abb. 6.1 dargestellt.

AEAD-Implementierung

Für die Verschlüsselung und Authentifizierung der Nachrichten kommt das *Authenticated Encryption with Associated Data*-Verfahren *ChaCha20-Poly1305* zum Einsatz. Dieses nutzt die Stromchiffre ChaCha (mit 20 Runden) und den MAC-Algorithmus Poly1305, um Vertraulichkeit, Integrität und Authentizität zu garantieren (siehe Abschnitt 2.1.3). Die Algorithmen *ChaCha* und *Poly1305* sind auf Effizienz und Robustheit gegen Side-Channel-Attacks optimiert, was sie zu guten Kandidaten für die Anwendung auf einem Mikrocontroller macht [NL15]. Darüber hinaus findet das Verfahren seit 2014 Anwendung im TLS-Stack bei der Kommunikation zwischen Googles Browser Chrome und von Google gehosteten Internetseiten. Die Ver- und Entschlüsselung auf Geräten ohne spezielle AES-Hardware[1] erfolgt mit *ChaCha20-Poly1305* ca. dreimal schneller verglichen mit dem etablierten *AES-GCM*-Verfahren [Bur14].

Auf dem Kaffee-Client wird die Implementierung aus der CryptographicLibrary von Rhys Weatherley genutzt. Auf dem Arduino Uno beträgt der Zeitaufwand für die Ent-/Verschlüsselung pro Byte nur etwa 40 μs und für das Key Setup weitere 900 μs, also weniger als zwei Millisekunden für eine 16-Byte lange Nachricht [Wea]. Auf Serverseite wird *ChaCha20-Poly1305* durch die Bibliothek *libsodium* ermöglicht, eine C-Bibliothek mit verschiedenen Verschlüsselungsverfahren. Der Python-Wrapper *pysodium* ermöglicht die Nutzung auf dem Django-Applikationsserver.

[1] Viele hochwertige Prozessoren implementieren einen erweiterten Befehlssatz, der auf die Ausführung des AES-Algorithmus spezialisiert ist. Es können beispielsweise ganze AES-Runden mit einem einzigen Prozessorbefehl ausgeführt werden. Mirkocontroller besitzen diesen Befehlssatz in der Regel aber nicht. [Gue10]

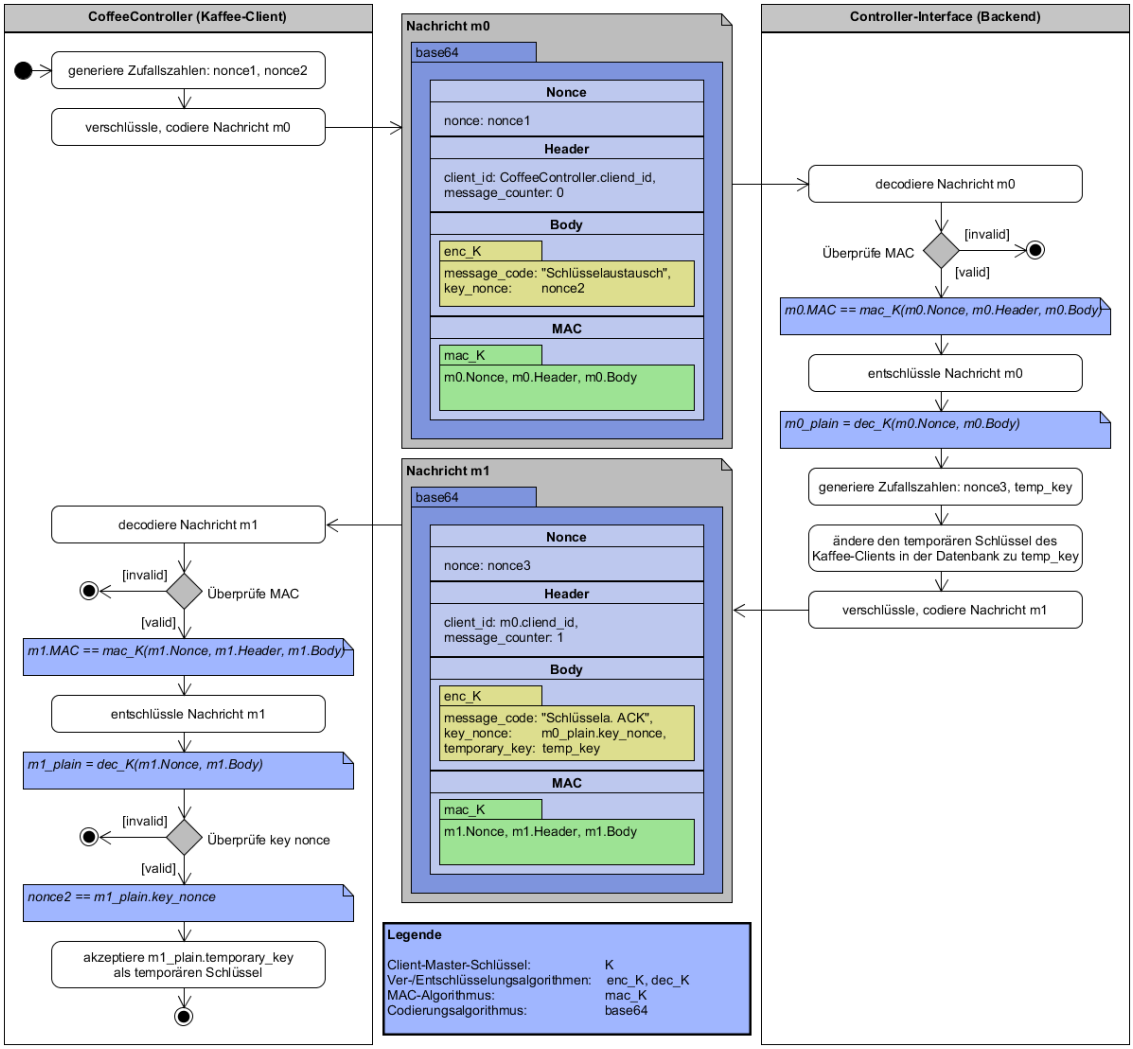

Abbildung 6.1: Aktivitätsdiagramm des Austauschs eines temporären Schlüssels zwischen CoffeeController und Controller-Interface inklusive Pseudocode und Nachrichtenstruktur.

6.1.3 Codierung

Die Kommunikation des Kaffee-Client mit dem Backend erfolgt über das rein textbasierte HTTP-Protokoll, welches nur druckbare Zeichen zur Übertragung vorsieht. Der Kaffee-Client überträgt jedoch Daten, in denen jedes Byte sämtliche Werte von 0 bis 255 annehmen kann. Es ist also eine Codierung dieser Rohdaten zu HTTP-konformen Zeichen nötig.

Base64 ist eine Codierungsmethode, die beliebige Byte-Strings anhand der Zeichen "A-Z", "a-z", "0-9", "+" und "/" darstellt. Dazu ist lediglich eine Expansion des String um 33% nötig, 3 Byte Rohdaten können als vier Base64-codierte Zeichen dargestellt werden. Falls die Länge des zu codierenden String kein Vielfaches von drei ist, wird am Ende des Codes Padding in Form von "="-Zeichen angefügt. [Jos06]

Die Codierung der Nachrichten zwischen Kaffee-Client und Backend in Base64 ermöglicht die Übertragung über HTTP mit vertretbarem Overhead. Die Buffer des Kaffee-Clients, in denen die Nachrichten zwischengespeichert werden, müssen um lediglich ein Drittel ihrer Größe erweitert werden, verglichen mit der Übertragung von uncodierten Bytes.

6.2 Backend

Das Backend des Abrechnungssystems, bestehend aus dem Applikationsserver, dem Datenbankserver und dem Webserver, wurde - mit kleinen Ausnahmen - wie im Entwurf geschildert umgesetzt. Alle Komponenten laufen auf einem virtuellen Debian Server, der vom lokalen Netz des LRZ aus auf einer ihm zugeteilten DNS-Adresse erreichbar ist. Anstatt eines Datenbankservers kommt beim Prototyp lediglich eine SQLite-Datenbank zum Einsatz, die jedoch bei der voraussichtlichen Belastung von wenigen Operationen in der Minute vollkommen ausreicht.

6.2.1 Apache Webserver

Der Apache Webserver bildet die Schnittstelle vom lokalen Netz des LRZ zum Applikationsserver. Die Einbindung des Applikationsservers erfolgt mithilfe des *Python Web Server Gateway Interface* (WSGI), das die Kommunikation zwischen Python-basierten Web-Frameworks und Webservern erlaubt. Das Apache-Modul *mod_wsgi* implementiert diesen Standard und erlaubt das Weiterleiten der HTTP-Anfragen an das entsprechende Interface des Applikationsservers.

Die Konfiguration des Apache Webserver beinhaltet zwei Virtual Hosts auf Port 80 (HTTP) und Port 443 (HTTPS), die entsprechende Web-Anfragen auf diesen Ports annehmen. Bei sämtlicher Kommunikation mit dem Web-Interface (also das Administrationstool) wird die Nutzung des HTTPS-Protokolls erzwungen. Geht eine HTTP-Anfrage ein, antwortet der Server mit einer Weiterleitung auf die entsprechende HTTPS-URL.

6.2.2 Django Applikationsserver

Der Applikationsserver wurde komplett im Django-Framework implementiert, was besonders die Umsetzung des Web-Interface enorm beschleunigt hat. Wie vom Django-Design-Pattern vorgesehen, sind die drei Module (Applikationskern, Web- und Controller-Interface) als eigenständige Django-Apps realisiert worden. Dadurch ist die nötige Trennung der Aufgabengebiete Abrechnung und Verwaltung, Präsentation der Daten und Verwaltungsfunktionen, sowie Mikrocontroller-Kommunikation, gegeben.

Eingehende HTTP-Anfragen auf eine bestimmte URL werden vom Django-*URL Dispatcher* an entsprechende Handling-Methoden der Interface-Apps übergeben. Die URLs des Web-Interface bilden eine RESTful API, d.h. sie bieten zustandslose Funktionen über das Netzwerk an. Somit hat jede Information bzw. Funktion, die vom Web-Interface bereitgestellt wird, immer dieselbe Adresse und die HTTP-Antworten des Backend beinhalten alle zum Verständnis benötigten Informationen. Das hat den Vorteil, dass bei Empfang einer Nachricht (auf Client- oder Serverseite) auch ohne Kontext zweifelsfrei die Bedeutung dieser festgestellt werden kann. Weder der Webclient noch der Applikationsserver muss also einen internen Zustand implementieren.

6.2.3 Abrechnungskern

Die *ORM*-Engine des Django-Framework ermöglicht es, einen Großteil der Geschäftslogik direkt in den Modellklassen zu implementieren. Dadurch kann die zentrale Klasse des Abrechnungskerns Aufgaben, die Operationen auf der Datenbank ausführen, fast vollständig delegieren. Eine typische Methode hat folgenden Aufbau:

- Durchsuche die Datenbank nach erforderlichen Objekten.
- Rufe Objekt-Methoden auf.
- Prüfe Rückgabewerte.
- Leite Ergebnis weiter an Interface.

Jeder dieser Schritte hat jedoch auch das Potenzial, unterschiedliche Exceptions auszulösen. Die Argumente der Methode spezifizieren in der Regel u.a. die IDs der zu manipulierenden Objekte. Es kann aber immer zu fehlerhaften Eingaben kommen, z.B. wenn eine Fehlfunktionen in einem Kaffee-Client auftritt und dadurch Anfragen zu nicht existenten Objekten zustande kommen (*InvalidInputException*). Aufgerufene Objektmethoden haben die Möglichkeit, auf Inkonsistenzen in der Datenbank mit Exceptions zu reagieren (*DatabaseInconsistencyException*). Genauso muss der Vorgang abgebrochen werden, wenn die Rückgabewerte der Objektmethoden offensichtlich unsinnig sind oder sie zu unerlaubten Zuständen führen würden (*InsufficientBalanceException, AccountBlockedException*). Alle diese möglichen Exceptions müssen von den Interfaces abgefangen werden, um trotzdem eine gültige HTTP-Antwort generieren zu können, auch wenn diese nur den internen Fehler beschreibt.

Datenbankschema

Die Struktur der Modellklassen definieren durch das *Object Relational Mapping* implizit auch das Datenbankschema. Abb. 6.2 zeigt ein UML-Diagramm der Modellklassen des Abrechnungskerns und des Controller-Interface. Die Klassen `User, Account, Product` und `Transaction` repräsentieren ihre namensgebenden Objekte und Konzepte aus der realen Welt und sind weitestgehend selbsterklärend. Die Transponder werden durch die Klasse `Card` und die Kaffee-Clients durch `Client` repräsentiert. Die Klassen des Abrechnungskerns haben die gemeinsame Superklasse `BillingModel`. Dazu kommen Modellklassen, die zeitlich begrenzte Assoziationen zwischen den Instanzen dieser Klassen darstellen. Eine Instanz der Klasse `ProductPricing` kann als Eigenschaft der Klasse `Product` verstanden werden, die jedoch nur in einem zeitlich definierten Rahmen gültig ist. Implementiert wird das durch

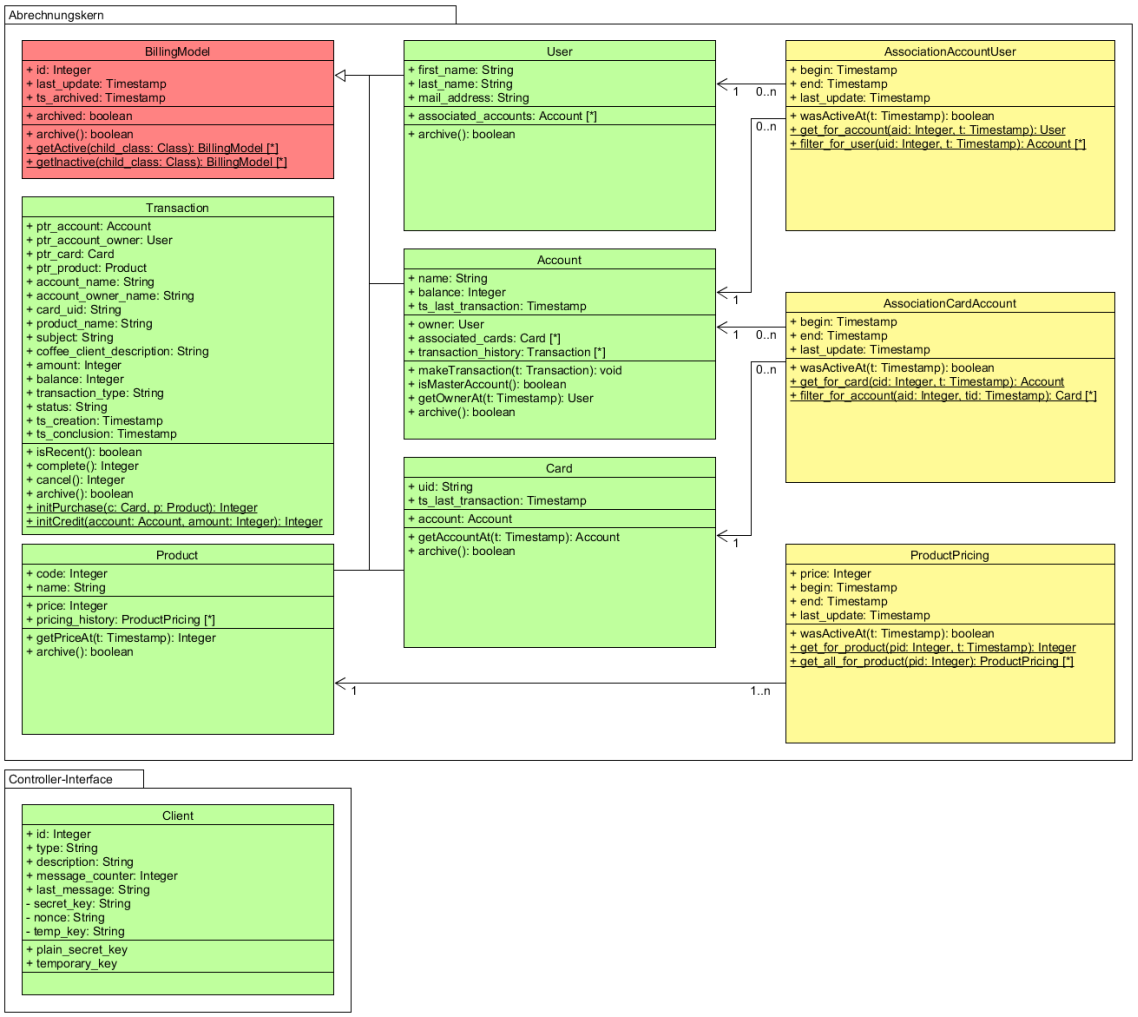

Abbildung 6.2: UML-Diagramm der Modellklassen des Abrechnungssystems. Jede Klasse hat Felder für Variablen, (Python-) Properties und Methoden.

1:n-Beziehungen von `Product` zu `ProductPricing`, wobei immer nur ein `ProductPricing` zu einem gegebenen Zeitpunkt gültig ist. Analog repräsentieren Instanzen der Klassen `AssociationAccountUser` und `AssociationCardAccount` zeitliche Assoziationen zwischen Nutzern, Konten und Transpondern, wobei ein Nutzer gleichzeitig mehrere Konten führen kann und ein Konto mit mehreren Transpondern gleichzeitig verknüpft sein kann.

Diese Struktur des Datenbankschemas ermöglicht es, alle temporären Beziehungen der Objekte untereinander permanent zu Speichern. Folglich dürfen keine einzelnen Objekte gelöscht werden, da die Referenzen von Instanzen der Assoziationsklassen bestehen bleiben. Anstatt nicht mehr benötigte Instanzen zu löschen, werden sie archiviert, wodurch sie aus dem aktiven Betrieb herausgefiltert werden. Zu einem beliebigen Zeitpunkt kann ein Administrator dann alle archivierten Objekte aus der Datenbank entfernen. Die Transaktionen werden jedoch gesondert behandelt und gelöscht. Da die Objekte, auf die eine Transaktion verweist, u.U. nicht mehr existieren, speichert diese nicht nur Referenzen sondern auch die

tatsächlichen Werte, beispielsweise für Kontoname, Name des Kontobesitzers und Name des gekauften Produkts.

6.2.4 Web-Interface

Das Web-Interface wurde nach dem Django-Design-Pattern realisiert. Eingehende HTTP-Anfragen werden vom URL-Dispatcher an die richtigen Handling-Methoden übergeben, welche die Applikationskern-API nutzen, um beispielsweise Kontostände abzufragen oder Transaktionen zu beginnen. Mithilfe der HTML-Templates können dann Webseiten generiert werden, die die Ergebnisse präsentieren. Das Web-Interface benötigt keinen Zugriff auf die Datenbank, da die Sitzungsverwaltung vom Django-Framework und die TLS-Verwaltung vom Webserver übernommen wird.

Das Web-Interface stellt den Administratoren folgende Funktionen in Form des Administrationstools bereit:

- **Übersicht über alle Objekte in der Datenbank inklusive Suchfunktion**
  Für jede Art von Objekten existiert eine Seite im Administrationstool, die eine interaktive Liste dieser Objekte präsentiert und auch die Möglichkeit archivierte Objekte einzusehen bietet.

- **Anlegen neuer Objekte**
  Es können jederzeit neue Nutzer, Konten, Transponder, Produkte und Kaffeestationen erstellt werden.

- **Detailansicht für jedes Objekt**
  Die Detailansicht eines Kontos beinhaltet die vergangenen Umsätze auf diesem Konto, die eines Nutzers seine verknüpften Konten, die eines Produkts sein Preisverlauf. Die Detailansicht einer Kaffeestation bietet die Möglichkeit einen neuen geheimen Schlüssel zu generieren. In jedem Fall ist es möglich, die simplen Eigenschaften eines Objekts wie den Namen, Besitzer oder Preis zu ändern.

- **Archivierungsfunktion für Nutzer, Konten, Transponder und Produkte**
  Nutzer, Konten, Transponder und Produkte, die nicht mehr verwendet werden, können aus dem aktiven Betrieb herausgenommen werden, indem man sie archiviert. Sie verbleiben in der Datenbank können weiterhin eingesehen werden. Bei Bedarf können sie auch wiederhergestellt werden.

- **Archivierte Objekte entfernen**
  Durch Angabe eines Datums können alle bis zu diesem Datum archivierten Objekte endgültig aus der Datenbank entfernt werden.

- **Kontoauszüge erstellen**
  Zu jedem Konto kann ein Kontoauszug mit definiertem Anfangs- und Enddatum erstellt werden. Alle Transaktionen dieses Kontos finden sich darin wieder. Zu Archivierungszwecken können auch die Kontoauszüge aller Konten gleichzeitig erstellt und als .zip-Datei heruntergeladen werden.

- **Transaktionslog exportieren**
  Um den Verlauf aller Transaktionen langfristig zu sichern, kann dieser als .txt-Datei heruntergeladen werden.

- **Rechnungsschnitt**
  Durch Angabe eines Datums können alle bis zu diesem Datum getätigten Transaktionen endgültig aus der Datenbank entfernt werden.

- **Verwaltung des Master-Transponders**
  Durch Verknüpfung eines beliebigen Transponders mit einem speziellen Konto, dem `MASTER_ACCOUNT`, erhält dieser seine Funktion als Master-Transponder. Dadurch kann er Kaffeestationen in den Registrierungsmodus für neue Transponder versetzen.

6.2.5 Controller-Interface

Auch das Controller-Interface nutzt das HTTP-Protokoll, um mit den Kaffee-Clients zu kommunizieren, was die Implementierung auf Serverseite vereinheitlicht und vereinfacht (mit dem Tradeoff, dass die Kaffee-Clients HTTP implementieren müssen). Da alle Anfragen auf dieselbe URL erfolgen, werden sie auch von nur einer Handling-Methode verarbeitet. Die Nachricht wird decodiert, entschlüsselt und dann entsprechend dem Message Code verarbeitet (wie in Abschnitt 6.1 beschrieben). Es folgen Aufrufe der Applikationskern-API, deren Rückgabewerte wiederum verschlüsselt und codiert zurückgesendet werden.

Das Controller-Interface verwaltet die Kaffee-Clients in der Model-Klasse `Client`. Sie speichert den Client-Master-Schlüssel in verschlüsselter Form (*ChaCha20-Poly1305*), den aktuellen temporären Schlüssel in Klartext sowie den Nachrichtenzähler.

6.2.6 Sicherheitsmaßnahmen

Da es trotz aller Gründlichkeit und Vorsicht immer die Möglichkeit eines erfolgreichen Angriffs auf das Abrechnungssystem gibt, müssen entsprechende Sicherheitsmaßnahmen implementiert werden, die über die Verschlüsselung und Authentifizierung der Kommunikation hinausgehen. In einem solchen Fall sollten Angreifer so wenig Möglichkeiten zur Manipulation wie möglich bekommen. Dazu wurden die folgenden Maßnahmen ergriffen, die das Backend betreffen.

- **Verschlüsselung der Client Master Keys**
  Die geheimen Master-Schlüssel der Kaffee-Clients werden in der Datenbank des Controller-Interface gespeichert, da der Abrechnungskern keine Kenntnis von externen Abläufen wie der Controller-Kommunikation hat. Erlangt ein Angreifer Lesezugriff auf diese Daten, erlauben ihm die Master-Schlüssel sämtliche Kommunikation zwischen Kaffee-Clients und Backend einzusehen. Hat er den Kommunikationskanal in der Vergangenheit bereits abgehört und ausgetauschte Nachrichten gespeichert, kann er nun den gesamten Zahlungsverkehr rekonstruieren. Außerdem kann er sich durch die Verwendung der Schlüssel problemlos als beliebiger autorisierter Kaffee-Client ausgeben, falls nicht schnell genug auf diesen Angriff reagiert wird. Die einzige Gegenmaßnahme ist in diesem Fall, alle Client-Master-Schlüssel auszutauschen, was physischen Zugriff auf die Clients erfordert und somit nicht wünschenswert ist.

  Das Erlangen der Schlüssel wird dadurch maßgeblich erschwert, dass sie wiederum mit einem Datenbank-Master-Schlüssel verschlüsselt werden. Dazu wird dasselbe Verfahren, wie es bei der Kommunikation mit den Kaffee-Clients zum Einsatz kommt, verwendet. Dieser neue Master-Schlüssel muss natürlich so sicher wie möglich abgelegt werden. Jetzt muss ein Angreifer zusätzlich zum Zugriff auf die Datenbank auch Zugriff

auf diesen Schlüssel erlangen, um an die Client-Master-Schlüssel zu kommen. Gelingt dies nicht, ist das Austauschen der Client-Master-Schlüssel nicht erforderlich.

- **Beschränkung der Zugriffsrechte**
  Falls ein Angriff auf das Backend gelingt, ist das Ziel der Angreifer oft, eigene Dateien auszuführen, die die Möglichkeiten zur Manipulation erweitern (Bootkits, ...). Deswegen ist es ratsam, die Zugriffsrechte des Systemnutzers, der Apache ausführt, so weit wie möglich zu reduzieren. Hat dieser nicht die Möglichkeit Dateien auszuführen, sind auch die Möglichkeiten der Angreifer stark eingeschränkt.

6.3 Kaffee-Client

Der Kaffee-Client besteht aus einem Arduino Board inklusive Ethernet Shield und bildet zusammen mit einem RFID-Lesegerät, einem LCD-Display und einer Kaffeemaschine zusammen eine Kaffeestation. Die auf dem Arduino ausgeführte Controller-Software nennt sich *CoffeeController*. Diese wird durch ein weiteres Modul, dem *CoffeeDriver*, um Kommunikationsmöglichkeiten mit der Kaffeemaschine ergänzt. Das Ethernet Shield und seine Software erfüllen die Funktion des Ethernet-Interface. Alle weiteren Funktionen, wie die Nutzung des EEPROM und das AEAD-Verfahren ChaChaPoly, werden durch externe Software-Bibliotheken implementiert. Kommunikation mit dem Lesegerät erfolgt über das SPI-Protokoll, mit dem Display über I2C[2]. Die Peripheriegeräte und auch die Kaffeemaschine sind an den digitalen und anlogen I/O-Pins des Arduino angeschlossen.

Im Prototypen des Abrechnungssystems wurde der CoffeeController auf einem Arduino Uno Rev3 implementiert. Der zentrale Mikrocontroller in diesem Modell ist ein Atmel ATmega328P, der bei 16 MHz läuft. Der Arduino Uno verfügt über 2 kB SRAM, 32 kB Flash Speicher und 1 kB EEPROM sowie 14 digitale I/O-Pins [Ard17c]. Für die Netzwerk-Funktionalität kommt ein Ethernet Shield zum Einsatz.

6.3.1 CoffeeController

Der CoffeeController ist das Software-Modul, das den Kaffee-Client betreibt. Er ist im Wesentlichen ein Zustandsautomat, der auf Eingaben verschiedener Quellen reagiert. Im Grundzustand (`READY`) wartet er auf ein Signal des RFID-Lesegeräts, dass das Scannen eines Transponders zu erkennen gibt. Im Verlauf eines Produktbezugs durchläuft er dann verschiedene Zustände, wie in Abb. 6.4 dargestellt. Ein Produktbezug involviert drei Server-Anfragen, wie auch schon in Abb. 5.2 zu erkennen ist. Weitere Anfragen sind bei der Registrierung neuer Transponder mittels der Master-RFID-Karte oder dem Austausch eines neuen temporären Schlüssels notwendig.

Falls der CoffeeController jemals in einen dauerhaft blockierten Zustand gerät, kann er innerhalb von 15 Sekunden durch einen Neustart wieder funktionsfähig gemacht werden. Ein Druck auf die Reset-Taste auf dem Board genügt dazu.

[2]Sowohl Serial Peripheral Interface Bus (SPI) von Motorola als auch Inter-Integrated Circuit (I2C) von Philips Semiconductors sind serielle Bus-Protokolle, die für die Kommunikation verschiedener Hardware-Elemente über kurze Distanzen entwickelt wurden.

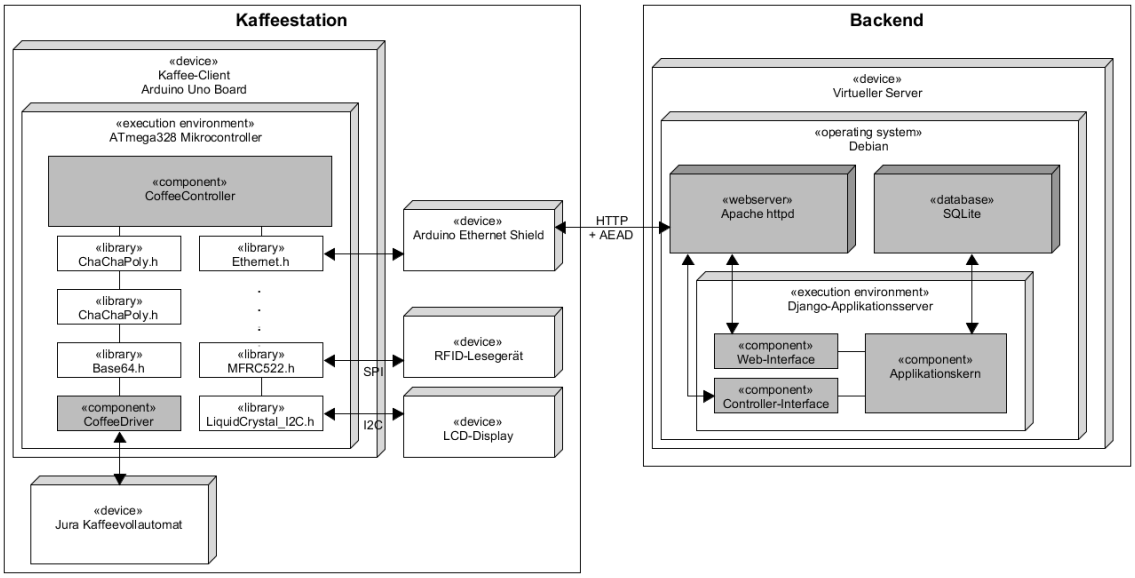

Abbildung 6.3: Komponentendiagramm von Kaffee-Client und Backend.

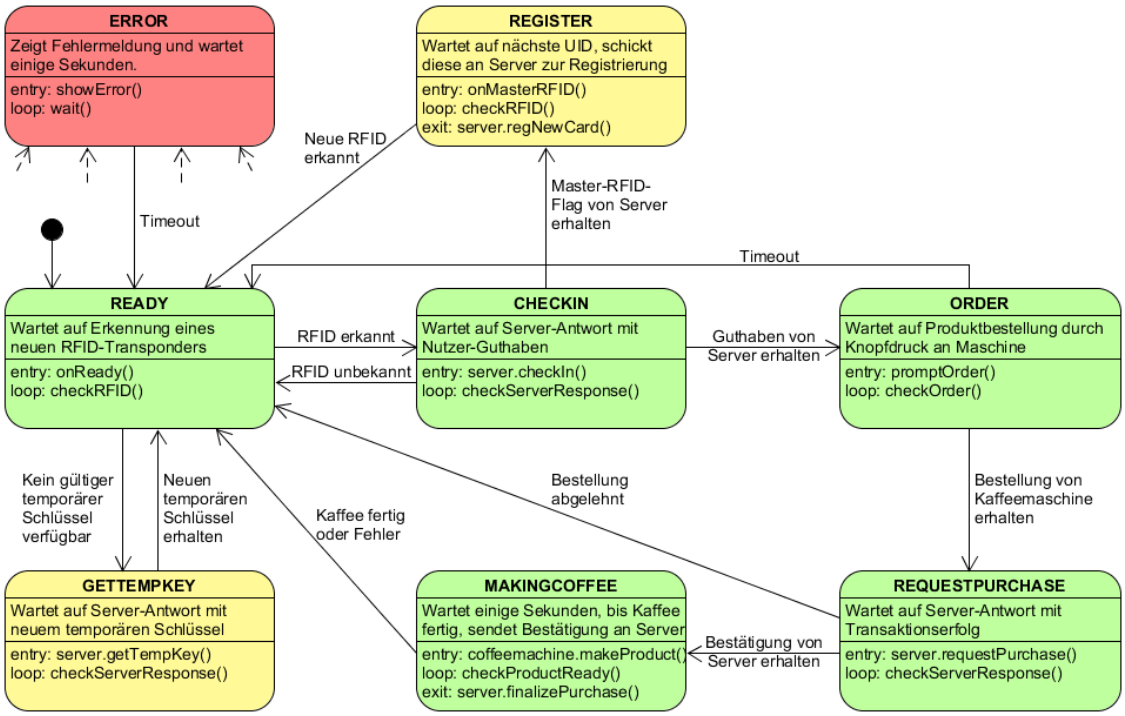

Abbildung 6.4: Zustandsdiagramm des CoffeeController. Der Ablauf eines Produktbezugs ist in grün dargestellt, seltenere Zustände in gelb und der Fehlerzustand in rot.

6.3.2 Code-Optimierungen

Die Hauptaufgabe des CoffeeController ist die Vermittlung von Information zwischen einer Vielzahl technischer Geräte, die jeweils andere Anforderungen an die Art der Kommunikation stellen. In Kombination mit dem Anspruch, sämtliche Kommunikation sicher gegenüber Angreifern zu gestalten, führt das zur Nutzung einer großen Anzahl an Software-Bibliotheken. Diese allein beanspruchen schon mehr als die Hälfte des auf dem Arduino Uno verfügbaren Programmspeichers (Flash Speicher). Andererseits ist für die Ver- und Entschlüsselung, sowie für die Konstruktion der Nachrichten relativ viel dynamischer Speicher (SRAM) notwendig. Aufgrund dieser Umstände war es extrem wichtig, den CoffeeController so effizient wie möglich zu implementieren. Im Folgenden wird auf einige Code-Optimierungen eingegangen.

- **Keine dynamische Speicherzuweisung**
  Bei der Programmierung von Embedded Systems im Allgemeinen ist die Verwendung dynamischer Speicherzuweisung zu vermeiden. Langfristig führt sie zur Fragmentierung des Speichers und letztendlich zu Fehlern, weil kein Speicherblock der benötigten Größe bereitgestellt werden kann. Darüber hinaus wurden auch Software-Bibliotheken wie die Arduino String-Bibliothek, die dynamische Speicherzuweisung nutzen, vermieden. Auf der gesamten Controller-Software werden ausschließlich C-Strings verwendet.

- **Statische Variablen in Flash-Speicher ablegen**
  Große, statische Variablen können in den Programmspeicher ausgelagert werden, um Platz im SRAM des Arduino zu sparen. Da in diesem Fall sowohl dynamischer als auch Programmspeicher knapp sind, ist das ein Tradeoff, der gut überlegt sein muss. Der Programmspeicher ist jedoch um ein Vielfaches größer als der SRAM (32 kB Flash vs 2 kB SRAM) und die dorthin verlagerten Daten fallen somit nicht zu sehr ins Gewicht. Bei den ausgelagerten Daten handelt sich hauptsächlich um Strings zur Ausgabe auf dem Display und für das Logging.

- **Logging deaktivieren**
  Typischerweise erfolgt das Debuggen eines Programms auf dem Arduino durch Logging über den Standard-Serial-Port (I/O-Pins 0 und 1), der Kommunikation mit einem PC über USB unterstützt. Deaktiviert man das komplette Logging, können einige hundert Byte Programmspeicher eingespart werden. Das liegt einerseits an dem Verzicht auf die `HardwareSerial`-Bibliothek (gut 100 Bytes), aber hauptsächlich an den entfallenen Logging-Statements und den statischen Strings (12 Bytes pro Statement plus Größe des Strings). Im Debugging-Modus wird das Logging wieder aktiviert, indem auf andere Funktionen verzichtet wird (z.B. den CoffeeDriver).

Mithilfe dieser Optimierungen konnte der CoffeeController erfolgreich auf dem Arduino Uno implementiert werden. Der Programmspeicher wird dabei jedoch zu 99% und der dynamische Speicher zu 89% beansprucht. Das Hinzufügen oder die Verbesserung von Funktionen könnte also ein großes Problem darstellen. Es gibt jedoch noch weitere Optimierungsmöglichkeiten, die das Potential haben, Raum für Erweiterungen zu schaffen.

- **Verwendung eines alternativen AEAD-Verfahren**
  Das AEAD-Verfahren *ChaCha20-Poly1305*, das bei der Kommunikation mit dem Kaffee-Client zum Einsatz kommt (siehe Abschnitt 6.1.2), ist zwar schon ein relativ effizientes

Verfahren. Doch es stehen noch einige Alternativen zur Auswahl, die noch weniger Anspruch an die Rechenleistung und Speicherkapazitäten stellen. Ein Beispiel ist der von der NSA entwickelte Verschlüsselungsalgorithmus *Speck*, der in Kombination mit allgemeinen AEAD-Konstruktionen wie *EAX* oder *GCM* allen Anforderungen gerecht wird. Allerdings sind hier die etwas längere Zeit für Verschlüsselungsoperationen und die zweifelhafte Sicherheit aufgrund seines Ursprungs zu berücksichtigen.

- **Nachrichtencodierung als Hexstrings statt Base64**
  Die Codierung der Kommunikation erfolgt über Base64 (siehe Abschnitt 6.1.3), was eine zusätzliche Software-Bibliothek erfordert. Eine alternative Methode ist die Codierung in Base16, oder auch Hexstring, die sehr kompakt selbst implementiert werden kann. Das würde einige hundert Byte Programmspeicher freigeben, allerdings die Vergrößerung der Nachrichten-Buffer zur Folge haben und so mehr dynamischen Speicher erfordern.

- **Kommunikation mit Backend über TCP**
  Momentan erfolgt die Kommunikation des Kaffee-Client mit dem Backend über das HTTP-Protokoll, was die Implementierung der Interfaces auf Serverseite vereinheitlicht. Eine effizientere Kommunikation könnte jedoch direkt über TCP erfolgen, da die HTTP-Header ohnehin weitestgehend ignoriert werden (da sie nicht gesichert sind). So könnte Flash-Speicher des Kaffee-Client freigegeben werden, der die Befehle zur Übertragung der HTTP-Header und die (statischen) Header selbst enthält. Darüber hinaus würde die Codierung der Nachrichten überflüssig, denn TCP stellt keine Anforderungen an den Inhalt des Nachrichtenrumpfs.

- **Minimale Versionen von Software-Bibliotheken implementieren**
  Die Funktionalität einiger Software-Bibliotheken, die in der CoffeeController-Software genutzt werden, wird nicht vollständig benötigt. Wo es die Lizenzen zulassen, könnte man eigene, leichtere Versionen dieser Bibliotheken implementieren, um Programmspeicher zu sparen.

6.4 Evaluierung

In diesem Abschnitt wird evaluiert, in welchem Umfang die Anforderungen an das Abrechnungssystem erfüllt wurden. Tabelle 6.3 gibt einen Überblick über die Sicherheits- und funktionalen Anforderungen, wie sie in den Abschnitten 3.3 und 3.4.3 definiert wurden. Die allgemeinen Anforderungen dagegen sind nicht präzise genug, um sie in dieser Art zu evaluieren. Sie dienten im Verlauf der Planung und Umsetzung des Systems lediglich als Orientierung.

6.4.1 Funktionale Anforderungen

Die funktionalen Anforderungen an das System konnten ausnahmslos erfüllt werden. Für Nutzer ist es möglich ihr Guthaben mittels ihres persönlichen Transponders einzusehen [FA1] und daraufhin ein Produkt zu beziehen [FA2], das mit dem Guthaben auf ihrem Konto abgerechnet wird [FA6]. Bei unzureichendem Guthaben kann das Abrechnungssystem die Ausgabe verweigern [FA3]. Der Nutzer wird während des gesamten Vorgangs durch

ID	Name der Anforderung	Erfüllt?
FA1	Guthabeneinsicht an Kaffeestation	✓
FA2	Kaffeebezug an Kaffeestation	✓
FA3	Kontrolle über Kaffeeausgabe	✓
FA4	Kaffeestationen geben Rückmeldung an Nutzer	✓
FA5	Registrierung eines Transponders an Kaffeestation	✓
FA6	Verlässliche Kontostände	✓
FA7	Buchführung über Transaktionen	✓
FA8	Anlegen, Modifizieren und Entfernen der Stammdaten	✓
FA9	Kontoauszüge erstellen	✓
FA10	Archivierung der Transaktionsdaten	✓
FA11	Rechnungsschnitt	✓
FA12	Integration beliebig vieler Kaffeemaschinen	✓
FA13	Statusabfragen der Kaffeestationen	✓
SA1	Abhörsichere Kommunikation mit Kaffeestationen	✓
SA2	In Kom. mit Kaffeest. kann nicht eingegriffen werden	✓
SA3	Zugriff auf Nutzerfunktionen wird nur durch registrierte Transponder gewährt	✗
SA4	Zugriff auf das Administrationstool ist den Administratoren vorbehalten	✓
SA5	Abhörsichere Kommunikation mit Administrationstool	✓
SA6	In Kom. mit Administrationstool kann nicht eingegriffen werden	✓

Tabelle 6.3: Evaluierung der Anforderungen an das Abrechnungssystem

einen Bildschirm an der Kaffeestation auf dem Laufenden über die aktuellen Abläufe gehalten [FA4]. Darüber hinaus ist es möglich, neue Transponder direkt an einer beliebigen Kaffeestation zu registrieren, indem zuvor ein Master-Transponder eingescannt wird [FA5].

Über das Administrationstool ist es möglich, Guthaben aufzuladen oder auszuzahlen, die Stammdaten des Abrechnungssystems zu verwalten [FA8], Kontoauszüge zu erstellen [FA9], Transaktionsdaten zu archivieren [FA10] und auch einen Rechnungsschnitt durchzuführen [FA11]. Die Transaktionen aller Produktkäufe und Guthabenänderungen werden persistent gesichert [FA7].

Das Abrechnungssystem ist dafür ausgelegt, beliebig viele Kaffeemaschinen zu integrieren. Es ist nur die Hardware einer Kaffeestation und ein kurzer Konfigurationsvorgang nötig, um eine neue Maschine anzuschließen [FA12]. Zudem bietet das Administrationstool einen Überblick über alle Kaffeestationen und ihren aktuellen Status [FA13].

6.4.2 Anforderungen an die Informationssicherheit

Auch die Sicherheitsanforderungen konnten größtenteils erfüllt werden. Durch den Einsatz des AEAD-Verfahrens werden Nachrichten zwischen Kaffee-Client und Backend sicher verschlüsselt übertragen, was Angreifern ein Abhören unmöglich macht [SA1]. Durch AEAD und die Implementierung von temporären Schlüsseln und Nachrichtenzählern konnte auch jede Manipulation der Kommunikation ausgeschlossen werden [SA2]. Allein die Störung der Kommunikation (Schutzziel: Verfügbarkeit, siehe Abschnitt 2.1.1) ist möglich, was jedoch bereits bei der Definition der Anforderungen berücksichtigt wurde. Auch die Kommunikation des Web-Client mit dem Backend läuft verschlüsselt [SA5] und manipulationsresistent ab [SA6], da das HTTPS Protokoll diese Eigenschaften sicherstellt.

Während die Administrationsfunktionen durch eine Passwortabfrage im Administrationstool geschützt werden [SA4], ist es jedoch möglich, wie in Abschnitt 5.1.2 beschrieben,von bestimmten Nutzerfunktionen ohne Berechtigung Gebrauch zu machen [SA3]. Diese Anforderung konnte im Rahmen dieser Arbeit als einzige nicht erfüllt werden.

7 Fazit

Das in dieser Arbeit vorgestellte Abrechnungssystem und der funktionstüchtige Prototyp zeigen, dass eine sichere Vernetzung und zentrale Verwaltung von Kaffeemaschinen auf Basis von kostengünstigen Mikrocontrollern möglich ist. Dieses Ergebnis ist natürlich genauso auf andere Anwendungsgebiete des vernetzten Alltags übertragbar. Der Prototyp des Systems wird künftig für eine übersichtliche Abrechnung des Kaffeekonsums der Mitarbeiter des LRZ sorgen und die manuelle Verwaltung ersetzen.

Durch die Erarbeitung des Anforderungskatalogs in Kapitel 3 konnte der Prototyp entsprechend der Situation am LRZ und den Bedürfnissen der Administratoren und Nutzer realisiert werden. Die Evaluierung in Abschnitt 6.4 ergab, dass die Anforderungen mit nur einer Ausnahme erfüllt werden konnten. Die Nutzer können ihre elektronischen Schlüssel zur Identifikation gegenüber dem System nutzen und so vollkommen bargeldlos Kaffee beziehen. Die Administratoren des Systems haben die volle Kontrolle über die Nutzerguthaben und alle angeschlossenen Kaffeemaschinen. Das Schlüsselmanagement der Kaffeestationen ist simpel und erfordert in der Regel nach der Einrichtung einer Station keine manuelle Wartung mehr. Mit Ausnahme von physischen Manipulationen der Kaffeestationen und der in Abschnitt 5.1.2 genannten Probleme mit der RFID-Transponder-Authentifizierung ist das System sicher gegen Angriffe durch Außenstehende.

7.1 Ausblick

An einigen Stellen ist eine Erweiterung des Abrechnungssystems in Zukunft sinnvoll. Zunächst sollte natürlich hinsichtlich der unsicheren Authentifizierung der Nutzer an den Kaffeestationen eine Entscheidung getroffen und entsprechende Sicherungsmechanismen implementiert werden. Falls die Authentifizierung eines Transponders durch das Zugangssystem des LRZ erfolgen kann, wäre diese Lösung komfortabel für die Nutzer, absolut sicher und deswegen optimal. Die Kaffee-Clients könnten ein gegenseitiges Authentifizierungsverfahren mit den Transpondern durchführen, wobei sie in Verbindung mit dem Backend stehen, das wiederum mit dem Zugangssystem kommuniziert. Die damit einhergehende Erweiterung der Kommunikation zwischen Kaffee-Clients und Backend ist problemlos in das bestehende Protokoll einzufügen. Andernfalls können separate Transponder für das Abrechnungssystem eingeführt werden, die zur Identifikation der Nutzer dienen.

Eine andere Erweiterungsmöglichkeit ist die Implementierung eines Webtools für die Nutzer, in dem sie ihren Kontoverlauf und Statistiken ihres Kaffeekonsums einsehen können. Da das Administrationstool sehr ähnliche Aufgaben erfüllt und entsprechende Strukturen im Abrechnungskern und in der Datenbank bereits vorhanden sind, ist die Umsetzung eines solches Webtools mithilfe des Django-Frameworks einfach durchzuführen. Auch Benachrichtigungen der Nutzer per E-Mail über aktuelle Vorgänge wie Aufladungen oder Abbuchungen und das Verschicken monatlicher Kontoauszüge wären eine naheliegende Erweiterung.

Außerdem sollen in Zukunft natürlich auch Kaffeemaschinen anderer Hersteller in das System integriert werden. Dazu ist nur die Entwicklung herstellerspezifischer Treiber (weiterer

CoffeeDriver) für die Kaffee-Clients nötig. Eine Erweiterung an anderer Stelle im Abrechnungssystem ist nicht erforderlich.

Während die Entwicklung neuer Funktionen auf Serverseite kein Problem darstellt, werden die verwendeten Arduino Uno Boards hinsichtlich ihrer Speicherkapazitäten bereits an ihre Grenzen gebracht. Die Erweiterung der Mikrocontroller-Software muss künftig also, falls notwendig, mit Bedacht und Fokus auf Effizienz erfolgen. Alternativ kann ein leistungsfähigerer Mikrocontroller als Client-Hardware verwendet werden, z.B. der Arduino Mega 2560.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Apa17] Apache Software Foundation: *Apache Documentation*. http://httpd.apache.org/docs/2.4/, 2017. Abgerufen am 06.06.2017.

[Ard14] Ardiri, Aaron: *Arduino - FAQ*. https://evothings.com/is-it-possible-to-secure-micro-controllers-used-within-iot/, 2014. Abgerufen am 01.09.2017.

[Ard17a] Arduino LLC: *Arduino - FAQ*. https://www.arduino.cc/en/Main/FAQ, 2017. Abgerufen am 01.09.2017.

[Ard17b] Arduino LLC: *Arduino - Introduction*. https://www.arduino.cc/en/Guide/Introduction, 2017. Abgerufen am 21.02.2017.

[Ard17c] Arduino LLC: *Arduino Uno Rev3*. https://store.arduino.cc/arduino-uno-rev3, 2017. Abgerufen am 22.06.2017.

[BO04] Britta Oertel, Michaela Wölk: *Risiken und Chancen des Einsatzes von RFID-Systemen*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ElekAusweise/RFID/RIKCHA_pdf.html, 2004. Abgerufen am 23.02.2017.

[Buc16] Buchmann, Johannes: *Einführung in die Kryptographie*. Springer Spektrum, Berlin, 6 Auflage, 2016. ISBN 978-3-6423-9775-2.

[Bur14] Bursztein, Elie: *Speeding up and strengthening HTTPS connections for Chrome on Android*. https://security.googleblog.com/2014/04/speeding-up-and-strengthening-https.html, 2014. Abgerufen am 25.06.2017.

[Dja17] Django Software Foundation: *Django Documentation*. https://docs.djangoproject.com/en/1.11/, 2017. Abgerufen am 06.06.2017.

[Fal15] Falch, Alexander: *Entwicklung eines web-basierten Verwaltungstools zur Protokollierung und Zugangssteuerung von Industriegeräten*. Bachelorarbeit an der Leopold-Franzens-Universität Innsbruck, Dezember 2015.

[Gue10] Gueron, Shay: *Intel Advanced Encryption Standard (AES) New Instructions Set*. https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set, 2010. Abgerufen am 03.07.2017.

[HPP07] Hunt, V. Daniel, Albert Puglia und Mike Puglia: *RFID - A Guide to Radio Frequency Identification*. Wiley Online Library, April 2007. ISBN 9780470107645.

[Jos06] Josefsson, Simon: *The Base16, Base32, and Base64 Data Encodings*. https://tools.ietf.org/html/rfc4648, 2006. Abgerufen am 24.06.2017.

[Kön15] König, Peter: *Arduino vs. Arduino - Der Graben wird tiefer.* https://www.heise.de/make/meldung/Arduino-vs-Arduino-der-Graben-wird-tiefer-2764692.html, 2015. Abgerufen am 23.02.2017.

[NL15] Nir, Yoav und Adam Langley: *ChaCha20 and Poly1305 for IETF Protocols.* https://tools.ietf.org/html/rfc7539, 2015. Abgerufen am 25.06.2017.

[NXP17] NXP Semiconductors: *MIFARE ICs - Product Overview.* http://www.nxp.com/products/identification-and-security/mifare-ics:MC_53422, 2017. Abgerufen am 16.06.2017.

[Rog02] Rogaway, Phillip: *Authenticated-Encryption with Associated-Data.* In: *ACM Conference on Computer and Communications Security*, Seiten 98–107. ACM press, 2002.

[Roh12] Rohrer, Maurus: *Bedienung und Verwaltung von Haushaltsgeräten mittels NFC am Beispiel eines Kaffeevollautomaten.* Projektarbeit an der Hochschule für Technik und Wirtschaft Berlin, September 2012.

[Sie16] Siering, Peter: *Sharespresso - NFC-Bezahlsystem für Kaffeevollautomaten.* In: *c't*, Band 02/2016, Seiten 110–113. Heise Medien GmbH, 2016. Abgerufen am 21.02.2017.

[Spi11] Spitz, Stephan: *Kryptographie und IT-Sicherheit : Grundlagen und Anwendungen.* Vieweg + Teubner, Wiesbaden, 2 Auflage, 2011. ISBN 978-3-8348-1487-6.

[TR01] Treese, Win und Eric Rescorla: *The Transport Layer Security (TLS) Protocol Version 1.1.* https://tools.ietf.org/html/rfc4346#appendix-F.2, 201. Abgerufen am 25.06.2017.

[vT11] Tilborg, Henk C.A. van: *Encyclopedia of Cryptography and Security.* Springer, New York, 2 Auflage, 2011. ISBN 978-1-4419-5906-5.

[Wea] Weatherley, Rhys: *ArduinoLibs - CryptographicLibrary Documentation.* https://rweather.github.io/arduinolibs/crypto.html. Abgerufen am 26.06.2017.

[Wil08] Willems, Wolfgang: *Codierungstheorie und Kryptographie.* Birkhäuser Verlag AG, Basel, 2008. ISBN 978-3-7643-8611-5.