eSIM Re-Selling on Mobile App

Albin Fridh
dic15afr@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisors: Stefan Höst (EIT) and Tibor Rathonyi (CellRebel)

Examiner: Maria Kihl

May 29, 2020

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

In recent years many devices such as smartphones, smart watches and laptops have started being equipped with embedded SIM cards called eSIM, which have the possibility to be reprogrammed with new subscription data through a process called remote SIM provisioning. To enable the eSIM in these devices, the mobile operators have had to implement new systems for the remote SIM provisioning process, and most of the research around this process has focused on security, leaving other areas unexplored.

Therefore, in this master's thesis the possibility for a 3rd party to be able to use eSIM and to take the role as re-seller of eSIM subscriptions for consumer devices on a mobile app has been explored by studying the existing eSIM and remote SIM provisioning specifications along with the specifications of the most commonly used mobile operating systems, Android and iOS.

The findings were that there is nothing in the eSIM standard that prevents 3rd parties from re-selling eSIM subscriptions for the mobile operators, but that there are some obstacles in both the Android and iOS operating systems as they need carrier privileges or eSIM access entitlements respectively to be able to access the APIs that handles the eSIM in the device.

Based on this research two different solutions, depending on whether the app has the possibility to gain the carrier privileges or eSIM access entitlements or not, were proposed. From these, an architecture for re-selling eSIM subscriptions was developed together with a proof of concept on the client part of the proposed solution in the form of an app for Android devices.

Keywords **-** eSIM, embedded SIM, eUICC, RSP, Remote SIM Provisioning.

Popular Science Summary

How a 3rd Party Company Can Use eSIM for Consumer Devices

The SIM card, a thing that most people use every day, perhaps without even thinking about it, is slowly starting to disappear. The introduction of the embedded SIM (eSIM), a SIM card that is built directly into a device (such as a smartphone, tablet, laptop, etc.), is making the old SIM card obsolete. This new technology implies new challenges for the mobile operators as they have to readjust to a new system, but it might at the same time be an opportunity for new players to take a place in the mobile communications ecosystem.

Imagine if it were possible to with just a few clicks be able to compare and instantly switch between different mobile carriers and subscription plans. With eSIM this is becoming a reality! In contrast to regular SIM cards where a change of carrier can take some time as you have to get a new SIM card to physically change it in your device, a change of carrier with the use of eSIM can be almost instant. The eSIM profile, which is the collective name for all the data and applications that would normally have been put on the physical SIM card, is simply downloaded and installed on the device over the network through a process called remote SIM provisioning. In short, this process involves four entities; the mobile operator, a server controlled by the operator where the profiles are created and downloaded from (called an SM-SP+ server), a device with an eSIM, and the end user. When the operator has created a profile in the SM-DP+ server, it will be uniquely identified by a so-called activation code. The end user can then enter this activation code into their device to download and install the profile. In this process there no longer exists a physical component as it does with SIM cards, which raises the question, will it still be possible for a 3rd party to re-sell eSIM subscriptions for the carriers like it is possible to re-sell physical SIM cards today?

The master's thesis *eSIM Re-Selling on Mobile App* has looked into this question and the short answer is yes, it is indeed possible, but there are a few obstacles. As the main use of eSIM for consumer devices is for mobile phones, the focus has been on how a re-selling solution could work together with a mobile app. What was found is that there is nothing in the specifications of eSIM or the download process that prevents a

3rd party from being a re-seller. However, the challenges lie in the two most commonly used mobile operating systems, Android and iOS. In both Android and iOS, the download and installation of an eSIM profile is performed through a set of functions where the profile to download is specified by its activation code. The problem for a 3rd party is that these functions are protected in a way that only an app made by the carrier owning the eSIM profile could use them. In iOS, the only way of being able to use these functions is by getting approval from Apple. In Android however, it is possible to gain this access solely from cooperation with the carrier in question. However, regardless if these functions can be used by the app or not, it is always possible to let the end user manually start the download from the eSIM settings on the device.

Based on these findings an architecture for a 3rd party eSIM re-selling solution was developed. In the proposed solution there is a back-end server where the activation codes for all the different eSIM plans from the partaking carriers are stored. The app can then communicate with the back-end to let the user buy and download one of the activation codes. When running low, the back-end server could request new activation codes from the carriers. This process is quite like the one existing in a physical store that sells prepaid SIM cards. The back-end server is like the store, the carriers supply this store with SIM cards (activation codes) and the customer (the user of the app) can browse the store and buy one of the SIM cards (download an activation code). Once the activation code is downloaded, the app could also download and install the profile if it is able to access the functions previously discussed. If it cannot, then the user can be instructed on how to manually start the download through the eSIM settings.

Acknowledgements

I would like to give thanks to CellRebel where the project was carried out, and my supervisor Tibor Rathonyi for giving me this opportunity and for his commitment and guidance during the project.

I would also like to thank Maria Kihl from the department of Electrical and Information Technology at Lund University for all her help and invaluable feedback and comments on both the project and the academic process.

Lastly, and above all, I would like to thank my wife Ellinor for all her love and support and for keeping me going these past few months. Without you, this thesis would not have been possible.

Thank you.
Albin Fridh

Table of Contents

List of Figures and Tables

Figures:

Tables:

Abbreviations

ARF	Access Rule File
EID	eUICC ID
eSIM	Embedded SIM
eUICC	Embedded Universal Integrated Circuit Card
EUM	eUICC Manufacturer
ICCID	Integrated Circuit Card Identifier
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ISD-P	Issuer Security Domain - Profile
ISD-R	Issuer Security Domain - Root
LDS	Local Discovery Service
LPA	Local Profile Assistant
LPD	Local Profile Download
LUI	Local User Interface
MNO	Mobile Network Operator
MVNO	Mobile Virtual Network Operator
OTA	Over-The-Air
QR	Quick Response
RSP	Remote SIM Provisioning
SIM	Subscriber Identity Module
SM-DP	Subscription Manager - Data Preparation
SM-DS	Subscription Manager - Discovery Service
SM-SR	Subscription Manager - Secure Routing
UICC	Universal Integrated Circuit Card

Chapter 1
Introduction

Over the last few years, the telecom industry has begun a change from regular SIM cards to eSIM (embedded SIM), which is a form of embedded, reprogrammable SIM card, and are right now in the middle of this transition. Many new wearables and mobile phones on the market has support for eSIM but to be able to use it, the mobile operators must also support eSIM. For the customers this means that because of eSIM it will be much easier to change operator, the eSIM can simply be reprogrammed over the network and the change will be almost instant.

1.1 Motivation

Recent market research in this area predicts that the number of eSIM enabled smartphones sold in the world will increase from 81 million in 2019 to 1 billion in 2024 [1]. Mobile Network Operators (MNOs) and Mobile Virtual Network Operators (MVNOs) are expected to build up automatic web-based services for selling and handling eSIM subscriptions. These service interfaces will likely look different between MNOs which may make it harder for the consumer to easily switch subscriptions. It is believed that many MNOs and MVNOs still want to continue re-selling subscriptions through 3rd party companies in a similar way to what exists today for prepaid SIM cards. Thus, there will be a need for 3rd party companies also for eSIM solutions that offer end users easy handling of eSIMs directly in their smartphone through a mobile app.

1.2 Project Aims and Challenges

The aim of this master's thesis is to explore how a 3rd party company can use eSIM and whether it is possible for a 3rd party company to take on the role as a re-seller of eSIM subscriptions for consumer devices on a mobile app. It will also look into if it can be made easier for the consumer to buy and manage the eSIM through the mobile app than it is directly from the different mobile operators.

To find answers to these questions, the aim is to create an architecture for a solution for re-selling eSIM subscriptions on a mobile app, as well as implement a proof of concept on parts of, or on the whole re-selling app if possible. This architecture will include:

- Definition of the whole end-to-end system including the re-selling company's assets and interfaces to mobile operators that enable the eSIM.
- Definition of limitations and difficulties to implement such a solution.
- Creation of process workflows between end-users, the re-selling company's systems and the mobile operators' systems.
- Detailed customer journeys to order eSIMs in the re-selling company's mobile app, which should describes all the activities needed from the customer to acquire and use an eSIM on the company's mobile app, as well as describe the system that are used to enable this customer journey.
- A mapping of how some existing solutions on how to buy eSIM works, for example from MNOs and MVNOs that are offering eSIM subscriptions.

The main challenge with the work is to define the feasibility, implementation strategy and architecture of the solution as it requires knowledge buildup and competence in various areas within telecommunication systems, mobile device operating systems and mobile app software development. Competence buildup in the area of current provisioning solutions of mobile services is also needed in the first phase of the project.

1.3 Approach

A large part of the thesis project will consist of gathering and compiling information from various technical areas around the eSIM ecosystem. In addition, making smaller prototype programs/scripts/simulations to test different aspects might be done. The gained knowledge will then be applied to design the different components needed for the eSIM re-selling architecture solution.

The work will contain both theoretical studies as well as interactions with other players in the industry in order to define the solution architecture. Below is a summarised suggestion on the approach:

Interviews with Operators:

- To understand preferred solutions from the operator's point of view.
- To understand integration points including options seen from the operators.
- Identify key values operators see with a 3rd party re-seller company.

Study eSIM specifications:

- Standardisation documentation exists primarily from GSMA but also from 3GPP. Documentation for eSIM and remote SIM provisioning relevant to the project is to be studied.

Interaction with a mobile app:

- Understanding of the existing APIs and libraries in iOS and Android in order to be able to define the client part of the solution.

When the architecture is done, a proof of concept based on the architecture will be created on parts of, or on the whole of the re-selling app to be able verify that the solution is feasible. The proof of concept will also be used to explore if the proposed solution makes it easier for the consumer to buy and manage eSIM or not by comparing it with some already existing solutions for selling eSIM.

1.4 Thesis Overview

An overview of the content in the chapters in this master's thesis is given below:

- **Chapter 2 Related Work:** Introduces the eSIM concept and brings up previous work related to eSIM and remote SIM provisioning.
- **Chapter 3 eSIM Ecosystem for Consumer Devices:** Explains the different parts of the eUICC (Embedded Universal Integrated Circuit Card), what they are for and how they work. Also gives a detailed explanation of the eSIM profiles and how remote SIM provisioning works. Finishes off by looking into which mobile phones and mobile operators that at the time of writing this thesis support eSIM and analyses some existing solutions for selling eSIM.
- **Chapter 4 Android and eSIM:** Gives an introduction to how the mobile operating system Android works and how eSIM is handled in Android devices.
- **Chapter 5 iOS and eSIM:** Gives a basic introduction to how Apple's mobile operating system iOS works and how eSIM is handled in iOS devices.
- **Chapter 6 Proposed eSIM Re-Selling Solution:** Discusses the limitations and challenges with a 3rd party eSIM re-seller solution, presents a couple of possible solutions to the problem and finally describes the solution proposed in this thesis for an architecture for re-selling eSIM subscriptions in a 3rd party mobile app.
- **Chapter 7 Proof of Concept:** Deals with the implementation of a proof of concept for the 3rd party eSIM re-selling solution proposed in chapter 6.
- **Chapter 8 Evaluation and Discussion:** Discusses and evaluates the proposed 3rd party eSIM re-selling architecture and the proof of concept that is based on it.
- **Chapter 9 Conclusions:** Presents the conclusions of this master's thesis and proposes areas for future work.

Chapter 2
Related Work

This chapter starts off by bringing up the concept of eSIM and the difference between regular SIM cards and eSIMs. It then continues to look into what has previously been researched and done within the area of eSIM and remote SIM provisioning.

2.1 From SIM to eSIM

The SIM card has changed considerably over the years, both physically from the 1FF (Form Factor) card to the 4FF, or more commonly known as the nano SIM card, and in its capability, such as connectivity speed, storage capacity and security. The next step for the SIM card is the eSIM, a SIM card embedded in the device. This evolution has been studied for example by Vahidian in [2] and by Koshy and Rao in [3]. This section gives an introduction on the eSIM concept and brings up how this evolution has led to differences between normal SIM cards and eSIMs.

2.1.1 SIM

The SIM (Subscriber Identity Module) or the SIM card as it is most commonly known as is a removable smart card, called UICC (Universal Integrated Circuit Card), used to authenticate the user to a mobile network in order to gain access to it. It is made up of an integrated circuit and is where the IMSI (International Mobile Subscriber Identity), which is the number that is used to identify the card to the network, and the authentication key is securely stored. The UICC contains the SIM and USIM (Universal Subscriber Identity Module) applications that are used for authentication in GSM (Global System for Mobile Communications) and UMTS (Universal Mobile Telecommunications System) networks respectively. The UICC is also tamper resistant and has security features built into it to help prevent unauthorised access to the IMSI and authentication key so that they cannot be copied or modified [2].

The SIM card has the benefit of being easy to use and can easily be swapped between devices, but there are also some drawbacks. It takes up quite a lot of space, especially if is to be used in for example smart watches or IoT devices of limited size. It can also be a problem for the manufacturers of devices that use

SIM cards as it puts constraints on things like design and waterproofing of the device. One of the biggest drawbacks though with regular SIM cards is the actual need for the physical card at all as this implies that it has to be acquired somehow, which may take time and makes it more difficult to change network operator.

2.1.2 eSIM

eSIM (embedded SIM), also called eUICC (embedded UICC) is a UICC that is soldered directly into a device during the manufacturing process. This allows the separation of the operators' profile and the physical chipset. The device can be manufactured with or without an initial profile and new profiles can be downloaded in a later stage from an SM-DP server through a process called remote SIM provisioning (RSP), where the profiles contain the same data that normally would have been put in a regular SIM card [4]. There are four alternative ways for operators to handle the provisioning and activation process as explained in [4], pre-printed eSIM vouchers, on-demand vouchers, entitlement based provisioning and pre-provisioned devices, where the first two are the most common and explained below:

- Pre-printed eSIM vouchers: This way is the one most like with normal SIM cards. The operator gets the eSIM profiles pre-provisioned in the SM-DP+ server and prints the details for each profile, for example in the form of a QR (Quick Response) code, on a plastic or paper card which then can be sold and distributed.
- On-demand vouchers: This is a more advanced option where the profile is created on the fly when sold, which can be either online or in a store. This requires that the operator must be able to communicate with the SM-DP+ at the time of the sale to create a new profile, retrieve the profile details and send them to the consumer, for example in an email. The main advantage with this solution is that it eliminates the logistics and paperwork required with regular SIM cards and pre-printed eSIM vouchers.

The usage of eSIM can then be divided into three main use cases as explained by Gerpott and May in [5]. They are machine-to-person (M2P) communication, machine-to-machine (M2M) communication and a mixture of both M2M and M2P. The M2P category contains the devices that involve user interactions from a human with the eSIM device to enable it to communicate. Consumer devices such as smartphones, tablets, laptops and wearables are all part of this category. In contrast the M2M category contains the devices which automatically can establish a data connection without the need for human interaction to do so. Things like IoT devices fall under this category. The last category, the mixture of both M2M and M2P, includes the devices whose primary purpose is to be used for M2M communication, but which also allows human interaction to for example gain access to telecommunication services. An example of this is connected cars, which will in the event of an emergency automatically connect to a back-end system, but which also lets the passengers of the car access the telecommunication services [5].

The advantages of eSIM are many. It consumes less energy than a regular SIM card, uses less space in the device and can be made to work in harsher conditions,

such as high pressure, humidity and high temperatures [5]. With eSIM it is also possible to download and install several profiles from different operators on the same device, though at most one profile can be enabled at any time [3].

It also makes it easier for travelers to buy a local subscription in the form of an eSIM when roaming, according to Hristova and Bryan in [6]. According to them, the opportunity to easily switch to a local network when traveling and being able to use it as a domestic user will lead to the extinction of roaming in just a few years.

2.1.3 Differences

Since the eSIM and the SIM card has the same purpose and performs the same tasks, the main difference between SIM and eSIM is how they are personalised and provisioned. For a regular SIM card, the user sets up a contract with an operator of their choice and in return they get a SIM card that they can insert into their device, enabling it to connect to the network. With eSIM however the user still sets up a contract with an operator of their choice, but instead of getting a SIM card the user receives an activation code, often in the form of a QR code. Scanning the QR code with the device lets the device start the remote download of a profile. When the profile is downloaded and installed, the device is able to connect to the network.

Another difference is that the UICC is owned and issued by a specific operator while the eUICC on the other hand is not owned by a specific operator but by the owner of the device. Even though the eUICC is owned by the user, the profile that is downloaded to it with the subscription credentials is still owned by the operator who only licenses its use to the end user [7].

A potential problem with eSIM from the consumers perspective may be that with eSIM you no longer get the physical component and instead must use a tool to handle and administer the eSIM in the device. As this will be new to many, Gerpott and May in [5] writes that it is important that the development of the tool used by the consumer to manage the eSIM profile strive to be as easy to use as possible.

2.2 eSIM in M2M Communication

Even though this thesis will mostly revolve around the use of eSIM in consumer devices, it is also important to bring up the use of eSIM in M2M applications, as this was the original reason that the eSIM and remote SIM provisioning was created. As the number of connected devices rises and the need for smaller and more energy efficient devices grows, for example for IoT use cases, the need for an easy way to manage them also grows. The traditional way of embedding normal SIM cards in these types of devices have several shortcomings, as explained in [8]. The manufacturing of the SIM card and the need to install the subscription data on it through a physical channel increases cost and lead times, normal SIM cards can only support a single carrier and the whole SIM card has to be changed to change a subscription, which can be a difficult and costly operation as the device may be in a remote or inaccessible location. To address all of these different shortcomings,

the remotely reprogrammable eSIM and the remote SIM provisioning system was developed [8]. The adaption of these re-programmable eSIMs in M2M communication is according to Vesselkov, Hammainen and Ikalainen in [9] an inevitable industry evolution.

The remote SIM provisioning system for M2M devices contains the following main entities, as explained in [8]:

- **SM-DP:** The SM-DP (Subscription Manager - Data Preparation) is responsible for creating new profiles containing the subscription data and transferring them to the eUICC.
- **SM-SR:** The SM-SR (Subscription Manager - Secure Routing) is mainly responsible for the communication with the eUICC and setting up secure channels to the eUICC.
- **ISD-P:** The ISD-P (Issuer Security Domain - Profile) acts as a secure container on the eUICC for the hosting of a unique profile and is created before the profile can be installed.
- **ISD-R:** The ISD-R (Issuer Security Domain - Root) in turn is responsible for creating the ISD-Ps.

The ISD-P ans ISD-R are further explained in section 3.1.2.

An overview of the main steps performed in the provisioning process are as follows [8]:

1. The MNO sends a *DownloadProfile* request to the SM-DP specifying the profile type and the EID (eUICC ID) of the device it should be installed on.
2. The SM-DP creates the profile as described by the MNO. It then sends a request to the SM-SR for a new ISD-P to be created on the eUICC.
3. The SM-SR instructs the ISD-R on the eUICC to create a new ISD-P.
4. The SM-DP and the newly created ISD-P then derives a secret key from a shared secret obtained using ECKA (Elliptic Curve Key Agreement).
5. The profile is then securely sent from the SM-DP to the eUICC using the key created in the previous step.
6. The profile is decrypted and installed on the eUICC.

The main difference between remote SIM provisioning for M2M devices and for consumer devices is that in M2M the download is initiated by the operator and the profile is pushed to the device, while in remote provisioning for consumer devices it is the user of the device that initiates the download and the device then retrieves the profile [10]. Remote SIM provisioning and eSIM for consumer devices will be covered more thoroughly in the next chapter.

2.3 Security

Much focus on the research concerning eSIM has been on the security of eSIM and how to make it at least as secure as when using a regular SIM card. The main problem with eSIM is that it is provisioned OTA (Over-The-Air) and with that comes security risks as it is sensitive data that is being transmitted.

2.3.1 Secure Provisioning

Park, Baek and Kang in their paper *Secure Profile Provisioning Architecture for Embedded UICC* [11] brings up the problem with how to make the process of provisioning eSIM profiles secure, and also proposes an architecture for how secure profile provisioning could work. They set up a number of security requirements a remote profile provisioning architecture should fulfill to reach the level of security needed, as well as considerations that should be taken into account regarding scalability, efficiency and flexibility. For example, since the eUICC is a resource-constraint medium, processing overhead should be kept to a minimum for all security-related operations in the eUICC. The security measures that needs to be taken into account according to their research are:

- **Mutual Authentication:** The eUICC and the SM-DP+ should first prove their authenticity to each other before any data is exchanged.
- **Confidentiality:** The data and profiles sent between the SM-DP+ and the eUICC should only be available to the authenticated entities.
- **Data Integrity:** Data sent between the SM-DP+ and the eUICC should not have been altered by any unauthorised entity.
- **Data Origin Authentication:** The SM-DP+ and the eUICC should be able to ensure that the received data actually came from the other authorised entity.
- **Non-Exposure of Key:** Any credentials between the SM-DP+ and the eUICC should not be exposed to any entity not directly involved in the profile provisioning process.
- **Verification of eUICC's Information:** Information sent from the eUICC should be possible to verify by the receiving entity.

2.3.2 Potential Vulnerabilities

The potential vulnerabilities concerning the eUICC and the remote SIM provisioning process has been analysed by Chitroub et al. in [12] and by Meyer, Quaglia and Smyth for M2M remote provisioning in [13].

Chitroub et al. found that during the provisioning process it might be possible for an attacker with the right equipment that are sniffing on the communication to gain knowledge of the SHS (Shared secret) used in the key agreement through a brute-force attack. With the SHS, the attacker will be able to derive the secret key used to encrypt the profile and gain access to the sensitive information in it. They

also found that due to the use of the LPA (Local Profile Assistant, which will be covered in the next chapter) in the device, which they found to be unsecure, when downloading a profile it might be possible for an attacker to install an application on the downloading device that intercept the communication between the LPA and the eUICC. Their proposed solution against this is an end-to-end encryption scheme between the eUICC and the SM-DP+ with enhanced protection against these attacks [12].

In the paper by Meyer, Quaglia and Smyth security aspects of the M2M have been analysed and they found several possible attacks on the system:

Memory Exhaustion Attack

This is an attack that can fill the eUICC's memory with empty ISD-Ps by an attacker that have the possibility to drop messages that are sent from the eUICC. In the process of downloading a new profile the ISD-R is instructed to create a new ISD-P where the profile can be stored. If the attacker then drops the response messages from the ISD-R after creating a new ISD-P. The ISD-P will remain on the card but not be associated with any MNO or SM-DP meaning that it cannot be deleted since only MNOs or SM-DPs are allowed to delete it. Repeating the attack will fill the eUICC's memory and then no new profiles can be installed.

The proposed solution to prevent this is to implement a mechanism in the eUICC to handle the ISD-P creation. When an ISD-P has been created this mechanism will wait for the next step in the process, i.e. the download of a profile. If the next step is not performed within a certain amount of time, the ISD-P will automatically be deleted.

Undersizing Memory Attack

In this attack it is possible for an SM-SR that is responsible for an eUICC to alter the field *remainingMemory* in a message sent to the SM-DP to 0. This is possible because that specific field is not signed in the message. This will lead to operators and SM-DPs not being able to install any new profiles on the eUICC.

The attack can be prevented by protecting the value in the *remainingMemory* field by letting the eUICC sign it using its private key. This way the SM-SR will not be able to modify this filed without the SM-DP noticing.

Inflated Profile Attack

This attack is performed by a malicious operator who can create a profile so big that it exhausts all the remaining memory on the eUICC. This will lead to other operators not being able to install new profiles on the eUICC. The attack can be prevented by defining an upper limit on the size of profiles.

Locking Profile Attacks

This is an attack that uses the fact that an MNO can create profile policy rules for a profile indicating whether it can be disabled, deleted, and so on. If an eUICC contains profiles with the rule *CannotBeDisabled* set to false, a malicious MNO can

install a profile with the rule *CannotBeDisabled* set to true. This will disable the previously enabled profile, and the operator of that profile will no longer be able to re-enable it. Several countermeasures were proposed, for example specifying an upper limit on the time an operator can lock the eUICC to prevent this kind of abuse.

2.4 Carriers Views on eSIM

In a survey made by Ernst & Young in [14] 11 operators answered questions regarding eSIM to get an understanding of their views on this upcoming technology and how it might impact them. Also, in the report by Hristova and Bryan in [6] MNOs have been asked about their opinions on eSIM, in this case 107 MNOs from 92 different countries.

The survey by Ernst & Young found that operators are most willing to adapt to eSIM for M2M communication and sees business opportunities in connecting to the internet of things, though might not do so if they perceive that eSIM poses a risk to their core market. Their views on using eSIM in consumer devices vary where some of the more conservative operators fear that it will make it easier for rival operators to poach their customers, but some also sees advantages of moving to eSIM. Despite this, all operators questioned in the study opposed the use of eSIM in smartphones. The main risks they saw with moving to eSIM was the possibility to lose their direct relationship with their customers, the real-time competition with other operators, security issues that comes from how the eSIM is being programmed and operational consequences that arise from implementing this new technology [14]. But as Gerpott and May write in their report in [5], it cannot be taken for granted that the introduction of eSIM will automatically lead to increased competitiveness between operators. They argue that MNOs and MVNOs with strong marketing power will still be able to limit the customer choices.

In the report by Hristova and Bryan in [6] however, they found a more positive attitude towards eSIM for consumer devices with 81% of MNOs seeing it as an opportunity. They found that users demand a combination of both network quality and reasonable pricing and that most users will not sacrifice quality for lower pricing.

Chapter 3

eSIM Ecosystem for Consumer Devices

This chapter deals with the different parts of the eSIM ecosystem. It also bring up concepts and theory relevant for this master's thesis, such as how remote SIM provisioning for consumer devices works and gives an overview of the eUICC architecture. This is specified by the GSMA in the SGP.21 and SGP.22 documents and is based on the previous work done in the field and is constantly being updated to deal with for example any security problems or threats. This chapter will then finish off by looking into which devices and operators that are currently supporting eSIM and analyse some existing solutions for selling eSIM.

3.1 eUICC Architecture

This section will describe the different parts of the eUICC as seen in figure 3.1 and what they are used for, as described by the GSMA in [15] and [16].

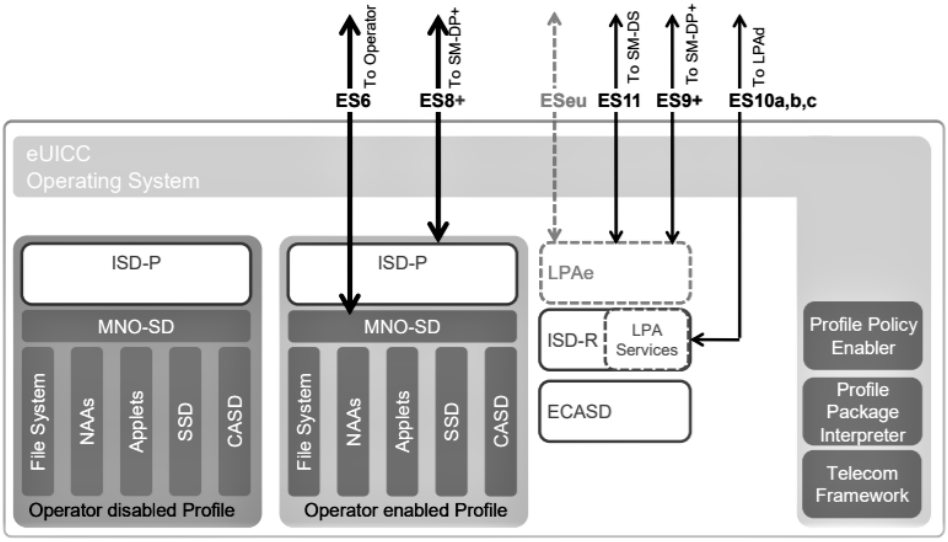

Figure 3.1: Overview of an eUICC containing two profiles [15].

3.1.1 ECASD

The ECASD (Embedded UICC Controlling Authority Security Domain) handles the secure storage of sensitive credentials. It contains the eUICC's private keys for creating signatures and its certificate with the eUICC's public key for authentication. It also contains the certificates and keyset belonging to the EUM (eUICC Manufacturer) which can be used for renewal of keys and certificates on the eUICC, as well as the GSMA CI (Certificate Issuer) root public keys that are used for authentication of off-card entities, such as the SM-DS and SM-DP+. There exists only one ECASD on an eUICC and it is installed by the EUM during the eUICC manufacturing process.

3.1.2 ISD-P and ISD-R

The ISD-P (Issuer Security Domain - Profile) in the eUICC acts as a secure container for the hosting of a unique profile. There may exist more than one ISD-P in the eUICC since it is possible to download and install several profiles. It is the on-card representative of the SM-DP+ (the SM-DP+ is explained in section 3.2.2) and is also used for downloading and installing the bound profile package together with the profile package interpreter, which is used to translate and decode the bound profile package into an installed profile on the eUICC. The bound profile package will be further explained in section 3.2.1.

The ISD-R (Issuer Security Domain - Root) in turn is responsible for the creation of the ISD-P on request from the SM-DP+ and the life cycle management of all the ISD-Ps it has created.

3.1.3 The Profile and Its Content

As seen in figure 3.1 the profile consist of many components, such as NAAs (Network Access Applications), SSD (Supplementary Security Domains) and CASD (Controlling Authority Security Domain), along with the file system and other applets and applications. It also contains the MNO-SD (Mobile Network Operator - Security Domain) which is the representative for the operator on the eUICC. It contains the operators OTA keys which are used to provide a secure OTA connection over which the operator can manage the different operator services in the profile, such as updating profile policy rules.

3.1.4 LPA

The LPA (Local Profile Assistant) may exist in the device (LPAd) or in the eUICC (LPAe), but the LPA in the eUICC is optional. When there exists an LPA both in the device and in the eUICC, the one that is to be used is optional and is configured in the device settings. A more thorough explanation of the LPA and its different parts is given in section 3.2.3.

3.1.5 LPA Services

The LPA services function is to provide the LPA with access to the data and services that the LAP requires to perform its function. This includes things like providing the EID (eUICC ID), root SM-DS address and default SM-DP+ address if they are configured, to the LPA. It is also responsible for transferring the bound profile package from the LPA to the ISD-P.

3.1.6 Telecom Framework

The purpose of the telecom framework is to provide the network access applications in the ISD-Ps with standardised algorithms for network authentication and the ability to configure these algorithms with different parameters from the profile.

3.1.7 Profile Policy Enabler

The profile policy enabler is a service in the eUICC operating system which is responsible for the verification, interpretation and the enforcement of the profile policy rules set in the profiles. The profile policy rules are a way for the operator to enforce certain conditions on the service provided to the user. The rules set in a profile only apply to that specific profile in which it is defined. Examples on profile policy rules are "disabling of this profile is not allowed" and "deletion of this profile is required upon its successful disabling".

3.2 Remote SIM Provisioning Architecture

RSP (Remote SIM provisioning) is the process that enables the use of the eSIM in a device. This section will bring up how remote SIM provisioning works as described in [7] and [16] and what purpose the different components in the system shown in figure 3.2 has, but first it will describe the profile and the different formats it takes on from the moment it is created to being installed on the eUICC.

3.2.1 Profile Package Types

The profile takes on three of four different formats before it is installed in the eUICC. It does this for protection and security both while it is stored in the SM-DP+ server and when it is sent to the device.

The different formats the profile can take are:

- Unprotected profile package.
- Protected profile package.
- Bound profile package.
- Segmented bound profile package.

Below follows a description of the different profile package formats along with an explanation of when and why it is needed [15].

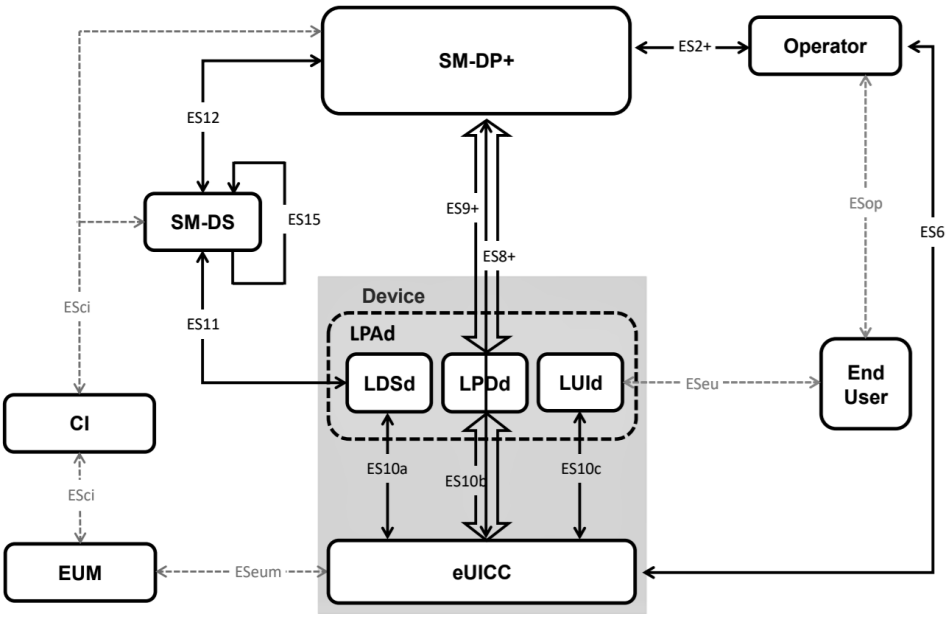

Figure 3.2: Overview of the RSP system. Figure adapted from [7].

Unprotected Profile Package

The unprotected profile package is what is created when the profile package generation function is called in the SM-DP+. As input the function takes profile specifications provided by the operator. The resulting profile is a set of TLV (tag-length-value) encoded profile elements, and as the name suggests this is a completely unprotected profile.

Protected Profile Package

The SM-DP+ takes the unprotected profile package and through the profile package protection function creates the protected profile package. It does this by encrypting segments of the encoded profile elements and adding a MAC (Message Authentication Code) at the end of it and the encrypting the resulting encrypted segment and MAC once again. The profile can now be securely stored on the SM-DP+ until it is to be downloaded to an eUICC.

The encryption can either be done using session keys agreed upon by the eUICC and the SM-DP+ or by using random keys for each profile. If random keys are used the protected profile package can be created by the SM-DP+ in advance without any communication or knowledge about the eUICC where the profile is to be installed. This way the operator is able to request the SM-DP+ to create multiple profiles in advance and in bulk.

Bound Profile Package

When a profile is to be downloaded to an eUICC, the SM-DP+ takes the protected profile package and through the profile package binding function creates the bound profile package. This cryptographically links the protected profile package to the target eUICC so that only that particular eUICC can install the profile. In this state the profile can safely be sent to the eUICC.

Segmented Bound Profile Package

If the LPA is in the device, then the LPD (which is explained in section 3.2.3) must generate the segmented bound profile package before further transferring the profile to the eUICC. This is done by dividing the bound profile package into smaller segments. If the LPA is in the eUICC this step is not necessary.

3.2.2 SM-DP+

The SM-DP+ (Subscription Manager - Data Preparation +) plays an important role in the RSP system. The "+" is added to the name because it performs the tasks of both the SM-DP and the SM-SR in the M2M remote provisioning process. It is the entity that is responsible for the creation of a profile on the request from the operator. Once the profile is created it is also responsible for the protection of the resulting profile and the secure delivery of the profile within a bound profile package to the eUICC.

As mentioned in section 3.1.2 the ISD-P is the on-card representative of SM-DP+, meaning of course that the SM-DP+ is the off-card representative of the ISD-P. It will be the one requesting the ISD-R create a new ISD-P and will be responsible for the management (such as enabling, disabling, deleting and updating) of the created ISD-P throughout its entire life cycle.

The SM-DP+ manages this through the use of six functions that are summarised below:

- **Profile package generation:** Used to generate a new profile package including for example the IMSI, authentication key K_i and ICCID (Integrated Circuit Card Identifier) upon agreement with the operator. This can either be done off-line or in real time.
- **Profile package protection:** Used to secure the created profile package, generating the protected profile package.
- **Profile package binding:** Used to bind the protected profile package to a specific eUICC generating the bound profile package.
- **Profile package storage:** Used to securely store the protected profile package or the bound profile package until delivery to the specified eUICC.
- **Profile package delivery:** Used to transmit the bound profile package to the target eUICC through the LPA for installation.
- **SM-DS event registration:** Used to notify the SM-DS that there is a pending operation for a specific eUICC.

3.2.3 LPA

The LPA (Local Profile Assistant) may as explained in section 3.1.4 exist both in the device and in the eUICC. In this section the focus is on the LPA in the device as shown in figure 3.2. The LPA in the device (LPAd) uses the LPA services in the eUICC to provide three functions, the LDS (Local Discovery Service), LPD (Local Profile Download) and the LUI (Local User Interface), which are described below.

- **LDS:** Used for the retrieval of a pending event record from the SM-DS.
- **LPD:** Used for the download of the bound profile package to the LPA from the SM-DP+ and for further transfer to the eUICC.
- **LUI:** The interface that lets the user of the device manage the profiles in the eUICC.

3.2.4 SM-DS

The purpose of the SM-DS (Subscription Manager - Discovery Service) is to let the LDS in the LPA of a device know that the SM-DP+ wants to communicate with it. The SM-DP+ does this by sending an event registration message for a specific device to the SM-DS. The device polls the SM-DS and if the SM-DS has a pending event it will respond with the SM-DP+ address, and if not it will respond with a null message.

The SM-DS is not compulsory when deploying an RSP solution for eSIM and the purpose and the implementation of it is optional and it is up to the network operator to decide if they want to use it or not [4].

3.2.5 Interfaces

The RSP system consists of several standardised interfaces for communication between the components in the system as shown in figure 3.2. The interfaces ESop (interface between the End User and the Operator) and ESeu (Interface between the End User and the LUI) are not standardised and may therefore look different between different operators and devices. Below is a short summary of some of the different interfaces:

- **ES2+:** The interface between the SM-DP+ and the operator. Used for example by the operator to order a new profile.
- **ES6:** Interface between eUICC and operator. Used by the operator to access the MNO-SD over a secure OTA channel to be able to manage the profile, such as updating profile policy rules and changing the SMS Center Address [4].
- **ES8+:** Interface between the eUICC and the SM-DP+, tunneled in ES9+ and ES10b. Used to provide a secure channel from end-to-end for example for the download of the profile.

- **ES9+:** Interface between the LPD and the SM-DP+. Used for secure delivery of the profile.
- **ES10a:** Interface between the LDS and LPA services in the eUICC. Used by the LDS to get the root SM-DS and default SM-DP+ addresses, if configured.
- **ES10b:** Interface between the LPD and LPA services. Used to transfer the received bound profile package to the eUICC.
- **ES10c:** Interface between the LUI and the LPA services. Used to enable the end user to perform local profile management.

3.3 Procedures

This section will describe some of the most important procedures in the remote SIM provisioning process, such as how profiles are downloaded and other related procedures for profile management. These procedures are described in the RSP Technical Specification [15] and RSP Architecture [16] by the GSMA.

3.3.1 Order eSIM

The process of ordering eSIM can be divided into four parts:

- Contract subscription,
- Download preparation,
- Contract finalisation,
- Subscription activation,

where the last part, the subscription activation, is optional and not always necessary.

The contract subscription works just as it does with regular SIM cards. The end user decides which operator to get the service from and can then either visit the operator's physical store, their web store or get in contact with the operator in any other way that the operator offers. How this is done does not matter, as long as the operator gets all the information they need to proceed to the next step in the process, such as personal details and billing information. The operator may also in this step optionally acquire information about the target device, such as EID and IMEI (International Mobile Equipment Identity). If they are acquired, the operator may at this point verify that the target device is supported, and if not this check will in any case always be performed again at a later stage before downloading the profile.

More often the case is that the end user is able to select what target device the profile is to be installed on by indicating the manufacturer and model of the device to the operator and that way the operator can select the appropriate profile for the device.

If the contract subscription succeeds, then the download preparation is started. This process is the same as with regular SIM cards but with the difference that the

data produced is not put into a physical SIM card but are kept by the SM-DP+ as a profile [2]. The operator calls the SM-DP+ with relevant input requesting it to prepare a new profile for download. The SM-DP+ now reserves an ICCID for the request either by picking one from its inventory or by being provided one from the operator. It also picks the protected profile package related to the selected ICCID from its inventory or creates a new protected profile package corresponding to the ICCID. The SM-DP+ responds with the selected ICCID to the operator, who now since the ICCID is known can perform other necessary back-end operations such as the provisioning of the HLR (Home Location Register). A so-called MatchingID is generated and associated with the ICCID for later identifying the profile in the download process. If the operator requires the end user to enter a confirmation code to be able to download the profile, this code should also be sent to the SM-DP+.

In the contract finalisation part, the operator provides the end user with all the necessary information needed for the download of the profile. This is done in the form of an activation code, which may be in the form of a QR code, containing the SM-DP+ address and the MatchingID. If the operator requires the end user to enter the optional confirmation code, this should also be provided but separate from the activation code.

The subscription activation process is optional and only used when the operator could not perform all the necessary back-end operations during the download preparation. If this is the case the operator must notify the SM-DP+ when this is done so that the profile can be downloaded. If the end user tries to download the profile before this is done an error will be returned.

3.3.2 Download and Install Profile

When all the steps in the previous section have been performed the profile is ready to be downloaded and installed. This is initiated by the end user by providing the activation code to the LPA through the LUI either by manually typing the activation code or scanning it through a QR code. The LPA parses the activation code and retrieves the SM-DP+ address and the activation code token, which is the same as the MatchingID in the SM-DP+. If a confirmation code is required the user is requested to input the confirmation code provided by the operator. The LPA will now initiate the common mutual authentication procedure, which authenticates the eUICC to the SM-DP+ as well as the other way around. The common mutual authentication procedure will not be further discussed here, but it ensures that both involved parties are properly authenticated.

When authentication is done, the SM-DP+ finds the pending profile download order based on the MatchingID provided by the LPA and if the profile is already linked to an EID, verifies that it is the EID of the authenticated eUICC and checks that the profile is in such a state that it can be downloaded. It also performs an eligibility check making sure that the profile is compatible with the target eUICC based on the information sent by the device and that the eUICC is able to install one more profile. If this succeeds, the SM-DP+ finally checks if the confirmation code is correct if this is used. If used and correct, the download can begin.

The SM-DP+ and the eUICC sets up a secure connection and establishes a

session key. The SM-DP+ prepares the bound profile package using the session key and sends it to LPA in the device, which in turn transfers it to the eUICC. If the eUICC can successfully install the profile, the eUICC proceeds by acknowledging this to the LPA and the SM-DP+. The LPA in turn notifies the end user about the installation and the SM-DP+ notifies the operator. The profile is now installed on the eUICC in a disabled state and must be enabled by the end user before use.

As a security measure, the SM-DP+ keeps count of the number of failed attempts to download the profile and the number of failed attempts to enter the correct confirmation code. If too many tries have been made, the SM-DP+ will abort the download and inform the operator.

3.3.3 Enable and Disable Profile

The end user is able to perform local profile management through the LUI in the LPA. Before any procedure starts, the LPA should verify the user's intent.

To enable a previously downloaded profile, the user selects the profile to be enabled in a list of all installed profiles. The LPA then sends a profile enable request to the ISD-R indicating which profile to enable. The ISD-R in turn checks the profile policy rules if the operation is allowed. If it is allowed, then the ISD-R will start by checking if any other profile is currently enabled and in that case disable it. The selected profile is then enabled, and the user is informed.

The procedure to then disable the enabled profile is almost the same as to enable it. The user selects to disable the currently enabled profile from a list over all the installed profiles. The LPA then sends a profile disable request to the ISD-R which checks the profile policy rules if the profile is allowed to be disabled. If it is, then the ISD-R disables the profile and notifies the user through the LPA, and if it is not, then the operation will be aborted.

3.3.4 Delete Profile

To delete a profile that is installed on the eUICC the user selects the profile from a list of all profiles on the device and verifies the intent to delete it. If the profile is enabled, then it will first be disabled before continuing. The LPA then send a request to the ISD-R to delete the profile. The ISD-R checks the profile policy rules if the profile can be deleted and if it is allowed, removes the profile and its ISD-P. Then both the user and SM-DP+ are notified that the profile has been deleted.

3.3.5 Transfer Profile

If the user for example switches mobile phone for some reason, he or she could with a regular SIM card just move the SIM card from the old device to the new. With eSIM this is not so easy. There currently exist no standardised way of transferring a profile between devices, but several suggestions on how it could be made possible have been made, for example in [4].

3.4 Devices Supporting eSIM

Since eSIM is still in its early stages, not that many devices have support for eSIM yet. From the beginning, it was mainly Apple's and Google's mobile phones that had eSIM, but more mobile phone manufacturers have since then followed and now also implement eSIM in their phones. Samsung has support in some of their smart watches and has only just started putting it in their phones. The Samsung Galaxy Fold is the only currently available Samsung with support for eSIM, but with the release of the Galaxy S20, S20+, S20 Ultra and the Z Flip, which all have eSIM, it appears that Samsung is going to give more of their phones support for eSIM in the future.

A big step for eSIM came with the Motorola Razr which is the first mobile phone to have only support for eSIM, meaning that there is no slot for a regular SIM card. All other mobile phones mentioned here still have the possibility to use a regular SIM card as well as eSIM.

A list of mobile phones that have eSIM is given in table 3.1 [17][18]. Note that the Google Pixel 2 and 2 XL has eSIM but only works with Google Fi, Google's own MVNO, and cannot be used with any other network operator. Also note that not all Google Pixel 3a, 3a XL, 3 and 3 XL have eSIM, for example if they are bought on the Japanese market they come without eSIM.

Mobile phone manufacturer	**Model**
Apple	iPhone 11, 11 Pro, 11 Pro Max iPhone XS, XS Max iPhone XR iPhone SE 2020
Google	Pixel 2, 2 XL (Google Fi only) Pixel 3a, 3a XL Pixel 3, 3 XL Pixel 4, 4 XL
Samsung	Galaxy S20, S20+, S20 Ultra Galaxy Fold Galaxy Z Flip
Motorola	Moto RAZR
NUU	Nuu Mobile X5
Huawei	P40, P40 Pro, P40 Pro+

Table 3.1: Mobile phones currently supporting eSIM.

3.5 Operators Supporting eSIM

To be able to use eSIM in a device the network operator must also offer this service for that device. Some operators for example only support eSIM for tablets and

wearables such as smart watches and sometimes only from a certain brand, for example Samsung or Apple.

Apple has put together a list of mobile operators in many different countries that support eSIM for their devices in [19]. This list is not comprehensive though, and mainly shows the MNOs supporting eSIM and does not list all the mobile virtual network operators that supports eSIM.

By looking into each of the Swedish MVNOs to see whether they support eSIM or not, table 3.2 was created listing all mobile network operators and mobile virtual network operators that are currently supporting eSIM for mobile phones in Sweden, and what type of mobile plans they are offering for eSIM, a monthly subscription or prepaid. In table 3.3 a list of some of the operators supporting eSIM in the US, UK and Germany is given. As can be seen, several of the big operators such as AT&T and T-Mobile support eSIM. T-Mobile has also launched an iOS app for buying and downloading prepaid eSIM for iPhones in the US [20].

MNO/MVNO	**Type of Plan**
Telia	Subscription
Halebop	Subscription
Telenor	Subscription
Tre	Subscription and Prepaid
Tele2	Subscription

Table 3.2: Mobile operators currently supporting eSIM in Sweden.

Country	**MNO**
United States	AT&T T-Mobile USA Truphone Ubigi Verizon GigSky
United Kingdom	EE O2 Truphone Ubigi
Germany	O2 Telekom Truphone Vodafone

Table 3.3: Mobile operators supporting eSIM in the US, UK and Germany.

3.6 Existing Solutions

This section looks into how the already existing solutions look like when buying eSIM, for example from a mobile operator. It will analyse how the consumer can find and order eSIMs, how the eSIM is delivered to the user and also how the consumer can keep track of how much is left on the eSIM in terms of amount of money, minutes left, amount of data remaining, etc. and how it can be refilled if it is a prepaid eSIM.

3.6.1 Buying eSIM From MNOs or MVNOs

eSIMs from MNOs and MVNOs can be found and bought either in one of their physical stores, online on their website, or as mentioned in section 3.5, from some operators like T-Mobile USA in their own app. Many operators also offer the possibility to replace a physical SIM card with an eSIM. The process of buying an eSIM is very much like buying a regular SIM card, the only real difference is how it is delivered. With a normal SIM card, the card has to either be sent to the consumer or retrieved at one of the operators stores. With eSIM however the delivery can be instantaneous for example in the form of an email with the profile credentials or if bought through the carrier's app, directly downloaded and installed. Then, when the eSIM profile is installed on the device, the eSIM acts and works just like a regular SIM card. So, things like tracking how much data has been used and so on, and the process of refilling a prepaid eSIM works just the same for eSIM as with a regular SIM card from the same carrier.

3.6.2 Buying eSIM From 3rd Party Re-Seller

Unlike regular prepaid SIM cards that are being sold by 3rd party re-sellers in places like grocery stores, gas stations, etc. eSIMs are, at least not yet, being sold this way. However, it is possible to find 3rd parties re-selling eSIM subscriptions both in web-based stores and in apps that are mainly marketed towards travelers. These solutions are of the kind where the consumer is presented with the SM-DP+ address and activation code token, or a QR code containing these, and has to manually install the profile through the eSIM settings on the device, meaning that the app is not able to directly download and install the eSIM profile on the device. This kind of solution and alternative solutions will be further discussed in chapter 6. Just as when buying directly from the MNO or MVNO, things like keeping track of amount of data used and refilling a prepaid eSIM works exactly as with a normal SIM card.

Chapter 4
Android and eSIM

Android is an open source mobile operating system developed primarily for smartphones and tablets. The core of the Android operating system is based on the Linux kernel and most Android applications are written in the Java programming language and run in the Android runtime (ART), but can also be written in languages like C++ and Kotlin [21]. This chapter deals with programming and security in Android as well as how it works together with eSIM.

4.1 SDK

The Android SDK (Software Development Kit) is a collection of tools used in the development of Android applications. It includes standard libraries, API documentation and tools used for debugging and compiling the application. The SDK compiles the code and all other data and files needed, such as the manifest file, into an APK (Android Package) file which is an archive file with the file extension *.apk*. This file can then be used to install the application on an Android device [21].

4.2 App Components

An Android application is built up by several components, each dedicated to handle a different part of the app. The four main components in an app are activities, services, broadcast receivers and content providers [21], which are described below.

4.2.1 Activities

An activity in an app is represented by a single screen with a user interface and is the part of the app that the user interacts with. An app may consist of one or several activities and each activity is an entry point to the app since an app might not always start with the same activity. For instance, one app can invoke an activity in another app which might not be the same activity as if the app was started on its own by the user. In general, an app has a main activity which is the first activity the user sees when the app is launched. This activity can then in turn start a new activity, creating a cohesive experience for the user, even though there typically only exist minimal dependencies between the activities [22].

Each implemented activity in the app must be a subclass of the Activity class by extending it and must be declared in the manifest [21].

To let the user transition between activities and leave and reenter the app without it crashing or losing the users progress, an activity contains a set of seven callback methods to manage the activity's life cycle [23]:

- **onCreate():** Called when the system creates the activity and is used to initialise the components needed, such as creating views.
- **onStart():** Called when the activity is becoming visible for the user. Performs tasks to prepare the activity to become visible and interactive.
- **onResume():** Called when the user starts to interact with the activity. It is here the main functionality of the activity is implemented.
- **onPause():** Called when the activity loses foreground state and the activity enters a paused state.
- **onStop():** Called when the activity is no longer visible for example because a new activity is starting
- **onRestart():** Called when the activity is restarted from a stopped state and is used to restore the state of the activity to the state it was when stopped.
- **onDestroy():** Called when the activity is destroyed by the system. Used to for example release all the resources held by the activity.

4.2.2 Services

Another important component in an app is a service. A service is a component that can perform long running tasks in the background, such as downloading data over the network, without blocking the user's interaction with an activity. A service does not provide a user interface to the user and can be started for example by an activity and continue to work even if the activity is stopped or destroyed [21].

4.2.3 Broadcast Receivers

A broadcast receiver is used by an app to receive announcements broadcasted by the system in the form of broadcast intents, even if the app is not currently running. It is a way for the apps and system to communicate with each other. It is also possible for apps to send broadcasts, and when a broadcasted intent is received by an application it can be used to start components in response to the content of the intent [21]. More about intents in section 4.3.

4.2.4 Content Providers

A content provider is a component that encapsulates a shared set of app data stored in the file system. Applications can query or modify the data through the content provider if the content provider allows it, for example if the querying application has the right permissions [24].

4.3 Intents

An intent is a simple messaging object used in communication between different components in the system, such as activities and services. Three of the components mentioned in the previous section are all initiated by intents and together constitute the three main use cases for intents [25]:

- **Launching an activity:** A new activity can be started by passing an intent specifying the activity to start and containing necessary data to the *startActivityForResult()* or the *startActivity()* method.
- **Starting a service:** A new service can be started, or an ongoing service can be given new instructions by passing an intent describing the service to start to the *startService()* method.
- **Broadcasting messages:** A broadcast can be initiated for example by passing an intent to the *sendBroadcast()* method.

There are two types of intents, implicit and explicit intents. Explicit intents are intents where the component called is explicitly defined, such as launching a specific activity in an app. Implicit intents however are intents where the component is not explicitly defined. Instead, the intent defines the action that is to be performed and any component able to perform this action may be invoked. This is done by comparing the intent with the intent filters that are declared in the manifest in the other apps on the device. If a match is found then the intent is delivered to that component [25].

4.4 App Manifest

Every Android app is required to have an AndroidManifest.xml file which describes all the essential parts of the app to for example the operating system. The manifest file contains information about all the different components in the app, such as activities and services, and intent filters used to describe how the components can be started. It also specifies the permissions the app needs in order to for example access parts of the system that is protected or to gain access to other apps [26].

4.5 Security in Android

Android has several built-in security features with the purpose of reducing the frequency of security issues and to mitigate the impact if a possible security breach does occur. The Android core operating system is built on top of the Linux kernel, so Android can also take advantage of Linux based protection mechanisms such as isolation of processes [27]. In figure 4.1 the components in the Android operating system is shown, where each of the components relies on that all of the underlying components are thoroughly secured.

On the top there are the applications and services. There are two types of apps, preinstalled and user-installed apps. Preinstalled apps or system apps as they are also called come already installed on the device, either by Android or the

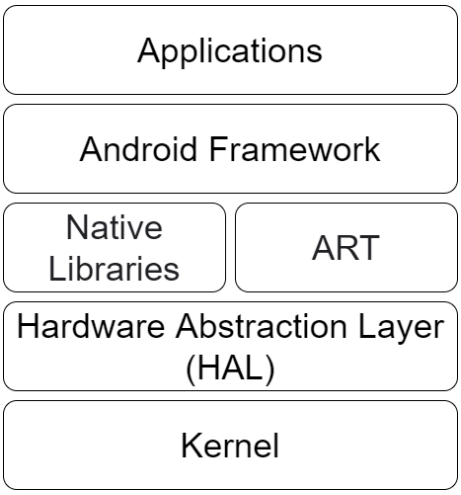

Figure 4.1: The Android operating system software stack.

device manufacturer and are placed under the */system/app* folder or */system/priv-app* folder in later releases of Android, both of which are read-only. System apps have the possibility to get permissions that user-installed apps are unable to get, meaning that they can access more of the system. For example, they can get permission to access the eUICC. User-installed app however are the apps that are possible for a 3rd party to develop. They can be downloaded by the end user and installed in the */data/app* folder on the device which has both read and write privileges. User-installed apps has the drawback that they are unable to access some APIs that require permissions only granted to system apps [27].

4.5.1 Application Sandbox

A big and important difference between Android and other Linux environments is the Android application sandbox. All software that run above the kernel, such as apps, including those written in native code and OS libraries are restricted to the application sandbox. The sandbox is a security mechanism used to isolate running apps and resources from each other and from the operating system in order to protect the apps and the system from malicious software. Android does this by using Linux user-based protection and assigning a UID (User Identifier) to each app and then runs them in their own separate process. The kernel then enforces the security between the apps and the system [28].

4.5.2 Permissions

Apps are run in the application sandbox and are only able to access a limited number of system resources and APIs. For an app to be able to access anything outside of the sandbox it must first be granted permissions to do so in order to protect the privacy of the user. Depending on what the app is trying to access and how sensitive this feature is, the system might automatically grant permission to

the app, or the user might first be prompted to approve or deny if the app should be given the permission. The permissions that an app needs must be declared in XML format in the app manifest using the *<uses-permission>* tag [29].

There are basically four different protection levels that the permissions in Android are divided up into:

- **Normal permissions:** Permissions to data or resources that do not pose any or a very little security risk to the user's privacy or to other apps. Any apps requesting these permissions will automatically be granted them during installation.
- **Signature permissions:** Permissions that are only given to apps that are signed by the same certificate as the app that defines the requested permission.
- **Dangerous permissions:** Permissions to data or resources that could affect other apps, data stored by the user or involve the user's private information, such as accessing the user's text messages. The user must first grant the app these permissions before they are granted.
- **System permissions:** Permissions only given to system apps and not to apps developed by a 3rd party.

Examples of some relevant permissions (although they are both system permissions) that might be useful and further discussed in this thesis are:

- android.permission.WRITE_EMBEDDED_SUBSCRIPTIONS
- android.permission.READ_PRIVILEGED_PHONE_STATE

4.6 eSIM in Android

Android has since Android 9 support for eSIM by providing standard APIs for accessing and managing eSIM profiles. This enables 3rd parties to both develop their own LPA apps as well as apps for managing eSIM, such as downloading, deleting and switching profiles for eSIM enabled devices [30]. This section will treat the *EuiccManager* used when downloading and managing profiles and will not deal with the subject of developing 3rd party LPA apps, as this is beyond the scope of this thesis.

4.6.1 EuiccManager

The *EuiccManager* is the application interface to the eUICC and the entry point for an application wishing to interact with the LPA. The APIs in *EuiccManager* provide high-level eSIM profile management for downloading, deleting and switching subscription profiles [30].

EuiccManager contains methods for checking if the running device has support for eSIM and to get information, such as the EID, about the eUICC. In addition to these, the four main methods in *EuiccManager* are:

- **downloadSubscription():** Attempts to asynchronously download a subscription profile. Takes a *DownloadableSubscription* as input, which in turn contains the activation code with the SM-DP+ address and activation code token.
- **switchToSubscription():** Enables the specified subscription profile (disabling the currently enabled profile if there is any). If the input is *SubscriptionManager.INVALID_SUBSCRIPTION_ID*, the currently active profile will be disabled, without enabling any other profile.
- **deleteSubscription():** Deletes the specified subscription profile. If the profile to be deleted is active when called, the device will first call *switchToSubscription()* with *SubscriptionManager.INVALID_SUBSCRIPTION_ID* as input, disabling it before it is deleted.
- **startResolutionActivity():** Used when the user of the device is able to resolve an issue that has occurred, for example when downloading or switching subscription.

Since all these methods may take some time to complete they are all performed asynchronously. They use PendingIntent callbacks where the PendingIntent contains a framework-defined result code that indicates if an error has occurred or if the operation succeeded, as well as a detailed result code from the LPA. The PendingIntent callback itself must be a broadcast receiver [30].

In some cases when the system is unable to perform an operation the error code is set to *EuiccManager.EMBEDDED_SUBSCRIPTION_RESULT_RESOLVABLE_ERROR*. This means that the error can be resolved by the user of the device. If for example a confirmation code is required to download the profile it will return this error code. The *startResolutionActivity()* in *EuiccManager* can then be called with the error code as input. This will prompt the user through the LUI to resolve the issue. If it was a confirmation code that was needed the user can enter this and then the profile download will resume [30]. The resolvable errors currently defined are:

- **ACTION_RESOLVE_DEACTIVATE_SIM:** which is used to prompt the user to accept that the action will deactivate the currently active SIM.
- **ACTION_RESOLVE_NO_PRIVILEGES:** which is used to prompt the user to accept that an action is performed even if the app does not have carrier privileges, such as switching profile where the app does not have carrier privileges over the currently active profile.
- **ACTION_RESOLVE_RESOLVABLE_ERRORS:** which is used to let the user resolve all resolvable errors, such as asking the user to input the carrier confirmation code.

The methods *downloadSubscription()*, *deleteSubscription()* and *switchToSubscription()* requires that the caller has either the *WRITE_EMBEDDED_SUBSCRIPTIONS* permission or is authorised to manage the target profiles by having carrier privileges. The method *getEid()* requires the *READ_PRIVILEGED_PHONE_STATE* permission or carrier privileges. Both permissions *WRITE_EMBEDD-*

ED_SUBSCRIPTIONS and *READ_PRIVILEGED_PHONE_STATE* are system privileges though and are only granted to system apps and not apps developed by a 3rd party.

4.6.2 Carrier Privileges

To make the profiles accessible only to the operator to which it belongs, Android only give apps access to some APIs if they have the required privileges. For a non-system app to be able to access the eUICC through the *EuiccManager*, the app needs to have carrier privileges. This is to prevent one operator being able to download, manage and delete another operators profiles [31].

This is done by the Android platform by loading the certificates that are stored in the profiles ARF (Access Rule File) and then only granting permission to make calls to the *EuiccManager* APIs if the app making these calls are signed by any of these certificates [30].

The process for an to gain carrier privileges can be divided up into five steps, which are described below [30]:

- The app's APK is signed and the public-key certificate is attached to the APK.
- In the preparation of the profile in the SM-DP+ the SHA-1 or SHA-256 signature of the app's public-key certificate, and optionally the package name of the app, are put in the ARF in the profile's metadata.
- The app tries to access a profile in the eUICC through the *EuiccManager*.
- Now the Android platform checks if the hash of the certificate of the app trying to access these APIs matches the signature in the profile's ARF, and if the optional package name is included in the ARF that this also matches the package name of the calling app.
- If signatures and optionally the package name matches then the app is given carrier privileges over that profile.

4.6.3 Devices With Multiple eSIMs

Starting from Android 10, it is also possible to build apps that support devices with multiple eUICCs installed in the same device through the *EuiccManager* class. This requires that an instance of the *EuiccManager* class is created for each of the eUICCs through the *createForCardId()* method, which pins the resulting *EuiccManager* object to a specific eUICC given its card ID, where the card ID is a value that uniquely identifies the eUICC. When the *EuiccManager* object has been created all following operations are directed to the eUICC with the given card ID [30]. Though no current mobile phone have more than one eUICC, it is very likely that as the eSIM replaces the regular SIM card that phones will come with more than one eUICC installed to be able to support dual SIM functionality.

Chapter 5
iOS and eSIM

iOS is the operating system developed by Apple based on the macOS and runs on their mobile phones and tablets, but unlike Android, iOS is not open source. Since the release of iOS 12.1, iOS has support for provisioning eSIM profiles, and as stated in section 3.4, many iPhones on sale today comes with eSIM capabilities. This chapter deals with how iOS apps are structured and work and also goes through how an iOS app could implement eSIM functionality.

5.1 SDK

The iOS SDK is a kit that contains the tools and frameworks needed to create applications for devices running iOS. The iOS architecture and SDK can be divided into four sets that can be seen as layers, as can be seen in figure 5.1. The lower layers contain fundamental technologies and services, while the higher layers are built upon the lower layers to create more complex and sophisticated services.

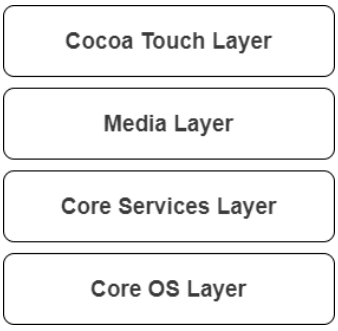

Figure 5.1: iOS architecture layers.

The Cocoa touch layer contains the main frameworks needed when building an iOS app, such as the UIKit framework which for example offers support for creating and managing the user interface and delivering user input as well as other types of input to the app. The Media layer and its frameworks are used when images, animations, audio or video is used in the app. The Core Services contain some frameworks that provide fundamental services needed to be able to build useful

apps, such as the Core Telephony framework which will be discussed later in section 5.5. The Core OS layer contains frameworks that are used to provide low-level services related to networks and hardware, such as access to the file system or access to Bluetooth. These frameworks are very limited to developers but are instead used by higher level frameworks to provide a service that the developer can use [32].

5.2 Xcode

Xcode is Apple's IDE (Integrated Development Environment) that is used to build and test software for Apple products, such as the iPhone, iPad and Mac [33]. It includes the latest SDKs for the different platforms and supports the officially supported programming languages, which include Objective-C and Swift. Xcode is only available for macOS and cannot be run on for example Windows.

5.3 UIKit

The UIKit framework contains all the core objects and infrastructure needed to build an iOS app. It provides ways for the app to interact with the system, display content on the screen and then let the user interact with the displayed content through various input methods. It also provides the main event loop through which it is possible to manage all the different interactions between the app, the user and the system [34]. Apps built with UIKit are structured based on the MVC (Model-View-Controller) design pattern where the model is responsible for all data and logic, the view presents the data visually to the user and the controller connects the model and view by moving data between them when needed [35]. This can be seen in figure 5.2 where also some core components in an app are show, such as the *UIApplication* object which runs the app's main event loop and is responsible for and manages the app's life cycle.

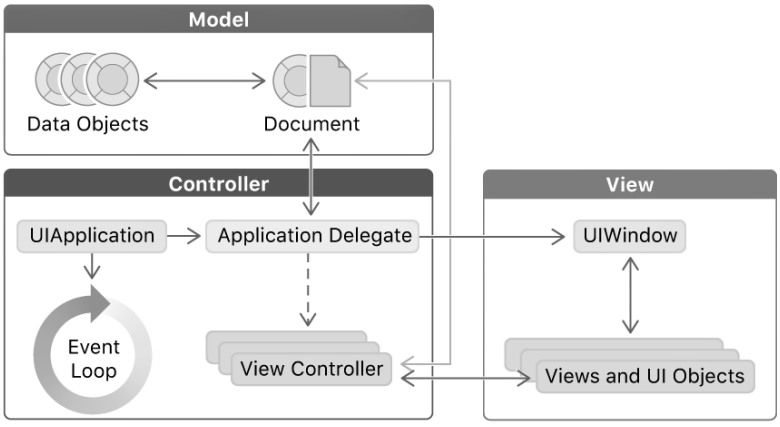

Figure 5.2: The core components in an iOS app implemented using UIKit, structured based on the MVC design pattern [35].

5.3.1 App's Life Cycle

During its life cycle, an iOS app will go through a set of several states, as shown in figure 5.3. When launched the system will put the app either in the background state or the inactive state if the UI is about to be displayed. Once the UI appears on screen in the foreground, the app is automatically transitioned to the active state by the system. Now the app will jump between the active and background state up until the time the app is closed, where it transitions to the suspended state and then back to not running [36]. At each of these transitions it is possible for the app to perform some tasks, such as saving data and releasing any shared resources when going into the suspended state and freeing up memory when going into the background state.

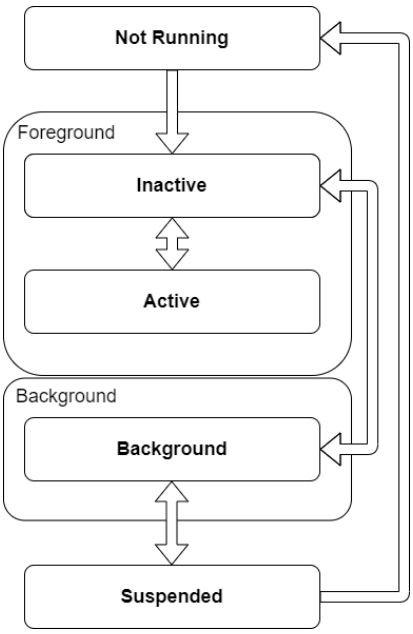

Figure 5.3: Life cycle of an iOS app.

Starting from iOS 13, it is possible to create several so-called scenes. A scene contains one instance of the app's UI and runs concurrently with the other scenes in the same app, sharing things like memory and process space [37]. The scenes in an app are all handled separately and have their own life cycle events independent of one another, and are even independent of the app itself. This means that each scene can be in different states at a given time [36].

5.4 Entitlements

Entitlements in iOS are rights or privileges that are used to grant apps particular capabilities, such as the use of a special service. They are configured in Xcode as key-value pairs and stored in a list file with the extension *.entitlements* [38].

The type of the value associated with an entitlement key depends on the key. Many keys take boolean values, i.e. true or false, but can take also string or arrays of strings as value [39].

The entitlements defined in the entitlements file for the app are then during the process of code signing the app combined with information from the developers account along with other relevant project information to determine and apply the final set of entitlements granted to the app. These entitlements are then embedded in the app's code signature [38][39].

5.5 eSIM in iOS

The *Core Telephony* framework located in the core services layer provides services to manage for example subscriber information and get information about the user's cellular service provider. Since iOS 12.1 it also has support for managing eSIM through the classes *CTCellularPlanProvisioning* and *CTCellularPlanProvisioningRequest* [40].

The *CTCellularPlanProvisioning* class contains two methods, *supportsCellularPlan()* and *addPlan()* [41]. Unlike in Android, there are no methods that can be used to enable, disable or delete a profile from an app. The only two existing methods are described below:

- The *supportsCellularPlan()* method checks to see if the the device has support for eSIM.
- The *addPlan()* method starts the provisioning process for a new eSIM profile. As input *addPlan()* takes a completion handler that executes after the provisioning process is complete, indicating if it succeeded, failed or ended up in an unknown state, and a *CTCellularPlanProvisioningRequest* used to identify the eSIM profile to download. When an app calls this method, an iOS wizard will start and guide the user through the installation and configuration process [42].

The *CTCellularPlanProvisioningRequest* in turn contains the SM-DP+ address and optionally the activation code token, confirmation code, EID, ICCID and OID, which is the provisioning request's Object Identifier and is used as input to the *addPlan()* method in *CTCellularPlanProvisioning* [43].

To be able to use any of these classes and methods, the calling app must have the right entitlements. Entitlement for eSIM access in iOS can be requested from Apple by submitting the form at `https://developer.apple.com/contact/request/esim-access-entitlement`, but this entitlement is not granted for everyone because it is exclusive for wireless carriers apps and can only be requested by the account holder of a development team in the Apple developer program.

Chapter 6
Proposed eSIM Re-Selling Solution

Most research concerning eSIM has been done in the area of security, which is only natural since it is still a relatively new technology and the aspect of security is extremely important due to the sensitive information involved. Also, as the specifications from GSMA only cover the case of a single operator that has control over the remote provisioning system, this leaves the area of how a 3rd party could fit into the eSIM ecosystem and how they might use eSIM unexplored.

In this chapter a proposition for an architecture for re-selling eSIM subscriptions in a 3rd party mobile app was developed, starting by discussing the different limitations and challenges that arise with a 3rd party re-selling solution and describing a couple of alternative solutions to the problem. It then continues with describing the different parts of the proposed solution and how they interface with each other. This chapter then finishes off with describing a few customer journeys intended to show how the system could work from the user's perspective.

6.1 Limitations and Challenges

The biggest limitation in designing and implementing an app for re-selling eSIM subscriptions is the need for eSIM access entitlements in iOS and carrier privileges in Android for it to be possible to initiate the profile download directly from the app. And as the profile is owned and protected by the carrier all the way from being created to being installed on the eUICC, as described in sections 3.2 and 3.3, it is not possible for anyone except the carrier that owns it to make changes to the profile. Both the Android and iOS operating systems implement eSIM capabilities in such a way that only a carrier could build an app to download and manage eSIM and to prevent 3rd parties from being able to so. As Apple states that eSIM access entitlements is exclusively for carriers it might not ever be possible for a 3rd party to get these privileges for their app, at least not in the time span for this thesis. Also, as previously stated, an iOS app would only be able to download profiles and would not be able manage them later, such as switching between profiles, as there are no APIs to support these operations, which would be possible in an Android app. A 3rd party Android app would also, as explained in section 4.6.2, be able to get the carrier privileges needed to download an eSIM just from cooperation with the carriers whose eSIM is to be sold in the app. This makes an Android app more interesting when designing and implementing the solution.

A non-technical challenge could be the fact that the operators might not want to sell their eSIM through a 3rd party app. As discussed in section 2.4, many of the operators fear that eSIM in consumer devices will lead to increased competition between the operators, which in turn could lead to a loss of customers for some of the operators. So, getting the operators to sell eSIM through an app where it can be directly compared with other operators' offerings might be hard.

6.2 Two Possible Solutions

Due to the possibility that the eSIM re-selling app may or may not have eSIM access entitlements or carrier privileges, two possible solutions immediately arise, one for each possibility. In this section these two different possible solutions to the problem of re-selling eSIM subscriptions in an app will be presented.

6.2.1 Solution Without Carrier Privileges or eSIM Access Entitlements

In the first solution the user can buy an eSIM in the app from their preferred carrier and get the activation code containing the SM-DP+ address and activation code token for the profile in return. The user then inputs them into the eSIM settings that already exist in the device used for downloading and managing eSIM profiles, i.e. the LUI. This means that the app is only used to buy and retrieve the activation code and not actually used to download or manage the eSIM profiles. This is the easiest solution because the app will not need any special privileges or entitlements but will negatively affect the user experience as there are more steps involved in the process for the end user. A sequence diagram over how this could work is given in figure 6.1.

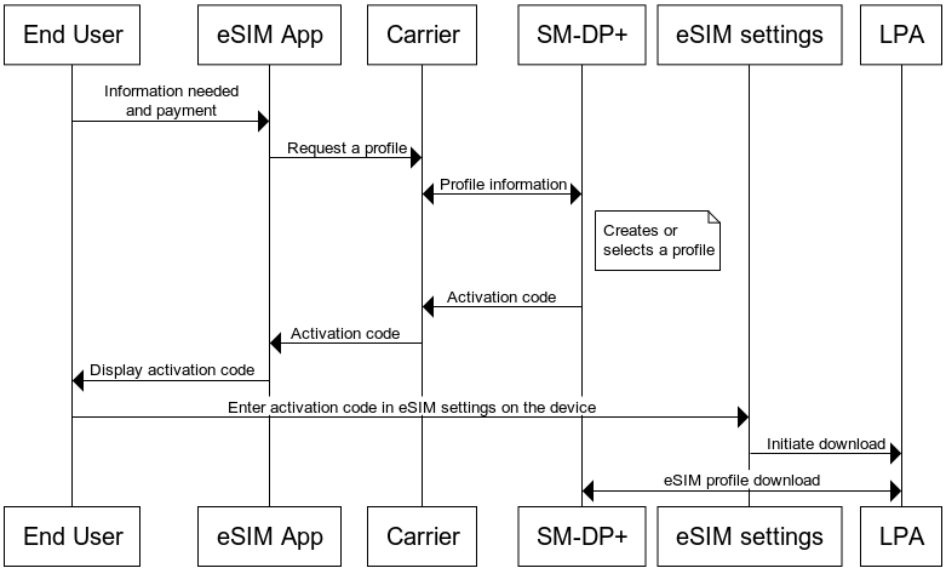

Figure 6.1: Solution without carrier privileges or eSIM entitlements.

The steps required in this solution and shown in figure 6.1 would be the following:

1. The end user selects their preferred carrier and eSIM plan to purchase. The end user inputs all necessary information, such as personal details and payment information.
2. When the payment has been approved, the app requests a profile from the selected carrier for the type of eSIM plan.
3. The carrier communicates with the SM-DP+, which either selects a previously created profile or creates a new one. The carrier gets the activation code in return from the SM-DP+ used to identify the profile later when it is to be downloaded.
4. The carrier returns the activation code to the app. If used, the carrier will also provide a separate confirmation code.
5. The app displays the activation code to the user and gives instructions on how to download the eSIM on that device.
6. The user follows the directions and inputs the activation code into the eSIM settings on the device. If a confirmation code is used, the user will be asked to enter it.
7. The eSIM settings can then communicate with the LPD in the LPA on the device and request it to start a profile download.
8. The LPD downloads the profile from the SM-DP+ and installs it on the eUICC.

Note that step 3 may be performed differently depending on how the carrier handles eSIM. It may be the case that the carrier has already had the SM-DP+ create several profiles and has stored the activation codes for these in a database. If that is the case then no interaction with the SM-DP+ is necessary in this step.

6.2.2 Solution With Carrier Privileges and eSIM Access Entitlements

A second, more advanced solution would be to also let the app handle the communication with the LPA so that it can start the download. This will remove the step where the user must enter the activation code manually into the eSIM settings on the device, thus making it more user friendly. The drawback with this solution would be that to be able to communicate with the LPA the app must have eSIM access entitlements in an iOS app and carrier privileges in an Android app. If an Android app has these carrier privileges though, then it will also be possible do more with the eSIM profiles in the app, such as switching between the profiles that it has carrier privileges for, and also deleting them. This will not be possible in an iOS app even with eSIM access entitlements as the only two methods available concerning eSIM are the *supportsCellularPlan()* that checks if the device supports eSIM and *addPlan()* that initiates the download of a new profile. There are no methods for switching or deleting profiles as there are in Android, this must always be done from the eSIM settings on the device.

The steps required in this solution, which are shown in the sequence diagram in figure 6.2, would be the following:

1-4. Steps 1-4 are the same as in the previously described solution.

5. The app can now, instead of displaying the activation code to the end user, directly communicate with the LPA and is therefore able to request the download of a new profile with the provided activation code. If a confirmation code is used this must still be displayed to the end user as it must be entered manually by the user.

6. The LPD downloads the profile from the SM-DP+ and installs it on the eUICC.

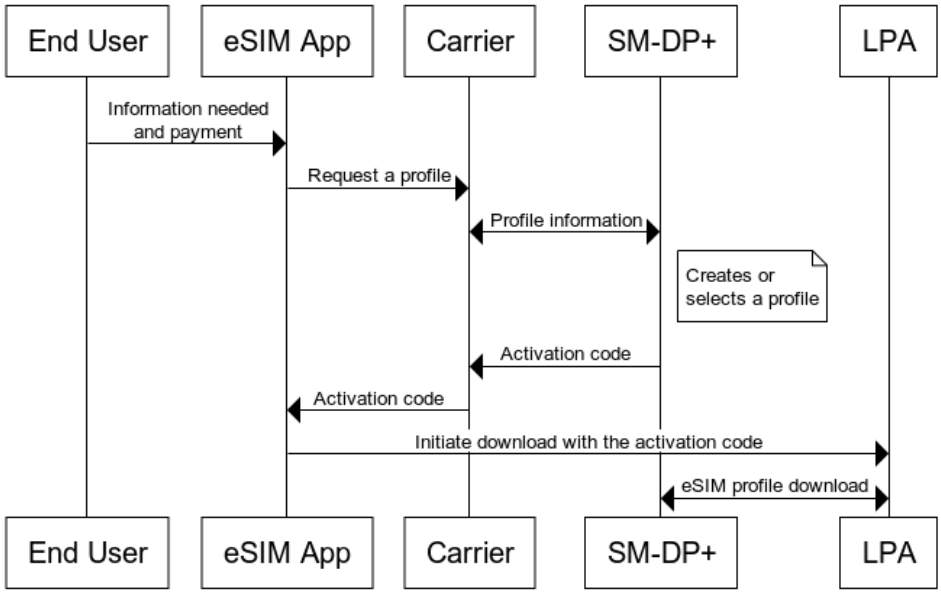

Figure 6.2: Solution with carrier privileges and eSIM entitlements.

This would be the preferred solutions if possible, because as stated in section 2.1.3, Gerpott and May writes in [5] that it is important that the development of the tool used by the consumer to manage the eSIM profile strive to be as easy to use as possible, and this solution where the app could perform the download without the user having to deal with any QR- or activation codes is from the users point of view the simplest.

6.3 Interface to MNOs and MVNOs

As mentioned in section 3.2.5, this interface is not standardised and will therefore look different between different carriers. The most straight forward solution would be to let the app directly handle the communication with the carriers as shown

in figures 6.1 and 6.2. A solution like this would however also mean, since the interfaces might look different, that all of the carriers included in the eSIM re-selling app would have to be treated differently in the phase when the app has to communicate with them. This will make it harder to add a new carriers to the app as this will include implementing a mechanism that can handle the communication between that specific carrier and the app.

6.3.1 Back-End Server

To get around this problem with the app having to be able communicate with every different carrier, a better solution would be having the app just communicate with a back-end sever containing the activation codes for all the profiles of the different plans available for download. The company behind the app would then get the activation codes from the partaking carriers and could handle both payment and delivery for the activation codes. When an eSIM is bought in the app, the company would pay most of the revenue to the concerning carrier but would also keep a small share for themselves as commission. Alternatively, the company could buy profiles from the carriers in bulk for a lower than retail price and then when an eSIM is sold in the app, keep the whole revenue.

The carriers would in any case have the eSIM profiles pre-provisioned as with the pre-printed eSIM vouchers explained in section 2.1.2, but instead of printing the details on a card, they would be sent and kept in the company's server. The profiles themselves would still be kept in the carriers SM-DP+ server.

This means that when a user wants to buy an eSIM in the app, the carrier would not need to be involved in any way as the company would handle both the payment and the delivery of the activation code. This would make the app simpler to implement as the interface to the server will look the same regardless of which carrier the end user wishes to purchase an eSIM from. It would also make it easier to add new carriers to the app. The problem now would instead be that this server would have to be able to be refilled with new activation codes from the carriers when running low. How this could be done and work will not be treated, but more details on how the server could work is given in section 6.6.

In any of the two cases described above, the app must be able to interface to either the carriers or the back-end server to retrieve what eSIM plans that the different carriers are currently offering to be able to present them to the user. This makes the app more dynamic and eSIM plans can easily be added, removed or changed without having to change anything in the app.

6.3.2 Tracking Usage and Refilling

Once the eSIM profile is downloaded and installed on the device it will work just like any other SIM from the operator it belongs to. So for example if the period of validity has expired or the amount of data, number of text messages or call minutes remaining on the eSIM is running low, this will be handled by the operator the same way as if the eSIM was bought directly from the operator, usually by sending a text message to the user. Then to top up a prepaid eSIM is done the same way

as with a regular SIM card, i.e. there is no special top up for eSIM because it uses the same top ups as regular SIM cards. This means that the refilling can be bought anywhere the operator sells top ups for prepaid SIM cards, for example directly on their own website and in their stores, but it can also be bought in the form of a voucher which is then used to top up the eSIM in places like grocery stores, gas stations and anywhere else that is a re-seller for the operator. It is therefore possible to also sell top ups for eSIM in the app because as mentioned, systems like these that let 3rd parties sell top ups for prepaid SIM cards already exist and are being used.

6.4 Interface With eSIM in Device

If the app is to support the download and possibly the management of eSIM profiles then it must interface with the LPA on the device. This is done through the APIs described in sections 4.6 and 5.5 for Android and iOS apps respectively.

If the solution however is to let the end user manually enter the activation code into the eSIM settings on the device then the app does not need to interface with the LPA on the device. On an Android device at least, it is possible to let the app directly open the eSIM settings on the device so that the users do not need to find this setting on their own. This is done by creating an intent with the string ACTION_MANAGE_EMBEDDED_SUBSCRIPTIONS in the *EuiccManager* class as input and starting a new activity with this intent.

6.5 End User Interface

The interface to the end user in the app must be able to handle many different tasks. It must be able to provide an interface to the end user where the user can select the country where the eSIM is to be valid and then be able to compare carriers and their offered eSIM plans for that country, as well as provide a way to proceed with the purchase when the user has decided from which carrier and which eSIM plan to buy. After that, the user should be shown a list of all possible payment methods available and be able to select their preferred payment method as well as enter any personal details needed, such as an email address for confirmation after the purchase. After this, the user interface depends on the payment method the user selected. If for example the user selected card payment the user will be given the possibility to enter their card details.

When the payment is completed the next step will depend on whether the app has carrier privileges or not. If it does not, the user must be given directions on how to install the profile in the device settings as well as be presented with the activation code. This can be done through a short text description and possibly images or a video for further clarification. The user can also in an Android app, as previously discussed, be given the option to go directly to the eSIM settings by for example pressing a button or a link.

If the app has carrier privileges however then the app can start the download of the profile. This may take some time so the user must be informed that the download is in progress via the user interface. Also, if a confirmation code is

needed the interface must be able to let the user to enter it. When the download is completed the user should be notified that the profile has been successfully downloaded. Also, if the app has carrier privileges, the app will be able to present an interface to the user with an overview of all the eSIM profiles installed on the device for which it has carrier privileges. It can then show the name of the carrier and the subscription for each profile and whether it is enabled or disabled. The user should also be able to switch between and delete the profiles from this interface.

In addition to this it should also be possible to present a user interface where the end user is able create an account with for example their email address and a password and then signing into this account. Why this is needed will be explained in the next section.

Even though, as stated in section 3.3.5, it is not yet possible to transfer a profile between devices when for example buying a new mobile phone, it might be worth keeping in mind that this operation might be realised in later specifications, leading to the app being able to also implement support for this operation.

6.6 Back-End Server and Security

Most apps require a back-end for some aspects and functionalities of the app to work as intended, this can for example be for storing data or in some cases much of the app's logic can even be located in the back-end. There are a few different choices on how to deploy and implement a back-end, and depending on the need some option may be better suited than the other. The back-end can for example be hosted in a cloud environment, but it is of course also possible to develop and host one's own back-end server. There exist many companies offering the service of hosting the back-end in the cloud, and the main advantage with this option is the flexibility of the server's capacity and that the service is managed by the cloud service provider. However, the advantage of hosting one's own server is that you will have complete control over it, but then will also have the management responsibility. Whichever way the server is hosted it still has to be implemented with a level of security that is suitable for the app in question. In this section the security concerns with a back-end for an eSIM re-selling solution will be discussed and some measures that can be taken to ensure the security will be proposed.

6.6.1 Secure Communication and Authentication

As previously described, the back-end in the solution would have to communicate both with the app and the different carriers. To ensure that the data sent is not compromised, the communication between all the entities has to be secured, and to ensure that the receiver of the data is the one intended, authentication in needed.

Between Back-End and Carriers

The back-end needs to communicate with the carriers to receive new activation codes for the eSIM profiles sold in the app. In addition to protecting the data when transporting it from the carrier to the back-end, the back-end needs to authenticate

the carrier to ensure that it is in fact the carrier it is communication with, and even more important, the carrier has to authenticate the back-end so not to send the activation codes to someone pretending to be the back-end server. All this can be solved by using TLS (Transport Layer Security) with mutual authentication when communicating. In the beginning of every communication the TLS handshake is performed where the back-end sends its certificate to the carrier and the carrier sends its certificate to the back-end. Both parties can then verify the validity of the certificates, and if they are authentic, proceed with the key negotiation for the key used to encrypt the data. The data can then be sent knowing that both parties are authenticated, and the data protected by being encrypted.

Between App and Back-End

The communication between the back-end and the app also needs to be protected so that the activation codes sent to the app cannot be read by anyone else. The server also has to authenticate itself to the app. As with the communication between the back-end and the carriers, a solution to this can be to use TLS. Then the app can be sure it actually communicates with the back-end by verifying its certificate and the data sent can be encrypted. The back-end server however must also be able to determine that it is the same user that downloads the activation code that it was that bought it in the first place. So, if for example one user buys an eSIM, no one else but that user should be able to download the activation code from the back-end server for that eSIM. This means that the user must also be authenticated to the back-end. This cannot be solved by using a certificate as the server does, as it is not a realistic solution that each user should have their own certificate. One possible solution however could be to let the user create an account with for example their email address and a password where these credentials are stored in the back-end. The email address would then uniquely identify the user and the user could be authenticated by knowing the password. The user can then sign into their account when buying an eSIM, making it possible for the back-end to determine which user it was and only letting that user get access to the activation code for the eSIM in question.

6.6.2 Secure Storage

Once the activation codes are in the back-end server they will have to be stored there until they are downloaded by a user, for example in an SQL (Structured Query Language) database. Also, if using accounts for authentication of the users this account information will also have to be stored in the database. In addition to other protection mechanisms, such as firewalls, to protect the server, the activation codes could also be protected when stored in the database by being encrypted. So, in the event of an attack where the attacker gains knowledge of the contests of the database, the activation codes will still be unknown to the attacker. The passwords stored in the database should also be protected and not stored in clear text, this is usually done by hashing the passwords and storing the hash.

6.7 Process Workflow

To illustrate how the entire process is intended to work and how the information flows between the different entities, figure 6.3 was created. Here it can be seen how the end user, the eSIM app and other parts of the device, the back-end server as well as the carriers communicate with each other and work together to enable the eSIM re-selling app. The arrows in the figure represent that the connected entities are communicating with each other and the numbers on the arrows is a step in the process involving communication between the connected entities.

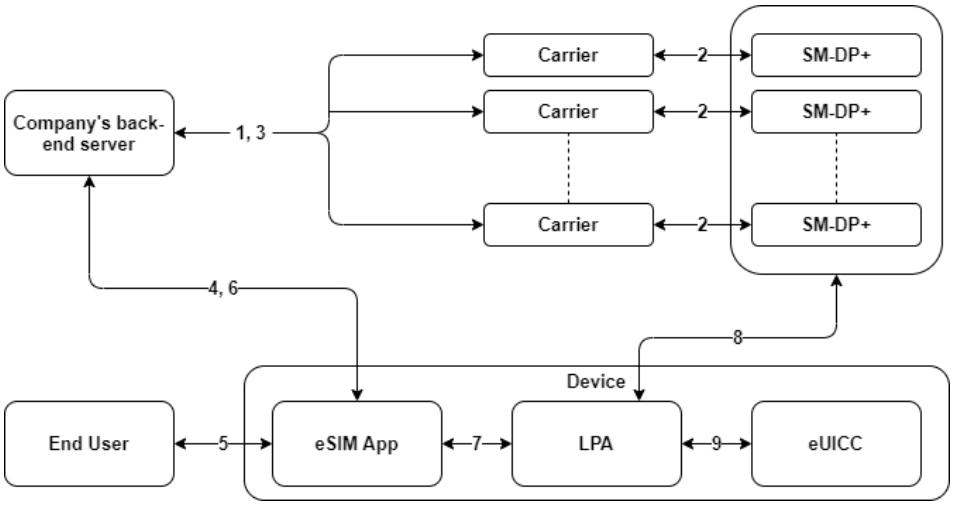

Figure 6.3: Process workflow for the proposed solution.

The steps in the process that enables the eSIM re-selling app are as follows:

1. The company's back-end server sets up a secure communication channel to a carrier with mutual authentication and requests new profiles for a certain plan from that specific carrier when they are starting to run low.
2. The carrier creates new profiles in their SM-DP+ server as if creating pre-printed eSIM vouchers and retrieves the activation codes for these.
3. The carrier responds to the company's back-end with the activation codes for the profiles it had created, which are then securely stored in the server's database.

Steps 1, 2 and 3 are done continuously and independent of the use of the eSIM re-selling app. This process is similar to the one in stores selling physical prepaid SIM cards, except the store is now replaced by the back-end server and the SIM cards are replaced by eSIM activation codes that are kept in the server's database. The customer, i.e. the user of the app, can then browse this store and purchase one of the products, which in this case is an activation code.

The rest of the steps in the process concerns the part of buying and downloading the eSIM profiles:

4. When the app is opened, it sets up a secure connection with server authentication to the back-end and retrieves the different plans available from the different carriers and displays them to the user.
5. The user selects one of the plans to buy and enters any relevant information to complete the purchase. If the user was not already signed into their account then this has to be performed before this step. The payment procedure can then for example be bought in as a service and will not be further treated here.
6. The app then connects to the back-end that is able to authenticate the singed in user and retrieves the activation code from the database for the plan bought by the user and sends this to the app.
7. The app starts the download process through the LPA with the activation code.
8. The LPA connects to the SM-DP+ and downloads the profile matching the activation code token specified in the activation code.
9. The profile is installed on the eUICC.

This illustrates how the system works if the app has eSIM access entitlements or carrier privileges. The only difference if it does not have the eSIM access entitlement is iOS is that step 7 could not be performed, and in Android without carrier privileges, in step 8 the LPA would first download the profile metadata and the app would then be denied carrier privileges and the download would stop. This means that the user would have to perform step 7 manually through the eSIM settings on the device as previously explained.

6.8 Customer Journeys

In the following section a few customer journeys will be described, showing how the app is intended to work from the end user's perspective.

6.8.1 A User Wants to Buy and Download an eSIM Profile

In this customer journey it is assumed that the user has created an account in the app and is authenticated by being signed in.

1. The user opens the app and selects the page for buying eSIM.
2. The user selects the country in which the eSIM should be valid.
3. All the carriers supporting eSIM for the chosen country and their plans are then displayed so the user can compare and select their preferred carrier and plan.
4. The user proceeds with the purchase by pressing the button to buy their chosen plan.

5. The user enters personal details such as email address for confirmation and selects payment method.
6. The user performs any steps necessary for the chosen payment alternative.

7a If the app has carrier privileges for the profile to be downloaded or eSIM access entitlements then the app will now automatically download and install the eSIM profile.

7b If the app does not have carrier privileges or eSIM access entitlements then the app will display the activation code to the user and give instructions on how to manually install it on that device and direct the user to the eSIM settings on the device.

6.8.2 A User Wants to Enable/Disable a Profile

1. The user opens the app and selects the page for managing profiles.
2. The user finds the profile to be enabled or disabled in the list over all profiles installed on the device for which the app has carrier privileges.
3. The user presses the switch next to the profile.

4a If the profile is enabled the app will now disable it.

4b If the profile is disabled, the app will first disable the active profile if there is any and then enable the selected profile.

If it is an Android app, and the app does not have carrier privileges for a profile, then the user will not be able to directly enable or disable it through the app. The user will however be able to enable any other profile that the app has carrier privileges for (and thereby disabling the active one) even if the app does not have carrier privileges for the currently active profile. The user will then be prompted to give consent to disable the currently active profile before it can be disabled, and the new profile enabled.

6.8.3 A User Wants to Delete a Profile

1. The user opens the app and selects the page for managing profiles.
2. The user finds the profile to be deleted in the list over all profiles installed on the device for which the app has carrier privileges.
3. The user presses the delete button next to the profile.
4. The user is prompted to verify the user's intent to delete the profile.
5. The user presses accept. If the profile was enabled then it will first be disabled. The app then proceeds to delete the profile from the eUICC.

Chapter 7
Proof of Concept

This chapter will deal with the implementation of a proof of concept for the solution suggested in the previous chapter and will be used to show that a 3rd party could indeed take the role of a re-seller of eSIM subscriptions on a mobile app. It will also be used to explore if the proposed solution actually makes it easier for the consumer to buy their eSIMs than it is to buy them directly from an operator.

7.1 Implemented Solution

The proof of concept consists of the client part of the solution in the form of an app for Android devices combining both of the solutions described in the previous chapter. When buying an eSIM, the app will try to download and install the profile, but if this fails due to the app not having carrier privileges the app will then instead fall back to the other solution where the user is presented with the activation code and directed to the device's eSIM settings. This way the app could be used with any carrier regardless if the app has carrier privileges to handle every carriers' profiles or not. The app should then also be able to manage any eSIM profiles that it has carrier privileges for. The app will not in this proof of concept have any communication with a back-end server as the main challenges with the eSIM re-selling solution lies in the mobile operating systems and not in implementing a secure back-end server.

7.2 Methodology

This section gives a brief overview of how the app was created, what it should be able to do, and how it was tested.

7.2.1 Android Studio

The app was written in Java and developed using the Android Studio IDE, a software containing the Android SDK that is specifically designed for developing Android applications, and that for example makes it easier for the developer to create new activities, change the app manifest and install the app on a device for testing.

7.2.2 Permissions and Carrier Privileges

Since it requires cooperation from a carrier to gain carrier privileges, as is described in section 4.6.2, which are needed for the app to be able to download and manage their profiles, the app will not be able to gain these privileges due to the fact that no such cooperation could not be realised in the time frame for this thesis. However, as discussed in section 4.6.1, the APIs used in Android to manage eSIM can also be used by an app that has the *WRITE_EMBEDDED_SUBSCRIPTIONS* and *READ_PRIVILEGED_PHONE_STATE* permissions. This means that an app that is granted these permissions will act just like an app that has carrier privileges. So, to be able to test the app as if it had carrier privileges, it needs to be granted these permissions. The problem with this is, as previously mentioned, that these permissions are only granted to system apps and not 3rd party apps. System apps are as mentioned in section 4.5 all the apps installed in the /system/priv-app folder, which is read-only. To be able to get write permissions to this folder the phone used was rooted, a process in which the user of the phone gains root access to the system. With root access it is possible to install any app as a system app by installing it in the /system/priv-app folder, and thereby being able to grant system permissions to the app by declaring them in the app's AndroidManifest.xml file.

7.2.3 Testing

The app was tested on actual hardware. It was installed on a Google Pixel 4 running Android 10 and tested using a prepaid eSIM from the Swedish mobile operator Tre. In a successful test the app was able to start the download and installation process of the eSIM profile and could then manage it when installed. With manage the profile means that it is possible for a user to through the app enable and disable a profile and also delete it.

7.3 System Design

The app consists of several activities that in turn handle all the different aspects of the eSIM re-selling solution. The main parts of the proof of concept are the download and management of the eSIM profiles, but as can be seen in the overview of the system design with all its different components in figure 7.1, there is more functionality a final product would have to implement.

When the app is opened, it starts up in the main activity. From here, depending on the user's input, it can continue to either an activity for creating an account or for signing into an already existing account, or it can continue to an activity where it is possible to see and manage all the eSIM profiles installed on the device for which the app has carrier privileges, or lastly to an activity where it is possible to browse the different eSIMs that can be bought in the app.

In the following sections the system as shown in figure 7.1 will be explained as well as functionality that is not yet implemented that would have to be for the whole system to work as intended.

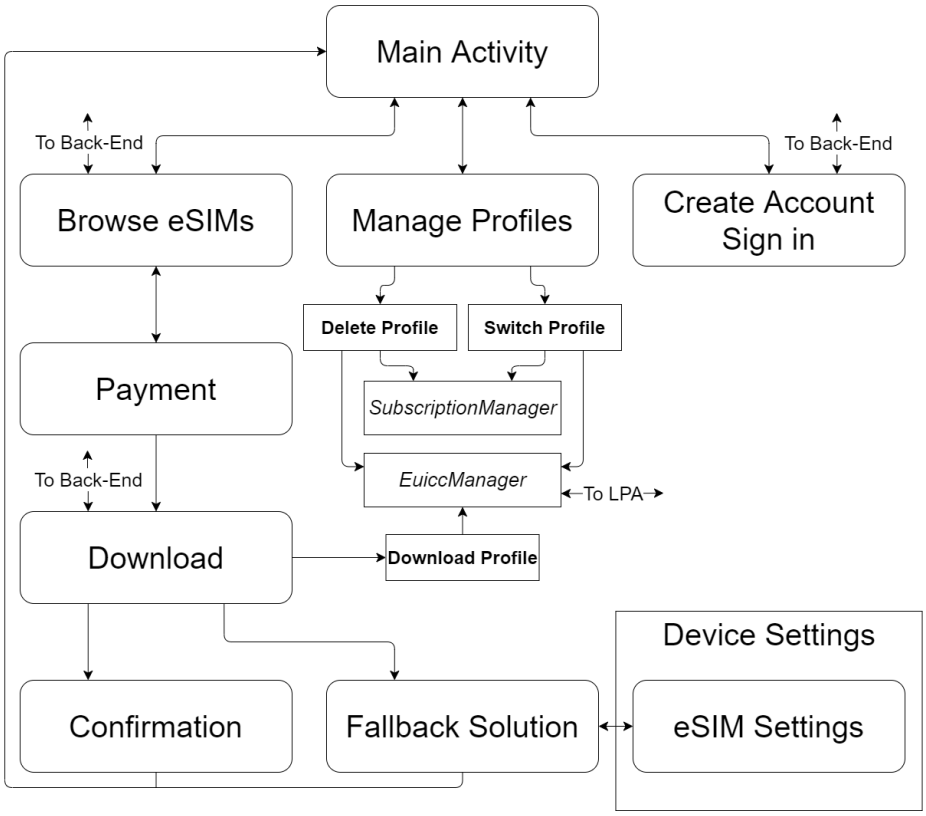

Figure 7.1: Overview of the system design.

7.3.1 Buying and Downloading eSIM

If the user selects to browse the available eSIM profiles, the app will set up a secure connection to the server and retrieve all the plans that are available. The user can then select the country where the eSIM should be valid and compare the eSIM plans offered by the different carriers. The user can then continue to purchase one of the eSIMs, leading to the activity that handles the selection of payment method and the payment.

Once the payment is completed the app changes to a new activity where the app can once again set up a secure connection to the back-end and downloads the activation code for the bought eSIM. This activity also contains all the logic needed to start the download of the eSIM profile as well as to handle any resolvable errors, and is shown as the *Download Profile* part in figure 7.1. It uses the *EuiccManager* class as discussed in 4.6.1 to communicate with the LPA as shown in figure 7.2. Once the app tries to download the profile, the *EuiccManager* makes calls to the *EuiccController* which is part of the Android platform. The *EuiccController* in turn finds the LPA to use as there might be more than one LPA on the device. The LPA can then communicate with the eUICC through the *EuiccCardController* and is able to start the download.

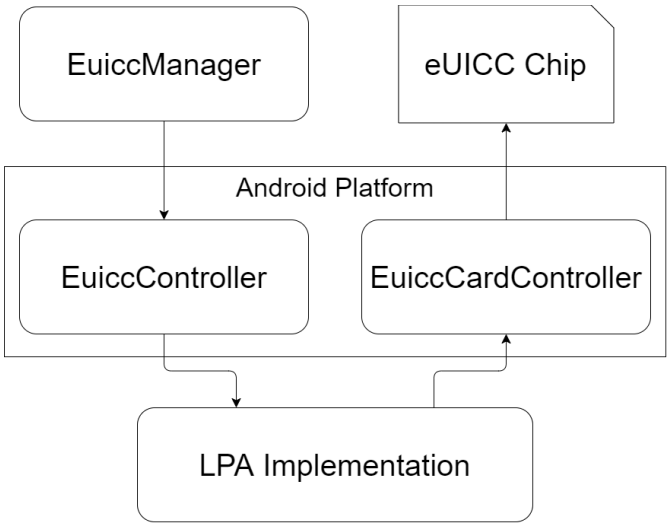

Figure 7.2: Communication between EuiccManager, LPA and the eUICC.

If the app has carrier privileges for the eSIM profile then the download will be successful, and the user is displayed a confirmation that the profile has been downloaded and is then directed back to the main activity. If the app does not have carrier privileges however, then the download will fail, and the activity called *Fallback Solution* in figure 7.1 is started. Then the user is presented with the activation code and given instruction on how to install the profile through the eSIM settings on the device. This activity is then able to directly start the eSIM settings activity as described in section 6.4.

7.3.2 Manage Profiles

If the user selects to go to the activity where it is possible to manage the installed profiles, all the profiles installed on the device for which the app has carrier privileges will be displayed in a list. This activity contains the logic for switching and deleting profiles which is shown as the *Switch Profile* and *Delete Profile* parts in the system overview in figure 7.1. They use the *SubscriptionManager* class to get the information about the installed profiles and then uses this information both to display the different profiles to the user and then also as input to the *EuiccManager* class when enabling, disabling or deleting a profile.

7.3.3 Unimplemented Functionality

As previously discussed, it is only the parts that handles the download, switching between and deleting the profiles that are fully implemented. The proof of concept app does not for example have communication with a back-end and can therefore not retrieve the eSIM plans available or the activation codes. Nor does it implement the payment or the part that handles creating and signing into an account.

Chapter 8

Evaluation and Discussion

There is nothing in the eSIM standard or specifications that prevents 3rd parties from being a part of the eSIM ecosystem and take the role as re-seller of eSIM subscriptions for the operators. The main difficulty however, and the biggest obstacle when designing and implementing a 3rd party mobile app for re-selling and downloading eSIM profiles is the need for carrier privileges in an Android app and eSIM access entitlements in an iOS app. As discussed, the eSIM access entitlements is approved by Apple and only given to mobile carriers. Whether it could be possible for a 3rd party to be granted this permission is uncertain and has not been further explored in this thesis, but if it is possible to show a partnership with at least one carrier, it should be possible to be granted the entitlement. Carrier privileges however that are needed by an Android app works differently and can as previously discussed be gained with cooperation from the carriers whose eSIMs are to be sold in the app. Based on this, the architecture for an eSIM re-selling system in chapter 6 was developed along with a proof of concept of the client part of the solution in chapter 7. This chapter in turn will discuss and try to evaluate the proposed solutions based on the challenges that exists with an eSIM re-selling solution.

8.1 Downloading eSIM Profile From 3rd Party Mobile App

The proof of concept in the previous chapter shows that it is indeed possible to build an Android app that could be used by a 3rd party to re-sell and download eSIM profiles to a mobile device. As explained though, this app was installed as a system app on a rooted phone to be able to gain some privileges only available to system apps to be able to download eSIM profiles. Distributing and using an app in the re-selling system that needs to be installed on a rooted phone as a system app by the user would of course be an unpractical and unrealistic solution. The app however works exactly the same as if it would have been given carrier privileges through being signed by the same certificate that exists in the profiles ARF. So if the app were to be signed and the corresponding certificate distributed to all the carriers whose eSIMs are to be sold in the app to be added in the profiles, then the app would not need the system permissions to work and could therefore be distributed and installed as any other app. A problem that might arise is that if a carrier only adds the certificate to the profiles that are then sold in the app and

not the other profiles they sell themselves, the app will only be able to manage the profiles bought in the app and not profiles from the same carrier bought somewhere else as these do not have the required certificate.

The alternative to this solution as discussed would be to not have the app download the eSIM but instead let the user do this through the eSIM settings on the device. This would solve the eSIM re-selling part but not the download and management of the profiles. It would be a simpler solution though as the app does not need carrier privileges and therefore operators do not need to create special profiles containing the certificate. The drawback would be the negative impact on the user experience as the solution would involve more steps and be more complex from the users' perspective. Due to this, the solution where the app has carrier privileges is to be considered as the better solution.

8.2 Does the Solution Make It Easier for the Consumer?

The whole reason a solution like this might be used is if it adds something that the consumer considers to be of value, something that makes it better than buying eSIM directly from the mobile operators, for example making it easier for the user of the app to buy their eSIM. So, does the proposed solution in fact make it easier to buy eSIM than it is directly from the mobile operator? The answer depends on a couple of things. The first is how the operators sell their eSIMs and the second if the solution has carrier privileges so that it can download the profile or not.

If the operator for example only sells eSIM online on their website or in their physical stores, meaning that the user needs to install the profile through the eSIM settings on the device, then the solution with carrier privileges should be considered as easier for the user as this step does not need to be performed as the download will be done automatically. If the operator however sells eSIM in their own app that has the possibility to download the profile then this will be easier for the user than the proposed solution without carrier privileges, and equally as easy as the solution with carrier privileges. So, the answer will vary between different operators as they sell their eSIMs differently and depending on whether the app has carrier privileges or not. Another thing though that would add to the value of the app from the consumers perspective is the possibility to compare the different operators and their offered plans in the app.

8.3 Unsolved Issues

There are still some unsolved issues with the proposed architecture however that needs to be solved first if the solution were to be implemented. As described in the previous chapter there are still some aspects of the eSIM re-selling app that are not yet defined and implemented, such as the communication to the back-end server and creating and signing into an account, which is also something that has to be solved in the back-end server. It is also not solved, besides on how to make the communication secure, how the re-selling app and back-end server, as well as how the carriers and the back-end server communicates. For example, it is not

solved how the back-end can request new activation codes from a carrier and how the response with the activation codes will look like.

A non-technical issue that has previously been mentioned is whether the mobile operators actually want to be part of a solution like the one proposed in this thesis. The studies looked at shows a varying attitude towards eSIM from the operators and a fear of the new competitiveness it might introduce. On the other hand, if a solution like this were to be implemented and some operators were to take part, more might follow as to not be left out and losing business by not being a part of it. The answer to the question of whether the operators want a 3rd party re-seller of eSIM subscriptions on a mobile app is quite uncertain but a very important one, as without the operators support there can of course be no re-selling system.

Chapter 9
Conclusions

9.1 eSIM Re-Selling

Mobile phones have only recently started being equipped with eSIMs and there are only a few models available on the market, but more and more are released all the time. Due to this, the mobile operators have had to implement remote SIM provisioning systems to be able to sell and provision eSIMs. This thesis has investigated how eSIM and remote SIM provisioning works to get an understanding of how a 3rd party could fit into the eSIM ecosystem, and has also studied the mobile operating systems Android and iOS to see if it could be possible for a 3rd party company to take the role as re-seller of eSIM subscriptions on a mobile app.

It found that there is nothing in the eSIM standard that prevents 3rd parties from being part of the eSIM ecosystem and to take the role as re-sellers of eSIM subscriptions for the mobile operators. However, there are some obstacles in both the Android and iOS operating systems. An eSIM re-selling app would need to have carrier privileges or eSIM access entitlements for Android and iOS respectively to be able to use the existing APIs used to download and manage the eSIM profiles. Whereas the carrier privileges can be gained from cooperation with the operator whose eSIMs are sold in the app, eSIM access entitlements are not as easy as it must be approved by Apple and are only granted to mobile carriers. It should however, as discussed, be possible for a 3rd party to gain these entitlements if it is possible to show a partnership with at least one carrier whose eSIMs are to be sold in the app.

Based on what was found, two different solutions were proposed. One where the app has the carrier privileges or eSIM access entitlements and one where it does not. An architecture for how a 3rd party eSIM re-selling system could work was developed together with a proof of concept on the client part of the solution for an Android device. All in all, it was found that it is indeed possible for a 3rd party to be an eSIM re-seller on a mobile app.

9.2 Future Work

As future work it could be interesting to further look into if it indeed could be possible for a 3rd party to be granted the eSIM access entitlements, thus also making it possible to download profiles from the app even on an iOS device. Also,

as the interviews with some of the operators that were planed for this project did not take place, it would be interesting to further explore how the proposed solution could be integrated to work with their systems and also to get a better understanding of their views on a solution for re-selling eSIM subscriptions. Also, exploring how the possibility of refilling a prepaid eSIM could be integrated in the system would be interesting future work.

References

[1] D. Gleeson, *eSIM Device Sales Forecast: Smartphones, Tablets, and Wearables, 2019–24*, 2019.

https://ovum.informa.com/resources/product-content/esim-device-sales-forecast-smartphones-tablets-and-wearables-201924-ces004-000118 Accessed: 2020-01-30.

[2] E. Vahidian, *Evolution of the SIM to eSIM*, Norwegian University of Science and Technology, Department of Telematics, 2013.

http://hdl.handle.net/11250/262765 Accessed: 2020-01-31.

[3] D. G. Koshy, S. N. Rao, *Evolution of SIM Cards – What's Next?*, International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, 2018, pp. 1963-1967.

https://ieeexplore.ieee.org/document/8554774 Accessed: 2020-01-21.

[4] B. A. Abdou, *Commercializing eSIM for Network Operators*, 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 616-621.

https://ieeexplore.ieee.org/document/8767260 Accessed: 2020-01-21.

[5] T. J. Gerpott, S. May, *Embedded Subscriber Identity Module eSIM*, Business & Information Systems Engineering 59, 2017, pp. 293–296.

https://doi.org/10.1007/s12599-017-0474-4 Accessed: 2020-01-21.

[6] P. Hristova and J. Bryan, *eSIM For The Roaming Consumer*, Roaming Consultancy Company Limited, 2018.

http://marketing.uros.com/ROCCO%20Roaming%20Consumer%20eSIM%20Strategy%20Report%202018.pdf Accessed: 2020-04-01.

[7] GSMA, *eSIM Whitepaper, The what and how of Remote SIM Provisioning*, 2018.

https://www.gsma.com/esim/wp-content/uploads/2018/12/esim-whitepaper.pdf Accessed: 2020-01-30.

[8] M. Meyer, E. A. Quaglia and B. Smyth, *An Overview of GSMA's M2M Remote Provisioning Specification*, 2019.

http://arxiv.org/abs/1906.02254 Accessed: 2020-03-30.

[9] A. Vesselkov, H. Hammainen and P. Ikalainen, *Value networks of embedded SIM-based remote subscription management*, 2015 Conference of Telecommunication, Media and Internet Techno-Economics (CTTE), Munich, 2015, pp. 1-7.

https://ieeexplore.ieee.org/document/7347220 Accessed: 2020-03-30.

[10] M. Tsurusawa, *Latest Trends in Remote SIM Provisioning Technology*, Special Feature - The future of cellular networks, New Breeze Vol. 29 No.3 Summer, 2017, pp. 1-10.

https://www.ituaj.jp/wp-content/uploads/2017/08/nb29-3_web.pdf Accessed: 2020-04-06.

[11] J. Park, K. Baek and C. Kang, *Secure Profile Provisioning Architecture for Embedded UICC*, 2013 International Conference on Availability, Reliability and Security, Regensburg, 2013, pp. 297-303.

https://ieeexplore.ieee.org/document/6657256 Accessed: 2020-01-21.

[12] S. Chitroub, N. Zidouni, H. Aouadia, D. Blaid and R. Laouar, *SIM Card of the Next-Generation Wireless Networks: Security, Potential Vulnerabilities and Solutions*, 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS), Bern, Switzerland, 2018, pp. 502-509.

https://ieeexplore.ieee.org/document/8910042 Accessed: 2020-01-30.

[13] M. Meyer, E. A. Quaglia and B. Smyth, *Attacks against GSMA's M2M Remote Provisioning*, Financial Cryptography and Data Security, 2018, pp. 243-252.

https://www.blackhat.com/docs/eu-17/materials/eu-17-Meyer-Attacks-Against-GSMAS-M2M-Remote-Provisioning-wp.pdf Accessed: 2020-04-02.

[14] Ernst & Young *Mobile network operator on-demand subscription management study*, 2015.

https://www.ey.com/Publication/vwLUAssets/EY-mobile-network-operator-on-demand-subscription-management/%24FILE/EY-mobile-network-operator-on-demand-subscription-management.pdf Accessed: 2020-03-30.

[15] GSMA, *RSP Technical Specification Version 2.2.1*, 2018.

https://www.gsma.com/newsroom/wp-content/uploads//SGP.22-v2.2.1-2.pdf Accessed: 2020-01-20.

[16] GSMA, *RSP Architecture Version 2.2*, 2017.

https://www.gsma.com/newsroom/wp-content/uploads//SGP.21_v2.2.pdf Accessed: 2020-01-20.

[17] T. Chang, *eSIMS: Everything You Need to Know*, WhistleOut, September 2019.

https://www.whistleout.com/CellPhones/Guides/esims Accessed: 2020-02-21.

[18] Huawei, *HUAWEI P40 Specifications*.
`https://consumer.huawei.com/en/phones/p40/specs/` Accessed: 2020-03-26.

[19] Apple, *Find wireless carriers that offer eSIM service*.
`https://support.apple.com/en-us/HT209096` Accessed: 2020-01-30.

[20] T-Mobile *T-Mobile Prepaid eSIM app*.
`https://www.t-mobile.com/support/plans-features/t-mobile-esim-app` Accessed: 2020-03-18.

[21] Android, *Application Fundamentals*.
`https://developer.android.com/guide/components/fundamentals` Accessed: 2020-02-07.

[22] Android, *Introduction to Activities*.
`https://developer.android.com/guide/components/activities/intro-activities` Accessed: 2020-02-10.

[23] Android, *Understand the Activity Lifecycle*.
`https://developer.android.com/guide/components/activities/activity-lifecycle` Accessed: 2020-02-10.

[24] Android, *Content providers*.
`https://developer.android.com/guide/topics/providers/content-providers` Accessed: 2020-02-10.

[25] Android, *Intents and Intent Filters*.
`https://developer.android.com/guide/components/intents-filters` Accessed: 2020-02-10.

[26] Android, *App Manifest Overview*.
`https://developer.android.com/guide/topics/manifest/manifest-intro` Accessed: 2020-02-10.

[27] Android, *Secure an Android Device*.
`https://source.android.com/security` Accessed: 2020-02-11.

[28] Android, *Application Sandbox*.
`https://source.android.com/security/app-sandbox` Accessed: 2020-02-12.

[29] Android, *Permissions overview*.
`https://developer.android.com/guide/topics/permissions/overview` Accessed: 2020-02-12.

[30] Android, *Implementing eSIM*.
`https://source.android.com/devices/tech/connect/esim-overview` Accessed: 2020-03-20.

[31] Android, *UICC Carrier Privileges*.
https://source.android.com/devices/tech/config/uicc Accessed: 2020-02-11.

[32] Naveen, *iOS Architecture*, IntelliPaat, December 2019.
https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/ Accessed: 2020-02-14.

[33] Apple, *Xcode*.
https://developer.apple.com/documentation/xcode Accessed: 2020-02-14.

[34] Apple, *UIKit*.
https://developer.apple.com/documentation/uikit Accessed: 2020-04-17.

[35] Apple, *About App Development with UIKit*.
https://developer.apple.com/documentation/uikit/about_app_development_with_uikit Accessed: 2020-04-17.

[36] Apple, *Managing Your App's Life Cycle*.
https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle Accessed: 2020-04-17.

[37] Apple, *Scenes*.
https://developer.apple.com/documentation/uikit/app_and_environment/scenes Accessed: 2020-04-17.

[38] Apple, *Entitlements*.
https://developer.apple.com/documentation/bundleresources/entitlements Accessed: 2020-03-03.

[39] Apple, *About Entitlements*.
https://developer.apple.com/library/archive/documentation/Miscellaneous/Reference/EntitlementKeyReference/Chapters/AboutEntitlements.html Accessed: 2020-03-04.

[40] Apple, *Core Telephony*.
https://developer.apple.com/documentation/coretelephony Accessed: 2020-01-30.

[41] Apple, *CTCellularPlanProvisioning*.
https://developer.apple.com/documentation/coretelephony/ctcellularplanprovisioning Accessed: 2020-02-14.

[42] Apple, *addPlan(with:completionHandler:)*.
https://developer.apple.com/documentation/coretelephony/ctcellularplanprovisioning/2981838-addplan Accessed: 2020-02-20.

[43] Apple, *CTCellularPlanProvisioningRequest*.
https://developer.apple.com/documentation/coretelephony/ctcellularplanprovisioningrequest Accessed: 2020-02-14.