Graz University of Technology

Institute of Neural Engineering
Stremayrgasse 16/IV
8010 Graz, Austria

Master Thesis

Security concept and prototype for patient-centred management of health data and services

to achieve the degree of
Master of Science

Author: Armin Johann Haas, BSc

Supervisor: Priv.-Doz. DI Dr. Günter Schreier
Advisor: DI Dr. Walther Pachler

February 05, 2018

This master thesis has been conducted
in cooperation with:

AIT Austrian Institute of Technology GmbH

Center for Health & Bioresources

Digital Health Information Systems

Reininghausstraße 13/1, 8020 Graz, Austria

Supervisor: Priv.-Doz. DI Dr. Günter Schreier

Infineon Technologies Austria AG

Development Center Graz

Babenbergerstraße 10, 8020 Graz, Austria

Supervisor: DI Dr. Walther Pachler

Content

Abstract

The quality of medical decision processes strongly depend on the access to essential medical and health data, but compared to the current technological possibilities, many data sources are still not used to its full extend for medical purposes (e.g. fitness account data). This condition is also due to a lack of data infrastructures capable of managing sensitive data like health records.

To strengthen data-driven methods for decision support systems and to tap into the potential of unused health data, the aim of the present master thesis was the development of a novel patient centred data infrastructure to provide medical professionals with medical, health and lifestyle data. High security and the empowerment of the patient should reduce privacy concerns, improve existing solutions and raise the motivation of patients to share their data.

To provide a system that can be used in everyday life, it was an essential goal to use smartphones as the main hardware components for the operational use. Therefore, near field communication (NFC) tags were used to realize the data environment by storing the patient's personal health data on contactless chip cards. The necessary data security, access rights management and file structure was designed using the open CIPURSE™ standard for smartcard technology.

For the implementation, a prototypical Android application was developed with the main task to introduce the CIPURSE™ functionality on mobile devices. This was possible with the integration of a special kind of microSD card, capable of storing a secure access module (SAM), which is essential for a secure CIPURSE™ application. Additionally, a personalization device was build, which was realized with a Raspberry Pi single-board computer. This device was needed to handle the initialization and personalization processes of the NFC- and smartcards in use.

The CIPURSE™ application, in combination with the Android application and the Raspberry Pi based personalization device, provided all requirements for a successful implementation of a novel data infrastructure.

In conclusion, the developed system provides new capabilities to manage personal health data with a very high level of privacy protection and patient control.

Zusammenfassung

Die Qualität medizinisch-diagnostischer Entscheidungsprozesse hängt stark vom Zugang zu hierfür essentiellen Daten ab. Im Vergleich zu den derzeitigen technologischen Möglichkeiten werden viele Datenquellen für medizinische Zwecke jedoch nicht in vollem Umfang genutzt (z. B. Fitness-Daten). Dieser Umstand ist auf die fehlende Dateninfrastruktur zurückzuführen, die es ermöglichen würde, sensible Gesundheitsdaten zu verwalten.

Als Ziel der vorliegenden Masterarbeit galt die Entwicklung einer neuartigen, patientenzentrierten Dateninfrastruktur, um datengestützte Methoden für medizinische Entscheidungsprozesse zu stärken. Diese sollte in der Lage sein, das Potenzial nicht genutzter Gesundheitsdaten zu erschließen, um Fachpersonal mit Medizin-, Gesundheits- und Lifestyle-Daten zu versorgen. Hohe Datensicherheit und eine ausgeprägte Eigenermächtigung der PatientInnen sollten dabei helfen, Bedenken hinsichtlich des Datenschutzes zu reduzieren und PatientInnen in weiterer Folge dazu animieren, hilfreiche Daten mit medizinischen BetreuerInnen zu teilen.

Ebenso vorrangig war es, die Verwaltung jener Dateninfrastruktur auf die Verwendung von Smartphones auszulegen, um die alltägliche Nutzung zu erleichtern und Kosten für die Implementierung gering zu halten. Aus diesem Grund wurden Near Field Communication-, respektive, NFC-Tags verwendet, um die Datenumgebung zu realisieren und die persönlichen Gesundheitsdaten der PatientInnen zu speichern. Die notwendige Datensicherheit, Zugriffsrechteverwaltung und Dateistruktur wurde hierbei mit Hilfe des offenen CIPURSE™ Standards für Smartcards realisiert.

Für die Implementierung wurde ein Android App Prototyp entwickelt, mit dem Ziel, die vollständige CIPURSE™ Funktionalität auf mobilen Geräten zu ermöglichen. Möglich wurde dies wurde durch die Integration einer speziellen microSD-Karte. Diese ist in der Lage, ein Secure Access Module (SAM) zu speichern, welches eine unerlässliche Komponente für eine sichere CIPURSE ™ -Anwendung darstellt.

Zusätzlich wurde eine Personalisierungsmaschine, basierend auf einem Raspberry Pi Single-Board Computer, gebaut. Diese war unabdingbar für die Handhabung der Initialisierungs- und Personalisierungsprozesse der verwendeten NFC- und Smartcards.

Mit der CIPURSE™ Anwendung, der Android App und dem Raspberry Pi basierten Personalisierungsgerät wurden alle Voraussetzungen für eine erfolgreiche Implementierung dieser neuartigen Dateninfrastruktur geschaffen.

Zusammenfassend bietet das entwickelte System neue Möglichkeiten zur Verwaltung persönlicher Gesundheitsdaten mit einem sehr hohen Maß an Datenschutz- bzw. PatientInnenkontrolle.

Acknowledgment

Special thanks go to my supervisors, Univ.-Doz. Dipl.-Ing. Dr. Günter Schreier from the Austrian Institute of Technology (AIT) and Dipl.-Ing. Dr. Walther Pachler from the Infineon AG, for all the advice and support during the process of writing this master thesis.

I would also like thank and mention my colleague Dipl.-Ing. Martin Tupy at Infineon for his counseling, especially concerning the topic of CIPURSE™.

Last but not least, I am thanking my supervisor Dipl.-Ing. Harald Witschnig and the entire colleagues at AIT and Infineon's "Contactless System Engineering" and "Application Engineering" department for their support.

1. Background

1.1. Motivation

Studies have shown that information and communications technology (ICT) can improve the quality and efficiency of healthcare [1]. Health information technology (HIT) can prevent medical errors, reduce costs and simplify the administrative processes in healthcare organizations [2]. Therefore, I see a lot of potential when it comes to the symbioses of new information technology and current and future healthcare systems.

In medicine the access to essential medical and health data is important to provide patients with high quality treatment and diagnosis. New data sources can potentially improve treatments and diagnoses and provide new possibilities for data-driven decision support systems in support of preventive healthcare.

In contrast to the technical capabilities provided by wearables and mobile devices like smartphones, data sources for health professionals are still quite limited and not used to its full extent. For example, many people are using fitness tracking platforms to collect data about their sports activities. This data could hold potential health information to prevent future diseases but, up to now, is mainly used privately without any medical purpose.

Including such data sources into the healthcare process could potentially help to further personalize medical care and might support prevention and medical treatment.

Many promising data sources do already exist but there is a lack of data access possibilities. On the one hand standardised data infrastructures are still quite limited or not available at all. On the other hand many patients are very sceptical and fear personal data loss and a lack of privacy.

These are two problems that need to be addressed to facilitate better health data usage in the future. To solve these issues, a new secure data infrastructure is needed where the patient has full control over the data and, thereby, the confidence to use such services. This is also crucial when privacy protection laws are involved.

Predictive modeling / Data-driven decision support

Predictive modeling and analytics use computing power and statistical methods to identify patterns and trends and to forecast events by processing huge amounts of data. Such data-driven methods for decision support can be particularly relevant in a medical context. The available information generated by medical research and day-to-day healthcare has already reached a dimension that makes it impossible to be processed by humans, while big data analytics and predictive modeling can lead to new undetected associations.

These new insights can trigger new research and be applied to individual patients by combining them with past treatment and current health data of the individual. Physicians can thereby be supported during the diagnosis process and be provided with new predictions about the risk of diseases progression or changes in the health status of individual patients.

Data can be used by such models to detect sources of risk and to trigger targeted interventions to prevent further harm by providing the necessary treatment in time. Contemporaneously unnecessary hospitalizations can be avoided, resulting in reduced costs for healthcare systems and a decreased burden to patients. Data-driven decision support systems can aid a physician during complex decision-making processes to differentiate between high-risk and lower-risk patients, with the ultimate goal of better outcomes for the patients and lower healthcare costs [3].

Besides health improvements, such methods can also improve organizational aspects and processes in hospitals, for example, when hospital readmissions can be predicted or prevented [3].

In summary, data-driven decision support can be expected to improve the quality of the clinical decision-making process and to help using resources efficiently, thus, reducing the costs of healthcare systems.

1.2. The Aim of the Thesis

The main goal for this thesis was to create a new data infrastructure that considers and provides the following four aspects:

1. New medical / health / lifestyle data sources
2. High level of information security and access rights management
3. A patient centred concept
4. Easy-to-use implementation for everyday use at reasonable costs

The intention was to create a proper concept and to develop a prototype implementation for the required hard- and software components.

1.2.1. Goals

In the following segment, the goals of the master thesis are discussed into detail. The four mentioned aspects of the main goal have to fulfil the defined requirements which were elaborated by the Infineon AG and the AIT

New medical / health / lifestyle data sources

The data environment should be able to provide access to multiple data sources which are not yet covered by traditionally established services. The sources should not be constrained to medical data alone. It should be possible to link to new sources like lifestyle data from fitness tracking platforms or telehealth systems.

Another requirement was to provide and protect very critical and highly sensitive data like genetic information. The attempt in this thesis is to eliminate some of the concerns people have against the analysis of genetic information by creating a secure and more comfortable data environment. This means that the infrastructure should not only provide access to new data, it should also motivate users to supply health professionals with more useful data. Finally, the possibility of combining all this information should be provided.

Data security and access rights management

Since the data, that is stored and handled by the defined data environment, is personal, up-to-date data security is essential. Therefore, the encryption of the stored data and a secure data exchange is mandatory for such a system. Additionally the restricted access to the data has to be manageable by providing the possibility to adjust the access rights to the type of information stored.

Overall it has to be ensured that authorized persons have access to sensitive data and others don’t. At the same time it should be possible to create individual access rights for different kinds of health professionals.

Patient centred system

The patient centred concept should give the patient (often called “data subject” in the corresponding data protection regulations) full control over his/her personal information. By being the only one in control of the access rights, the patient should be able to allow or restrict the data access to single datasets. The possibility of combining these datasets together should only be provided by the approval of the data subject.

This direct control, in combination with strong security, should dissolve privacy concerns and provide higher confidence to share helpful data for new technologies like data-driven decision support systems.

Easy implementation for everyday use

Finally, the data environment should be able to be easily introduced into people’s everyday lives without the need of a new complex hardware infrastructure.

1.2.2. Basic Idea

The basic idea for such a new health data infrastructure was an implementation based on NFC tags. These contactless chip cards can be used to store different kind of health data and they are low-cost and very portable, thanks to their small size and flexible form factor.

Another huge advantage is that NFC tags can easily be included into our everyday life, because of the high distribution of NFC supportive smartphones. Therefore, NFC represents a cost-effective possibility to build the envisaged infrastructure with only minor needs for new hardware components. Furthermore, the low range contactless communication itself is a feature for security sensitive services, since it is impossible to interact with it over higher distances.

Additionally, to optimally utilize the functionality of a smartphone, it can not only be used as a reader device, it could also directly process health data on the phone or via online services.

By using contactless chip cards, the card owner would always be in full control over the data, since the information is not stored on a server or any other service. The data would be stored directly on the NFC tag itself. This implies that the card owner has to hand-out the NFC tag willingly to provide a health professional with access to his/her personal data.

The question that needs to be solved is: how can a proper security and access control management be provided using NFC? Fortunately, recently, a new open standard called CIPURSE™ became available which is capable of fulfilling all these requirements. CIPURSE™ will be discussed adequately in later sections of this thesis.

1.2.3. Use Case

Based on the fundamental idea of using NFC tags as storage devices in a data infrastructure for medical / health data, a use case for this thesis had to be defined, capable to show the possibilities and the potential of such a technology.

There are several possible scenarios where a secure NFC based data infrastructure could be used. For example it would be feasible to integrate contactless chip cards into bendable paper strips. Therefore, it could be used as an alternative to QR code wristbands which are frequently utilized in hospitals to identify patients.

To encounter the current lack of open patient-determined health record systems, this thesis will concentrate on a patient-centred data infrastructure use case that will allow a patient to collect and manage his/her own personal data. The patient will be the only one capable of providing the personal information to health professionals, e.g. for further data processing in data-driven decision support systems.

The most effective way to make sure that the user/patient is the only person in control over the full dataset is by ensuring that he or she is the exclusive owner. This would mean that each individual health professional owns only selected parts of a patient's potentially diverse dataset, while the patient is able to collect and combine multiple records from different health professionals.

This concept can be imagined, with a patient collecting single jigsaw puzzle pieces and fitting them together, to create a picture of himself that can be shown to anyone he wants to.

For example, a physician holds information about allergies of an individual patient and a genetic analytics company holds the patient's pharmacogenetic information. If the patient collects these two datasets on his NFC card he could provide it to a pharmacist. By combining this data the pharmacist could be empowered to help selecting the most efficient drug for the patient.

The underlying idea is that the physician and the genetic analytics lab are not aware of a second dataset and, therefore, might miss insights that can be derived by the combination of the entire information. Only the health professional who is trusted by the patient can join and use the personal information for new insights. This patient empowering approach is supposed to strengthen the patients trust into the data infrastructure and might provide an additional electronic health record component in Austria, next to the established ELGA system.

The stored information could be of any kind, the only limitation is the available memory, although such limitations can be overcome by using online of cloud services. By storing login data on a NFC card, the access to almost unlimited amounts of data could be realized.

The use case seen in Figure 1.1 is simplifying a possible scenario by limiting the bound roles to an administrator, health professionals and the card owner, i.e. the patient. The basic interactions between these roles can be seen in the figure.

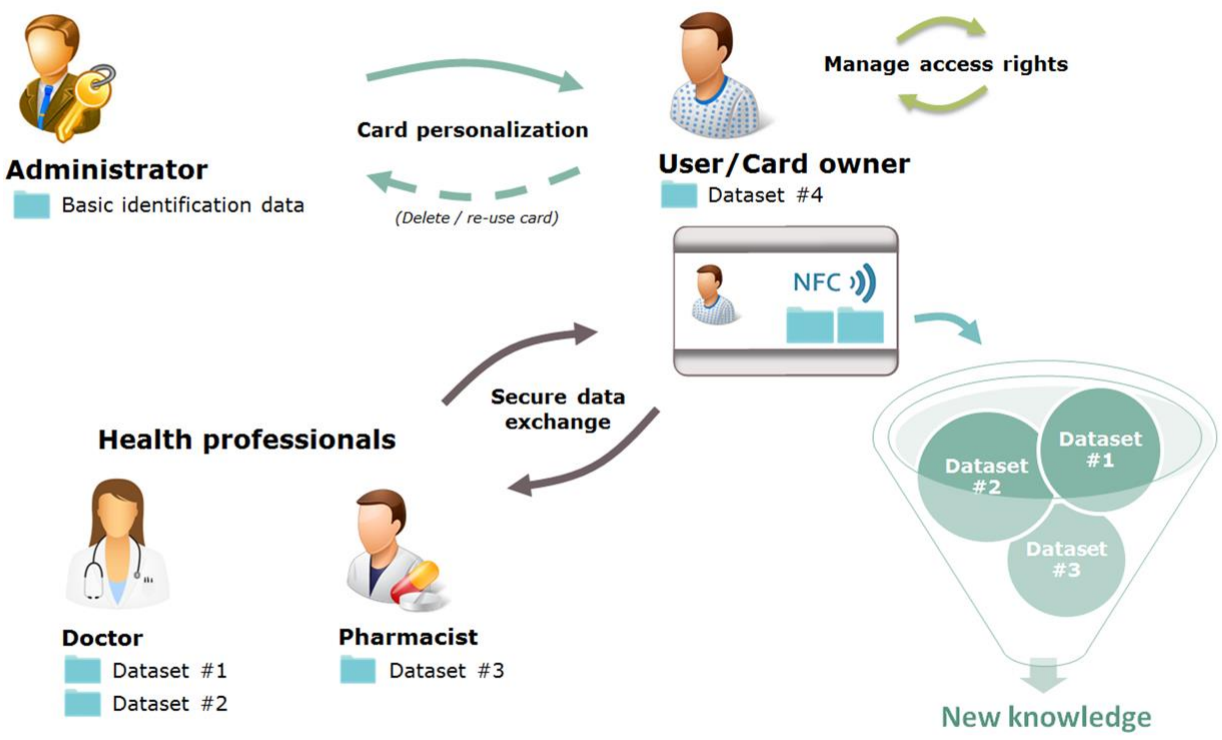


Figure 1.1: Use case for a NFC based health data infrastructure

In summary, the administrator is personalizing and delivering a NFC card to the user/patient. The card owner and the health professionals can both store information onto the NFC card memory, while the card owner alone is capable of managing the access rights on his or her card.

A more detailed description of the scenario and its implemented application are documented in the results of the thesis.

1.3. **European Privacy Protection Law**

Data security is crucial for such a use case and there are at least three important aspects that need to be considered concerning data protection laws. This section gives an overview to the most important aspects for handling personal data in the healthcare context in Europe. These are regulated by the general data protection regulation (GDPR), which will come into full effect as of May 25th, 2018 [4].

The following legal aspects are only addressing some issues concerning this particular use case and do not cover the whole subject-matter.

The most important issue from this thesis perspective is the processing of personal data. Here we have to differentiate between personal data in general (sensitive information connected to an identified natural person) and special categories of personal data (highly sensitive and very critical personal data like genetic or biometric data).

The processing of personal data is handled in Chapter 1, Article 6 (Lawfulness of processing):

"1. Processing shall be lawful only if and to the extent that at least one of the following applies:

(a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes; … "

The processing of special personal data is handled in Chapter 1, Article 9 (Processing of special categories of personal data):

"1. Processing of personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade-union membership, and the processing of genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person's sex life or sexual orientation shall be prohibited.

2. Paragraph 1 shall not apply if one of the following applies:

(a) the data subject has given explicit consent to the processing of those personal data for one or more specified purposes, except where Union or Member State law provide that the prohibition referred to in paragraph 1 may not be lifted by the data subject; … "

In both cases the data subject has to give explicit consent to the processing of the data. Therefore, it has to be made sure that a possible scenario includes a concept that allows a person to give consent to such data processing. Another approach would be to make sure that the used data is anonymous.

The next issue is the data portability, which is handled in Section 3, Article 20 (Right to data portability):

“1. The data subject shall have the right to receive the personal data concerning him or her, which he or she has provided to a controller, in a structured, commonly used and device-readable format and have the right to transmit those data to another controller without hindrance from the controller to which the personal data have been provided, where: (a) the processing is based on consent pursuant to point

(a) of Article 6(1) or point (a) of Article 9(2) or on a contract pursuant to point … ”

Therefore, it is lawful to hand the data over to the data subject itself. The data subject is also allowed to further transfer the data to, for example, health professionals. Once again it has to be made sure that the data subject consents to all data processing when taking part in such a use case scenario.

In summary, the most important aspect for handling personal data is the explicit consent of the data subject.

2. Methods

2.1. Near Field Communication (NFC)

NFC builds on RFID technology. It describes a set of protocols that allow contactless data exchange between two NFC devices over a distance of several centimetres. The NFC technology was developed by Sony and Philips and is described in the "ISO/IEC 18092" and "ECMA 340" standard. In 2004 the NFC Forum was founded by Sony, Philips and Nokia with the goal to specify and further promote and spread NFC. [5]

The data transfer is based on the same physical principle as the RFID technology which is using electromagnetic induction between two loop antennas to exchange information. A reader device is emitting an electric current into a coil to create a magnetic field. This field is inducing current into the coil of a transponder and transformed into electrical impulses to communicate data back to the reader (Figure 2.1). The radio frequency used for such an interaction is 13.56MHz.

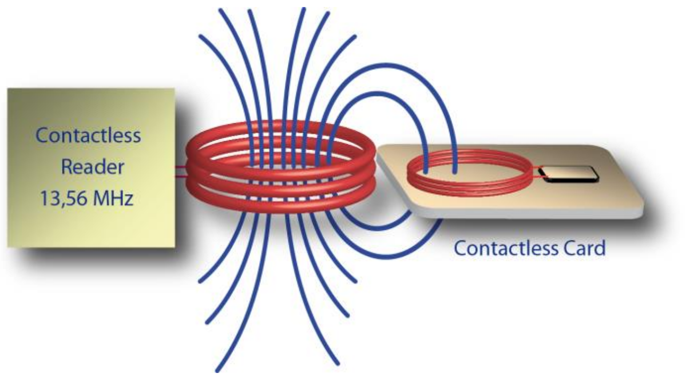


Figure 2.1: Physical principle of the RFID data transfer [F1]

The main difference between RFID and NFC is that an NFC reader can not only act as a reader and initiator, it can also act as a passive tag (or target). Therefore, the scope of application for NFC is mainly found in mobile and handheld devices.

Meanwhile the NFC Forum already counts more than 100 members, including well-known companies such as Microsoft, Visa, Infineon, NXP, Samsung or Google. Today, NFC is used in common services like contactless payment systems, electronic ticketing or other applications like the quick pairing of two Bluetooth devices. Thanks to this widespread acceptance it became a common feature in today's smartphones. [6]

2.1.1. Operating Modes

A NFC communication involves two NFC devices, an initiator and a target. They have to be placed close next to each other to initiate the communication using the “ISO/EC 18000-3” air interface. The data rates that can be reached are 106, 212 and 424Kbits per second and like mentioned before, the used frequency is 13,56MHz.

The device that is initiating the connection is the active device and is called initiator, the responding device is acting passive and is called the target. The initiator has to be an active device with a power supply since it has to actively create a magnetic field. The target although can be an active or a passive device, this depends on the used operation mode.

NFC provides three types of operating modes, the reader/writer mode, the peer-to-peer mode and the card emulation mode. The different modes use their own standards and communication interfaces. Which one is in use depends on the participating communication partners and explained in Figure 2.2. [6]

It shows that NFC allows the communication between an active and a passive device (e.g. smartphone / NFC tag), between two active devices (e.g. smartphone / smartphone) and additionally an active device can also act as a passive one using the card emulation mode.

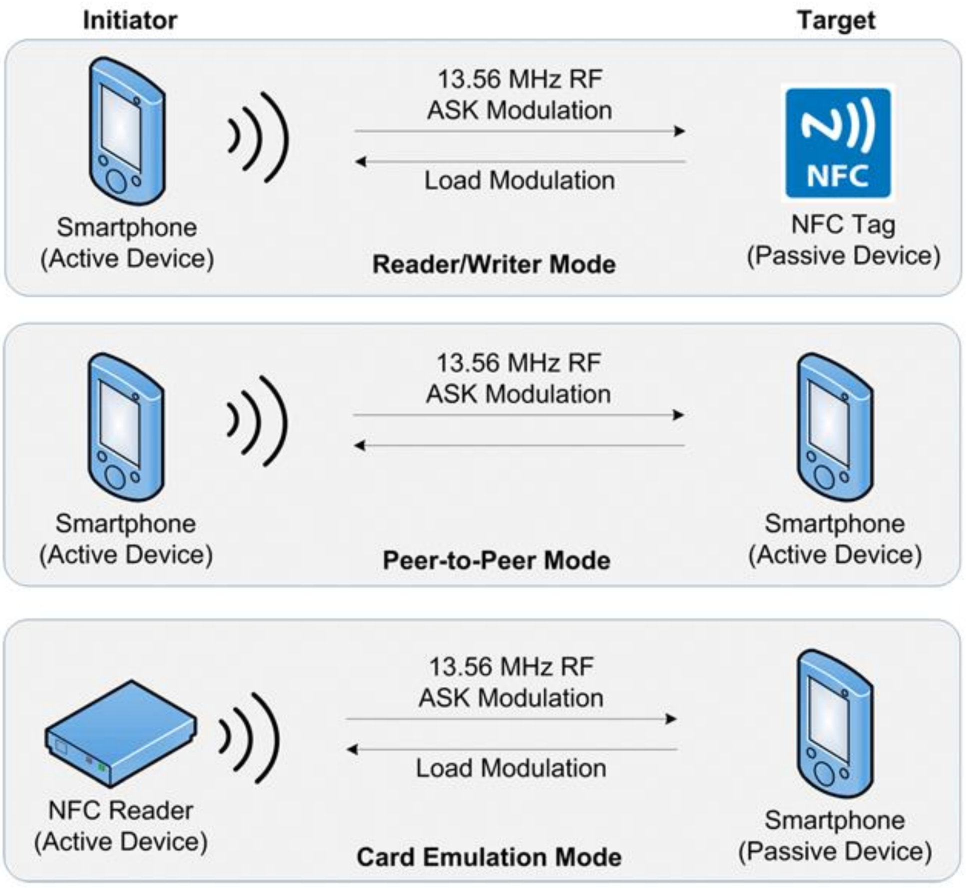


Figure 2.2: NFC operating modes [6]

2.1.2. Secure NFC

To allow the usage of NFC for services like payment and ticketing systems it has to be assured that transactions are handled in a protected environment with the necessary data security. These security features, such as encrypting of the contactless communication interface, are provided by the Secure NFC technology.

Additionally to a NFC controller that is needed in every standard NFC device, a secure element (SE) is essential for the functionality of Secure NFC. The SE provides secure transactions between NFC devices and enables secure storage. [7]

There are multiple options for the integration of an SE in a NFC device like a smartphone. They can be embedded into the hardware itself, SIM card based or via host card emulation (HCE). Another novels form factor of the secure element will be part of this thesis and discussed in another section.

The basic setup of a device capable of handling Secure NFC with a SE can be seen in Figure 2.3.

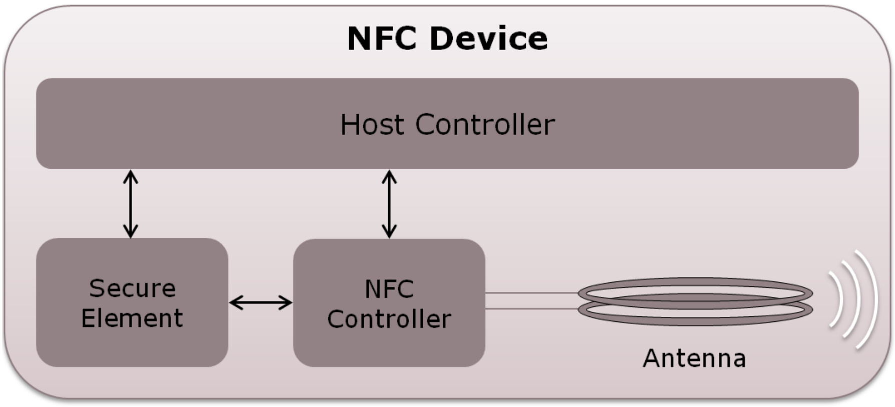


Figure 2.3: NFC device with a secure element

Thanks to the expansion to the NFC functionality the range of possible applications is enhanced massively by allowing security critical services.

2.2. **Open-Source**

Open-source implies the concept of an open exchange and collaborative participation and is best known in the context of software development. It provides public accessibility to, for example software source code, so that people are able to view, modify and share the code. This enables a community-oriented development, transparency and rapid prototyping [8].

All this contribution leads to an amplified innovation process and shows a lot of potential for further improvement [9] while patents can be an obstacle for innovation processes [10]. At the same time safety and security can be enhanced by giving more people the possibility to inspect the features and the design of the devices [11]. The development of encryption algorithms for example is a very complex procedure and it is said that the best algorithms are developed in the open-source projects [12].

Today open-source projects can combine the software and hardware development and the term delivers a broad set of values. It does not constrain itself to software development and to the access to the source code. Altogether there are 10 criteria for the distribution terms of open-source software alone.

There is also open-source licensing and in case of open-source this means that the product is licensed by the copyright holder with a license that provides the right to change, improve and distribute the product to everyone [8].

Some examples for open-source software licenses are [13]:

<u>“Berkeley Software Distribution” (BSD)</u>

> Written at the University of California Berkley in 1970 and meant to allow the commercial use of the licensed software.

<u>“GNU Public License” (GPL)</u>

> Richard Stallman introduced GPL in 1991. Software licensed under GPL allows using, viewing and modifying of the software. It does not allow marketing the software or to link to exclusive rights.

<u>“Lesser General Public License” (LGPL)</u>

> LGPL was introduced by the free software foundation in 1991. It is similar to GPL, although it allows to link to a library or an LGPL/GPL program.

<u>“Mozilla Public License” (MPL)</u>

> Provides a combination of the BSD and GPL licenses and was introduced by the Netscape Communications Corporation.

2.2.1. Software

"*Open-source software is computer software for which the product and the source code is available for free. This permits users to freely use, change, and improve the software, and to redistribute it in modified or unmodified forms. It is very often developed in a public, collaborative manner*" [14].

The concept of open-source started with open-source software, the term is therefore well known and established when it comes to software development. The range of open-source software (OSS) products is huge and many advantages have already been mentioned before.

Open source software is used in many areas, privately and for non-commercial applications. In 2008 the Standish Groups reported that the usage of open-source software saves the consumers about $60 billion per year [15]. Although it is worth mentioning that the aspect of the lower costs is just one advantage of open-source solutions.

2.2.1.1. Linux

Linux is the name of an open-source operating system and is also designated to a group of open-source software operating system distributions that are built around the Linux kernel. The kernel was first released in 1991 and is the defining component of every Linux distribution. [16]

Linux is a modular operating system and is developed by software developers around the world. Many companies, volunteers and organisations are involved in the development process. Therefore, many adaptations are possible, which leads to an extensive supply of Linux distributions. A distribution combines the Linux kernel with different software to an operating system to meet specific demands and purposes. [17]

Originally Linux was developed for personal computers, but over time it was ported to several different platforms. One of the best known distributions is the Linux kernel based Android operating system for smartphones.

Android

Android is an open platform and operating system developed by Google and is intended for mobile devices like smartphones and tablets. It is a Linux distribution which is partially based on the programming languages Java and C, this is quite uncommon for most Linux distributions. [18]

According to market share statistics offered by providers like the "International Data Corporation" (IDC), Android is the most spread mobile operating system by far. [19]

Raspbian

Raspbian is another Linux distribution. It is based on the Debian distribution and is specifically designed for the Raspberry Pi single-board computer. It is optimized for the

Raspberry Pi hardware and provides all the necessary programs, packages and software-bundles. [20]

2.2.2. Hardware

"*Open-source hardware is hardware whose design is made publicly available so anyone can study, modify, distribute, make and sell the design or the hardware based on that design.*" [21].

Today there are several open-source hard- and software solutions based on low-cost single-board computers like the Raspberry Pi or the Arduino. Some open-source projects include healthcare sensors and are specifically designed for an active medical use.

Around several open devices large communities arose, which can provide great support and make many open software libraries available to simplify feasibility. The openness and the accessible design specifications motivate developers to realize new projects and interoperability to variable hardware and devices. Next to the benefits of a low price and a high flexibility, there are more arguments pointing towards the usage of open-source products, especially when it comes to a medical use case.

It can take several years for a medical device to get through the clinical trials, which could be a long time for small companies. Such additional expenses could be distributed between multiple incorporated businesses during an open-source project. Additionally it is also worth mentioning that many of these devices are already CE and FCC certified.

There are already several medical applications based on single-board computers like the Raspberry Pi and the Arduino. They include CT scanners, syringe pumps, health sensor platforms, blood pressure monitors and many more. [21]

2.2.2.1. Raspberry Pi

The Raspberry Pi is a single-board computer with the size of a credit card. It is a low-cost, complete functional computer that fits on one circuit board containing all required features. It can be used like a personal computer, but at the same time it provides many more possibilities when it comes to system development and prototyping.

A wide range of additional hardware, extensions and shields in addition to direct accessible processor pins and open-source software like Linux are the main advantages. There are also several operating systems available for the Raspberry Pi, but the most common is the open Linux distribution Raspbian. [22]

If the Raspberry Pi itself is open-source hardware is controversial, since not all hardware components are open-source. This concerns the "System-on-a-Chip" (SoC) module provided by Broadcom.

Most importantly the Raspberry Pi provides a fast, open and stable Linux platform that is portable, low-cost and has a good availability. Additionally it is also CE and FCC certified.

Since the first Raspberry Pi was released, several versions were added to the portfolio. The newer versions include performance updates or some additional hardware features like on-board Bluetooth. Usually they share a very similar design. Figure 2.4 shows an illustration of the in 2015 released Raspberry Pi 3 Model B with all the present connections and interfaces.

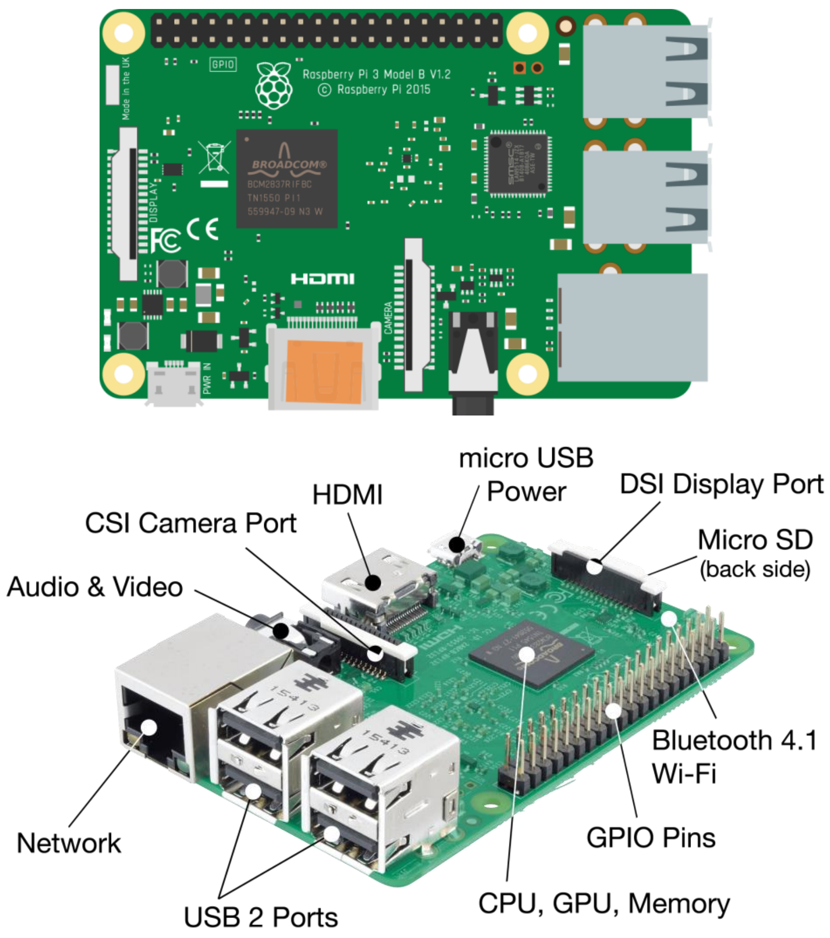


Figure 2.4: Raspberry Pi 3 Model B [F2]

One essential feature that’s shared by all the Raspberry Pi versions is the SD or microSD card slot. The Raspberry Pi is using a SD/microSD card as the main storage running the operating system. Giving the possibility to share and distribute completely configured operating systems by copying the memory cards using image files.

Next to standard interfaces for USB, HDMI or audio there are also the general purpose input/output (GPIO) pins. These pins provide connectivity to interfaces like SPI, I²C or UART and are the physical gateway to the surrounding world. They can be used for all sorts of purposes, to send and receive diverse signals and they also serve as the connection to

Raspberry Pi shields. There are many extensions available, capable of broadening the functionality and potential of the Raspberry Pi. One example would be a NFC shield providing NFC reader functionality to the Raspberry Pi (Figure 2.5).

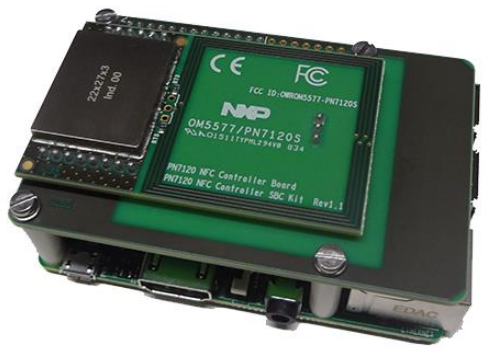


Figure 2.5: Mounted NFC shield on top of a Raspberry Pi [F3]

2.3. CIPURSE™

CIPURSE™ is an open security standard that is using smart card technologies to provide secure messaging and encryption for NFC solutions.

Figure 2.6 illustrates a host with a NFC reader and a card holder in possession of a NFC card that stores sensitive data. Here, the CIPURSE™ standard provides the necessary security features that allow the host to securely access the encrypted data on the NFC card if he is authorized to do so.

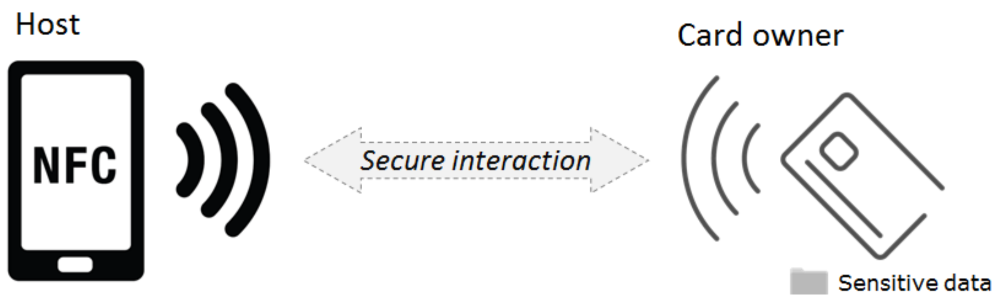


Figure 2.6: NFC card reader interaction [F4]

The data exchange protocol uses mutual authentication (3-pass as per “ISO/IEC 9798-2”), dynamic session keys, sequence integrity protection and secure messaging (based on “ISO/IEC 7816-4”) with AES-MAC or AES-encryption. This makes CIPURSE™ resistant to common attacks like differential power analysis (DPA) and differential fault analysis (DFA).

CIPURSE™ is thereby providing advanced security features to hold sensitive data, making it superior to other established NFC standards like MIFARE, which was hacked in 2008 using reverse engineering [23].

Systems using CIPURSE™ based products (CBP) for NFC applications need terminal devices (NFC readers) that are capable of performing cryptographic operations. These operations allow the terminal to authenticate and perform secure messaging with a CBP to be able to access files and data on the device. It is also critical that the cryptographic secret keys are stored in a secure place to inhibit modifications and observations without authorization.

Therefore, a device that provides the security functionality and the key vault is needed in such a terminal, the so called SAM. The SAM is a smart card that enhances the security of a reader device by providing the necessary functionality to perform secure transactions between a terminal and the data stored on a CIPURSE™ card.

Additionally to the security features the CIPURSE™ file structure is also supporting multiple applications on a CIPURSE™ card and is compatible with legacy systems. Moreover, flexible access rights and secure messaging rules are possible for each file stored on the memory of the NFC card.

In general, the cards holding the user data are also called “Proximity Integrated Circuit Card” (PICC). In operational use the PICC is interacting with a “Proximity Coupling Device” (PCD), which is basically a compliant terminal, holding a SAM.

To keep the extent of this document in scale, most descriptions are high-level and are not covered in detail. For further detailed information to CIPURSE™ and its features you can refer to the referred documents and industry standards [28], [29], [30] and [31].

2.3.1. File System

The file system on a CIPURSE™ device, like seen in Figure 2.4, is defined according to “ISO/IEC 7816-4” and “ISO/IEC 7816-9”. It is a simple file structure providing arbitrary numbers of applications and files per application.

On top of the file system stands the “Master File” (MF), it can hold multiple subfolders, so called “Application Dedicated Files” (ADFs). The ADFs represent the applications that are present on the particular CIPURSE™ device.

Underneath a MF or the ADFs so called “Elementary Files” (EFs) can be created, these are files that can hold specific user data. The specifications do not limit the number of EFs hold by an application, although the maximum number has to be defined while creating the data structure of the application (the ADF). The EFs can be of different types (binary, linear record, cyclic record and linear value-record files), but since the application described in this thesis uses binary files only, this documents remarks will be limited to this file type.

There are also some EFs present in the MF that are mandatory (EF.ID_INFO, EF.IO_CONFIG and EF.FILELIST). These three files are essential for a CIPURSE™ file system and are pre-defined by the OSPT Alliance specifications [28]. The same applies for the EF.ID_INFO and EF.FILELIST file in an ADF. A CIPURSE™ SAM ADF has two additional mandatory files, the EF.SAMInfo file and EF.SAMPwd file.

Every ADF can be identified and selected by its application identifier (AID) or optional by its file identifier (FID), a two-byte number that’s also used to reference EFs.

Additionally to the FID EFs can (optionally) have a short EF identifier (SFID), a five bit number which allows a direct (and therefore faster) access to an EF without selecting its ADF before. While FIDs (which are mandatory for EFs) only have to be unique within an ADF, a SFID has to be unique on the CIPURSE™ card. This uniqueness makes it possible to a directly access an EF using its SFID.

It is important to consider that there are reserved file identifies defined in the “ISO/IEC 7816-4”. These IDs must not be used for custom ADFs and EFs.

<u>Reserved on MF level:</u>

- 2F00H - identifies EF.DIR
- 2F01H - identifies EF.ATR

<u>Reserved on ADF level:</u>

- 0000H - identifies the current EF
- 2FF7H - identifies EF.ID_INFO
- 3F00H - identifies the MF
- 3FFFH - identifies the current DF
- 1000H - identifies the EF.SAMInfo file (only on SAM)
- 1001H - identifies the EF.SAMPwd file (only on SAM)
- FFFFH - is reserved for future use
- 0001H , 0009H , 000AH, 2FF0H to 2FF6H, 2FF8H to 2FFFH - may be used for administrative purpose
- The FID of the parent ADF

As an example, Figure 2.7 is showing the file structure of a CIPURSE™ card with one ADF containing one custom EF.

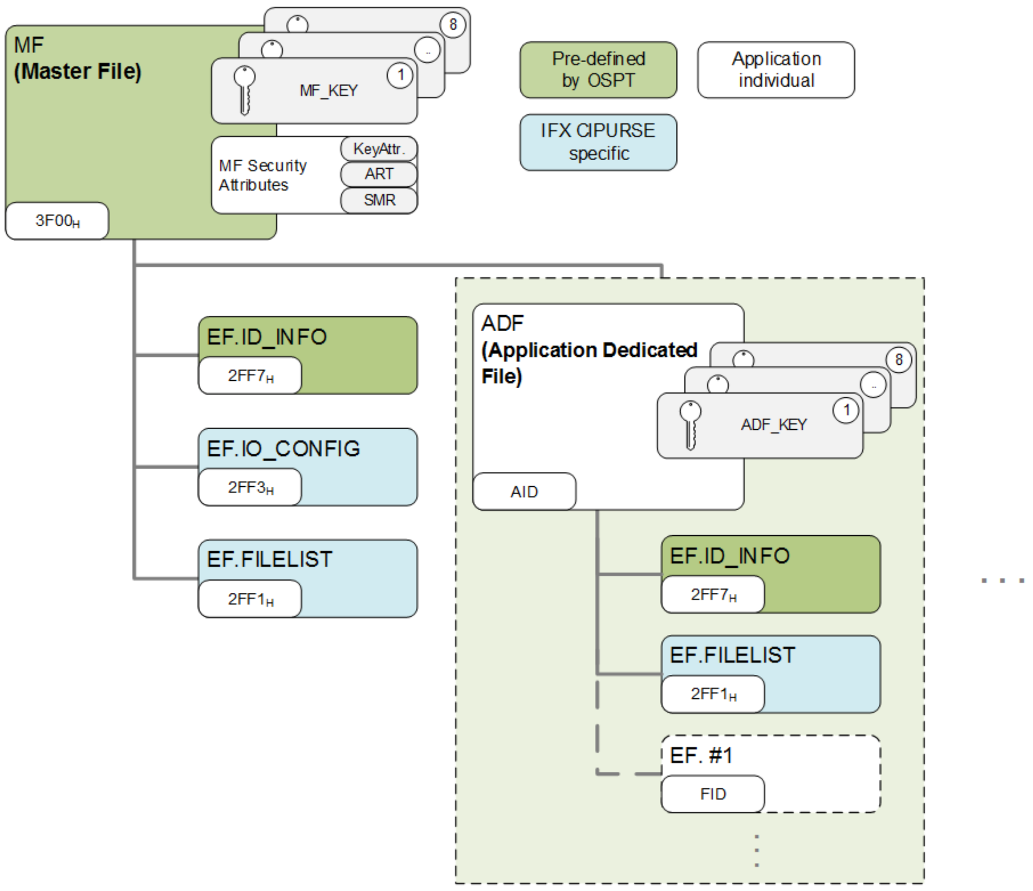


Figure 2.7: CIPURSE™ card file system

The MF and the ADF have the same security architecture to inhibit unauthorized access or modifications of the card or application. They are secured using 128-bit keys for AES encryption. Up to eight keys can be used for every application and MF. These keys allow configuring flexible access rights and securing messaging rules for each file.

2.3.2. Security Architecture

The security architecture of CIPURSE™ includes keys, an authentication mechanism, security attributes for access control management and secure messaging.

During every interaction with a CIPURSE™ device it is made sure that all security requirements are fulfilled. These include the possession of the necessary key for the affected data object, which is proofed by a mutual authentication and the applied security level of the communication (the secure messaging) that's specified in the security attributes of the data object.

2.3.2.1. Secure Messaging

Secure messaging makes sure to have a secure channel between a PCD and the data objects on a PICC. CIPURSE™ is therefore specifying secure messaging based on "ISO/IEC 7816-4" and [31].

The commands and the appended data are usually transferred in so called APDUs (Application Protocol Data Units), in case of secure messaging the commands are handled in secure messaging APDUs (SM-APDUs).

CIPURSE™ provides three different security levels for transferring data:

- SM_PLAIN
  - Plain data transfer, no encryption and no integrity protection field
- MAC'ed
  - Plain data transfer, no encryption but integrity protected communication with message authentication code (MAC) field
- ENC'ed
  - Encrypted data transfer and integrity protected communication

All the cryptographic methods are specified in [31].

2.3.2.2. Security Attributes

The access to a CIPURSE™ device and to the data stored on a PICC is controlled by the settings within the access right table (ART). As the name suggests, the table defines the access rights after establishing a secure session with the device. Thus it assigns particular access rights to a dedicated set of keys for the interaction with the MF, the ADFs and the single EFs.

The secure messaging rules to access any of the data objects on a PICC need to be defined within the secure messaging rules (SMRs). This means that the SMR can be used to define the minimum communication security level that's necessary to exchange specific data between a PCD and a PICC. For sensitive information, encryption might be used while other information could be communicated without encryption to reach higher data exchange rates. Keep in mind that the file security attributes (ATR & SMR) need to be defined for every single EF, ADF, as well as for the MF.

Application keys (ADF keys) are used to secure the application while MF keys are used to secure the card or SAM against unauthorized changes. For example, the MF key may be used to modify a cards content (e.g. adding an application, changing security attributes on MF level), but to read or change any application data an ADF key is needed.

2.3.2.3. Authentication and Security State

If the access to a data object on a PICC is restricted, a PCD needs to move into a security state by successfully carrying out a mutual authentication with a valid key. With this process the PCD acquires the right to exercise any interactions that are permitted in the access rights assignment for this key.

The security state is linked to one key and the current directory on the PICC. After selecting another ADF for example, the mutual authentication needs to be renewed with a valid key. The mutual authentication itself is done by a three-way challenge-and-response protocol and is specified in [31].

A high-level view of the authentication process between the PICC (P) and the PCD (T) is shown in Figure 2.8. Next to the three parts of the authentication process (preparation of the raw data, derivation of the session key, mutual authentication), the figure is also showing one of the crucial security features, a dynamic session key (k_0). This method is one of the key elements that make CIPURSE™ superior to standards like MIFARE and is also securing transactions against DPA attacks.

When performing DPA, attackers are monitoring power signals to statistically analyse data from multiple cryptographic operations. These allow them to extract information that is potentially correlated to secret keys [24]. A dynamically and continuously changing session key can prohibit such an attack.

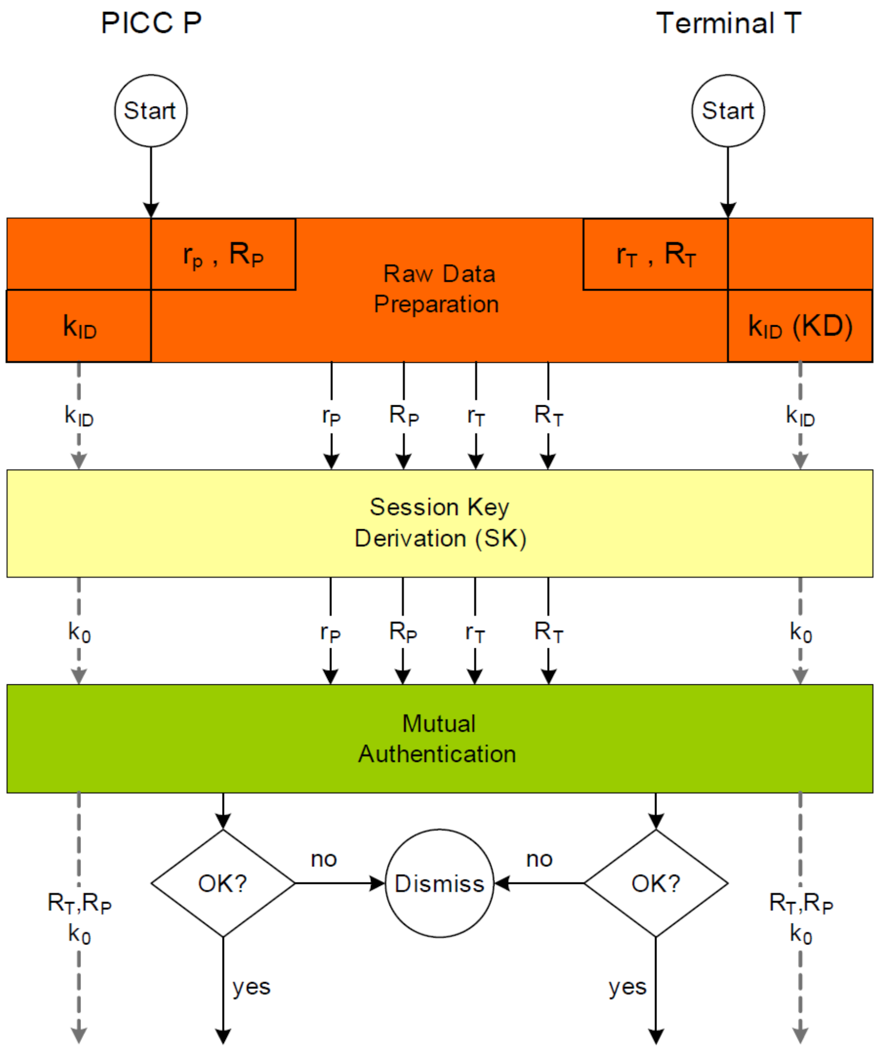


Figure 2.8: Scheme for authentication parts [31]

Figure 2.9 shows a more detailed diagram of the authentication process. While both parties are sharing a common secret (k_{ID}) there is also another temporary secret, the session key (k_0). It is dynamically derived from the k_{ID} and changes with every session. The k_0 is used as an AES key to encrypt random values which are exchanged between the PICC and PCD. These random challenges R_P (send and generated by the PICC) and R_T (send and generated by the PCD) are followed by the responses c_P and c_T, respectively.

To be able to successfully derivate a session key, the function SK depends on additional random values before the authentication. Therefore, the values r_P and r_T are exchanged between the PICC and PCD, allowing a derivation that is secure against DPA attacks.

Note: k_{ID} and k_0 are private and R_P, r_P , R_T, r_T are public.

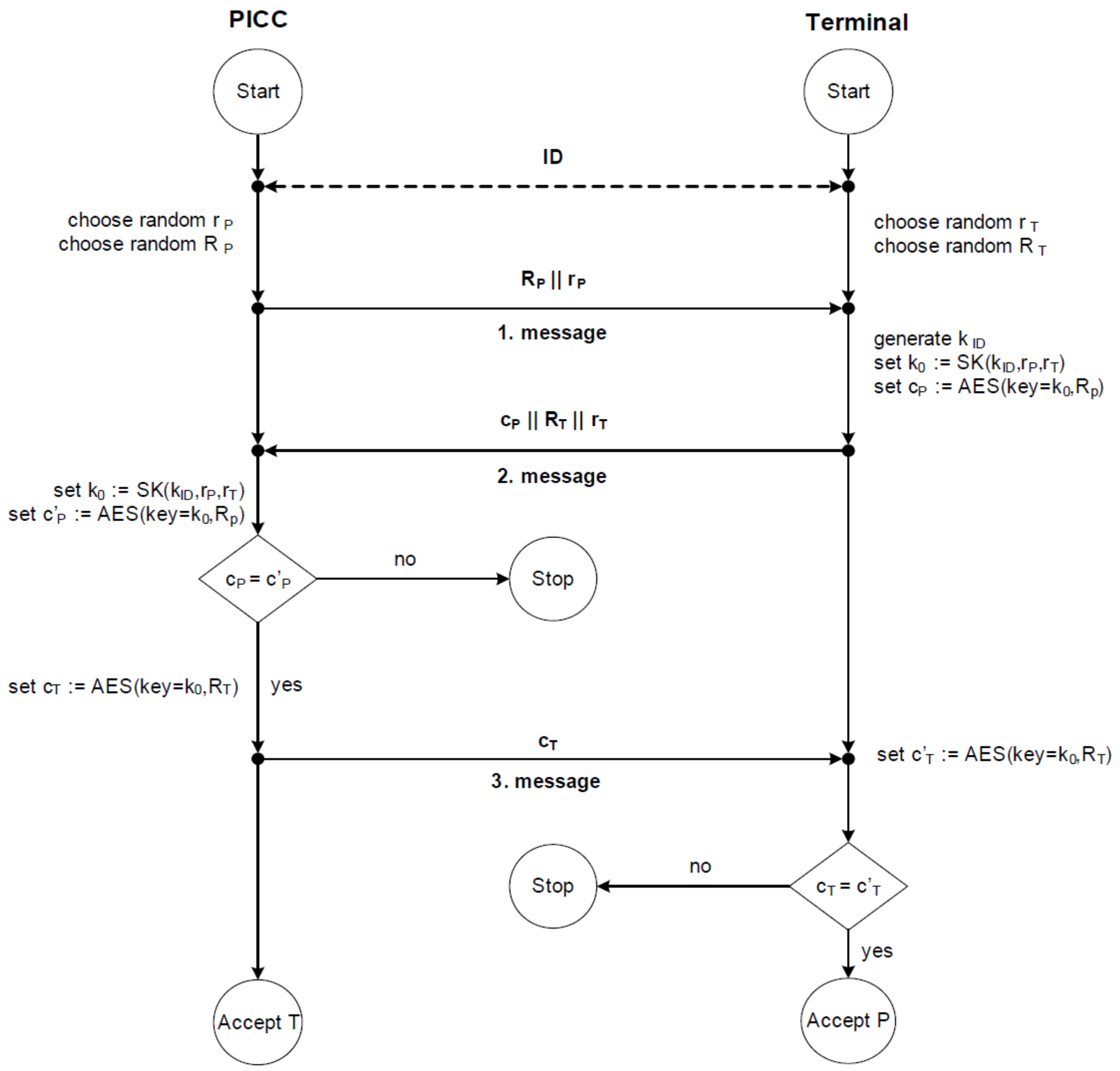


Figure 2.9: Scheme for authentication protocol [31]

2.3.2.4. Keys

CIPURSE™ uses 128-bit AES keys. The maximum number of keys for a SAM MF or ADF is eight but in case of a CIPURSE™ card it depends on the used CIPURSE™ profile. A security controller, like it is used in this thesis, is able to hold up to eight keys for the MF and every application on the card.

The maximum number of keys for an application (=n) is defined when creating the ADF and cannot be changed afterwards. The keys are referenced with numbers from 1 to n and are used to define the ART for all types of data objects (MF, ADFs or EFs).

If no security functionality is needed it is also possible to create a plain application without a key, in that case there is also no ART and SMR for this specific ADF.

A KVV (Key Verification Value) has to be calculated by the PCD and verified by the PICC when creating an ADF. There are also key security attributes for every key defining the operations permissible with or on that key, for example the validity of the key. Further more detailed information can be found in the [28].

2.3.2.5. Access Right Table (ART)

By allowing the execution of particular commands during operation, the ART is assigning the rights to every single key that was loaded onto a PCD. Additionally it is possible to grant access without the usage of any key (the execution of commands without being in a secure state).

Therefore, the ART consists of a sequence of 1 + n bytes. The first byte describes the unconditional access rights, independently of the secure state and key ("Always"). While every single byte of the additional n bytes, describes the rights of one of the n keys used in the directory (MF or ADF).

All possible commands are clustered in access groups (ACGs), which differentiate between MF, ADF and EFs. That's because of the varying command sets for the different file types or directories.

Every bit in one of the ARTs bytes is assigned to one ACG. If one of these bits is set to 1, all the commands in the corresponding ACG are executable in a security state with the key linked to this ART entry.

Table 2.1 to 2.3 show in what way the ART of an MF, ADF and EF binary file is build up, by revealing how the keys are mapped to the present access groups in these three cases.

MF		**Access Right Table (ART)**						
Access Control Group (ACG) MF	**Command**	**Always**	**Key 1**	**Key 2**	**...**	**...**	**Key (x-1)**	**Key x**
ACG_MF_1	UPDATE_KEY UPDATE_KEY_ATTRIBUTES	1	1	1	...	...	1	1
ACG_MF_2	PERFORM_TRANSACTION1) 2) CANCEL_TRANSACTION1) 2)	1	1	1	...	...	1	1
ACG_MF_3	-	1	1	1	...	...	1	1
ACG_MF_4	FORMAT_ALL	1	1	1	...	...	1	1
ACG_MF_5	CREATE_FILE (EF)1) DELETE_FILE (EF or ADF)1)	1	1	1	...	....	1	1
ACG_MF_6	CREATE_FILE (ADF)	1	1	1	...	...	1	1
ACG_MF_7	READ_FILE_ATTRIBUTES	1	1	1	...	...	1	1
ACG_MF_8	UPDATE_FILE_ATTRIBUTES	1	1	1	...	...	1	1
	Value	FF_H	FF_H	FF_H	$..._H$	$..._H$	FF_H	FF_H

Table 2.1: Shows how the ACGs are connected to the different keys in case of an MF (x is the maximum number of keys)

ADF		Access Right Table (ART)						
Access Control Group ADF	**Command**	**Always**	**Key 1**	**Key 2**	**⋮**	**⋮**	**Key (x-1)**	**Key x**
ACG_ADF_1	UPDATE_KEY UPDATE_KEY_ATTRIBUTES	1	1	1	...	...	1	1
ACG_ADF_2	PERFORM_TRANSACTION1)2) CANCEL_TRANSACTION1)2)	1	1	1	...	...	1	1
ACG_ADF_3	DEACTIVATE_FILE(ADF)1)	1	1	1	...	...	1	1
ACG_ADF_4	ACTIVATE_FILE(ADF)1)	1	1	1	...	...	1	1

ACG_ADF_5	CREATE_FILE(EF)
	DELETE_FILE(EF)1)
		1	1	1	...	...	1	1

ACG_ADF_6	DELETE_FILE(ADF)1)	1	1	1	...	...	1	1
ACG_ADF_7	READ_FILE_ATTRIBUTES	1	1	1	...	...	1	1
ACG_ADF_8	UPDATE_FILE_ATTRIBUTES	1	1	1	...	...	1	1
	Value	**FF$_H$**	**FF$_H$**	**FF$_H$**	**...$_H$**	**...$_H$**	**FF$_H$**	**FF$_H$**

Table 2.2: Shows how the ACGs are connected to the different keys in case of an ADF (x is the maximum number of keys)

EF BINARY		Access Right Table (ART)						
Access Control Group (ACG)	**Command(s)**	**Always**	**Key 1**	**Key 2**	**⋮**	**⋮**	**Key (x-1)**	**Key x**
ACG_EFBIN_1	READ_BINARY	1	1	1	...	...	1	1
ACG_EFBIN_2	UPDATE_BINARY	1	1	1	...	...	1	1
ACG_EFBIN_3	-	1	1	1	...	...	1	1
ACG_EFBIN_4	-	1	1	1	...	...	1	1
ACG_EFBIN_5	-	1	1	1	...	...	1	1
ACG_EFBIN_6	-	1	1	1	...	...	1	1
ACG_EFBIN_7	READ_FILE_ATTRIBUTES	1	1	1	...	...	1	1
ACG_EFBIN_8	UPDATE_FILE_ATTRIBUTES	1	1	1	...	...	1	1
	Value	**FF$_H$**	**FF$_H$**	**FF$_H$**	**...$_H$**	**...$_H$**	**FF$_H$**	**FF$_H$**

Table 2.3: Shows how the ACGs are connected to the different keys in case of an EF binary file (x is the maximum number of keys)

Next, there are some examples that show how to read the tables describing the ART and how they work and look like.

Example for an ART on MF level

ART: “**40 48**”

The shown MF ART is two bytes long, this means there is only one key on MF level. The first byte describes the access rights without a security state (“Always”) and the second one the access rights using the first key (“Key 1”).

For better understanding it helps to write the byte as a sequence of bits. The bit string can then be transferred to Table 2.1 to map them to the access control groups.

Always: 40$_H$ = 0**1**00 0000$_B$

In case of the byte value 40$_H$ the seventh bit is set, this means that the 7th MF access control group (ACG_MF_7) is executable without any security state.

Key 1: 48$_H$ = 0**1**00 **1**000$_B$

In case of the first MF key (Key 1, byte value 48$_H$) there are two bits set, the seventh and the forth one. These settings allow the execution of the 4th and 7th access control group (ACG_MF_4 & ACG_MF_7).

Example for an ART on ADF level

ART: “**40 C3 41**”

The shown ADF ART has three bytes, this means there are two keys on ADF level. The first byte describes the access rights without a security state (“Always”), the second the access rights using the first key (“Key 1”) and the third for the second key (“Key 2”).

Let’s write the bit sequence down to map it to Table 2.2.

Always: 40$_H$ = 0**1**00 0000$_B$

ACG_MF_7 is executable without any security state.

Key 1: C3$_H$ = **11**00 00**11**$_B$

ACG_MF_1, ACG_MF_2, ACG_MF_7 and ACG_MF_8 are executable with the first ADF key.

Key 2: 41$_H$ = 0**1**00 000**1**$_B$

ACG_MF_1 and ACG_MF_8 are executable with the second ADF key.

2.3.2.6. Secure Messaging Rules (SMR)

While the ART describes which data and commands can be exchanged or executed, the SMR (Secure Messaging Rules) are data object-specific rules describing how the information is transferred between PICC and PCD.

Like in the ACGs, all possible commands for the different file types are clustered into four groups, the secure messaging groups (SMGs), they are needed to create a clearer scheme for the SMR.

The SMR itself consists out of two bytes which are split up into four nibbles, one for each SMG. The nibble itself is split up again, into two bits describing the outgoing commands and two bits describing the incoming responses (with respect to the PICC).

These two bits are coding the minimum secure messaging security level that have to be used while exchanging data using any commands in the corresponding SMG.

SMG		**SMG_MF_1**		**SMG_MF_2**		**SMG_MF_3**		**SMG_MF_4**	
Command(s)		FORMAT_ALL DELETE_FILE (EF or ADF)1)		UPDATE_KEY UPDATE_KEY_ATTRIBUTES		READ_FILE_ATTRIBUTES		UPDATE_FILE_ATTRIBUTES CREATE_FILE (EF)1) CREATE_FILE (ADF)	
Messaging		**CMD SM-APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**
	SM_PLAIN	00_B	00_B	00_B	00_B	00_B	00_B	00_B	00_B
	MAC'ed	01_B	01_B	01_B	01_B	01_B	01_B	01_B	01_B
	ENC'ed	10_B	10_B	10_B	10_B	10_B	10_B	10_B	10_B
SMR entry		0000_B		0000_B		0000_B		0000_B	
SMR		**00_H**				**00_H**			

Table 2.4: Shows how the SMGs are mapped to the different security levels in a SMR for a MF

SMG		**SMG_ADF_1**		**SMG_ADF_2**		**SMG_ADF_3**		**SMG_ADF_4**	
Command(s)		ACTIVATE (ADF) DEACTIVATE (ADF)1) DELETE_FILE (EF)1) DELETE_FILE (ADF)1)		UPDATE_KEY UPDATE_KEY_ATTRIBUTES		READ_FILE_ATTRIBUTES		UPDATE_FILE_ATTRIBUTES CREATE_FILE(EF)	
Messaging		**CMD SM-APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**
	SM_PLAIN	00_B	00_B	00_B	00_B	00_B	00_B	00_B	00_B
	MAC'ed	01_B	01_B	01_B	01_B	01_B	01_B	01_B	01_B
	ENC'ed	10_B	10_B	10_B	10_B	10_B	10_B	10_B	10_B
SMR entry		0000_B		0000_B		0000_B		0000_B	
SMR		**00_H**				**00_H**			

Table 2.5: Shows how the SMGs are mapped to the different security levels in a SMR for an ADF

SMG		**SMG_EFBIN_1**		**SMG_EFBIN_2**		**SMG_EFBIN_3**		**SMG_EFBIN_4**	
Command(s)		READ_BINARY		UPDATE_BINARY		READ_FILE_ATTRIBUTES		UPDATE_FILE_ATTRIBUTES	
Messaging		**CMD SM-APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**	**CMD SM_APDU**	**RSP SM_APDU**
	SM_PLAIN	00_B	00_B	00_B	00_B	00_B	00_B	00_B	00_B
	MAC'ed	01_B	01_B	01_B	01_B	01_B	01_B	01_B	01_B
	ENC'ed	10_B	10_B	10_B	10_B	10_B	10_B	10_B	10_B
SMR entry		0000_B		0000_B		0000_B		0000_B	
SMR		**00_H**				**00_H**			

Table 2.6: Shows how the SMGs are mapped to the different security levels in a SMR for an EF binary file

Next, there is an example that shows how to read the tables describing the SMR and how they work and look like.

Example for an SMR on ADF level

SMR: “**09 28**”

A SMR is always two bytes long and separated into four nibbles. For better understanding it helps to write them down as a sequence of bits. The bit string can then be transferred to Table 2.5 to map them to the SMG.

$09_H = 0000\ 1001_B$

The first and the second bit pair of the first nibble (0000_B) is set to 00_B. Meaning that the first secure messaging group (SMG_ADF_1) can be transmitted in plain (SM_PLAIN) as a command and as a response APDU (for example the command to delete an EF).

The first bit pair of the second nibble is 10_B and the second one 01_B. This means that an APDU of the second secure messaging group (SMG_ADF_2) has to be encrypted (ENC’ed) as a command and MAC’ed as a response.

$28_H = 0010\ 1000_B$

The third secure messaging group (SMG_ADF_3) can be transmitted in plain as a command and encrypted as a response APDU. The APDU for the fourth SMG (SMG_ADF_4) has to be encrypted as a command and can be plain as a response.

2.3.2.7. SAM Password

For a CIPURSE™ SAM there is an additional security feature to prohibit misuse, the SAM password. It needs to be verified to move the SAM to authorized state. Otherwise the SAM application cannot be used. Therefore, there is a 16 byte password stored in every SAM applications EF.SAMPwd file.

2.3.3. CIPURSE™ Card

The CIPURSE™ card is a PICC that holds a CIPURSE™ application with its file structure and stored data. Its functionality can be manifold and depends on the present applications. During the operational use the card is communicating with a CIPURSE™ compliant terminal to execute its intended purpose.

There are multiple types of CIPURSE™ cards, so called CIPURSE™ profiles. These profiles have different properties and it is important to mention that this document refers to a CIPURSE™ security controller (profile T) only. The T profile provides all CIPURSE™ features and offers the biggest memory and is therefore the most convenient choice for this application.

The OSPT standard requires the security controller to support at least 4 kilobytes of user memory, eight configurable applications and 32 configurable files per application. These are the minimum requirements. For example, the security controller used in this thesis was provided by Infineon and supports up to 11 kilobytes of storage. [29]

Figure 2.10 gives a basic overview of the variable functionalities of the different CIPURSE™ profiles.

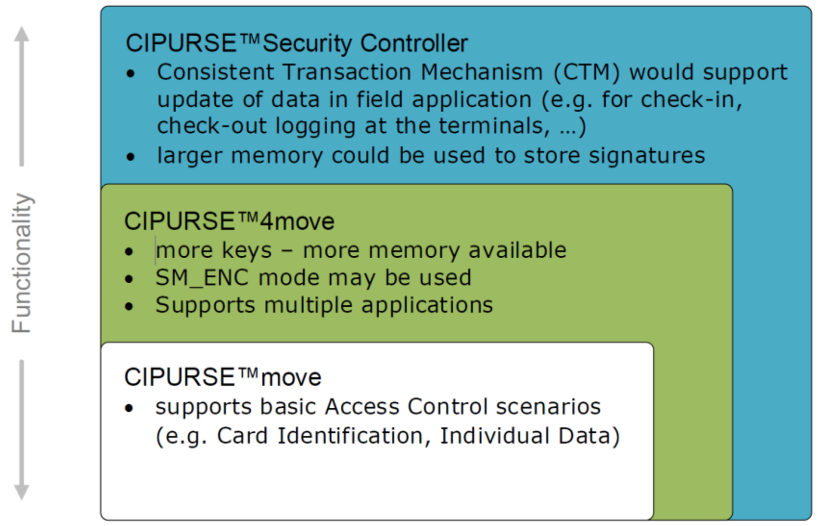

CIPURSE™Security Controller
• Consistent Transaction Mechanism (CTM) would support update of data in field application (e.g. for check-in, check-out logging at the terminals, ...)
• larger memory could be used to store signatures
CIPURSE™4move
• more keys – more memory available
• SM_ENC mode may be used
• Supports multiple applications
CIPURSE™move
• supports basic Access Control scenarios (e.g. Card Identification, Individual Data)
Functionality

Figure 2.10: CIPURSE™ profile functionalities

2.3.4. CIPURSE™ SAM

In order to support CIPURSE™ based products (CBP) in an application, the terminal device is required to perform cryptographic operations. These allow it to authenticate and provide secure messaging to access the CIPURSE™ files and data.

To be able to sustain the security of the system, the cryptographic secret keys must be in a secure device where the keys cannot be observed or modified without authorization. The traditional solution to this problem is a secure access module (SAM). Therefore, such module has to be present in the terminal device to provide all necessary security functions.

Next to the operational functions in a terminal, a CIPURSE™ SAM can also be used for the personalization of other CBPs and provides additional security functions to support overall application security.

2.3.4.1. CIPURSE™ SAM Types

The CIPURSE™ SAM functionality can be restricted by limiting the commands that are supported. These restrictions can be done by adjusting a single byte in the elementary file EF.SAMInfo. Depending on this byte the SAM functionality is defined and it is possible to differentiate between multiple SAM types:

SAM type	Notation	Description
Load SAM	SAM_K	Provides the function to generate and load keys onto CIPURSE™ SAMs. The SAM_K is the root of the whole key and SAM hierarchy and supports the personalization of other SAMs

		used in the application. Therefore, it should be operated and maintained in a secure environment only.
Backup Office SAM	SAM_A	Provides functions to verify and decrypt transaction messages, like validating the signed records created by a terminal.
Personalization SAM	SAM_P	Supports personalization functions only, like the personalization of CBPs. These SAMs are operated in personalization devices only.
Standard SAM	SAM_T	Provides standard functionality for terminals by supporting CBP applications to interact and perform operations with cards.
General SAM	SAM_G	There are no functional restrictions. General SAMs are able to perform all operations and are therefore capable to provide a simple CIPURSE™ solution.

Table 2.7: List of CIPURSE™ SAM types

2.3.4.2. Keysets

The file system of a CIPURSE™ SAM needs to contain certain keysets within their SAM application to function properly. A keyset file includes a key file (contains a key) and its corresponding key attribute file (contains the attributes for that key). Additionally there can be a key counter file to limit the number of operations that can be performed by this key.

There are different types of keyset files (Figure 2.11) and it depends on the SAM type which keysets need to be present on the SAM.

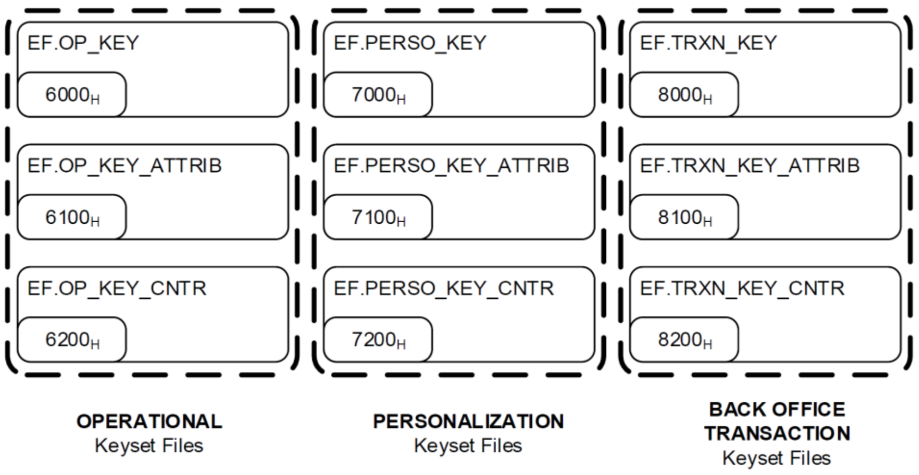


Figure 2.11: CIPURSE™ SAM keysets and additional keyset files

The operational keyset (FID: $60xx_H$) stores keys that are used to establish secure sessions between two SAMs or a SAM and a card. Therefore, at least one key on MF or ADF level has to match between the SAM and the card. Such a keyset is needed in a SAM_T that is operating in a terminal.

The personalization keyset (FID: 70xx$_H$) contains the keys used to personalize other CBPs. During the personalization process these keys are loaded into other key files on a SAM or onto CIPURSE™ cards. This keyset is essential for a SAM_P that's used to personalize a user's card.

The back-office-transaction keyset (FID: 80xx$_H$) can be used to verify the integrity of transactions performed at terminals and is essential for a SAM_A. This keyset file has no use in this thesis.

Table 2.8 shows which keysets have to be present in the different types of SAMs.

	Operational	**Personalization**	**Back office transaction**
SAM_K	x	x	-
SAM_A	-	-	x
SAM_P	x	x	-
SAM_T	x	-	x
SAM_G	x	x	x

Table 2.8: Necessary keysets for the different CIPURSE™ SAM types

2.3.4.3. Key Types

There are different types of keys with unique characteristics. To help to differentiate between these key types acronyms are used for the key names. These types are not standardized, therefore, it is important to mention that the following types in Table 2.9 refer to this document only.

Key type	**Acronym**	**Description**
Transport key	-	The default MF key(s) of a SAM or card when shipped to a customer.
Grand master key	KGM	These keys are always generated and stored in personalization keysets in SAM_Ks and used to derive master keys (KMs) and static keys (KSs). The exporting of grand master keys (KGMs) requires key diversification.
Master key	KM	KMs can reside in any type of key file and derive from a SAM_Ks KGMs using keyset-specific diversification data. KMs are used to create derived keys (KDs). The exporting of KMs requires key diversification.
Derived key	KD	KDs derive from KMs using card-, SAM- or other diversification data and they are usually stored in the MF or ADF containers of a card or SAM.

Table 2.9: Explanation and acronyms for different key types

2.3.4.4. Key Diversification

Key diversification is one of the most important features in order to secure a CIPURSE™ application. The key diversification technique is used to derive keys from KGMs or KMs. They are diversified to prevent an attacker, who has somehow obtained a key, to learn about the original keys.

The key diversification makes sure that a KM or a KGM is not compromised even if any of the derived keys are revealed due to a successful attack. Therefore, a key (KGM or KM) and diversification data are processed together by the diversification algorithm to get a derived key as output. Therefore, it is common to use keyset-specific (constant) or card/SAM-specific (unique) diversification data for the key diversification.

The static keyset-specific diversification data is used for the diversification of the keys in a SAM_Ts or SAM_Ps operational keysets. This ensures that the KMs within a SAM_P and a SAM_T are the same. The same keys are required because the card is getting personalized by a SAM_P and has to operate with a SAM_T.

The diversification of keys for MFs or ADFs is using card- or SAM-specific diversification data. This chip identification data is unique for each card and SAM. When the keys on MF and ADF level are personalized using this unique diversification data (from the SAM or card), the same diversification data is used to derive the key from the KMs in the operational keyset of the application SAM_P. The derived key can then be used to establish a secure session on MF or ADF level of the card or SAM.

Figure 2.12 shows how the key diversification technique is used. The KGM of the SAM_K is diversified using constant diversification data (keyset-specific) to get a KM for a SAM_P or SAM_T. Next the KM (for example of a SAM_P) is derived again using unique diversification data (Card or SAM specific data). This creates exclusive keys to assure that the original key will not be compromised if these KDs are ever revealed.

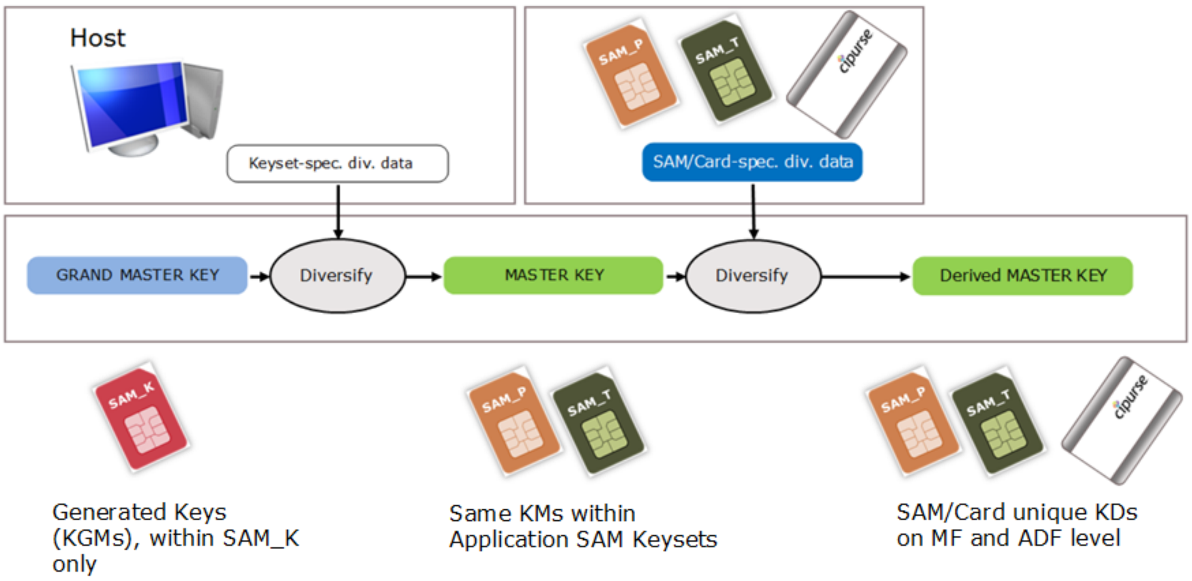

Host
Keyset-spec. div. data
SAM/Card-spec. div. data
GRAND MASTER KEY
Diversify
MASTER KEY
Diversify
Derived MASTER KEY
Generated Keys (KGMs), within SAM_K only
Same KMs within Application SAM Keysets
SAM/Card unique KDs on MF and ADF level

Figure 2.12: Sequence of key diversification steps

2.3.4.5. CIPURSE™ SAM Application

Like a CIPURSE™ card a SAM can hold multiple applications. In Figure 2.13 you can see the content of a SAM holding one general SAM_G application.

The CIPURSE™ SAM ADF is secured with up to 8 application keys (ADF KEY 1 - 8). Within the ADF the application type is configured in the EF.SAMInfo file. The EF.SAMPwd file holds the SAM password, which is needed to assure only authorized access and usage of this application. Only with the SAM password it is possible to move the SAM application to authorized state.

The keyset files can be found in the applications ADF and contain all the application keys, including key attribute and key counter files.

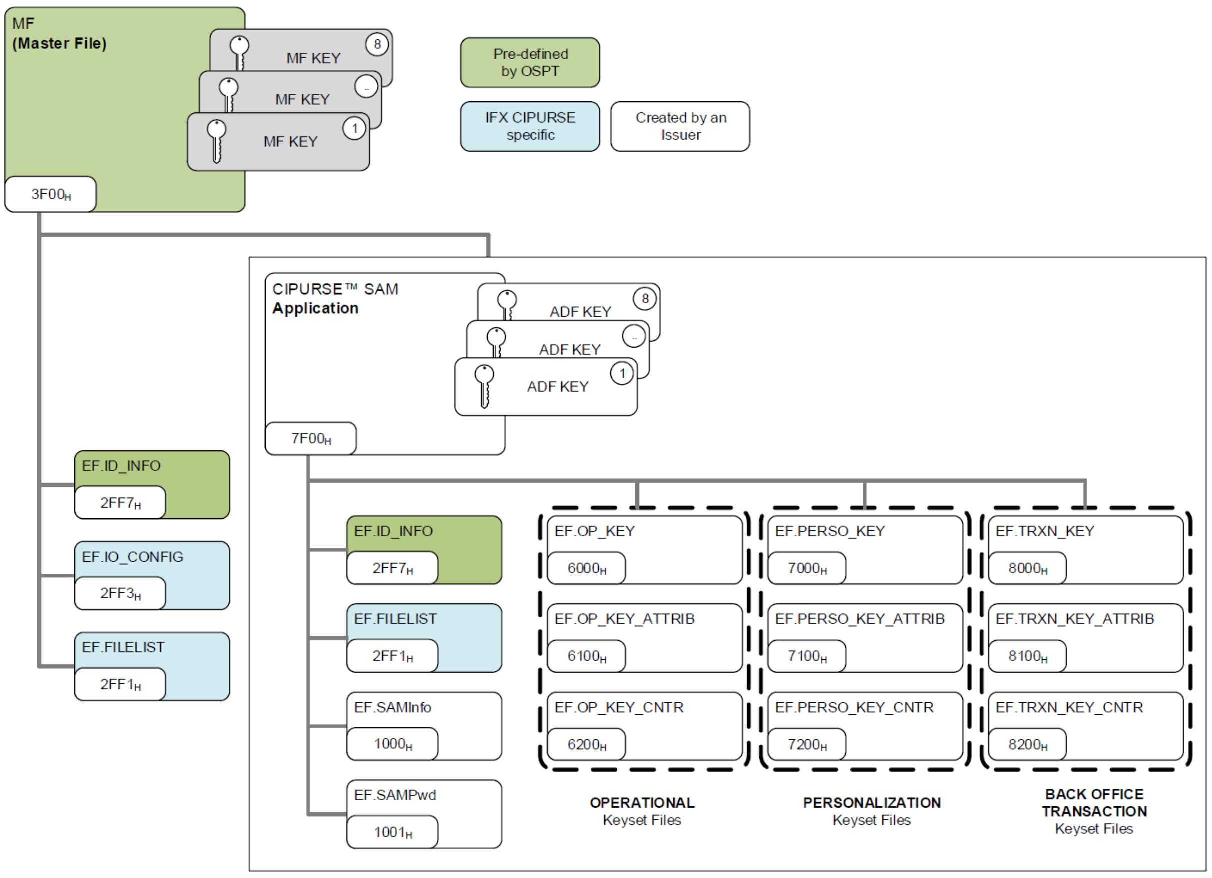


Figure 2.13: CIPURSE™ SAM filesystem of a general personalized SAM_G

2.3.5. CIPURSE™ Terminal Library

The CIPURSE™ Terminal Library is a library offered by Infineon to provide CIPURSE™ specific functionalities to terminals, thus it reduces the effort when developing a CIPURSE™ terminal application by providing a user friendly software interface.

It implements all necessary commands for the personalization of CIPURSE™ file systems on cards and SAMs and is handling the communication between CIPURSE™ based products during the operational use.

The terminal library provides source code in C, C++ and Java and can thereby be integrated on most operating systems such as Windows, Linux or Android. This way it can be implemented on any “ISO/IEC14443” and “ISO/IEC 7816” compliant hardware.

Figure 2.14 shows where the terminal library is integrated in a CIPURSE™ terminal to act as an interface between SAM and card.

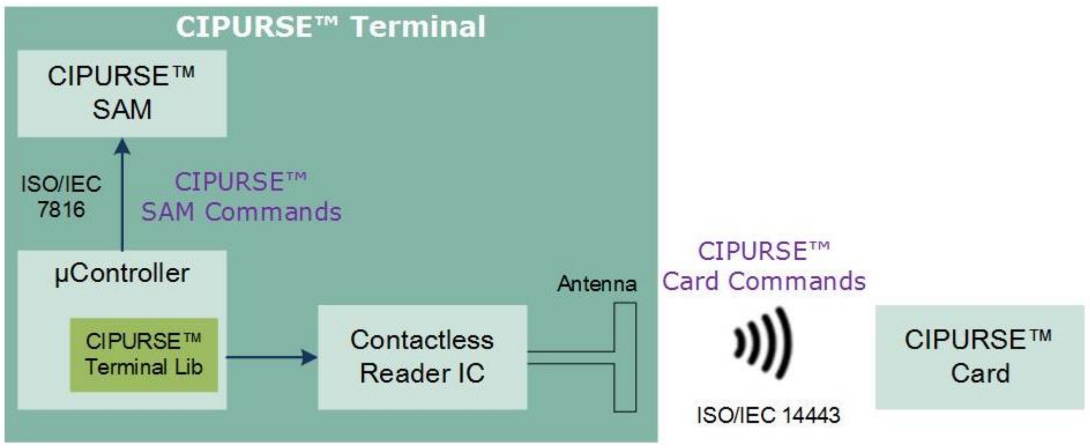


Figure 2.14: CIPURSE™ terminal diagram

2.3.5.1. Terminal Library Architecture

The terminal library is separated into four layers. It consists of the presentation layer, crypto services, the framework layer and the platform abstraction layer. Every one of these layers holds multiple modules with similar functions.

For example, the presentation layer includes interfaces to perform CIPURSE™ personalisation use cases in the command API and interfaces for SAM personalisation use cases in the SAM personalization API module.

Figure 2.15 shows a diagram of the whole architecture and its single components.

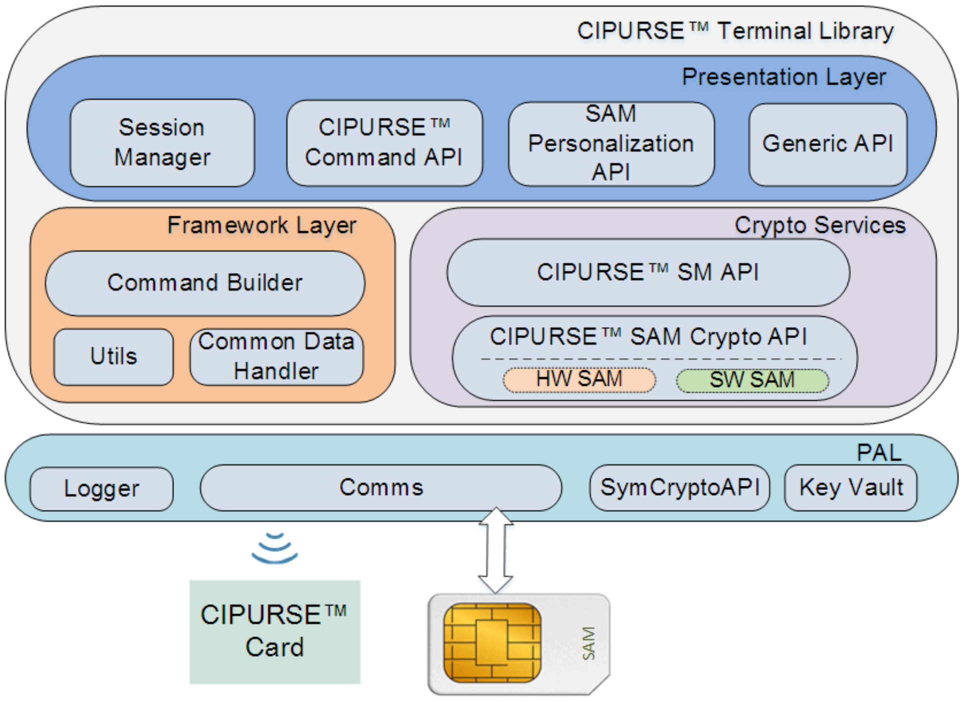


Figure 2.15: Terminal library architecture

Presentation layer

The modules included in this layer implement services required by terminal applications to develop application specific use cases.

Crypto services

This layer holds all the modules to implement the required CIPURSE™ cryptographic functionalities for a terminal application.

Framework layer

This layer holds some common data needed by the terminal library and is responsible to build the commands (the APDUs) needed to communicate with the CPBs. Therefore, it offers its services to the presentation layer and the crypto services.

Platform abstraction layer (PAL)

This layer connects the hardware and software by disclosing the hardware functionality via universal interfaces. It provides software implementations if necessary features are not support by the target hardware and can be adapted to the operating system and the hardware used for the terminal application.

More detailed information on the terminal library and its modules can be found in the release notes and enclosed additional documentation.

3. Results

This section is about the final results of the thesis. The results define an elaborated and implemented concept for a use case of a novel health data infrastructure based on the open CIPURSE™ standard. It comprises the final CIPURSE™ card and SAM application, an Android prototype app and the additional needed hardware for the personalization device.

3.1. CIPURSE™ Application

3.1.1. CIPURSE™ Card Application

The CIPURSE™ card application is essential for a clear picture on the card side as a basis to develop further infrastructure elements like the SAM hierarchy and the SAM application. The following aspects were considered during the design of the card application and are described in the following section:

Card data
- Which data needs to be stored on the card?

Card application roles
- Are there different roles that interact with the card?

Card application keys and security settings
- What are the security aspects for the data and the card?

Card states
- What is the cards life cycle?

3.1.1.1. Card Data

For this use case the card stores medical and lifestyle data of the cardholder. Therefore, during the personalization of the card, multiple EFs are created to hold such data. The file type, file ID and file size are predefined. The card application is holding the following data:

EF name	File ID	File size	File type	Content
MyID	$0x0001_H$	$0x60_H$	Binary file	Identification data (Insurance number, name). This data is encrypted additionally with a PIN code.
MyTelehealth	$0x0002_H$	$0x30_H$	Binary file	Telehealth login (username, password)
MyPharmaGen	$0x0003_H$	$0x300_H$	Binary file	Pharmacogenetic data (DNA markers).
MyFit	$0x0004_H$	$0x30_H$	Binary file	Fitness account login (username, password).
MyEmergency	$0x0005_H$	$0xC8_H$	Binary file	Emergency data (Blood group, allergies, diseases).

Table 3.1: Data that is stored in the CIPURSE™ application

Two vertical lines "||" are used as separators to separate the different entries in the EFs. For example, the "MyFit" EF contains a username and a password. If the password is "MyPassword" and the username "MyName" the EF entry would be "MyName||MyPassword||". Since the file size is predefined, the rest of the entry is filled with blank spaces, which act as place holders.

To be able to read the data of the card it is necessary to own the key with the proper access rights, with two exceptions. The first exception is about the emergency data. The "MyEmergency" EF can be read by everyone with a capable device. The second one is about the identification data. To ensure the anonymity of the card owner the data in the "MyID" EF is encrypted a second time using a PIN code. This means that even if someone has the access rights to read the identification data on the CIPURSE™ card, the card owner has to approve the access by entering a PIN code.

3.1.1.2. Card Application Roles

There are four different roles that have to be handled by the application. The administrator, the doctor, the pharmacist and the user (card owner). As illustrated in Figure 3.3, every role has different keys and access rights to the CIPURSE™ card and therefore, unique tasks. This requires everyone who has one of these defined roles to own the proper SAM.

The administrator works in a secure environment and personalizes all the user cards. This means that the admin loads the application on the CIPURSE™ card, with its file structure, security attributes and basic user data.

The health professionals (doctor and pharmacist) have adjusted access rights based on the data confidentiality. They are the ones who read, write and process the data from the card. For example, only a doctor can write emergency data on the card and has access to the telehealth account data. On the other hand, only a pharmacist is able to write pharmacogenetics data on the user's card.

The user is the card owner and is physically holding and managing the personalized card. He is able to change the access rights of his card to prevent others to access his data and is able to write his fitness account data on the card.

The card owner's access rights are limited to his own card while the rights of the other roles apply for all CIPURSE™ cards running this application.

3.1.1.3. Card Application Keys

MF keys

In this use case only one MF key is used (MF key 1: "KD_CARD_MF_ADMIN"). This key is used to secure the card against unauthorized changes. It will also allow the administrator to delete the whole card content if it is necessary to set it back to its initialized state, for example for re-use. As the other keys on MF level are not used, they are invalidated (Key Attributes on MF Level are set to “invalid” for MF key 2 to MF key 8).

ADF keys

The application itself is secured with four application keys (Table 3.2). These keys are used for administration purposes (personalization) as well as for the operational use.

ADF key No.	Function	Name	Description
1	ADF administrator key	KD_CARD_ADF_ADMIN	Needed by the administrator to personalize or delete the user card content.
2	ADF doctor key	KD_CARD_ADF_DOCTOR	Allows doctors to read/write approved data from/to the user card.
3	ADF pharmacist key	KD_CARD_ADF_PHARMACIST	Allows pharmacists to read/write approved data from/to the user card.
4	ADF user key	KD_CARD_ADF_USER	Unique key for every card owner to change the access rights and write certain data on the card.

Table 3.2: List of all ADF keys used by the application

3.1.1.4. Card Security Settings

Protection of the card

The security attributes on the MF level secure the CIPURSE™ card and inhibit unauthorized modifications of the card content. Therefore, the 1st MF key is updated during personalization to hold the “KD_CARD_MF_ADMIN” key.

This key and the corresponding ART & SMR (MF ART: “40 6B”, SMR: “48 14”) allow the administrator to update the key, create new applications or delete the whole card content to reuse the card. The first byte of the ART (the “Always” field) allows everyone to read the file attributes but is giving them no further access to any other data on the card.

All the other keys on MF level are invalidated and cannot be used.

Protection of the application

The security attributes on the ADF level secure the CIPURSE™ application on the card to avoid unauthorized data access and application manipulation.

The administrative operations can be handled using the 1st ADF key "KD_CARD_ADF_ADMIN", it allows the administrator to personalize and delete the application. These and further access rights for the other three keys used in the ADF are described in the ADFs ART & SMR (ADF ART: "40 72 40 40 40", SMR: "48 14").

Basically, this ART only allows the administrator to do any interactions on ADF level. All the other keys are only allowed to read the file attributes, any changes of the application are prohibited.

Card data security

Based on the data confidentiality, the access to the content of the present elementary files is restricted. Therefore, the security attributes of the EFs are set individually by configuring the respective ART and SMR.

The security attributes of the different EFs can be seen in Figure 3.1.:

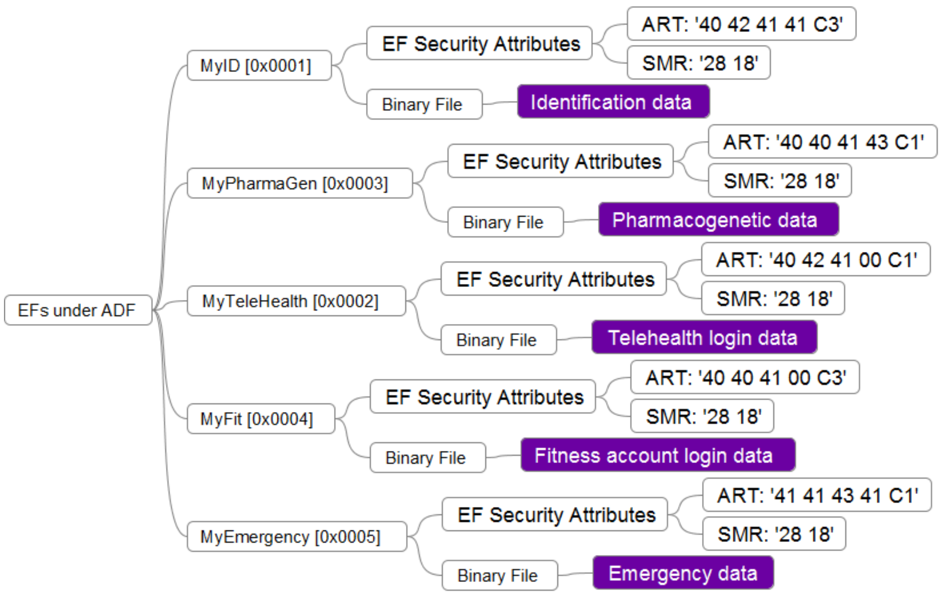


Figure 3.1: The applications elementary files and their security attributes

Although the administrator is downloading the application itself on the CIPURSE™ card, it is noticeable that the administrator is not able to read any of the personal data of the card owner. That is one of the crucial data security features of this application.

It is also important to remember that the card owner is able to change the security settings for the health professionals to inhibit or allow the data access to single EFs. The ARTs shown in Figure 3.1 do only represent the settings after the personalization of the card.

For example, if the card owner is blocking the access to the emergency file (MyEmergency) the original ART ("41 41 43 41 C1") would be replaced by the ART "00 41 00 00 C1".

Figure 3.2 shows the whole application with its present EFs, keys and the adjusted access rights for the different roles that are connected to these keys.

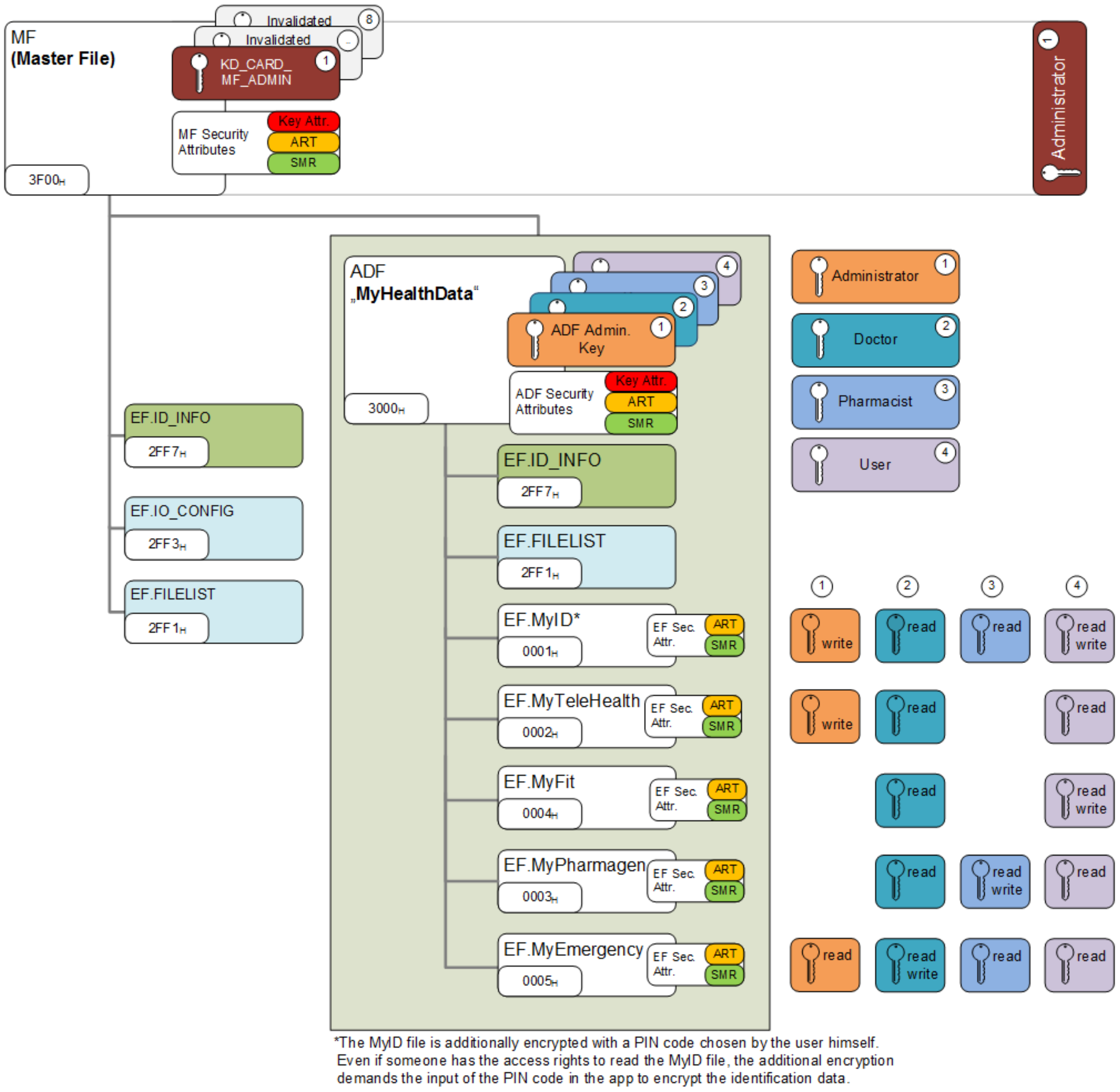


Figure 3.2: The applications file structure and a simplified visualization of the access rights to the different EFs

3.1.1.5. Card Application – Big Picture

The big picture (Figure 3.3) shows the defined roles for the CIPURSE™ application, their keys and the interactions to the card and between each other. The life cycle of the card can be derived from these considerations.

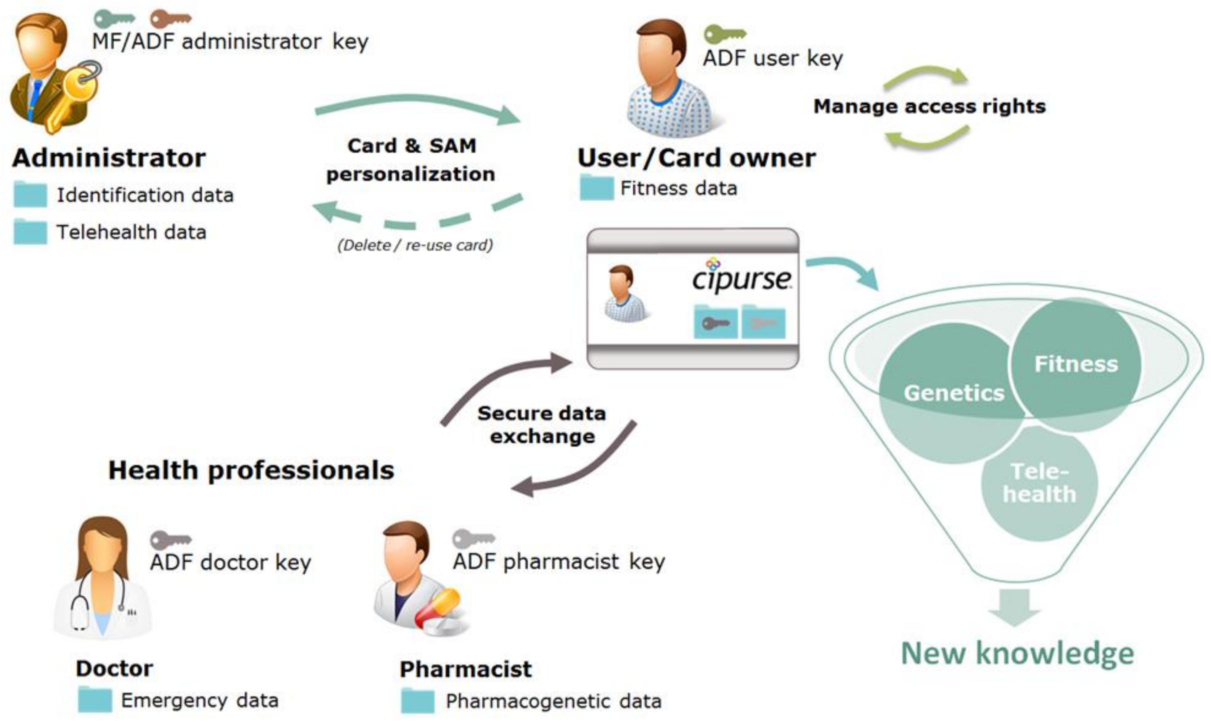


Figure 3.3: Big Picture of the card application

Administrator

The administrator owns the “KD_CARD_MF_ADMIN” and the “KD_CARD_ADF_ADMIN” key and is responsible to do the personalization of the user card. With the help of a personalization device the admin is able to personalize the user card by downloading the application on the CIPURSE™ device. During this process the file structure and necessary EFs are created, the security attributes are set and the basic user information (identification and telehealth data) is written into the “MyID” and “MyTelehealth” EF.

However, the administrator is not able to read the data on the user card but is capable of deleting the whole application to set the CIPURSE™ card back into its delivery state for re-personalization. This allows the re-use of older cards.

It is important to mention that all the administrative operations have to be executed in a secure environment to ensure data security.

Doctor

The doctor is a health professional. The key “KD_CARD_ADF_DOCTOR” allows a doctor to access all the data on the CIPURSE™ card. Additionally, doctors are the only ones who are authorized to write emergency data on the user’s card.

Pharmacist

The pharmacist is another health professional. The key “KD_CARD_ADF_PHARMACIST” allows a pharmacist to access the identification (“MyID”), pharmacogenetics

(“MyPharmaGen“) and emergency (“MyEmergency”) data on the CIPURSE™ card. Additionally, pharmacists are the only ones who are authorized to write pharmacogenetics data on the user’s card.

User/Patient (Card owner)

The user or patient is the card owner. The owner gets the personalized card from the administrator and is handling the physical card itself. The user is the only one who can collect or present personal data from or to the health professionals, giving the card owner full control over the data.

Additionally to the physical control, the user is also able to change the access rights of the individual files on his card. The unique “KD_CARD_ADF_USER” key allows the card owner to block the access to single files. If blocked by the user, a doctor or pharmacist cannot access files they normally would have access to. The card owners are also the only ones who can write the fitness account data on the card.

Like mentioned before, the card owner’s access rights are limited to the personally owned card. All the access rights of the other roles apply for the entire cards holding this application.

3.1.1.6. CIPURSE™ Card States – Life Cycle

The card states describe the different states the card is in during its life cycle, from the delivery to its operational use.

Delivery state

In the delivers state (Figure 3.4) the content of the CIPURSE™ card is according to the transport configuration. The chip holds the initial configuration and content including the “CARD_TRANSPORT_KEY” on MF Level and predefined elementary files on MF Level. There are no ADFs or application linked EFs on the card.

The “CARD_TRANSPORT_KEY” serves the administrator to access and personalize the card in its delivery state. This information needs to be shared between the card manufacturer and the administrator who is personalizing the card and the application.

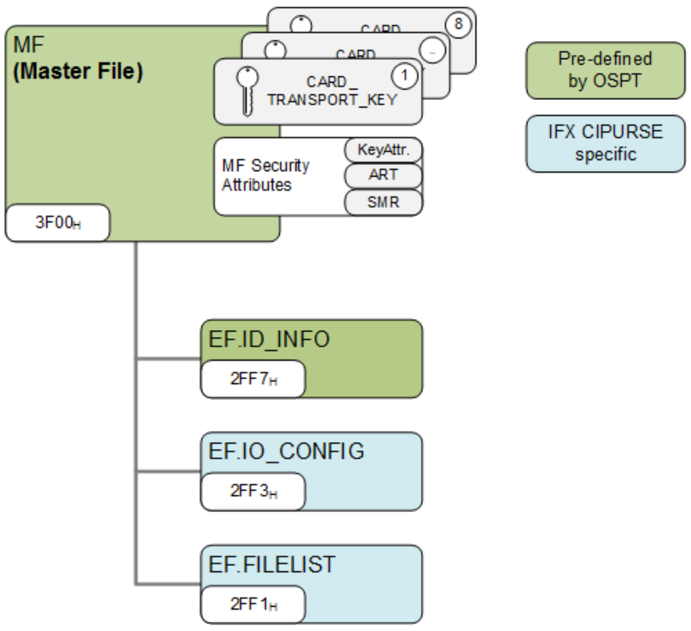


Figure 3.4: Visualization of the cards delivery state

Personalized state

During the personalization process the application ADF "MyHealthData" is generated with the necessary number of application keys (four). The security attributes (ARTs and SMRs) which inhibit unauthorized modifications are set and the four application keys are loaded onto the card. On MF level the MF key is updated and all the other keys on MF level are invalidated by setting the key security attributes accordingly.

Finally all the elementary files ("MyID", "MyTelehealth", "MyEmergency", "MyPharmaGen" and "MyFit") which will be used to hold the personal user information are created as well by the administrator.

After the personalization by the administrator, only the EFs "MyID" and "MyTelehealth" will contain user data, the missing data (in "MyEmergency", "MyPharmaGen" and "MyFit") will be added by other application roles like the doctor during operation.

The card will be delivered to the card owner after the personalization, in its personalized state. Figure 3.5 shows the card content after the personalization process.

Initialized state

Before the personalized card is ready for the operational use the card owner has to perform the final initialization step. When starting the corresponding app the first time, the user will be asked to enter a PIN code. This code is used to additionally encrypt the identification data in the "MyID" file. After this final step the card is ready for the operational use.

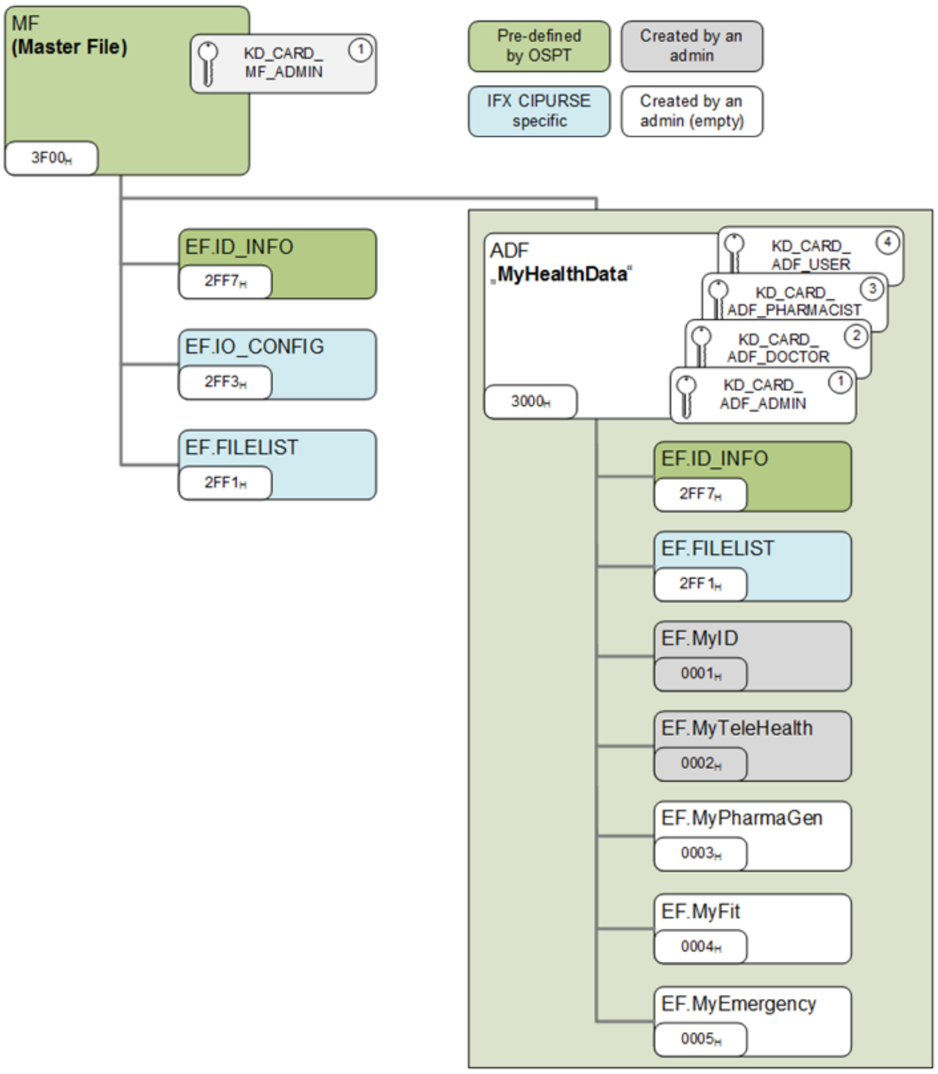


Figure 3.5: Visualization of the cards personalization state

Life cycle

During the operational use it is also possible that the card owner returns the card to the administrator. This would allow the administrator to delete the application to return the card into the delivery state. Figure 3.6 shows the whole life cycle of the card with its different states.

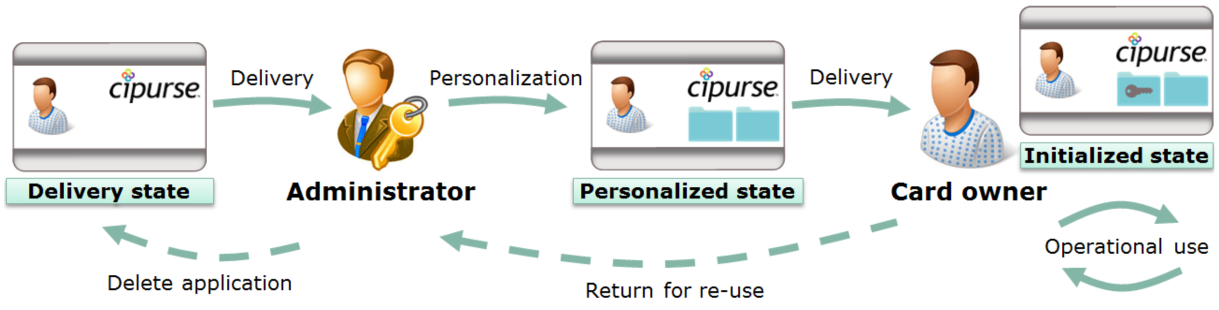


Figure 3.6: Visualization of the cards personalization state

3.1.2. CIPURSE™ SAM Application

An application can be realized by using only general SAMs (for the card personalization, the operational use and back office transactions) or by combining multiple SAMs which are only offering dedicated functionalities.

The advantage of using only SAM_Gs would be that all the SAMs are of the same type with the same content and could be used for the personalization as well as for the operational use in a terminal. This means that all SAMs would hold the same secrets, which would lead to higher security and misusage risks. Therefore, this solution is not recommended.

The benefit of using multiple SAMs with dedicated functionalities (key loading, personalization, terminal operation) is that the whole application is additionally safeguarded by the SAM hierarchy. The hierarchy ensures that the SAMs are only holding the keys they inevitably need for their purpose.

For example, in this case a terminal SAM would only hold keys for the operational use and none of the keys which are necessary to personalize a CBP. At the same time the keys for the personalization process would only be present on the SAM used in the personalization device. This supports the overall security by reducing the risk in case of an attack.

3.1.2.1. SAM Types and Hierarchy

Due to its advantages for security, this application uses three different SAM types and five different SAMs for the present roles and purposes involved in this use case. Therefore, a key and SAM hierarchy needs to be developed. Table 3.3 shows all five SAMs used in this application.

SAM name	SAM type	Role	Description
SAM_K	Load SAM	Application administrator	Generated and used by an administrator to personalize all other SAMs for this application.
SAM_P	Personalization SAM	System administrator	Allows the administrator to personalize or delete the user cards.
SAM_T1	Standard SAM	User	The card owners SAM for the operational use.
SAM_T2	Standard SAM	Doctor	SAM distributed to doctors for the operational use.
SAM_T3	Standard SAM	Pharmacist	SAM distributed to pharmacists for the operational use.

Table 3.3: List of SAMs used in this application

SAM_K (Load SAM)

The SAM_K is used to personalize all other SAMs according to the applications requirements. The SAM_K itself is generated and configured by an application administrator (application and system administrator can be the same person) using a personalization device (the host).

The host needs to know some initial secrets (Card and SAM transport keys, default SAM password) in order to be able to operate CIPURSE™ cards and CIPURSE™ SAMs in delivery state. This information is defined by the OSPT alliance.

For the SAM_K configuration the administrator needs to define keys (the SAMs MF and ADF key) to secure the SAM_K and its application. Also, to prevent misusage, a SAM password needs to be selected. The additional keys (KGMs) that are used for the personalization process of the other SAMs are generated and stored by the SAM_K itself.

SAM_P (Personalization SAM)

The SAM_P is used by the system administrator to personalize the CIPURSE™ cards. It is needed to create the "MyHealthData" application ADF including the users EFs and to load the necessary keys on the card (on MF and ADF level).

SAM_T$_1$ (Standard SAM)

Standard SAMs are deployed in terminals for operational use. The SAM_T$_1$ is the card owner's SAM. It is holding one unique key that allows the user to access his CIPURSE™ card. In this scenario the terminal is a smartphone that is used to write and read data to or from the card.

SAM_T$_2$ (Standard SAM)

The SAM_T$_2$ is almost identical to the SAM_T$_1$ and serves the same purpose in a terminal. However, the SAM_T$_2$ is used by doctors, therefore, it is holding a different key with different access rights to the user cards.

SAM_T$_3$ (Standard SAM)

Like the SAM_T$_2$ it is almost identical to the SAM_T$_1$. In this case the SAM is used by pharmacist and is therefore, also holding a different key with different access rights to the user cards.

SAM hierarchy roles

Like mentioned before, there are four roles for this application, the administrator, doctor, pharmacist and user. In the SAM hierarchy the administrator role can also be separated into application and system administrator. However, it is not mandatory to have two administrators.

A SAM hierarchy allows the splitting of responsibilities. In this example an application administrator with the SAM_K is holding all essential secrets to do the personalization of the SAMs. While a system administrator with the SAM_P holds the essentials for the card personalization, which is critical for the operational use.

The possibility for a wider distribution of responsibilities creates a more flexible and secure environment for possible implementations.

SAM hierarchy

There are dependencies between the multiple SAMs and keys, thus a SAM hierarchy develops for this application. Since all SAMs derive from the SAM_K it is always on top, being the root of two branches. One is for the personalization process and the other one for the operational use of the CIPURSE™ cards. Figure 3.7 is showing the SAM hierarchy and how the SAMs relate to each other in a bigger picture.

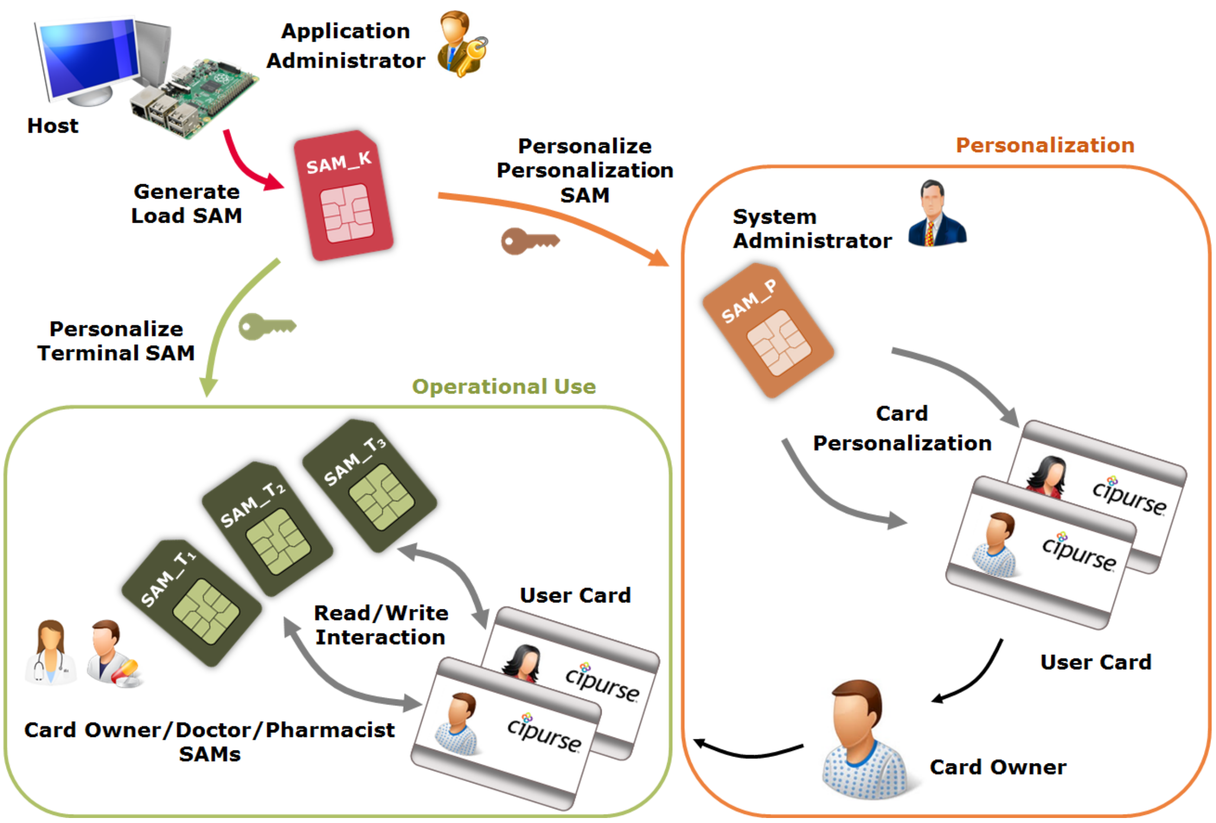


Figure 3.7: SAM hierarchy

3.1.2.2. Key Notation and Key Diversification

Key names

The format to name the keys used in this application is the following:

[KeyType]_[DestinationProduct]_[DesinationtFile]_[Role]

[KeyType]	Description
KGM	Grand master key
KM	Master key
KD	Derived key

Table 3.4: List of used key types

[DestinationProduct]	Description
SAM	Key is intended to be loaded onto a CIPURSE™ SAM
CARD	Key is intended to be loaded onto a CIPURSE™ card

Table 3.5: List of used products for the key notation

[DestinationFile]	Description
MF	Key is intended to be used as a MF key
ADF	Key is intended to be used as a ADF key

Table 3.6: List of used file types for the key notation

[Role]	Description
ADMIN	Key is intended to be used by an administrator
DOCTOR	Key is intended to be used by a doctor
PHARMACIST	Key is intended to be used by a pharmacist
USER	Key is intended to be used by the card owner

Table 3.7: List of used roles for key naming

For example:

The generated key (KGM) in a SAM_K's personalization keyset for the ADF of a doctor's SAM is called "KGM_SAM_ADF_DOCTOR". The ADF key on the card owner's card, which is derived from user-specific diversification data, is called "KD_CARD_ADF_USER".

There are also two special cases, the first one is the transport key ("CARD_TRANSPORT_KEY" or "SAM_TRANSPORT_KEY"). This key is used when the card or SAM is shipped to a costumer. The second one is the MF key of the SAM_K ("KD_KSAM_MF_DEFAULT"). This key is the transport key diversified using SAM-specific diversification data. It is used in this example to secure the SAM_K. If used as a product or in the field, this key has to be exchanged with a secret key only the administrator is aware of.

Key diversification

There are three types of key diversification data used in this CIPURSE™ SAM application:

- Keyset-specific (constant) diversification data (has to be defined for the application)
- Card or SAM-specific (unique) diversification data
- User-specific (unique) diversification data based on the user's identification data

Using keyset-specific and card/SAM-specific data is a common approach, but to use user-specific data for the card owner's key diversification is a new method introduced in this thesis. It is necessary to ensure that only the card owner himself has the user-specific access rights to his own card. Therefore, the key ("KM_CARD_ADF_USER") is derived with the unique and confidential identification data (Name and insurance number) of the card owner to receive a unique key ("KD_CARD_ADF_USER"). The whole process of the card owner key diversification can be seen in Figure 3.8.

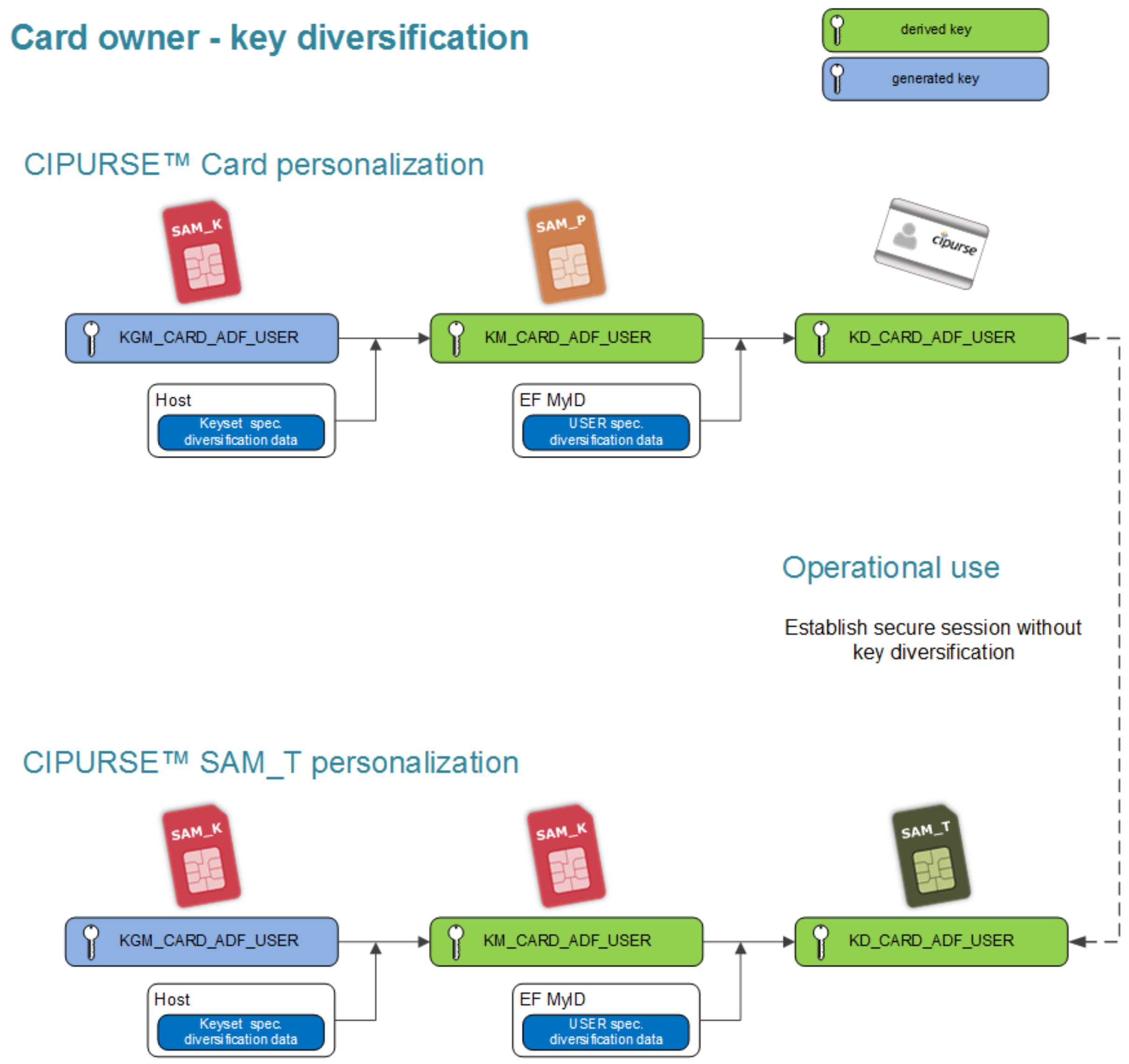

Card owner - key diversification
derived key
generated key
CIPURSE™ Card personalization
SAM_K
SAM_P
cipurse
KGM_CARD_ADF_USER
KM_CARD_ADF_USER
KD_CARD_ADF_USER
Host
Keyset spec. diversification data
EF MyID
USER spec. diversification data
Operational use
Establish secure session without key diversification
CIPURSE™ SAM_T personalization
SAM_K
SAM_K
SAM_T
KGM_CARD_ADF_USER
KM_CARD_ADF_USER
KD_CARD_ADF_USER
Host
Keyset spec. diversification data
EF MyID
USER spec. diversification data

Figure 3.8: Card owner key diversification procedure

For the personalization of the card it is required to personalize the SAM_P. Therefore, the generated KGM of the SAM_K is derived using the keyset-specific diversification data resulting in a KM. This KM on the SAM_P can now be used in the card personalization. By deriving the key with user-specific diversification data, we receive the unique KD key that can be stored on the user’s card during personalization.

The usage of user-specific data also requires the adaptation of the card owners SAM_T personalization process. Hence the KGM of the SAM_K needs to be derived using the keyset-specific diversification data and stored on the SAM_K again. The resulting KM can now be used to derive the key with user-specific diversification data, giving us the unique KD for the user’s SAM_T.

Thanks to this approach the keys on the SAM_T and the user card are both KDs, which mean that it is possible to establish a secure session during an operational usage without further key diversification. At the same time the key used is unique, giving the card owner exclusive user-specific access rights.

In comparison, the key used by a doctor (or pharmacist) to access the data on a CIPURSE™ card (“KD_CARD_ADF_DOCTOR”) is derived using card-specific diversification data and the key “KM_CARD_ADF_DOCTOR”. This way, all doctors have the same access rights on all the user cards. The whole process of the doctor’s key diversification can be seen in Figure 3.9.

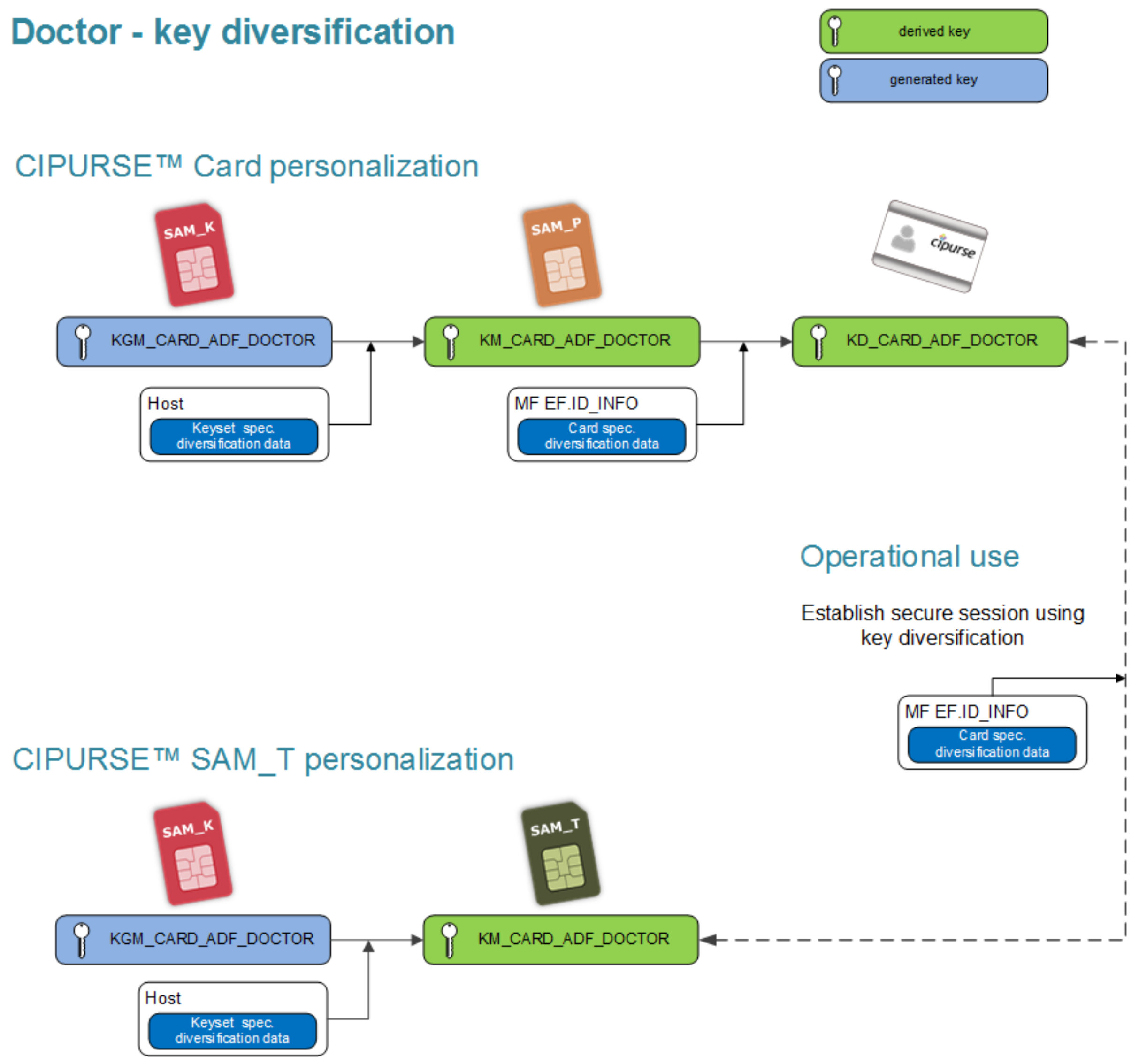

Doctor - key diversification
derived key
generated key
CIPURSE™ Card personalization
SAM_K
SAM_P
cipurse
KGM_CARD_ADF_DOCTOR
KM_CARD_ADF_DOCTOR
KD_CARD_ADF_DOCTOR
Host
Keyset spec. diversification data
MF EF.ID_INFO
Card spec. diversification data
Operational use
Establish secure session using key diversification
MF EF.ID_INFO
Card spec. diversification data
CIPURSE™ SAM_T personalization
SAM_K
SAM_T
KGM_CARD_ADF_DOCTOR
KM_CARD_ADF_DOCTOR
Host
Keyset spec. diversification data

Figure 3.9: Doctor key diversification procedure

During the personalization of the card the diversification process of the doctor's key is almost identical to the card owner's key, with one difference. The diversification data used to get the KD is not user- but card-specific. This approach allows the key to have the same access rights on all cards.

For the personalization of a doctor's SAM_T the KGM of the SAM_K needs to be derived using the keyset-specific diversification data, giving a KM.

During the operational use the KM allows to establish a secure session with a user's card by using key diversification. Hence card-specific diversification data is used to get a matching pair of keys for a successful authentication.

3.1.2.3. Security Settings

Protection of the SAM

The security attributes on the MF level are securing the CIPURSE™ SAM itself and inhibit unauthorized modifications. Therefore, the 1st MF keys of all the SAMs are updated during personalization to hold the "KD_SAM_MF_ADMIN" key, which is derived from the SAM_K. But there is one exception, the SAM_K. For the personalization of the SAM_K the administrator has to choose another secret key for the MF (in this example the derived transport key is used to secure the SAM_K).

The MF key and the corresponding security attributes (MF ART: "40 FF", SMR: "59 19") give the administrator full access to the SAM. The first byte of the ART (the "Always" field) allows everyone to read the file attributes, giving them no access to any data. All the other keys on MF level are invalidated and cannot be used.

Protection of the SAM application

Next to the SAM password the security attributes on the ADF level secure the CIPURSE™ SAM application to avoid unauthorized data access and application manipulation.

Again there is just one key on ADF level ("KD_SAM_ADF_ADMIN"), which derives from the SAM_K during the personalization of the different SAMs. Like on MF level, the ADF key of the SAM_K application has to be chosen by the administrator during the personalization of the SAM_K.

The ADF key gives the administrator full access to the application (ADF ART: "40 FF", SMR: "19 19").

SAM File Security

The security attributes of the single elementary files are identical for all the used SAMs and can be seen in Table 3.8.

EF file type	ART	SMR
Keysets	40 C4	09 19
Key attributes	40 C3	29 19
SAM password file	40 C2	29 19
SAMInfo	40 C3	29 19
FILELIST	41 C3	29 19
IO_CONFIG	40 C3	29 19
ID_INFO (ADF)	41 C2	29 19
ID_INFO (MF)	41 C3	29 19

Table 3.8: SAM ART & SMR of the different file types

3.1.2.4. File Structure and Content

In this section the final file structure and the keysets on the different SAMs, which are used for this CIPURSE™ application, are discussed.

SAM_K

The SAM_K (Figure 3.10) is used to personalize the CIPURSE™ SAMs. The keys that enable the personalization of the SAMs are listed in the personalization keyset files. The keys within the operational keyset allow establishing a secure session to the other SAMs.

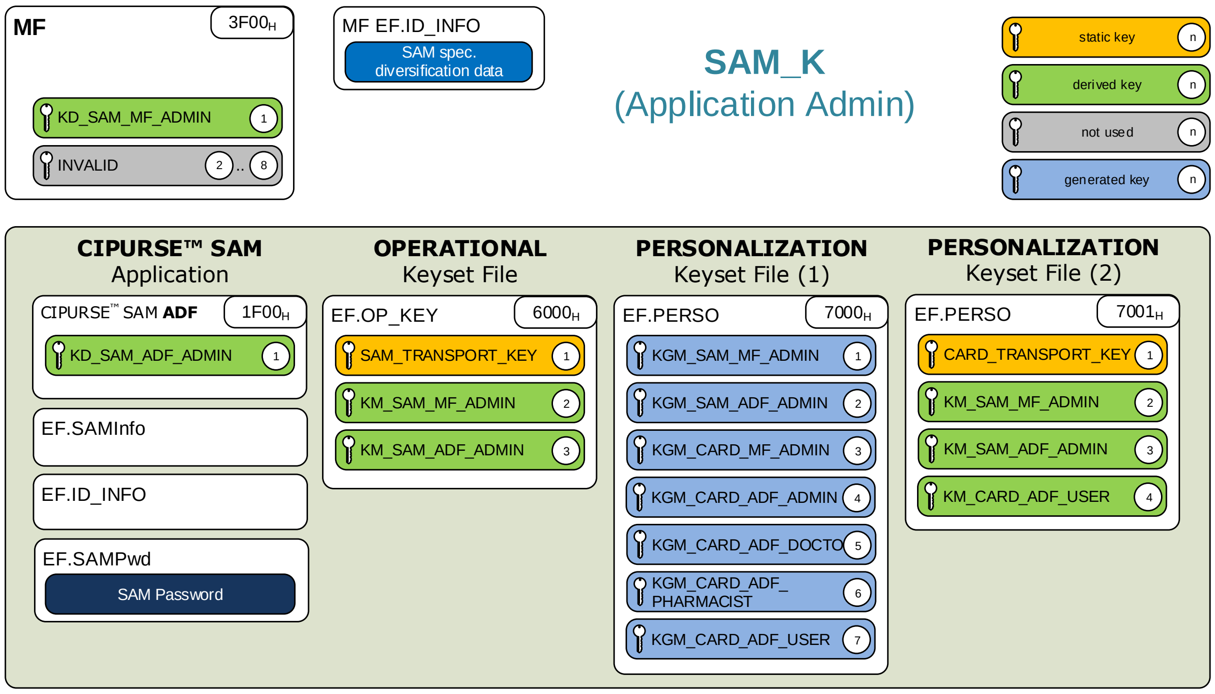


Figure 3.10: File structure and keys of the SAM_K

The operational keyset (FID 6000H) in the SAM_K needs to hold following keys:

Key name	Description
SAM_TRANSPORT_KEY	SAM transport key is used to establish initial secure sessions between the SAM_K and other SAMs for their personalization.
KM_SAM_MF_ADMIN	The MF administration key to establish a secure session between the SAM_K and other SAMs. It needs to be diversified with the SAMs-specific diversification data first.
KM_SAM_ADF_ADMIN	The ADF administration key to establish a secure session between the SAM_K and other SAMs. It needs to be diversified with the SAMs-specific diversification data and allows the administration of the SAMs application (according to the defined access rights).

Table 3.9: Keys of the operational keyset within the SAM_K

The personalization keyset 1 (FID 7000H) in the SAM_P needs to hold following keys:

Key name	Description
KGM_SAM_MF_ADMIN	Base key to secure other SAMs on MF level.
KGM_SAM_ADF_ADMIN	Base key to secure other SAMs application.
KGM_CARD_MF_ ADMIN	Base key for the card MF that is used in derived form to administrate the card.
KGM_CARD_ADF_ADMIN	Base key for the card ADF that is used in derived form to administrate the card application.
KGM_CARD_ADF_DOCTOR	Base key for the card ADF that is used in derived form as the doctor's key in operational use.
KGM_CARD_ADF_ PHARMACIST	Base key for the card ADF that is used in derived form as the pharmacist's key in operational use.
KGM_CARD_ADF_USER	Base key for the card ADF that is used in derived form as the card owner's key in operational use.

Table 3.10: Keys of the personalization keyset 1 within the SAM_K

The personalization keyset 2 (FID 7001H) in the SAM_K needs to hold following keys:

Key name	Description
CARD_TRANSPORT_KEY	Used to be loaded into the operational and personalization keysets of SAM_P. There it will be used to establish secure sessions with cards in delivery state.
KM_SAM_MF_ADMIN	Administrator key that will secure other SAMs on MF level after being diversified with the SAMs-specific diversification data.
KM_SAM_ADF_ADMIN	Administrator key that will secure other SAMs application after being diversified with the SAMs-specific diversification data.
KM_CARD_ADF_USER	The card owner's key needs to be stored as KM (diversified with keyset-specific diversification data) for the SAM_T1 personalization.

Table 3.11: Keys of the personalization keyset 2 within the SAM_K

On the administrative host-side there are some initial secrets that need to be known to operate an application on CIPURSE™ cards and CIPURSE™ SAMs. On one hand there are the card and SAM transport keys needed to access empty cards and SAMs in delivery state.

On the other hand, there are the SAM_K's MF and ADF keys and its SAM password. These have to be defined by the application administrator. This information is needed for the personalization of the SAM_K.

The prior defined keys for the SAM_K are applied during the personalization of the SAM to secure the SAM and its application. They should also be derived with the SAM_K-specific diversification data to support the application security, even though a SAM_K should be kept in a secure environment only.

The SAM password for the SAM_K is also defined by the administrator and loaded into the “EF.SAMPwd” file during its personalization.

A complete list of the secrets needed to personalize the SAM_K can be found in Table 3.12:

Secret	Description
KD_SAM_MF_ADMIN	The SAM_K’s MF administration key, needed to establish a secure session to the SAM_K. It is diversified with the SAM-specific diversification data and secures the card.
KD_SAM_ADF_ADMIN	The SAM_K’s ADF administration key, needed to establish a secure session to the SAM_K. It is diversified with the SAM-specific diversification data and secures the SAM application.
SAM Password (SAM_K)	It is needed to move the SAM_K to authorized state enabling all SAM operations.

Table 3.12: Secrets that need to be defined to personalize the SAM_K

The keyset-specific diversification data and the passwords for the other application SAMs are additional secrets. While the constant keyset-specific diversification data needs to be defined by the administrator, the SAM passwords for the other application SAMs are generated by the personalization device automatically for every SAM. It is essential that the SAM passwords are only provided to the authorized holder of the particular SAM.

Additional administrative secrets:

Secret	Description
Keyset-specific diversification data	Keyset-specific diversification data (static) is used to derive KMs from KGMs. The static data ensures that these keys are the same in all present SAMs.
SAM Password (other SAMs)	On all application SAMs a password will be set during the personalization process (in the “EF.SAMPwd” file). It is needed to move the SAM to authorized state enabling the SAM operation.

Table 3.13: Additional secrets that need to be defined by the administrator

SAM_P

The SAM_P (Figure 3.11) is used to personalize the CIPURSE cards. The keys on MF and ADF level of the cards are KDs, so the necessary KMs to perform the operations need to be present within the SAM_P key files. These keys, which enable the personalization of the cards, are listed in the personalization keyset file. The keys within the operational keyset allow to establish a secure sessions to the cards.

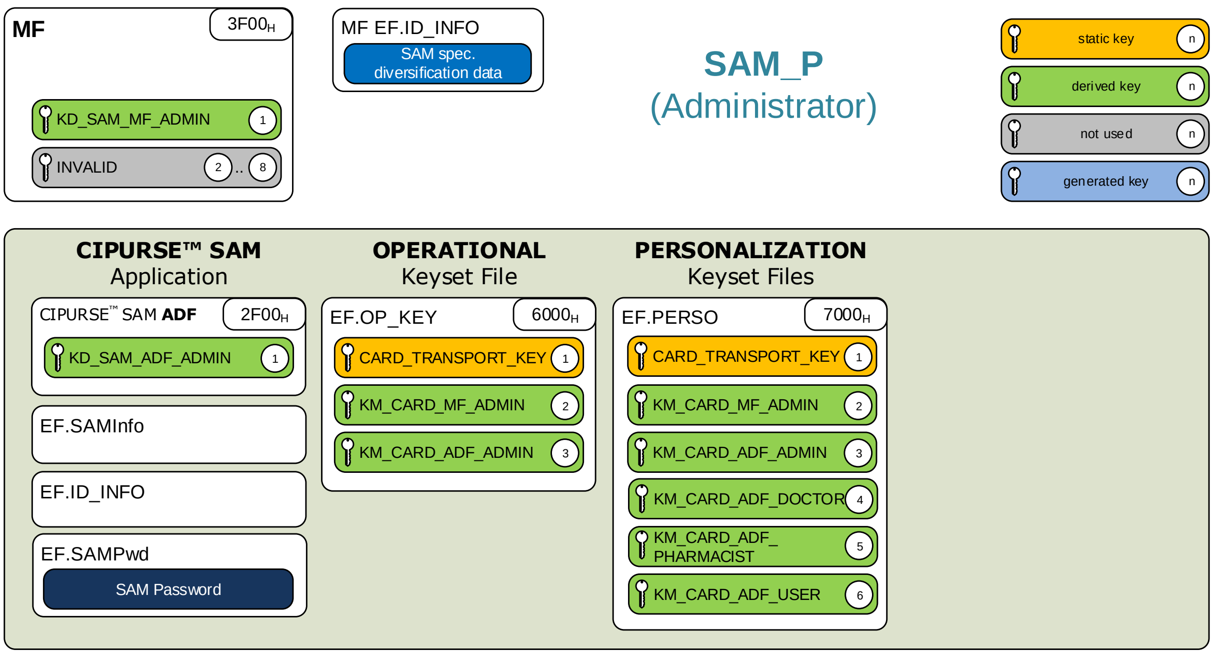


Figure 3.11: File structure and keys of the SAM_P

To secure the SAM application, the ADF has been created with an application key ("KD_SAM_ADF_ADMIN") to allow the administration of the SAM application. On the MF level one administration key ("KD_SAM_MF_ADMIN") is used to secure the SAM on MF level. All these keys derive from the SAM_K.

The operational keyset (FID 6000H) in the SAM_P needs to hold following keys:

Key name	Description
CARD_TRANSPORT_KEY	Allows the administrator to authenticate with cards in delivery state.
KM_CARD_MF_ADMIN	Administration key which replaces the transport key on the cards MF level, required to establish a secure session with the card for future card modifications.
KM_CARD_ADF_ADMIN	Administration key for the card application, required to establish a secure session for future application modifications on the card.

Table 3.14: Keys of the operational keyset within the SAM_P

The personalization keyset (FID 7000H) in the SAM_P needs to hold following keys:

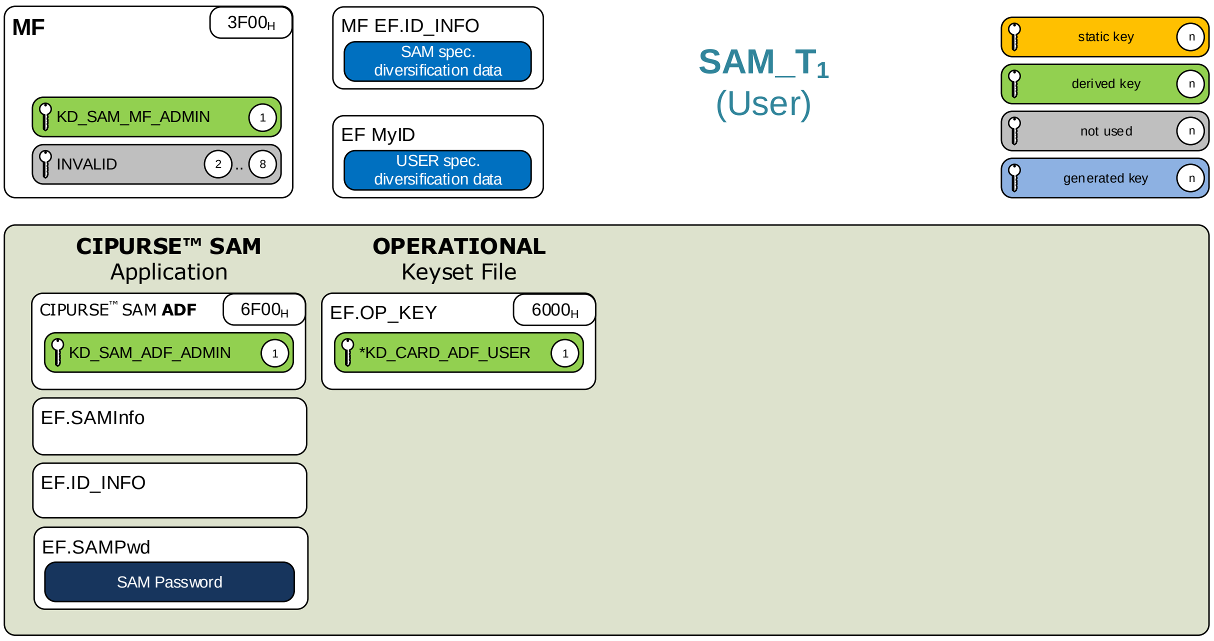
Key name	Description
CARD_TRANSPORT_KEY	To be able to restore cards to the delivery state.
KM_CARD_MF_ADMIN	Administrator key to secure the card's MF level. It will be loaded to the card after being diversified with card-unique information.
KM_CARD_ADF_ADMIN	Administrator key that will be loaded to the card's ADF after diversification with card-unique information.
KM_CARD_ADF_DOCTOR	Doctor key that will be loaded to the card's ADF after diversification with card-unique information.
KM_CARD_ADF_PHARMACIST	Pharmacist key that will be loaded to the card's ADF after diversification with card-unique information.
KM_CARD_ADF_USER	User key that will be loaded to the card's ADF after diversification with user-unique information.

Table 3.15: Keys of the personalization keyset within the SAM_P

SAM_Ts

Standard SAMs are configured for the operational use only. Therefore, they just have an operational keyset, allowing them to set-up secure sessions with a CIPURSE™ card. In this scenario the SAM_Ts are holding only one key, providing the proper access rights to the particular SAM. This application provides three different SAM_Ts (Figure 3.12, 3.13 & 3.14).

To secure the SAM applications the ADF has been created with an application key ("KD_SAM_ADF_ADMIN"). It allows the administration of the SAM application. On the MF level one administration key ("KD_SAM_MF_ADMIN") is used to secure the SAM on MF level. All these keys derive from the SAM_K.



*The USER key on the SAM_T1 is derived from the user specific data from the MyID file, unlike the other KM keys which are derived from the keyset diversification data

Figure 3.12: File structure and keys of the SAM_T1

The operational keyset (FID 6000H) in the SAM_T1 needs to hold following keys:

Key name	Description
KD_CARD_ADF_USER	Card owner's key (diversified with user-specific diversification data) allows establishing a secure session with the owner's card to read or write from or to his card.

Table 3.16: Keys of the operational key file within the SAM_T1

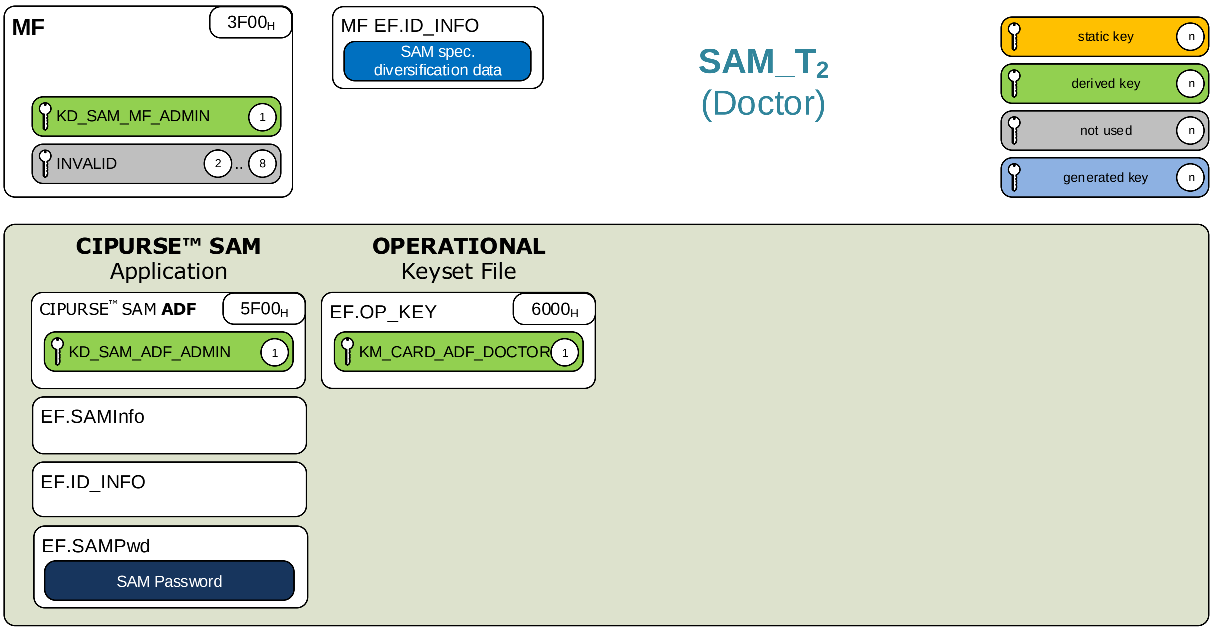


Figure 3.13: File structure and keys of the SAM_T2

The operational keyset (FID 6000_H) in the SAM_T2 needs to hold following keys:

Key name	Description
KM_CARD_ADF_DOCTOR	The doctor's key (diversified with keyset-specific diversification data) allows establishing a secure session with a card to read or write from or to it (depending on the defined access rights).

Table 3.17: Keys of the operational key file within the SAM_T2

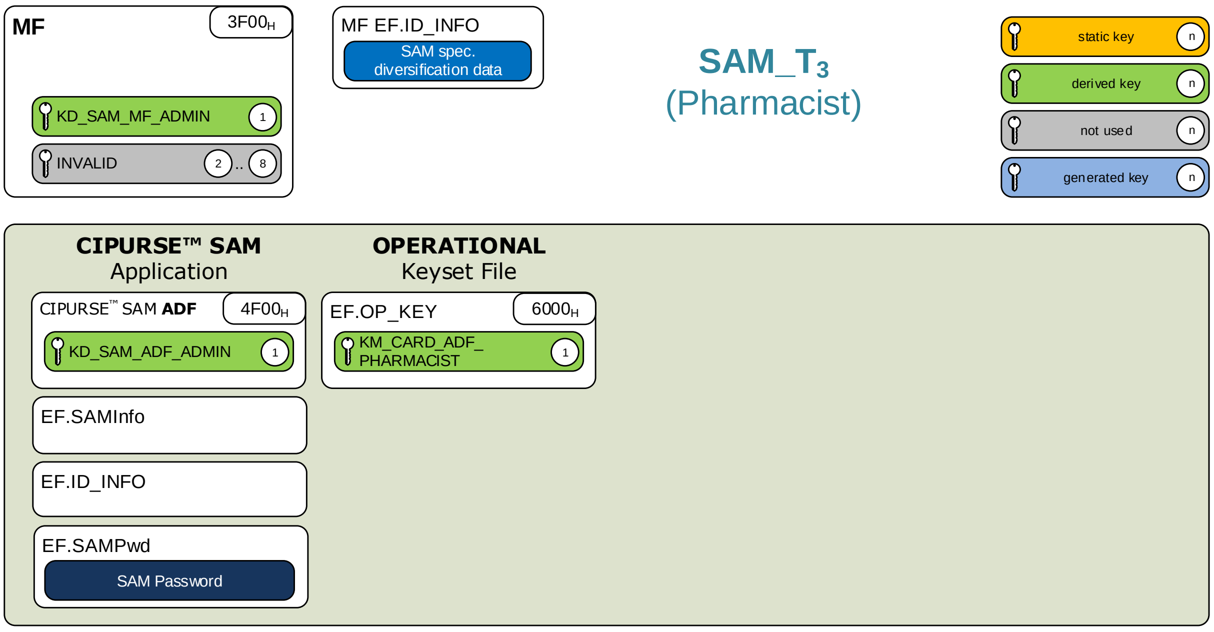


Figure 3.14: File structure and keys of the SAM_T3

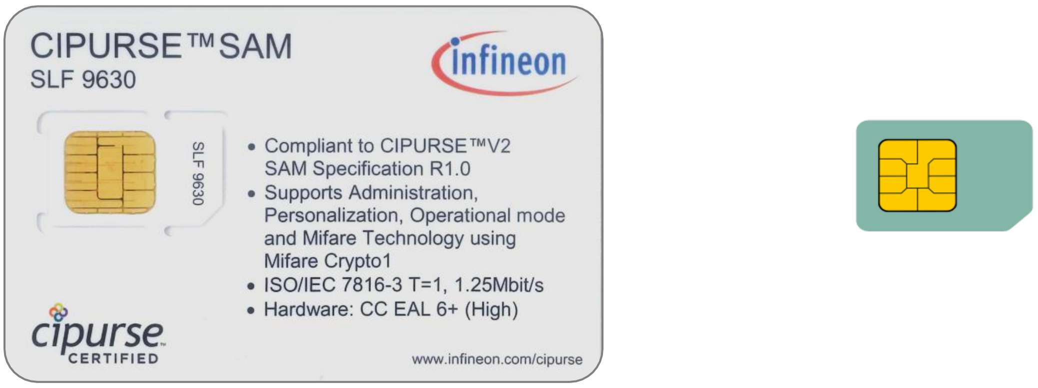
The operational keyset (FID 6000_H) in the SAM_T3 needs to hold following keys:

Key name	Description
KM_CARD_ADF_ PHARMACIST	The pharmacist's key (diversified with keyset-specific diversification data) allows establishing a secure session with a card to read or write from or to it (depending on the defined access rights).

Table 3.18: Keys of the operational key file within the SAM_T3

3.2. **CIPURSE™ SAM – Form Factor**

The CIPURSE™ SAM is a secure element based on the “ISO/IEC 7816” standardised smartcard technology. A common form factor for such a smartcard is the ID-000 format defined in the “ISO/IEC 7810” standard (Figure 3.15). Usually the ID-000 format is included into an ID-1 format card and can be detached if necessary**.** The ID-000 format is best known as the classical SIM card used in a mobile phone and the ID-1 format as a credit card.


CIPURSE™SAM
SLF 9630
Infineon
SLF 9630
• Compliant to CIPURSE™V2 SAM Specification R1.0
• Supports Administration, Personalization, Operational mode and Mifare Technology using Mifare Crypto1
• ISO/IEC 7816-3 T=1, 1.25Mbit/s
• Hardware: CC EAL 6+ (High)
cipurse CERTIFIED
www.infineon.com/cipurse
Figure 3.15: A CIPURSE™ SAM, ID-1 format on the left, ID-000 format on the right

Since it is the goal of CIPURSE™ to implement an open standard based secure data infrastructure using NFC, it is crucial to have the full CIPURSE™ security functionality available on mobile NFC devices like smartphones. This makes a SAM indispensable. Unfortunately, most mobile phones do only provide one SIM card slot. Therefore, it is unfavourable to use the ID-000 form factor to provide a smartphone with a CIPURSE™ SAM.

3.2.1. Form Factor - microSD

A possible solution to the form factor problem of the SAM is the Weltrend’s Secure microSD Turnkey Solution. The microSD card uses Weltrend’s smartSD controller which is able to combine storage space management with an “ISO/IEC 7816” interface (Figure 3.16). This means that it is capable of holding a secure element like a CIPURSE™ SAM, additional to its flash memory. All the needed background information for the implementation of the microSD card was provided by Weltrend through direct consultation.

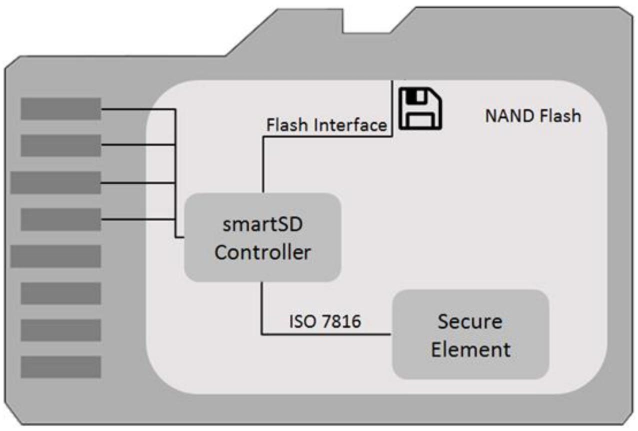


Figure 3.16: Diagram of Weltrend's Secure microSD Turnkey Solution

Thanks to this new CIPURSE™ SAM form factor, it is now possible to introduce the full CIPURSE™ functionality to a smartphone and other mobile devices with microSD card slots.

The present "ISO/IEC 7816" interface on the microSD card and the usage of an ID-1 form adapter (Figure 3.17) enables the possibility to use the microSD card like a common smartcard. This makes the usage very practical since there are no adaptions necessary to use this new form factor. At the same time it creates many new possible applications.

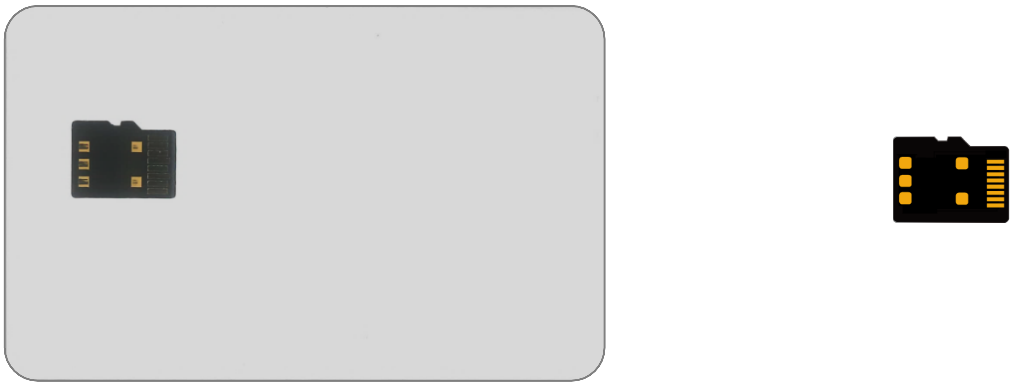


Figure 3.17: microSD card with "ISO/IEC 7816" interface (in an ID-1 form adapter on the left)

An interface file on the microSD card is used to provide the connectivity between an application like an Android App and the secure microSD card. When establishing a connection, this I/O file gets bind/mounted in order to be used to exchange commands and responses between the communication partners.

Any regular file can act as an I/O file, there are only two limitations. The filename must be in the 8.3 format (filename convention by Microsoft) and the letters have to be uppercase [25].

3.2.1.1. Improvements, Limitations & Security Concerns

Improvements and limitations

Weltrend is providing a library that allows establishing a connection between the secure element on the microSD card and an App on an Android smartphone. This library was in an early stage, therefore, I tried to give feedback to the developers during the Android App development. Thanks to the direct consultation and great support from Weltrend it was possible to improve some of the functionalities of the microSD and the library.

New firmware versions for the microSD card could noticeable improve the stability and adjusting the pulse per seconds (PPS) doubled the transaction speed, thanks to an enhanced signal synchronisation with a higher clock.

Unfortunately, there are two limitations on Android 4.4.2. This Android version restricts the microSD access permission of an App to nothing but its own directory. Therefore, it is important that the I/O file is located in the App's own directory. Also, the size of the I/O file must be 1.99GB (2147483136 bytes). That's because Android 4.4.2 disables direct input and output, this makes it necessary to use a larger I/O file to avoid file cache.

The library provided by Weltrend is considering these limitations. Although it is very inconvenient and not advisable to use Android 4.4.2 since the App takes several minutes during the installation just to create the large I/O file.

Security concerns

There were also some security concerns regarding the SD controller and the I/O file but after consulting with Weltrend it was possible to dissolve them. I assumed that it might be possible to access the SD controller or I/O file to read out sensible data transfers between an App and the microSD card. Fortunately, Weltrend assured us that the SD controller does not keep a record of the exchanged command. The controller is only passing commands to the secure element, so if somebody is trying to read out sensitive data from it, it will not work.

The SD controller actually reads the commands first and will recognize if it is a standard memory write/read or an I/O command. The I/O commands do not get written in the interface file, it is passed directly to the SE chip.

Assuming someone would remove the SD card while sensitive data is exchanged between the App and the secure element, the attacker has no possibility to access any transferred data. The I/O file would be empty and does not contain any, possibly sensitive, information at all.

3.2.1.2. Alternative SAM Integrations

There are also alternative solutions to the form factor problem of the secure element. For example, if a smartphone has no microSD card slot, a Bluetooth card reader could be used to connect your phone to a CIPURSE™ SAM. Another approach would be to provide a secure connection via the internet to a server which is directly connected to a SAM.

There are and there will be several possible solutions to provide full CIPURSE™ functionality to a mobile device, but this thesis is focused on the novel Android based integration of the microSD card form factor and all its advantages.

3.3. **Personalization Device**

The personalization device is an essential administrative element in the infrastructure of the presented CIPURSE™ application. The device is used by the administrator to personalize all the SAMs that are needed for the particular use case. Additionally, it is also capable of performing the personalization of the user card and it's formatting for future reuse.

The administrative functions of the personalization device include:

- Generation and personalization of the SAM_K
- Personalization of the SAM_P, SAM_T1, SAM_T2 & SAM_T3
- Personalization of the user card
- Formatting the user card (reuse purpose)

In a real-world implementation it would make sense to separate the different administrative functions (SAM_K, SAM_P and card personalization) into multiple devices. This improves the flexibility and assignment of tasks. However, it was my goal to build a prototype that is capable of handling the whole administrative functionality needed for this application.

3.3.1. Hardware

The personalization device is based on a Raspberry Pi 3 board. The Raspberry Pi 3 provides a fast and stable Linux platform that is portable, low-cost and has a good availability. It offers direct accessible processor pins as GPIOs and a multitude of different extensions, shields and add-ons. Additionally, it is also CE and FCC certified, which makes it a very attractive solution for prototyping.

In addition to the Raspberry Pi there is some additional hardware needed to build the personalization device, all of which is CE certified:

Raspberry Pi 7" touchscreen

The Raspberry Pi touchscreen is displaying the graphical user interface and thanks to the integrated touchscreen, it is also acting as an input device. It is the only in- and output device needed to control and interact with the device.

Identiv SCR3500A

The Identiv SCR3500A is a compact, contact-based USB smart card reader. Two of these readers are used as a personal-computer/smart-card (PC/SC) reader in the personalization device. They are needed to communicate with the SAMs, for example during a SAM-SAM or a SAM-NFC card interaction.

NXP PN7120 NFC controller single board computer (SBC) kit

This NFC kit includes a Raspberry Pi interface board and a PN7120 NFC controller board and is capable to provide full NFC functionality to the Raspberry Pi.

Although there is already an integrated NFC antenna in the controller board, a new self-made antenna was designed, to be able to integrate it into the chassis of the personalization device.

3.3.2. Setting up the Raspberry Pi 3

The first essential step to setup the Raspberry Pi is to choose an operating system (OS). This decision should be based on your requirements and needs for your project. Depending on your selected system it might be necessary to install additional drivers, packages and software for the software development.

3.3.2.1. Operating System

There is a wide range of different operating systems and next to multiple Linux distributions there is also a Windows 10 Version available. One of the most common Linux distributions for the Raspberry Pi is Raspbian. It is based on the Debian Linux distribution and is optimized for the hardware of the Raspberry Pi.

NXP, who is providing the NFC controller, is also offering a current Raspbian OS for the Raspberry Pi, with preinstalled drivers for their PN7120 NFC Controller SBC Kit. By using this OS you can avoid the quite complex installation process of the NFC controller drivers. There are still several packages to install for the additional hardware in use, but in this case the installation process is plain and easy.

The OS (OM5577 Raspberry Pi Linux Demo Image$^{(\text{REV } 1.2)}$) can be downloaded as an image file on the NXP website [26]. After downloading, it can be loaded onto a SD card by using tools like Win32DiskImager. The SD card can now be inserted into the Raspberry Pi, which makes it ready to boot up.

3.3.2.2. Drivers & Packages

The next essential step to setup the Raspberry Pi is the installation of all the drivers and packages that are essential for the used hardware. Depending on the Linux distribution in use there might be some differences. Some operating systems might require more or less hardware drivers to be installed. This depends on the pre-installed packages.

The following instructions are based on the OM5577 Raspberry Pi Linux Demo Image$^{(\text{REV } 1.2)}$ provided by NXP with the PN7120 NFC Controller SBC Kit.

Raspberry Pi 7" touchscreen

Fortunately when using the official Raspberry Pi 7" touchscreen there are no drivers to install. The panel works immediately without further adaptations after connecting it as specified. Next to the good functionality of the touchscreen, that’s one of the biggest advantages to other displays.

Identiv SCR3500A

There are several packages that need to be installed to get the PC/SC reader to work. Essential are universal serial bus (USB) programming, chip card interface device (CCID) and PC/SC libraries.

Most of them can be installed using the advanced packaging tool (APT) in the console. There is just one CCID driver that has to be downloaded from the website [27]. This is the website of the “Movement for the Use of Smart Cards in a Linux Environment” (M.U.S.C.L.E.) project, providing CCID and PC/SC drivers for Linux.

All the other essential packages can be installed with the “sudo apt-get install” command and are listed below:

- libusb-dev libusb++-0.1-4c2
- pcscd
- libccid
- libpcsclite1
- libpcsclite-dev
- libusb-1.0-0-dev

NXP PN7120 NFC controller single board computer (SBC) kit

The biggest advantage of using the Raspberry Pi image provided by NXP with the NFC kit, is the preinstalled driver for the NFC controller. The information on the NXP website, on how to install the drivers for the controller is very vague, therefore, I recommend using the provided operating system.

Additional packages

Additional packages that need or should be installed are:

Package name	Description
libnfc5	The installation of this package is necessary to be able to add the library functions for the NFC controller into the development environment.
libgtk3.0-dev & libgtkmm-3.0-dev	These libraries are used for the GUI in the application. While GTK is the name of the basic C library, GTKMM is the addition for C++.
pcsc-tools	This tool is not necessary, but it offers useful commands to obtain

	more information on the used PC/SC readers. For example, the command “pcsc_scan” shows all connected readers and their status.

Table 3.19: List of additional software packages that need or should be installed

With this specified Raspberry Pi setup it should be possible to compile and execute the Code::Blocks project for the personalization device for the CIPURSE™ health data application.

3.3.2.3. Integrated Development Environment (IDE)

When it is your intention to develop software on the Raspberry Pi or any other computer, it is recommended to use an IDE. An IDE is an application that provides several features and facilities for computer programmers and is therefore essential for software development. There is a wide range of tools with different features and it depends on the developer’s preferences and circumstances on which to choose.

I have chosen Code::Blocks as IDE to develop the application directly on the Raspberry Pi. Code::Blocks is a free C and C++ IDE that offers many practical features. At the same time, it is slim enough and not overloaded to keep the hardware requirements down. This makes Code::Blocks a well-balanced IDE, providing comfort and speed for developing on a Raspberry Pi.

The application can easily be installed using the APT on the terminal. The necessary command is “sudo apt-get install codeblocks”.

After creating a new project in Code::Blocks, the user interface allows you to easily add additional libraries by adding them to your linker and compiler settings in your project build options. For this application it was necessary to add three libraries to the project. The “gtkmm” library is used for the GUI, “libpcsclite” to establish a connection to the PC/SC reader and “libnfc5” to include the connection to the NFC controller. Make sure that these packages are installed on your operating system before adding them to your project. Next you can add them to your linker and compiler settings:

In the linker settings, select “other linker options” and add:

- \`pkg-config gtkmm-3.0 --libs\`
- \`pkg-config --libs libpcsclite\`
- \`pkg-config --libs libnfc5\`

In the compiler settings, select “other options” and add:

- \`pkg-config gtkmm-3.0 --cflags\`
- \`pkg-config libpcsclite --cflags\`
- \`pkg-config libnfc5 –cflags\`

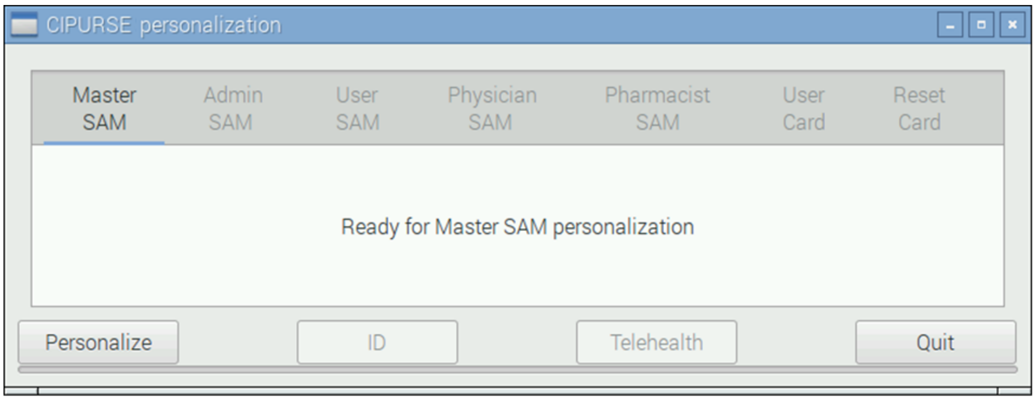
3.3.3. Personalization Application

This section describes the C++ application that was implemented on the personalization device to execute all needed administrative tasks.

3.3.3.1. Graphical User Interface (GUI)

The graphical user interface allows the administrator to interact with the application of the personalization device via the touchscreen. It is possible to execute all the necessary steps of the personalization process and a simple output will inform the administrator if the actions were successful.

The GUI is split into three sections. The upper part shows seven tabs, each for the personalization of a different kind of CIPURSE™ product. In the middle is the status indicator and on the bottom there are multiple buttons to select files or trigger an action. The interface can be seen in Figure 3.18 and the functions of the buttons and tabs are explained in Table 3.20 and Table 3.21.



Ready for Master SAM personalization

Figure 3.18: Graphical interface of the personalization device

List of the buttons present in the GUI:

Button	Description
Personalize	Starts the proper personalization script. It will only be selectable if all requirements for the selected personalization process are met. For example, when personalizing a user card, the card has to be present on the NFC reader before the button can be selected.
ID	Opens a file dialog to select the identification data for the card owner. The button is only selectable in the “User SAM” and “User Card” tab.
Telehealth	Opens a file dialog to select the telehealth data for the card owner. The button is only selectable in the “User Card” tab.
Quit	Quits the application.

Table 3.20: Buttons of the graphical user interface

List of the tabs present in the GUI:

Tab	Description
Master SAM	When selected, this tab can be used to personalize a SAM_K. An empty SAM needs to be inserted into one of the two PC/SC readers. After that the "Personalize" button can be pressed to execute the script.
Admin SAM	Allows the personalization of a SAM_P. An empty SAM needs to be inserted into one and a SAM_K into the other PC/SC reader.
User SAM	This tab allows the personalization of a SAM_T1. Next to the insertion of an empty SAM and a SAM_K, it is also required to provide the identification data of the card owner using the "ID" button.
Physician SAM	For the personalization of a SAM_T2. An empty SAM needs to be inserted into one and a SAM_K into the other PC/SC reader.
Pharmacist SAM	For the personalization of a SAM_T3. An empty SAM needs to be inserted into one and a SAM_K into the other reader.
User Card	This tab is used to personalize the CIPURSE™ card for the card owner. First the card owner's identification and telehealth data have to be selected using the proper buttons. With the SAM_P and an empty card in place on the readers, it is possible to start the personalization process.
Reset Card	When there is a personalized card on the reader, it is possible to delete its content to reuse the card.

Table 3.21: Selectable tabs of the graphical user interface

3.3.3.2. Terminal Library

The terminal library for C or C++ is not a classical library that can be integrated into a project to access the included functions. In reality it is the full source code (c- and h-files) that can be added to your project and adapted to your needs if necessary.

Adaptations

To use the terminal library on the personalization device it is required to adapt the library to the hardware in use. Since the PAL layer is exposing the hardware functionality via platform neutral interfaces, all that's necessary to ensure compatibility are some minor changes in this layer. The other elements of the library are designed to work on multiple platforms.

The mayor changes have to be made in the *TransReceive()* function found in the file IFXCTL_Comms.c of the PAL layer. *TransReceive()* executes the functions that are sending the commands directly to the particular reader device. Since the library is designed to work with PC/SC readers only, some adaptations have to be made in order to use the NFC controller as contact-less reader.

To be able to handle multiple readers so called handlers are created for every device. These handlers are committed to *SCardTransmit()* by the *TransReceive()* function. *SCardTransmit()* provides the interface to the PC/SC readers and is able to differentiate between multiple readers with the help of this handler.

Since the NFC controller is not a PC/SC reader it has to be made sure that *TransReceive()* is using the correct interface when sending commands to the NFC card. Therefore, the *nfcTag_transceive()* function has to be implemented into *TransReceive(). nfcTag_transceive()* is the interface provided by the NXP library to send commands to the NFC reader.

The first step is to make sure that *TransReceive()* is able to differentiate between a command for a contact-based PC/SC reader and the NFC controller. Therefore, if the command is intended for the contact-less NFC reader, the handler is set to NULL during the initialization. This way, the function is aware that the target device is the NFC controller if there is no handler committed to the function. Now it is possible for *TransReceive()* to call *nfcTag_transceive()* if the application wants to communicate with the NFC card or *SCardTransmit()* if a SAM is the target device.

These adaptations are not necessary if there are only PC/SC readers used in your application. Although it is important to mention that it is always mandatory to use the correct reader names of your hardware when creating a handler for your reader.

In this application the reader names are set in the file IFXCTL_DataStore.h. IFXCTL_DataStore.h is not part of the terminal library, it is only used to hold commonly used variables and functions in this application.

3.3.3.3. Application Sequence

In this section I would like to describe how the basic structure of the application looks and works like, for example by explaining and illustrating a sequence diagram.

The *Main()* function starts the graphical user interface class (*GuiWindow())* which is handling the whole user interaction. Therefore, it is also responsible for starting further operations.

When the user is executing a personalization process by pressing the button “Personalize”, the function *InitializeSAM()* initializes the readers needed to handle the selected operation. Next the *InitializeTargets()* function is opening the reader handlers, which are required by the *InitializeSession()* function to create a new session.

After the session was successfully launched and the whole initialization process has finished, the GUI class can execute the function for the chosen personalization process. The graphical interface will also keep the user informed about the status of the current operation. Finally, the user has the possibility to perform another personalization or to quit the application by pressing the button “quit”. The described procedure is visualized in the sequence diagram in Figure 3.19.

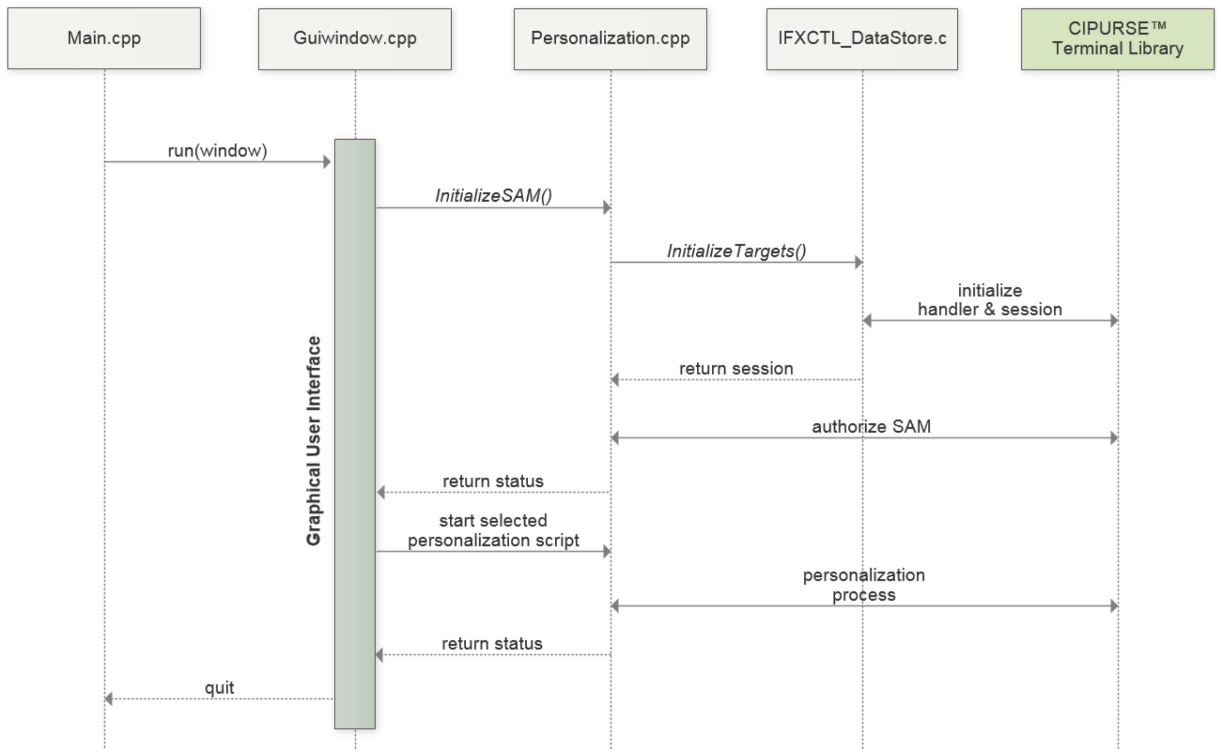


Figure 3.19: Sequence diagram of the personalization device application

Personalization functions

Like mentioned before, it is possible to choose between the different personalization processes by selecting the appropriate tab in the GUI. Each of these seven tabs will lead to the execution of the proper function which will start the command sequences necessary to fulfill the desired task. This sequence will handle the whole communication and data transmission necessary to reach the defined result for the CIPURSE™ application.

For example, the tab “Master SAM” will execute the function *PersonalizeKSAM()* after pressing the button “Personalize”. The function *PersonalizeKSAM()* itself will execute all the necessary commands to personalize a SAM_K. All the administrative commands needed to interact with the CIPURSE™ based product are provided by the terminal library.

The complexity of the different tasks varies and depends on the amount of data exchange that has to take place. Table 3.22 lists and describes all the seven functions handling the SAM and card personalization.

Note that the command and data exchange is not displayed by the GUI because of the significant amount of data that is exchanged in the process. For debugging reasons, it is possible to observe the data exchange via the consol.

Function name	Description
PersonalizeKSAM()	Personalizes the SAM_K. The function creates the application and additional essential files on the SAM, sets all the necessary configurations and security settings and secures the SAM with a SAM password. It is also generating the required keys which are later needed to personalize keyset files and secure cards and SAMs. Finally, the MF and ADF level is secured with the proper key.
PersonalizePSAM()	Personalizes the SAM_P for the system administrator. The SAM application and all essential files are created, all the needed keys are loaded into the personalization and operational keyset file, application and security settings are updated and a SAM password is set. Finally, the MF and ADF level is secured with the proper key.
PersonalizeT1SAM()	Personalizes the SAM_T1 for the card owner. The SAM application and all essential files are created, the card owner's key is loaded into the operational keyset file, application and security settings are updated and a SAM password is set. Finally, the MF and ADF level is secured with the proper key.
PersonalizeT2SAM()	Personalizes the SAM_T2 for the doctor. The SAM application and all essential files are created, the doctor's key is loaded into the operational keyset file, application and security settings are updated and a SAM password is set. Finally, the MF and ADF level is secured with the proper key.
PersonalizeT3SAM()	Personalizes the SAM_T3 for the pharmacist. The SAM application and all essential files are created, the pharmacist's key is loaded into the operational keyset file, application and security settings are updated and a SAM password is set. Finally, the MF and ADF level is secured with the proper key.
PersonalizeCARD()	Personalizes the CIPURSE™ NFC card. The card application and all essential files are created, the MF and ADF level is secured with the proper key, application and security settings are updated and the user files in the application are personalized.
ResetCARD()	Authenticates on the card with the administrator key to delete the application on the NFC card by formatting it.

Table 3.22: List of the personalization functions

Next I would like to give an insight about how a command sequence of the personalization process of a SAM can look like. Therefore, Table 3.23 is listing the full ordered sequence of the *PersonalizePSAM()* function, showing all essential steps for the personalization of the SAM_P, with a short explanation.

	Commands	Description
#1	*FormatAll*	Formatting of the card to be sure that it is empty (should not be necessary and is only possible if the card is not secured)
#2	*SelectFileByFID*	Selecting the file EF.ID_INFO on MF level
#3	*ReadBinary*	Read chip identification data from the EF.ID_INFO file (will be needed for the key diversification)
#4	*CreateADF*	Loading the SAM_P application with all necessary settings and the derived ADF keys (from SAM_K) onto the SAM
#5	*EstablishSecureChannel*	Establish secure session with the ADF administrator key
#6	*SetSMIValue*	Setting secure messaging mode to encrypted
#7	*CreateElementaryFile*	Create EF.SAMInfo file (stores basic SAM information)
#8	*UpdateBinary*	Configure EF.SAMInfo file for SAM_P
#9	*CreateElementaryFile*	Create EF.SAMPwd file (stores the SAM password)
#10	*UpdateBinary*	Writing the SAM password into EF.SAMPwd
#11	*CreateElementaryFile*	Create personalization keyset attribute file
#12	*UpdateRecord*	Update personalization keyset attribute file
#13	*CreateElementaryFile*	Create personalization keyset file for 6 keys
#14	*CreateElementaryFile*	Create operational keyset attribute file
#15	*UpdateRecord*	Update operational keyset attribute file
#16	*CreateElementaryFile*	Create operational keyset file for 3 keys
#17	*PerformTransaction*	Perform transaction (Mechanism against data loss, ensures that data is stored on the device)
#18	*VerifySAMPassword*	Verify SAM password to move the SAM to authorized state
#19	*LoadKey*	Loading transport key to the operational keyset file
#20	*LoadKey*	Loading transport key to the personalization keyset file
#21	*LoadKey*	Loading further (derived) keys to the operational keyset
#22	*LoadKey*	Loading further (derived) keys to the personalization keyset
#23	*SelectFileByFID*	Select EF.ID_INFO on ADF level
#24	*UpdateFileAttribute*	Update the EF.ID_INFO file security attributes
#25	*SelectFileByFID*	Select EF.Filelist on ADF level
#26	*UpdateFileAttribute*	Update the EF. Filelist file security attributes
#27	*PerformTransaction*	Perform transaction
#28	*SelectMF*	Selecting the master file
#29	*Reset_SecureSession*	Resetting the secure session
#30	*EstablishSecureChannel*	Establish secure session with the transport key on MF level
#31	*SetSMIValue*	Setting secure messaging mode to encrypted
#32	*SelectFileByFID*	Select EF.ID_INFO on MF level
#33	*UpdateFileAttributes*	Update the EF.ID_INFO file security attributes
#34	*SelectFileByFID*	Select EF.IO_CONFIG on MF level
#35	*UpdateFileAttributes*	Update the EF.IO_ CONFIG file security attributes

#36	*SelectFileByFID*	Select EF.Filelist on MF level
#37	*UpdateFileAttributes*	Update the EF. Filelist file security attributes
#38	*SelectMF*	Selecting the master file
#39	*Reset_SecureSession*	Resetting the secure session
#40	*EstablishSecureChannel*	Establish secure session with the transport key on MF level
#41	*SetSMIValue*	Setting secure messaging mode to encrypted
#42	*UpdateKeyAttributes*	Update MF key attributes
#43	*UpdateKey*	Replace the transport key with the MF administrator key on MF level
#44	*UpdateFileAttributes*	Update the MF security attributes
#45	*PerformTransaction*	Perform transaction

Table 3.23: List of all essential steps for the personalization process of the SAM_P

3.3.3.4. The Station

To get a solid station for the personalization device all the hardware is built into a wooden casing (Figure 3.20). The touchscreen for the interaction with the graphical user interface is arranged in the center of the wooden front. Next to the screen there are the two contact-based PC/SC readers enwrapped into the frame. At the bottom of the personalization device a wooden area is lead out. This area is used for the contactless communication with the card. Thus, an antenna is embedded on the bottom of this wooden area.

It is possible to remove the lid on the back of the station to get access to the build in hardware. There you can see the mounted Raspberry Pi on the back of the screen, a power distribution and the power supply itself (Figure 3.21).

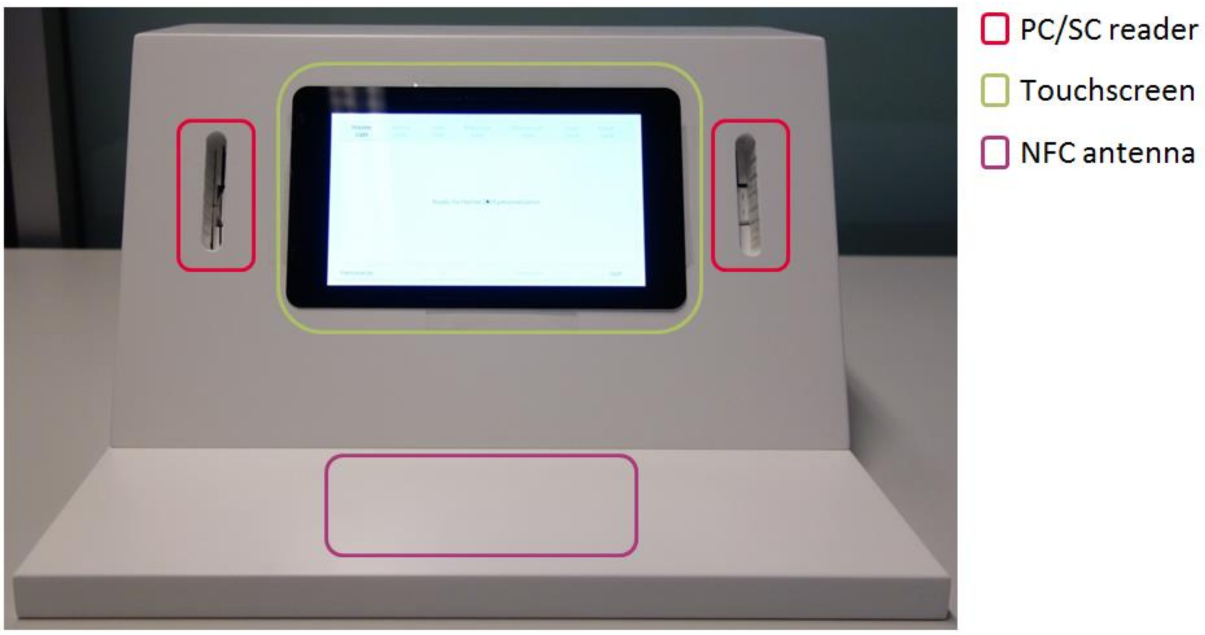


Figure 3.20: Personalization device (front)

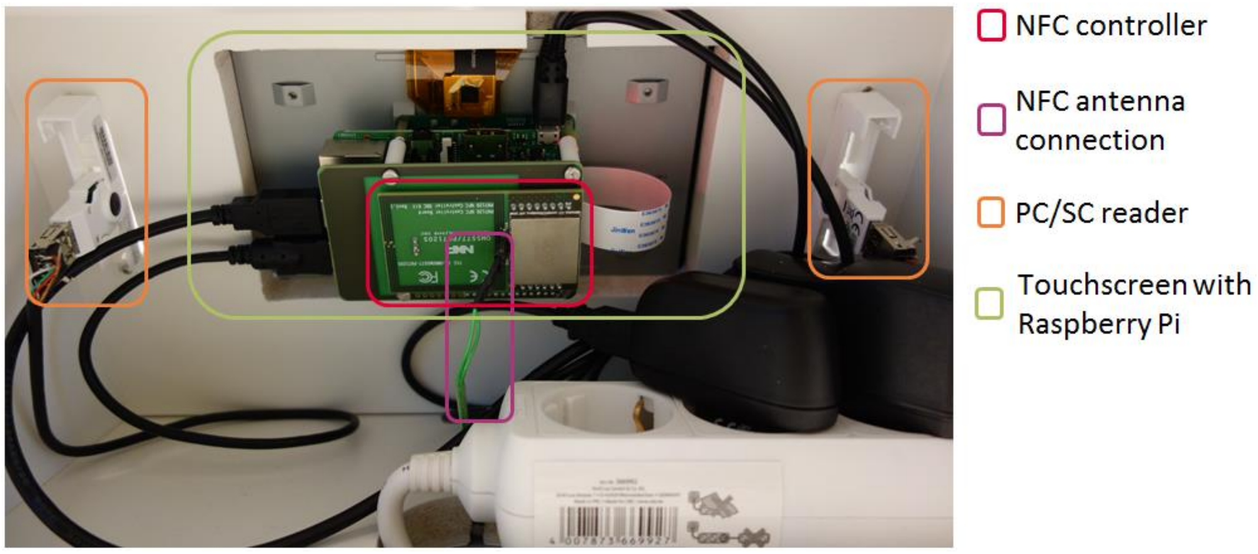


Figure 3.21: Personalization device (inside)

The NFC controller can also be spotted, it is mounted directly on the Raspberry Pi 3. The cable on the NFC controller leads to the custom NFC antenna which is embedded into the wooden bottom. Two USB cables are used to connect the contact based PC/SC readers to the Raspberry Pi 3 and the station is also providing enough space to allow retrospective changes and other optimizations, if necessary. Of course, it would also be possible to reduce the size of the station by changing the design and arrangements of the single components (e.g. keeping the power supply outside of the casing).

3.4. **Android Application**

The Android App "CIPURSE™ Health Data" is a prototype to show the feasibility of the full CIPURSE™ functionality on a smartphone using the microSD form factor as CIPURSE™ SAM, its functionality and GUI is kept quite simple.

The Apps purpose is to communicate with the CIPURSE™ card, thus reading and writing data from or to it (Figure 3.22). The smartphone is basically the terminal in this CIPURSE™ application scenario. It is the counterpart of the personalization device, intended for the operational use of the CIPURSE™ application.

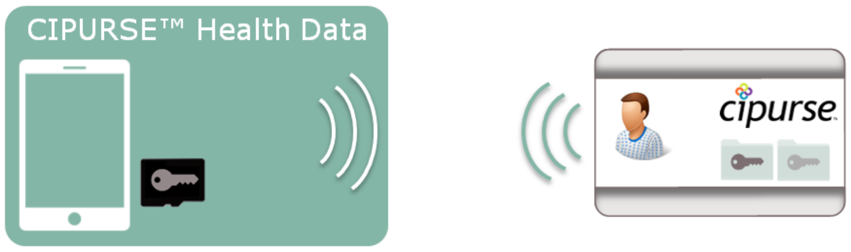


Figure 3.22: Android App interaction with NFC card.

In the operational use the Android App is used by doctors or pharmacists to access the secured data on the patients CIPURSE™ card. At the same time, it allows them to write data onto the card, if they are authorized to do so. The card owner himself uses the App to set the access rights on his own card. And all this functionality is facilitated by the CIPURSE™ SAM on the microSD card.

The Apps summarized main task is to provide a graphical user interface and the communication between a microSD SAM and a NFC card.

3.4.1. Graphical User Interface & Functionality

There are three different scenarios for the Android App and it is necessary to adapt the GUI and the functionality of the App to the appropriate one. What kind of scenario is needed depends on the role that is using the application. Therefore, it has to be clarified if it is a doctor, a pharmacist or the card owner himself who is using the App. This can be achieved by checking which kind of terminal SAM is present in the smartphone.

The GUI for a doctor or a pharmacist is very similar, but the GUI for the card owner stands out. That's because the required functionality for this use case is very different compared to the others. But there is also a similarity for all three scenarios.

At the first start of the App it asks all three types of users for the SAM password which is needed to authorize to the SAM. Without a successfully authorization to the SAM the application is not operating. In case of a doctor or a pharmacist the App is initialized after the successful authorization and is therefore showing the main GUI and awaiting the interaction with a NFC card.

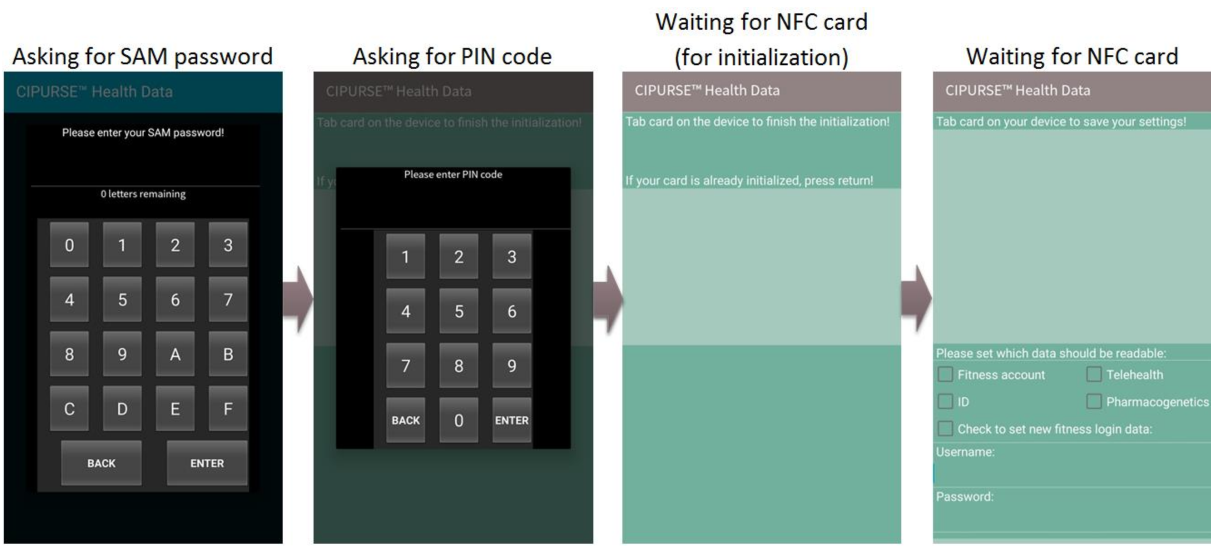
In case of the card owner, the application is asking for a PIN code. This PIN code is needed to additionally encrypt the identification data on the CIPURSE™ card with a XOR algorithm. After entering the code, the card owner has to tab the card to the phone to execute the encryption of the identification data. This process is the final step for the initialization of the NFC card which can now be used in the field.

Card owner's main GUI

The card owner needs the Android App to be able to set the access rights of the different elementary files on the CIPURSE™ card, setting a PIN code for the encryption of the identification data and to add fitness account data onto the card.

In an illuminated field on the top, all the output after the interaction with the NFC card is printed out. Underneath there are multiple checkboxes for the present files. If checked, the belonging files are accessible for authorized persons, otherwise the files are locked for everyone. Additionally there are two text fields to enter fitness account data and another checkbox that needs to be checked if the entered fitness data should be written onto the card. To transmit the changed settings to the card, the user has to tab the card to the NFC antenna of his phone.

Figure 3.23 shows all the different stages of the card owner's GUI.



Please enter your SAM password!

0 letters remaining

Please enter PIN code

Tab card on the device to finish the initialization!

If your card is already initialized, press return!

Tab card on your device to save your settings!

Please set which data should be readable:

☐ Fitness account ☐ Telehealth

☐ ID ☐ Pharmacogenetics

☐ Check to set new fitness login data:

Username:

Password:

Figure 3.23: Different stages of the GUI for the card owner (main window on the right)

Doctor's main GUI

A physician needs the Android App to readout the data on the card owners CIPURSE™ card and to add the card owner's personal emergency data onto the card.

After tapping the card on the phone, the accessible data on the NFC card is printed out in the illuminated area on top of the GUI. The button “Ask for ID information” opens a keyboard to ask the card owner to type in his or her PIN code, which is needed to encrypt the identification data on the card. If the PIN is correct the App will also read and print out the identification data on the card. The second button “Select Emergency File” opens a file dialog to select a file with personal emergency data of the card owner. The belonging “Write Emergency File” checkbox needs to be checked to write the emergency data onto the NFC card as soon as the card is tapped onto the phone.

Figure 3.24 shows the different stages of the doctor’s GUI.

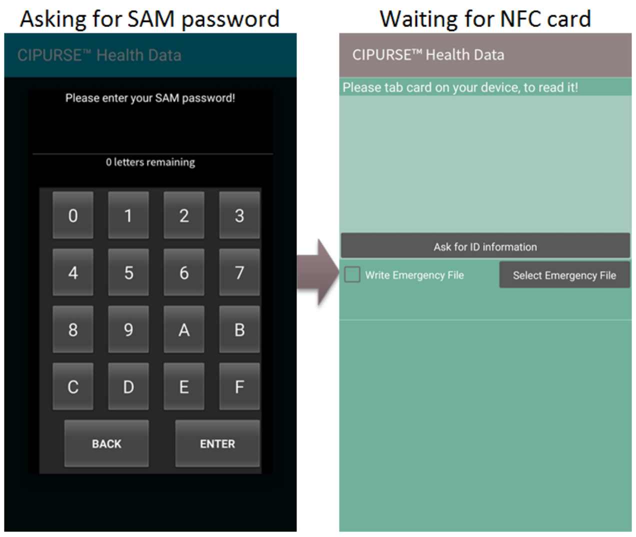


Figure 3.24: Different stages of the GUI for a doctor (main window on the right)

Pharmacist’s main GUI

The GUI of the pharmacist is very similar to the doctors, with the same functionality. There is just one difference, instead of writing an emergency file, the pharmacist is writing a pharmacogenetics dataset onto the CIPURSE™ card. Therefore, the button and the belonging checkbox are called “Select PharmaGen File” and “Write PharmaGen File”.

Figure 3.25 shows the different stages of the pharmacist’s GUI.

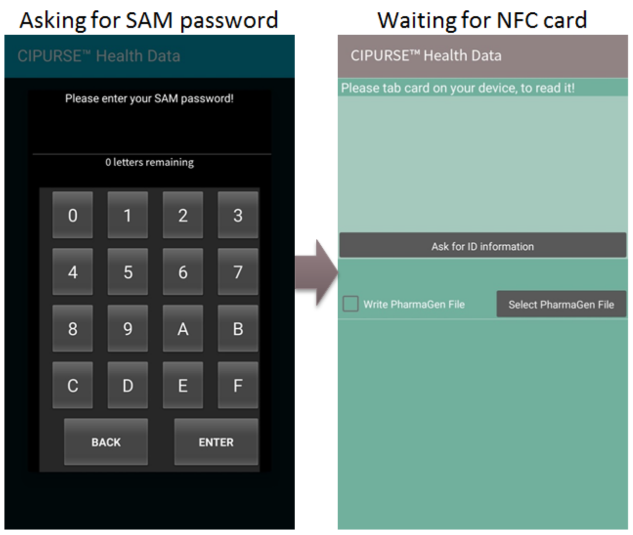


Figure 3.25: Different stages of the GUI for a pharmacist (main window on the right)

3.4.2. Security, Libraries & Application Sequence

3.4.2.1. Security

Security is one of the most important aspects for a use case that is handling personal data like health data. In case of the smartphone App the misuse of the microSD with a CIPURSE™ SAM is the most critical issue. Therefore, the user has to enter a 16 byte long SAM password at the first start of the application, allowing the application to authorize to the SAM. The password will be saved by the App, which provides a dependency between the users unique microSD SAM and the installed App on the device.

The App is designed on the assumption that the smartphone is secured and no one but the owner himself has access to the data and installed applications. This is a critical security issue, because once the App was initialized by entering the SAM password the App is always able to authorize to the SAM on the microSD card.

3.4.2.2. Libraries

This section mentions two libraries which are crucial for the App to provide the necessary functionality for operating CIPURSE™ on an Android phone. This does not concern the NFC or other standard Android libraries. The two libraries that have to be mentioned are the CIPURSE™ Terminal Library and the PCSC library that is providing the interface to the microSD card.

The CIPURSE™ Terminal Library was adequately discussed in the previous sections and although there are three different versions, they are used very similar. The three versions of the Terminal Library are for different programming languages (C, C++ and Java). The C++ version, which is used for the personalization device, is very similar to the C version, since C++ is an extension to C. Java on the other hand is an independent object oriented programming language and the library had to be completely adapted to it. The basic structure of the library, with its separated layers, is equal in all the present versions. This means, that all three versions are executed the same way when creating a CIPURSE™ command sequence. In case of Android the Java version is in use.

The PCSC library was provided by Weltrend and is used to implement the communication between the App and the secure element on the microSD card. Therefore, it is more appropriate to call it an application programming interface (API).

The basic functionality of this interface is provided by the *PcscJni* class which can be explained with an example flow of the communication process. The process flow is visualized in the flow chart in Figure 3.26.

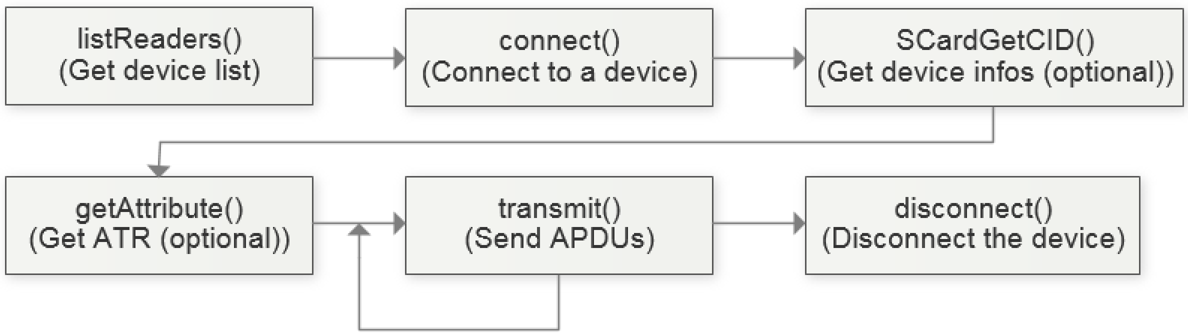


Figure 3.26: Flow of a communication process example using the PcscJni class

At first the class has to list all the present readers to be able to connect to one. After selecting the desired reader with its reader name, the connection to the reader device can be established. If successful it is possible to read out additional information like the “Answer To Reset” (ATR).

The ATR is the first information a smart card is transmitting to a card reader. It is conform to the “ISO/IEC 7816” standards and proposes communication parameters like the maximum clock frequency to the reader.

After the successful connection to the reader device it is also able to send and receive APDUs to and from the secure element of the microSD card. At the end of the data transmission the connection needs to be disconnected to avoid undesirable behavior.

The App and all the integrated libraries were developed and tested on a Samsung Galaxy S5 running Android 6.0.1.

3.4.2.3. Application Sequence

Next to some basic initializations the *MainActivity* class is running the whole graphical user interface activity for the Android application.

To be able to display the correct user interface for the different users (card owner, doctor or pharmacist), the application has to be aware of the type of terminal SAM that's present in the microSD slot of the phone. Therefore, one of the first crucial steps is to initialize a connection to the CIPURSE™ SAM.

To establish a connection to the SAM on the microSD card it is necessary to create a SAM handler. Such a handler is created by the class *SamCommunicationHandler* which utilizes the class *PcscJni* to establish a connection to the microSD card. *PcscJni* is operating the whole communication with the secure element on the microSD.

Next, the application has to be aware if it is the first time the App was started by checking if a SAM password is already present in the application data. To access this information the *sharedPreferences* interface is used.

If a password is available it is clear that the application was used before and is therefore also aware of the type of terminal SAM that is in use. It is now possible to directly authorize to the SAM using the *AuthorizeSam* class and the App can adapt the GUI to the suitable use case (card owner, doctor or pharmacist).

If there is no SAM password present it means that it is the first time the application is in use. Therefore, the *MainActivity* has to ask the user for the 16byte SAM password by calling the *HexadecimalKeyboard* class to open a hexadecimal keyboard. With the correct password entered, the application is able to authorize at the SAM using the *AuthorizeSam* class. This class is handling the authorization by utilizing the CIPURSE™ Terminal Library. By authorizing to the SAM it is also possible to learn the type of terminal SAM that's present in the smartphone. Depending on the present SAM_T type, the GUI is now able to adapt to the use case (card owner, doctor or pharmacist).

In case a SAM_T1 (card owner) is in use the class *DecimalKeyboard* is called to ask the card owner to enter a PIN code. This PIN code is later used to encrypt the present identification data on the CIPURSE™ card.

At this point the application is authorized to the SAM on the microSD card, it is aware of the use case and has therefore adapted the graphical user interface. All these steps are obviously only possible if the entered SAM password is correct. Otherwise the authorization would not work. This assures that it is not possible to misuse a SAM if stolen or lost.

Next the *MainActivity* is using the *nfcAdapter* class, provided by Android, to start a NFC listener. This listener is executing a callback function whenever a present NFC card is detected by the smartphone.

As soon as the phone has detected a NFC card the callback function is triggering the *CardComm* class, which is creating a NFC handler using the *NfcCommunicationHandler* class. The classe *NfcCommunicationHandler* is establishing the connectivity to the NFC card by implementing the Android class *IsoDep*. *IsoDep* itself provides direct access to “ISO 14443-4” properties and I/O operations on a NFC Tag.

With the SAM and NFC connectivity assured, the class *CardComm* is now able to handle all the CIPURSE™ communication using commands of the CIPURSE™ Terminal Library. After the successful data transaction, the status and the results are returned and shown on the graphical user interface. At this point the application is waiting for the next NFC card to interact with.

This basic overview of the application sequence is visualized in the sequence diagram in Figure 3.27.

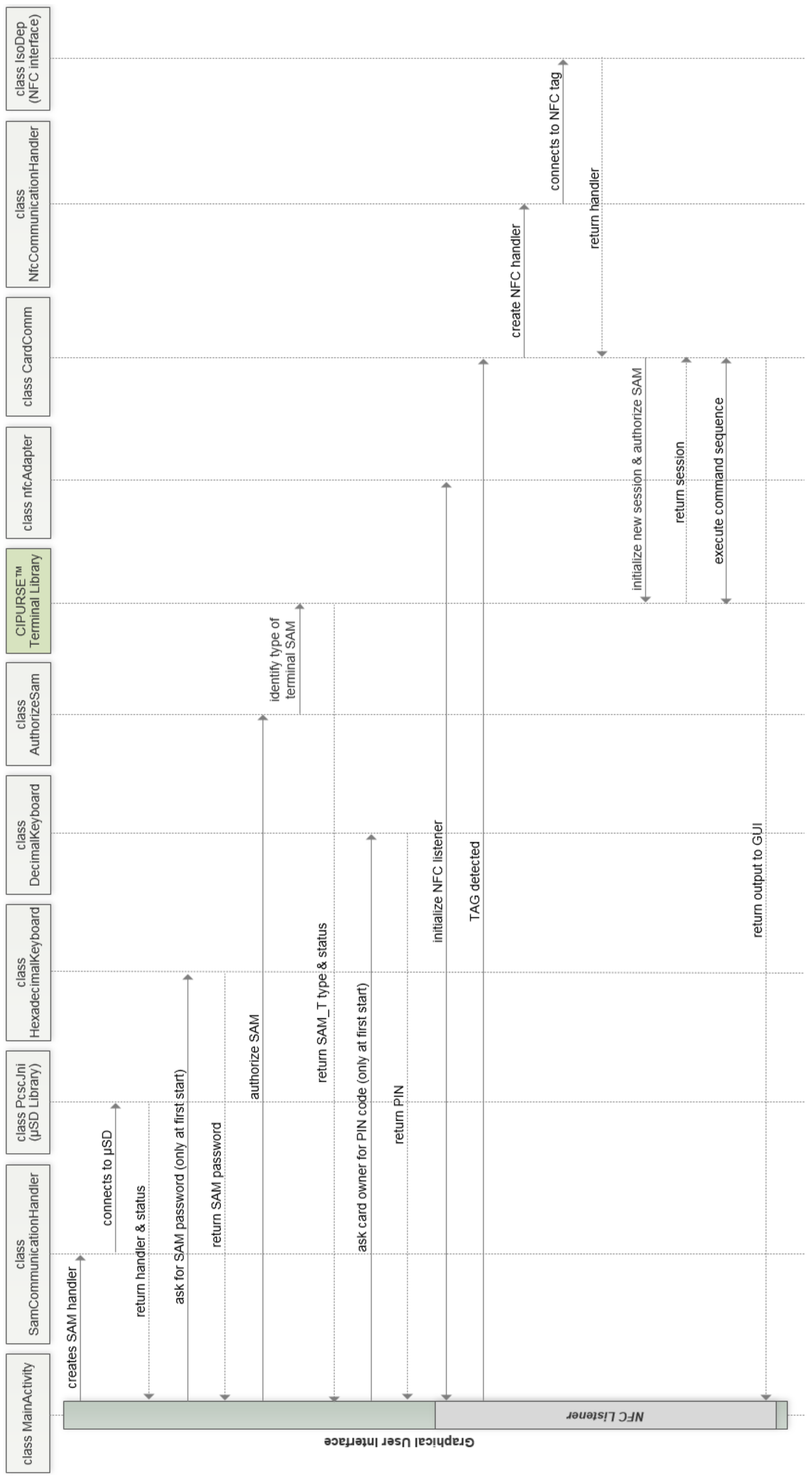


Figure 3.27: Sequence diagram of the Android App

4. Discussion and Conclusion

It was possible to reach all the predefined goals of the thesis. The implemented data infrastructure is capable to store and distribute various types of medical data. Data security and access rights management is integrated using the open CIPURSE™ standard, while the NFC technology is providing the chance for an easy integration into our everyday life at low-cost. Most importantly, a novel health data infrastructure was realized based on a patient-determined and patient-centred design, thereby providing an open and unique electronic health record system.

Inconveniences

While the utilities and the advantages of the elaborated data infrastructure have been highlighted already, it is also important to mention the drawbacks and inconveniences. These are mostly related to the requirement of a high security level.

The most obvious one may be that some people could be repelled by the notion that they are holding more responsibility for the data themselves. This concern is probably very individual and some people will favor the advantages of being in full control over the data.

A similar apprehension is the fact that all the personal information could get lost if stored on a contactless chip card. In that case, it has to be considered that the data is not lost irretrievable. The health professionals are still in possession of the single datasets, therefore, a new CIPURSE™ card could be personalized again. Unfortunately, this would still lead to an extra effort and burden for the card owner.

These two drawbacks are unavoidable and mandatory for the concept of this thesis and it has to be considered that paper data storage has the same disadvantages. However, a CIPURSE™ card could not be read out if it is lost, unlike paper.

Some security features also lead to inconvenient circumstances. Such as the fact, that it is quite sophisticated to change the access rights retrospectively. Once the SAMs and the NFC cards are personalized with the defined security settings and distributed to the customer, it is impossible to change any security settings without directly interacting with all the affected chip cards.

Eventually, while it is an important ambition, it will never be possible to eradicate all inconveniences in high security applications and services. That is why I think these drawbacks are negligible compared to the capabilities and opportunities provided by this implementation.

Potential improvements

Since the goal of the thesis was the development of a prototype, the focus on the performance of the software and the data transfer rates was subsidiary. Nonetheless, it was still possible to accelerate the data transmission of the microSD card with a firmware update.

Of course, there is still a lot of potential for further improvements by optimizing the Android application design and its graphical user interface to enable a better user experience. The same applies to the microSD firmware and its library, where further optimizations could allow even higher data transmission rates.

Prospective potential

The potential of novel data infrastructures in present health care systems is undeniable and the prospects go far beyond this thesis. Especially application areas for data-driven methods for clinical decision support and preventive healthcare are very promising and need further endorsement.

But there are more completely different use cases for the CIPURSE™ technology. For example, a NFC card holding keys for authentication processes. These could be used to implement a patient identity management with multi-factor authentication.

Another use case would be the integration of CIPURSE™ chips into patient wristbands, which are utilized in hospitals for patient identification. They could hold additional data to double check important information. While similar QR code based systems depend on a centralized server, a NFC tag would be independent by storing the data locally on the wristband.

I think this thesis also shows the potential of what is possible in addition to the known and established frameworks, when using new technologies like CIPURSE™. A technology originally intended for completely different applications.

Conclusion

The current lack of open methods for patient-centred health records was one of the main drivers for this thesis. The implemented use case is an example and a proposal for a new and unique solution. It is approaching present and future challenges regarding electronic health records by ensuring the utility of available health data to support data-driven methods for decision support systems.

In conclusion, the developed system provides the comfort of using smartphones, a patient-determined design and new capabilities to manage personal health data with a very high level of privacy protection and patient control. These features, in combination with the empowerment of the data subject itself, will hopefully endorse people to offer additional helpful data to improve diagnosis, treatment and healthcare in general through data-driven methods.

5. References, Standards, Lists & Tables

5.1. List of References

[1] Chaudhry B, Wang J, Wu S, Maglione M, Mojica W, Roth E, et al. Systematic Review: Impact of Health Information Technology on Quality, Efficiency, and Costs of Medical Care. Ann Intern Med. 2006; 144:742–752.

[2] Shekelle PG, Morton SC, Keeler EB. Costs and Benefits of Health Information Technology. Evidence Report/Technology Assessment No. 132. (Prepared by the Southern California Evidence-based Practice Center under Contract No. 290-02-0003.) AHRQ Publication No.06-E006. Rockville, MD: Agency for Healthcare Research and Quality. April 2006.

[3] Vogenberg FR. Predictive and Prognostic Models: Implications for Healthcare Decision-Making in a Modern Recession. Am Health Drug Benefits. 2009; 2(6):218–222.

[4] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016. (2016, April 27). Retrieved from http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf, last access: 10.1.2018

[5] NFC Forum, Specifications & Application Documents. Retrieved from https://nfc-forum.org/our-work/specifications-and-application-documents/, last access: 10.1.2018

[6] Coskun V, Ozdenizci B, Ok K. The Survey on Near Field Communication. *Sensors* 2015, *15*, 13348-13405.

[7] Madlmayr G, Dillinger O, Langer J, Scharinger J. Management of Multiple Cards in NFC Devices. In Proceedings of 8th IFIP WG 8.8/11.2 International Conference (CARDIS 2008), London, UK, 8–11 September 2008; pp. 149–161.

[8] St. Laurent AM. (2008). Understanding Open Source and Free Software Licensing. O'Reilly Media. p. 4. ISBN 9780596553951.

[9] Pearce JM. Quantifying the value of open source hardware development. Mod Economy 2015;6:1–11.

[10] Chao TE, Mody GN. The impact of intellectual property regulation on global medical technology innovation. BMJ Innov 2015;1:49–50.

[11] Williams A, Gibb A, Weekly D. Research with a hacker ethos: what DIY means for tangible interaction research. ACM Interact 2012;19:14.

[12] Doctorow C. (2014). Information doesn’t want to be free: laws for the internet age. McSweeney’s. ISBN: 9781940450285

[13] Beard HK. A survey on open source software licenses. Journal of Computing Sciences in Colleges, Volume 22 Issue 4, April 2007, Pages 205-211.

[14] Manar AT. Open Source Software in the UAE: Opportunities, Challenges and Recommendations (A Survey Research Study), Journal of Computer Science, Volume 13, Issue 6, 2017, Pages 165-174

[15] Rothwell R. (2008, August 5). Creating wealth with free software. Retrieved from http://freesoftwaremagazine.com/articles/creating_wealth_free_software/, last access: 15.11.2017

[16] Eckert JW. (2012). Linux+ Guide to Linux Certification (Third ed.). Boston, Massachusetts: Cengage Learning. p. 33. ISBN 9781111541538.

[17] Was ist Linux, Retrieved from https://wiki.ubuntuusers.de/Was_ist_Linux/, last access: 17.11.2017

[18] Bill A. (2014, Mai 13), Android is Just Another Distribution of Linux, Retrieved from http://www.all-things-android.com/content/android-just-another-distribution-linux, last access: 17.11.2017

[19] International Data Corporation. (2017, May). Smartphone OS Market 2017 Q1. Retrieved from https://www.idc.com/promo/smartphone-market-share/os, last access: 17.11.2017

[20] Raspbian internet presence, Retrieved from www.raspbian.org

[21] Niezen G, Eslambolchilar P, Thimbleby H. Open-source hardware for medical devices. BMJ Innovations. 2016;2:78–83

[22] Raspberry Pi Foundation internet presence, Retrieved from www.raspberrypi.org

[23] de Koning Gans G, Hoepman JH, Garcia FD. A Practical Attack on the MIFARE Classic, Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications, p.267-282, September 08-11, 2008, London, UK.

[24] Kocher P, Jaffe J, Jun B. Differential Power Analysis, technical report, 1998; later published in Advances in Cryptology - Crypto 99 Proceedings, Lecture Notes In Computer Science Vol. 1666, M. Wiener, ed., Springer-Verlag, 1999.

[25] Naming Files, Paths, and Namespaces. Retrieved from https://msdn.microsoft.com/en-us/library/aa365247.aspx, last access: 10.1.2018

[26] OM5577 Raspberry Pi Linux Demo Image (REV 1.2), Retrieved from https://nxp1.sharepoint.com/teams/12_33/NFCshare/OM5577/_layouts/15/guestaccess.aspx?docid=0a8e22f402431479a80d352b6a759c1ad&authkey=ARwSPawqR1zk3QSFVqFOR7A, last access: 22.6.2017

[27] CCID free software driver, Retrieved from https://pcsclite.alioth.debian.org/ccid.html, last access: 16.12.2017

[28] OSPT Alliance: CIPURSE™ V2, Operation and Interface Specification, Revision 2.0 / 2013-12-20 incl. Errata and Precision List, Revision 1.0 / 2014-09-18

[29] OSPT Alliance: CIPURSE™ V2, CIPURSETM T Profile Specification, Revision 2.0 / 2013-12-20 incl. Errata and Precision List, Revision 1.0 / 2014-09-18

[30] OSPT Alliance: CIPURSE™ V2, CIPURSETM SAM Specification, Revision 1.0 / 2013-10-14 incl. Errata and Precision List, Revision 1.0 / 2015-02-06

[31] OSPT Alliance: CIPURSE™ V2, CIPURSETM Cryptographic Protocol, Revision 1.0 / 2012-09-28 incl. Errata and Precision List, Revision 1.0 / 2014-09-18

5.2. List of Online Figure References

[F1] Introduction to RFID, Retrieved from http://telepathhero.blogspot.de/2013/07/rfid-inductive-backscatter-coupling.html, last access: 10.1.2018

[F2] Raspberry Pi, Retrieved from https://en.wikipedia.org/wiki/Raspberry_Pi, last access: 16.12.2018 and https://ocw.cs.pub.ro/courses/iot/labs/01, last access: 16.12.2017

[F3] Checking out NFC with the Raspberry Pi 2, Retrieved from http://www.electronicdesign.com/dev-tools/checking-out-nfc-raspberry-pi-2, last access: 10.1.2018

[F4] NFC logos, Retrieved from http://flexpoint.me/images/nfc-card-2.png, last access: 18.1.2018 and https://cdn.patrickvankleef.com/contentassets/ce17aca44b494658842392f9682a722c/nfc-logo-wide.png, last access: 18.1.2018

5.3. **List of Figures**

5.4. **List Of Tables**

6. Abbreviations

A

ACG	Access Group
ADF	Application Dedicated File
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
ART	Access Right Table
ASK	Amplitude-Shift Keying
ATR	Answer To Reset

C

CBP	CIPURSE™ Based Product
CCID	Chip Card Interface Device
CRC	Cyclic Redundancy Check
CTM	Consistent Transaction Mechanism

D

DF	Dedicated File

E

EF	Elementary File
ENC’ed	Encrypted and Integrity Protected Secure Messaging

F

FID	File Identifier

G

GUI	Graphical User Interface
GPIO	General Purpose Input / Output

I	
IDE	Integrated Development Environment
I/O	Input / Output
K	
KGM	Grand master key
KM	Master key
KD	Derived key
M	
MAC	Message Authentication Code
MAC’ed	Integrity Protected Secure Messaging with MAC field in SM-APDU
MF	Master File
N	
NFC	Near Field Communication
O	
OSPT	Open Standard for Public Transport
OS	Operating System
P	
PICC	Proximity Integrated Circuit Card
PCD	Proximity Coupling Device
PC/SC	Personal Computer / Smart Card
S	
SAM	Secure Access Module
SD	Secure Digital
SE	Secure Element
SFID	Short File IDentifier
SM	Secure Messaging

SM-APDU	APDU format resulting from CIPURSE™ Secure Messaging
SMG	Secure Messaging Groups
SMR	Secure Messaging Rules
SoC	System-on-a-Chip