DIPLOMARBEIT

Sicherheitsmodul für ein Smart Metering Gateway

Eingereicht an der
Fakultät für Elektrotechnik und Informationstechnik,
Technische Universität Wien.
Ausgeführt zur Erlangung des akademischen Grades eines
Diplom-Ingenieurs

unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr. techn. Dietmar Dietrich
Institutsnummer: 384
Institut für Computertechnik

und

Univ.Ass. Dipl.-Ing. Georg Kienesberger
Institutsnummer: 384
Institut für Computertechnik

von

Peter Mahlknecht
Matr.Nr. 0526314
Redtenbachergasse 10/17
1160, Wien

September 2014 _____

Kurzfassung

Die verteilte und volatile Einspeisung von Elektrizität aus erneuerbaren Energien in das Niederspannungsnetz nimmt stetig zu. Deshalb ist eine Modernisierung der Versorgungsinfrastruktur zu Smart Grids nötig. Security ist hierbei eines der Gebiete in denen besondere Herausforderungen entstehen. Smart Metering, als eine Smart-Grid-Anwendung, dient dem Fernsteuern und Fernauslesen von Energiemessgeräten. Dabei wird auf bidirektionale Kommunikationsdienste und Intelligenz in Form von Rechnern gesetzt. Infolge der hierfür obligatorischen Datenverbindung reicht der bisherige Zugriffsschutz (bauliche Maßnahmen bzw. Schaltschränke) nicht mehr aus um die gespeicherten und verarbeiteten Daten zu schützen. Ohne zusätzliche Maßnahmen im Securitybereich stellt Smart Metering daher eine Bedrohung für die Privacy der Endverbraucher sowie eine Gefahr für die Datensicherheit dar, welche zu finanziellen und physikalischen Schäden führen kann. Im Rahmen dieser Arbeit wurde ein Sicherheitsmodul für Smart Grid Metering Gateways entwickelt, mit dessen Unterstützung eine geschützte Datenspeicherung sowie Datenübertragung zwischen dem Endverbraucher und der Stromversorgungsinfrastruktur ermöglicht wird. Das Sicherheitsmodul wurde hierbei in Form eines *Java Card* Applets für programmierbare Smartcards realisiert und bietet besonderen Schutz vor Angreifern mit physikalischem Zugriff. Es stellt dem Gateway kryptografische Dienste für Verschlüsselung, Schlüsselvereinbarung, digitale Signaturen sowie einen geschützten Speicher für Daten und Schlüsselmaterial zur Verfügung. Neben der Definition der Anforderungen und der Entwicklung eines Prototyps wurde das Sicherheitsmodul in das *OGEMA*-Framework für Smart Metering Gateways eingebunden und auf einem Embedded System evaluiert. Hierbei konnte gezeigt werden, dass Smartcards eine geeignete Plattform für Sicherheitsmodule in Smart Metering Gateways sind, mit deren Hilfe auch verschiedene regulatorische Vorgaben bezüglich Datensicherheit und Privacy erfüllt werden können. Weiters wurden genaue Anforderungen für die einzusetzenden Smartcards ermittelt, sowie eine generische Softwarebibliothek zur Nutzung des Sicherheitsmoduls in Java-Systemen entwickelt.

Abstract

The distributed and volatile feeding of electricity from renewable energy sources into the low voltage network is steadily increasing. This makes the modernization of the supply infrastructure to smart grids necessary. Now security has become one of the areas with manifold new challenges. Smart metering is a smart grid application used for remote control and reading of energy meters. It employs bidirectional communication services and intelligence in the form of computers. As a result of the mandatory data connection, the previously utilized access protection (structural measures or cabinets) is no longer adequate to protect the stored and processed data. Smart Metering without additional security measures is therefore a threat to the privacy of the consumers and to data security which can lead to financial and physical damage. The result of this thesis is a security module for Smart Grid Metering Gateways that supports the gateway with protected data storage and transfer. The security module was realized as a Java Card applet for programmable smart cards and provides special protection from attackers with physical access. It offers cryptographic services like de- and encryption, key agreement, digital signatures, as well as a protected storage for data and keys for the gateway. In addition to the definition of requirements and the development of a prototype, the security module was integrated into the OGEMA framework for smart metering gateways and evaluated on an embedded system. The thesis demonstrates that smart cards are a suitable platform for security modules in smart metering gateways and that they can be used to meet different regulatory requirements regarding data security and privacy. Additionally precise requirements for the smart cards were acquired and a generic software library for the use of the security module in Java systems was developed.

Danksagung

An dieser Stelle möchte ich mich bei den vielen Personen bedanken, die mich während meines Studiums unterstützt und begleitet haben.

Besonders möchte ich mich bei meinen Eltern bedanken, die mir das Studium ermöglicht haben. Ihrer Unterstützung konnte ich mir immer sicher sein, ebenso wie ihres Verständnisses und aufmunternder Worte, welche mir durch Zeiten fehlender Motivation geholfen haben.

Meiner Freundin Anna danke dafür, dass du ein Teil meines Lebens bist.

Meinen Studienkollegen und Freunden möchte ich für die schöne Zeit danken, ebenso wie für die Hilfen, wenn sie notwendig waren.

Danken möchte ich auch Herrn Dipl.-Ing. Georg Kienesberger und Herrn Prof. Dr. Dietmar Dietrich, welche mir ermöglicht haben diese Diplomarbeit am Institut für Computertechnik durchzuführen und mich dabei betreut haben.

Danke

Inhaltsverzeichnis

Abkürzungen

AES	Advanced Encryption Standard
AID	Applet Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Certificate Authority
CBC	Chain-Block-Cipher
CLA	Class-Byte
CMAC	Cipher-based Message Authentication Code
CP	Certificate Policy
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DF	Dedicated File
DSA	Digital Signature Algorithm
DSL	Digital Subscriber Line
DSM	Demand Side Management
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
ECDLP	Elliptic Curve Discrete Logarithm
ECDSA	Elliptic Curve Digital Signature Algorithm
EF	Elementary File
FID	File Identifier
HAN	Home Area Network
IEC	International Electrotechnical Commission
IMA-VO	Intelligente Messgeräte-AnforderungsVO
IME-VO	Intelligente Messgeräte-Einführungsverordnung
INS	Instruction
IP	Internet Protocol
ISO	International Organization for Standardization
IV	Initialisierungsvektor
JCIDE	Java Card Integrated Development Environment
JCOP	Java Card Operating System
JCWDE	Java Card Workstation Development Environment
JVM	Java Virtual Machine
LAN	Controllable Local Systems
LMN	Local Meterological Network
MAC	Message Authentication Code
M-Bus	Meter-Bus
MF	Master File
MUC	Multi Utility Communication
NIST	National Institute of Standards and Technology
NVRAM	Non Volatile Random Access Memory

NXP	Next eXPerience
OGEMA	Open Gateway Energy Management
OID	Object Identifier
OSI	Open Systems Interconnection
PACE	Password Authenticated Connection Establishment
PAKE	Password Authenticated Key Exchange
PII	Personally Identifiable Information
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RAM	Random Access Memory
RMI	Remote Method Invocation
SELMA	Sicherer Elektronischer Messdaten-Austausch
SFI	Short File Identifier
$(SG)^2$	Smart Grid Security Guidance
SGMS	Smart Grids Modellregion Salzburg
SHA	Secure Hash Algorithm
SMaDA	Smart Metering und Datenschutz in Österreich
SMGW	Smart Metering Gateway
SSC	Send Sequence Counter
SyM^2	Synchronous Modular Meter
TLS	Transport Layer Security
TLV	Tag Length Value
TR	Technische Richtlinie
USB	Universal Serial Bus
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e. V.
WAN	Wide Area Network
WiMAX	Worldwide Interoperability for Microwave Access

1 Einleitung

Energie und Elektrizität sind die treibenden Faktoren hinter vielen technologischen und wirtschaftlichen Entwicklungen. Die steigenden Preise von fossilen Rohstoffen und die Abkehr von nuklearer Energiegewinnung erfordern eine Modernisierung der Elektrizitätsversorgungsinfrastruktur. Nur so können die entstehenden Herausforderungen bewältigt werden. Der Ausbau der vorhandenen Netze zu Smart Grids soll dabei helfen den Stromverbrauch zu verringern, erneuerbare Energien besser zu nutzen und das Elektrizitätsnetz im Allgemeinen effizienter zu gestalten. Die Integration von bidirektionalen Kommunikationsdiensten und die Einbringung von Intelligenz in Form von Rechnern bringen jedoch eigene Herausforderungen mit sich.

1.1 Smart Grid

Smart Grid ist ein Begriff aus dem Englischen, mit dem ein intelligentes Stromnetz beschrieben werden soll, welches sich vor allem durch Kommunikation zwischen allen Netzteilnehmern und Komponenten auszeichnet. Als Stromnetz ist hier die komplette Infrastruktur für die Erzeugung, Übertragung und Verteilung elektrischer Energie gemeint. Die genaue Definition, was ein Smart Grid ist und welche Aufgaben es zu erfüllen hat, hängt von der definierenden Organisation ab. Grundlage für die weitere Beschreibung von Smart Grids sollen vor allem folgende zwei Definitionen sein:

- *European SmartGrids Technology Platform*: „An electricity network that can intelligently integrate the actions of all users connected to it – generators, consumers and those that do both – in order to efficiently deliver sustainable, economic and secure electricity supplies.“ [ESG12]
- *SmartGrids Austria Technologieplattform*: „Smart Grids sind Stromnetze, welche durch ein abgestimmtes Management mittels zeitnaher und bidirektionaler Kommunikation zwischen Netzkomponenten, Erzeugern, Speichern und Verbrauchern einen energie- und kosteneffizienten Systembetrieb für zukünftige Anforderungen unterstützen.“ [SGA10]

Beide Definitionen geben bereits die Hauptmotivation für Smart Grids an, die Sicherung und Effizienzerhöhung der Versorgung. Die schon vorhandene Elektrizitätsversorgungsinfrastruktur soll hierfür um bidirektionale Kommunikation erweitert werden, wodurch sich neue Möglichkeiten für das Energiemanagement ergeben. Die Einführung von Smart Grids soll ein wesentlicher Faktor bei der Bewältigung verschiedener Herausforderungen und Probleme sein. Von der *European SmartGrids Technology Platform* werden hierfür drei treibende Faktoren ausgemacht [ESG06]:

- der Europäische Binnenmarkt: Durch die Liberalisierung und Harmonisierung des Marktes (durch ein einheitliches Reglement als auch durch den Zusammenschluss der verschiedenen europäischen Netze) soll die Flexibilität des Marktes erhöht werden, und neue Produkte und Dienstleistungen ermöglichen. Dadurch sollen die Wahlmöglichkeiten für die Verbraucher erhöht, sowie finanzielle Einsparungen erreicht werden.
- Die Sicherung der Versorgung und ihrer Qualität: Die Modernisierung der Infrastruktur (Netz und Kraftwerke), sowie die Erhöhung des Automatisierungsgrads sollen nicht nur die Versorgung und ihre Qualität sichern, sondern auch den Energieverbrauch vermindern und die Effizienz der Übertragung steigern. Zudem sollen Engpässe an fossiler Primärenergie durch die bessere Nutzung regenerativer Energieerzeuger abgeschwächt werden. Die Entwicklung von zentraler zu verteilter Energieerzeugung erfordert neue Managementmechanismen wie Demand Side Management, d. h. die Steuerung des Energiebedarfs durch das Smart Grid.
- die Umwelt: Die Erzeugung von Treibhausgasen soll durch die Senkung des Energieverbrauchs und die bessere Einbindung von regenerativen Energien erreicht werden. Beides kann durch Smart Grids erleichtert werden.

Die traditionelle Infrastruktur beruht vor allem auf wenigen zentralen Kraftwerken mit hoher Leistung. Diese befinden sich meist in der Nähe von notwendigen Ressourcen wie Kohle oder Wasser [ESG06] und sind durch ein Hochspannungsnetz verbunden, durch welches die Mittelspannungs- und Niederspannungsverteilungsnetze versorgt werden. Der Energiefluss findet dabei charakteristischerweise nur in eine Richtung statt. Die Netze werden dabei meist von lokalen Monopolisten betrieben und stehen unter Aufsicht von Regulierungsbehörden.

Mit dem Umbau der vorhanden Infrastruktur zu Smart Grids soll ein bidirektionaler Energie- und Datenfluss zur definierenden Charakteristik werden [SGA10]. Der bidirektionale Energiefluss soll die gewünschte Integration dezentraler Erzeugung und Speicherung ermöglichen. Dabei sollen vor allem regenerative und umweltfreundliche Energieträger verstärkt genutzt werden. Die Notwendigkeit verteilter Erzeugung ergibt sich daraus, dass ihre Nutzung nahe am geografischen Vorkommen erfolgen soll, oder oft auch muss [SGA10]. Die Einführung einer bidirektionalen Datenkommunikation hat mehrere Gründe. So erfordert der effiziente Einsatz dezentraler Erzeugung, eine Möglichkeit die Kraftwerke und Speicher zu managen und zu steuern. Weiters ist ein automatisiertes Messen und Fernablesen der Stromzähler gewünscht, sogenanntes Smart Metering. Hierdurch soll ein flexiblerer Strommarkt ermöglicht werden. Ein weiterer wichtiger Grund ist die Einführung neuer Technologien zur Steuerung des Stromverbrauchs. Damit ist zum einen die Umsetzung von Verfahren wie Demand Side Management (DSM) gemeint, aber auch die reine Übermittlung von aktuellen Daten zu Strompreis und -verbrauch an den Kunden, um dadurch sein Verhalten zu beeinflussen [ESG06]. Smart Grids sollen also eine bessere Integration von vorhandenen und neuen Technologien in das Energienetz ermöglichen. So wurden für die *Smart Grids Modellregion Salzburg* [SGMS13] fünf Anwendungsgebiete definiert: Integration erneuerbarer Energien in die Verteilernetze, Integration der Elektromobilität, Integration von Haushaltskunden, Integration von Gebäuden, sowie Lastflexibilisierung in Gewerbe und Industrie. Es existieren jedoch noch weitere Anwendungsgebiete, wie etwa die Integration von Energiespeichern.

1.2 Sicherheitsbedenken in Smart Grids

Der Ausbau der traditionellen Energieversorgungsinfrastruktur zu einem Smart Grid sieht den weitreichenden Einsatz von Datenkommunikation für das Management vor. War dies auch bisher schon der Fall, so wird der Datenfluss jetzt auch bis zum Endverbraucher ausgedehnt.

Da die Datenkommunikation nicht nur ein Zusatzdienst, sondern ein integraler Bestandteil des Smart Grids werden soll, muss diese besonders geschützt werden. Versuchte Angriffe auf die Kommunikationsinfrastruktur sind, bedenkt man die Entwicklungen der Cyberkriminalität im Internet, durchaus wahrscheinlich. Denkbar sind dabei verschiedene Arten von Angriffen [LLS12]:

- Angriffe auf Geräte: Die Kontrolle über ein Gerät des Smart Grids soll übernommen werden.
- Angriffe auf Daten: Einfügen, Löschen oder Veränderung von Daten, sodass falsche Steuerungsentscheidungen getroffen werden, oder finanzielle Schäden entstehen.
- Angriffe auf Privacy: Gewinn privater Informationen, durch die Analyse der Verbrauchsdaten.
- Angriffe auf die Verfügbarkeit der Kommunikation: Zielt auf die Verzögerung oder den Ausfall der Kommunikationsverbindung (durch Überlastung der Rechen- oder Übertragungskapazität) ab.

Auf die verschiedenen Bedrohungen beim Smart Metering wird noch einmal in Abschnitt *2.2 Security und Privacy* eingegangen.

1.3 Smart Metering Gateway

Durch die gewünschte Erweiterung der Datenkommunikation bis zum Endverbraucher müssen bei diesem auch Geräte vorhanden sein, welche Daten zur Verfügung stellen, oder verarbeiten, sowie zur Datenübermittlung geeignet sind. Mit der Einführung von Smart Metering (siehe Abschnitt 2.1) in verschiedenen Ländern, sind diese oft bereits vorhanden. Dabei wurden im allgemeinen Geräte verwendet, welche alle drei Aufgaben (Bereitstellung, Verarbeitung und Übermittlung von Daten) in einer Einheit, dem Smart Meter vereinen [FLM10]. In Zukunft sollen jedoch mehrere Geräte mit verschiedenen Aufgaben bei den Verbrauchern vorhanden sein, von denen jedes an dem Datenaustausch beteiligt sein soll. In diesem Fall ist es sinnvoll die Aufgabe der Datenübermittlung einem dedizierten Gerät, dem Gateway zu übergeben [BSE13-1]. Da die Hauptaufgabe des Gateways im Zusammenhang mit Smart Metering gesehen wird, wird es meist als Smart Metering Gateway (SMGW) bezeichnet. Der Einsatz eines SMGW hat den Vorteil, dass die Kommunikation zur Außenwelt an einem Punkt zusammenläuft, und von dort gesteuert und gesichert werden kann. Dadurch muss nur eine Verbindung zur externen Welt (WAN, Wide Area Network) vorhanden sein [SGMS13]. Dem Gateway können natürlich auch weitere Aufgaben, wie das Speichern und Auswerten von Messergebnissen, oder die Filterung von Daten, sowie Aufgaben der Heimautomatisierung übertragen werden. In vielen Fällen werden die Smart Metering Gateways mit den Stromzählern in ein gemeinsames Gehäuse integriert werden [BSI13-1]. Für den Endverbraucher werden sie in solchen Fällen wahrscheinlich nicht, bzw. nur als Zähler wahrgenommen werden. Der Einsatz eines

einzigen Gateways für mehrere Haushalte oder Zähler ist auch geplant, evtl. sogar für verschiedene Energieträger wie Wärme oder Gas als sogenannte *Multi Utility Unit* [EKS11]. Aus Überlegungen zu Sicherheit und Privacy ist jedoch ein Gateway pro Nutzer vorzuziehen. Auch die, zumindest logische Trennung der Gateway- und Messfunktionalität, sollte deshalb durchgeführt werden. Hierdurch soll der Zugriff auf Daten durch Dritte weiter eingeschränkt werden, sowie eine sicherere Plattform, mit weniger Angriffsfläche geboten werden. Für die Kommunikation zwischen dem SMGW und dem Netzbetreiber soll entweder auf vorhandene Verbindungen wie das Telefonnetz, DSL-Anschluss (Digital Subscriber Line), Kabelnetz bzw. Mobilfunk zurückgegriffen werden, oder neue wie Powerline-Übertragung, oder drahtlose Verbindungen wie *WiMAX* (Worldwide Interoperability for Microwave Access) genutzt werden [EKS11]. Für die Kompatibilität mit diesen verschiedenen Kommunikationsformen und eine weitgehende Interoperabiltiät sollte die darüber abgewickelte Kommunikation über das Internet Protocol (IP) erfolgen [BSI13-6].

1.4 Problembeschreibung

Durch die von der Europäischen Union geplante europaweite Forcierung von Smart Grids [ESG06, EDL2006, EC09] wird die Anzahl der damit verbunden Geräte stark ansteigen. Die beim Endkunden vorrangig anzutreffenden Geräte werden im ersten Schritt Smart Meter und Smart Metering Gateways sein. Ob sich dabei die Trennung der beiden Geräte durchsetzt, oder diese als eine Einheit verteilt werden muss sich erst zeigen. Der Einsatz dieser SMGWs ist nur sinnvoll, wenn diese in ein Kommunikationsnetz eingebunden werden. Das Smart Grid bildet dann ein sogenanntes Cyber-Physical System [EKS12]. Durch die steigende Ausbreitung dieses Systems, die offene und dezentrale Architektur, sowie die weitreichenden Aufgaben die es übernehmen soll, entsteht für Angreifer ein attraktives Ziel. Die Bandbreite der denkbaren Angreifer ist dabei groß und kann grob in verschiedene Klassen unterteilt werden: Cyber-Terroristen, Endkunde, Kriminelle, sowie Mitarbeiter oder Dienstleister von Infrastrukturbetreibern [EKS12]. Die Motivationen hinter den Attacken werden dabei meist monetäre oder ideologische Gründe sein [FLM10]. Der teilweise geplante Einsatz des Internet, bzw. zumindest von IP-basierter Kommunikation für die Kommunikation in Smart Grids hat sowohl Vor- als auch Nachteile [EKS12]. Ein Nachteil ist, dass durch die dadurch entstehende *Monokultur* und die Überlappung mit dem Internet eine Ausbreitung von Schadsoftware und die Gefahr durch traditionelle Angreifer steigt. Der größte Vorteil ist natürlich die Einsparung von Kosten. Durch den Einsatz bestehender Infrastruktur, sowie etablierter Techniken können ein teurer Neuaufbau und Neuentwicklung vermieden werden. Weiters können die mit dem Internet gesammelten Erfahrungen, sowie Techniken zum Schutz von Daten und Privacy auch für Smart Grids angewandt werden.

Da es sich bei dem Smart Grid um kein reines Datennetz handelt, sondern es zudem aus physikalischen Komponenten, mit Wirkungen auf die Umwelt und Personen (Cyber-Physical System) handelt, muss von Anfang an auf die Sicherheit des Systems geachtet werden. Eine wesentliche Gefahr geht nach aktuellen Einschätzungen [EKS12] von ungenügend abgesicherten Geräten für das Smart Metering aus. Da das Smart Metering Gateway die zentrale Datenverbindung vom Kunden zum Smart Grid darstellt, ist es der ideale Ansatzpunkt zur Steigerung der Datensicherheit und Privacy

von Privatpersonen. Deshalb wurden in den letzten Jahren von verschiedenen Seiten Sicherheitskonzepte für das Smart Metering entwickelt. Einige davon sind [FZK12]:

- das *EDL40* Lastenheft [VDE10] vom Forum Netztechnik/Netzbetrieb im VDE (VDE Verband der Elektrotechnik Elektronik Informationstechnik e.V)
- das *Synchronous Modular Meter (SyM²)* Pflichtenheft [VDE09] (ebenfalls vom Forum Netztechnik/Netzbetrieb im VDE)
- das *SELMA*-Projekt (Sicherer Elektronischer Messdaten-Austausch) [SCHAUB03]
- das Schutzprofil für die Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen vom deutschen Bundesinstitut für Sicherheit in der Informationstechnik (BSI)
- das Schutzprofil für das Sicherheitsmodul der Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen (BSI)
- diverse Technische Richtlinien zu den Schutzprofilen des BSI

Die Schutzprofile des BSI sind am umfangreichsten, und umfassen neben der eichkonformen Sicherung und Übertragung von Messwerten, und der Sicherung der externen Kommunikation auch die zwingende Trennung zwischen Gateway und Zählern [BSI13-1]. Um die Sicherheit der Smart-Metering-Komponenten beim Kunden zu erhöhen sehen die BSI Schutzprofile und das SELMA-Projekt den Einsatz eines Sicherheitsmoduls vor. Dieses soll dem mit der Infrastruktur kommunizierenden Gerät kryptografische Dienste zur Verfügung stellen. Der Einsatz eines dedizierten Sicherheitsmoduls soll dabei vor allem gegen Angreifer mit physikalischem Zugriff auf das Gerät schützen. Auch in Österreich wurden und werden verschiedene Projekte zum Thema Sicherheit und Datenschutz durchgeführt. Dazu zählen das *Smart Grid Security Guidance ((SG)²)* [SBKK12], das *Smart Metering und Datenschutz in Österreich (SMaDA)* Projekt, sowie *das Metering & Privacy – Smart Metering and the protection of privacy of consumers* Projekt [REN10]. Durch die *Intelligente Messgeräte-Einführungsverordnung* [BGB12] vom 24. April 2012 sind die Netzbetreiber zur schrittweisen Einführung von Smart Metern verpflichtet. An die Messgeräte werden bindende Anforderungen gestellt, welche in der *Intelligente Messgeräte-AnforderungsVO* [BGB11] von der *E-Control* definiert wurden. Darin ist eine Absicherung und Verschlüsselung nach Stand der Technik vorgeschrieben, jedoch nicht weiter definiert.

1.5 Aufgabenstellung

Wie in der in Abschnitt 1.4 umrissenen Problembeschreibung dargestellt, entstehen durch den Aufbau eines Smart Grids viele sicherheitstechnische Herausforderungen. Einen besonders verwundbaren Punkt der Infrastruktur stellen dabei die beim Endverbraucher platzierten Systeme dar. Dies liegt daran, dass der physikalische Zugriff auf diese, aber auch auf die Kommunikationsverbindung zwischen Infrastrukturbetreiber und diesen nur schwer und unzureichend geschützt werden kann. Für den Betrieb von Smart Metering Systemen werden deshalb vielfach der Einsatz von Sicherheitsmodulen vorgesehen (siehe Projekt *SELMA* Abschnitt 2.3.2 und BSI Schutzprofile 2.3.3). Dadurch soll die Sicherheit der Daten und der Zugriff auf die Infrastruktur besser geschützt werden.

Ziel dieser Arbeit ist es, ein Sicherheitsmodul für Smart Metering Gateways zu entwickeln. Hierzu sollen die Anforderungen an das Sicherheitsmodul definiert werden, und die dafür zu verwendende Plattform festgelegt werden. Weiters soll die Integration des Sicherheitsmoduls in die Software eines Smart Metering Gateways erfolgen, sowie auf einem Embedded System, welches als Plattform für ein Gateway verwendet wird, getestet werden.

2 Stand der Technik

Im folgenden Kapitel wird der Stand der Technik betreffend Smart Metering, die dabei auftretenden Sicherheits- und Privacy-Herausforderung, sowie diesbezügliche Sicherheitskonzepte und gesetzliche Regelungen betrachtet. Ebenso werden die Konzepte der Smart Metering Frameworks OpenMUC und OGEMA betrachtet

2.1 Smart Metering

Intelligente Messgeräte, sogenannte Smart Meter sind ein wichtiger Bestandteil des Smart Grid. Sie zeichnen sich durch eine bidirektionale Kommunikationsverbindung aus. Diese verbindet sie mit einem Informationsverarbeitungssystem, typischerweise dem des Netzbetreibers. Verwendung findet die Datenverbindung um Messdaten zu übertragen, Parametrierungen durchzuführen, sowie Schaltaufgaben anzuweisen [SGMS13]. In vielen Fällen wird auch eine Schnittstelle für den Verbraucher zur Verfügung gestellt, welche die aktuellen Verbrauchsdaten, sowie evtl. weitere Informationen, etwa zu den aktuellen Preisen ausgibt. Das Smart Metering umfasst das gesamte IKT-System für den Betrieb der Smart Meter, d. h. es umfasst zusätzlich zu den intelligenten Messgeräten, die Kommunikationsverbindung, sowie das Informationsverarbeitungssystem an welches die Messgeräte angebunden werden [SGMS13]. Die dabei verarbeiteten Informationen umfassen neben den Verbrauchsdaten, auch Gerätedaten, Nutzerdaten sowie Daten über das System [EKS11]. Einige Funktionen welche das Smart Grid bieten soll können auch ohne Smart Meter realisiert werden. Sie sind jedoch für viele Anwendungsszenarien notwendig. Nur mit Smart Metern kann z. B. die Integration der Haushaltskunden durch Verbrauchsfeedback oder zeitlich differenzierte Verbrauchsabrechnung erfolgen. Smart-Metering-Systeme können in intelligenten Stromnetzen verschiedene Aufgaben übernehmen [SGMS13]:

- Liefern von Messdaten vor Ort, z. B. für Heimautomatisierungssysteme,
- Lieferant zeitnaher Verbrauchsdaten, als Information für Kunden, sowie für die Abrechnung mit dem Lieferanten
- Zeitgenaue Tarifierung, im Speziellen von Anreiztarifen, mit dem Ziel die Spitzenlast zu senken,
- Messgerät für den Netzbetrieb, z. B. für Spannung, Leistung und Phase der Elektrizitätsversorgung beim Verbraucher,
- Ermittlung von Netzverlusten oder Netzstörungen.

Die Nützlichkeit von Smart Metern wird auch dadurch unterstrichen, dass ihre Einführung von der Europäischen Richtlinie 2006/32/EG [EDL2006] gefördert wird und in Österreich durch eine gesetzliche Verordnung vorgeschrieben ist (siehe Abschnitt 2.3.1). Die Kommunikationsverbindung des Smart Meters zum Informationsverarbeitungssystem des Smart Grid kann direkt von diesem übernommen werden. Wie in Abschnitt 1.3 bereits dargestellt, ist in vielen Fällen der Einsatz eines Gateways vorzuziehen.

2.2 Security und Privacy in Smart Grids

Die Sicherheitsbedenken durch den Einsatz und Ausbau von Smart Grids wurden in Abschnitt 1.2 bereits angesprochen. Für einen ganzheitlich sicheren Betrieb eines Smart Grids im Sinne von Security und Privacy wurden von einer Task Force des Projekts Smart Grids Modellregion Salzburg (SGMS) ein auf drei Säulen basierendes Konzept definiert [SGMS13]:

- Die technische Security umfasst die Sicherheit der technischen Komponenten (Hardware, Datenübertragung, Softwaresysteme) (siehe Abschnitt 2.2.1).
- Der Datenschutz behandelt den Schutz vor Missbrauch personenbezogener Daten (siehe Abschnitt 2.2.2).
- Trust soll das Vertrauen und die Akzeptanz der Kunden zum System und zur Verarbeitung und Nutzung Ihrer Daten beinhalten. Dies soll z. B. durch offenen Dialog, informelle Selbstbestimmung, Information und die Möglichkeit zielgerichteten Feedbacks erreicht werden.

Grundlage für diese drei Säulen soll eine „exakt definierte und dokumentierte Organisation der Betriebsführung der IKT-Systeme“ [SGMS13 Seite 62] sein.

2.2.1 Technische Security – Sicherheitskonzepte

Die technische Security von Informations- und Kommunikationssystemen wird im Allgemeinen durch die drei Sicherheitsziele Vertraulichkeit (Confidentiality), Integrität (Integrity) und Verfügbarkeit (Availability) definiert, welche oft durch Authentizität (Authenticity) und Zurechenbarkeit (Accountability) ergänzt werden [SB12]. In Abschnitt 2.2.1.1 bis 2.2.1.4 sollen diese, sowie ihr Einfluss auf den Betrieb von Smart Grids und Smart Metering kurz beschrieben werden.

Von der Task Force der SGMS wurden einige wesentliche Maßnahmen zur Erhöhung der technischen Security genannt [SGMS13]:

- Einsatz einer zentralen Infrastruktur für Verschlüsselung und Schlüsselmanagement,
- Sicherung sämtlicher Kommunikationsverbindungen und Schnittstellen,
- Verkleinerung des Wirkungsbereichs von Kommunikationsverbindungen, sowie
- Sichern von Servern (Clustern, Update und Change Management, Firewalls) und Anwendungen.

2.2.1.1 Vertraulichkeit

Zur Vertraulichkeit zählen die Datenvertraulichkeit und der Datenschutz (Privacy). Die Datenvertraulichkeit beschreibt dabei den Schutz vertraulicher oder privater Informationen davor, dass unberechtigte diese erlangen. Die Privacy soll garantieren, das Individuen kontrollieren und beeinflussen können welche Informationen über sie gesammelt und gespeichert werden, sowie von wem und gegenüber wem diese Informationen offen gelegt werden dürfen [SB12] (siehe Abschnitt 2.2.2). Typische Angriffe auf die Vertraulichkeit von Daten sind das Abhören und der unerlaubte Zugriff. Unter Abhören versteht man im Allgemeinen das Erlangen von Informationen ohne aktive Einflussnahme. Dazu zählt das Mitlesen von Kommunikation, Traffic-Analyse oder die Wiederherstellung von absichtlich gelöschten Daten. Beim unerlaubten Zugriff auf Informationen wird meist das System, für welches die Daten bestimmt sind, in unerlaubter Weise für den Bruch der Vertraulichkeit verwendet.

Tritt in einem Smart Grid eine Verletzung der Vertraulichkeit auf, so sind in erster Linie der Datenschutz und die Privacy der Kunden betroffen [CLE08] (Abschnitt 2.2.2). Die Datenvertraulichkeit ist auch durch organisierte Kriminalität, welche einen finanziellen Vorteil erlangen will gefährdet. Diese kann durch den Versuch an Informationen zu gelangen (z. B. für Insider Handel), durch Industriespionage, oder den Versuch den Markt zu beeinflussen eine Gefahr für die Geheimhaltung der Daten darstellen [EKS11]. Schutz vor dem Bruch der Vertraulichkeit von Informationen bieten vor allem die Sicherheitsdienste Verschlüsselung (Abhören), sowie Zugriffskontrolle und Authentifizierung (unerlaubter Zugriff).

2.2.1.2 Integrität und Authentizität

Integrität garantiert den Schutz von Informationen und Programmen vor unautorisierten oder unspezifizierten Änderungen (Datenintegrität). Es umfasst somit auch die Authentizität der Informationen sowie ihre Nichtabstreitbarkeit. Weiters umfasst es die Systemintegrität, welche den Betrieb des Systems in unbeeinträchtigter Weise, ohne unautorisierte Manipulation (egal ob absichtlich oder unabsichtlich) beschreibt [BS12]. Angriffe auf die Integrität können vielfältig sein: unerlaubter Zugriff, Veränderung von Kommunikation, Mehrfachsendung von Kommunikationspaketen oder Manipulation des Systems. Authentizität beschreibt das Sicherheitskonzept, welches sich um die Echtheit eines Nutzers, seiner Daten und seiner Kommunikation kümmert. Sie erfordert, dass der Identität des Nutzers vertraut werden kann und diese verifiziert ist, sowie dass die am System ankommenden Eingaben von diesem Nutzer gemacht wurden. Authentizität wird in vielen Fällen als Integrität und Aktualität beschrieben [SB12]. Das heißt, es wird nicht nur die Integrität der Daten sichergestellt, sondern auch, dass diese für den aktuellen Zeitpunkt bestimmt sind. Ein typischer Angriff gegen dieses Sicherheitsziel sind Replay-Attacken bzw. allgemeine Angriffe auf die Integrität. Wird die Integrität und Authentizität in Smart Metering Infrastrukturen nicht ausreichend geschützt, so kann es zu vielfältigem Missbrauch kommen [CLE08].

Bei der Manipulation von Verbrauchsdaten kann dem Kunden oder dem Energiekonzern finanzieller Schaden zugefügt werden (zu hohe Rechnung, bzw. Stromdiebstahl). Werden die Verbrauchsdaten auch für die Netzregelung verwendet, so können auch durch ihre Manipulation die Regelsysteme gestört werden und Netzinstabilitäten erzeugt werden.

Die Manipulation von Steuerdaten: Durch unautorisierte Kommandos könnte eine Vielzahl von Verbrauchern und Erzeugern missbräuchlich aktiviert, oder deaktiviert werden. Dies kann Auswirkungen auf die Netzstabilität (Schwankungen der Spannung und Frequenz, bis hin zum Blackout) haben, finanzielle (Senden von falschen Abrechnungsprofilen, Nutzung teurer Energiequellen), sowie physische Schäden (z. B. Verschleiß durch ungeeignete Schaltsequenzen, oder Deaktivierung von Kühlschränken und damit das Verderben von Lebensmitteln) verursachen.

Die Sicherheit der Integrität eines Systems und der Daten kann durch Dienste wie digitale Signaturen und Zugriffskontrollen erhöht werden. Wichtig ist hierbei immer auch der Einsatz von physikalischen Schutzmaßnahmen [CLE08] gegen Manipulation. Das Sicherheitsziel Authentizität kann neben den Sicherheitsmaßnahmen für Integrität, durch Mechanismen wie Sequenzzähler und Zeitstempel verbessert werden.

2.2.1.3 Verfügbarkeit

Mit Verfügbarkeit wird in der Sicherheitsforschung das Konzept beschrieben, dass ein System zeitnah arbeitet, sowie dem Gebrauch durch autorisierte Nutzer zur Verfügung steht. Eine weitere Sicht auf Verfügbarkeit ist, dass zeitnaher und zuverlässiger Zugriff auf und Nutzung von Daten möglich ist [BS12]. Will ein Angreifer die Verfügbarkeit eines Systems stören, versucht er in vielen Fällen eine Überlastung des Systems mit unwichtigen Anfragen und Daten zu erreichen. Man spricht dann von einer Denial-of-Service-Attacke. Die Verfügbarkeit kann aber auch durch direktere Angriffe, etwa eine Manipulation oder Außerbetriebnahme des Systems erfolgen. Attacken gegen die Verfügbarkeit können z. B. von Kunden verursacht werden, um eine korrekte Abrechnung zu verhindern [CLE08], denkbar sind aber auch ideologisch motivierte Angriffe von Cyberterroristen [EKS11], welche finanziellen Schaden anrichten oder die Infrastruktur selbst schädigen möchten.

Das Sicherheitsziel Verfügbarkeit kann durch einen Dienst zur Zugriffskontrolle verbessert werden. Es muss jedoch vor allem durch geeignete physikalische Schutzmaßnahmen und eine entsprechende Architektur des Systems z. B. des Kommunikationsnetzwerks (um eine Überlastung der Übertragungskapazität zu verhindern) geschützt werden. In vielen Fällen ist es nur möglich auf die Nichtverfügbarkeit des Systems zu reagieren, deshalb ist die Entdeckung und Bewertung des Ausfallgrundes wichtig [CLE08]. Hierzu werden typischerweise Systeme zur automatischen Diagnose sowie zur Erkennung von physikalischen und informationstechnischen Angriffen eingesetzt.

2.2.1.4 Zurechenbarkeit

Das Sicherheitsziel der Zurechenbarkeit erfordert, dass Aktionen und Eingaben eines Nutzers eindeutig auf diesen zurückgeführt werden können. Hierzu zählen Konzepte wie Nichtabstreitbarkeit, Einbruchserkennung und -vermeidung. Mit Zurechenbarkeit wird der Tatsache Rechnung getragen, dass kein System ganz sicher ist und für den Fall eines Bruchs der Sicherheit die Quelle des Angriffs (der Angreifer und die Sicherheitslücke) gefunden werden soll [SB12]. Ein Bruch der Zurechenbarkeit kann durch Angreifer in verschiedener Weise erfolgen. So ist durch Identitätsdiebstahl eine die Zurechenbarkeit verletzt. Weiters kann nach einem unerlaubten Zugriff auf ein System, durch eine unerlaubte Manipulation von Protokolldateien der Angriff verschleiert werden.

Da die Zurechenbarkeit vor allem für den Fall eines Angriffs bzw. eines Fehlverhaltens nützlich ist, sind keine eigenständigen Attacken darauf zu erwarten. Viel mehr werden bereits beschriebene Angriffe auf Verbrauchs- und Steuerdaten des Smart Metering System, auch darauf abzielen die Zurechenbarkeit zu erschweren bzw. zu verhindern. Sicherheitsdienste, mit denen die Zurechenbarkeit verbessert werden kann, sind deshalb vor allem Authentifizierungsdienste und Zugriffskontrollen. Es muss jedoch überhaupt ein Mechanismus für die Zurechenbarkeit bestehen, welcher Aufzeichnungen über die Aktivitäten des Systems führt (Protokollierung) [SB12]. Beispiele von Zugriffen auf und Änderungen von Daten, deren Protokollierung notwendig ist, sind die Verbrauchs- und Tarifdaten (Preisinformationen, Uhrzeit), sowie Kommandos zur Lastregelung und Netztrennung. Weiters sind die Verwendung von Zeitstempeln und die Synchronisierung der entsprechenden Uhren unumgänglich [CLE08].

2.2.2 Privacy

Der Datenschutz gilt oft als ein Teilgebiet des Sicherheitskonzepts Vertraulichkeit [SB12]. Der Begriff wird jedoch im Allgemeinen zur Beschreibung des rechtlich notwendigen Informationsschutzes verwendet. Privacy ist ein Konzept, welches das Recht des Einzelnen an der Kontrolle persönlicher Informationen beschreibt, im speziellen, welche Daten über ihn gesammelt, verwendet und offengelegt werden [CAV09]. Zu diesen persönlichen Informationen (Personally Identifiable Information, PII) werden alle Daten gezählt, welche die Identifizierung einer Person zulassen (z. B. Adressen, Telefonnummern, Fotos). Zudem werden auch alle Daten, welche der Identität eines Individuums zugeordnet werden können, als persönlich angesehen [CAV09]. Die Privacy ist somit ein umfassenderer Schutz der persönlichen Daten als der vom gesetzlichen Datenschutz geforderte.

In Smart Grids sollen neben der Vertraulichkeit der Daten, diese auch vor unerwünschter Übertragung, Verarbeitung und Speicherung geschützt werden. Denn durch die steigende Dichte von Verbrauchsdaten und ihr beinahe unmittelbares Vorliegen könnten daraus detaillierte Profile zum Stromverbrauch in einzelnen Haushalten gewonnen werden. Es besteht dabei das Risiko, dass aus deren Analyse Persönlichkeitsprofile erstellt werden, durch die der Tagesablauf der Bewohner beschrieben wird [EKS11]. Eine vom Kunden unerwünschte Offenlegung dieser vom Smart Meter gesammelten Daten gegenüber Dritten wäre eine Störung ihrer Privatsphäre. Zudem würde die informationelle Selbstbestimmung verletzt, denn der Nutzer verliert dabei die Kontrolle darüber, wer diese Informationen über seinen Tagesablauf erhält. Aus Sicht des Datenschutzes sind eine möglichst weitgehende Anonymisierung von Lastprofilen und ähnlichen Daten sowie ihre dezentrale Verarbeitung gewünscht. Es besteht hierbei jedoch ein Konflikt zu den für einen sinnvollen Betrieb des Smart Grid notwendige Datenspeicherung, -verarbeitung und -kommunikation, sodass ein Kompromiss gefunden werden muss. Hierzu sollten verschiedene datenschutzrechtliche Prinzipien angewandt werden, in erster Linie Datensparsamkeit, Zweckbindung, Erforderlichkeit und Transparenz. Zudem sollte den Privacy-Anforderungen bereits während der Planung eine hohe Priorität beigemessen werden und den Nutzern die Möglichkeit zur Bestimmung über die Weitergabe von persönlichen Daten ermöglicht werden [EKS11]. Bei deren Weitergabe an Dritte sollten zudem folgende Punkte zum Schutz der Privacy beachtete werden [CPW10]:

- Nur die minimal zur Erbringung des Dienstes nötigen Informationen weitergeben.
- Verwendung von Pseudonymen, anstatt persönlich zuordenbarer Informationen, wo möglich.
- Kunden sollen bestimmen, welche persönlichen Daten an Dritte weitergegeben werden.
- Die Übertragung der Daten muss ausreichend gesichert sein.
- Die Empfänger der persönlichen Daten müssen einwilligen diese nicht ohne Zustimmung Daten aus anderen Quellen zuzuordnen.

2.3 Regelungen und Sicherheitskonzepte

Smart Grids und die darauf aufbauenden Technologien sind auf eine möglichst weite Verbreitung ausgelegt. Für einen weitgehend reibungslosen Einsatz sind Regelungen und Standards notwendig. Die in Abschnitt 2.2 und 1.2 dargelegten Erfordernisse für einen sicheren Betrieb, sowie Sicherheits- und Datenschutzbedenken haben dazu geführt, dass von verschiedenen Stellen Vorschriften, Sicherheitskonzepte und gesetzliche Bestimmungen erarbeitet wurden.

2.3.1 Richtlinien und gesetzliche Regelungen

Die Europäische Union hat verschiedene Richtlinien mit Bezug zu Smart Grids und Smart Metering erlassen. Artikel 13 der *Richtlinie 2006/32/EG über Endenergieeffizienz und Energiedienstleistungen (...)* [EDL2006] fordert, dass individuelle Zähler welche den tatsächlichen Energieverbrauch und die tatsächliche Nutzungszeit widerspiegeln installiert werden (falls dies technisch und wirtschaftlich sinnvoll möglich ist). Weiters wird eine häufige, auf dem effektiven Verbrauch beruhende Abrechnung gefordert. Dies soll den Kunden die Möglichkeit geben, den Verbrauch zu steuern. Dabei werden neben der Elektrizität auch die Bereiche Erdgas, Fernheizung und -kühlung sowie Warmbrauchwasser eingeschlossen. In dieser Richtlinie besteht noch kein direkter Bezug zu intelligenten Messgeräten, dieser wird erst in der späteren *Richtlinie 2009/72/EG* über gemeinsame Vorschriften für den Elektrizitätsbinnenmarkt (…) [EC09] hergestellt. So werden in Artikel 3 (11) die Mitgliedstaaten aufgefordert, „dass die Elektrizitätsunternehmen den Stromverbrauch optimieren, indem sie beispielsweise Energiemanagementdienstleistungen anbieten, neuartige Preismodelle entwickeln oder gegebenenfalls intelligente Messsysteme oder intelligente Netze einführen“. In der Empfehlung der Kommission zu Vorbereitungen für die Einführung intelligenter Messsysteme [EDK12] werden auch Datenschutz- und Datensicherheitserwägungen, sowie gemeinsame Mindestanforderungen an intelligente Messsysteme im Stromsektor definiert. Als Mindestanforderungen an das System, sowie seiner Sicherheit (und Datenschutz) wurden folgende Punkte definiert:

- direkte Bereitstellung der Messwerte für den Verbraucher sowie für von ihm benannte Dritte
- ausreichende Aktualität der Messwerte
- Fernablesung der Zähler, mit ausreichender Häufigkeit (auch für die Netzplanung)
- bidirektionale Kommunikation zu den intelligenten Messsystemen
- Möglichkeit die Versorgung aus der Ferne ein- und auszuschalten, sowie Lastflüsse oder Strombegrenzungen zu steuern
- Einsatz sicherer Datenkommunikation im ganzen Smart-Metering-System

- Funktionen zur Betrugsaufdeckung und -verhinderung
- Bereitstellung von Import- und Exportmessungen

Es wird zudem empfohlen konzeptionsbedingten Datenschutz (data protection by design) und Datenschutz aufgrund von standardmäßigen Voreinstellungen (data protection by default) zu forcieren. Es wird weiters die Entwicklung einer Datenschutzfolgenabschätzung durch die Kommission angekündigt, deren Annahme und Anwendung angeraten wird. Im Sinne des Datenschutzes soll zudem auf Anonymisierung der Daten, Datensparsamkeit, Zweckbindung und Erforderlichkeit gesetzt werden. Durch konzeptionsbedingten Datenschutz und die Verwendung von verschlüsselten Kanälen soll die Datensicherheit verbessert werden. Auch wird auf die notwendige Einhaltung bestehender Richtlinien und Normen bzgl. Datenschutz und –sicherheit hingewiesen.

In Österreich wurde die Einführung von Smart Metering durch die *Intelligente Messgeräte-Einführungsverordnung (IME-VO)* [BGB12] für Netzbetreiber verpflichtend. Die Smart Meter müssen dabei der von der *E-Control* verfassten Verordnung *Intelligente Messgeräte-AnforderungsVO* (IMA-VO) [BGB11] entsprechen. In dieser werden folgende Punkte vorgeschrieben:

- Nutzung einer bidirektionalen Kommunikationsverbindung zum Netzbetreiber für Datenaustausch, Uhrensynchronisation und Softwareupdates
- Messung und Speicherung von Zählerständen, Leistungsmittelwerten und Energieverbrauchswerten in einem 15-Minuten-Intervall, für mindestens 60 Tage
- Übermittlung der Daten des Vortags bis spätestens 12 Uhr des nächsten Tages
- Bereitstellung der Daten für den Endverbraucher über eine offene Schnittstelle
- Bereitstellung von Gatewayfunktionalität für mindestens vier weitere Messgeräte
- Sicherung aller Kommunikationsschnittstellen nach dem Stand der Technik, mittels Verschlüsselung und Autorisierung sowie kundenspezifischen Schlüsseln
- Implementierung von Zugriffs-, Status- und Fehlerprotokollen und einer Manipulationserkennung sowie Kalender und Uhr
- Funktionen zur Fernabschaltung und -aktivierung der Anlage sowie zur Begrenzung der maximal bezogenen Leistung
- Einhaltung aller Richtlinien bzgl. Datenschutz, Maß- und Eichgesetz

Fundamentale Mindestanforderungen an den Datenschutz werden in Österreich durch das *Bundesgesetz über den Schutz personenbezogener Daten 2000* (Datenschutzgesetz 2000) geregelt. Die Analyse zu den rechtlichen Voraussetzungen im Rahmen der SGMS für den Einsatz von Smart Metering [SGMS13] ergab, dass es sich bei den Energieverbrauchsprofilen um keine sensiblen Daten handelt. Jedoch werden explizite Zustimmungen der Nutzer zur Verwendung der personenbezogenen Daten benötigt und es ist eine Meldung bei dem Datenverarbeitungsregister vorgeschrieben. Sollte das Smart Metering in den Regelbetrieb übergehen, so könnten die Zustimmungen zur Datenverarbeitung in die Verträge für die elektrischen Stromanschlüsse aufgenommen werden. Sollte durch eine Aufnahme des Smart Metering als Standardanwendung in die *Standard- und Muster-Verordnung 2004* die Meldung beim Datenverarbeitungsregister entfallen, so sind die entsprechend der Standardanwendung definierten Dateninhalte, Aufbewahrungsfristen und sonstigen Regelungen einzuhalten.

2.3.2 Sicherheitskonzepte

Bereits in Abschnitt 1.4 wurden verschiedene Sicherheitskonzepte für Smart Metering angesprochen, welche im Folgenden kurz betrachtet werden sollen.

Vom Forum Netztechnik/Netzbetrieb im VDE wurde das *EDL40* Lastenheft erstellt [VDE10]. Darin wird ein Zähler spezifiziert, welcher dem Energiewirtschaftsgesetz Deutschlands entspricht, und zusammen mit einem MUC-Controller das *EDL40*-Smart-Metering-System bildet. Die Sicherheitsvorkehrungen, die dabei vorgesehen sind, sehen digitale Signaturen mit elliptischen Kurven und 192 Bit Schlüssellänge (*ECC-192*) für die Telegramme mit den Zählerstanden vor. Weiters ist eine AES-CBC-Verschlüsselung (Advanced Encryption Standard, Chain-Block-Cipher) mit 128 Bit für drahtlose Verbindungen (Meter-Bus / M-Bus) vorgeschrieben. Für drahtgebundene M-Bus-Verbindungen wird der physikalische Zugriffsschutz (Schnittstelle auf Rückseite des Geräts) als ausreichende Sicherung betrachtet. Die uneingeschränkt zugängliche Kundenschnittstelle verfügt über keine Sicherung. Dies wird damit begründet, dass darüber nur unidirektional reine Zählerstände ausgegeben werden und diese Informationen auch bisher durch einfaches Ablesen erfahrbar waren. Schlüsselvereinbarungen und die Sicherung der WAN-Schnittstelle werden vorgeschrieben, aber nicht definiert.

Das *Synchronous Modular Meter (SyM²)* Pflichtenheft wurde ebenfalls vom Forum Netztechnik/Netzbetrieb herausgegeben [VDE09]. Es beschreibt Anforderungen an taktsynchrone Lastgangzähler, welche aus einem Basisgerät, einem Messmodul (IW Modul) und einem Kommunikationsmodul (KM Modul) besteht. Bei der Kommunikation zwischen diesen Modulen wird wiederum auf physischen Zugriffsschutz vertraut und es ist keine Verschlüsselung vorgesehen. Die Daten mit den Messwerten sollen wiederum mittels digitaler Signaturen geschützt werden. Zusätzlich werden Firmwareupdates spezifiziert, welche signiert sein müssen. Dabei wird zwischen einfachen Updates des Kommunikationsmoduls und Updates der eichrechtlichen Funktionen unterschieden. Letztere müssen mit der Signatur der Eichbehörde authentifiziert werden.

Das *Projekt Sicherer Elektronischer Messdatenaustausch* (*SELMA*) [SCHAUB03] setzt auf drei Grundkonzepte zur Sicherung des Smart Metering Systems: authentifizierte Messdaten, gesicherte Kanäle und zertifizierte Gerätekomponenten. Für die Authentifizierung der Messdaten sind digitale Signaturen vorgesehen. Dabei wird ein eichrechtlich relevanter, sowie ein nicht relevanter Bereich definiert. Auch die gesicherten Kanäle sollen die Sicherheitsdienste Authentifizierung und Datenintegrität durch digitale Signaturen erhalten. *SELMA* definiert weiters ein Schlüsselkonzept zur Nutzung mit den Signierdiensten. Zur Erhöhung der Sicherheit ist die Auslagerung kryptografischer Funktionen in ein dediziertes Sicherheitsmodul vorgesehen.

Im Rahmen des Smart Metering und Datenschutz Projekts [REN10] der Österreichische Energieagentur und der *e-commerce monitoring GmbH* wurde kein Sicherheitskonzept, aber unter anderem Empfehlungen für die Einführung intelligenter Messgeräte in Österreich erarbeitet. Diese umfassen unter anderem den offenen Dialog, Akzeptanz durch Wahlfreiheit, Beachtung der informationellen Selbstbestimmung, Informationen für die Kunden, die Möglichkeit für zielgerichtetes Feedback, sowie Privacy by Design.

2.3.3 BSI Schutzprofile

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) hat im Auftrag des Bundesministeriums für Wirtschaft und Energie der Bundesrepublik Deutschland Vorgaben mit verbindlichen Datenschutz- und Datensicherheitsstandards für Smart Metering entwickelt [EKS11]. Dabei wurden unter anderem zwei Schutzprofile erarbeitet, sowie mehrere technische Richtlinien veröffentlicht:

- Schutzprofil für die Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen (Smart Meter Gateway PP) [BSI13-1]
- Schutzprofil für das Sicherheitsmodul der Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen (Security Module PP) [BSI13-2]
- Technische Richtlinie BSI TR-03109-1, Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems
- Technische Richtlinie BSI TR-03109-2, Anforderungen an die Funktionalität und Interoperabilität des Sicherheitsmoduls [BSI13-3]
- Technische Richtlinie TR-03109-3, Kryptographische Vorgaben für die Infrastruktur von intelligenten Messsystemen (referenziert nur BSI TR-03116 Technische Richtlinie für eCard-Projekte der Bundesregierung [BSI12-1])
- Technische Richtlinie BSI TR-03109-4, Smart Metering PKI - Public Key Infrastruktur für Smart Meter Gateways [BSI13-5]

Das Schutzprofil für das Smart Meter Gateway definiert dieses als Gerät, welches für die Sammlung und Verarbeitung der Messdaten, für die Bereitstellung von Kommunikationsdiensten für Geräte des lokalen Messnetzwerks, für den Schutz der Geräte im LAN (Local Area Network) und die Bereitstellung von Sicherheitsdiensten verantwortlich ist. Zum Betrieb des Smart Metering Systems werden die Komponenten in verschiedene Netzwerke segmentiert, welche nur über das Gateway miteinander kommunizieren können. Es handelt sich dabei um das Wide Area Network (WAN), welches die Verbindung zur externen Welt darstellt, das Local Meterological Network (LMN), in dem sich die Messgeräte befinden, sowie das Home Area Network (HAN), welches die Schnittstelle zum Endverbraucher und zu steuerbaren Geräten darstellt. HAN und LMN bilden zusammen das Local Area Network (LAN).Die Struktur des Netzwerks ist in Abbildung 1 dargestellt.

Kommunikation mit dem WAN sind nur mit autorisierten Nutzern vorgesehen und sollte nur vom Gateway aus aufgebaut werden können. Zur Initialisierung eines Datenaustausches durch den Infrastrukturbetreiber im WAN ist eine Anklopffunktion vorgesehen. Die Messgeräte im LMN sollen die aufgezeichneten Daten an das Gateway übermitteln. Im Falle von drahtlosen Verbindungen ist eine Signierung der Messdaten bereits an diesem Punkt vorgesehen, eine Verschlüsselung soll hingegen immer möglich sein. Im HAN befinden sich die Schnittstelle zum Endverbraucher sowie die durch das Smart Grid steuerbaren Systeme. Dabei kann es sich um Verbraucher, Erzeuger oder Speicher handeln, sie werden als Controllable Local Systems (CLS) bezeichnet. Das Gateway übernimmt die Aufgabe einer Smart Metering Firewall, sowie eines Datenkonzentrators. Für die Ausführung von sicherheitsrelevanten Funktionen wie digitalen Signaturen, Schlüsselmanagement und Authentisierungen soll das Gateway von einem Sicherheitsmodul unterstützt werden. Die im Security Module PP beschriebenen Anforderungen an das vorgesehene Sicherheitsmodul werden in Abschnitt 3.1 beschrieben.

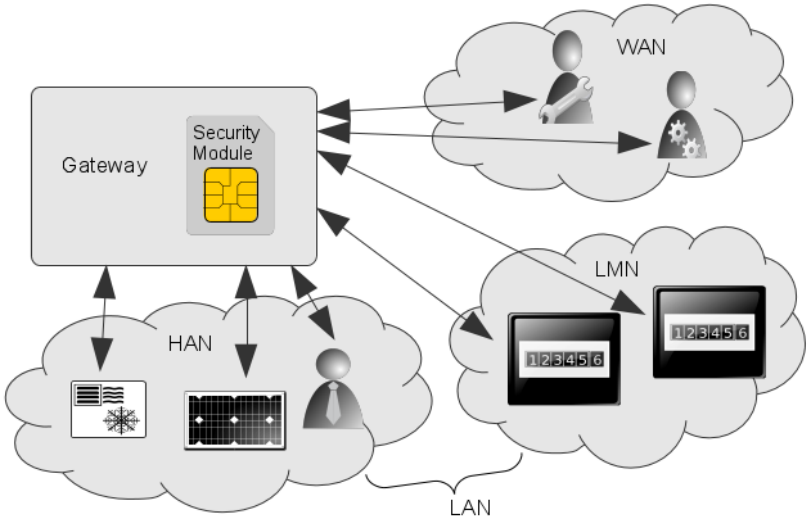


Abbildung 1: Netzwerkstruktur nach dem BSI Schutzprofil. WAN: Wide AreaNetwork, LMN: Local Meterological Network, HAN: Home Area Network, LAN: Local Area Network.

Im Schutzprofil werden folgende Sicherheitsrisiken für das SMGW genannt: Datenmodifikation im WAN oder lokal, Manipulation der Uhr bzw. des Kalenders des Gateways, Vertraulichkeitsverletzungen im WAN, lokal oder im Gateway, Manipulation von Daten im Gateway, Übernahme der Kontrolle über das System, unerlaubte Manipulation von Daten im Gateway. Weiters werden die Risiken der Privacy-Verletzung durch zu umfangreiche Datenhaltung, sowie des Zugriffs auf temporäre Daten im Gateway beschrieben. Diesen Bedrohungen soll durch verschiedene Maßnahmen entgegengewirkt werden, welche in zwei Kategorien unterteilt werden:

- Sicherheitsziele im Gateway: Firewall-Funktionalität, getrennte Schnittstellen für die verschiedenen Netzwerke, Verschleierung von Kommunikationshäufigkeit, sichere Kommunikation mit den Messgeräten und Bereitstellung der Messdaten, kryptografischer Schutz des Systems, Verwendung von Zeitstempeln, Funktionen zum Schutz gegen sicherheitskritische Fehlfunktionen und Manipulationen, Schutz der Managementfunktionen, sowie Schutz der Endverbraucherschnittstelle, Führung von Protokollen zum Systemstatus, zu Datenverbindungen aus dem WAN sowie zu Kalibrierungen und deren Schutz.
- Sicherheitsziele für das Betriebsumfeld: vertrauenswürdige und kompetente Administrierung, Vertrauenswürdigkeit externer Datenempfänger und der Zugriffsregelungen, physikalischer Schutz des Gateways, der Messgeräte und ihrer Kommunikation, Einsatz eines zertifizierten Sicherheitsmoduls und Verwendung dessen Funktionen, Zertifizierung von Firmwareupdates, Bereitstellung der WAN-Verbindung, von Zeitquellen im WAN, Sicherstellung, dass Messgeräte nur mit dem Gateway kommunizieren, Schutz von weiteren Kommunikationsverbindungen des HAN (falls vorhanden).

Die sichere Kommunikation mit den Messgeräten und Bereitstellung der Messdaten umfasst die regelmäßige Abfrage der Messdaten in definierten Intervallen, Verifizierung der Messdaten, Verschlüsselung der Kommunikation, Verarbeitung der Daten nur nach definierten Regeln. Weiters werden die Messdaten für den Endempfänger verschlüsselt und signiert, sowie zwischengespeichert

falls dieser nicht erreichbar ist. Falls die Herkunft der Messdaten für deren Verwendung nicht notwendig ist, sollen diese entsprechend anonymisiert werden.

Der kryptografische Schutz des Systems soll in Form von Authentifizierung, Integritätsschutz und Verschlüsselung der Kommunikation mit dem WAN, dem LMN und dem Endverbraucher erfolgen. Weiters sollen Replay-Attacken erkannt werden und die persistent im Gateway gespeicherten Informationen geschützt werden.

2.4 Frameworks für Smart Metering Gateways

Die unvermeidliche Einführung von Smart Metering hat zur Entwicklung verschiedener Software Frameworks für den Einsatz in diesen Systemen geführt. Zwei dieser Frameworks, welche unter einer Open-Source-Lizenz verfügbar sind, sollen im Folgenden betrachtet werden.

2.4.1 *OpenMUC*

OpenMUC ist ein Software-Framework, welches vom Frauenhofer-Institut für Solare Energiesysteme entwickelt wurde [FZBW09]. Es soll als Basis für eine offene Smart Metering Referenzplattform dienen. Es ist dabei als Software für einen Multi Utility Communication-Controller (MUC-Controller) ausgelegt, und soll die Messwerte aus verschiedenen Energie- und Mengenzählern sammeln und für die Infrastrukturbetreiber zur Verfügung stellen. Weiters soll der MUC-Controller die Möglichkeit bieten, die gemessenen Daten dem Endverbraucher zur Verfügung zu stellen. Die Ziele hinter der Entwicklung von *OpenMUC* sind [FZBW09]:

- Offene Testplattform: Implementierung neuer Standards und Technologien, Erkennung von eventuellen Problemen der Spezifikationen
- Modularität: leicht erweiterbar, etwa durch neue Kommunikationstechnologien.
- Smart Grid Integration: Erweiterbarkeit um Smart-Grid-Funktionalitäten, z. B. Demand Side Management
- Hardwareunabhängigkeit: Basierend auf Java und entwickelt für GNU/Linux, dadurch relativ hardwareunabhängig

Damit erfüllt *OpenMUC* die Anforderungen für die Softwarebasis eines Smart Metering Gateways.

Als Basis für die *OpenMUC* Software wird das OSGi-Framework Knopflerfish eingesetzt. Dadurch steht eine Softwareplattform für den modularen Einsatz von Anwendungen und Diensten als Komponenten zur Verfügung. *OSGi* erlaubt es sogenannte *Bundles* während der Laufzeit zu installieren, aktivieren, deaktivieren und auch zu deinstallieren. Der Funktionsumfang eines *Bundles* kann dabei von einer einfachen Erweiterung der Bedienoberfläche, über neue Kommunikationsprotokolle, bis hin zu kompletten Applikationen umfassen, welche jeweils Dienste für andere *Bundles* bereitstellen können. In *OpenMUC* wird diese *Bundle*-Funktionalität umfassend genutzt. So sind die Funktionen für das Auslesen von Messdaten in einem *Bundle* realisiert. Dieses greift auf die Dienste der *Bundles* mit den Kommunikationsprotokollen zurück, um mit den Zählern Daten auszutauschen. In *OpenMUC* können somit Daten über verschiedenste Kommunikationsprotokolle (z. B. M-Bus, Ethernet, RS232) in verschiedenen Datenformaten (Smart Message Language u. a.) ausgetauscht werden. Für

das Speichern von Zähler- und Tarifdaten verwendet *OpenMUC* eine *SQLite* Datenbank, wodurch der Zugriff auf die gemeinsam von mehreren *Bundles* genutzten Daten vereinfacht werden soll.

2.4.2 Open Gateway Energy Management - *OGEMA*

Vom Frauenhofer Institut für Windenergie und Energiesystemtechnik wurde das Open Gateway Energy Management (*OGEMA*) Framework entwickelt. Es handelt sich dabei um eine offene Softwareplattform, deren zentrale Aufgaben im Energiemanagement liegen. Hauptfunktion ist die Bereitstellung eines einzelnen Gateways für die Kommunikation zwischen Endverbraucher und Versorgungsinfrastruktur [OGEMA09]. Das *OGEMA*-Framework soll beim Endverbraucher eine Umgebung für Applikationen im Bereich Energiemanagement und -effizienz mittels Smart Grids bereitstellen. Für eine möglichst reibungslose Zusammenarbeit verschiedener Komponenten und Applikationen wird intensiver Gebrauch von standardisierten Datenmodellen und Diensten gemacht. Dadurch sollen verschiedene Szenarien vereinfacht bzw. automatisiert werden. Der Zugriff auf an das Gateway angeschlossene Geräte und deren Funktionen soll einheitlich über solche standardisierte Datenmodelle und Dienste erfolgen. Ebenso sollen externe Daten für die Applikationen in standardisierten Datenmodellen bereitgestellt werden. Neue Geräte können damit automatischen in die Applikationen und in das Gateway eingebunden werden. Weiters werden vom Framework Dienste und Funktionen zur Nutzung der Datenmodelle und Services sowie für das Benutzerinterface, die persistente Datenspeicherung und die Protokollierung angeboten. *OGEMA* basiert ebenso wie *OpenMUC* auf einem *OSGi*-Framework. Dadurch können ebenso Bundles zur Laufzeit geladen oder entfernt werden. Wiederum können Applikationen, Kommunikationstreiber und Gerätetreiber getrennt entwickelt und eingebunden werden. Die standardisierten Datenmodelle und Dienste dienen dabei als Abstraktionsebene für die Hardware [OGEMA12]. Die Architektur des *OGEMA*-Frameworks wird in Abbildung 2 dargestellt.

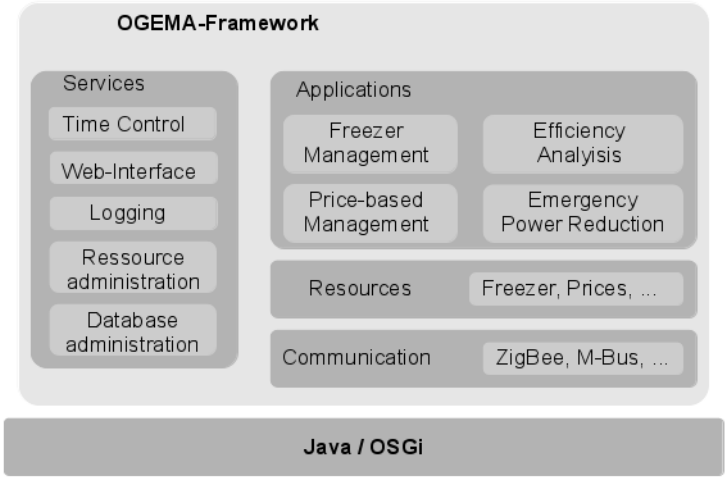


Abbildung 2: Architektur des *OGEMA*-Frameworks

Die im System vorhandenen Komponenten können folgendermaßen kategorisiert werden:

- **Ressourcen** repräsentieren Informationen, welche ihren Ursprung außerhalb des Frameworks haben. Dies umfasst neben Daten aus dem WAN (z. B. Preisprofile) auch die Parameter und die aktuellen Eigenschaften von Geräten (z. B. Temperatur im Kühlschrank). Für die verschiedenen Kategorien von Informationen stehen abstrahierte Modelle, sog. *Resource Type* (bestehend aus Datenmodellen und Services) zur Verfügung und können damit eingebunden werden.
- **Kommunikationssysteme** bilden die Verbindung zwischen den physikalischen Geräten und ihrer Repräsentation als Ressourcen.
- **Applikationen** sind Softwarekomponenten, welche im Framework ausgeführt werden und Aufgaben für einen bestimmten Anwendungsfall übernehmen. Sie verfügen im Allgemeinen über keine direkte Verbindung zu physikalischen Geräten, sondern greifen nur auf die Ressourcen zu.
- Die **Dienste des Frameworks** bilden das sogenannte Application Programming Interface (API). Es werden hierbei Funktionen für die Administration von Ressourcen und Anwendungen, das Management von Ausführungszeit, die persistente Datenspeicherung, für die Benutzeroberfläche (Webinterface) und für Protokollierung zur Verfügung gestellt.

Für die in Entwicklung befindliche Version 2.0 des *OGEMA*-Frameworks ist eine Integration von *OpenMUC* geplant.

3 Sicherheitsmodul

Für die Bereitstellung von sicherheitsrelevanten Diensten bzw. für die Abarbeitung sicherheitsrelevanter Funktionen soll das Smart Metering Gateway durch ein Sicherheitsmodul unterstützt werden. Dabei sollen vielfältige kryptografische Funktionen in das Sicherheitsmodul ausgelagert werden. Gleichzeitig soll es das Gateway um einen sicheren Speicher für kryptografische Schlüssel, Zertifikate und allgemeine Daten erweitern. Die Bezeichnung „sicherer Speicher“ soll hierbei die Möglichkeit einer integritäts-, authentizitäts- und vertraulichkeitsgeschützten Datenspeicherung beschreiben. Durch diese Auslagerung von Funktionalität soll die Sicherheit des Gesamtsystems erhöht werden. Das Sicherheitsmodul soll als dedizierte Hardware mit darauf ausgeführter Software realisiert sein [BSI13-2]. Dabei ist nicht unbedingt ein eigenständiges System auf einer eigenen Platine gemeint, sondern vor allem der Einsatz eines dedizierten Mikroprozessors, Speicher und Peripherie, welche nur über eine Kommunikationsschnittstelle Daten mit dem Gateway austauschen, jedoch keinen gemeinsamen Speicher haben. Auch soll der sichere Betrieb nicht von zusätzlicher nicht zum Sicherheitsmodul gehörender Hard- oder Software beeinflusst werden. Es geht also vor allem um die Abgrenzung zu einer reinen Softwareerweiterung des Smart Metering Gateway. Das Sicherheitsmodul soll auf einer möglichst sicheren Plattform realisiert werden. In den meisten Fällen wird hierfür eine Smartcard eingesetzt werden. Smartcards für Sicherheitsanwendungen verfügen im Gegensatz zu gewöhnlichen Embedded Systems (welche als zweckvolle Implementierungsplattform für Smart Metering Gateways gelten) bereits über einen hohen Grundschutz. So sind im Allgemeinen bereits Schutzfunktionen gegen Hardwaresabotage (Tampering) und Seitenkanalattacken (Side Channel Attack) wie z. B. Rechenzeitangriffe (Timing Attacks) und Energieverbrauchsanalyse (Power Analysis) geschützt. Auch der verwendete Speicher ist im Allgemeinen bereits besonders gegen unerlaubte Manipulations- und Ausleseversuche geschützt [RANKL08]. Smartcards sind jedoch nicht die einzig mögliche Implementierungsplattform für ein Sicherheitsmodul. Viele Mikrocontrollerhersteller bieten auch spezielle Mikrocontroller für den Einsatz in Sicherheitsanwendungen. Diese verfügen meist über ähnliche Sicherheitseigenschaften wie Smartcards und könnten daher auch als Plattform für das Sicherheitsmodul verwendet werden. Smartcards verfügen jedoch über den Vorteil, der Austauschbarkeit. Zum einen gilt dies für die Karte selbst, die von verschiedenen Herstellern für dieselben Einsatzzwecke angeboten wird. Wird eine Plattform wie *Java Card* eingesetzt, kann ähnliche Hardwarefunktionalität vorausgesetzt, sogar die Software großteils weiterverwendet werden. Die Austauschbarkeit gilt aber auch auf der Ebene des Sicherheitsmoduls selbst. Entspricht dieses einer anerkannten Spezifikation wie dem BSI Schutzprofil [BSI13-2], so kann der Smart-Metering-Gateway-Hersteller das Sicherheitsmodul von verschiedenen Anbietern erwerben, ohne das Gateway selbst anpassen zu müssen. Die Austauschbarkeit hat aber auch einen konkreten Sicherheitsvorteil.

Wird der Schutz einer bestimmten eingesetzten Smartcard, oder einer bestimmten Implementierung des Sicherheitsmoduls kompromittiert, so ist der einfache Austausch dieser einzelnen Komponente möglich. Für den Betreiber des Gateways hat dies enorme finanzielle Vorteile, da das Gateway selbst nicht ausgetauscht werden muss und auch der zeitliche Aufwand geringer gehalten werden kann. Dies erhöht auch die Wahrscheinlichkeit eines Austausches der kompromittierten Komponente, im Gegensatz zum Weiterbetrieb eines unsicheren Gateways. Die erhöhte Komplexität, die durch das zusätzliche Sicherheitsmodul und die Kommunikationsschnittstelle zum Gateway entsteht, erhöht auch die Möglichkeiten und Bandbreiten von Angriffen. Werden diese bei der Entwicklung und Integration in das Gateway beachtet, scheinen die Sicherheitsvorteile zu überwiegen.

3.1 Gewünschte Funktionalität des Sicherheitsmoduls

Das Sicherheitsmodul soll als Dienstanbieter diverser kryptografischer Funktionen für das Smart Metering Gateway dienen. Diese Dienste umfassen die Generierung und Verifizierung digitaler Signaturen, Unterstützung für Schlüsselvereinbarung im Rahmen des Aufbaus sicherer Verbindungen (Transport Layer Security – TLS) und für die Verschlüsselung von Informationsdaten, die Generierung asymmetrischer Schlüsselpaare, die Generierung sicherer Zufallszahlen, die Authentifizierung von Komponenten mittels digitaler Signaturen, die Authentifizierung von Komponenten und Schlüsselvereinbarung mittels Passwort sowie die sichere Speicherung von Schlüsselmaterial und spezifischen Daten des Gateways. Weiters sollen diese Dienste über eine gesicherte Verbindung (Secure Messaging) zur Verfügung stehen. Als einzige Speicherstelle für private Schlüssel kann das Sicherheitsmodul als kryptografische Identität des Smart Metering Gateway betrachtet werden. Die im Folgenden beschriebenen Anforderungen an ein Sicherheitsmodul für ein Smart Metering Gateway sind an das BSI *Schutzprofil für das Sicherheitsmodul der Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen* [BSI13-2] angelehnt.

3.1.1 Anforderungen an die Plattform

Das Sicherheitsmodul soll wie bereits erwähnt über eine eigenständige Hardware mit darauf ausgeführter Software verfügen. Diese soll über eine Kommunikationsschnittstelle Daten mit dem Gateway austauschen. Die Ausführung der Schnittstelle soll nicht drahtlos erfolgen und die darüber ausgetauschten Daten sollen dem Standard ISO/IEC 7816 Id*entifikationskarten – Chipkarten mit Kontakten*, im speziellen 7816-4 *Regeln, Sicherheitsfunktionen und Befehle für den Datenaustausch* [ISO7816-4] entsprechen.

An die Implementierungsplattform werden zudem spezielle Sicherheitsanforderungen für den Schutz sicherheitssensitiver Daten gestellt. Als solche gelten alle im Sicherheitsmodul gespeicherten Daten, im Besonderen private Schlüssel, Passwörter für die Authentifizierung sowie die im Rahmen kryptografischer Operationen berechneten Ergebnisdaten sowie Ausgabewerte des Zufallszahlengenerators, aber auch die Zwischenergebnisse dieser Operationen. Dabei sollen diese Daten gegen verschiedene Arten der Angriffe auf ihre Vertraulichkeit und Integrität geschützt werden:

- Schutz gegen Seitenkanalangriffe: Durch die Messung elektromagnetischer Abstrahlung, Energieverbrauch oder Abarbeitungszeit von Operationen des Moduls dürfen keine Rückschlüsse auf sensitive Daten und Informationen über sie getroffen werden können.
- Schutz gegen Hardwaresabotage (Tampering): Die sicherheitssensitiven Daten müssen auch gegen Angriffe und Sabotageversuche auf die Hardware geschützt sein. Hierzu zählen unter anderem die Anbringung von elektrischen Kontakten auf dem freigelegten Chip oder die gezielte Verursachung von Rechen-, oder Abarbeitungsfehlern. Dies kann z. B. durch den Betrieb mit Spannungen und Taktfrequenzen welche außerhalb der Spezifikation liegen, durch Bestrahlung oder durch das Anlegen von elektromagnetischen Feldern versucht werden [KK99].
- Zugriffsschutz auf Hardwaretestfunktionen und -debugschnittstellen: Auch durch die physikalische Freilegung von internen Testpunkten oder die Freischaltung von Debuggingfunktionen dürfen keine Informationen über sicherheitssensitive Daten erlangt werden.
- Schutz gegen Fehlfunktionen: Der Betrieb soll nur unter stabilen Betriebsbedingungen erfolgen. Bei auftretenden Störungen soll immer ein sicherer Betriebszustand eingenommen werden und eventuell Funktionalität deaktiviert werden. Hierzu zählt auch der Ausfall der Stromversorgung.

3.1.2 Authentifizierung

Für den Betrieb des Sicherheitsmoduls ist ein Dienst zur Authentifizierung des Gateways, sowie des Gateway Betreibers am Sicherheitsmodul, sowie zur Authentifizierung des Sicherheitsmoduls am Gateway erforderlich. Weiters soll die Möglichkeit bestehen, dass sich das Gateway mithilfe des Sicherheitsmoduls an externen Stellen authentifizieren kann. Dies soll durch den Einsatz von Public Key Kryptografie mittels elliptischer Kurven, bzw. mit einem PAKE-Protokoll (Password Authenticated Key Exchange) erfolgen.

3.1.3 Zugriffskontrolle

Für den Zugriff auf das Sicherheitsmodul, die zur Verfügung stehenden Dienste sowie die gespeicherten Daten ist eine Zugriffskontrolle nötig. Es soll dabei die Möglichkeit bestehen den Zugriff immer zu erlauben, immer zu verbieten, oder abhängig von den aktuellen Authentifizierungsstatus zu ermöglichen. Weiters sollen Dateizugriff und -nutzung abhängig von der auszuführenden Funktion einschränkbar sein.

3.1.4 Geschütztes Dateisystem

Für die Speicherung von Binärdaten und kryptografischen Informationen soll ein geschütztes Dateisystem zur Verfügung gestellt werden. Dieses soll dem ISO 7816-4 Standard entsprechen, mit Erweiterungen für die Speicherung von Schlüsselinformationen und Passwortdateien. Das Dateisystem soll für die Integrität der Daten sorgen, sowie deren Vertraulichkeit schützen. D. h. der Zugriff auf die Daten soll nur über die im Rahmen der Zugriffskontrolle (siehe 3.1.3) spezifizierten und einschränkbaren Funktionen möglich sein.

3.1.5 Dienst zur Erzeugung und Verifizierung von digitalen Signaturen

Für die Signierung von Inhaltsdaten, den Aufbau von TLS-Verbindungen, sowie die Verifizierung von Zertifikaten und Zertifikatsketten soll das Gateway die kryptografischen Funktionen des Sicherheitsmoduls nutzen. Hierzu soll das Sicherheitsmodul dem Gateway einen Dienst zur Erzeugung und Verifizierung von digitalen Signaturen auf Basis von Public Key und Elliptic-Curve-Kryptografie zur Verfügung stellen. Dabei soll das ECDSA (Elliptic Curve Digital Signature Algorithm) verwendet werden. Die Signaturfunktionen, welche im Rahmen der Authentifizierung genutzt werden, sind nicht Teil dieses Dienstes.

3.1.6 Dienst zur Schlüsselvereinbarung

Für Schlüsselvereinbarungen im Rahmen der Inhaltsdatenverschlüsselung und dem Aufbau von TLS-Verbindungen soll dem Gateway ein entsprechender Dienst angeboten werden. Dieser soll wiederum auf Elliptische-Kurven-Kryptografie aufbauen. Dabei sollen die Verfahren Diffie-Hellman-Schlüsseltausch, bzw. El-Gamal-Verschlüsselungsverfahren zur Erzeugung des gemeinsamen Geheimnisses genutzt werden. Dieses soll an das Gateway übermittelt werden, wo es zur Schlüsselableitung verwendet wird.

3.1.7 Schlüsselmanagement

Für die im Rahmen der angebotenen kryptografischen Dienste genutzten Schlüssel soll ein sicheres Management angeboten werden. Dieses soll neben der einfachen Speicherung von Schlüsseln, welche bereits bei der Initialisierung integriert wurden, die Generierung von Schlüsselpaaren, den Export von generierten öffentlichen Schlüsseln, den Import von öffentlichen Schlüsseln auch das Löschen der Schlüsseldaten umfassen. Die für die Signaturerzeugung verwendeten privaten Schlüssel sollen dabei exklusiv im Sicherheitsmodul erzeugt werden und nur dort vorgehalten werden. Für die Signaturverifizierung sollen intern gespeicherte öffentliche Schlüssel verwendet werden. Hierzu soll auch eine Funktion zum temporären Import von Public Keys vorhanden sein. Gleiches gilt für die im Rahmen der Schlüsselvereinbarung genutzten öffentlichen und privaten Schlüssel.

3.1.8 Geschützte Kommunikation zwischen Sicherheitsmodul und Gateway

Für die Kommunikation zwischen Sicherheitsmodul und Gateway soll der Einsatz eines geschützten Kanals unterstützt werden. Hierbei soll die Integrität der Nachrichten, sowie die Vertraulichkeit der darin enthaltenen Datenfelder geschützt werden. Die Integritätssicherung soll dabei die gesamten Kommunikationsnachrichten umfassen. Um eine verlässliche Aussage über den Erfolg von Kommandos treffen zu können, müssen auch die Statusmeldungen der Antwortdaten in die Integritätssicherung inkludiert werden. Die Nutzung des geschützten Kanals ist nicht immer vorgeschrieben. Hat sich aber der Gateway-Betreiber am Sicherheitsmodul authentifiziert und ist damit der entsprechende Authentifizierungsstatus *PACE* gesetzt, so ist dessen Nutzung verpflichtend notwendig. Die Schlüsselvereinbarung soll mittels eines Passworts im Rahmen des zur Authentifizierung genutzten PAKE-Protokolls erfolgen. Durch den Zusammenhang zwischen dem Authentifizierungsstatus *PA-*

CE und der vorgeschriebenen Nutzung eines sicheren Kanals, kann auch der Dateizugriff implizit auf geschützte Kommunikation beschränkt werden.

3.1.9 Zufallszahlengenerator

Das Sicherheitsmodul soll einen kryptografisch sicheren Zufallszahlengenerator beinhalten. Dieser soll dem Gateway über einen Dienst zur Verfügung gestellt werden, sowie intern im Rahmen von kryptografischen Operationen (z. B. Schlüssel- oder Challenge Generierung) genutzt werden. Der Zufallszahlengenerator soll dabei einem der Standards DRG.3, DRG.4 PTG.3 oder NTG.1 genügen [BSI12-1].

3.1.10 Sicherer Betrieb und Inbetriebnahme

Das Sicherheitsmodul soll dermaßen ausgeführt sein, dass ein sicherer Betrieb garantiert wird. Ein Kriterium hierfür ist die Implementierung limitierter Funktionalität, d. h. nur die für den Betrieb notwendigen Funktionen werden realisiert. Weiters soll die Verfügbarkeit der Funktionalität limitiert sein, z. B. Funktionen für die Integration des Sicherheitsmoduls, oder für das Softwaredebugging sollen nicht während des Normalbetriebs verfügbar sein. Dies soll die missbräuchliche Verwendung von Funktionen verhindern. Weiters ist dafür zu sorgen, dass die Inbetriebnahme durch technische und organisatorische Sicherheitsmaßnahmen geschützt ist. So muss garantiert werden, dass die Daten (vor allem die Zertifikate und öffentlichen Schlüssel), welche bei der Initialisierung in dem Dateisystem gespeichert werden, gültig sind und nicht ausgetauscht oder manipuliert wurden.

3.2 Durch Sicherheitsmodul vermeidbare Security- und Privacy-Probleme

Der Einsatz eines Sicherheitsmoduls kann wesentlich dazu beitragen die Sicherheit des Smart Metering Gateways und damit des gesamten Smart Grids zu erhöhen. Dieser muss dabei immer in Zusammenarbeit mit dem Gateway selbst und der konsequenten Nutzung der zur Verfügung gestellten Dienste und kryptografischen Funktionen betrachtet werden. Im Folgenden sollen kurz die Auswirkungen eines umfassenden Einsatzes des Sicherheitsmoduls auf die Smart-Metering-Gateway-spezifischen Privacy- und Sicherheitsprobleme erörtert werden. Dabei wird von einem vollständig ausgeführten Sicherheitsmodul, welches alle in Kapitel 3.1 beschriebenen Anforderungen erfüllt, ausgegangen. Weiters wird von einer sicherheitstechnisch fehlerfreien Implementierung des Sicherheitsmoduls und auch des Gateways ausgegangen. Auch der umfassende und lückenlose Einsatz der Sicherheitsfunktionalität wird angenommen:

Die in Abschnitt 2.2 beschriebenen Säulen für den sicheren Betrieb in Smart Grids (technische Sicherheit, Privacy und Trust) werden alle durch die Verwendung des Sicherheitsmoduls verbessert. Durch den Einsatz der in Abschnitt 3.1 beschriebenen Dienste können verschiedene Sicherheitsziele der technischen Sicherheit verbessert werden:

- Vertraulichkeit: Die zu implementierende Zugriffskontrolle und das geschützte Dateisystem verbessern direkt die Vertraulichkeit von im Sicherheitsmodul abgelegten Daten. Dies sind jedoch im Allgemeinen keine Nutzinformationen, sondern Managementdaten für die Sicher-

heitsdienste. Dadurch ist jedoch ein sicherer Schlüsselspeicher vorhanden, in welchem das SMGW auch die Schlüssel für die lokal gespeicherten Daten ablegen soll. Authentifizierung soll sicherstellen, dass nur autorisierte Stellen Zugriff auf die Daten erhalten.

- Integrität und Authentizität: Durch den Einsatz von digitalen Signaturen und der Möglichkeit das Sicherheitsmodul als sicheren Speicher für Zertifikate zu verwenden kann die Integrität von Nachrichten besser garantiert werden. Die Integrität und Authentizität der Kommunikation mit dem Sicherheitsmodul soll durch die Implementierung des geschützten Kommunikationskanals (Secure Messaging) erreicht werden.
- Verfügbarkeit: Die Implementierung von Authentifizierungsdiensten im Sicherheitsmodul soll unerlaubte Zugriffe und damit eventuelle Manipulationen des SMGW verhindern.
- Zurechenbarkeit: Durch das Sicherheitsmodul werden keine Protokolle und Aufzeichnungen geführt. Der Authentifizierungsdienst und jener zur Überprüfung von Zertifikaten ermöglichen es aber, den Ursprung von Nachrichten nachzuweisen. Dieser kann von Protokollierungsdiensten im Gateway aufgezeichnet werden.

Einige der vom Modul angebotenen Dienste führen nicht direkt zu einer Erhöhung der Sicherheit, sondern stellen Funktionen für andere Sicherheitsdienste zur Verfügung. Hierzu zählen u. a. der Zufallszahlengenerator, die Dienste zur Schlüsselvereinbarung und zur Schlüsselgenerierung.

Der Datenschutz, im speziellen die Einhaltung von Datensparsamkeit, Zweckbindung, Erforderlichkeit kann nicht durch das Sicherheitsmodul verbessert werden. Diese müssen im Gateway implementiert werden. Die verschiedenen zur Verfügung gestellten Dienste erhöhen jedoch die Manipulationssicherheit des Gateways, sodass zumindest davon ausgegangen werden kann, dass die implementierten Mechanismen für die Einhaltung der Privacy nicht durch unerlaubte Zugriffe deaktiviert werden, bzw. eine unerlaubte zusätzliche Datenaufzeichnung eingerichtet werden kann.

Trust, das Vertrauen der Kunden in das System wird erhöht, wenn diesen kommuniziert wird, dass spezielle Sicherheitshardware zum Schutz der Sicherheit und der Privatsphäre entwickelt wurde und eingesetzt wird. Weiters sollte sich eine anerkannte Zertifizierung, wie die nach den Common Criteria des BSI Schutzprofils, vertrauensfördernd auswirken.

4 Plattform und eingesetzte Technologien

In diesem Kapitel werden die für die Implementierung des Sicherheitsmoduls eingesetzten Technologien und Plattform beschrieben. Zudem werden die Entscheidungen für deren Einsatz begründet.

4.1 Plattform

Bei der Entscheidung, auf welcher Plattform das Sicherheitsmodul implementiert und getestet werden soll, wurde vor allem auf folgende Punkte Wert gelegt: Skalierbarkeit, Kosten, Austauschbarkeit, Energieverbrauch, Erweiterbarkeit, Beschaffbarkeit, Security.

4.1.1 Sicherheitsmodul

Das zentrale Element dieser Arbeit ist die Entwicklung eines Sicherheitsmoduls für ein Smart Metering Gateway. Deshalb sind die Entscheidungen über die restliche Evaluierungsplattform vor allem von diesem Element abhängig. Es ist eine universelle Einsetzbarkeit des Sicherheitsmoduls gewünscht, was bei der Wahl der Plattform hierfür berücksichtigt werden soll. Die universelle Einsetzbarkeit wird einzig durch die Kommunikationsschnittstelle und das Kommunikationsprotokoll des Sicherheitsmoduls beschränkt. Durch den Einsatz einer auf [ISO7816-3] aufbauenden Schnittstelle und auf [ISO7816-4] aufbauenden Protokolls wird eine einfache Integration in verschiedene Systeme, vom Mikrocontroller (siehe z. B. [BLEIER04]) zum vollwertigen PC ermöglicht.

Als Implementierungsplattform für das Sicherheitsmodul wurde eine Smartcard welche der Java-Card-Spezifikation (im Weiteren als Javacard bezeichnet) entspricht ausgewählt. Gründe hierfür sind deren weite Verbreitung [RANKL08] sowie deren ständige Weiterentwicklung [ORACLE11]. Auch sind Javacards, welche die Standards ISO7816-3 und ISO7816-4 beherrschen einfach beschaffbar. Zudem sind Kryptografische- und Sicherheitsfunktionen bereits in der Java-Card-Spezifikation [SUN03] vorgesehen und werden von den Smartcardherstellern meist noch um eigene Funktionalität erweitert [NXP12].

Für die Evaluierung des Sicherheitsmoduls wurde hauptsächlich eine Smartcard des Typs JCLX80JTOP20ID von Infineon verwendet. Da hiermit nicht alle gewünschten Sicherheitsmerkmale implementiert werden konnten (siehe Abschnitt 6.1), wurden zusätzlich Simulatoren für die Evaluierung dieser Funktionen eingesetzt. Diese werden kurz in Abschnitt 4.4 und 5.11 beschrieben.

Die Software des Sicherheitsmoduls wurde in einem *Java Card* Applet implementiert. Dabei wurde die Spezifikation *Java Card (TM) Specification 2.2.1* [SUN03] als Grundlage herangezogen. Obwohl es mittlerweile Smartcards nach der neueren Spezifikation 3.0 Classic gibt, wurde aufgrund der vorhandenen Simulatoren und Smartcards diese ältere Spezifikation gewählt. Hierdurch ist eine einfachere Evaluierung des entwickelten Applets möglich. Zudem sind Smartcards nach der neuen Java-Card-Spezifikation 3.0 abwärtskompatibel und können auch mit für ältere Standards entwickelten Applets betrieben werden [ORACLE11].

Die Entwicklung des Applets selbst erfolgte in der *IFX JCIDE*, einer auf Eclipse basierenden Entwicklungsumgebung des Herstellers Infineon, von welchem auch die verwendete Smartcard stammt.

4.1.2 Smart Metering Gateway Plattform

Die weiteren Entscheidungen für die Implementierungsplattform zur Evaluierung des Sicherheitsmoduls hängen davon ab, auf Basis welches Smart Metering Gateway Software Framework das Sicherheitsmodul evaluiert und getestet werden soll. Aufgrund des bereits vorhandenen Einsatzes des *OGEMA* Frameworks am Institut sollte dieses eingesetzt werden. Es baut auf der dynamischen Softwareplattform *OSGi* auf und setzt hierfür eine JVM (Java Virtual Machine) voraus. Andere Abhängigkeiten ergeben sich durch die Wahl von *OGEMA* nicht.

Als Hardwareplattform für den Smart Metering Gateway wurde ein *Raspberry Pi* Model B Revision 1 (im Folgenden als *Raspberry Pi* bezeichnet) Einplatinencomputer gewählt. Hierfür spricht zum einen der geringe Preis von ca. 35 USD aber auch der geringe Stromverbrauch (ca. 2,5 W). Diese Punkte scheinen auf Grund des geplanten flächendecken Einsatzes von Smart Metering Gateways besonders wichtig zu sein. Zudem verfügt der *Raspberry Pi* über alle für die Evaluierung notwendigen Schnittstellen (Ethernet, USB) [RASP12].

Der *Raspberry Pi* verfügt über einen *ARM11* Prozessorkern, welcher mit 700 MHz getaktet ist. Weiters sind 256 MByte Arbeitsspeicher, 512 MByte in der neueren Revision 2 integriert. An Schnittstellen verfügt das Modul wie schon erwähnt über eine 100-Mbit-Ethernet-Schnittstelle und eine USB-2.0-Schnittstelle, welche bei der Evaluierung zum Zugriff auf das *OGEMA*-Webinterface sowie zum Anschluss des Smartcard-Lesegeräts verwendet wurden. Auch können weitere Sensoren oder Messsysteme über USB, oder über die weiteren vorhandenen Schnittstellen (I^2C, UART, GPIO) angebunden werden, wodurch auch die Integration eines Smart Meters in das System möglich ist.

Als Betriebssystem für die speziell für den *Raspberry Pi* entwickelte GNU/Linux-Distribution *Raspbian*. Diese basiert auf der als sehr stabil geltenden Debian-Distribution, welche um zusätzliche Unterstützung für die Fließkommaeinheit des *ARM11* erweitert wurde. Zudem ist für diese Distribution die Oracle JVM (Java7 SE) verfügbar, welche gegenüber der *OpenJDK* Implementierung eine signifikante Geschwindigkeitssteigerung bringen soll [ORACLE13]. Aus diesem Grund wurde auch die Oracle JVM jener von *OpenJDK* vorgezogen.

4.1.3 Smartcardlesegerät

Zur Verbindung des Sicherheitsmoduls mit dem Gateway wird ein Smartcardlesegerät benötigt. Zur Wahl stehen hier alle Lesegeräte, für welche ein Treiber für *ARM*-basierte GNU/Linux-

Distributionen verfügbar ist. Während der Entwicklung wurden hierbei ein *SCM Microsystems SDI010* sowie ein *Gemalto IDBridge CT40* verwendet und getestet. Bei dem SDI010 handelt es sich um ein Lesegerät, welches sowohl über eine Kontaktschnittstelle nach ISO7816-3 sowie über eine Funkschnittstelle nach ISO14443 verfügt. Dieses Gerät wird auch von der *Java-Card*-Entwicklungsumgebung *IFX JCIDE* unterstützt und wurde vor allem deshalb eingesetzt. Das *ID-Bridge* Lesegerät ist hingegen ein kleineres Gerät, welches nur über die Kontaktschnittstelle verfügt. Es würde sich daher besser für eine Integration eignen, auch da eine kontaktbasierende Integration in das Gateway vom BSI Schutzprofil vorgeschrieben wird [BSI13-2]. Beide Geräte werden unter GNU/Linux von den *libccid* Treibern [CCID] unterstützt.

4.2 Kryptografische Verfahren

Im Folgenden soll ein kurzer Überblick über einige der verwendeten kryptografischen Verfahren gegeben werden. Diese bilden die Grundlagen für die vom Sicherheitsmodul zur Verfügung zu stellenden Dienste. Spezielle Algorithmen wie AES (Advanced Encryption Standard) oder SHA (Secure Hash Algorithm) sollen dabei nicht erläutert werden, sondern nur grundlegende Verfahren, deren Kenntnisse für die Implementierung des Sicherheitsmoduls und das Verständnis des selbigen nötig erscheinen.

4.2.1 Public-Key-Kryptografie

Kryptografische Public-Key-Verfahren zählen zu den asymmetrischen Kryptosystemen. Bei den hierbei eingesetzten Verfahren zur Verschlüsselung, Signaturbildung und Schlüsselaustausch werden von den Kommunikationspartnern zwei verschiedene Schlüssel verwendet. Im Gegensatz zu symmetrischen Verfahren, wo es nur einen Schlüssel gibt, werden die beiden Schlüssel jeweils nur für einen Teil des Verfahrens verwendet, also entweder für Verschlüsselung oder Entschlüsselung bzw. Signaturbildung und Verifikation. Dies ermöglicht es, dass der Schlüssel für die Verschlüsselung (bzw. Verifikation) veröffentlicht wird (Public Key) und nur der Schlüssel zur Entschlüsselung (bzw. Signaturbildung) geheim gehalten werden muss (Private Key). Voraussetzung hierfür ist jedoch, dass sich der private Schlüssel nicht aus dem veröffentlichten Schlüssel berechnen lässt [BUCH10]. Durch die Aufspaltung des Schlüsselmaterials und die dadurch mögliche Veröffentlichung des Public Key entfällt die Notwendigkeit eines sicheren Kanals für den Schlüsselaustausch. Aufbauend auf bestimmte Public-Key-Verschlüsselungen können weitere kryptografische Dienste, z. B. für Signaturen und Schlüsseltausch realisiert werden. Einige dieser Verfahren werden in den Abschnitten 4.2.3, 4.2.4 und 4.2.5 beschrieben. Auch wenn die Notwendigkeit eines sicheren Kanals für den Schlüsselaustausch entfällt, so bedarf es doch einer Infrastruktur für die Verteilung der Schlüssel und die Sicherstellung, dass die öffentlichen Schlüssel auch zu dem privaten Schlüssel des gewünschten Kommunikationspartners gehören. Hierzu werden meist digitale Signaturen und eine Public-Key-Infrastruktur (PKI) verwendet. Diese und deren spezielle Anforderungen im Rahmen von Smart Grids werden in Abschnitt 6.3.1 erläutert. Ein Problem der Public-Key-Verschlüsselungen ist, dass sie bei gleichem Sicherheitsniveau mehr Rechenaufwand, und meist auch höhere Schlüssellängen als symmetrische Algorithmen benötigen [SPS11]. Deshalb werden diese Verfahren meist nicht zur Verschlüsselung von Datenströmen verwendet, sondern zum Aus-

tausch von symmetrischen Sitzungsschlüsseln, welche dann in der eigentlichen symmetrischen Stromchiffre eingesetzt wird.

Die Sicherheit von Public-Key-Verfahren hängt im Wesentlichen davon ab, dass sich der private Schlüssel unmöglich in vertretbarer Zeit aus den öffentlich verfügbaren Informationen berechnen lässt [BUCH10]. Um dies sicherzustellen, beruhen die Verfahren auf schwer zu lösenden Berechnungsproblemen der Zahlentheorie, welche als sogenannte Einwegfunktionen verwendet werden. Im speziellen basieren die meisten Verfahren auf den Problemen der Primfaktorzerlegung (z. B. RSA) oder auf diskreten Logarithmen (Digital Signature Algorithm - DSA). Die Schwierigkeit dieser Einwegfunktionen ist jedoch nicht garantiert. So ist bekannt, dass diese Verfahren z. B. durch Quantencomputer unsicher würden [SHOR97, PS08]. Weiters müssen Public-Key-Verfahren robust gegen passive (semantische Sicherheit und aktive Angriffe) sein, um als sicher zu gelten [SHOUP98]. Auch für kryptografische Verfahren, welche die vorgenannten Bedingungen erfüllen, gibt es keinen Beweis Ihrer mathematischen Sicherheit. Es werden sogenannte Sicherheitsreduktionen verwendet, welche beweisen, dass die Verfahren sicher sind, solange die zugrunde liegenden schwierigen mathematischen Berechnungsprobleme nicht gelöst wurden [BUCH10].

4.2.2 Elliptic-Curve-Kryptografie

Als Public-Key-Verfahren für den Einsatz im Sicherheitsmodul bieten sich die auf Elliptic-Curve-Kryptografie (ECC) basierenden Verfahren wie ECDH (Elliptic Curve Diffie Hellman) und ECDSA (Elliptic Curve Digital Signature Algorithm) an. Hauptgrund hierfür sind geringere Berechnungszeiten gegenüber Verfahren wie RSA und Diffie-Hellman-Schlüsseltausch, bei gleicher Sicherheit. Dies ist darauf zurückzuführen, dass für dasselbe Sicherheitsniveau wesentlich kürzere Schlüssellängen benötigt werden. Dadurch sinkt trotz der komplexeren Berechnungen bei ECC die Ausführungszeit [SPS11]. Weiterer Vorteil ist natürlich der geringere Speicherverbrauch auf dem Sicherheitsmodul sowie die geringere Nachrichtenlänge, welche aus den kürzeren Schlüsseln folgt.

Als Einwegfunktion für ECC wird der sogenannte Elliptic Curve Discrete Logarithm (ECDL) verwendet. Die Grundlagen hierzu sollen kurz erläutert werden:

Elliptische Kurven werden durch eine Formel des Typs $y^2 = x^3 + ax + b$ beschrieben. Für die Punkte auf diesen Kurven wird eine Additionsoperation definiert. Dies soll zuerst geometrisch erfolgen [SPS11] und auch in Abbildung 3 gezeigt werden:

- Für eine Gerade durch zwei gewählte Punkte (P und Q) der elliptischen Kurve wird immer ein dritter Punkt (R^*) der Kurve geschnitten.
- Eine vertikale Gerade durch R^* schneidet die Kurve in einem zweiten Punkt R.
- Punkt R wird als die Summe der Punkte P und Q, sowie als Inverses des Punkts R^* definiert. $P+Q = -R^* = R$

Ist die Gerade durch die Punkte P und Q vertikal (also $Q = -P$), so ist der dritte Schnittpunkt im Unendlichen und die Summe wird als 0 definiert. Damit ist auch $P+0=P$. Der Punkt im Unendlichen ist damit das Nullelement.

Voraussetzung für die Durchführbarkeit für alle Punkte auf einer Kurve ist, dass sich diese nicht selbst schneidet (wie z. B. für $a=-3$ und $b=2$ der Fall wäre), das Polynom $x^3 + ax + b$ also keine

mehrfachen Nullstellen hat. Eine Herleitung der algebraischen Formeln für die Addition kann in [SPS11] Kapitel 4.5.4 nachgeschlagen werden und führt mit *P(x1, y1), Q(x2, y2)* sowie *L* als Steigung der Geraden durch *P* und *Q* zu:

$$x_R = L^2 - x_1 - x_2$$

$$y_R = (x_1 - x_R)L - y_1$$

$$L = (y_2 - y_1)/(x_2 - x_1) \text{ für } P \neq Q$$

$$L = (3x_1^2 + a)/(2y_1) \text{ für } P = Q$$

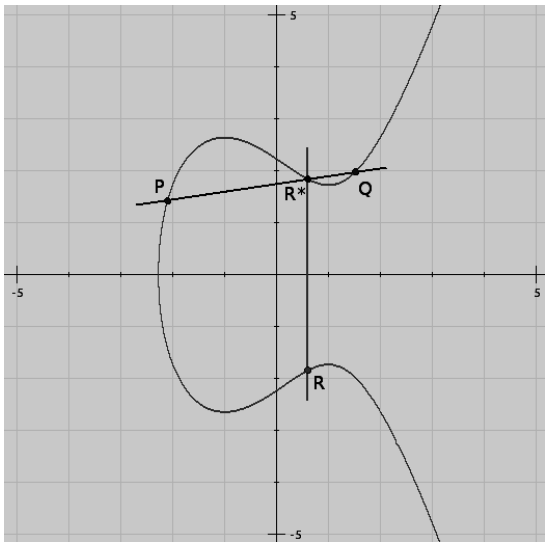


Abbildung 3: Additionsoperation auf elliptischen Kurven

Durch Verdoppelung und Addition ist daraus auch die Multiplikation herleitbar. Die genannten Formeln gelten für reelle Zahlen. Für kryptografische Funktionen werden diese für diskrete Werte abgewandelt, wobei aus der elliptischen Kurve eine Punktemenge wird. Dazu wird für ECC entweder in einem finiten Körper F_p oder in einem binären Körper F_{2m} des Grads *m* gearbeitet. Für das Sicherheitsmodul werden elliptische Kurven über F_p [BSI13-3] verwendet und die Operationen werden modulo einer Primzahl durchgeführt [SPS11]. Als ECDLP (Elliptic Curve Discrete Logarithm) wird nun das Problem bezeichnet, dass man in sehr großen Körpern für einen Punkt *P* der Kurve und eine ganze Zahl *n*, sich der Punkt $Q = nP$ einfach über die definierte Addition und Multiplikation berechnen lässt, *n* bei bekannten *P* und *Q* jedoch nicht. In kryptografischen Verfahren welche auf dem diskreten Algorithmus basieren, kann dieser mit wenigen Abwandlungen durch den ECDLP ersetzt werden [SPS11]. Da nicht alle elliptischen Kurven gleich gut für den kryptografischen Einsatz geeignet sind (z. B. Anzahl der Punkte auf der Kurve zu gering) wurden von verschiedenen Stellen standardisierte Kurvenparameter für gut geeignete Kurven veröffentlicht (siehe Abschnitt 5.4). Weiters ist anzumerken, dass viele Verfahren im Zusammenhang mit ECC patentiert sind und diese bei einer Implementation beachtet werden müssen [LOCH05].

4.2.3 Elliptic-Curve-Diffie-Hellman-Schlüsselaustausch

Das Diffie-Hellman-Verfahren wurde für die sichere Schlüsselvereinbarung zwischen zwei Kommunikationspartnern entwickelt. Dabei kann ein unsicherer Kanal verwendet werden, ohne dass dies die Vertraulichkeit des Schlüssels verletzt wird. Es handelt sich dabei nicht um ein Public-Key-Verfahren, hat aber Ähnlichkeiten zu diesen [SPS11]. Der Diffie-Hellman-Schlüsselaustausch beruht auf der Einwegfunktion des diskreten Logarithmus. In 21.4 [SB12] wird dieser von Stallings folgendermaßen definiert:

Eine Primitivwurzel einer Primzahl *p* ist dadurch definiert, dass jedes diskrete Element 1 bis p-1 durch die Potenzen der Primitivwurzel dargestellt werden kann. Falls also *a* eine Primitivwurzel von *p* ist, so sind die Ergebnisse von

a mod(p), a²mod(p), ... a^(p-1)mod(p)

alle verschieden und die Werte zwischen 1 und p-1, jedoch nicht unbedingt geordnet.

Für jede Ganzzahl *b*, kleiner *p* und Primitvwurzel *a* von *p* existiert ein Exponent *i*, so dass:

$b=a^i \bmod(p)$ *mit* $0<=i<=(p-1)$. Dieser Exponent *i* wird als diskreter Logarithmus bezeichnet. Für gegebene Werte *a, i* und *p* lässt sich sehr einfach *B* berechnen, jedoch ist kein effizienter Algorithmus zur Berechnung von *i* bekannt [SHOR97, BUCH10].

Soll zwischen den zwei Kommunikationspartnern Alice und Bob eine Schlüsselvereinbarung mit dem Diffie-Hellman-Verfahren erfolgen, gehen diese folgendermaßen vor [SPS11]:

1. Vereinbarung einer Primzahl *p* und einer Basis *g*,
2. Wahl einer Zufallszahl *a* durch Alice und *b* durch Bob,
3. Alice überträgt das Ergebnis $A=g^a \bmod(p)$ an Bob,
4. Bob überträgt das Ergebnis $B=g^b \bmod(p)$ an Alice,
5. Berechnung des gemeinsamen Geheimnisses $z=(g^b \bmod p)^a=B^a$ durch Alice bzw. $z=(g^a \bmod p)^b=A^b$ durch Bob,
6. Verwendung des Geheimnisses *z* zur Ableitung eines Schlüssels oder direkt als Schlüssel.

Allein durch die Kenntnis der übertragenen Parameter (*p, g, A, B*) kann *z* nicht berechnet werden. Dies wird als Diffie-Hellman-Problem bezeichnet [BUCH10].

Wie in Abschnitt 4.2.2 angemerkt, können solche Verfahren auch mit elliptischen Kurven und dem diskreten Logarithmus für elliptische Kurven (ECDLP) verwendet werden. Hierdurch verändert sich der Ablauf folgendermaßen [SPS11]:

1. Einigung auf die Parameter der Kurve (*a, b, p* siehe Abschnitt 5.4) und einen Basispunkt *G* welcher auf der Kurve liegt
2. Wahl einer geheimen Zahl *a* durch Alice, so dass $Q_A = aG$ ein Punkt der Kurve ist, und senden von Q_A an Bob
3. Wahl einer geheimen Zahl *b* durch Bob, so dass $Q_B = bG$ ein Punkt der Kurve ist, und senden von Q_B an Alice
4. Alice berechnet den Punkt *R*, welcher als gemeinsames Geheimnis genutzt wird durch $a*Q_B$, Bob berechnet es durch $b*Q_A$

Wiederum kann ohne Kenntnis der geheimen Zahlen *a* und *b* nicht auf *R* geschlossen werden. Die gewählten Parameter müssen sowohl für das Diffie-Hellman- (DH), als auch für das Elliptic-Curve-Diffie-Hellman-Verfahren (ECDH) bestimmte Voraussetzungen erfüllen, welche unter anderem in [BUCH10] nachgeschlagen werden können. Auf Diffie-Hellman basierende Verfahren sind anfällig für Man-In-The-Middle-Attacken und müssen über eine zusätzliche Authentifikationsschicht abgesichert werden [SPS11].

4.2.4 ElGamal-Verfahren

Das ElGamal-Verfahren baut auf dem Diffie-Hellman-Schlüsseltausch auf. Dessen Sicherheit ist dabei ebenfalls durch die Schwierigkeit des Diffie-Hellman-Problems begründet. Das ElGamal-Verfahren zählt jedoch zu der Public-Key-Kryptografie. Hierbei werden die bei DH mittels $A=g^a mod(p)$ ermittelten Berechnungsergebnisse zusammen mit der Primzahl *p* und der Basis *g* als öffentliche Schlüssel verwendet, sowie *a* als privater Schlüssel. Das ElGamal-Verfahren kann dann z. B. zum Schlüsseltausch verwendet werden [SPS11]:

- Alice generiert einen Sitzungsschlüssel $k=B^a mod(p)$ sowie einen Schlüsselwert $\alpha=g^a mod(p)$. Hierbei ist *a* eine von Alice gewählte Zufallszahl, *B* der öffentliche Schlüssel von Bob, *p* und *g* sind die auch bei DH verwendeten Primzahlen und Basis.
- Der Schlüsselanteil α wird an Bob gesendet, welcher *k* mithilfe seines privaten Schlüssels *b* berechnet: $(\alpha^b) mod\ p = k$.

Der nun beiden bekannte Schlüssel *k* kann z. B. für die Verschlüsselung mit einem symmetrischen Algorithmus verwendet werden. Das ElGamal-Verfahren kann auch direkt als Verschlüsselung genutzt werden [BUCH10]. Hierbei wird ein Klartext *m* aus dem Raum $\{0, 1, \ldots, p-1\}$ zum Schlüssel multipliziert: $c = B^a m\ mod(p)$. Dieser bildet nun gemeinsam mit dem Schlüsselanteil α den Schlüsseltext (α, *c*). Bob kann mithilfe seines geheimen Schlüssels den Klartext m entschlüsseln: Bob bestimmt $x = p - 1 - a$ und berechnet $m=\alpha^x c\ mod(p)$. Für den analytischen Beweis dafür sei auf [BUCH10] Kapitel 9.6 verwiesen.

Die Berechnungen für das ElGamal-Verfahren sind im Vergleich zu z. B. RSA aufwendiger und benötigen daher mehr Rechenleistung. Soll dieses Verfahren auf einem leistungsschwachen System wie einer Smartcard eingesetzt werden, so können die Berechnungen $B^a\ mod(p)$ und $\alpha=g^a\ mod(p)$ im Vorhinein auf Vorrat durchgeführt und gespeichert werden [BUCH10]. Wichtig ist hierbei, dass diese jeweils nur einmal verwendet werden.

4.2.5 Digitale Signaturverfahren

Soll die Authentizität einer Nachricht bzw. ihr Urheber garantiert werden, so kann auf digitale Signaturverfahren zurückgegriffen werden. Es handelt sich dabei im Allgemeinen um kryptografische Public-Key-Verfahren. Möchte der Kommunikationsteilnehmer Alice eine Nachricht signieren, so berechnet er aus seinem privaten Schlüssel *a* und der Nachricht *m* eine Signatur *s*. Die Signatur und die Nachricht werden zusammen versendet. Der Nachrichtenempfänger Bob kennt den öffentlichen Schlüssel *A* von Alice. Bob kann nun durch ein Verifikationsverfahren überprüfen, ob die Signatur über die mitgesendete Nachricht *m* mit dem privaten Schlüssel *a* von Alice erstellt wurde, oder nicht. Damit kann, die Vertraulichkeit des privaten Schlüssels vorausgesetzt, auf den Urheber ge-

schlossen und die Authentizität garantiert werden [BUCH10]. Zur Vereinfachung der Verfahren werden die Signatur- und Verifikationsalgorithmen nicht direkt auf die Nachrichten angewandt. Es werden zuerst kryptografische Prüfsummen (z. B. SHA – Secure Hash Algorithm), sogenannte Hash-Funktionen über die zu signierenden Daten gebildet. In speziellen Fällen ist der Einsatz von Hash-Funktionen vor der Signaturbildung auch notwendig um die Sicherheit zu erhöhen und Fälschungen zu vermeiden [BUCH10].

Das meist verwendete Verfahren für digitale Signaturen stellt wohl DSA (Digital Signature Algorithm) dar, ein effizienter auf den ElGamal-Verfahren beruhender Algorithmus, welcher vom US-amerikanischen NIST (National Institute of Standards and Technology) standardisiert wurde [BUCH10]. Dieser Algorithmus kann wiederum mit elliptischen Kurven und dem ECDL als Einwegfunktion verwendet werden. Man spricht dann vom Elliptic Curve Digital Signature Algorithm (ECDSA), welcher ebenfalls von NIST standardisiert wurde.

4.3 Smartcards und *Java Card*

Smartcards haben vielfältige Einsatzzwecke. Sie werden z. B. als Informationsspeicher, für Anwendungen in der Telekommunikation, in Banken, als Identifikationsmittel, sowie als Teil sicherheitskritischer Systeme und Anwendungen wie Zutritts- und Zugriffskontrollen verwendet [SPS11]. Zu den Smartcards zählen sowohl Speicherchipkarten als auch Prozessorchipkarten. Im Rahmen dieser Arbeit sind nur Letztere interessant, da sich nur mit Ihnen die gewünschten Dienste eines Sicherheitsmoduls realisieren lassen. Im Folgenden wird deshalb der Begriff Smartcard nur für Prozessorchipkarten verwendet.

Grundlegende Funktionen und Eigenschaften von Smartcards werden durch den mehrteiligen Standard ISO7816 spezifiziert. So werden in ISO7816-1 die physikalischen Eigenschaften, in ISO7816-2 die Platzierung und Größe von Kontakten und in Teil 3 die elektrischen Daten, sowie Übertragungsprotokollen von Smartcards beschrieben. Die Teile 4, 6, 7, 8 und 9 des Standards enthalten Spezifikationen für die Anwendungsschicht der Smartcard, im speziellen den Aufbau der Kommunikationseinheiten (APDU – Application Protocol Data Unit), Datenstrukturen sowie Kommandos zum Zugriff auf Daten, Dienste und Funktionen der Smartcard [RANKL08]. Für das im Rahmen dieser Arbeit implementierte Sicherheitsmodul wurden diese Standards, so weit als möglich berücksichtigt. Wo sich Widersprüche mit dem Schutzprofil [BSI13-2] ergaben, wurde diesem der Vorzug gegeben. Für die zugrunde liegende Spezifikation von Kommandos und Datenstrukturen sei deshalb in erster Linie auf die *Technische Richtlinie BSI TR-03109-2* des BSI [BSI13-3] verwiesen. Für den Einsatz und die Nutzung des Sicherheitsmoduls relevante Informationen bezüglich implementierter Datenstrukturen und Kommandos werden aber auch in Abschnitt 5 erläutert.

Programmierbare Smartcards verwenden zurzeit vor allem Java als Programmiersprache [SPS11]. Diese verwenden meist die Java-Card-Spezifikationen [SUN03, ORACLE11] als Grundlage. Es handelt sich dabei um eine ursprünglich von der Firma Sun Microsystems entwickelte Spezifikation für Multi-Applikations-Smartcards mit auf Java basierenden Applets [SPS11]. Hierfür wurde ein vom Standard-Java für PCs abweichendes Java definiert und entwickelt. Gründe dafür sind unterschiedlichen Anforderungen an Applikationen für Smartcards und für PCs sowie die zur Verfügung

stehenden Rechen- und Speicherkapazitäten. Die wesentlichen Einschränkungen der Programmiersprache, wie sie in der *Java Card Specification 2.2.1* [SUN03] definiert ist gegenüber dem Standard-Java sind Folgende [ORTIZ03]:

- Datentypen: Der Datentyp *int* ist optional, die Datentypen *long, float, double* und *char* sind nicht vorhanden, Arrays (Datenfelder) können nur eindimensional sein.
- Funktionen: keine Unterstützung für dynamisches Laden von Klassen, Threads, Klonen von Objekten, Security Manager und Package Zugriffskontrollen.
- Schlüsselwörter: native, synchronized, transient, volatile, strict werden nicht unterstützt.
- Standardklassen und -interfaces: *java.io, java.lang* und *java.util* sind auf *Object* und *Throwable* beschränkt und enthalten nur ein Subset deren Methoden.
- JVM: diverse Einschränkungen bzgl. Namenslängen für Packages und Klassen sowie für die Anzahl implementierbarer Interfaces, ableitbarer Klassen, Anzahl statischer Methoden

Die Java-Card-Spezifikation sieht jedoch auch, vor allem im Hinblick auf Smartcards, nützliche Erweiterungen gegenüber Standard-Java vor [RANKL06, ORTIZ03]:

- Datenpersistenz: Mit dem Schlüsselwort *new* erzeugte Objekte werden grundsätzlich im persistenten Speicher der Smartcard abgelegt und behalten auch nach Trennung der Smartcard von der Stromversorgung ihre Daten.
- Transaktionen: *Java Card* enthält Unterstützung für atomare Transaktionen. Transaktionen sind Teile des Programms, welche entweder komplett, oder gar nicht ausgeführt werden sollen, bzw. deren Änderungen an Daten entweder so erfolgt als wäre der Programmteil komplett abgearbeitet worden, oder gar nicht.
- *javacard.framework* API: Enthält Klassen für die Kommunikation (*APDU, ISO7816*), das Java Card System (J*CSystem*) und Helferklassen (*Util*). Hierüber wird effiziente Kommunikation, das Garbage Collecting, das Management von transientem Speicher sowie Zugriff auf Transaktionen und transaktionssicherer Methoden bereitgestellt.
- *javacard.rmi* API*:* Stellt Methoden und Interfaces für die RMI (Remote Method Invocation), also die Ausführung von Methoden auf der Smartcard durch den Host, zur Verfügung.
- *javacard.security* und *javacardx.crypto* API: Enthält Interfaces und Klassen, die zum Zugriff auf kryptografische Funktionen der Smartcard dienen. Dazu zählen Schlüsselcontainer, Prüfsummen, Schlüsselvereinbarung, Schüsselgenerierung, Hashfunktionen, Zufallszahlengenerierung, Ver- und Entschlüsselung sowie Signaturen.

Bevor ein *Java Card* Applet auf einer Smartcard installiert und ausgeführt werden kann, muss dieses zuerst in für den Interpreter der Smartcard gültigen Bytecode umgewandelt werden. Dazu wird der Quellcode zuerst von einem Java Compiler zu einer Class-Datei kompiliert. Die Class-Datei wird von einem Konverter in ein C*ard Application File* (CAP file) umgewandelt, welche auf die die Smartcard übertragen werden kann. Dort befindet sich ein Interpreter (JVM) für die Ausführung des in der CAP-Datei enthaltenen Bytecodes und für dessen Zugriff auf die Funktionen der Smartcard. Auf der Smartcard befindet sich ein weiteres vorinstalliertes Applet für das Management, der sogenannte Card Manager. Er stellt Kommandos zum Laden, Installieren und Löschen von Applets sowie zur Selektion zur Verfügung [RANKL08].

4.4 *Java-Card*-Simulatoren

Die zur Verfügung stehende Smartcard JCLX80JTOP20ID stellt nicht genügend Ressourcen für die Evaluierung des gesamten Applets des Sicherheitsmoduls zur Verfügung (siehe Abschnitt 6.1). Deshalb wurden zusätzlich *Java-Card*-Simulatoren eingesetzt, um das vollständige Applet testen und evaluieren zu können. Die beiden eingesetzten Simulatoren sind das *Java Card Workstation Development Environment (JCWDE)* und *jCardSim*.

4.4.1 *Java Card Workstation Development Environment (JCWDE)*

Der *JCWDE* ist ein Simulator, welcher im *Java Card Kit* [SUN03] enthalten ist. Er erlaubt das Ausführen von *Java Card* Applets und emuliert die hierfür nötige Smartcardumgebung. Hierfür wird die JVM des Wirtsystems verwendet. Die Funktionalität der *Java Card* wird hierbei nicht komplett nachgebildet [SUN03]. Fehlende Features sind:

- Paketinstallation: Der Vorgang der Paketinstallation findet nur rudimentär statt, indem die *install* Methode des Applets aufgerufen wird.
- Persistenz zwischen Neustarts: Daten des Applets bleiben nicht über Neustarts des Simulators hinweg erhalten.
- Firewall: Es existiert keine Firewall, welche den Zugriff auf die Daten des Applets beschränkt.
- Transaktionen: Zugriffe erfolgen nicht atomar, auch für Methoden, die dies laut Java-Card-Spezifikation können müssen.
- Garbage Collector: Arrays, die im flüchtigen Speicher angelegt werden, werden nicht durch die bei der Generierung angegebenen Aktion (Reset oder Deselektieren des Applets) gelöscht. Nicht referenzierte Objekte werden nicht durch den Aufruf der *JCSystem.requestObjectDeletion* Methode gelöscht.
- Löschen von Applets und Packages: Die hierfür vorgesehenen Funktionen des Installer Applets können nicht genutzt werden.
- APDU (Application Protocol Data Unit) Datenfelder mit mehr als 127 Byte: Das Datenfeld wird nach 127 Bytes abgeschnitten. Auch *extended APDUs* werden nicht unterstützt.
- Erweiterte Kryptografie: Es werden nur die von der Java-Card-Spezifikation vorgesehen kryptografischen Funktionen unterstützt und diese nur mit eingeschränkten Schlüssellängen (z. B. 192 Bit für Elliptic Curve Cryptography ECC). Auch stellt der Zufallgenerator nur den Modus für Pseudozufallszahlen zur Verfügung.

Für die Evaluierung der Funktionalität des Sicherheitsmoduls ist keine dieser Funktionen notwendig. Einzig die beschränkte Länge der APDU Datenfelder hat Einschränkungen beim Testen einzelner Funktionen erfordert (z. B. Größe der mit einem Kommando auslesbaren Daten aus einem Elementary File).

Die auszuführenden Packages des Applets müssen im *Classpath* dieser JVM verfügbar sein. Das zu verwendende Applet und dessen AID (Applet Identifier) muss dem Simulator über eine Konfigurationsdatei bekannt gegeben werden. Zusätzlich muss das Installer Applet und dessen AID angegeben werden (siehe Listing 1).

Listing 1: Konfigurationsdatei jcwde.app für den *JCWDE* Simulator

```
//jcwde.app
//applet                                    AID
com.sun.javacard.installer.InstallerApplet  0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0x8:0x1
securityModuleApplet.SecurityModule         0x01:0x00:0x00:0x00:0x02:0x01
```

Der *JCWDE* kann über eine Shell mit dem Aufruf jcwde *[-p port]* <*config-file*> gestartet werden. Zur Kommunikation mit dem Simulator kann das ebenfalls zum *Java Card Kit* gehörende *APDU-Tool* genutzt werden. Dazu werden in einer Skriptdatei die an den Simulator zu sendenden Befehle und die an das Applet zu sendenden Kommando-APDUs aufgelistet werden. Dies erlaubt aber nur die Abarbeitung einer zuvor festgelegten linearen Sequenz von Befehlen und Kommandos. Deshalb wurde das *com.sun.javacard.apduio* Package für die Kommunikation mit dem *JCWDE* Simulator gewählt. Dieses im *Java Card Kit* [SUN03] enthaltene Package ermöglicht es von einer Java Applikation aus Kommandos an den Simulator zu senden und die Antwort-APDUs zu empfangen. Detailliertere Informationen zur Implementierung werden in Abschnitt 5.11 beschrieben.

Da die Simulation des Applets in der JVM des Wirtsystems stattfindet, können die Standarddebugger für Java für die Fehlersuche und Evaluierung eingesetzt werden. Konkret wurde die Eclipse-Entwicklungsumgebung genutzt.

4.4.2 *jCardSim*

jCardSim [JCARDSIM] ist ein Open-Source-Simulator des Herstellers LICEL LLC [LICEL]. Dieser soll die Simulation einer Smartcard, die den Java-Card-Spezifikationen der Version 2.2.1 entspricht, ermöglichen. Hierzu verfügt er über Implementierungen der *javacard.framework, javacard.framework.security* und *javacardx.crypto* Packages. Im Gegensatz zu *JCWDE* werden auch größere Schlüssellängen unterstützt, sowie zusätzliche Algorithmen für die Schlüsselerzeugung und Zufallszahlengenerierung. Der für diese Arbeit wichtigste Unterschied war die Unterstützung von APDU-Datenfeldern mit mehr als 127 Byte.

Befehle können an den Simulator als Skriptdatei übergeben werden. Diese entsprechen syntaktisch denen des *JCWDE* und haben damit dieselbe Einschränkung des nur linearen Abarbeitens von Befehlssequenzen. Der Simulator verfügt auch über eine Java-API über die APDUs an das simulierte Applet gesendet, und von dem Antwort-APDUs empfangen werden können. Zusätzlich kann *jCardSim* ein Kartenlesegerät für Smartcards simulieren, auf das mittels der Standard Java Bibliothek *javax.smartcardio* zugegriffen werden kann. Diese Bibliothek wird auch zum Zugriff auf den externen Hardwarekartenleser benutzt. Daher kann damit auch die Implementierung des Zugriffs auf diese Lesegeräte evaluiert werden.

Die Simulation des Applets findet wiederum in der JVM des Wirtsystems statt, da es sich dabei nur um ein reines Java-Package handelt. Es können also auch hier die Standarddebugger für Java (wiederum die Eclipse-Entwicklungsumgebung) eingesetzt werden.

Während des Einsatzes des *jCardSim* wurden einige Bugs entdeckt, welche ein fehlerfreies Simulieren verhinderten. Aufgrund der Open-Source-Lizenz des Simulators konnten diese aber problemlos selbst behoben werden [GITH1].

5 Realisiertes Sicherheitsmodul und Umgebung

Im folgenden Kapitel wird das implementierte Sicherheitsmodul sowie die Einbindung in das Smart Metering Gateway beschrieben. Besondere Beachtung wurde einer weitgehenden Vereinbarkeit der Implementierung mit dem Schutzprofil für das Sicherheitsmodul der Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen des BSI [BSI13-02] gewidmet. Dies konnte jedoch nicht in allen Punkten realisiert werden. Auf die Abweichungen zum Schutzprofil wird nochmals genauer in Kapitel 6 eingegangen.

5.1 Architektur des Applets

Die Implementierung des *Java Card* Applets erfolgte hauptsächlich in der Klasse *SecurityModule*. Dies ist eine Subklasse der *Applet*-Klasse und implementiert die wesentlichen Funktionen. Die gesamte Klassenstruktur ist in Abbildung 4 dargestellt und wird im Folgenden beschrieben.
Die Klasse *SecurityModule* enthält die *Java-Card*-spezifischen Methoden für die Applet-Installation und Ausführung:

- *install*: wird bei der Installation des Applets auf der Smartcard ausgeführt und ruft den Konstruktor der Klasse auf. Dabei werden vor allem Initialisierungen vorgenommen. So wird z. B. das Dateisystem initialisiert, sowie die vom Secure Messaging verwendeten Schlüssel Instanzen werden erzeugt.
- *process*: Diese Methode wird vom Smartcardbetriebssystem aufgerufen, wenn eine APDU empfangen wurde. In der konkreten Implementierung wird zuerst die Methode *preProcess* aufgerufen. Diese übernimmt das Einlesen in den internen Empfangsbuffer, und falls das Secure Messaging (siehe Abschnitt 5.9) aktiviert ist, die Entschlüsselung und Integritätsverifikation. Die Abarbeitung der unverschlüsselten APDUs, also der Kommandointerpreter wird von der Methode *processPlain* übernommen. Die Methode *postProcess* dient der eventuell nötigen Umwandlung in ein gesichertes Antwort-APDU für das Secure Messaging, sowie der Ausgabe von aufgetretenen Exceptions.

Auch Variablen welche Debugging- und Entwicklungsfunktionen vor dem Kompilieren festlegen sind in dieser Klasse hinterlegt (siehe Abschnitt 6.3.5), ebenso wie die Variablen, welche den aktuellen Zustand des Sicherheitsmoduls speichern, etwa das Security-Environment (siehe Abschnitt 5.5) sowie aktuell verwendetes Schlüsselmaterial für das Secure Messaging. Auch die Konstanten, welche die Größen der Dateisystemelemente festlegen, sind hier deklariert (siehe Abschnitt 5.3).

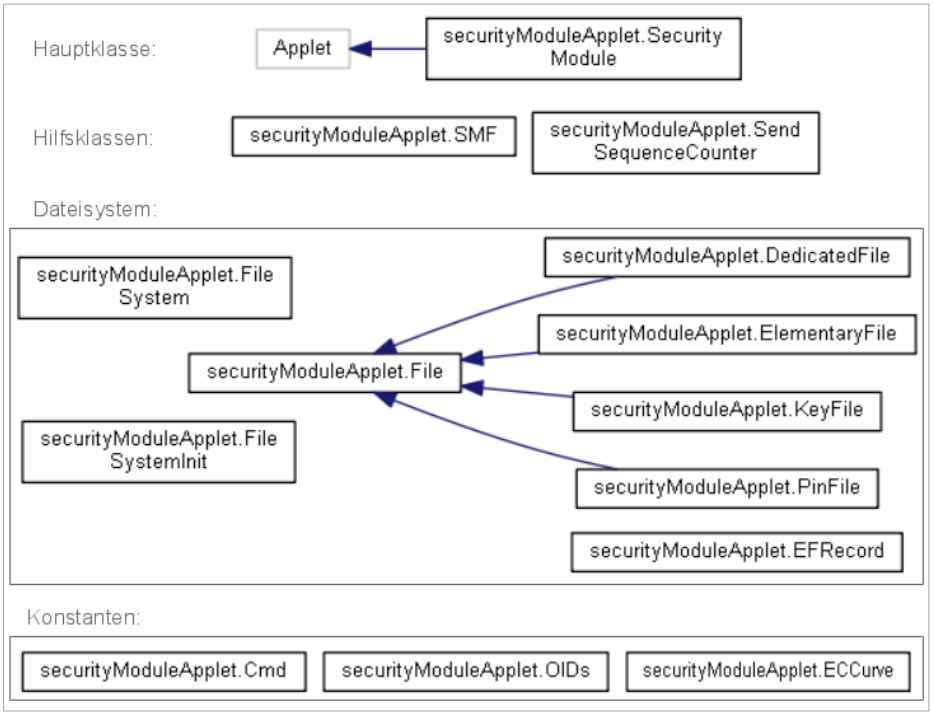


Abbildung 4: Klassenstruktur des Sicherheitsmoduls

Die Klassen *Cmd* und *ECCurve* dienen der Speicherung von Konstanten, welche während des Betriebs verwendet werden. So sind sämtliche Kodierungen für die Befehlsverarbeitung (Instruktionen, Parameter, Antwortdaten, Tags der BER/TLV kodierten Daten) in der *Cmd*-Klasse als *final static* Variablen hinterlegt. Die Parameter der verwendeten elliptischen Kurven (Domain Parameter, Schlüssellängen) sind in der *ECCurve* Klasse hinterlegt, welche weiters Hilfsfunktionen für die Verwendung der Domain Parameter enthält. Von der Klasse *File* sind die Subklassen *DedicatedFile*, *ElementaryFile (EF)*, *KeyFile* und *PinFile* abgeleitet. Instanziierte Objekte dieser Klassen repräsentieren Dateien im Dateisystem. Für die Speicherung von Daten in Record-basierten Elementary Files wurde die Klasse *EFRecord* angelegt, da somit die *Java Card* spezifische Beschränkung auf eindimensionale Arrays umgangen werden kann. Die instanziierten Dateien selbst werden in einem Objekt der Klasse *FileSystem* verwaltet. Es enthält Listen über die gespeicherten Daten, Methoden zur Selektion der Dateien über ihre Identifier sowie zum Anlegen und Entfernen von Dateien. Zugriffe auf die Dateien erfolgen ansonsten nur während der Befehlsverarbeitung aus dem Applet-Objekt über die *SecurityModule* Klasse. Diese erfolgen direkt über die *File*-Klassen, was einer Reduzierung des Overheads dient (siehe Abschnitt 5.2). Die notwendige Initialisierung des Dateisystems bei dessen Instanziierung wird von statischen Methoden der Klasse *FileSystemInit* erledigt (siehe Abschnitt 5.3.7).

Die Klasse *SendSequenceCounter* implementiert einen Zähler beliebiger Bitlänge, welcher für das Secure Messaging eingesetzt wird. Statische Hilfsfunktionen, welche während der Befehlsverarbeitung oder von kryptografischen Funktionen verwendet werden, wurden in die Klasse *SMF* ausgelagert.

5.2 Designentscheidungen bei der *Java Card* Applet Implementierung

Das *Java Card* Applet wird zwar in Java entwickelt, jedoch sind hierbei wichtige Unterschiede zu beachten, auf die im Folgenden eingegangen werden soll.

- Begrenzter nichtflüchtiger Speicher.
  Der nichtflüchtige Speicher, meist als NVRAM (Non Volatile Random Access Memory) ausgeführt, ist im Vergleich zu anderen Systemen für die JVMs existieren sehr klein. Die verwendete Infineon JCLX80JTOP20ID Smartcard verfügt z. B. über 75,5 kByte NVRAM [INFINEON08]. Dieser Speicher wird nicht nur für das Applet selbst verwendet, sondern alle Variablen, welche über das Java Schlüsselwort new instanziiert werden, werden dort gespeichert.
- Begrenzter flüchtiger Speicher
  Die eingesetzte Smartcard verfügt über ca. 4 kByte flüchtiges RAM. Dieser teilt sich auf den Stack, den APDU-Buffer, den Transaktionsbuffer und den Heap auf. In dem Heap können flüchtige Datenobjekte gespeichert werden, hierfür stehen aber nur ca. 1,8 kByte zur Verfügung.
- Begrenzte Rechenleistung
  Die Rechenleistung der Smartcard ist begrenzt. Die JCLX80JTOP20ID setzt einen SLE66CLX800PE mit maximal 30 MHz ein. Der Prozessor selbst besitzt eine 8/16 Bit 8051-Architektur. Zur Beschleunigung einiger kryptografischen Funktionen besitzt die Smartcard noch zwei Coprozessoren. Alle nicht beschleunigten Funktionen sind jedoch durch die zusätzliche Abstraktionsschicht der JVM relativ langsam. Deshalb ist es auch nicht sinnvoll fehlende kryptografische Funktionen im Applet selbst zu implementieren.
- Beschränkter Java-Sprachumfang
  Dies wurde in Abschnitt 4.3 bereits beschrieben. Hier ist noch einmal die nur rudimentär vorhandene Garbage Collection zu erwähnen.

Aufgrund dieser Beschränkungen wurden einige Unterschiede gegenüber der gewohnten Java Entwicklung gewählt, um die Ausführung des Applets zu optimieren:

- Vermeidung von Abstraktionsschichten
  Jede Abstraktionsschicht durch Vererbung und Interfaces führt zu zusätzlichem Overhead, sowohl im Speicherverbrauch als auch in der benötigten Rechenleistung. Deshalb wurde die Klassenstruktur möglichst flach gehalten und auf den Einsatz von Interfaces verzichtet. Auch wurden ähnliche Klassen (recordbasierende EF und transparente EF) als eine Klasse implementiert. Die Objekte werden mit unterschiedlichen Konstruktoren erzeugt, wobei nicht benötigte Attribute nicht instanziiert werden. Eine weitere Ressourceneinsparung wurde durch die Reduktion von Klassen insgesamt erreicht. So wurden alle statischen Methoden, welche von mehreren Klassen genutzt wurden, in einer gemeinsamen Helferklasse gesammelt.
- Vermeidung von Zugriffsmethoden auf Objektattribute
  Durch eine Erhöhung der Sichtbarkeit von Objektattributen konnte auf die Implementierung von Abfrage- (*get*) und Änderungsmethoden (*set*) verzichtet werden. Dies führt zu kleinerem

Applikationscode und weniger Methodenaufrufen, also einem geringeren Bedarf an Rechenzeit und Stack-Speicher.

- Einsatz der Framework Funktionen und Objekte
  Falls verfügbar wurden Funktionen des *Java Card* Frameworks genutzt. Diese müssen nicht von der JVM verarbeitet werden, sondern sind in nativem Code vorhanden und werden somit schneller abgearbeitet. Zudem wird Speicherplatz gespart. Auch wurden wenn möglich die *Java Card* spezifischen Objekte für die Speicherung von Schlüsselmaterial verwendet. Diese sind laut Spezifikation (I-SUN03) in besonders vor Manipulation und unerlaubtem Auslesen geschützten Speicherbereichen abzulegen. Dies hat jedoch einen höheren Speicherverbrauch zur Folge, da die Domain Parameter der elliptischen Kurven mehrfach Speicher belegen. Die Abarbeitungszeit und die Codegröße selbst sind jedoch geringer, da das Schlüsselmaterial vor der Nutzung der kryptografischen Methoden immer umkopiert werden müsste.
- Vermeidung von Schreibzugriffen auf den nichtflüchtigen Speicher (NVRAM)
  Schreibzugriffe auf den NVRAM sind langsam, da dieser meist auf einem EEPROM implementiert ist [RANKL08]. Schreibzugriffe darauf sollten also soweit nicht nötig vermieden werden. Buffer für die Verarbeitung der Daten und Abarbeitung der Kommandos wurden deshalb als flüchtige Objekte im Heap angelegt. Zudem werden alle im Vorhinein bekannten Objekte bei der Installation des Applets instanziiert. Dadurch kann die spätere Abarbeitung von Kommandos beschleunigt werden. Weiters werden Schreibzugriffe auf Objektattribute nur mit Endergebnissen und nicht mit temporären Daten ausgeführt.
- Allokation von Arrays
  Die JVM der Smartcard alloziert Speicher für Arrays in 32 Byte großen Blöcken [RUIM11]. Zusätzlich wird vor jedes Array ein bis zu 12 Byte langer Header angefügt. Aufgrund des beschränkten Speichers ist deshalb bei der Array-Allokation besondere Sorgfalt nötig. Es wurde bei der Wahl der Größe der Arrays auf diese Eigenheiten geachtet. Auch wurden einige Arrays zu einem größeren zusammengefasst, um den Overhead zu minimieren.
- Speicherminimierung primitiver Objekte
  Objektattribute mit primitivem Datentyp wurden, wo möglich als *static* deklariert, da hierfür weniger nichtflüchtiger Speicher benötigt wird [RUIM11]. Zudem wurden an vielen Stellen lokale Variablen durch globale (klassenweite) Attribute ersetzt. Dadurch muss weniger flüchtiger Speicher verwaltet werden. Zudem wird der rudimentär vorhandene Garbage Collector entlastet, was zu geringerer Ausführungszeit führen sollte.

Die hier beschriebenen Optimierungen wurden nicht immer angewandt. In vielen Fällen wurden sie bewusst ignoriert um die Les- und Wartbarkeit des Programmcodes zu erhalten.

5.3 Dateisystem

Wie vom BSI-Schutzprofil [BSI13-2] vorgesehen enthält das Sicherheitsmodul ein Dateisystem zum Speichern von technischen Datenfeldern sowie zur sicheren Aufbewahrung kryptografischer Daten. Hierzu wurden zum einen transparente und Record-orientierte Datenfelder zum Speichern von tech-

nischen Daten und Zertifikaten implementiert, aber auch, speziell vor dem Auslesen geschützte Schlüssel- und PIN-Dateien. Weiters sind sogenannte Dedicated Files (DF) implementiert, welche als Verzeichnis für andere Dateien dienen und in Ihrer Funktion einem Ordner in klassischen Dateisystemen entsprechen. Dateien werden im Allgemeinen über eine File-ID referenziert, bei Schlüsseln und PINs durch entsprechende Key- und PIN-IDs. Die verschiedenen Dateitypen werden im Folgenden beschrieben und sind in Abbildung 5 dargestellt.

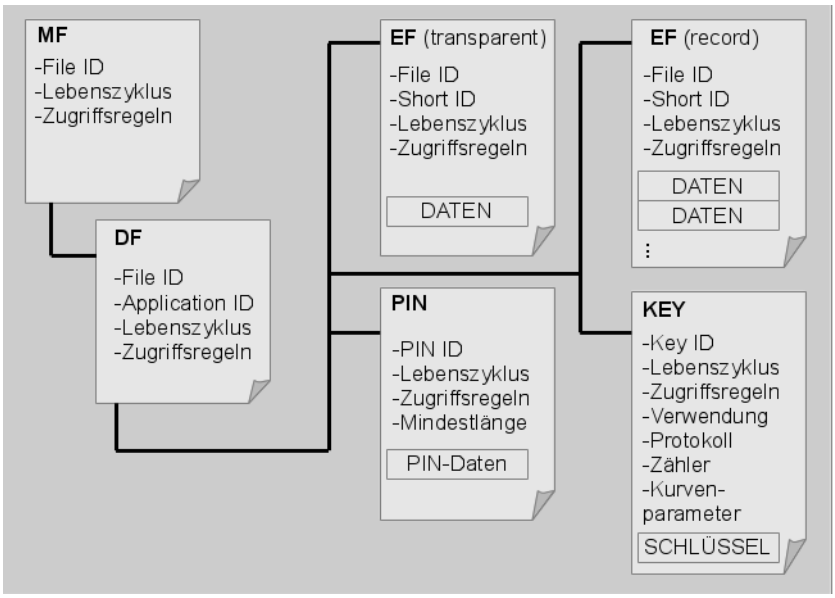


Abbildung 5: Dateisystem des Sicherheitsmoduls

5.3.1 Transparente Elementary Files

Transparente Elementary Files (EF) sind Datenfelder zum Speichern von beliebigen Rohdaten. Der Zugriff erfolgt entweder durch eine für das komplette Filesystem eindeutige File-ID oder durch eine SFI (Short File ID) welche immer auf den aktuell selektierten Ordner bezogen sind. Die gespeicherten Daten selbst können als Sequenzen von Bytes, von beliebigen Offsets und mit beliebiger Länge gelesen und geschrieben werden. Die Datei selbst wird hierbei als Speicher, rein für externen Zugriff verwaltet. Eine Verarbeitung der Daten innerhalb des Sicherheitsmoduls ist nicht vorgesehen. In der aktuellen Implementierung sind maximal 1024 Byte pro transparenten Elementary File speicherbar. Diese Einschränkung ist jedoch nur dem begrenzten Speicherplatz der eingesetzten Smartcard geschuldet und kann bei Austausch dieser nötigenfalls erhöht werden. Haupteinsatzzweck der transparenten Elementary Files ist das Speichern von Zertifikaten, jedoch ist auch die Speicherung technischer Informationen, welche das Sicherheitsmodul betreffen, vorgesehen. Weiters wird zu jedem EF der Lebenszyklus in dem sich die Datei befindet mitgespeichert. Hierzu zählt, ob die Datei aktiviert oder deaktiviert ist. Ein lesender und schreibender Zugriff ist nur bei aktiver Datei zulässig. Zusätzlich können Dateien terminiert werden. Dies ist eine irreversible Aktion und verhindert den späteren Zugriff auf die Datei und die darin enthalten Informationen.

5.3.2 Record-basierende Elementary Files

Record-basierende Elementary Files sind ebenfalls Datenfelder zum Speichern von Rohdaten. Der Zugriff erfolgt wiederum entweder durch eindeutige File ID oder ordnerbezogene Short File ID. Im Gegensatz zu den transparenten Elementary Files können diese aus mehreren Einträgen bzw. Records bestehen, auf die unabhängig zugegriffen werden kann. Ein Zugriff über einen Offset ist jedoch nicht möglich. Eine maximale Anzahl von drei Records mit je 128 Byte ist vom Sicherheitsprofil [BSI13-3] vorgesehen. Eine Erweiterung ist jedoch nur durch den Speicher der eingesetzten Smartcard begrenzt.
Record-basierende Elementary Files dienen vor allem der Speicherung von technischen Informationen über das Sicherheitsmodul. Hierzu zählen zum Beispiel Informationen zum Lebenszyklus, in dem sich das Modul befindet, aber auch Informationen über die implementierte Spezifikation. Weiters ist die Speicherung von symmetrischen Schlüsseln, welche vom Gateway genutzt werden, in Record-basierenden Elementary Files vorgesehen. Der für transparente EFs existierende Lebenszyklus wird auch für Record-basierende EFs eingesetzt.

5.3.3 Dedicated Files

Dedicated Files (DF) sind, wie bereits angesprochen, Container für weitere Dateien und auch weitere Dedicated Files. Sie ermöglichen es eine Ordnerstruktur auf dem Filesystem zu realisieren. Ein Dedicated File kann außer über die für das Filesystem eindeutige FID, auch über einen Application Identifier (AID) selektiert werden. Diese AID ist bis zu 16 Byte lang und sollte für jede Anwendung eindeutig sein. Um dies zu garantieren, werden die AIDs von Standardisierungsstellen zugewiesen und sollten nicht frei gewählt werden [RANKL08].
Lebenszyklusinformationen stehen auch für DFs zur Verfügung. Deaktivierte oder terminierte DFs können nicht selektiert werden, wobei die Terminierung im Gegensatz zur Deaktivierung nicht reversibel ist.
Die aktuelle Implementierung sieht bis zu 16 Elementary Files, bis zu 32 Key Files und bis zu fünf PIN Files pro Dedicated File vor. Mit diesen Werten sind die vom Sicherheitsprofil für Sicherheitsmodule [BSI13-3] vorgesehenen Dateien speicherbar, eine Erweiterung ist jedoch wiederum möglich.

5.3.4 Master File

Das Master File (MF) ist eine spezielle Form des Dedicated File und entspricht dem Wurzelverzeichnis des Dateisystems. Es ist mit der festen FID 0x3F00 versehen und hat keine AID. Das Master File wird bei jedem selektieren des Applets als aktuelles Verzeichnis ausgewählt. Ein späteres manuelles Selektieren des Master File führt zum Zurücksetzen bereits erfolgter Authentifizierungen gegenüber dem Sicherheitsmodul.

5.3.5 PIN File

Die PIN Files verfügen im Gegensatz zu Elementary und Dedicated Files über keine dateisystemweit eindeutigen FIDs und können nur relativ zum selektierten DF angesprochen werden. Hierzu

wird eine 1 Byte lange PIN-ID verwendet. PIN Files werden für die sogenannte *PACE* (Password Authenticated Connection Establishment) Authentifizierung eingesetzt. Mit dieser authentifiziert sich zum einen der Gatewaybetreiber am Sicherheitsmodul, zum anderen erfolgt hierüber ein Schlüsselaustausch um eine gesicherte (verschlüsselte und authentifizierte) Kommunikation zwischen Gateway und Sicherheitsmodul zu initiieren (Secure Messaging). Der Inhalt des PIN File wird also auch intern im Sicherheitsmodul verwendet. Zusätzlich zu der Bytesequenz, die als PIN verwendet wird, wird auch jeweils eine Mindestlänge für diese Bytesequenz gespeichert. Diese darf bei Änderungen der PIN nicht unterschritten werden. Eine weitere Information, die jeweils mitgespeichert wird, ist der Lebenszyklus, in dem sich die Datei befindet (Initialisierung, aktiviert oder deaktiviert). Abhängig von diesem Lebenszyklus können bestimmte Befehle auf die Datei nicht angewandt werden. Hierauf wird genauer bei der Beschreibung dieser Befehle eingegangen (siehe Abschnitt 0 und 5.11.5.7). In der aktuellen Implementierung ist kein nachträgliches Erzeugen von PIN Files vorgesehen. Es sind also nur jene PIN Files vorhanden, welche bereits bei der Initialisierung des Dateisystems angelegt wurden. Aufgrund einer festgelegten Mindestlänge der Bytesequenz von 10 Bytes bei der Initialisierung wird wie von [BSI13-03] vorgeschlagen auf einen Fehlversuchszähler verzichtet. Die Bytesequenz, welche als PIN verwendet wird, ist nach ISO/IEC8859-1 kodiert. Dadurch dürfen keine Werte aus den Bereichen 0x00-0x1F sowie 0x7F-0x9F enthalten sein [BSI13-4].

5.3.6 Key File

Zentrales und wichtigstes Element des Dateisystems sind die Key Files bzw. Schlüsselobjekte. Alle kryptografischen Operationen, die das Sicherheitsmodul anbietet, sind auf Key Files angewiesen. Es handelt sich stets um asymmetrische Schlüssel für Elliptic-Curve-Kryptografie. Bei der Verwaltung der Schlüsselobjekte sind zwei Typen zu unterschieden:

- Schlüssel, welche im Besitz des Gateways sind: Diese werden im Sicherheitsmodul generiert und nur der öffentliche Teil des Schlüsselpaars darf die Smartcard verlassen. Zertifikate für die Verwendung der Schlüssel werden extern von einer Zertifizierungsstelle erzeugt und anschließend im Sicherheitsmodul gespeichert.
- Schlüssel, welche nicht im Besitz des Gateways sind: Von diesen Schlüsselpaaren werden nur die öffentlichen Keys und die dazugehörigen Zertifikate im Sicherheitsmodul gespeichert. Diese können entweder bereits bei der Inbetriebnahme in das Sicherheitsmodul geladen werden, oder im Betrieb durch den Empfang eines signierten Kommandos importiert werden. Zudem ist die Verifikation einer Signatur auch mit einem nicht im Dateisystem gespeicherten Schlüssel möglich. Hierzu ist ein temporärer Import eines Schlüssels möglich (siehe Abschnitt 5.11.4).

Im Folgenden wird zur Unterscheidung von diesen beiden Schlüsseltypen auf die Bezeichnung Schlüsselpaar und öffentlicher Schlüssel zurückgegriffen.
Sowohl öffentliche Schlüssel als auch Schlüsselpaare werden mit verschiedenen Zusatzinformationen gespeichert. Hierzu zählen zum einen der Lebenszyklusstatus des Schlüsselobjekts (*initialisation, active, deactivated*), welche wiederum Einfluss auf die durchführbaren Sicherheitsoperationen haben (siehe Abschnitt 5.6). Zu jedem Schlüsselobjekt wird weiters der Einsatzzweck, sowie mit welchem kryptografischen Protokoll der Schlüssel verwendet werden kann, gespeichert. Unterschie-

den wird zwischen Schlüsseln für Authentifikation und solchen für digitale Signaturen. Als Protokolle werden in der aktuellen Implementierung zwischen ECDSA und ECKA-EG unterschieden.
Wichtigstes gespeichertes Schlüsselattribut ist jedoch der Schlüssel selbst und die elliptische Kurve auf welcher der Schlüssel basiert. Die Auswahl eines Schlüsselpaars erfolgt über eine 1 Byte lange Key ID, die eines öffentlichen Schlüssels über eine 4 bis 8 Byte lange.
Für Schlüsselpaare wird zusätzlich ein Verwendungszähler vorgesehen. Wird die beim Generieren des Objekts vorgegebene Maximalanzahl an Verwendungen überschritten, wird das Schlüsselpaar deaktiviert und ein neues muss generiert werden.
Weiters unterscheiden sich Schlüsselpaare und öffentliche Schlüssel durch die mit ihnen ausführbaren Sicherheitskommandos. So werden Kommandos zur Schlüsselgenerierung und zum Erstellen von digitalen Signaturen, sowie die Authentifizierung gegenüber dem Gateway nur von Schlüsselpaaren unterstützt. Kommandos zum Verifizieren von digitalen Signaturen und Zertifikaten, sowie die Authentifizierung des Gateways gegenüber dem Sicherheitsmodul benötigen einen öffentlichen Schlüssel. Für die Kommandos zur Schlüsselübereinkunft werden je nach gewählter Variante öffentliche oder Schlüsselpaare benötigt.

5.3.7 Initialisierung des Dateisystems

Die Initialisierung des Dateisystems generiert alle Dateien, welche bei der ersten Inbetriebnahme des Sicherheitsmoduls zur Verfügung stehen sollen. Hierzu wurde die Klasse *FileSystemInit* des Applets implementiert. Für die Funktionsfähigkeit des Sicherheitsmoduls muss mindestens das Master File angelegt werden, da dieses dem Wurzelverzeichnis des Dateisystems entspricht. Dies kann über die Methode *initFileSystemLight* der genannten Klasse erreicht werden. Für einen sinnvollen Betrieb sind jedoch weitere Dateien erforderlich. So wird bei Nutzung der Methode *initFileSystem* das vom Technical Report [BSI 13-3] in Tabelle 2 und 3, sowie Abbildung 2 definierte initiale Dateisystem angelegt. Die Objekte werden dabei leer angelegt und müssen entweder im Betrieb, oder durch Erweiterung der Initialisierungsmethode befüllt werden. Falls der Debuggingmodus des Applets aktiviert ist (DEBUG = true), werden auch Testschlüssel und PIN-Dateien erstellt. Diese enthalten bereits Schlüsseldaten, welche in Tabelle 1 angegeben sind. In der konkreten Implementierung wird die Methode *initFileSystem* einmalig, im Rahmen der Instanziierung des Applets ausgeführt.

Tabelle 1: Testschlüssel und PIN-Dateien für Debuggingzwecke, in Dedicated File mit FID 0x1111 und AID 0x000102030405060708

Testdateien und IDs	Schlüsselwerte (elliptische Kurve BRAINPOOLP192R1)
Schlüsselpaare: 0x01, 0x02, 0x03, 0x04	geheimer Schlüssel: {0, -89, 88, -124, -89, -68, 30, 29, 6, 72, -95, 58, -43, 9, 44, 41, -23, -23, -16, 96, -29, 120, -111, -58, -55}
öffentliche Schlüssel: 0x00000101, 0x00000102, 0x00000103, 0x00000104	Kurvenpunkt: {4, 77, -123, 44, -15, -39, 81, -17, -124, -53, 82, 13, -110, -81, 117, -104, -21, 15, -38, 80, 75, 56, -102, 125, 25, 59, 127, 74, 62, 8, 113, -33, -108, 49, 126, -25, 48, 42, -53, -85, 111, -76, 0, -61, 26, 81, 9, -32, 46}
PIN: 0x01, 0x02	{65, 66, 67, 68, 69, 70, 71, 72, 73, 74} bzw. leer

5.4 Elliptische Kurven und OIDs

Für die Verwendung von kryptografischen Funktionen, die auf elliptischen Kurven basieren, bilden die dem jeweiligen Schlüssel zugrunde liegenden Kurven ein wichtiges Kriterium für die Sicherheit der Schlüssel (siehe auch Abschnitt 4.2.2). Deshalb sollten nicht beliebige, sondern nach mathematischen Verfahren überprüfte Kurven [LOCH05] verwendet werden. Die Kurven sind dann durch ihre charakteristischen Parameter (Primzahl, Koeffizienten, Basispunkt, Ordnung und Cofaktor), die sogenannten Domain Parameter definiert und beschrieben.
Für viele bereits überprüfte und von Standardisierungsinstitutionen veröffentlichte Kurven besteht jedoch auch die Möglichkeit diese über eine eindeutige OID (Object Identifier) zu beschreiben. Von dieser Möglichkeit wird auch im Sicherheitsmodul Gebrauch gemacht. Solche OIDs existieren auch für kryptografische Algorithmen und Protokolle. Für die Referenzierung von Domain Parametern einer elliptischen Kurve, von Protokollen oder Algorithmen in Kommandos an das Sicherheitsmodul, oder in den Antwortdaten werden deren OIDs verwendet.
Vom Sicherheitsmodul werden die elliptischen Kurven mit der Bezeichnung BrainpoolP_R1 mit 192, 256, 384 und 512 Bit Länge [LOCH05], sowie die NIST_P (auch als SEC_P) [NIST99] bezeichnete Kurven mit 256 und 384 Bit Länge unterstützt. Weiters werden die OIDs für die Algorithmen *id-ecdsa-plain-signatures* und *id-ecka-eg*, sowie für die Protokolle *id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-ECKA-DH-woKDF, id-ECKA-EG-REC-woKDF*, sowie *id-ECKA-EG-INI-woKDF* im Sicherheitsmodul unterstützt und verwendet.

5.5 Security Environment

Der Zustand des Sicherheitsmoduls bezüglich der Security-Funktionalität wird durch das *Security Environment* beschrieben. Die darin verwalteten Informationen werden nachfolgend beschrieben.

- Lebenszyklus des Moduls: Beim Lebenszyklus handelt es sich im Grunde nur um die Information, ob das Sicherheitsmodul noch aktiv und einsatzbereit ist, oder ob es bereits terminiert wurde. Es werden hierbei die Phasen *nicht-initialisiert (Initialisierung), initialisiert* und *terminiert* verwaltet. Wurde das Sicherheitsmodul terminiert, so werden keine Befehle mehr verarbeitet, sondern es wird nur mehr ein entsprechender Fehlercode generiert. Die Terminierung eines Sicherheitsmoduls ist endgültig und irreversibel. Auf gespeicherte Daten, wozu vor allem auch das Schlüsselmaterial gehört, kann nicht mehr zugegriffen werden.
- Aktuelle Betriebsart (SEID): Durch die SEID (Security Environment ID) wird beschrieben, in welcher Betriebsart sich das Modul gerade befindet. Es stehen hierbei die SEID 1 für den *Normalbetrieb*, sowie die SEID 2 für die *Personalisierung und Integration* des Moduls zur Verfügung. Für die beiden Betriebsarten gelten unterschiedliche Zugriffsregeln auf das Modul. Insbesondere werden für Schlüssel- und PIN-Dateien getrennte Zugriffsrechte, je nach Aktivem SEID verwaltet.
- Status der Authentifizierungen: Alle Zugriffsrechte für Dateien und Ausführungsrechte für Kommandos sind von dem aktuellen Status der Authentifizierungen abhängig. Es wird hier-

bei die Information gespeichert, ob sich das Gateway erfolgreich am Sicherheitsmodul authentifiziert hat (externe Authentifizierung, Sicherheitszustand *AUTH*) und ob sich der Betreiber am Modul authentifiziert hat (*PACE* – Authentifizierung, Sicherheitszustand *PACE*). Die weiteren zur Verfügung stehenden Authentifizierungsmethoden (interne Authentifizierung des Sicherheitsmoduls gegenüber dem Gateway und Schlüsselvereinbarungen für gesicherte Verbindungen) beeinflussen das Security Environment nicht.

- Informationen für nachfolgende kryptografische Kommandos: Sind für kryptografische Kommandos weitere Daten nötig welche nicht im Kommando selbst übertragen werden, so werden diese zuvor über Management-Kommandos für das Security Environment (MSE) gesetzt. Hierzu zählen je nach Bedarf: das zu verwendende Schlüsselobjekt, die zu verwendenden elliptischen Kurven, das zu verwendende kryptografische Protokoll, sowie für welches nachfolgende Kommando diese Daten gesetzt wurden. Diese im Security Environment gesetzten Daten sind jeweils nur für den unmittelbar danach durchgeführten Befehl gültig und werden dann zurückgesetzt.
- In diese Kategorie fallen weiters die gespeicherten Zufallszahlen, welche für ein Challenge-Response Verfahren zur externen Authentisierung benötigt werden. Diese werden im Rahmen eines speziellen GET CHALLENGE Befehls generiert, ausgegeben und für die spätere Verwendung gespeichert (siehe Abschnitt 5.6).

5.6 Zugriffsrechte

Für einen sinnvollen Betrieb des Sicherheitsmoduls sind umfangreiche Zugriffssicherungen auf die Kommandos und Dateien vorhanden. Für die Kommandos sind diese fest implementiert, für die Dateien (Datenfelder und Schlüsselobjekte) sind diese jedoch individuell für jede Datei festlegbar. Für Zugriffe auf Datenfelder (EF) und Verzeichnisse (DF und MF) werden die in Tabelle 2 beschriebenen Aktionen unterschieden.

Tabelle 2: Beschreibung der reglementierten Zugriffsaktionen auf Daten

Aktion	Beschreibung
ACTIVATE (aktivieren)	Ändern des Lebenszyklus auf aktiv
DEACTIVATE (deaktivieren)	Ändern des Lebenszyklus auf inaktiv
TERMINATE (terminieren)	Ändern des Lebenszyklus auf terminiert
DELETE (löschen)	Löschen der Datei
READ (lesen)	Lesen von einem Datenfeld (nur für EF)
WRITE (schreiben)	Schreiben in ein Datenfeld (nur für EF)
APPEND (Record hinzufügen)	Hinzufügen eines Records zu einem Record-basierten EF

Für jede dieser Zugriffsaktionen können individuell Zugriffsbedingungen festgelegt werden:

- erfolgreiche *PACE* Authentifizierung
- erfolgreiche *PACE* und externe Authentifizierung
- immer

- niemals

Die Zugriffsbedingungen werden in jeweils 2 Byte pro Aktion gespeichert (siehe Abbildung 4).

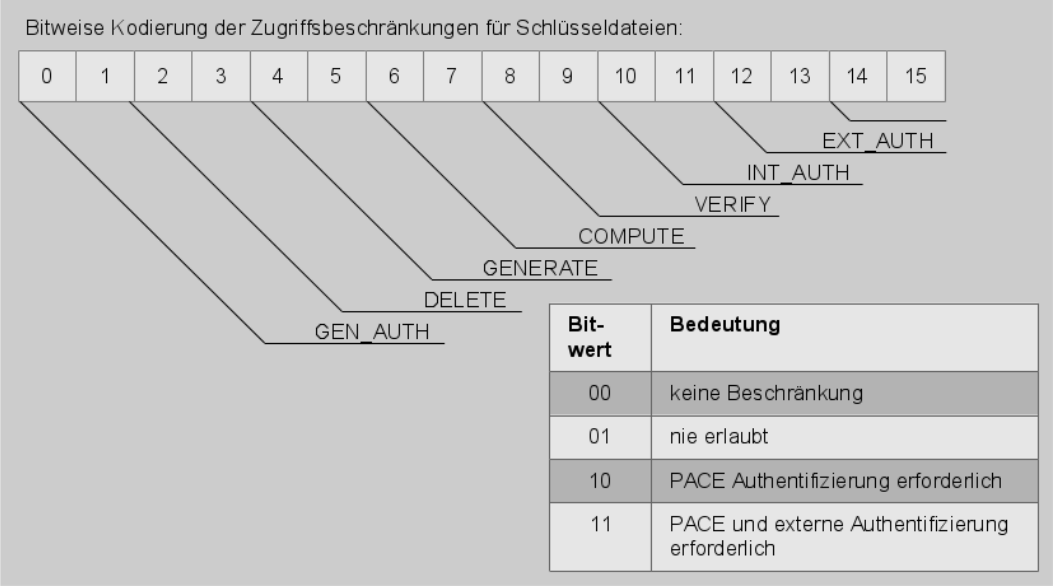


Abbildung 6: Zugriffsbeschränkungen auf Elementary und Dedicated Files

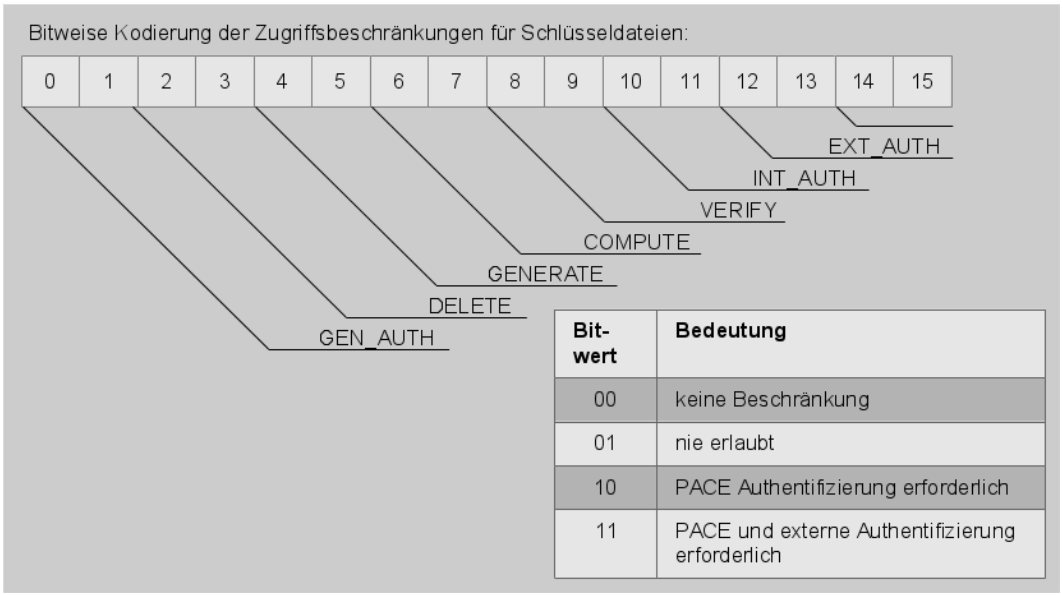
Für Schlüsselobjekte und PIN-Dateien wird ein ähnlicher Ansatz wie für Datenfelder und Verzeichnisse verfolgt. Jedoch werden hier für die beiden Betriebsarten *Personalisierung* und *Normalbetrieb* getrennte Zugriffsbeschränkungen verwaltet. Zudem enthält jede dieser Dateien die Information, in welchen der beiden Betriebsarten die Ausführung kryptografischer Kommandos möglich ist. Die Zugriffaktionen unterscheiden sich hier jedoch von denen der Datenfelder und der Verzeichnisse.

Tabelle 3: Zugriffsaktionen auf Schlüsselobjekte und PINs

Aktion	**anwendbar auf Objekttyp**
SET (Referenzdaten setzen)	PIN
CHANGE (Referenzdaten ändern)	PIN
DELETE (Löschen der Datei)	alle Schlüsseldateien
GENERATE (Generieren von Schlüsseldaten)	Schlüsselpaare
COMPUTE (Erzeugen von digitalen Signaturen)	Schlüsselpaare
VERIFY (Verifizieren von digitalen Signaturen)	alle Schlüsseldateien
INT_AUTH (interne Authentifizierung)	Schlüsselpaare
EXT_AUTH (externe Authentifizierung)	alle Schlüsseldateien
GEN_AUTH (Schlüsselvereinbarung)	alle Schlüsseldateien

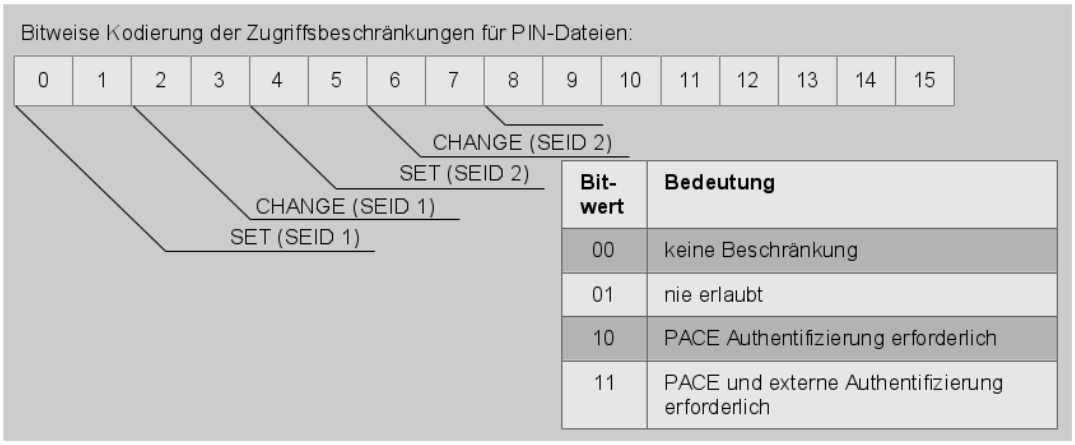
Die Zugriffsregeln auf Schlüsselobjekte und PIN-Dateien werden bei der Initialisierung des Sicherheitsmoduls festgelegt und später nicht mehr veränderbar. Die Kodierung wird wie in Abbildung 8 gezeigt für PIN-Dateien implementiert. Hierbei werden die Zugriffsrechte für SEID 1 und 2 in derselben Variable als verschiedene Aktionen verwaltet. Die Zugriffsbeschränkungen für Schlüsselob-

jekte werden wie in Abbildung 7 dargestellt kodiert. Hierbei werden zwei verschiedene Variablen, jeweils eine pro SEID für die Verwaltung eingesetzt.



Bit-wert	Bedeutung
00	keine Beschränkung
01	nie erlaubt
10	PACE Authentifizierung erforderlich
11	PACE und externe Authentifizierung erforderlich

Abbildung 7: Zugriffsbeschränkungen auf Schlüsselobjekte



Bit-wert	Bedeutung
00	keine Beschränkung
01	nie erlaubt
10	PACE Authentifizierung erforderlich
11	PACE und externe Authentifizierung erforderlich

Abbildung 8: Zugriffsbeschränkungen auf PIN-Dateien

Die Zugriffsbeschränkungen für die Kommandos sind im Applet festgelegt und nicht änderbar. Sie ergänzen die Beschränkungen auf Dateiebene um weitere Bedingungen, aber verringern diese in keinem Fall. Für die meisten Befehle gilt, dass sie nur nach Erreichen des *PACE* und *AUTH* Sicherheitszustandes genutzt werden können. Einige Befehle sind jedoch immer nutzbar, für einige ist nur *PACE* notwendig, eine dritte Gruppe von Befehlen ist nur durch dateispezifische Richtlinien beschränkt. Die Zugriffsbeschränkungen auf Befehlsebene sind in Tabelle 4 aufgelistet.

Tabelle 4: Zugriffsbeschränkungen auf Befehlsebene

immer möglich	**nur *PACE* erforderlich**	**nur dateispezifsche Beschränkungen**
GENERAL AUTHENTICATE (PACE), MSE SET, MSE RESTORE, SELECT FILE	GENERAL AUTHENTICATE (KEY EXCHANGE), GENERATE KEYPAIR (EXPORT PUBLIC KEY), EXTERNAL AUTHENTICATE, INTERNAL AUTHENTICATE, COMPUTE SIGNATURE, VERIFY SIGNATURE, GET CHALLENGE, CHANGE PIN REFERENCE DATA (CHANGE)	READ EF, UPDATE EF, APPEND RECORD
nicht aufgeführte Befehle: PACE und AUTH Sicherheitszustand erforderlich		

5.7 Kommunikation mit Sicherheitsmodul

Die Kommunikation mit dem Sicherheitsmodul erfolgt nach dem ISO-Standard 7816. Abschnitt 3 der Norm beschreibt hierbei die Transportschicht. Auf eine Beschreibung dieser soll hier nicht weiter eingegangen werden, da deren Umsetzung in den Treibern des Kartenlesers, bzw. vom Betriebssystem der Smartcard erfolgt. Es ist also unerheblich, ob diese nach dem byteorientierten T0 oder dem blockorientierten T1 abläuft. Auch eine Kommunikation über USB nach ISO7816-12 ist möglich, falls diese von der entsprechenden Smartcard unterstützt wird.

5.8 Kodierung der Datenpakete - APDU

Die Kommunikation auf der Anwendungsschicht (OSI-Schicht 7) wird vom Standard ISO7816-4 beschrieben. Hierzu werden sogenannte APDUs (Application Protocol Data Unit) eingesetzt. Die Kommunikation mit der Smartcard wird dabei immer extern begonnen, d. h., die Smartcard kann nur auf Anfragen antworten.

5.8.1 Kommando APDUs

Die an die Smartcard gesendeten APDUs werden als sogenannte Kommando-APDUs bezeichnet. Diese setzen sich aus einem Header und einem Body zusammen (siehe Abbildung 9).
Das Class Byte (CLA) wird zur Kennzeichnung des verwendeten Befehlssatzes und einer eventuell vorhandenen kryptografischen Sicherung der Daten (Verschlüsselung und Prüfsumme) verwendet.
Durch das folgende Instruction Byte (INS) wird das auszuführende Kommando übertragen. Ist eine Kompatibilität zum T0 Transportprotokoll gewünscht, dürfen nur geradzahlige Werte verwendet werden (nicht geradzahlige Instruktionskodierungen dienen der Aktivierung einer Programmierspannung [ISO7816-3]). Zum Header gehören weiters die Parameterbytes P1 und P2. Sie werden als Parameter für das auszuführende Kommando genutzt. Der Body der Kommando-APDUs dient zur

Datenübertragung und zur Übergabe von Längeninformationen für diese. Das LC-Feld beschreibt die Länge des zur Smartcard übertragenen Datenfeldes. Dieses wird vom Datenfeld selbst gefolgt. Als Letztes wird das LE-Feld angefügt, welches die Länge der Antwortdaten beschreibt.

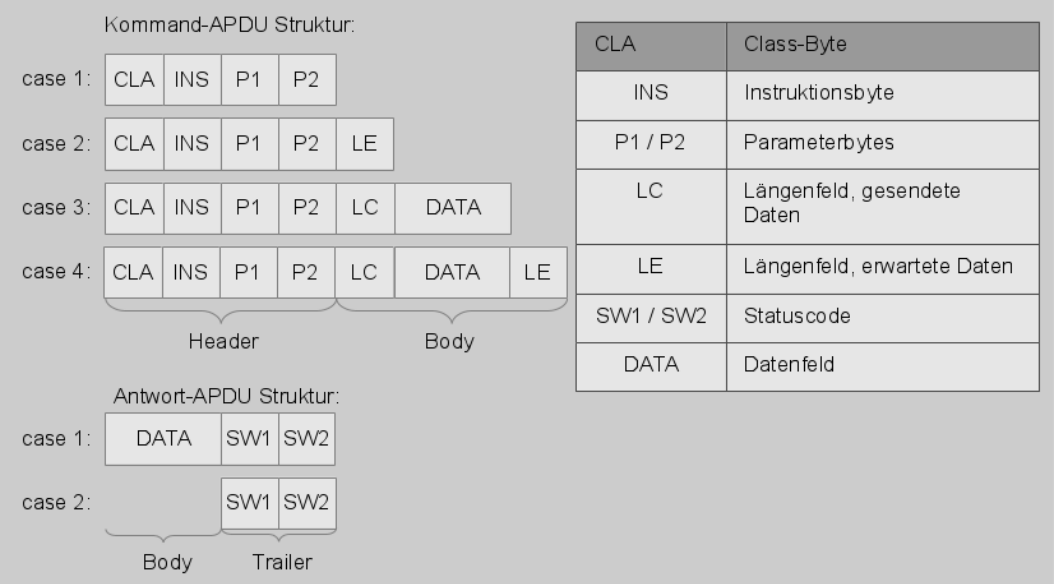


Abbildung 9: APDU Struktur und Klassifizierung

Die Kodierung der Längenangaben erfolgt für Werte zwischen 0 und 255 binär als 1 Byte langes Feld. Sind Daten mit größerer Länge zu übertragen, so wird auf die Kodierung als sogenannte *extended APDUs* zurückgegriffen. Es werden dann jeweils 2 Byte pro Längenangabe verwendet. Somit können Längenangaben mit Werten zwischen 1 und 65536 übertragen werden. Die erste auftretende Längenangabe wird jeweils durch ein drittes Signalisierungsbyte erweitert (dieses entspricht einer vorangestellten 0 und soll die Kodierung als *extended APDU* signalisieren).
Der Body der Kommando-APDU enthält immer nur die benötigten Felder. Soll kein Datenfeld an die Karte übertragen werden und werden auch keine Antwortdaten erwartet, so entfällt der Body. Die Kommando-APDU würde also nur aus dem Header bestehen. Je nach den im Body enthaltenen Feldern werden die Kommando-APDUs in Klassen eingeteilt (siehe Abbildung 9).

5.8.2 Antwort APDU

Auf jede Kommando-APDU muss die Smartcard mit einer Antwort-APDU antworten. Diese bestehen aus einem Body und einem Trailer. Wie bei den Kommando-APDUs ist der Body optional und enthält die nur bei Bedarf gesendeten Daten (siehe Abbildung 9). Der Trailer besteht aus einem 2 Byte langen Statuswort. Dieses dient zur Anzeige einer erfolgreichen Abarbeitung des Kommandos (Wert 0x9000) bzw. zur Anzeige eines aufgetretenen Fehlers. Bei den Fehlern kann es sich zum einen um nicht erfüllte Vorbedingungen (z. B. kein File selektiert), aber auch um Fehler bei der Abarbeitung des Kommandos handeln. Tabelle 5 enthält eine Auflistung der verwendeten Statuscodes.

Tabelle 5: Verwendete Statuscodes der Antwort-APDUs

Statuscode	Fehlerbedingung
0x9000	NoError
0x6283	FileDeactivated
0x6285	FileTerminated
0x6D00	InstructionNotSupported
0x6A82	FileNotFound
0x6982	SecurityStatusNotSatisfied, ObjectTerminated, KeyActivated, KeyInitialisation, ObjectDeactivated, ObjectActivated
0x6A8	DfNameExists
0x6a89	DuplicatedObject
0x6a90	SfiOutOfRange
0x6A84	OutOfMemory, FullEF
0x6581	MemoryFailure
0x6700	WrongLength
0x6986	NoCurrentFile, NoCurrentEF
0x6981	WrongFileType
0x6B00	OffsetTooBig
0x6A87	DataTooBig, WrongRecordLength, LongPassword, ShortPassword
0x6A83	RecordNotFound
0x6A80	Incorrect Parameters
0x6A88	KeyNotFound, PinNotFound, PasswordNotFound
0x6400	KeyInvalid, PinInvalid
0x6985	NoKeyReference, NoPinReference, NoRandom, WrongRandomLength
0x6A81	UnsupportedFunction

Die Kodierung der Fehlermeldung als Statuscodes entspricht soweit als möglich jenen des Schutzprofils. Einige Statuscodes sind hierdurch mehrfach belegt. Zur Interpretation aller von *NoError* (0x9000) abweichenden Antworten sollte die Spezifikation des auszuführenden Kommandos [BSI13-3] betrachtet werden.

5.8.3 Datenkodierung (BER/TLV)

Für den Austausch von Datenobjekten in APDUs werden diese kodiert. Hierfür wird die sogenannte BER/TLV (basic encoding rules/tag length value) Kodierung eingesetzt. Diese ist in dem Standard ISO/IEC 8825 [ISO/IEC8825] beschrieben. Die Kodierung ist Teil des umfassenderen ASN.1 (Abstract Syntax Notation One) Beschreibung von Datenobjekten, welcher aufgrund beschränkter Ressourcen bei Smartcards kaum in vollem Umfang umgesetzt wird [RANKL08].
Die Kodierung der Datenobjekte erfolgt in einer dreigeteilten TLV-Struktur. Auf einen ein bis zwei Byte langen Kennzeichner (*Tag*) folgt die ein bis drei Byte lange Längenangabe (*Length*) und der bis zu 65535 Byte lange Datenblock (*Value*). Der Tag beschreibt den Objekttyp des Datenfelds. Diese sind unter anderem im ISO 7816 Standard für viel benötigte Anwendungsfälle spezifiziert. Es werden hierbei die ersten drei Bit des Tags zur Kodierung einer Anwendungsklasse verwendet. Die fol-

genden fünf Bit dienen als Kennzeichen des Datentyps in der jeweiligen Klasse bzw. für den Fall 0b11111 zum Hinweis auf ein zweites Tag Byte, welches den Datentyp angibt.
Für das Sicherheitsmodul wurden bei der Wahl der Tags, soweit möglich jene in dem Schutzprofil [BSI13-2] und der Technical Guideline TR-03110-3 [BSI13-4] genutzten gewählt. Das Längenfeld gibt die Länge des Datenfelds an, welches die eigentlichen Nutzdaten enthält. Die Kodierung des Längenfelds wird in den DER (Distinguished Encoding Rules), welche auch Teil der BER sind, spezifiziert. Hierbei wird bis zu einer Länge von 127 nur ein Byte verwendet. Für größere Längen werden zwei bis drei Byte eingesetzt und das erste Byte als Kennzeichner für die nachfolgenden verwendet. Diese Art der Längenkodierung soll noch einmal in Tabelle 6 dargestellt werden.

Tabelle 6: DER-Längenkodierung (nach [RANKL08] Tabelle 4.6)

Byte 1	Byte 2	Byte 3	Wertebereich
Längenangabe (0 … 127)	nicht vorhanden	nicht vorhanden	0 bis 127
0x81	Längenangabe (0 … 255)	nicht vorhanden	128 bis 255
0x82	Längenangabe (256 … 65535)		256 bis 65535

Die Kodierung der eigentlichen Nutzdaten im Datenfeld ist je nach Objekttyp unterschiedlich, aber für jeden Tag konsistent. Diese entsprechen der in der Technical Guideline TR-03110-3 [BSI13-4] (speziell Abschnitt D) beschriebenen Kodierung. Für die Nutzung mit komplexeren Kommandos können die BER/TLV-Datenobjekte auch ineinander verschachtelt sein. Dies wird z. B. bei der Übertragung von Zertifikatsdaten an das Sicherheitsmodul genutzt. Dabei wird eine Reihe von primitiven BER/TLV-Datenobjekten als Datenfeld eines konstruierten BER/TLV-Objekts verwendet (Tabelle 7).

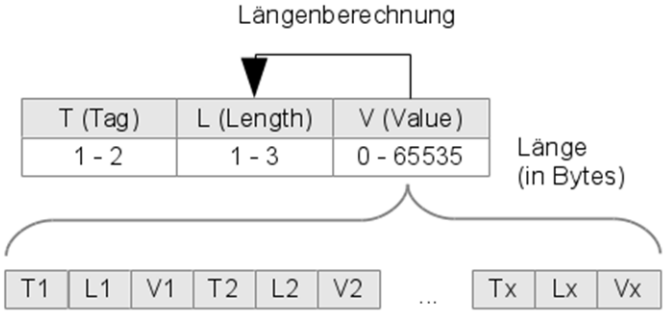


Abbildung 10: BER/TLV Kodierung von Datenobjekten und deren Verschachtelung

Für die Auswertung der kodierten Datenobjekte in einer APDU wird von einer vordefinierten, festen Reihenfolge der einzelnen BER/TLV-Objekte ausgegangen. Dies kann wiederum durch die eingeschränkten Ressourcen des Parsers (*Java Card* Applet) gerechtfertigt werden. Es wurden jedoch auch Methoden für das Auffinden von BER/TLV-Objekten in einem APDU-Datenfeld implementiert, welche für die Implementierung von komplexeren Kommandos genutzt wurden.

5.9 Secure Messaging

Um die Kommunikation zwischen dem Sicherheitsmodul und dem Gateway zu sichern, wird diese mithilfe kryptografischer Verfahren vor Änderung und Mitlesen geschützt. Dies wird im Umfeld von Smartcards als Secure Messaging bezeichnet [RANKL08]. Es wird hierbei das in [BSI13-4] Technical Guideline TR-03110-3 beschriebene Verfahren verwendet. Dabei wird der Body der APDUs verschlüsselt und über die gesamte APDU eine kryptografische Prüfsumme gebildet. Für die Verschlüsselung wird wie vom Sicherheitsprofil [BSI13-2] vorgesehen *CBC* (Cipher Block Chaining Mode) mit einer 128-Bit-AES (Advanced Encryption Standard) Blockchiffre verwendet. Für die Prüfsumme wird ein *MAC* (Message Authentication Code) mit *DES* (Data Encryption Standard) verwendet.
Die *PACE*-Authentifizierung (siehe Abschnitt 5.10) dient als Start für jede Secure-Messaging-Sitzung. Der Abbruch einer Secure-Messaging-Sitzung wird durch das Senden einer nicht geschützten APDU veranlasst. Dies führt weiters zum Selektieren des Master File sowie zum Zurücksetzen des Authentifizierungsstatus-*PACE*, sowie des Authentifizierungsstatus *AUTH*. Für auftretende Fehler beim Secure Messaging werden neue Statuscodes eingeführt:

- 0x6987, falls ein erwartetes Secure-Messaging-Objekt nicht empfangen wurde.
- 0x6988, falls eines der Secure-Messaging-Objekte fehlerhaft ist.

Für das Secure Messaging wird der Body in seine drei Bestandteile LC, DATA und LE aufgeteilt. Nach der Verschlüsselung des DATA-Felds werden dieses und das LE-Feld BER/TLV-kodiert (siehe Abschnitt 5.8.3) und wieder zusammengefügt. Das LC-Feld wird nicht weiter benötigt, da die Länge des DATA-Felds bereits in der Kodierung enthalten ist. Die verwendeten Tags werden in Tabelle 7 beschrieben.

Tabelle 7: Tags der Secure-Messaging-Objekte

Tag	Beschreibung
0x87	verschlüsseltes Datenfeld (variable Länge)
0x97	geschütztes LE-Feld (1-2 Byte)
0x99	geschützter Statuscode (2 Byte)
0x8E	kryptografische Prüfsumme (8 Byte)

Die Prüfsumme wird über den Header, den BER/TLV-kodierten Objekten des Body und einen zusätzlichen Sequenzzähler (*SSC* – Send Sequence Counter) gebildet. Diese wird wiederum BER/TLV-kodiert und bildet mit den anderen Secure-Messaging-Objekten das neue Datenfeld des Body. Der genaue Ablauf wird in Abbildung 11 für *case 4* APDUs gezeigt. Für die *cases* 1 bis 3 werden die entsprechend entfallenden Objekte weggelassen. Die Bildung der Antwort APDUs erfolgt ähnlich und wird in Abbildung 12 gezeigt. Um die Sicherheit zu erhöhen und Replay-Attacken zu erschweren, wird ein SSC (Send Sequence Counter) verwendet. Dieser Zähler hat die Länge der verwendeten Blocklänge (128 Bit) und wird für jede gesicherte APDU um eins erhöht. Er wird weiters als IV (Initialisierungsvektor) für die AES-Verschlüsselung verwendet. Wie bereits geschrieben wird er auch als erster Block für die Prüfsummenbildung verwendet.

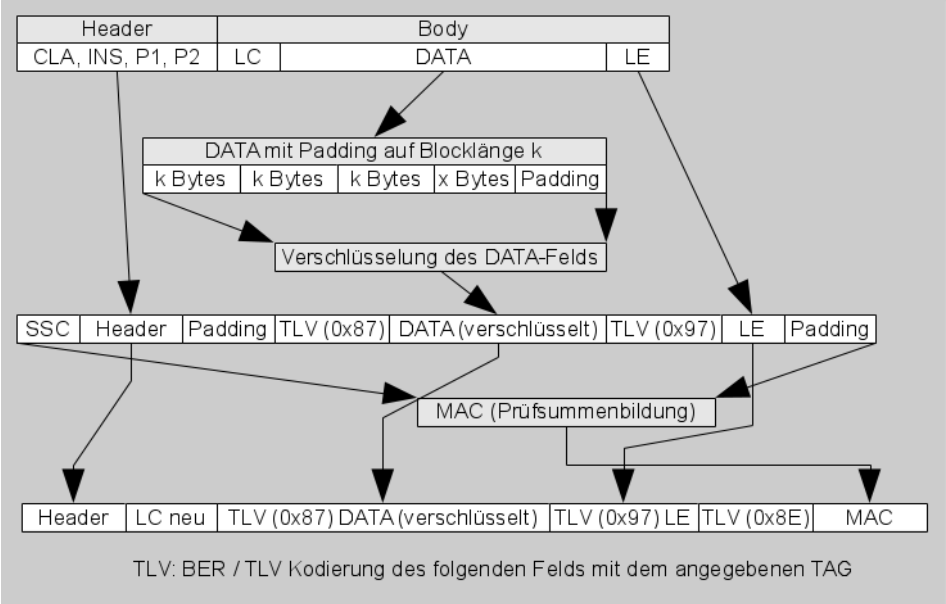


Abbildung 11: Secure Messaging, Generierung Kommando APDU

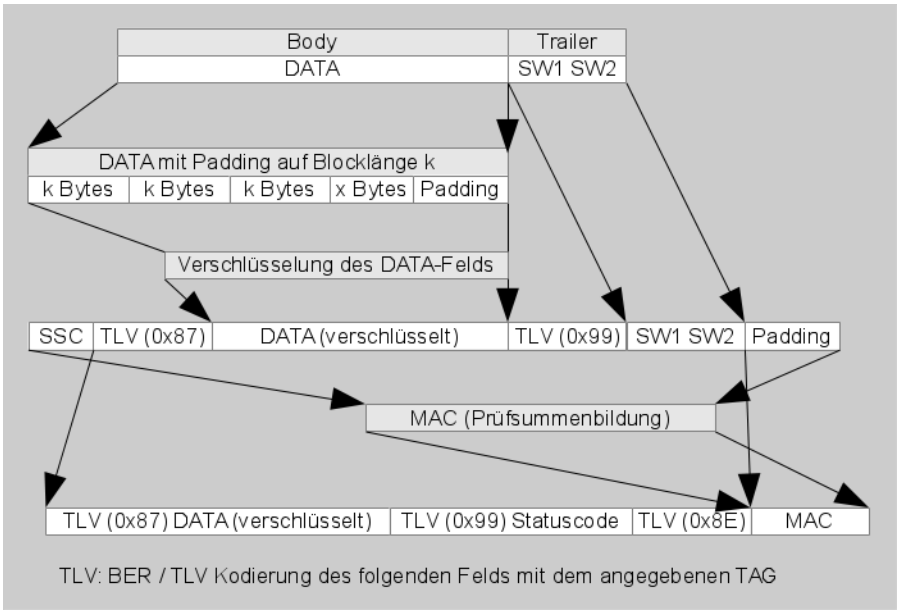


Abbildung 12: Secure Messaging, Generierung Antwort APDU

Die Schlüsselvereinbarung für das Secure Messaging wird im Rahmen der *PACE*-Authentifizierung durchgeführt. Da sowohl der CBC als auch der MAC eine Blocklänge von 128 Bit erwartet, müssen teilweise Daten mit einem Padding versehen werden. Hierzu wird ein Padding-Indikator (0x80) angefügt und der restliche Block mit Bytes des Werts 0x00 aufgefüllt.
Für die Verarbeitung von gesicherten APDUs wurden im Applet die *preProcess*- und *postProcess*-Methoden implementiert. Diese werden nur bei aktiven Secure-Messaging-Sitzungen aufgerufen und

dienen dazu gesicherte in ungesicherte Kommando-APDUs umzuwandeln und ungesicherte Antwort-APDUs in gesicherte. Diese explizite Umwandlung der Secure Messaging APDUs außerhalb der eigentlichen Verarbeitungsroutinen hat einen höheren Speicherverbrauch (durch zusätzliche Buffer) zur Folge. Es ermöglicht jedoch den Einsatz derselben Befehlsverarbeitungsmethoden für gesicherte und ungesicherte APDUs, wodurch Entwicklung, Test und Evaluierung vereinfacht wurden.

5.10 *PACE*

Damit der Betreiber des Gateways auf Administrationsfunktionen des Sicherheitsmoduls zugreifen kann, muss sich dieser authentifizieren. Hierfür wird eine Abwandlung des sogenannten *PACE*-Protokolls (Password Authenticated Connection Establishment) eingesetzt. Das *PACE*-Protokoll selbst wurde vom BSI für den Einsatz im elektronischen Personalausweis der Bundesrepublik Deutschland entwickelt.
Bei dem Einsatz im Sicherheitsmodul dient es zur Authentifizierung und dem Setzen des entsprechenden Status im Security Environment. Zusätzlich wird mit dem *PACE*-Protokoll auch eine Sitzung für die gesicherte Kommunikation zwischen Sicherheitsmodul und Gateway (Secure Messaging) gestartet. Auch die Schlüsselvereinbarung hierfür erfolgt mittels des *PACE*-Protokolls. Die zugrunde liegende Spezifikation ist in der Technischen Richtlinie TR-03110 [BSI 13-3] enthalten. Aufgrund von Einschränkungen der eingesetzten Smartcard wurde das Protokoll in etwas abgewandelter Form implementiert. Im speziellen fehlt der Smartcard die Unterstützung der Funktionen zur Addition von Punkten auf elliptischen Kurven und für den Schlüsseltausch nach Diffie-Hellman ohne Schlüsselableitung (siehe auch Abschnitt 6.1.5).
Für das *PACE*-Protokoll wird ein beiden Partnern bekanntes Passwort benötigt. Im Rahmen dieser Implementierung wird dieses als PIN bezeichnet und in entsprechenden PIN-Dateien gespeichert (siehe Abschnitt 5.3.5).
Die implementierte Variante des *PACE*-Protokolls besteht aus folgenden Schritten:

- Das Gateway sendet das Kommando zum Setzen des Security Environments für die *PACE*-Authentifizierung. Hierbei wird die zu verwendende PIN-Datei über deren PIN-ID ausgewählt und die Variante des *PACE*-Protokolls (Schlüssellänge) über die OID gewählt.
- Das Gateway sendet ein Kommando zum Starten des *PACE*-Protokolls. Das Sicherheitsmodul generiert hierauf eine Zufallszahl (Länge entsprechend der gesetzten Protokollvariante im Security Environment). Diese wird mit AES im CBC-Modus verschlüsselt, wofür ein vom PIN abgeleiteter Schlüssel verwendet wird. Die Schlüsselableitung erfolgt durch eine XOR–Verknüpfung mit einer festen Bytesequenz. Die verschlüsselte Zufallszahl wird dann an das Gateway gesendet.
- Die empfangene Zufallszahl wird vom Gateway entschlüsselt und dient als Grundlage für die Schlüssel des Secure Messaging. Die Schlüsselableitung erfolgt wiederum durch eine XOR–Verknüpfung der Zufallszahl mit zwei verschiedenen fixen Byte-Sequenzen. Hierdurch werden zwei getrennte Schlüssel für Verschlüsselung und für die Prüfsumme gebildet.
- Um die Authentifizierung abzuschließen, wird die Zufallszahl nochmals verschlüsselt, wofür der Secure-Messaging-Schlüssel verwendet wird. Diese wird an das Sicherheitsmodul ge-

sendet. Falls die Zufallszahl nach dem Entschlüsseln übereinstimmt, wird der *PACE*-Status im Security Environment gesetzt.

- Die gesamte Kommunikation nach diesem Zeitpunkt erfolgt gesichert durch Secure Messaging.

Um die Authentifizierung aufzuheben, muss das Master File im Dateisystem selektiert werden. Dies beendet auch das Secure Messaging.

5.11 Implementierte Kommandos

Die für die Kommunikation mit dem Sicherheitsmodul implementierten Kommandos werden im Folgenden aufgelistet und beschrieben. Diese orientieren sich, soweit spezifiziert, an dem BSI Technical Report für das Security Module [BSI13-2]. Die Kommandos selbst werden nur in Ihrer ungesicherten Form, d. h., bevor diese durch ein eventuelles Secure Messaging geschützt wurden, beschrieben. Voraussetzung für die erfolgreiche Ausführung der Kommandos ist jeweils die Erfüllung der Zugriffsbedingungen (siehe Abschnitt 5.6). Zusätzlich werden noch die dazugehörigen Methoden der Client-Bibliothek angegeben, mit denen das Kommando ausgeführt werden kann. Die Erklärung des Instruktionssatzes kann als Beschreibung der API des Sicherheitsmoduls und der Client-Bibliothek betrachtet werden.

5.11.1 Management des Dateisystem

Unter das Management des Dateisystem fallen alle Kommandos die zum Management der nicht kryptografischen Dateien (MF, DFs und EFs) dienen.

5.11.1.1 SELECT Kommando

Mit diesem Kommando wird eine Datei (*MF*, *DF* oder *EF*) selektiert (siehe Tabelle 8). Die Attribute der selektierten Datei können im darauf folgenden Kommando bearbeitet werden. Sollte ein *DF* selektiert worden sein, so wird dieses als Ausgangspunkt für folgende Kommandos mit impliziter Adressierung (*SFI* für *EFs*, ID für Schlüssel und PINs) verwendet. Die Selektion des *MF* hat zudem Einfluss auf das Security Environment. Es werden die Sicherheitszustände *PACE-Authentisierung erfolgt* und *externe Authentisierung erfolgt* zurückgesetzt. Wurde ein *EF* selektiert, so können im Folgenden lesend oder schreibend darauf zugegriffen werden.

Tabelle 8: Kodierung des SELECT Kommandos

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xA4	0x00	0x0C	DATALength	0x3F00	-	Selektierung des MF
0x00	0xA4	0x01	0x0C	DATALength	FID	-	Selektierung eines DF (FID)
0x00	0xA4	0x02	0x0C	DATALength	FID	-	Selektierung eines EF (FID)
0x00	0xA4	0x04	0x0C	DATALength	AID	-	Selektierung eines DF (AID)

Auf das SELECT-Kommando kann von der Client-Bibliothek über die Methoden *selectMF*, *selectEF* und *selectDF* zugegriffen werden.

5.11.1.2 CREATE FILE Kommando

Das Kommando dient zum Anlegen von *DFs* und sowohl transparenten als auch Record-orientierten *EFs*. Für die Ausführung des Kommandos muss zuerst ein *DF*, oder das *MF* selektiert werden. In diesem wird die neue Datei erstellt. Bei der Erstellung der Datei werden auch gleichzeitig die Zugriffsrechte gesetzt. Diese werden entsprechend Abschnitt 5.6 bitkodiert als zwei Bytes übertragen und im Folgenden als *RULES* bezeichnet. Zusätzlich müssen beim Aufruf des Kommandos noch die IDs der zu erstellenden Datei (*FID* und *SFI* für *EFs*, *FID* und *AID* für *DFs*) angegeben werden (siehe Tabelle 9). Auf das CREATE FILE Kommando kann von der Client-Bibliothek über die Methoden *createFile*, *createFileDF* und *createFileEf* zugegriffen werden. Für die Erstellung der bitkodierten Zugriffsregeln RULES kann das Enum *rules* und deren Methode *makeRules* zur Hilfe genommen werden (siehe Abschnitt 5.12.1).

Tabelle 9: Kodierung des CREATE FILE Kommandos

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xE0	0x00	0x00	DATALen	0x02 FID RULES AID	-	DF anlegen
0x00	0xE0	0x00	0x00	DATALen	0x04 FID RULES SFI	-	transparentes EF anlegen
0x00	0xE0	0x00	0x00	DATALen	0x05 FID RULES SFI	-	Record EF anlegen

5.11.1.3 Weitere Kommandos zum Dateimanagement

Für das Löschen von Dateien, sowie zur Änderung des Lebenszyklus von Dateien (aktiviert, deaktiviert, terminiert, siehe Abschnitt 5.3) existieren weitere Befehle. Die Kommandos werden jeweils auf die zuletzt selektierte Datei angewandt. Diese werden nur bei Erfüllung der in den Zugriffsbeschränkungen der Dateien vorgegebenen Voraussetzungen ausgeführt. Die Kodierung dieser Befehle wird in Tabelle 10 dargestellt. Für den Zugriff aus der Client-Bibliothek dienen die Methoden *deleteFile*, *activateFile*, *deactivateFile*, *terminateEF* und *terminateDF*.

Tabelle 10: Kodierung der DELETE FILE, ACTIVATE FILE, DEACTIVATE FILE, TERMINATE EF und TERMINATE DF Kommandos

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xE4	0x00	0x00	-	-	-	Datei löschen
0x00	0x44	0x00	0x00	-	-	-	Datei aktivieren
0x00	0x04	0x00	0x00	-	-	-	Datei deaktivieren
0x00	0xE6	0x00	0x00	-	-	-	DF terminieren
0x00	0xE8	0x00	0x00	-	-	-	EF terminieren

5.11.2 Zugriff auf EF Dateien

Zur Speicherung von Daten im Dateisystem werden *EF* Dateien verwendet. In diese können Rohdaten wie Zertifikate geschrieben werden, die nicht vom Sicherheitsmodul selbst interpretiert werden müssen. Der Zugriff auf diese Datenfelder erfolgt mit den Kommandos READ BINARY und UPDATE BINARY für transparente EFs und mit READ RECORD, UPDATE RECORD, sowie APPEND RECORD für Record-orientierte EFs. Für die Ausführung der Kommandos müssen Zugriffsbeschränkungen der jeweiligen Datei beachtet werden. Der Zugriff erfolgt jeweils auf die zuletzt selektierte Datei, oder im Falle eines selektierten DF, auf das im Kommando per SFI referenzierte EF.

5.11.2.1 READ BINARY und UPDATE BINARY Kommandos

Die Kommandos zum Schreiben und Lesen von transparenten EFs erlauben den Zugriff über einen Offset und die gewünschte Länge der Daten. Für den Fall eines Zugriffs über die Dateiselektierung ist ein Offset zwischen 0x0000 und 0x7F00 angebbar, falls der Zugriff über eine SFI-Referenz erfolgt, ist nur ein Offset zwischen 0x00 und 0xFF möglich. Die Zugriffsbeschränkungen werden beim Aufruf der Kommandos beachtet. Über die Client-Bibliothek kann mit den Methoden *readBinary* und *updateBinary* zugegriffen werden. Die Kodierung der für die Kommandos zu verwendenden APDUs ist in Tabelle 11 angegeben.

Tabelle 11: Kodierung der READ BINARY und UPDATE BINARY Kommandos

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xB0	Offset		-	-	ReadLen	aus selektierter Datei lesen
0x00	0xB0	0x80 + SFI	Offset	-	-	ReadLen	Daten lesen, SFI-Referenz
0x00	0xD6	Offset		DATALen	Daten	-	in selektierte Datei schreiben
0x00	0xD6	0x80 + SFI	Offset	DATALen	Daten	-	Daten schreiben, SFI-Referenz

5.11.2.2 READ RECORD, UPDATE RECORD und APPEND RECORD Kommandos

Die Kommandos zum lesenden und schreibenden Zugriff auf die Felder von Record-basierten EFs sind READ RECORD und UPDATE RECORD. Weiters existiert ein Kommando zum Hinzufügen eines neuen Datenfelds, APEND RECORD. Die Steuerung der zu bearbeitenden Daten erfolgt über die Angabe des Datenfelds (Record ID, RID) und die Länge der Daten (siehe Tabelle 12). Die Ausführung der Kommandos ist von den Zugriffsbeschränkungen abhängig. Die Methoden der Client-Bibliothek für die Ausführung dieser Kommandos sind *readRecord*, *updateRecord* und *appendRecord*. Tabelle 12 beschreibt die Kodierung der Kommando-APDUs.

Tabelle 12: Kodierung der READ RECORD, UPDATE RECORD und APPEND RECORD (SFI<<3 entspricht einer dreimal bitweise nach links geshifteten SFI)

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xB2	RID	0x04	-	-	ReadLen	Record lesen, selektierte Datei
0x00	0xB2	RID	SFI<<3 +0x04	-	-	ReadLen	Record lesen, SFI-Referenz
0x00	0xDC	RID	0x04	DATALen	Daten	-	Record schreiben, selektierte Datei
0x00	0xDC	RID	SFI<<3 +0x04	DATALen	Daten	–	Record schreiben, SFI-Referenz
0x00	0xE2	0x00	0x00	-	-	–	Record anfügen, selektierte Datei
0x00	0xE2	0x00	SFI<<3	-	-	–	Record anfügen, SFI-Referenz

5.11.3 Management von Schlüsseldateien

Für das Anlegen von neuen, für das Löschen von vorhandenen, sowie zum Management des Lebenszyklusstatus von Schlüsseldateien sind die Kommandos CREATE KEY, DELETE KEY, ACTIVATE KEY und DEACTIVATE KEY vorhanden. Diese sind in Tabelle 13, Tabelle 14 und Tabelle 15 beschrieben. Das Schlüsselobjekt, auf das der Befehl angewendet wird, wird dabei durch die Key-ID referenziert und im aktuell selektierten DF oder MF gesucht. Bei der Ausführung der Kommandos werden die Zugriffsbeschränkungen für Schlüsselobjekte (Abschnitt 5.6) ausgewertet. Kommandos zum Schreiben von Schlüsselpaaren existieren nicht, da dies die Vertraulichkeit des privaten Schlüssels verletzen würde. Schlüsselpaare müssen immer im Sicherheitsmodul generiert werden. Auch das Auslesen von öffentlichen Schlüsseln für Schlüsselpaare wird mit demselben Kommando unterstützt (siehe 5.11.5.1 GENERATE ASYMMETRIC KEYPAIR). Für Schlüsselobjekte mit nur öffentlichen Schlüsseln besteht die Möglichkeit diese mit dem Befehl PSO VERIFY CERTIFICATE (Abschnitt 5.11.5.4) zu füllen. Das Auslesen dieser öffentlichen Schlüssel wird nicht unterstützt, da diese der externen Welt bereits bekannt sind.

Tabelle 13: Kodierung der Managementkommandos für Schlüsseldateien: CREATE KEY, DELETE KEY, ACTIVATE KEY und DEACTIVATE KEY

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xE0	0x21	0x00	DATALen	CreateKey-Template	-	Schlüsseldatei anlegen
0x00	0xE4	0x21	0x00	DATALen	KeyID-Template	-	Schlüsseldatei löschen
0x00	0x44	0x21	0x00	DATALen	KeyID-Template	-	Schlüsseldatei aktivieren
0x00	0x04	0x21	0x00	DATALen	KeyID-Template	-	Schlüsseldatei deaktivieren

5.11.3.1 CREATE KEY Kommando

Das Kommando zum Anlegen eines neuen Schlüsselobjekts ist CREATE KEY (Methode der Client-Bibliothek: *createKey*). Hierbei wird ein leeres Schlüsselobjekt mit dem Lebenszyklusstatus *Initialisierung* im Dateisystem erstellt. Dieses wird durch das Füllen mit Schlüsseldaten (GENERATE ASYMMETRIC KEY PAIR für Schlüsselpaare, bzw. PSO VERIFY CERTIFICATE für öffentliche Schlüssel) auf *aktiviert* gesetzt. Für die Ausführung des Kommandos ist die Angabe der Key-ID, des

Schlüsseltyps (Paar oder öffentlicher Schlüssel), des Verwendungszwecks (Signatur oder Authentifizierung), die damit verwendbaren Algorithmen und, optional im Falle eines Schlüsselpaars, der Wert für den Verwendungszähler notwendig. Wird dieser nicht angegeben, kann das Schlüsselpaar unbegrenzt genutzt werden. Die in Kapitel 5.6 beschriebenen Zugriffsbeschränkungen können nur für Dateien die bereits während der Initialisierung des Sicherheitsmoduls angelegt werden gesetzt werden. Für die mit diesem Kommando angelegten Schlüsselobjekte gelten nur die in Tabelle 4 angegebenen Zugriffsbeschränkungen auf Befehlsebene. Die Kodierung der Kommando-APDUs ist in Tabelle 13 angegeben, die BER/TLV-Kodierung der Kommando-Parameter (*CreateKey-Template*) ist in Tabelle 14 gezeigt.

Tabelle 14: BER/TLV-Kodierung des CreateKey-Template zum Anlegen von Schlüsseldateien (hexadezimal,T_{xx}: BER/TLV-Tag mit dem Wert xx, L_{yy}: Längenangabe des BER/TLV Objekts mit Tag yy)

BER/TLV-Objekt	Beschreibung
T_{62}-L_{62}-(	BER/TLV Rahmen
T_{AD}-L_{AD}-(	BER/TLV Rahmen
T_{83}-L_{83}-KEYID	ID des Schlüsselobjekts
T_{96}-L_{96}-Counter	Verwendungszähler (nur für Schlüsselpaare, optional)
T_{A1}-00 oder T_{A2}-00	TA1 falls Schlüsselpaar, TA2 falls öffentlicher Schlüssel
T_{7B}-L_{7B}-(	BER/TLV Rahmen
T_{80}-01-SEID	ID des Security Environments, in dem der Schlüssel verwendet werden kann (evtl. mehrere)
T_{A4}-L_{A4}-(oder T_{B6}-L_{B6}-(	TA4 falls Schlüssel für Authentifizierung, TB6 falls Schlüssel für Signaturen
T_{80}-L_{80}-CrytoID))))	OID des verwendbaren Algorithmus

5.11.3.2 DELETE KEY, ACTIVATE KEY und DEACTIVATE KEY Kommandos

Für das Löschen von Schlüsselobjekten dient der Befehl DELETE KEY, zum Setzen des Lebenszyklus auf aktiv oder inaktiv die Befehle ACTIVATE KEY bzw. DEACTIVATE KEY. Eine Möglichkeit den Lebenszyklusstatus auf „terminiert“ zu setzten existiert für Schlüsseldateien nicht. Für die Selektierung der zu bearbeitenden Datei sind neben der Key-ID zusätzlich der Schlüsseltyp (Paar oder öffentlicher Schlüssel) sowie der Verwendungszweck des Schlüssels (Signatur oder Authentifizierung) anzugeben. Die Ausführung dieser Befehle von der Client-Bibliothek erfolgt durch die Methoden *deleteKey*, *activateKey* und *deactivateKey*. Die Kodierung der Kommandos ist in Tabelle 13 dargestellt. Für die Angabe der Key-ID und der Dateiattribute werden diese BER/TLV kodiert übertragen (siehe Tabelle 15).

Tabelle 15: BER-TLV Kodierung des KeyID-Template (hexadezimale Werte,T_{xx}: BER/TLV-Tag mit dem Wert xx, L_{yy}: Längenangabe des BER/TLV Objekts mit Tag yy)

	Key Pair	Public Key
Authenticate	T_{A4}-L_{A4}-(T_{84}-L_{84}-KeyID)	T_{A4}-L_{A4}-(T_{83}-L_{83}-KeyID)
Signature	T_{B6}-L_{B6}-(T_{84}-L_{84}-KeyID)	T_{B6}-L_{B6}-(T_{83}-L_{83}-KeyID)

5.11.4 Kommandos zum Management des Security Environment (MSE)

Das Security Environment dient zur Verwaltung des Lebenszyklus des Sicherheitsmoduls, der aktuellen Betriebsart (SEID), dem Status der Authentifizierungen, sowie den Informationen für nachfolgende kryptografische Kommandos. Zum Setzen der SEID ist das Kommando MSE RESTORE vorgesehen. Es kann von der Client-Bibliothek mit Hilfe der Methode *mseRestore* ausgelöst werden. Die Kodierung der Kommando-APDU ist in Tabelle 16 angegeben.

Tabelle 16: Kodierung des MSE RESTORE Kommandos zum Setzen der SEID

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0x22	0xF3	SEID	-	-	-	Ändern der SEID auf die in P2 übergebene

Für das Ausführen der in Tabelle 17 angegebenen kryptografischen Kommandos ist die vorherige Übergabe von weiteren Daten an das Sicherheitsmodul notwendig. Diese werden mit dem Kommando MSE SET übertragen und im Security Environment gespeichert. Die zu übergebenden Informationen sind dabei von dem danach auszuführenden Kommando abhängig. Diese sind ebenfalls in Tabelle 17 aufgelistet. Werden mit dem Kommando Key-IDs als Referenzen auf Schlüsselobjekte übergeben, so wird deren Existenz, und ob sie für das nachfolgende kryptografische Kommando geeignet ist sofort geprüft. Sollte eine der Prüfungen negativ ausfallen, kann das anschließende kryptografische Kommando nicht ausgeführt werden. Für das Setzen dieser Informationen mithilfe der Client-Bibliothek dienen die Methoden *mseSet*, *mseSetAtKeyAgreementEckaEgRec*, *mseSetAtKeyAgreementEckaEgInit*, *mseSetAtKeyAgreementEckaDh*,*mseSetDstGenerate*, *mseSetDstVerify*, *mseSetAtExt*, *mseSetAtInt* und *mseSetAtPace*. Die Kodierung der Kommandos entspricht jener, die von der Technischen Richtlinie [BSI 13-3] in den Tabellen 117 bis 134 beschrieben wird.

Tabelle 17: Im Rahmen des MSE SET Kommandos zu übertragende Informationen

Kommando	im Security Environment zu setzende Daten
PSO COMPUTE DIGITAL SIGNATURE, INTERNAL AUTHENTICATE	Key-ID des zu verwendenden Schlüsselpaars
PSO VERIFY DIGITAL SIGNATURE, PSO VERIFY CERTIFICATE, EXTERNAL AUTHENTICATE	zu verwendender öffentlicher Schlüssel (Key-ID) nicht nötig für PSO VERIFY DIGITAL SIGNATURE Variante mit öffentlichem Schlüssel in Kommandodaten
GENERAL AUTHENTICATE (PACE)	zu verwendendes PACE-Protokoll (OID), PIN-Datei (Pin-ID), Domain Parameter der elliptischen Kurve (ID)
GENERAL AUTHENTICATE (Key Agreement Diffie-Hellman)	zu verwendende Protokollvariante (OID)
GENERAL AUTHENTICATE (Key Agreement ElGamal als Recipient)	zu verwendende Protokollvariante (OID), Schlüsselpaar (Key-ID)
GENERAL AUTHENTICATE (Key Agreement ElGamal als Initiator)	zu verwendende Protokollvariante (OID), öffentlicher Schlüssel (Key-ID)

5.11.5 Kryptografische Kommandos

Das zentrale Element des Sicherheitsmoduls sind die Dienste zum Ausführen von kryptografischen Funktionen. Die Kommandos die hierzu verwendet werden sind GENERATE ASYMMETRIC KEY PAIR, PSO COMPUTE DIGITAL SIGNATURE, PSO VERIFY DIGITAL SIGNATURE, PSO VERIFY CERTIFICATE, EXTERNAL AUTHENICATE, INTERNAL AUTHENTICATE, sowie GENERAL AUTHENTICATE (in verschiedenen Varianten). Wird im Kommando ein intern gespeichertes Schlüsselobjekt referenziert, so wird vor dem Ausführen des Befehls die Eignung des referenzierten Schlüssels geprüft. Dies umfasst die Überprüfung des Vorhandenseins, des Lebenszyklusstatus, eines eventuell vorhandenen Verwendungszähler, den Verwendungszweck (Signatur oder Authentifizierung), den Schlüsseltyp (Paar oder öffentlicher Schlüssel) und die Zugriffsbeschränkungen. Für die Kodierung der Nachrichten-APDUs und der BER/TLV-Kodierung der Datenfelder sei hier auf die Tabellen 84 bis 116 der Technischen Richtlinie [BSI 13-3] verwiesen.

5.11.5.1 Kommando GENERATE ASYMMETRIC KEY PAIR

Das Kommando GENERATE ASYMMETRIC KEY PAIR dient der Generierung von Schlüsselpaaren und der Ausgabe der öffentlichen Schlüssel der Paare. Hierzu sind drei Varianten des Kommandos implementiert:

- Variante 1: Schlüsselgenerierung
- Variante 2: Schlüsselgenerierung mit Ausgabe des öffentlichen Schlüssels
- Variante 3: Ausgabe eines früher generierten öffentlichen Schlüssels

Für den Aufruf des Kommandos in den Varianten 1 und 2 müssen die Key-ID des zu generierenden Schlüsselpaars, der Verwendungszweck, sowie die OID der zu verwendenden elliptischen Kurve angegeben werden. Das Schlüsselobjekt selbst muss bereits existieren d. h. entweder mit dem Kommando CREATE KEY oder während der Initialisierung erstellt worden sein. Für die Generierung neuer Schlüsseldaten wird ein Lebenszyklusstatus *nicht-initialisiert* oder *deaktiviert* der Schlüsseldatei vorausgesetzt. Eventuell bereits vorhandene Schlüsseldaten werden hierbei überschrieben, der Lebenszyklusstatus wird im Falle einer korrekten Ausführung immer auf *aktiv* gesetzt. Die Varianten 1 und 2 unterscheiden sich dabei nur durch die Antwort, welche für Variante 1 nur den Statuscode, für Variante 2 die öffentlichen Schlüsseldaten (Kurvenpunkt) und die OID der dazugehörigen elliptischen Kurve enthält. Für die Ausgabe der öffentlichen Schlüsseldaten ohne Neugenerierung dient die Variante 3 des Kommandos. Hierbei sind nur die Key-ID und der Verwendungszweck des Schlüssels für die Referenzierung zu übergeben. Die Antwort entspricht jener der Variante 2 mit BER/TLV-kodiertem Kurvenpunkt und OID der elliptischen Kurve. Für die Ausführung dieses Kommandos durch die Client-Bibliothek können die Methoden *readPublicKeyFromAsymmetricKey* und *generateAsymmetricKey* verwendet werden.

5.11.5.2 Kommando PSO COMPUTE DIGITAL SIGNATURE

Das Sicherheitsmodul kann durch das Kommando PSO COMPUTE DIGITAL SIGNATURE (Methode *psoComputeDigitalSignature* der Client-Bibliothek) zur Erzeugung einer digitalen Signatur angewiesen werden. Der hierzu zu verwendende private Schlüssel muss zuvor durch ein MSE SET Kommando referenziert werden. Im Rahmen des Kommandos wird nicht die komplette zu signierende Nachricht an das Sicherheitsmodul gesendet, sondern nur der kryptografische Hash (SHA-1)

über die Nachricht. Die mittels des Elliptic Curve Digital Signature Algorithm (ECDSA) erstellte Signatur wird ohne weitere Kodierung oder Message Recovery an das Sicherheitsmodul zurückgegeben, wo diese weiterverwendet werden kann.

5.11.5.3 Kommando PSO VERIFY DIGITAL SIGNATURE

Zur Überprüfung von digitalen Signaturen nach dem ECDSA stellt das Sicherheitsmodul das Kommando PSO VERIFY DIGITAL SIGNATURE (Methode *psoVerifyDigitalSignature* der Client-Bibliothek) zur Verfügung. Beim Aufruf des Kommandos müssen hierbei immer der Hash über die zu verifizierende Nachricht sowie die zu verifizierende Signatur übergeben werden. Der zur Überprüfung verwendete Schlüssel kann entweder durch ein zuvor ausgeführtes MSE SET Kommando spezifiziert werden, oder direkt im PSO VERIFY DIGITAL SIGNATURE Kommando angegeben werden. Hierzu muss der Kurvenpunkt und die verwendete elliptische Kurve in Form ihrer OID angegeben werden. Die so übergebenen Schlüsseldaten werden zur Erstellung eines temporären Schlüsselobjekts verwendet. Die zu überprüfenden Signaturen sind ohne Message Recovery auszuführen.

5.11.5.4 Kommando PSO VERIFY CERTIFICATE

Für den permanenten Import von öffentlichen Schlüsseln und zur Überprüfung von X.509 Zertifikaten und Zertifikatsketten kann das Gateway auf das Kommando PSO VERIFY CERTIFICATE (*psoVerifyCertificate* Methode der Client-Bibliothek) zurückgreifen. Das Sicherheitsmodul überprüft dabei ein vom Gateway aufbereitetes Zertifikat. Ist dieses gültig, wird der darin enthaltene Schlüssel in ein ebenfalls im Zertifikat referenziertes Schlüsselobjekt importiert. Das Zertifikat selbst enthält also die Referenz auf das Schlüsselobjekt für den zu importierenden Schlüssel, die zu importierenden Schlüsseldaten (Kurvenpunkt und OID der elliptischen Kurve) sowie die Signatur über das Zertifikat. Der für die Überprüfung des Zertifikats zu verwendende Schlüssel muss zuvor über ein MSE SET Kommando gesetzt worden sein. Es wird das ECDSA Verfahren eingesetzt, und die zu überprüfenden Zertifikate müssen fremd signiert und *self descriptive* sein.

5.11.5.5 Kommando EXTERNAL AUTHENTICATE

Zur Authentifizierung des Gateways am Sicherheitsmodul dient das Kommando EXTERNAL AUTHENTICATE (*externalAuthenticate* Methode der Client-Bibliothek). Es erfordert den vorherigen Aufruf des MSE SET Kommandos sowie des GET CHALLENGE Kommandos mit interner Speicherung (siehe Abschnitt 5.11.6). Die so erhaltene Zufallszahl dient als Challenge für die Authentifizierung. Für sie wird eine Mindestlänge von 8 Byte vorausgesetzt. Die Challenge wird dazu von dem Gateway mit dem eigenen privaten Schlüssel für die externe Authentifizierung verschlüsselt und an das Sicherheitsmodul gesendet. Dieses nutzt den im Security Environment referenzierten öffentlichen Schlüssel zur Entschlüsselung. Ist diese erfolgreich, gilt das Gateway als authentifiziert und der interne Sicherheitszustand AUTH wird gesetzt.

5.11.5.6 Kommando INTERNAL AUTHENTICATE

Um das Sicherheitsmodul anzufordern die Authentifizierung am Gateway durchzuführen, wird das Kommando INTERNAL AUTHENTICATE verwendet (*internalAuthenticate* Methode der Client-

Bibliothek). Das Kommando kann weiters für generelle Authentifizierung des Gateways an anderen Stellen verwendet werden. Für die Ausführung des Kommandos muss zuerst ein MSE SET zur Festlegung des zu verwendenden Schlüsselpaars durchgeführt werden. Im Rahmen des Kommandos wird dann ein Token an das Sicherheitsmodul gesendet, welches eine Signatur darüber erstellt. Es wird hierfür ECDSA mit dem privaten Schlüssel des Schlüsselpaars eingesetzt. Diese wird dann an das Gateway zurückgegeben, welches sie selbst überprüfen (Authentifizierung des Sicherheitsmoduls), oder an die Gegenstelle der Authentifizierung weiterleiten kann.

5.11.5.7 Kommando GENERAL AUTHENTICATE (Variante *PACE*)

Die Authentifizierung eines Gatewayadministrators am Sicherheitsmodul verwendet ein passwortbasierendes Verfahren, welches auch zur Schlüsselvereinbarung des Secure Channel verwendet wird. Hierzu wird das Kommando GENERAL AUTHENTICATE in der Variante *PACE* verwendet, wobei zuvor mehrere Parameter (siehe Tabelle 17) über das MSE SET gesetzt werden müssen. Das dabei verwendete Protokoll wird in Abschnitt 5.10 beschrieben, und ist eine auf die Fähigkeiten der eingesetzten Smartcard angepasste Abwandlung des *PACE*-Protokolls. Die Client-Bibliothek führt dieses Verfahren mittels der Methode *generalAuthenticatePace* durch. Ist die Authentifizierung erfolgreich, wird der interne Sicherheitszustand *PACE* gesetzt und die darauf folgende Kommunikation wird gesichert (siehe Abschnitt 5.9) durchgeführt.

5.11.5.8 Kommando GENERAL AUTHENTICATE (Variante Key Agreement)

Für die Schlüsselvereinbarung mit externen Stellen soll das Gateway auf die entsprechende Funktionalität des Sicherheitsmoduls zurückgreifen. Hierzu bietet das Kommando GENERAL AUTHENTICATE die Erzeugung des gemeinsamen Geheimnisses mittels *Anonymous Diffie-Hellman* an. Die Schlüsselableitung selbst muss vom Gateway übernommen werden. Das Kommando setzt einen vorherigen Aufruf des MSE SET Kommandos voraus, um die zu verwendende Protokollvariante festzulegen. Bei der Schlüsselvereinbarung werden zwei *Ephermal*-Schlüsselpaare verwendet, wobei jeder der Kommunikationspartner über beide öffentlichen Schlüssel und einen der privaten Schlüssel verfügt. Im Rahmen der Abarbeitung des Kommandos werden folgende Schritte durchgeführt:

- Der externe Partner generiert ein neues *Ephermal*-Schlüsselpaar und sendet den öffentlichen Schlüssel an das Gateway.
- Das Gateway sendet das Kommando GENERAL AUTHENTICATE an das Sicherheitsmodul. Dabei wird der öffentliche Schlüssel des Partners in Form des Kurvenpunkts und der OID der verwendeten elliptischen Kurve mit übertragen.
- Das Sicherheitsmodul generiert ein eigenes *Ephermal*-Schlüsselpaar und verwendet den öffentlichen Schlüssel des Partners, sowie den eigenen privaten für das *Anonymous-Diffie-Hellman*-Verfahren. Ergebnis des Verfahrens ist das gemeinsame Geheimnis.
- Das Sicherheitsmodul gibt den eigenen öffentlichen *Ephermal*-Schlüssel sowie das generierte Geheimnis an das Gateway zurück.
- Das Gateway sendet den öffentlichen Schlüssel an den Partner und der Partner kann nun selbst das gemeinsame Geheimnis ermitteln.

Die Schlüsselableitung aus dem gemeinsamen Geheimnis ist nicht Teil der Implementierung.

5.11.6 Kommando GET CHALLENGE zur Zufallszahlgenerierung

Um vom Sicherheitsmodul zur Generierung und Ausgabe einer Zufallszahl anzufordern, dient das Kommando GET CHALLENGE (*getChallenge* Methode der Client-Bibliothek). Die so generierten Zufallszahlen können entweder rein für die externe Nutzung im Gateway verwendet werden, oder bei der Authentifizierung des Gateways am Sicherheitsmodul als Challenge im Rahmen des Challenge-Response-Verfahrens verwendet werden. Für diesen Fall wird die Zufallszahl auch intern im Sicherheitsmodul gespeichert. Diese steht nach einmaliger Ausführung des EXTERNAL AUTHENICATE Kommandos, nach Selektierung des MF, oder nach nochmaliger Ausführung des GET CHALLENGE Kommandos mit interner Speicherung nicht mehr zur Verfügung. Die interne Bereithaltung der Zufallszahl ist bereits bei der Ausführung des GET CHALLENGE Kommandos als Parameter anzugeben. Die Länge der zu generierenden Zahl wird durch das APDU-Le-Feld angegeben. Die Kodierung, der gesamten Kommando APDU zur Zufallszahlengenerierung ist in Tabelle 18 dargestellt.

Tabelle 18: Kodierung des GET CHALLENGE Kommandos zur Zufallszahlengenerierung

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0x84	0x00	0x00	-	-	ChallengeLen	Zufallszahl generieren, ausgeben und intern speichern
0x00	0x84	0x01	0x00	-	-	ChallengeLen	Zufallszahl generieren und ausgeben

5.11.7 Kommando CHANGE REFERENCE DATA zum Ändern von Passwörtern in PIN-Dateien

Zum Zugriff auf PIN-Dateien steht nur das Kommando CHANGE REFERENCE DATA zur Verfügung. PIN-Dateien können nicht während des Betriebs erstellt werden, sondern müssen bereits bei der Initialisierung des Sicherheitsmoduls angelegt werden. Es dient dazu die gespeicherte Bytesequenz, welche als Passwort für die *PACE*-Authentisierung verwendet wird, zu ändern bzw. zu setzen. Dabei werden zwei Varianten unterstützt:

- Setzen einer PIN: Dies ist nur möglich, falls die Passwortsequenz noch nicht gesetzt wurde und der Lebenszyklusstatus noch auf *Initialisierung* steht. Die Ausführung des Kommandos ist auch dann nur möglich, wenn die Zugriffsbeschränkungen der PIN-Datei dies erlauben. Diese werden meist so gesetzt sein, dass ein Setzen der PIN nur für den Betriebszustand *Personalisierung und Integration* (SEID=2) möglich ist. Das erfolgreiche Ausführen des Kommandos ändert den Lebenszyklusstatus der Datei auf *aktiv*.
- Ändern einer PIN: Dies ist nur für den Fall einer bereits gesetzten Passwortsequenz und für PIN-Dateien mit dem Lebenszyklusstatus *aktiv* möglich. Die Zugriffsbeschränkungen werden wiederum ausgewertet. Der Lebenszyklusstatus der PIN-Datei wird durch das Ausführen dieses Kommandos nicht beeinflusst.

Bei der Ausführung des Kommandos wird auch überprüft, ob das neue Passwort die Anforderung an die Mindestlänge erfüllt, und keine unerlaubten Zeichen enthalten sind (siehe Kapitel 5.3.1). Das

Kommando kann über die Methoden *setPinRefData* und *changePinRefData* der Client-Bibliothek ausgeführt werden. Die Kodierung des Kommandos kann in Tabelle 19 gesehen werden.

Tabelle 19: Kodierung des CHANGE REFERENCE DATA Kommandos zum Setzen bzw. Ändern der Passwortsequenz einer PIN-Datei

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0x24	0x01	PIN-ID	DataLen	neues Passwort	-	Setzen der Passwortsequenz in der PIN-Datei mit PIN-ID
0x00	0x24	0x00	PIN-ID	DataLen	altes Passwort - neues Passwort	-	Ändern der Passwortsequenz in der PIN-Datei mit PIN-ID

5.11.8 Kommando TERMINATE CARD USAGE für die Terminierung des Sicherheitsmoduls

Das Sicherheitsmodul verfügt über das Kommando TERMINATE CARD USAGE (*terminateCardUsage* der Client-Bibliothek) zur Außerbetriebnahme. Dabei wird der Lebenszyklusstatus des Sicherheitsmoduls auf *terminiert* gesetzt. Dies ist irreversibel und hat zur Folge, dass alle darauf folgenden Kommandos mit der Fehlermeldung *InstructionNotSupported* (0x6D00) abgebrochen werden. Wie in Tabelle 4 aufgeführt, ist für die Ausführung des Kommandos der *PACE*- und *AUTH*-Sicherheitszustand erforderlich. Die Tabelle 20 zeigt die Kodierung des TERMINATE CARD USAGE Kommando-APDUs.

Tabelle 20: Kodierung des TERMINATE CARD USAGE Kommandos zum Setzen des Lebenszyklusstatus des Sicherheitsmoduls auf terminiert

CLA	INS	P1	P2	LC	Data	Le	Beschreibung
0x00	0xFE	0x00	0x00	-	-	-	Terminieren des Sicherheitsmoduls

5.12 Sicherheitsmodul Client-Bibliothek

Zur Integration des Sicherheitsmoduls in ein Smart Metering Gateway Software Framework wurde eine Java-Bibliothek erstellt. Diese stellt die Dienste des Sicherheitsmoduls als dedizierte Java-Methoden zur Verfügung. Weiters werden der Verbindungsaufbau zum Sicherheitsmodul sowie die Kommunikation durch die Bibliothek übernommen. Die Umwandlung von Datentypen, ein Plausibilitätscheck für Parameter, sowie eine grundlegende Verarbeitung der empfangenen Daten, speziell von Statuscodes erfolgt ebenfalls in der Client-Bibliothek. Neben der Nutzung einer Schnittstelle zu einem Kartenleser mit einem physikalischen Sicherheitsmodul können auch *Java Card* Simulatoren (siehe auch Abschnitt 4.4) als zu verwendendes Sicherheitsmodul angegeben werden. Auch wenn die Ausführung des Sicherheitsmoduls sinnvollerweise in einem dedizierten System erfolgen sollte [BSI13-2], welches sicherheitstechnisch speziell geschützt ist (siehe Abschnitt 3.1.1), soll die Integration der Simulatoren neben dem einfacheren Testen und Debugging, auch den Einsatz auf sehr

einfachen Systemen ohne Schnittstelle zu physikalischen Sicherheitsmodulen erlauben. Eine Übersicht über die erstellte Klassenstruktur ist in Abbildung 13 zu sehen.
Diese ist in dem Java-Package *secmodule* implementiert, deren Hauptklasse ist *SecModule*. Über diese erfolgt die Integration in ein Smart Grid Framework. Die Verwendung der Bibliothek erfolgt durch den Aufruf des Singleton-Konstruktors *getInstance()* und wird durch den Aufruf der Methode *initSecModule()* für eine physikalisches Sicherheitsmodul, bzw. mit *initSimulatedSecModule()* für ein simuliertes Sicherheitsmodul konfiguriert. Wird ein physikalisches Sicherheitsmodul verwendet, so müssen bei der Initialisierung die Applet ID sowie der Name des zu verwendenden Kartenlesers übergeben werden. Für den Einsatz eines Simulators muss die ID des zu verwendenden Simulators (siehe Tabelle 21), die AID sowie der Klassenname des zu verwendenden Applets angegeben werden.

Tabelle 21: Identifier der verschiedenen Simulatoren

ID	Simulator
1	JCardSim über die API
2	JCardSim als simulierter Kartenleser
3	JCWDE mit der *apduio* Klasse als Schnittstelle

Die Methoden, über welche die Dienste des Sicherheitsmoduls aufgerufen werden können, wurden bereits bei der Erklärung der Kommandos in Kapitel 5.11 angegeben. Eine beispielhafte Verwendung der Bibliothek wird in Listing 2 angeführt.

Listing 2: Beispielhafte Verwendung der Client-Bibliothek für das Sicherheitsmodul

```
//Aufruf des Singleton-Konstruktors
SecModule sm = SecModule.getInstance();

//Applet ID
byte[] aid = new byte[] {0x01, 0x00, 0x00, 0x00, 0x02, 0x01};

//Klassenname des zu simulierenden Sicherheitsmoduls
String className = "securityModuleApplet.SecurityModule";

//Name des Kartenlesers mit dem physikalischen Sicherheitsmodul
String readerName = "SCM Microsystems Inc. SDI010 Smart Card Reader 0";

//Initialisiere simuliertes Sicherheitsmodul, mit JCardSimTerminal Simulator
sm.initSimulatedSecModule(2, aid, className);

//Initialisiere physikalisches Sicherheitsmodul
sm.initSecModule(aid, readerName);

// fordere 4 Byte lange Zufallszahl vom Sicherheitsmodul an
byte[] r = sm.getChallenge(4, false);
```

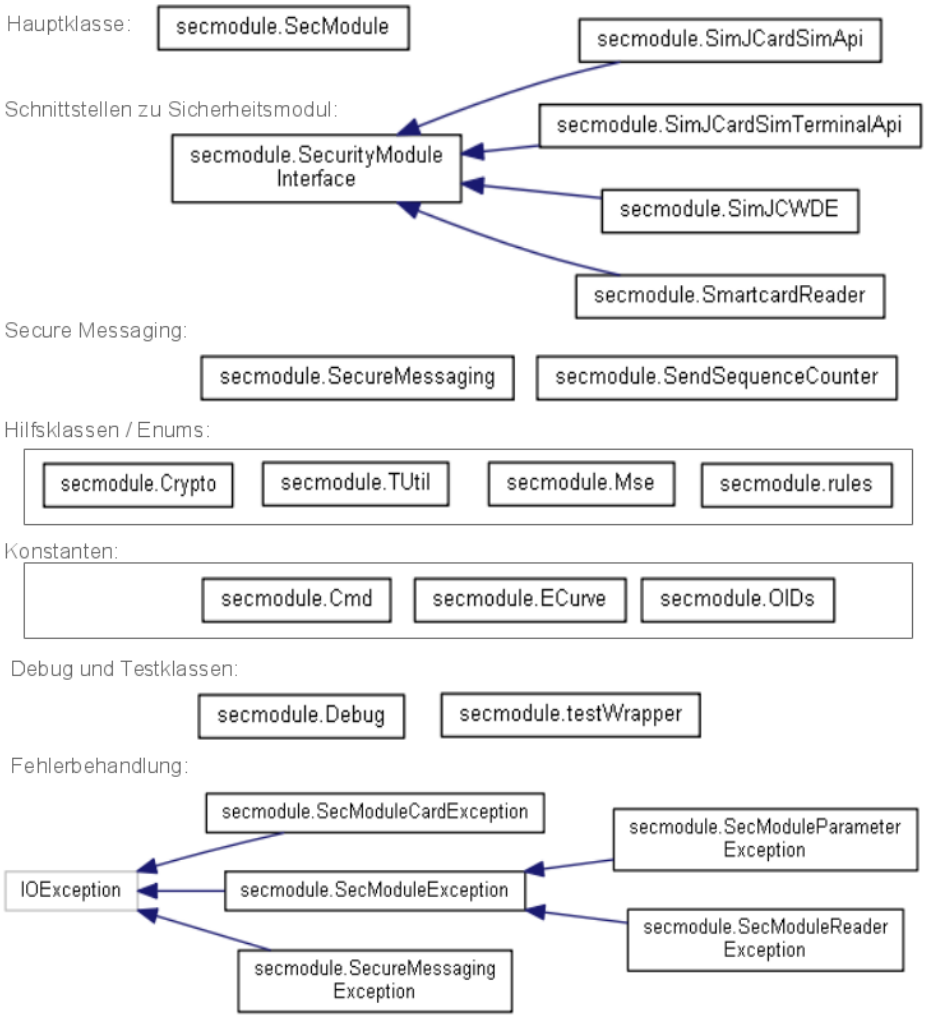


Abbildung 13: Klassenstruktur der Client-Bibliothek

5.12.1 Zugriffsbeschränkungen und Security Environments

Um die Nutzung der Client-Bibliothek des Sicherheitsmoduls zu vereinfachen, wurden die Enum *rules* und *Mse* implementiert. Das Enum *rules* vereinfacht die Generierung der bitkodierten Zugriffsbeschränkungen beim Anlegen von neuen Dateien. Über den Aufruf der Methode *makeRules(rules read, rules update, rules delete, rules activate, rules deactivate, rules terminate, rules append)* wird eine Variable des Typs *short* erzeugt, welche die kodierten Zugriffsbeschränkungen enthält und als Parameter für den Aufruf der *createFile* Methoden verwendet werden kann. Der beispielhafte Einsatz wird in Listing 3 gezeigt.
Das Enum *Mse* dient der Selektion des zu setzenden Security Environment für die Methode *mseSet(byte[] keyID, Mse mseType).* Es stehen hierbei folgende Security Environments zur Verfügung *DigitalSignatureVerification, DigitalSignatureGeneration, AuthenticationKeyAgreementEgInitiator, AuthenticationKeyAgreementEgRecipient, AuthenticationKeyAgreementDh, AuthenticationExt,*

AuthenticationInt. Das Setzen des Security Environment *AuthenticationPace* steht nur über die Methode *mseSetAtPace()* zur Verfügung.

Listing 3: Beispielhafte Verwendung der Enum rules beim Erzeugen einer Datei

```
//Erzeugen eines Record-basierten EF, setzt initialisiertes Sicherheitsmodul voraus

//erzeugen der bitkodierten Zugriffsbeschränkungen:
//read ohne Beschränkung, update niemals,
//delete gesetzte AUTH und PACE Sicherheitszustände,
//activate, deactivate, terminate, append gesetzter PACE Sicherheitszustand
short rule = rules.makeRules(rules.ALWAYS, rules.NEVER, rules.AUTH_PACE, rules.PACE,
rules.PACE, rules.PACE , rules.PACE , rules.PACE);

// File ID der zu erzeugenden Datei
short fid = 0x43ab;

// Short File ID der zu erzeugenden Datei
int sfi = 0x01;

// Kommando an Sicherheitsmodul zum Erzeugen der Datei
sm.createFile(false, fid, sfi , rule);
```

5.12.2 Exceptions

Für den Einsatz der Client-Bibliothek ist die Einbindung der von ihr genutzten Exceptions notwendig. Diese sind für verschiedene Kategorien von Ausnahmen unterschiedlich:

- *secmodule.SecModuleException*: Exception für allgemeine Ausnahmen der Client-Bibliothek sowie Superklasse der restlichen verwendeten Exceptions.
- *secmodule.SecModuleParameterException*: Ausnahme für fehlerhafte Parameter, die als Kommando an das Sicherheitsmodul gesendet werden sollen. Diese Ausnahmen umfassen vor allem die Überschreitung des Wertebereichs, sowie nicht erlaubte Werte (z. B. nicht implementierte Bitlängenangaben oder OIDs).
- *secmodule.SecModuleReaderException*: Ausnahmen, die vom Kartenleser erzeugt werden. Diese umfassen Lese- und Schreibfehler, sowie nicht vorhandene Smartcards.
- *secmodule.SecModuleCardException*: Enthält Ausnahmen, die vom Sicherheitsmodul im Rahmen der Abarbeitung von Kommandos erzeugt wurden. Diese sind in Tabelle 5 aufgelistet und werden in der Exception über die dort aufgelisteten Fehlernamen referenziert. Über die Methode *is(exceptionCause)* der Klasse kann überprüft werden, ob die Ausnahme aufgrund des als Parameter übergebenen Fehlers ausgelöst wurde. Besondere Bedeutung kommt der Ausnahmebehandlung dieser Exceptions im Rahmen von Signaturüberprüfungen zu. Die Ungültigkeit einer Signatur führt zu einer *SecModuleCardException* des Typs *„Verification Error“*. Ein typisches Beispiel für die Überprüfung auf diesen Fehler wird in Listing 4 gegeben. Wird die Signaturüberprüfung direkt von der Client-Bibliothek über deren Methoden *psoVerifyDigitalSignature(), externalAuthenticate(), psoVerifyCertificate()*durchgeführt, so übernehmen diese auch die Überprüfung der Exception und liefern

einen booleschen Rückgabewert. *SecModuleCardException* des Typs *„Verification ror“* werden also nicht an das externe Framework weitergegeben, sondern nur durch den Rückgabewert der Methode angezeigt.

Listing 4: Ausnahmebehandlung für Signaturverifikation

```
try {
   // Senden des Kommandos an das Sicherheitsmodul
   sendApdu(new CommandAPDU(ISO7816.CLA_ISO7816, Cmd.INS, P1, P2, data));

   //Falls keine Exception aufgetreten ist, war Signatur gültig
   sigIsValid = true;
}
catch (SecModuleCardException ce){
   if (ce.is("VerificationError")){

     //Signatur ist ungültig, da VerificationError aufgetreten ist
     sigIsValid = false;
   } else {

     //Ausnahme war nicht VerificationError, weitergeben für Fehlerbehandlung
     throw ce;
   }
}
```

5.13 Integration in das *OGEMA*-Framework

Zur Demonstration der Nutzbarkeit des Sicherheitsmoduls in einem Smart Metering Gateway Framework, wurde eine Applikation für das *OGEMA*-Framework erstellt. Hierzu wurde ein sogenanntes *RunningModule* implementiert, welches über das Webinterface den Zugriff auf die Dienste des Sicherheitsmoduls ermöglicht. Für den späteren Feldeinsatz des Smart Metering Gateway ist dieses Webinterface nicht vorgesehen. Hierbei würden die über das *RunningModule* zur Verfügung gestellten Dienste des Sicherheitsmoduls direkt von anderen Applikationen des Frameworks genutzt.
Die Implementierung erfolgte in den Packages *org.securitymodule.app* und *org.securitymodule.app.servlet*. Das App Package enthält mit der Klasse *SecurityModuleApp* die Schnittstelle zwischen dem *OGEMA*-Framework und dem Sicherheitsmodul. Hierbei wird die in Abschnitt 0 beschriebene Client-Bibliothek eingesetzt. *SecurityModuleApp* übernimmt also die Initialisierung des Sicherheitsmoduls und stellt die Instanz der Client-Bibliothek für das Framework zur Verfügung. Das Servlet Package mit der Java-Klasse *SecurityModuleAppServlet* und der JSP-Seite *secModApp.jsp* stellt die Benutzerschnittstelle als Web-Interface zur Verfügung. *SecurityModuleAppServlet* stellt hierbei Wrapper-Funktionen für den Zugriff auf die Client-Bibliothek Instanz der *SecurityModuleApp* zur Verfügung, sowie Hilfsfunktionen für den Aufbau der Website. Die vollständige Struktur der Einbindung ist in Abbildung 14 dargestellt.

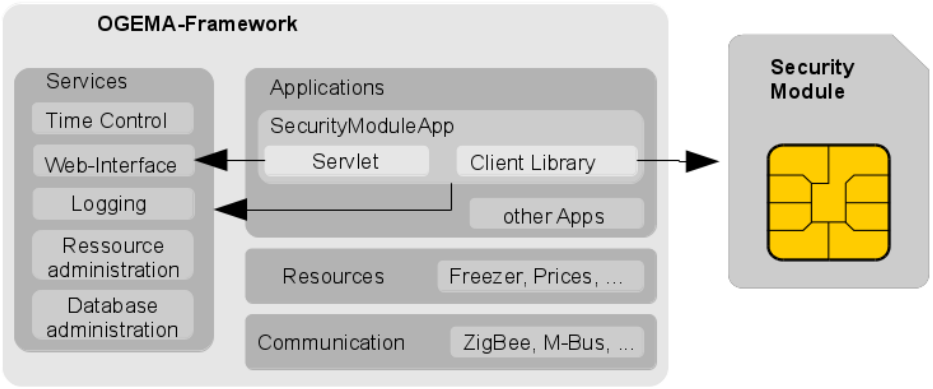


Abbildung 14: Einbindung des Sicherheitsmoduls in das *OGEMA*-Framework.

Das Webinterface ist in mehrere Bereiche gegliedert, wie in Abbildung 15 zu sehen. Über diese nach Diensten getrennten Bereiche kann auf die Funktionen des Sicherheitsmoduls zugegriffen werden:

- **Manage Filesystem** ermöglicht Zugriff auf die Dateimanagementfunktionen zum Selektieren, aktivieren, deaktivieren, terminieren und löschen von Elementary und Dedicated Files.
- **Authentication** stellt Zugriff auf die Authentifizierungsdienste (internal Authenticate, external Authenticate und *PACE* Authenticate) zur Verfügung.
- **File Access** erlaubt lesenden und schreibenden Zugriff auf Elementary Files.
- **Key Management** dient dem Erstellen von Key-Dateien, deren Aktivierung, Deaktivierung, Löschen und dem Generieren von Schlüsseldaten für Schlüsselpaare.
- **Sign/Verify** stellt die Schnittstelle für die Dienste zur Generierung von Signaturen sowie zur Verifikation von Signaturen und Zertifikaten dar.

Bei der Nutzung des Webinterfaces müssen die Aktionen im Allgemeinen in derselben Reihenfolge durchgeführt werden, wie diese auf dem Sicherheitsmodul ausgeführt werden sollen. So muss vor der Selektion eines EF, das übergeordneter DF (oder das MF) selektiert worden sein. Für den Lese- und Schreibzugriff auf ein EF muss dieses zuvor selektiert werden. Für die Authentifizierungs- und Signaturfunktionen werden implizit die dafür geeigneten Schlüsseldateien im Dateisystem referenziert. Weiters werden die Aktionen für das Management des Security Environment implizit durchgeführt, wenn diese für eine gewählte Funktion benötigt werden.

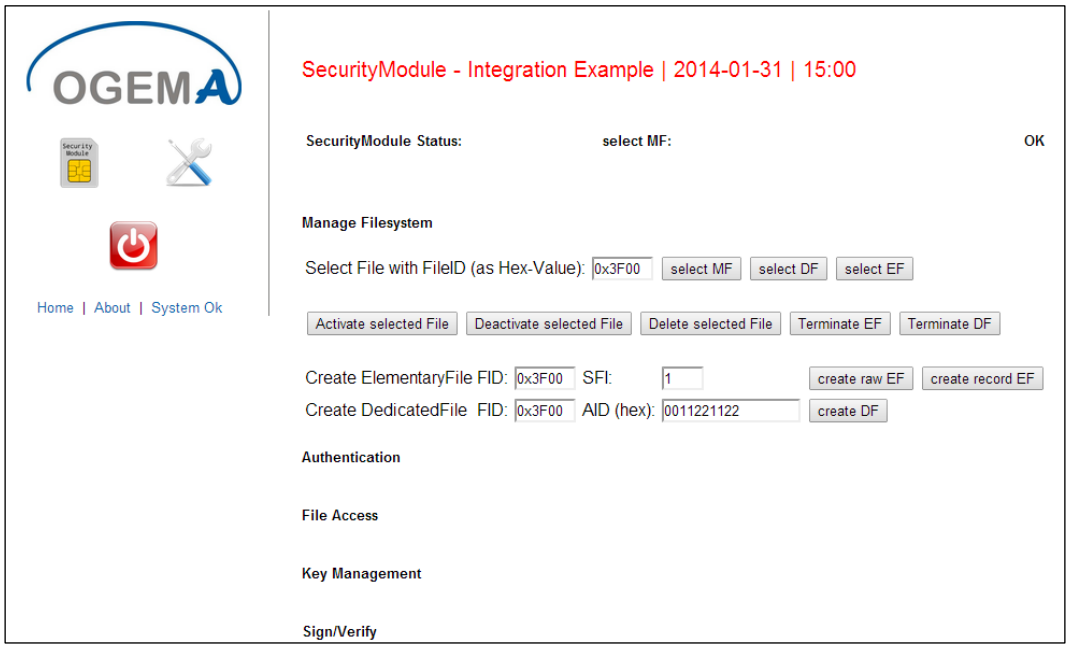


Abbildung 15: Dateimanagementbereich des Webinterface für das Sicherheitsmodul

6 Diskussion

Bei der Realisierung des Sicherheitsmoduls traten verschiedene Herausforderungen auf, welche es nötig machten Anpassungen an den geplanten Features vorzunehmen. Die vorgesehene Funktionalität des Sicherheitsmoduls konnte jedoch beibehalten werden, ohne die Ansprüche an die Sicherheit zurückzuschrauben. Zudem wurden weiterreichende Erkenntnisse bezüglich der einzusetzenden Plattform und den externen, nicht von dem Sicherheitsmodul abhängenden, Voraussetzungen für einen sicheren Einsatz des SMGW gewonnen.

6.1 Durchgeführte Abwandlungen zum BSI Schutzprofil

Bei der Entwicklung des Sicherheitsmoduls wurde hoher Wert auf die Vereinbarkeit mit dem BSI *Schutzprofil für das Sicherheitsmodul der Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen* [BSI13-2] gelegt. Aufgrund der eingesetzten Smartcard und dafür verfügbaren Simulatoren wurden jedoch einige Abwandlungen bei der Implementierung durchgeführt. Diese sollen im Folgenden aufgelistet und begründet werden.

6.1.1 Schlüssellänge für elliptische Kurven Kryptografie

Das Schutzprofil sieht den Einsatz von Schlüsseln mit Längen von 256, 384 und 512 Bit vor. Die für die Entwicklung verwendete Smartcard JCLX80JTOP20ID unterstützt nur Schlüssellängen von bis zu 192 Bit. Es wurden deshalb auch die Domain Parameter für die 192 Bit *BrainpoolP192R1* elliptische Kurve hinzugefügt. Die Unterstützung für längere Schlüssel wurde im Sicherheitsmodul implementiert, die konkrete Verwendung dieser konnte aber nicht überprüft werden.

6.1.2 Kryptografische Prüfsumme für Secure Messaging

Für die geschützte Kommunikation zwischen Sicherheitsmodul und Gateway (Secure Messaging) soll laut Schutzprofil eine kryptografische Prüfsumme des Typs *AES-CMAC* mit 128 Bit Schlüssellänge verwendet werden. Für die eingesetzte Implementierungsplattform (Smartcard JCLX80JTOP20ID) steht diese Form der Prüfsummenbildung nicht zur Verfügung. Sie wurde deshalb durch eine *CMAC* nach DES (Data Encryption Standard) ersetzt. Die konkret im Applet eingesetzte Variante des Algorithmus ist *javacard.security.Signature.ALG_DES_MAC8_NOPAD*. Im Gegensatz zu der *AES-CMAC* wird hierbei nur eine Prüfsumme mit 8 Byte Länge, anstatt 16 Byte erzeugt. Dies ist aber insoweit unwichtig, da das Sicherheitsprofil nur die Übertragung der 8

höchstwertigen Bytes der Prüfsumme vorsieht. Auch wird nur ein 64 Bit langer Schlüssel verwendet. Es wird bei der Abarbeitung des *PACE*-Protokolls aber trotzdem ein 128 Bit Schlüssel für die Prüfsummenbildung generiert, von dem aber wiederum nur die 64 höchstwertigen Bit verwendet werden.

6.1.3 Signaturbildung

Für die Generierung und Validierung von Signaturen wird das ECDH-Verfahren verwendet. Laut Schutzprofil werden hierzu nicht die zu signierenden Nachrichten an das Sicherheitsmodul gesendet, sondern nur der kryptografische Hash über die Nachricht. Die Java-Card-Spezifikation [SUN03] sieht für das ECDH-Verfahren aber immer eine Hashbildung über die Eingangsdaten der Signier- und Verifiziermethoden vor. Um nicht die komplette Nachricht, welche mehrere kByte groß sein könnte, an das Sicherheitsmodul senden zu müssen, wurde eine zweifache Hashbildung über die Nachricht implementiert. Das heißt, der Hash über die Nachricht wird im Gateway gebildet und an das Sicherheitsmodul gesendet, dort wird der empfangene Hash bei der Durchführung des ECDH-Verfahrens nochmals gehasht. Wird dies bei der Verifizierung der Nachrichten berücksichtigt und ebenso gehandhabt, ergibt sich hierdurch keine Verringerung der Sicherheit. Einzig die zu erwartende Gesamtrechenzeit erhöht sich.

6.1.4 Schlüsselvereinbarung

Für das Kommando GENERAL AUTHENTICATE ist im Sicherheitsprofil neben dem EC-Diffie-Hellman-Verfahren auch die Schlüsselvereinbarung mit dem ElGamal-Verfahren für elliptische Kurven vorgesehen. Aufgrund der fehlenden Unterstützung des ElGamal-Verfahrens in der eingesetzten Smartcard wurde nur die ECDH-Schlüsselvereinbarung implementiert.

6.1.5 *PACE*-Protokoll

Das *PACE*-Protokoll dient zur Authentifizierung des Gateway Betreibers am Sicherheitsmodul, sowie zur Herstellung einer geschützten Verbindung zwischen Gateway und Sicherheitsmodul (siehe Abschnitt 5.10). Die genaue Spezifikation dieses *Password Authenticated Connection Establishment* Protokolls wird in [BSI13-4] beschrieben. Der Ablauf ist hierbei wie folgt:

1. Das Sicherheitsmodul erzeugt eine Zufallszahl. Diese wird zusammen mit den Domainparametern der zu erzeugenden Schlüssel verschlüsselt (der Schlüssel wird durch eine statische Funktion aus dem Passwort abgeleitet). Die verschlüsselte Nachricht wird an das Gateway gesendet.
2. Die Nachricht wird im Gateway mit dem bekannten Passwort entschlüsselt.
3. Das Sicherheitsmodul und das Gateway verwenden die Domain Parameter und die Zufallszahl zur Erzeugung temporärer (ephemeral) Domain Parameter (Mapping).
4. Ein anonymer Diffie-Hellman-Schlüsselaustausch wird durchgeführt, wobei die ephemeral Domain Parameter zur Erzeugung des gemeinsamen Geheimnisses verwendet werden.
5. Aus dem gemeinsamen Geheimnis werden die Schlüssel für die MAC (Prüfsumme) und Verschlüsselung abgeleitet.

6. Die Authentifizierung wird durch den Austausch eines Tokens, welches mit dem MAC-Schlüssel erzeugt wurde, abgeschlossen.

Für die Implementierung des *PACE*-Protokoll [BSI13-4] im Sicherheitsmodul soll das Generic Mapping verwendet werden. Eine Analyse des Einsatzes von *PACE* auf *Java Card* Smartcards und die Voraussetzungen hierfür sind bei [UKNVS08] und [WHBSB11] nachlesbar. Hieraus geht hervor, dass für eine sinnvolle Speicher- und Rechenzeitnutzung zusätzliche, nicht zur Standard API der Java-Card-Spezifikation gehörende Funktionen nötig sind. Konkret werden für die Implementierung des Generic Mapping Funktionen zur Addition von Punkten auf elliptischen Kurven benötigt. Zusätzlich muss das Verfahren des Diffie-Hellman-Schlüsseltausches in einer Form implementiert sein, in der auf die Ergebnisse des Verfahrens vor der Ableitung der Schlüssel zugegriffen werden kann. Diese beiden Funktionen werden von einigen neueren Smartcards mit dem *JCOP (Java Card Operating System)* Betriebssystem von NXP unterstützt (siehe auch Abschnitt 6.2.5), sind jedoch nicht in der eingesetzten Infineon JCLX80JTOP20ID Smartcard verfügbar. Es wurde deshalb ein in Abschnitt 5.10 beschriebenes, abgewandeltes Protokoll zur Authentifizierung des Betreibers und für den Secure-Messaging-Schlüsseltausch implementiert.

6.2 Resultierende Anforderungen an die Smartcard

Wie in Abschnitt 6.1 beschrieben, wurden bei der Implementierung des Sicherheitsmoduls einige Abwandlungen im Bezug zum BSI Sicherheitsprofil [BSI13-2] durchgeführt. Für die lückenlose Implementierung eines Sicherheitsmoduls, welches dem BSI Sicherheitsprofil entspricht, wurden die im folgenden beschriebenen Anforderungen ermittelt.

6.2.1 Kryptografische Funktionen

An kryptografischen Funktionen werden folgende Anforderungen gestellt:

- Generieren von bis zu 64 Byte langen Zufallszahlen nach einem der folgenden Standards: *DRG.3, DRG.4 PTG.3* oder *NTG.1* [BSI12-1]
- *AES CBC* Verschlüsselung mit 128 Bit Schlüssellänge (256 Bit empfohlen) [BSI12-1]
- *AES CMAC* kryptografische Prüfsummenbildung mit 128 Bit Schlüssellänge (256 Bit empfohlen) [BSI12-1]
- Schlüsselgenerierung mit bis zu 512 Bit für elliptische Kurvenkryptografie über *Fp* (Primzahlenfeld) [BSI12-1]
- Generieren und Verifizieren von *ECDSA* (Elliptic Curve Signature Algorithm) Signaturen mit bis zu 512 Bit über *Fp* [BSI12-1]
- Generieren von kryptografischen Hashes nach den Standards *SHA-1* und *SHA-256 (SHA-2)* [BSI12-1]
- API-Funktion zur Addition von Punkten auf elliptischen Kurven [WIES 11]
- Diffie-Hellman Schlüsselvereinbarung mit elliptischen Kurven, ohne Ableitung des Schlüssels [BSI13-3]
- ElGamal-Schlüsselvereinbarung mit elliptischen Kurven, ohne Ableitung des Schlüssels [BSI13-3]

- Generieren von *ECDSA* Signaturen für bereits gehashte Nachrichten [BSI13-3]

6.2.2 Nichtflüchtiger Speicher

Besonders hohe Anforderungen werden an den nichtflüchtigen Speicher der Smartcard gestellt. Neben dem Applet selbst wird dort auch das Dateisystem des Sicherheitsmoduls abgelegt. Zur Erfüllung des Sicherheitsprofils muss Speicher für mindestens 26 Public-Key-Dateien, 8 Schlüsselpaardateien, 14 Zertifikate und 4 technische Datenfelder bereitgestellt werden [BSI13-3]. Geht man jeweils von der maximal zu unterstützenden Schlüssellänge von 512 Byte aus, ist hierfür bereits ein Speicherbedarf von mindestens 55,4 kByte erforderlich (siehe Tabelle 22). Zusätzlich sollte noch freier Speicher für im Betrieb erzeugte Datenfelder und Schlüsselpaare sowie für importierte öffentliche Schlüssel vorhanden sein. Die verwendete Smartcard sollte daher über mindestens 64 kByte nichtflüchtigen Speicher verfügen.

Tabelle 22: Bedarf an nichtflüchtigem Speicher.

Anzahl	Beschreibung	Speicherverbrauch einzeln (ca.)	Speicherverbrauch gesamt (ca.)
1x	Bytecode Applet	24,00 kByte	24,00 kByte
26x	öffentliche Schlüssel	0,42 kByte	10,92 kByte
8x	Schlüsselpaar	0,81 kByte	6,48 kByte
14x	Zertifikate	1,00 kByte	14,00 kByte
			55,40 kByte

6.2.3 Flüchtiger Speicher (RAM)

Eine Abschätzung des erforderlichen Arbeitsspeichers ist schwieriger als jene für nichtflüchtigen. Mit einbezogen werden müssen neben generierten Datenstrukturen im Arbeitsspeicher auch der Stack sowie die für die Verarbeitung vom nichtflüchtigen in den Arbeitsspeicher kopierten Daten berücksichtigt werden.

Für die Datenstrukturen wird ein Speicher im Umfang von ca. 1 kByte verwendet. Aus den Tests mit der zur Verfügung stehenden Smartcard JCLX80JTOP20ID muss weiters von einem Bedarf an Stack-Speicher von 1 kByte ausgegangen werden. Bei einer maximalen Schlüssellänge von 512 Byte reicht für den den APDU-Buffer ein Speicherbereich von 280 Byte aus. Sollen Zertifikate mit einem einzelnen Kommando gelesen und geschrieben werden ist 1 kByte für den APDU-Buffer vorzusehen.

6.2.4 Sonstige Anforderungen

Zusätzlich zu den genannten Merkmalen ist Unterstützung für *extended APDUs* mit Datenfeldern größer als 256 Byte Voraussetzung für die vollständige Implementierung des Sicherheitsmoduls.

An die Rechenleistung der Smartcard werden keine besonderen Anforderungen gestellt. Wird eine Smartcard mit wenig Rechenleistung eingesetzt, wird die Rechenzeit für die Abarbeitung der einzelnen Kommandos entsprechend zunehmen. Dies sollte für den Einsatz mit einem Smart Metering

Gateway, dessen Datenaustausch eher gering ist und keine kurzen Latenzzeiten erfordert, kein Problem darstellen.

Weiters ist eine kontaktbezogene Kommunikationsschnittstelle nach dem ISO 7816-3 Standard notwendig.

6.2.5 Möglicher Smartcard Kandidat

Als eine mögliche Smartcard, die diese Anforderungen weitestgehend erfüllt, wurde die *NXP JCOP v2.4.1 R3* in Betracht gezogen [NXP12]. Sie unterstützt jedoch nur Schlüssellängen bis 320 Bit für elliptische Kurven basierende Kryptografie. Die weiteren kryptografischen Merkmale, inklusive der Addition von elliptischen Kurvenpunkten werden unterstützt. Die Smartcard verfügt über bis zu 128 kByte nichtflüchtigen und 6,1 kByte Arbeitsspeicher [NXP11]. *Extended APDU* Unterstützung ist vorhanden und es sind Karten mit ISO 7816-3 Schnittstelle verfügbar.

6.3 Externe Abhängigkeiten für sicheren Betrieb

Durch das Hinzufügen des Sicherheitsmoduls in ein Smart Metering Gateway wird das System nicht automatisch sicherer. Es müssen einige zusätzliche Vorkehrungen und Maßnahmen getroffen werden, die außerhalb des direkten Einflussbereichs des Sicherheitsmoduls liegen, bzw. die von der Nutzungsweise des Sicherheitsmoduls abhängig sind.

6.3.1 Public-Key-Infrastruktur

Für die Nutzung von Public-Key-Kryptografie zur Absicherung der Kommunikation, ist ein System nötig, welches die Ausstellung, Überprüfung und Verteilung von Schlüsseln mit Hilfe von Zertifikaten unterstützt. Dieses System wird als Public-Key-Infrastruktur bezeichnet. Da das Smart Metering Gateway der zentrale Kommunikationsknoten für den Datenaustausch zwischen dem als vertrauenswürdig betrachteten HAN (Home Area Network) und dem ungesicherten WAN (Wide Area Network) ist, muss es in diese PKI eingebunden werden. Die PKI dient also dazu, dass das Smart Metering Gateway über die WAN-Schnittstelle Zugriff auf die zur Kommunikation verwendeten Zertifikate (und darin enthaltenen Zertifikatsketten) erhält und diese überprüfen kann. Eine solche PKI wird in der Technischen Richtlinie TR-03109-4 des BSI [BSI13-5] für Smart Meter Gateways beschrieben. Diese technische Richtlinie spezifiziert neben der Architektur der Smart Metering PKI, auch die Mindestanforderungen an Sicherheit und Anforderungen, welche für die Sicherstellung der Interoperabilität des Systems zu erfüllen sind.

In Abbildung 16 wird eine mögliche PKI-Architektur für Smart Meter Gateways gezeigt. Zentrales Element ist dabei die hoheitliche Zertifizierungsstelle Root CA (Certificate Authority). Diese ist die Wurzelzertifizierungsinstanz der streng hierarchisch aufgebauten PKI. Alle anderen gültigen Zertifikate in der PKI sind auf die Wurzelzertifikate des Root CA rückverfolgbar. Damit dient die Root CA als alleiniger Vertrauensanker des Systems. Die Sub CA dient als Zertifikatsaussteller für die Endnutzer der Zertifikate. Eine Sub CA kann dabei für unternehmensintern für einen einzigen Teilnehmer (z. B. einen Lieferanten) oder für eine Vielzahl von Teilnehmern (z. B. alle Smart Meter Gate-

ways eines Netzbetreibers) zuständig sein. Sie erhalten ihre Signaturrechte von der Root CA, welche auch von dieser entzogen werden können, wodurch die Zertifikate der Sub CA und die damit erstellten ihre Gültigkeit verlieren. Die Zertifizierungsstellen (Root CA und Sub CAs) verwalten zudem Sperrlisten der widerrufenen und Verzeichnisse der ausgestellten Zertifikate.

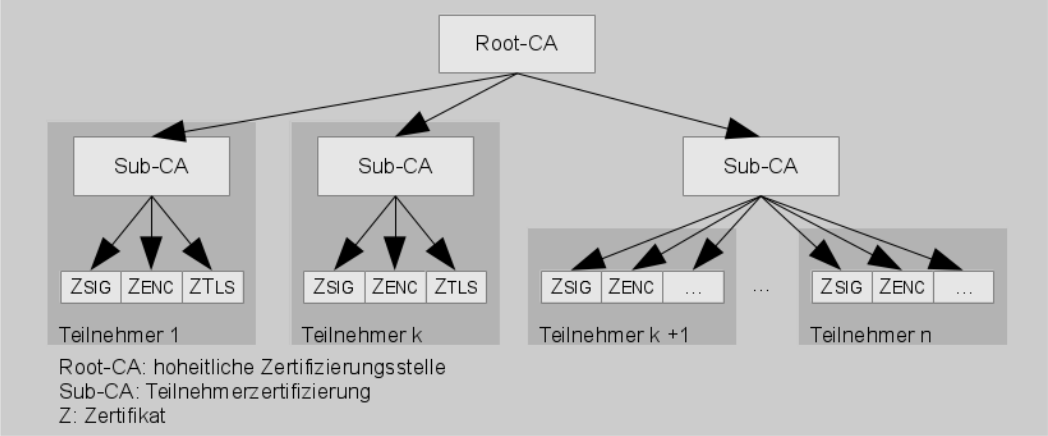


Abbildung 16: Architektur der Public-Key-Infrastruktur für Smart Meter Gateways nach [BSI13-5]

Auch die Zertifizierungsrichtlinien (Certificate Policy, CP) werden von den CAs verwaltet und bereitgestellt. Diese spezifizieren unter anderem die organisatorischen und technischen Anforderungen für den Umgang der Zertifikatsendnutzer mit den eigenen und den Zertifikaten bisher unbekannter Kommunikationsteilnehmer. Hierzu zählen die Anforderungen und Richtlinien für die Benutzung, die Anerkennung, das Ausstellen, die Verwaltung, das Zurückziehen und die Erneuerung von Zertifikaten. Auch die Architekturbeschreibung der PKI, die Richtlinien für die Namen- und Identifier-Vergabe, sowie die Beschreibung der Sicherheitsmaßnahmen zum Schutz der PKI sind in einer typischen Certificate Policy enthalten [RFC3647].

Endnutzer der von der PKI verwalteten Zertifikate welche in Abbildung 16 als Teilnehmer bezeichnet werden, sind neben den Smart Metering Gateways und den Betreibern der Gateways (Gatewayadministratoren) auch externe Marktteilnehmer. Zu ihnen zählen alle potenziellen Kommunikationspartner, die nicht in die ersten beiden Kategorien fallen. Dazu zählen etwa Verteilernetzbetreiber, Messstellenbetreiber oder Lieferanten [BSI13-5]. Die für die Endbenutzer auszustellenden Zertifikate sind an einen Verwendungszweck gebunden und dürfen nur für diesen eingesetzt werden. Jedes Smart Metering Gateway sollte deshalb getrennte Zertifikate für zumindest folgende Verwendungszwecke erhalten:

- TLS: Authentisierung von Marktteilnehmern und gesicherter Verbindungsaufbau mit diesen (verschlüsselt und integritätsgesichert).
- Verschlüsselung: Sicherung von Daten durch Verschlüsselung, unabhängig von Kommunikationsverbindungen (TLS).
- Signatur: Erstellung und Prüfung elektronischer Signaturen auf Datenebene, unabhängig von Kommunikationsverbindungen (TLS).

Die von den Sub CAs ausgestellten Zertifikate verfügen über eine beschränkte Gültigkeitsdauer. Die in dem Sicherheitsmodul gespeicherten Zertifikate müssen deshalb vom Gatewayadministrator vor einem eventuellen Ablauf durch aktuelle ausgetauscht werden. Das Sicherheitsmodul selbst verfügt nicht über genügend Ressourcen (Rechen- und Speicherleistung), aber vor allem nicht über eine Kommunikationsmöglichkeit mit den Zertifizierungs- und Registrierungsstellen für die Gültigkeitskontrolle von Zertifikaten. Die Überprüfung der Zertifikate und der darin enthaltenen Zertifikatsketten sowie die Abfrage der Sperrlisten werden deshalb nicht vom Sicherheitsmodul selbst, sondern vom Gateway durchgeführt. Das Sicherheitsmodul hat hierbei jedoch die Aufgabe die Signaturen der Zertifikate zu überprüfen (Kommando *VERIFY_CERTIFICATE*).

Für den Datenaustausch mit der externen Welt werden vor allem die Zertifikate für die TLS-Kommunikation, sowie für die Sicherung der für die externe Welt bestimmten Daten (Verschlüsselungs- und Signaturzertifikate) verwendet. Zudem enthält das Sicherheitsmodul weitere Zertifikate für die gegenseitige Authentisierung des Gateways mit den Smart Metern im LMN (Local Metrological Network), sowie solche zur Authentifizierung von Geräten im HAN [BSI13-5]. Da diese nicht von externen Teilnehmern aus dem WAN verwendet werden, sind diese weniger sicherheitskritisch und werden auch nicht in der Smart Grid PKI verwaltet.

6.3.2 Sicherung des Gateways vor physikalischem Zugriff von Dritten

Die im Sicherheitsmodul implementierten Sicherheitsfunktionen dienen vor allem dem Schutz der Kommunikation und der sicheren Speicherung von Daten. Diese sind aber kein ausreichender Schutz, falls ein Angreifer direkten, also physikalischen Zugriff auf das Smart Metering Gateway, das Sicherheitsmodul oder die Kommunikationsverbindung zwischen den beiden hat. Daher sollte auch ein physikalischer Zugriffsschutz bestehen [BSI13-1]. Dieser kann etwa aus einem verschließbaren Schaltschrank bestehen. Um einen unberechtigten physikalischen Zugriff auf das System zu erkennen, sollte dieses durch ein Siegel markiert werden. Das zu schützende System sollte zumindest das Gateway und das Sicherheitsmodul umfassen. Falls es die räumlichen Gegebenheiten zulassen, sollten auch die angebundenen Energiemessgeräte und sonstigen Smart Meter in den geschützten Bereich positioniert werden.

6.3.3 Umfassende Nutzung des Sicherheitsmoduls

Durch den Einsatz des Sicherheitsmoduls im Smart Metering Gateway wird dessen Sicherheit nicht automatisch erhöht. Es reicht also nicht, nur für einige willkürlich ausgewählte Funktionen die vom Sicherheitsmodul zur Verfügung gestellten kryptografischen Dienste einzusetzen. Um die Sicherheit des Systems objektiv zu erhöhen, muss das Sicherheitsmodul in einer konsistenten und umfassenden Weise in das Kommunikations- und Datenverarbeitungsverhalten des Gateways integriert werden. Hierzu sollten die in den Schutzprofilen des BSI für Smart Metering Gateways [BSI13-1] und im Besonderen für das Sicherheitsmodul des Gateways [BSI13-2] beschriebenen Vorgaben beachtet werden. Wichtig ist hierbei die genaue Beachtung und Einhaltung der Aufgabenteilung zwischen Gateway und Sicherheitsmodul. Diese wird in Tabelle 23 beschrieben.

Tabelle 23: Aufgabenteilung kryptografischer Funktionen zwischen Smart Metering Gateway und Sicherheitsmodul

Aufgabe	Aufgabenbereich Gateway	Aufgabenbereich Sicherheitsmodul
Kommunikation mit externen Partnern, Verbrauchern und Messgeräten	Verschlüsselung Entschlüsselung, Hash-Berechnung, Schlüsselableitung, Zertifikatsüberprüfung	Schlüsselvereinbarung (Authentifizierung, Speicherung privater Schlüssel, Zufallszahlengenerierung), Generierung und Verifikation von Signaturen
Signaturerstellung für zu versendende Daten	Hash-Berechnung	Signaturerstellung, Speicherung privater Schlüssel
Verifikation von Messdaten der Messgeräte	Hash-Berechnung, Speicherung öffentlicher Schlüssel	Signaturverifizierung, Speicherung privater Schlüssel
Verschlüsselung und Integritätsschutz zu speichernder Daten	Verschlüsselung, Entschlüsselung, MAC-Berechnung, Schlüsselableitung, Speicherung öffentlicher Schlüssel	Schlüsselvereinbarung (Speicherung privater Schlüssel, Zufallszahlengenerierung)

6.3.4 Restriktive Nutzung von Zugriffsbeschränkungen für Dateien und Initialisierung des Dateisystems

Die Kommandos zur Nutzung der Dienste des Sicherheitsmoduls sind durch Zugriffsbeschränkungen geschützt. Trotzdem ist bei der Erstellung von Dateien, insbesondere von Schlüssel- und PIN-Objekten besondere Sorgfalt auf das Setzen der dateibezogenen Zugriffsbeschränkungen zu legen. Dies bedeutet, dass die Rechte so restriktiv wie möglich gesetzt werden müssen. Dabei darf die Funktionalität aber nicht soweit eingeschränkt werden, dass die Nutzung des Sicherheitsmoduls nicht mehr sinnvoll möglich ist. So darf z. B. die Zugriffsbeschränkung für das Schlüsselpaar zur externen Authentifizierung nicht dermaßen gesetzt werden, dass für dessen Nutzung der *AUTH*-Sicherheitsstatus gesetzt sein muss. Für die PIN-Datei zur *PACE*-Authentisierung muss weiters beachtet werden, dass diese zumindest bei der Initialisierung mit den Referenzdaten befüllt werden kann.

Auch bei der Erstellung des initialen Dateisystems, welches durch die Klasse *FileSystemInit* erzeugt wird, sind die beschriebenen Maßnahmen zur restriktiven Nutzung von Zugriffsbeschränkungen für Dateien zu beachten. Hierbei sollten vor allem den für die Nutzung unbedingt nötigen Dateien und Schlüsselobjekten (z. B. das Root-Zertifikat des Smart Grid PKI) und den PIN-Dateien, welche nur in dieser Phase erstellt werden können, sehr limitierende Zugriffsbeschränkungen zugewiesen werden. Es wird empfohlen die in den technischen Richtlinien für das Schutzprofil vorgeschlagene Umsetzung des initialen Dateisystems zu verwenden [BSI13-3] (Kapitel 3.1.2). Diese enthält die Beschreibung (Identifier, Zugriffsbeschränkungen, Vorbelegung mit Daten und Schlüsselmaterial) aller für den Betrieb des Sicherheitsmoduls im Rahmen des Schutzprofils [BSI13-2] benötigten Dateien.

6.3.5 Deaktivieren der Debuggingfunktionalität

Während der Entwicklung wurden diverse Kommandos implementiert, die nur zum Debugging genutzt wurden und in einem Sicherheitsmodul im Einsatz deaktiviert werden müssen. Besonders sicherheitskritisch sind hierbei die Kommandos *DBINS_Pace, DBINS_Auth* und *DBINS_PaceAuth*, welche das Setzen des *PACE*- und des *AUTH*-Sicherheitszustandes ohne Authentifizierung ermöglichen. Sicherheitskritisch sind auch die Befehle *DBINS_smOn* und *DBINS_smOff*, mit denen eine Secure-Messaging-Sitzung gestartet bzw. beendet werden kann, ohne das restliche Security Environment zu beeinflussen. Die Debuggingkommandos können durch das Setzen der Variable *DEBUG* auf *false* in der *SecurityModul*e Klasse deaktiviert werden. Da diese Variable mit dem Modifizierer *final* deklariert wurde und sie bei der Deklaration auch initialisiert wird, ist ein Ändern während der Laufzeit nicht möglich, was aus sicherheitstechnischer Sicht auch nicht gewünscht ist. Weiters wird durch das Deaktivieren der Debuggingfunktionalität auch die Erstellung der Schlüsseldateien für das Debugging verhindert. Zudem sollte genauso die Variable *EMU* auf *false* gesetzt werden, wodurch das Applet einen sicheren Zufallszahlengenerator verwendet. Die Variable dient dazu beim Einsatz des *JCWDE*-Simulators den unterstützten Algorithmus *RandomData.ALG_PSEUDO_RANDOM* für Pseudo-Zufallszahlen zu verwenden, anstatt des *RandomData.ALG_SECURE_RANDOM* zum erzeugen sicherer Zufallszahlen.

6.3.6 Zusätzliche Sicherheitsvorkehrungen

Das Schutzprofil für das Sicherheitsmodul [BSI13-2] schreibt die Beachtung weiterer Schutzziele für den Betrieb vor:

- Einhaltung technischer und organisatorischer Sicherheitsmaßnahmen während der Integration des Sicherheitsmoduls in das Gateway. Der Schutz der Integrität, Vertraulichkeit und Authentizität der zu diesem Zeitpunkt generierten, importierten bzw. installierten Schlüsseldaten, Zertifikate und PIN-Objekte muss sichergestellt werden.
- Die Vertrauenswürdigkeit des Administrators des Sicherheitsmoduls muss sichergestellt werden, sowie die Integrität, Vertraulichkeit und Authentizität der zum Zugriff auf das Sicherheitsmodul genutzten Schlüsseldaten und PIN-Objekte muss gewahrt werden.
- Die vom Sicherheitsmodul generierten kryptografischen Daten (Signaturen, shared Secrets für Schlüsselvereinbarung) müssen vom Gateway in sicherer Weise weiterverarbeitet werden bzw. ihre Authentizität, Vertraulichkeit und Integrität geschützt werden.
- Die Protokolle für die PIN geschützte Authentifizierung und Schlüsselableitung, sowie für das Secure Messaging müssen vom Gateway implementiert und genutzt werden. Die Implementierung ist zwar durch die Nutzung der Client-Bibliothek erfüllt, jedoch muss für die kontinuierliche Nutzung der Funktionalität auf einer höheren Programmebene gesorgt werden.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit konnte gezeigt werden, dass Smartcards mit *Java Card* Technologie eine geeignete Plattform für die Implementierung eines Sicherheitsmoduls für Smart Metering Gateways sind. Mittels der durchgeführten Implementierung konnten weiters die genauen Anforderungen an die zu verwendende Smartcard ermittelt werden. Die Integration in das *OGEMA*-Framework für Smart Metering Gateways zeigt die prinzipielle Nutzbarkeit mit solchen Systemen, im speziellen den möglichen Einsatz auf einer Embedded-System-Plattform mit GNU/Linux und ARM-Technologie. Das entwickelte Sicherheitsmodul demonstriert weiters die Integrierbarkeit in *OSGi*-Frameworks. Nicht nur *OGEMA*, sondern auch andere Smart Metering Frameworks (*OpenMUC*, siehe Abschnitt 2.4.1) basieren auf *OSGi*. Dadurch kann das Sicherheitsmodul, mit Anpassungen, auch in solchen Smart Metering Systemen eingesetzt werden. Die entwickelte Client-Bibliothek ist nicht auf die Nutzung des im Rahmen dieser Arbeit entwickelten Sicherheitsmoduls beschränkt. Durch die enge Anlehnung an das BSI-Schutzprofil sollte eine Kommunikation mit allen darauf beruhenden Sicherheitsmodulen möglich sein. Um den vollen Funktionsumfang zu nutzen, sind jedoch die in Abschnitt 6.1 durchgeführten Abwandlungen zu beachten.

Für den Einsatz des entwickelten Sicherheitsmoduls in einem Smart Metering System ist es empfehlenswert, eine Smartcard einzusetzen, welche den in Abschnitt 6.2 dargelegten Anforderungen entspricht, z. B. der *JCOP v2.4.1 R3* des Herstellers NXP. Weiters muss evaluiert werden, ob die in Abschnitt 6.1 beschriebenen Abwandlungen zum BSI Schutzprofil ein Hindernis für den Einsatz sind. In diesem Fall müsste das Sicherheitsmodul entsprechend angepasst werden.

Für die Nutzung des Sicherheitsmoduls in einem bestehenden Smart Metering Framework sollten die im BSI Schutzprofil für Smart Metering Gateways beschriebenen Anforderungen beachtet werden. Aus der Arbeit mit dem *OGEMA*-Framework wurde jedoch deutlich, dass die Anpassung eines bestehenden Systems an das Schutzprofil sehr schwer möglich sein wird. Eine vollständige Konformität mit dem Schutzprofil und eine eventuell notwendige Zertifizierung nach diesem, scheinen nur durch komplexe Restrukturierungen möglich. Soll die Neuentwicklung eines Smart Metering Gateways erfolgen und ist Konformität zum Schutzprofil gewünscht, sollte unbedingt schon bei dessen Planung darauf geachtet werden.

Anhang A: Installation und Ausführung auf dem *Raspberry Pi*

Eine Referenzimplementierung des *OGEMA*-Frameworks mit integriertem Sicherheitsmodul wird in dem Zip-Archiv *ogema_sm.zip* bereitgestellt. Um diese auf dem *Raspberry Pi* zu installieren und auszuführen, sind folgende Schritte notwendig:

- Download der aktuellen *Raspbian*-Distribution von http://www.raspberrypi.org/downloads
- Entpacken und auf eine SD-Karte schreiben:
- user@pc ~ $ sudo dd if=2014-01-07-wheezy-raspbian.img of=/dev/mmcblk0
- SD-Karte in den *Raspberry Pi* einlegen und mit Monitor, Tastatur, Netzwerk und der Stromversorgung verbinden,
- Das nach dem Start erscheinende automatische Konfigurationsmenü verlassen,
- Systemupdates durchführen:
- pi@raspberrypi ~ $ apt-get update && sudo apt-get dist-upgrade
- *ogema_sm.zip* auf den *Raspberry Pi* kopieren (Standardpasswort ist *raspberry*):
- user@pc ~ $ scp ogema_sm.zip pi@raspberrypi:/home/pi/ogema_sm.zip
- Per SSH mit dem *Raspberry Pi* verbinden:
- user@pc ~ $ ssh pi@raspberrypi -L8443:localhost:8443
- ogema_sm.zip entpacken:
- pi@raspberrypi ~ $ unzip ogema_sm.zip
- in das neue Verzeichnis wechseln:
- pi@raspberrypi ~ $ cd ogema_launcher
- OGEMA Framework starten:
- pi@raspberrypi ~ $ ./launch.sh
- Fehlermeldungen zu nicht gefundenen Artefakten können ignoriert werden (hängen mit fehlender Maven Installation zusammen), warten bis folgende Meldung erscheint:
- Select applet answer: ResponseAPDU: 2 bytes, SW=9000
- Auf dem PC in einem Browser folgende Seite öffnen:
- https://localhost:8443/login
- mit *root / root* einloggen.

Das *OGEMA*-Framework mit dem Sicherheitsmodul ist nun bereit und kann getestet werden.

Literatur

[BLEIER04] Bernhard Bleier. Integration von Smart Cards in das Feldbussystem LonWorks - Diplomarbeit. Jänner 2004. TU-Wien.

[BSI12-1] Bundesamt für Sicherheit in der Informationstechnik (BSI). BSI TR-03116 Technische Richtlinie für eCard-Projekte der Bundesregierung. Version 3.16. Fassung: August 2012.

[BSI13-1] Bundesamt für Sicherheit in der Informationstechnik (BSI). Protection Profile for the Gateway of a Smart Metering System (Smart Meter Gateway PP). Schutzprofil für die Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen. SMGW-PP Version 1.2 - 18. March 2013 (Final Release) Certification-ID: BSI-CC-PP-0073.

[BSI13-2] Bundesamt für Sicherheit in der Informationstechnik (BSI). Protection Profile for the Security Module of a Smart Meter Gateway (Security Module PP). Schutzprofil für das Sicherheitsmodul der Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen. SecMod-PP Version 1.0 – 18 March 2013 Certification-ID BSI-CC-PP-0077.

[BSI13-3] Bundesamt für Sicherheit in der Informationstechnik (BSI). Technische Richtlinie BSI TR-03109-2. Smart Meter Gateway – Anforderungen an die Funktionalität und Interoperabilität des Sicherheitsmoduls. Version 1.0 – 18.03.2013

[BSI13-4] Bundesamt für Sicherheit in der Informationstechnik (BSI). Technical Guideline TR-03110-3. Advanced Security Mechanisms for Machine Readable Travel Documents – Part 3 Common Specifications. Version 2.11. 12.07.2013.

[BSI13-5] Bundesamt für Sicherheit in der Informationstechnik (BSI). Technische Richtlinie BSI TR-03109-4. Smart Metering PKI - Public Key Infrastruktur für Smart Meter Gateways. Version 1.0. 18.03.2013.

[BSI13-6] Bundesamt für Sicherheit in der Informationstechnik (BSI). Technische Richtlinie. BSI TR-03109-1. Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems. Version 1.0. 18.03.2013.

[BUCH10] Johannes Buchmann. Einführung in die Kryptographie. 5. Auflage. 2010. Springer-Verlag Berlin Heidelberg. ISBN 978-3-642-11185-3.

[CAV09] Cavoukian, Ann. Privacy and Government 2.0: The Implications of an Open World. Information and Privacy Commissioner of Ontario, 2009.

[CLE08] Frances M. Cleveland. Cyber Security Issues for Advanced Metering Infrastructures. 2008. Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE. Pages 1-5. ISBN: 978-1-4244-1905-0

[CPW10] Cavoukian, Ann, Jules Polonetsky, and Christopher Wolf. "Smartprivacy for the smart grid: embedding privacy into the design of electricity conservation." Identity in the Information Society 3.2 (2010): 275-294.

[EKS11] Claudia Eckert, Christoph Kraus, Peter Schoo. Sicherheit im Smart Grid – Eckpunkte für Energieinformationsnetze. 2011. Stiftung-Verbundkolleg / Projekt Newise Nr. 90 2011.

[FLM10] Tony Flick, Justin Morehouse. Securing the Smart Grid: Next Generation Power Grid Security. 2010. Syngress Media. ISBN-10: 1597495700

[FZBW09] Implementierung einer offenen Smart Metering Referenzplattform-OpenMUC." 2009. ETG-Fachbericht-Internationaler ETG-Kongress 2009. VDE VERLAG GmbH.

[FZK12] Stefan Feuerhahn, Michael Zillgith, Dr. Robert Kohrs. Vergleich und Bewertung von Smart Metering Sicherheitskonzepten. 2012. VDE-Kongress 2012 05.-06.11.2012 in Stuttgart. VDE VERLAG GMBH Berlin, Offenbach. ISBN 978-38007-3446-7.

[INFINEON08] Infineon. JCLX80jTOP20ID Data Book. Revision 1.0. Edition 7/14/2008. Published by Infineon Technologies AG.

[ISO7816-3] INTERNATIONAL STANDARD ISO/IEC 7816-3. Third edition. 01.11.2006. Identification cards - Integrated circuit cards - Part 3: Cards with contacts - Electrical interface and transmission protocols. Reference number: ISO/IEC 7816-3:2006(E)

[ISO7816-4] INTERNATIONAL STANDARD ISO/IEC 7816-4. Third edition. 04.04.2013. Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange. Reference number: ISO/IEC ISO/IEC 7816-4:2013-04(E)

[ISO/IEC8825] INTERNATIONAL STANDARD ISO/IEC 8825-1. Fourth edition. 15.12.2008. Information technology -ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). Reference number ISO/IEC 8825-1:2008(E)

[KK99] Oliver Kömmerling, Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. 1999.

[LLS12] Xu Li, Xiaohui Liang, Rongxing Lu, Xuemin Shen, Xiaodong Lin, Haojin Zhu. Securing smart grid: cyber attacks, countermeasures, and challenges. August 2012. Communications Magazine, IEEE (Volume:50, Issue: 8). Pages: 38 – 45

[LOCH05] Manfred Lochter. ECC Brainpool Standard Curves and Curve Generation v. 1.0 19.10.2005

[NIST99] NIST. Recommended Elliptic Curves For Federal Government Use. July 1999.

[NXP11] P5Cx012/02x/40/73/80/144 family Secure dual interface and contact PKI smart card controller. Rev. 3.2 — 29 August 2011

[PS08] John Proos, Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. 2008. QIC 3 (No. 4) (2003) pp.317-344.

[RANKL06] Wolfgang Rankl. Chipkarten-Anwendungen: Entwurfsmuster für Einsatz und Programmierung von Chipkarten. 2006. Carl Hanser Verlag GmbH & Co. KG. ISBN-13: 978-3446404038

[RANKL08] Wolfgang Rankl. Handbuch der Chipkarten. 4. Auflage. 2008. Carl Hanser Verlag München Wien. ISBN-13: 978-3-446-40402-1

[RFC3647] Internet X.509 Public Key Infrastructure - Certificate Policy and Certification Practices Framework. The Internet Society. Network Working Group. November 2003.

[SB12] William Stallings, Lawrie Brown. Computer Security, Principles and Practice. 2012. Second Edition. Pearson Education. ISBN-13: 978-0-13-277506-9.

[SBKK12] Florian Skopik1,Thomas Bleier, Markus Kammerstetter, Georg Kienesberger. Smart Grid Security Guidance: Eine Sicherheitsinitiave für Intelligente Stromnetze. 2012. AIT Austrian Institute of Technology, Safety & Security Department, Technische Universität Wien, Institut für Technische Informatik, Technische Universität Wien, Institut für Computertechnik

[SHOR97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26:1484–1509, 1997.

[SHOUP98] Victor Shoup. Why chosen ciphertext security matters. 1998. IBM Research Report RZ3076.

[SPS11] Stephan Spitz, Michael Pramateftakis, Joachim Swodoba. Kryptographie und IT-Sicherheit. 2.Auflage. 2011. Vieweg+Teubner Verlag. ISBN-10: 3834814873

[UKNVS08] Markus Ullmann, Dennis Kugler, Heike Neumann, Matthias Vogeler and Sebastian Stappert. Password Authenticated Key Agreement for Contactless Smart Cards. 2008

[WHBSB11] Alexander Wiesmaier, Moritz Horsch, Johannes Braun, Falko Strenzke Johannes Buchmann. An Efficient Pace Implementation. 2011. ASIACCS '11 Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. Pages 176-185. ACM New York. ISBN: 978-1-4503-0564-8.

Internetreferenzen

[BGB11] Bundesgesetzblatt für die Republik Österreich. 339. Verordnung: Intelligente Messgeräte-AnforderungsVO 2011 – IMA-VO 2011. Verfügbar unter: http://www.e-control.at/portal/page/portal/medienbibliothek/strom/dokumente/pdfs/IMA-VO_BGBL_2011_II_339.pdf [abgerufen am 01.02.14]

[BGB12] Bundesgesetzblatt für die Republik Österreich. 138. Verordnung: Intelligente Messgeräte-Einführungsverordnung - IME-VO. Verfügbar unter: http://www.e-control.at/portal/page/portal/medienbibliothek/service-beratung/dokumente/pdfs/Intelligente-Messgeraete-Einfuehrungsverordnung%E2%80%93IME-VO.pdf [abgerufen am 01.02.14]

[CCID] CCID free software driver. [ONLINE]. Verfügbar unter: http://pcsclite.alioth.debian.org/ccid.html [abgerufen am 05.02.2014]

[EC09] Amtsblatt der Europäischen Union. RICHTLINIE 2009/72/EG DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 13. Juli 2009 über gemeinsame Vorschriften für den Elektrizitätsbinnenmarkt und zur Aufhebung der Richtlinie 2003/54/EG [ONLINE] Verfügbar unter: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:211:0055:0093:DE:PDF [abgerufen am 25.02.2014]

[EDK12] Amtsblatt der Europäischen Union. EMPFEHLUNG DER KOMMISSION vom 9. März 2012 zu Vorbereitungen für die Einführung intelligenter Messsysteme (2012/148/EU) [ONLINE] Verfügbar unter: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2012:073:0009:0022:DE:PDF [abgerufen am 25.02.2014]

[EDL2006] RICHTLINIE 2006/32/EG DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 5. April 2006 über Endenergieeffizienz und Energiedienstleistungen und zur Aufhebung der Richtlinie 93/76/ EWG des Rates [ONLINE] Verfügbar unter: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:114:0064:0064:DE:PDF [abgerufen am 25.02.2014]

[ESG06] European SmartGrids Technology Platform. Vision and Strategy for Europe's Electricity Networks of the Future. [ONLINE] Verfügbar unter: http://ec.europa.eu/research/energy/pdf/smartgrids_en.pdf [abgerufen am 01.12.2013]

[ESG12] SmartGrids European Technology Platform for the Electricity of the Future. [ONLINE]. Verfügbar unter: http://www.smartgrids.eu/documents/TRIPTICO%20SG.pdf [abgerufen am 01.12.2013].

[GITH1] GitHub Repository der verwendeten jCardSim Implementierung. [ONLINE]. Verfügbar unter: https://github.com/mali1/jCardSim [abgerufen am 05.02.2014]

[JCARDSIM] jCardSim - Java Card Runtime Environment Simulator. [ONLINE]. Verfügbar unter: http://jcardsim.org [abgerufen am 05.02.2014]

[LICEL] Homepage der LICEL LLC. [ONLINE]. Verfügbar unter: http://licel.ru [abgerufen am 05.02.2014]

[NXP12] Java CardTM OS for NXP's SmartMX family of secure microcontrollers. Datenblatt der Smartcard NXP JCOP v2.4.1 R3. [ONLINE]. Verfügbar unter: http://www.mifare.net/files/1813/4018/4219/Java%20Card%20OS%20for%20Smart%20MX.pdf [abgerufen am 05.02.2014]

[OGEMA09] Introduction to the OGEMA Framework. Rev:28.08.2009 C:29/02/2012 09:02:02 [ONLINE] Verfügbar unter: http://www.ogema.org/downloads/ogema-framework-introduction.pdf [abgerufen am 11.10.2013]

[OGEMA12] OGEMA Technology in Brief - How OGEMA works and what it can do for you [ONLINE] Verfügbar unter: http://www.ogema.org/downloads/ogema_technology-brief.pdf [abgerufen am 11.10.2013]

[ORACLE11] JAVA CARD CLASSIC PLATFORM SPECIFICATION 3.0.4. [ONLINE]. Verfügbar unter: http://www.oracle.com/technetwork/java/javacard/specs-jsp-136430.html [abgerufen am 10.12.2013]

[ORACLE13] Comparing JVMs on ARM/Linux. [ONLINE]. Verfügbar unter: https://blogs.oracle.com/jtc/entry/comparing_jvms_on_arm_linux [abgerufen am 05.02.2014]

[ORTIZ03] C. Enrique Ortiz. An Introduction to Java Card Technology – Part 1. May 29 2003. [ONLINE]. Verfügbar unter: http://www.oracle.com/technetwork/java/javacard/javacard1-139251.html [abgerufen am 14.02.2014]

[RASP12] Raspberry PI Technical, help and Resource Documents. [ONLINE]. Verfügbar unter: http://www.raspberrypi.org/technical-help-and-resource-documents [abgerufen am 02.12.2013]

[REN10] Stephan Renner. Smart Metering und Datenschutz - Möglichkeiten zur Umsetzung des 3. EU-Binnenmarktpakets in Österreich. 2010. [ONLINE] http://www.energyagency.at/fileadmin/dam/pdf/veranstaltungen/SmartMetering_Konferenz_Handout.pdf Verfügbar unter: http://www.selma-project.de/workshop2/tanw1_1.pdf [abgerufen am 25.02.2014]

[RUIM11] Java Card: Programming Guidelines and Best Practise [ONLINE]. Verfügbar unter: http://ruimtools.com/doc.php?doc=jc_best [abgerufen am 10.12.2013]

[SUN03] JAVA CARD[TM] PLATFORM SPECIFICATION 2.2.1. [ONLINE]. Verfügbar unter: http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javame-419430.html#java_card_kit-2.2.1-oth-JPR [abgerufen am 02.12.2013]

[SCHAUB03] Thomas Schaub. Selma – Technologie und Anwendung. Landi+Gyr. [ONLINE] Verfügbar unter: http://www.selma-project.de/workshop2/tanw1_1.pdf [abgerufen am 25.02.2014]

[SGA10] Smart Grids Austria – Die österreichische Technologieplattform zum Thema Smart Grids. Smart Grids. [ONLINE]. Verfügbar unter: http://www.smartgrids.at/smart-grids/ [abgerufen am 01.12.13]

[SGMS13] Ergebnisse & Erkenntnisse aus der Smart Grids Modellregion Salzburg. Mai 2013. [ONLINE] Verfügbar unter: http://www.smartgridssalzburg.at/fileadmin/user_upload/downloads/SGMS_Ergebnisse_Erkenntnisse_05-2013.pdf [abgerufen am 01.12.13].

[VDE09] Synchonous Modular Meter SyM2 Pflichtenheft 1.03 06.10.2009, Aktualisierung 10.11.2010 [ONLINE] Verfügbar unter: http://www.vde.com/de/fnn/arbeitsgebiete/messwesen/Sym2/Infomaterial/documents/sym2_ph_103_091006.pdf [abgerufen am 25.02.2014]

[VDE10] FNN Forum Netztechnik / Netzbetrieb im VDE. Latenheft EDL – Elektronische Haushaltszähler – Funktionale Merkmale und Protokolle. [ONLINE] Verfügbar unter: http://www.vde.com/de/fnn/arbeitsgebiete/messwesen/documents/fnn_lastenheft-edl_1-0_2010-01-13.pdf [abgerufen am 25.02.2014]

Abbildungsverzeichnis

Tabellenverzeichnis