Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

AG ID-Management

Automatic Detection of Android Device Security Properties

Janik Besendorf
Matrikelnummer: xxx
$\{vorname\}.\{nachname\}\{at\}fu-berlin.de$

Eingereicht bei:	Prof. Dr. Marian Margraf
Zweitgutachter:	Prof. Dr. Jörn Eichler

Berlin, May, 3, 2021

Abstract

Smartphones are becoming more and more popular. As a result smartphone security is an increasingly important subject, especially with state actors discussing eIDs on smartphones. However, information about a smartphone's specific security features is not readily available. There has been research to automatically gather smartphone security features, but the properties collected are not sufficient for evaluation of a smartphone's compliance to eID regulations such as eIDAS. In our thesis we explore sources of such information and aggregate information from these sources using web scraping, and by gathering information with an Android app. We found that most of the information required for evaluation according to eIDAS is available to the public and suitable for automatic aggregation. However, since information on websites is sparse, usually an app on a smartphone is required to gather all information. Also, information about security certifications is not readily available. We conclude that the stakeholders in the smartphone market should make an effort to improve this situation by providing more information on public websites and by increasing machine-readability of this information.

[1] `https://creativecommons.org/licenses/by-nc-sa/4.0`

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

June 1, 2021

Janik Besendorf

Contents

1 Introduction

1.1 Motivation

As online usage increasingly shifts from laptops and desktops to smartphones, and with smartphones being the only device for a lot of people [1], especially in emerging markets [2], consumers and application developers alike face the question of how secure those devices are. However, manufacturers often provide little information regarding the security properties of their devices [3]. Also, there is no central database gathering such information for all smartphones.

It would be especially helpful for banking and e-government applications, which have to comply with regulations, if such information were readily available in order to evaluate which smartphones are capable of fulfilling certain requirements. An example of a regulation which focuses on certain hardware security features is the EU regulation on electronic identification and trust services for electronic transactions (eIDAS) [4]. eIDAS defines electronic identifications (eIDs) that are recognized in the whole EU single market, and which can be used online to identify and authenticate users to public or private entities [5]. This enables e-government services like filing taxes online or can be used for age verification for video-on-demand sites.

In Germany, the only currently notified eID is the "Online Ausweisfunktion" that is integrated in the state issued identity card [6]. In order to eliminate the need to carry the identity card and thus increase the usage of the eID, solutions that integrate the eID in smartphones are being discussed [7]. Therefore, information about the security features available on smartphones would be of great help when evaluating which smartphones are suitable for eIDAS eIDs. An ideal solution to make this information readily available would be a centralized database with information about smartphone security features.

1.2 Research Question

To address the shortage of readily available information on smartphone security feature, this thesis shall explore the following research questions:

Which smartphone security features are required to evaluate compliance to regulatory requirements for a mobile eID solution? What publicly available sources offer access to such information, and how might this information be aggregated automatically?

1.3 Scope

The security features available on a smartphone are dependent on the hardware installed in the smartphone and then utilized by the operating system [8][9]. At the moment, Android and iOS are the two main smartphone operating systems with approx. 97 % market share and Android accounting for the bulk of this [10]. When considering an eID solution that aims to support most smartphones currently in use, Android and iOS are the natural targets. As the security architectures and application SDKs differ quite significantly from one another [8][9], this thesis can only focus on one of those operating systems. In contrast to iOS, Android is the more open ecosystem, the source code is open and security features are well documented [11]. iOS is not open source, and developing apps for the OS requires a Mac computer, so we focus on Android smartphones [12].

eIDAS eID systems are classified into different assurance levels. In this thesis we want to focus on the requirements for an eID solution in compliance with the assurance level 'substantial', as in Germany related work like the OPTIMOS 2.0 project and technical regulations like the BSI TR-03159 mostly discuss 'substantial' [13]. Also, some phones are already certified for the use with an eID solution with assurance level 'substantial' such as the Samsung S20 model line [14].

When considering the scalability of our methods, we want to look only at those that can be automated or are at least semi-automatic in order to fill a large database of security properties.

As mentioned in the motivation, an eID system is not the only reason to improve availability of information on smartphone security properties. Therefore, we do not limit the gathered information to those needed for eIDAS compliance and try also to also gather other properties that might be relevant for other regulations, certificates or security-conscious users.

Security vulnerabilities and patches are not within the scope of this thesis, as this topic is too large and complex in detail. We will rather focus on hardware-backed features and certifications.

1.4 Methodology

There are two main sources of information about smartphone security features: First, publicly available resources on the web like the websites of phone manufacturers, websites from the Android ecosystem and websites regarding security certifications. Second, information that can be accessed from the smartphone operating system by an application running on the smartphone itself.

After researching which publicly available sources are of interest for us, we use web scraping techniques to access this information automatically and save it in a machine-readable way.

In order to access the information provided by the operating system of a specific smartphone, we developed an app that uses various APIs to gather the information of interest.

1.5 Contribution

Related work already explored the topics of a secure eIDs implementation on smartphones [15] and methods to gather security features of smartphones automatically (see 2.1). In this thesis we identify the smartphone security features needed to evaluate whether smartphones can fulfill the requirements for a mobile eID solution. Existing solutions only automatically gather information about some of those features [16].

Also, existing solutions focus on either a web scraping or an app approach for certain information [16][3].

In this thesis we provide approaches to gather information about more security properties so that all information needed for an evaluation for a mobile eID solution is available. This includes information such as the presence and quality of biometric authentication hardware and the presence and security certification of dedicated hardware for secure key storage. These were previously not automatically gathered by existing approaches. We also describe web scraping solutions and approaches via an app for all features for which this is possible.

1.6 Outline

At first we provide an overview of related work regarding mobile eIDs and also methods of automated Android security information collecting in Section 2. Then, we explain the fundamentals necessary for our work in Section 3. This includes a basic introduction to hardware security (3.1), an overview of the Android Security Architecture (3.2) and a detailed look at the eIDAS regulations in Section 3.5. This is followed up by Section 4 where we work out the specific requirements for a mobile eID solution. How these properties shall be collected is then explained in Section 5 across three subsections. First, we look at publicly available information online, then explain how this information can be automatically gathered using web scraping and lastly explore methods to gather information using an application installed on an Android device. The results of this approach are presented in Section 6.2, after which we draw a conclusion in Section 7 and provide ideas for future work in Section 7.2.

2 Related Work

2.1 Android Device Security Database

The Android Device Security Database (ADSDB) [17] is a research project by the University of Cambridge and the Institute of Networks and Security at Johannes Kepler Universität Linz which gathers information about Android smartphones and collects it in a public database. At the moment it contains information about updates and security patches, device encryption, Strongbox support and pre-granted app permissions [16]

These attributes are collected via different methods. Some attributes like the guaranteed patch availability are gathered from the respective manufacturer's website [16], whereas others are gathered by an app or script as shown in the Presentation on the Android Security Symposium 2020 [3]. The attributes related to preinstalled apps are collected by Uraniborg/Hubble [18]. At the time of writing, the app used by ADSDB has not been published. ADSDB has a similar goal and approach to what we try to achieve with this thesis, but the focus of the information gathered is different. Information about Trusted Execution Environments (TEE), biometrics and security certifications are currently not available in ADSDB. Also, some attributes are only gathered via an app whereas we try to get as much information via web scraping as possible.

2.1.1 UraniBorg/Hubble

Uraniborg and Hubble are part of a repository containing an Android App and various scripts to analyze preinstalled apps, their permissions and security implications before the initial first setup [18]. It then generates a score based on the gathered information using a special metric developed by the Android Security Database Project [19].

2.2 SnoopSnitch

SnoopSnitch is an Android App that tries to detect threats via the mobile network like IMSI catchers, user tracking or malicious over-the-air updates. In a patch in 2018 the developers introduced a new feature that checks the patch level of the Android operating system it is running on, using binary analysis by comparing function signatures to precompiled Android Open Source Project (AOSP) binaries. This is necessary as for one the OS is obviously only available in compiled form on a running smartphone and also many device manufacturers tend to omit certain critical patches or deviate from Google's Patch Date system without being transparent about it. SnoopSnitch then tells the user to which vulnerabilities the found missing patches correspond and to which attacks they might be vulnerable. [20]

2.3 OPTIMOS 2.0

OPTIMOS 2.0 was a project initiated by the German Federal Ministry for Economic Affairs and Energy and led by the Bundesdruckerei. It was completed in November 2020 and developed a non-discriminatory ecosystem for service providers to utilize Secure Element (SE)s and embedded universal integrated circuit cards (eUICCs) [21]. One central use case was to enable the use of mobile eIDs, that comply to the requirements that eIDAS defines for possession-based authentication factors for assurance level substantial [22][23][24]. A number of relevant stakeholders participated in the project, including service providers, OEMs, mobility companies, eID technology providers and universities.

A central concept introduced by OPTIMOS 2.0 is the Trusted Service Manager (TSM), the task of which is to bring together service providers like car-sharing services or tax offices, who require a high level of assurance of authentication when using eIDs on one side and SE or eUICC owners like OEMs or chip manufacturers on the other side [23]. This concept is illustrated in Figure 1.

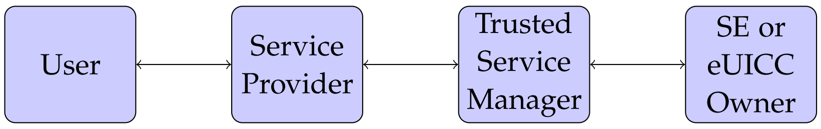

Figure 1: Architecture of the OPTIMOS 2.0 eID ecosystem [23]

It does so by introducing itself as an abstraction layer between those parties(see Figure 1). This means that the TSM provides an API for developers of the service provider, so they can use the eID without taking care of managing different SEs or eUICCs and without having to evaluate whether a certain device fulfills all necessary requirements set by eIDAS [23].

At the time of writing this thesis, only the S20 lineup of Samsung was verified by OPTIMOS 2.0 to fulfill the necessary requirements. This was verified manually by the OPTIMOS 2.0 project. An automatic solution like proposed in this thesis was not used according to the publicly available information [14].

3 Fundamentals

In this Section we will provide on overview of the underlying concepts discussed in this thesis. This divided into a subsection about Hardware Security, Android Security Architecture, Common Criteria and the eIDAS regulation.

3.1 Hardware Security

The main problem with storing cryptographic keys for authentication or encryption, is keeping them safe from attackers. A problem here is that when an attacker even briefly gains access to a device physically or remotely, they could extract such keys and use them maliciously, even when they can no longer access the device the keys have been stored on. This can be mitigated via encryption so a password is needed in order to use a key, but an attacker could still copy a key and try to break the encryption with, for example, a brute-force attack. Ideally, to prevent this, a key would be stored in such a way that no one, not even the legitimate user, can copy the key. [25][26]

This is only possible with hardware-backed security solutions [25][26]. Different approaches for such hardware-backed architectures that offer secure key storage and other security-focused features such as random number generation and key generation are introduced in this section.

3.1.1 Hardware Security Module (HSM)

When designing hardware-backed security solutions an idea that naturally comes to mind is to introduce a dedicated co-processor that handles security-related tasks. Such a security co-processor is usually bundled on one chip with its own memory and called Hardware Security Module(HSM). When an application needs to perform cryptographic operations it hands them off to the HSM, where the keys are stored. The HSM performs all necessary operations like signing or encrypting data, so that the key never leaves the HSM. This effectively prevents key extraction and duplication as not even an attacker with kernel privileges could extract keys. [26]

In order for the key to always stay inside the HSM different requirements need to be fulfilled:

- capability to generate key with sufficient amount of entropy
- support of all needed cryptographic operations (encryption, signing, hashing)
- protection from side-channel attacks
- tamper resistance

[26]

To resist side channel attacks some HSMs are also electronically shielded, to prevent electro-magnetic waves omitted by the HSM to leak information. Others also provide physical tamper resistance, for example they delete all keys when someone damages the enclosure or potting [26].

As HSMs are often used in very critical infrastructure which sometimes is under strict regulation, HSMs usually are certified to standards such as FIPS 140 or Common Criteria [26].

HSMs originated as large deployments in data centers where a large number of cryptographic operation are calculated [26]. But in the last years HSMs in the form of small chips for use in smartphones have also emerged [27][26]. However, there has not yet been agreement on a universally valid definition. For the purpose of this thesis, we use HSM as a broad term for any dedicated chip that offers effective protection of cryptographic keys in order to prevent key extraction and duplication.

3.1.2 Secure Element (SE)

A (embedded) Secure Element (SE) is a tamper-resistant chip designed for cryptographic operations in mobile devices[28][29]. In modern smartphones they are most often used as a hardware security solution for mobile payment solutions in combination with Near Field Communication (NFC) or as replacement for SIM cards [29]. As such they fit the definition of HSM from Section 3.1.1 and will also be referred to as HSM in this thesis.

They provide the same features as smart cards, which are specified by ISO/IEC 7810 [30] and ISO/IEC 7816 [31] and are available in numerous form factors such as UICC, smartSD, smartmicroSD, etc. [28].

SEs run a specialized operating system such as Sm@rtCafé Expert or JCOP that supports the deployment of multiple applets on the SE. The hardware is usually quite simple. Most SEs include an 8-bit microprocessor for the CPU and offload cryptographic operations to a dedicated Numerical Processing Unit where those operations are calculated in hardware [32].

The *Secure Element Configuration* [28] by GlobalPlatform provides a universal specification for Secure Elements. Some SEs also comply to the Java Card specification [33]. Due to the security implications of SEs they are usually certified according to Common Criteria (CC) or FIPS 140-2 [32]. Most SEs are evaluated according to at least CC EAL

4 [29][34] but the evaluation for the SE's operating system also has to be taken into account as it may differ from the evaluation of the hardware [29].

3.1.3 Trusted Execution Environment (TEE)

Another approach to similar problems as an HSM is the Trusted Execution Environment (TEE). In contrast to an HSM, a TEE is not necessarily a dedicated processor but a secure processing environment in which code can be executed and data can be stored persistently. Data and code in the TEE are guaranteed to be authentic and confidential. The TEE resists software and hardware attacks. It relies on a tamper resistant hardware module to established a root of trust that asserts the authenticity of the loaded code in the TEE. [35]

The TEE combined with a dedicated part of memory is usually called trusted area, whereas the main processing environment, where also the rich operating system and user-facing applications reside, is called untrusted area. This is illustrated in Figure 2 [36].

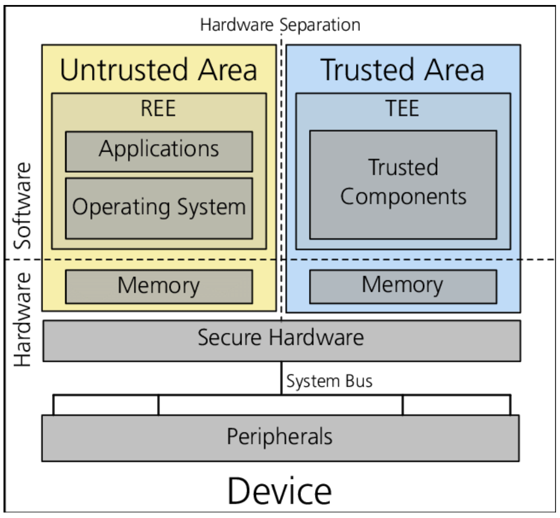

Figure 2: Trusted Execution Environment[36]

The TEE's capabilities are not limited as much as an HSM's and it can execute arbitrary code. This can only be code that is signed by a device manufacturer or user supplied code. Also, as the TEE can (usually) utilize the full computational capacity of the main processor, it can be used for complex tasks that also require a certain level of trust, such as biometric authentication. [37]

The Open Mobile Terminal Platform published a TEE specification called "Advanced Trusted Environment: OMTP TR1" [38] but other definitions also exist [35].

Different chip manufacturers and architectures implement TEEs. Notable examples are Intel SGX [39] and ARM TrustZone [40].

3.1.4 ARM TrustZone

ARM TrustZone is a security architecture marketed by ARM that implements a TEE. The specifications differ for the products Cortex-A and Cortex-M. As Cortex-M is not used for smartphones, we focus on the Cortex-A architecture.

Sandro Pinto and Nuno Santos provide a comprehensive overview about the ARM TrsutZone architecture in their 2019 paper "Demystifying Arm TrustZone: A Comprehensive Survey" [41]. TrustZone utilizes hardware to achieve mutual isolation of the so-called secure world and non-secure world. The *Non-Secure (NS)* processor bit indicates in which state the processor operates. Processes in both worlds may request a world change using a *Secure Monitor Call*. When a change of worlds is requested, another special processing mode called the monitor mode handles the request, saves the state of execution, changes the *NS* bit and sets up the processor with the previous state of the world that is being entered. [41]

Regarding separation of memory TrustZone defines the optional TrustZone Platform Controller (TZPC) and Address-Space Controller (TZASC) which can separate memory regions for the secure and non-secure world. They also provide access control, so that for example the secure world can access memory of the non-secure world, but not vice-versa. [41].

This separation is also enforced at the cache level with a flag in the cache tag which indicates whether data in cache belongs to the secure world or the non-secure world [41].

Past research has shown that there have been several exploits to the ARM TrustZone [42, 43].

3.2 Android Security Architecture

The Android Operating System implements various measures to ensure the security of the device such as app sandboxing, app signing, encryption, access control and verified boot. This Section gives an overview about the features crucial for the usage of a mobile eID.

3.2.1 Trusty TEE

Trusty is a TEE operating system provided by Android. It runs on the main device's processor and is isolated from the the rich operating system by hardware and software [44]. It supports both ARM TrustZone and Intel virtualization technology [44].

It consists of a small kernel derived from Little Kernel [45], a Linux kernel driver to transfer data between the secure and non-secure world and an Android userspace library to communicate with applications in the secure world via the kernel driver [44].

An application running in Trusty is a collection of binary files, a binary manifest and a cryptographic signature. Those applications run unprivileged under the Trusty kernel with their own virtual memory assigned by the TEE memory management. Those applications are scheduled using a round-robin scheduler. [44]

Use cases for the TEE on Android include DRM, mobile payment, secure PIN, and biometric authentication [44].

3.2.2 Android Keystore

The Android Keystore provides a way to securely store both symmetric and asymmetric cryptographic keys securely with either software or hardware security measures, and provides the ability for signatures, verification, generation and import of keys. [46]

Android supports three variants of the Keystore. These are illustrated in Figure 3.

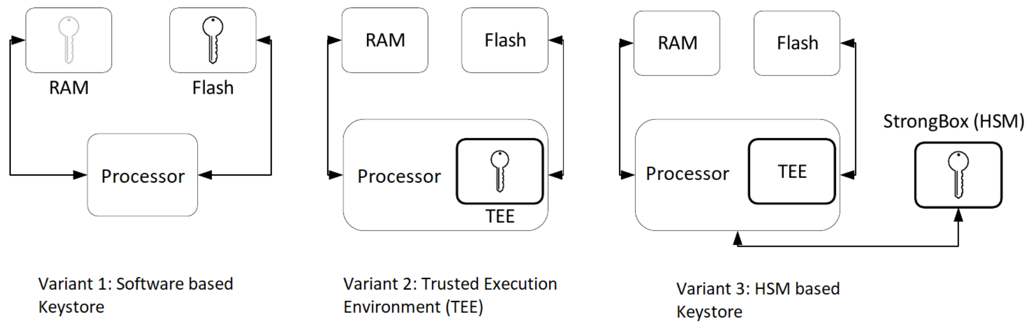
RAM
Flash
Processor
Variant 1: Software based Keystore
RAM
Flash
Processor
TEE
Variant 2: Trusted Execution Environment (TEE)
RAM
Flash
Processor
TEE
StrongBox (HSM)
Variant 3: HSM based Keystore

Figure 3: Variants of Keystores [47]

Variant 1 is only secured by software, meaning that only privileged programs can access the keys but not userspace applications. As this provides only little attack resistance and is rarely used in modern smartphones we will not discuss this further.

Variant 2 uses a TEE to secure the keys which also prevents privileged software from accessing keys.

Variant 3 isolates the keys even better by adding a HSM , which results in protection from cold-boot attacks and speculative side channel attacks like [48]. This was introduced with Android 9 and is called Strongbox. When using the Keystore, app developers can indicate that a key should be stored on an HSM with a specific flag [49].

The Figure 4 on the next page provides an overview of the Keystore architecture when a TEE is utilized. The entities of the architecture are divided into the secure and the non-secure world. The Client, Keymaster HAL and Kernel interface are located in the non-secure world whereas the Keymaster is located in the secure world. [46]

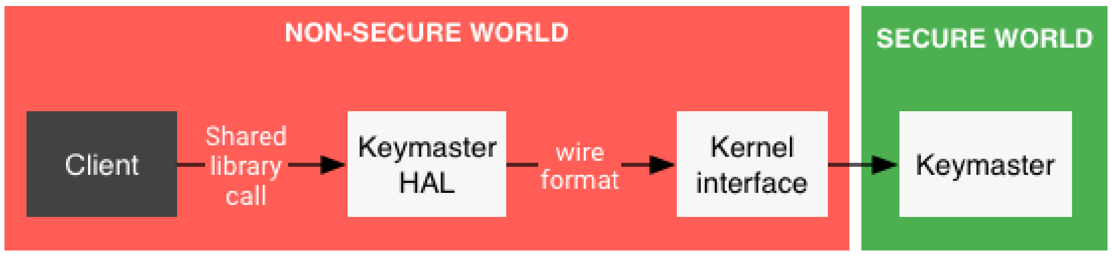

Figure 4: Keystore Architecture [46]

The Client in this figure refers to multiple pieces of software (app, framework, Keystore daemon) that ultimately call the Keymaster HAL. The Keymaster HAL is an abstraction layer for different Keymaster driver implementations and provides a generic, low-level interface for the Client. [46]

When a Client calls the Keymaster HAL, the Keymaster HAL marshals the request for the Keymaster and forwards it to the Keymaster through the Kernel interface. The response from the Keymaster is then unmarshaled by the Keymaster HAL and forwarded to the Client. [46]

The Keystore also includes access control features that can be specified when a key is generated. This means that for example keys can be set to be only usable after a user has authenticated or that the usage of the key could be limited to a specific number of uses each boot. [46]

Additionally key attestation was introduced as a way to strongly determine if an asymmetric key pair is stored on secure hardware and if certain properties are fulfilled. It is also possible to ensure that a key is stored in a Strongbox Keymaster. This is achieved by hardware identifiers stored in the TEE during the manufacturing of the device. These identifiers cannot be altered by the user even when unlocking the bootloader an flashing other firmware. [50]

In order to prevent downgrade attacks where an attacker would try do downgrade the operating system or the TEE firmware and exploit vulnerabilities in earlier versions the Keystore implements version binding. This ensures that a key is bound to a specific operating version. When the operating system is updated all keys are updated and the old keys are invalidated. [46]

3.3 Authentication

As mentioned in the previous section, keys in the Android Keystore can be set to be only usable when a user has authenticated. One example of this is that a user has to authenticate so that the encryption key for the Android device can be used to decrypt files. This is known as the lockscreen. This Authentication requires the Android Keystore implementation and Gatekeeper/Weaver. Authentication can be done with either PIN/password/pattern or biometric. [51]

3.3.1 Gatekeeper/Weaver

Gatekeeper is a Android subsystem that performs password authentication in lockscreens in the TEE [52]. When the correct password is entered, Gatekeeper forwards this information to the Keymaster who then releases the bound key, which can either be an encryption key for the device or a key used by an app [48]. Gatekeeper also restricts further authentication attempts after a set number of failed attempts were registered in order to mitigate brute-force attacks [52]. Weaver provides the same functionality but is implemented in a HSM, and thus provides more security [48].

3.3.2 Authentication process

Figure 5 shows the authentication flow.

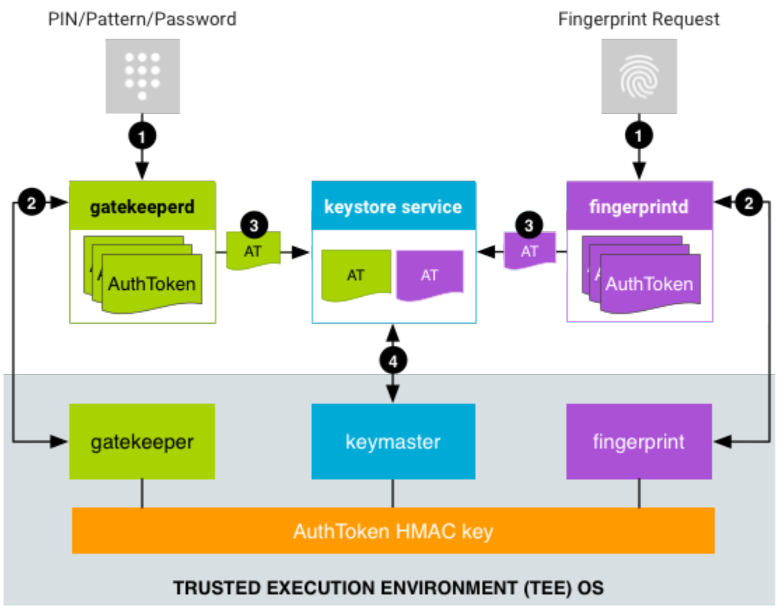

Figure 5: Authentication flow [51]

It consists of the Gatekeeper, the Keymaster and FingerprintService/BiometricPrompt which reside in the TEE and their respective daemons that reside in the non-secure world. Authentication start either when a PIN/password/pattern is entered or when a biometric scan is supplied. These requests are either handles by gatekeeperd for password or a respective biometric daemon (fingerprintd, faced, etc.). The daemon then sends the biometric scan or the hash of the password to its counterpart in the TEE which then checks if the authentication is valid. If the authentication was successful the send a signed AuthToken back to the daemon. The daemon then send the AuthToken to the Keystore service. The Keystore service send the AuthToken to the Keymaster in the TEE where the signature on the AuthToken is verified using a key shared with the Gatekeeper and the biometric TEE service. If the signature is valid, then the requested key from the Keystore may be used.

3.4 Common Criteria (CC)

The *Common Criteria for Information Technology Security Evaluation (Common Criteria or CC)* and the *Common Methodology for Information Technology Security Evaluation (CEM)* are the result of efforts to establish an international standard for evaluation of IT security [53]. The mutual recognition of CC evaluations is ensured via the *Common Criteria Recognition Arrangement (CCRA)* [54], an international agreement between state security agencies such as the German *Federal Office for Information Security* and the US-American *National Security Agency (NSA)*. These state security agencies also take the role of *Certificate Authorizing Schemes* which are responsible for issuing certificates after a successful evaluation process, and creating *Protection Profiles (PP)* under which *Targets of Evaluation (TOE)* are evaluated.

A PP is a set of requirements and objectives for a specific category of TOEs such as databases, operating systems or mobile devices. Those requirements are implementation independent so that they can be reused for a variety of TOEs [55]. When a developer wants to evaluate a TOE they write a *Security Target (ST)* that claims conformance to one or more PPs. The ST also describes the security objectives and requirements of the TOE [53]. The ST then forms the basis of the evaluation of the TOE. The evaluation is done by testing laboratories which are certified by a scheme according to ISO/IEC 17025 [56] [57] [58]. Based on this evaluation the scheme then issues a CC evaluation.

In order to provide a measurement of assurance of the criteria a TOE is evaluated with, the CC defines 7 *Evaluation Assurance Levels (EAL)*[59]. With each higher level the developer has to describe their product and security measures in greater depth and the testing is also done in greater depth [59]. An overview of the EALs is provided in Table 1. The assurance levels can also be augmented by additional components of those that are defined in the *Security assurance components* of the CC. These components include for example life-cycle support, tests or vulnerability assessment [59].

EAL1	functionally tested
EAL2	structurally tested
EAL3	methodically tested and checked
EAL4	methodically designed, tested, and reviewed
EAL5	semiformally designed and tested
EAL6	semiformally verified design and tested
EAL7	formally verified design and tested

Table 1: Evaluation Assurance Levels

3.5 eIDAS

eIDAS is an EU regulation about eIDs and electronic transactions. It specifies electronic signatures, certifications, seals, timestamps and means of authentication in form of eIDs. For the purpose of this thesis the eID is the most relevant part of the eIDAS regulation. The goal of eIDAS is to provide interoperability, transparency and security for digital communication in the European Single Market for the private and public sector. This means that any EU citizen were able to use their eID to prove their identity to another party when filing taxes or use e-Signatures to prove the authenticity of a document.

3.5.1 eID

An eID is an electronic proof of identity similar to an identity card. It is used to identify a natural or legal entity. An eID is issued by an eID system. An eID system can either be operated by a state or a private company [60]. Before eIDAS was established in 2014, a few member states already had eID systems in place, whose respective technological backgrounds differed significantly [60], so bringing all eID systems together into one system was difficult. Instead, the commission decided to make the existing solutions interoperable by introducing a notification system [60]. When an eID system is notified all member states are obligated to accept eIDs of that system with assurance level substantial or high for their electronic public services within 12 months of the announcement of the notification [61].

3.5.2 Notification

Notifications are discussed in the Cooperation Network, a formal body with all member states set in place by the EU Commission with Commission Implementing Regulation (EU) 2015/1501 [62]. Only member states but not private companies can notify an eID system. In order to do so a member state submits a specification of the eID system, information about oversight and responsibility, evidence of the fulfillment of the requirements for the desired assurance level and explanations about interoperability to the Cooperation Network. The Cooperation Network then starts a peer review process, where other member states evaluate the given information based on the Implementing Regulation [63] and the Level of Assurance-Guidance document [64].

The findings of the peer review process are put together in the peer review report which is then discussed in the Cooperation Network. After that, the Cooperation Network compiles an opinion about the assurance level, which is published by the commission. Lastly, the member state notifies the eID system and its assurance level. From a formal point of view the member state alone decides which assurance level the eID system complies to and could ignore the opinion of the Cooperation Network. However, that is highly unlikely because it would defy the idea of consensus in the Cooperation network, and moreover, the member state is liable for the eID system. [60]

3.5.3 Assurance levels

Another step the EU Commission took to address the problem of different, already existing eID solutions was to introduce three assurance levels ("low", "substantial" and "high") for eIDs with eIDAS. Depending on the security measures implemented, an eID solution will be assigned an assurance level during the notification process. For certain applications of an eID an adequate assurance level is mandatory based on the consequences of compromise of that eID. Requirements for each level are defined in Commission Implementing Regulation (EU) 2015/1502 [63] and explained further in the LoA-Guidance [64]. Member states usually also issue regulation on requirements for assurance levels. In Germany the relevant regulations are BSI TR-03107-1 [65] and BSI TR-03159 [13], issued by the Federal Office for Information Security.

4 Requirements for a mobile eID

In the following paragraphs we will explore the relevant technical requirements when considering an eID system on Android smartphones according to eIDAS assurance level substantial. As the focus of this thesis is hardware security features of Android smartphones, we ignore requirements that do not rely on those features such as identifying users per identity card, regulations about letter-post or cryptographic properties like dynamic authentication.

4.1 Authentication Factors

The eIDAS Implementation Act defines three different authentication factors. For the assurance level low, one authentication factor for the electronic identification is sufficient whereas two authentication factors from different categories are needed for the assurance levels substantial and high [63]. Those are defined in the Annex Nr. 1(2) [63] as:

> 'authentication factor' means a factor confirmed as being bound to a person, which falls into any of the following categories:
>
> (a) 'possession-based authentication factor' means an authentication factor where the subject is required to demonstrate possession of it;
>
> (b) 'knowledge-based authentication factor' means an authentication factor where the subject is required to demonstrate knowledge of it;
>
> (c) 'inherent authentication factor' means an authentication factor that is based on a physical attribute of a natural person, and of which the subject is required to demonstrate that they have that physical attribute;

4.1.1 Possession-based factors

Possession-based authentication factors are unique physical tokens that cannot be duplicated and that the owner has sole control over [65][13]. This could be a certificate or cryptographic key stored in secure hardware so that the key cannot be duplicated or extracted. Alternatively SMS verification codes could be used as SMS codes prove possession of the corresponding SIM card. However, SMS verification as a second factor has been criticized in notification peer reviews because SMS is vulnerable to SS7 attacks and attackers could issue a second SIM card using social engineering without knowledge of the victim. Both attacks are used in the field [60]. It would still be possible to use SMS verification as a second factor since a member state can notify an eID regardless of the peer review process. Moreover, the BSI TR-03107 [65] allows for “SMSTan” as a factor for assurance level substantial. However, it would clearly be suboptimal from a security standpoint, and so we shall focus on hardware-backed key storage solutions in this thesis.

Hardware one-time password (OTP) generators could be such a hardware-backed solution, and there are even generators for smartphones available [66]. But potential security benefits notwithstanding, it is both far more complicated and beyond the scope of this thesis, as it requires additional hardware.

As SMS and hardware OTP generators are not suitable for our purposes, the solution for possession-based factors is a cryptographic key stored inside an HSM or a TEE. For such a factor to be suitable for eIDAS it has to withstand extraction attempts from an attacker with moderate attack potential to qualify for the assurance level substantial [13]. Also, the user needs to be able to verify that their token is only used for the intended authentication [65].

4.1.2 Knowledge-based factors

Knowledge based factors are based on information that is only known to the user. This is usually a password or passphrase.

4.1.3 Inherent authentication factors

Inherent authentication factors work by proving that a user has a physical biometric attribute. This could be “fingerprints, palm prints, palm veins, face, hand geometry, iris, etc.” as the LoA Guidance [64] states. We will call this a biometric factor. BSI TR-03107 [65] also states the importance of liveness detection for such a system, and of protection against replay attacks. This cannot be done using remote verification of biometrics, as an attacker could capture the data and replay it to the server, therefore the verification needs to happen locally on the user’s device. Also, when using a biometric factor the chance of triggering an authentication for an attacker by chance or guessing must not be better than the chances with knowledge-based factors. For knowledge-based factors the BSI TR-03107 requires that a numerical PIN that locks the authentication after 3 tries must consist of 4 digits for ’low’ 5 digits for ‘substantial’ and 6 digits for ‘high’. This means the chance is $\frac{3}{10^n}$ with n being the number of digits. This results in 0.03 % for ‘low’ 0.003 % for ‘substantial’ and 0.0003 % for ‘high’. Ergo, the False Acceptance Rate (FAR) of a biometric factor must be correspondingly smaller than 0.03 % for ‘low’ between 0.003 % and 0.0003 % for ‘substantial’ and smaller than 0.000 03 % for ‘high’.

4.2 Two Factor Authentication

When using two factors the BSI TR-03107 [65] requires that they are combined in a way that no one factor can be used by itself so that an attacker cannot compromise the factors individually. In particular an attacker must not be able to tell which factor was responsible for a failed authentication. For example, this can be achieved if the token of a possession-based factor is secured with a password (the knowledge-based factor). [65] The eIDAS Implementation Act requires that the means of authentication can be assumed to be under the sole control of the holder and also requires user interaction when authentication for assurance levels higher than low. This means that at least one factor must be either knowledge based or biometric [65]. As a locally verified biometric factor cannot be combined with a knowledge-based factor so that they cannot be used by itself the TR-03107 concludes that this leaves us with two options for mobile eIDs: [65]

- possession-based factor with locally verified knowledge
- possession-based factor with locally verified biometrics

Both options include a possession-based factor. When ruling out additional hardware this can only be an HSM or a TEE.

4.3 Attack potential

Besides the different factors, for compliance with assurance level substantial, the eIDAS Implementation Act also requires resistance against attackers with moderate attack potential. Definitions for moderate potential can be found in ISO 29115 [67] and ISO 18045 [68] which describe the same standards as the Common Criteria Evaluation Methodology[65][69]. The definitions are rather complex and in the context of eIDAS different member states have their own interpretations. But for the purpose of this thesis we can say that according to BSI TR-03159 a CC evaluation of the HSM with EAL4+[13], '+' meaning augmented by AVA_VAN.4, is needed for resistance against moderate attackers [13]. As an HSM is usually installed in many different phones with different operating systems a so-called composite certification is required that evaluates the HSM in a way that is defined by CC so that it can be considered certified in different systems [70].

TEEs could also be considered to store keys for a possession-based factors but at the moment TEEs are only certifiable up to EAL2 according to the TEE Protection Profile by the GlobalPlatform Device Committee[71]. Thus, HSMs are needed to fulfill the requirement of EAL4+ certified key stores. TEEs of course offer other advantages than HSMs as by their design they can leverage the main processors computation capacity and memory. [72] This can be useful for applications like machine-learning based biometrics or network access; therefore, information about TEEs is also relevant.

Another consideration for resistance against a given attack potential is of course also the state of known security vulnerabilities and corresponding patches of software or firmware. [69] This however is out of scope for this thesis.

4.4 Summary of requirements

In summary, we can say that we need information about the following attributes in Table 2 to evaluate whether a smartphone has the hardware capabilities for a mobile eID solution with assurance level substantial:

Feature	Reason for Relevance
Presence of a TEE	used for possession-based factors
Features the TEE	
Certification of the TEE	needed for resistance against attack potential
Presence of an HSM	used for possession-based factors
Features of the HSM	
Certification of the HSM	resistance against (moderate) attack potential
Presence of Biometric Sensors	can be used for biometric factors
FAR of Biometric Sensors	BSI TR-0310 resistance against attackers manipulating the biometric authentication

Table 2: Smartphone features required for mobile eIDs

5 Approach

As mentioned in the introduction, information about smartphone security features is not readily available. The Android market is very fragmented with multiple device manufacturers, and information on their websites is often very sparse. There are also different websites offering information, but this information is very much scattered across different sources. Also, there is information that can be obtained by using the developer API provided by the Android operating system when installing an app on a device. Our goal is to combine this information and ideally make it available in a database in a machine-readable format.

In order to do this we will first present an overview of publicly available sources of information in this section, select websites for web scraping based on the information they provide and then explain in detail how we want to use web scraping acquire this information. Then we explain how we gather information directly from an Android device by running an app on it. An explain which Android libraries were used to access the specific properties.

5.1 Sources of Information

During our research we were able to identify several sources of information about smartphone security features. An overview about these sources, what information they provide and their credibility will be presented in the following section.

5.1.1 Device Manufacturers

The most obvious place to look for information about smartphone security features would be the websites of the corresponding manufacturers. This section should give an overview of which information is usually available there by presenting 3 manufacturers as an example.

5.1.1.1 Google

The Google Pixel web store has a dedicated category listing security features for every smartphone. For the current phones (Pixel 4a, Pixel 4a 5G, Pixel 5) it lists, the included HSM, called "Google Titan M", and "Google Imprint" meaning the device has a fingerprint sensor. Furthermore the System on a Chip (SoC), the Android version the device was first released with and the current Android version are mentioned. These can be used deduce other properties. [73][74]

5.1.1.2 Samsung

The Samsung website is quite large and complex. Information is scattered across multiple sites: a consumer site, a "Samsung Business" site and the "Samsung Knox" documentation which is Samsung's security program [75].

For example, in the Galaxy S21 phone listing on the consumer site, Samsung writes, it supports "Samsung Knox Vault", which stores biometric data. This describes the Android Keystore and the SE used by Samsung [75].

Samsung also states which phones support a fingerprint sensor and which phones support face unlock [75].

The Samsung Knox documentation [76] lists security features and declares since which version of Knox they have been available on the Knox platform. This includes the presence of an SE, TEE, iris authentication, VPN support and others [77].

There is also a list [78] available specifying which device supports which Knox version. So one can conclude which features are available on a device [78]. This list includes 326 devices [78].

Lastly Samsung also lists all the security certificates [79] they received for their smartphones.

The current Android version is listed on the website that lists security updates [80].

5.1.2 SoC Manufacturers

Smartphone security features like a TEE and sometimes also an HSM are directly depended on the SoC that is used in the smartphone. Therefore, we can conclude that a certain feature is supported by a smartphone if we know what SoC is used. To do this we also need to know which SoC has which features. This information is might be available on chip manufacturers websites.

5.1.2.1 Qualcomm

Qualcomm, the market leader for smartphone SoCs [81], lists security components of their chips in their product brief. This includes information about the presence of a Trusted Execution Environment, a Secure Processing Unit, a fingerprint sensor, and a hardware hypervisor. [82]

As this is first-party information it can be considered reliable.

5.1.2.2 MediaTek

MediaTek, the second-largest smartphone SoC producer [81], does not list any security features such as information about secure co-processors or TEEs, in contrast to its competitor Qualcomm. Only a single press release about a GlobalPlatform TEE certification was available. [83][84]

5.1.3 Other Sources

5.1.3.1 Android CDD

The Android Compatibility Definition Document (Android CDD) is a document released by Google for each new Android Version. It lists all the requirements that must be met in order for a device to be compatible with the corresponding Android Version. It is part of the Android Compatibility Program, which consists of the Android CDD, the AOSP source code and the Compatibility Test Suite (CTS). The CTS is a set of software also released by Google with each Android version and performs tests on an Android device to check its compatibility with the CDD. If a device fulfills the requirements in the CDD and passes the CTS tests, it can then license the Google Mobile Services (GMS), Google's proprietary suite of apps including the Play Store, Maps and others. [85] All devices that have licensed GMS are then listed in Google's List of supported Android devices [86].

This is useful for us as we can conclude that certain features are available if a device is running with a certain version of Android. However, some requirements are only needed for devices that ship with the Android version on release. This includes most that are relevant to us such as the requirement for a CC certification when Strongbox is supported. Requirements with a condition attached to them are marked by the second character in the prefix (1 meaning the requirement has a condition). It is not possible to prove the absence of a feature this way, however.

The Android version a given device was released with is listed on websites like described in 5.1.3.2 or via an app like described in 5.3.2.

The following requirements are of interest when looking for hardware security features:

Android 8 CDD

- " If device implementations support File Based Encryption (FBE), they: " [87]
  - "[C-1-7] MUST be cryptographically bound to a hardware-backed Keystore. " [87]

Android 9 CDD

- " If device implementations support StrongBox, they:" [88]
  - "[C-1-10] MUST include the hardware that is certified against the Secure IC Protection Profile BSI-CC-PP-0084-2014 or evaluated by a nationally

accredited testing laboratory incorporating High attack potential vulnerability assessment according to the Common Criteria Application of Attack Potential to Smartcards." [88]

- "[C-1-11] MUST include the firmware that is evaluated by a nationally accredited testing laboratory incorporating High attack potential vulnerability assessment according to the Common Criteria Application of Attack Potential to Smartcards." [88]

Android 10 CDD

- "[C-0-3] MUST meet the above data storage encryption requirement via implementing File Based Encryption (FBE)." [89]
- "[C-1-4] MUST use Verified Boot [. . .]" [89]
- " Handheld device implementations (* Not applicable for Tablet): " [89]
  - " [9.11/H-0-2]* MUST back up the keystore implementation with an isolated execution environment. " [89]

5.1.3.2 GSMArena

GSMArena is a website which seeks to list all mobile phones on the market and their specifications. Unfortunately, it does not list information about security features like TEEs or HSM but it lists the Android version at launch and the updates available. This information can be used to conclude that according to the Android CDD a CC certification for the Keystore is present. It also lists the SoC and the presence of a Fingerprint Scanner. At the moment of writing, GSMArena lists 9681 unique phones[90]

As this is a website by a private company that lists information about smartphones primarily for consumers and not an official source provided by device manufacturers this information has to be taken with caution. GSMArena states that they gather their information from the websites and press releases of manufacturers. [91]

5.1.3.3 Common Criteria Portal

The Common Criteria Portal is a website of the Common Criteria Recognition Arrangement that provides information about CC in general such as the regulatory publications and technical specifications but also provides a list of all CC certified products including their certification report, security target and where applicable the maintenance report. [34]

This list is available as a CSV file for download. It also lists URLs to the reports in PDF format. [92]

Unfortunately, phones are not listed individually but only the title of the report is listed. Examples are "Samsung Galaxy Devices on Android 10 – Fall" or "Google Pixel

Phones on Android 10". The individual phones that are certified are only listed in the PDF reports.

Despite being hard to search, the information provided is accurate and can be checked as the corresponding certification report is also provided.

5.1.3.4 Android Enterprise Solutions Directory

The Android Enterprise Solutions Directory (AESD) is a website by Google with information about Android devices for enterprises. It offers a search functionality for Android devices with special features that might be interesting to enterprises such as whether the phone has a rugged case, a fingerprint reader or whether it supports Enterprise Mobility Management. Regarding security information, the search function also provides information about the Android version, security update frequency, security update end date and Common Criteria as well as ioXt certifications. [93]

The directory lists 239 smartphones [93]. This information can be considered reliable as the information is gathered from the device manufacturers and verified by Google.

5.1.4 Summary of available sources

The Table 3 below provides an overview of all the websites described in Section 5.1. It lists the information available and how many smartphones are listed.

Property/Website	GSMArena	AESD	CC Portal	G[a]	S[b]
minimum Android version	✓	x	x	✓	x
current Android version	✓	x	x	✓	✓
Fingerprint sensor	✓	x	x	✓	✓
Other Biometric Sensors	x	x	x	x	✓
ioXt certification	x	✓	x	x	x
CC certification	x	✓	✓	x	✓
HSM	x	x	x	✓	✓
Number of Data Points	2000	239	24[c]	3	326

[a] Google
[b] Samsung
[c] 24 entries in the category *Mobility*; 575 additional entries in the category *ICs, Smart Cards and Smart Card-Related Devices and Systems*

Table 3: websites with security information on Android smartphones

5.2 Scraping

After evaluating the presented sources of publicly available information we decided to implement web scrapers for the websites GSMArena, the Android Enterprise Solutions Directory and the Common Criteria Portal as we identified those as the ones with either the most listed devices or crucial information that could not be found otherwise such as information about certificates.

5.2.1 GSMArena

To get the information on gsmarena.com in a machine-readable format the Python framework Scrapy, a web scraper, is used. Relevant attributes from a phone listing are extracted using CSS selectors. To get all the URLs to all the phones, first the file `makers.php3` on `gsmarena.com` is crawled for all the device manufacturers and then each manufacturer listing is crawled for all the phone listings. As we are only interested in Android phones we discarded every listing that reported any OS different from Android. Then for each phone the OS, the chipset and the presence of a Fingerprint reader was extracted. Scrapy supports saving in various formats but for simplicity we choose CSV.

This resulted in a dataset of 2030 entries.

5.2.2 Android Enterprise Solutions Directory

As the AESD website uses javascript to dynamically load phone entries depending on selected properties scraping the information cannot be implemented using simple CSS selectors. However, it is possible to access the API[2] directly and get the information in JSON format.

Using this endpoint we can manipulate the size of the response to our liking and extract all 230 items including their Android version, Fingerprint Reader presence and ioXT or CC certification.

5.2.3 Common Criteria Portal

For the Common Criteria Portal we use a Python script to download the CSV file provided on their website, filter the entries for the *Mobility* and the *ICs, Smart Cards and Smart Card-Related Devices and Systems* categories, which list smartphones and HSMs and then download all Certification Reports, Security Targets, Maintenance Reports, Maintenance ST and the Maintenance dates. This resulted in a dataset of 24 entries for the *Mobility* category and 575 additional entries in the category *ICs,Smart Cards and Smart Card-Related Devices and Systems*.

As mentioned earlier the CSV does not list all the smartphone models for which the certification is applicable so the content of the PDF file has to be searched for model names. This can be done using PDFGrep [94], a command line utility to search text in PDF files. PDFGrep then displays if the supplied string could be found in the PDF files and displays the line in which it was found.

The PDF must then be read by a human to determine if the certification is applicable for the given model.

[2] `https://androidenterprisepartners.withgoogle.com/_ah/spi/search/v1/devices?size=300&sort=createDate:desc,sort_name:asc`

5.3 Android App

After we described our approach to gather data from websites, we now want to take a look at data a physical device can provide us. For this purpose we developed an Android App that interacts with various libraries provided by the Android operating system. It then generates a JSON-formatted text with the security properties that is displayed to the user, saved to a local file and shareable via the Android share functionality in order to send the data to a computer for further analysis. The source code is available in our repository[3].

In the following sections we will present an overview of the libraries Android provides and which property is available with which library.

5.3.1 /proc/cpuinfo

As previously explained in 5.1.2 certain security features like a TEE or an HSM can be derived from the installed SoC. As Android uses the linux kernel we can read the content of `/proc/cpuinfo` to find out which SoC is used.

5.3.2 android.os.SystemProperties

`android.os.SystemProperties` is a AOSP internal class that gives access to the Android system properties store which contains a list of string key-value pairs [95]. As it is an internal class it is not documented in the Android SDK and cannot be imported without tricks. It is used by internal applications, for example when the command `adb shell getprop` is executed. In our App we use a method from the open source project kaltura-device-info[96] that accesses the library using the Java methods `Class.forName`, `Class.getMethod` and `Method.invoke` as a workaround.

The following attributes are gathered by our app:
ro.board.platform: CPU name
ro.product.first_api_level: The API Level the device was first released with. As explained in the Android CDD section other properties can be derived from that information
ro.boot.flash.locked: states whether the bootloader is locked meaning that users can't flash another bootloader or device partitions. [97]
ro.boot.verifiedbootstate: communicates the verified boot state. Can be set to green, yellow, orange, red (eio) and red (no os found) [98]
ro.boot.veritymode dm-verity is part of androids verified boot implementation. It provides integrity checking for block devices. veritymode can be set to "enforcing", "logging" and "eio"[99]
ro.crypto.type returns either 'file' or 'block' whether the device uses block File Based Encryption or Full Disk Encryption[100]

[3]`https://git.imp.fu-berlin.de/besendorf/android-security-scanner/`

5.3.3 android.os.Build

The Android SDK library `android.os.build` provides easy access to information about the operating system. The following attributes are gathered by our app:
Build.MANUFACTURER: The manufacturer of the product/hardware.
Build.MODEL: The end-user-visible name for the end product.
Build.HARDWARE: The name of the hardware (from the kernel command line or `/proc`).
Build.DEVICE: The name of the industrial design.
Build.PRODUCT: The name of the overall product. [101]

5.3.4 Biometrics

Android offers two libraries for biometric authentication: FingerprintManager[102] and BiometricManager[103]. FingerprintManager was deprecated with Android 28 but is also used in our app for compatibility with older devices. To check for presence of a fingerprint sensor FingerprintManager offers `isHardwareDetected()`.

BiometricManager on the other hand also introduced a measurement for biometric security levels. [104]The security levels are "Strong", "Weak", and "convenience"; however, convenience biometric sensors may not be exposed to applications and therefore are not recognizable by us. Depending on whether a device was released with Android 10 or upgraded to it certain Spoof Acceptance Rates (SARs), Imposter Accept Rates (IARs) and FARs are needed to achieve a certain security level. However for the purpose of eIDAS eIDs onyl the FAR is relevant and must be below 0.003 % as explained in 4.1.3. This is met by all security levels as all of them mandate a FAR below 0.000 02 %. [105] To find out which security level is supported by the hardware the method `BiometricManager.canAuthenticate()` can be used. [105] BiometricManager also supports biometric sensors other than fingerprint sensors. The kind of sensor used may can be queried with `PackageManager.hasSystemFeature(FEATURE_FINGERPRINT)` and
`PackageManager.hasSystemFeature(FEATURE_IRIS)`. [106]

5.3.5 PackageManager

The PackageManager class in the Android SDK also offers a method to retrieve hardware capabilities, namely `PackageManager.hasSystemFeature`. It returns information about the features that are also listed in the PackageManager class as constants. The following attributes are gathered by our app:
FEATURE_FINGERPRINT: Presence of a fingerprint sensor
FEATURE_IRIS: Presence of a iris sensor
FEATURE_MANAGED_USERS: Whether the device supports creating secondary users and managed profiles via `DevicePolicyManager`. This is relevant as files of a secondary user are encrypted using a different key from the main user and can therefore be used for compartmentalization. Managed profiles can furthermore be used to set security relevant policies like password complexity, strict permissions for setting the common criteria mode. [107]

FEATURE_SECURE_LOCK_SCREEN: The device has a secure implementation of keyguard, meaning the device supports PIN, pattern and password as defined in the Android CDD
FEATURE_SECURELY_REMOVES_USERS: The device supports secure removal of users. When a user is deleted the data associated with that user is securely deleted and no longer available.
FEATURE_DEVICE_ADMIN: The device supports device policy enforcement via device admins.
FEATURE_STRONGBOX_KEYSTORE: The device has a StrongBox hardware-backed Keystore. [108]

5.3.6 Keystore

As explained earlier the Android Keystore can be implemented solely in software, with a TEE or with an HSM. The Keystore library provides two methods to check or influence how a key is stored. `KeyInfo.isInsideSecureHardware()` was added in API level 23 and can be used after a Key is generated to check whether it is stored in a TEE or HSM but can not differentiate between the two. In our app we generate a dummy key and then use `isInsideSecureHardware()` to check whether a TEE is available. [109]

`setIsStrongBoxBacked()` can be set during key generation to enforce storage in a StrongBox security chip meaning an HSM. If no HSM is available a `StrongBoxUnavailableException` is thrown. In our App we generate a dummy key, use the `setIsStrongBoxBacked()` method and monitor the exception to check whether an HSM is available. [110]

Another option to check for HSM support is the PackageManager library with the `FEATURE_STRONGBOX_KEYSTORE` feature flag. [108]

Addendum: *After our experiments Android added a the Method for API Level S (Android 12) `getSecurityLevel()`, which makes this much easier as it only requires one method that returns whether the Keystore is software backed, utilizes a TEE or an HSM.* [111]

5.3.7 DRM Information

Digital Right Management only works with secure devices. A technique often used by DRM systems is a TEE or HSM to store a key that is used to decrypt media but that can not be extracted from that TEE or HSM, so media can only be accessed in a way defined by the DRM system.

Android supports both the HDCP and Widevine DRM technologies. HDCP comes in different versions whereas Widevine defines different security levels. Information on the HDCP version or Widevine security levels supported by a given device can be accessed using the `MediaDRM` library. In our app we use code from the afore-mentioned open source project kaltura-device-info [96] to access this information. [112]

6 Evaluation

In the last section we presented an overview of both the types of information available online and means to access information on a device. Furthermore we described the methods we implemented for the web scraper and the Android App we developed.

In the following section we will evaluate the methods available to gather information and also compare web scraping to an app approach. We will also discuss the feasibility of evaluating a smartphone for eIDAS eID requirements.

6.1 Experimental Setup

6.1.1 Android App Setup

The Android App was tested on the following devices that were available to us:

Model	Android Version
Oneplus 5	Android 9 (LineageOS)
Google Pixel 2	Android 11 (Stock)
Google Pixel 4a	Android 11 (GrapheneOS)
Nokia 3.4	Android 11 (Stock)
Oppo A72	Android 11 (Stock)

Table 4: Test devices.

Some of the devices used were running custom firmware. This however did not influence the results except for the verified boot status. All devices have been updated to the latest available firmware and security patches. Then our app was installed and run. The reports were extracted via `adb` and then saved in our repository[4].

6.1.2 Web Scraping

For Web Scraping no special setup is needed. The scripts we used can be found in a separate repository[5].

They can be run with any computer with a standard internet connection. The only problem we faced was that gsmarena.com rate-limited our requests and banned our IP address when we sent too many requests. Introducing bigger delays did not fix this issue, so we used a trial account from scraperapi.com, a service that proxies requests to multiple servers to circumvent this problem.

[4]`https://git.imp.fu-berlin.de/besendorf/android-security-scanner/`

[5]`https://git.imp.fu-berlin.de/besendorf/android_security_scraper/`

6.2 Results

Table 5 below provides an overview of all the security properties discussed in this thesis and whether they are obtainable by app or web scraping. The properties needed for an evaluation of the smartphones for eIDAS eIDs are written in bold letters.

Property	Web Scraping	App
TEE	(not implemented)	✓
HSM	(not implemented)	✓
CC certification for HSM	✓*	✓*
CC Certification for Smartphone	✓	x
Biometric Sensors	✓	✓
Biometric Quality (FAR)	x	✓
DRM	x	✓
Minimum Android Version	✓	✓
Current/Installed Android Version	✓	✓
CPU	✓	✓
Managed Users	x	✓
Device Admin	x	✓
Securely Delete Users	x	✓
Secure Lockscreen	x	✓
Flash Locked	x	✓
Verified Boot	✓*	✓
File Based Encryption	✓*	✓

Table 5: overview of detected properties

'✓' means that it is possible to check if the feature is supported we implemented a method to do so and documented it in this thesis.

'x' means that we could not find a method to gather the information automatically.

'✓*' means that feature presence is only possible to deduce from other information, e.g. by checking the minimum Android version and the Android CDD requirements.

Presence of TEE and HSM Usually, device manufacturers do not provide information on whether their device has a TEE or HSM. Two notable exceptions are Google, which advertises HSMs, and Samsung, which advertises TEEs and SEs [73][77]. In general this information is only accessible by checking which SoC is installed and looking up whether that specific SoC supports a TEE or includes an HSM. If the installed SoC supports TEE or includes an HSM, it is still not certain whether the Android Keystore is implemented using the TEE or HSM.

We thus consider scraping to be a suboptimal method to check for the presence of a TEE or HSM, and checking via an app should be preferred, which is done with our approach as described in 5.3.6.

CC certifications CC certifications can either target the whole device, the operating system or certain hardware components in a smartphone like a TEE or HSM. However, the information whether such a certification exists is not available from the Android operating system and thus could not be accessed by our app.

CC certification for smartphones Scraping all CC certifications is possible as shown in 5.2.3, but it is not possible to automatically derive a certain device's certification status as reports are not tagged with the device names. This is especially a problem for smartphone CC certifications as they often list different models in one certification report. Also, the report has to be read to understand how and under which circumstances the certification was done. The Android version is especially important here as any smartphone is only evaluated on a certain Android version, and usually the certification of a device with a new Android version lags behind the release of the latest Android version. For example Android 11 was released on 8th of September 2020 but the CC certification for Google Pixel phones on Android 11 was issued on 8th of February 2021. [113] Another problem is that some smartphones have different versions for different markets and, for example, ship with another SoC which also requires another certification.

CC certification for HSM Scraping CC certifications of HSMs is also possible but even more complicated than for smartphones as, one would first need to know which HSM is used in the smartphone. This can already be difficult as sometimes an HSM is included in the SoC, like the Qualcomm SPU [114], and sometimes a dedicated chip is built into a smartphone as Google does with its Titan M chip that is used in their Pixel phones.

A particularly complex situation exists with devices like the Samsung S20, which uses the Qualcomm Snapdragon 865 that includes the Qualcomm SPU in its SoC but uses a dedicated chip, called S3FV9RR, that is certificated with CC EAL6+ whereas the Qualcomm SPU is only certified according to EAL-4+ [115][116].

Because of this rather complex situation we did not focus on scraping certification for HSMs in our experiments and rather used the fact that according to the Android CDD, devices, that support Strongbox are required to use an HSM that has been certified according to CC. To make use of this fact we need to know the Android version a certain device was released with. This is done with our web scraping approach as explained in 5.1.3.2 and with our app approach by reading the SystemProperty `first_api_level` as described in 5.3.2.

Biometric Quality Scraping During research we could not find any resources on the quality of biometric sensors online, so checking via app is necessary.

DRM information During research we could not find any resources on the DRM systems that are supported by devices so checking via app is necessary.

System Properties Lines 11–17 are all stored in the Android System Properties Store and can be accessed with an app. However during research we could not find any resources online listing the system properties of a device, so checking via app is necessary.

Considerations Web Scraping vs App The two approaches we chose for gathering information have different advantages and shortcomings. Web scraping is easily scalable to thousands of devices but the information might not be very reliable depending on the scraped websites. The app approach, however, yields more reliable results, but one needs to possess the smartphone for evaluation, which results in high costs when evaluating many phones.

7 Conclusion

The goal of this thesis was to find out which smartphone security features are required to evaluate compliance to regulatory requirements for a mobile eID solution, to list publicly available sources of this information and to research ways to automatically aggregate it. After doing so, we may conclude that it is generally possible and feasible to gather most of the information about required features. We did encounter a number of limitations, however.

First of all, we could show that most of the collected attributes are only accessible with an app rather than by web-scraping public resources.

One exception is security certification for smartphones and for HSMs, whose certification can only be indirectly concluded via the first api level and the Android CDD. The presence of a certification for a smartphone cannot be found by any means with only an app and can only be retrieved online. All certification reports are only available online and not directly from an Android device. Those certifications are not tagged with the device name and often include multiple devices, so it is hard to tell automatically if a certain device is certificated or not. Also, usually one does not know what kind of hardware is used for the Keystore implementation as sometimes additional secure co-processors are built into a device or the implementation could simply not leverage included hardware, and so one can not be sure if it is certified or not.

Also, with the methods presented we were not able to gather more detailed information about a TEE or HSM including the supported algorithms, the version of the secure OS of a TEE or a list of features of a TEE and HSM. This was mainly because of time constraints so we cannot estimate if this is theoretically feasible. However, it would be possible to look into this in future work.

7.1 Conclusion regarding requirements for a mobile eID

One motivation for this thesis was to evaluate if it is possible to gather enough information about Android devices to be able to determine whether they offer enough security features to fulfill the requirements for a mobile eID solution according to eIDAS assurance level substantial. It should be noted again, that known security vulnerabilities and security patches are also relevant for a mobile eID solution but out of scope of this thesis.

Table 5 in 6.2 shows the properties we identified to be requirements for a mobile eID solution and compares them to our results from web scraping and our app approach. Consistent with our general conclusion, the app approach yields more results. It provides all needed information with the previously mentioned exception that a security certification for the HSM can only be deduced via the first api level and the Android CDD.

By means of web scraping the information whether a biometric sensor is available is accessible. However, the quality of the biometric sensor is only accessible via the app. This is important information as the quality of the biometric sensor is a requirement for inherent factors as written in 4.1.3. There we also identified that the FAR of a biometric authentication must be lower than 0.003 % to qualify for assurance level substantial.

Information about the presence of an HSM was not available via web scraping as this type of information is very sparse. The only sources stating that a smartphone comes with an HSM are press releases [115][116] or the product pages for Google's Pixel phones [73]. As this information is a critical requirement for eIDAS eID solutions the web scraping approach is clearly suboptimal.

As the presence of a CC certification can only be concluded from the first api level and the Android CDD we cannot be certain whether a specific device fulfills all requirements because we can only assume that a certification is present but cannot access the Certification Report. This is not enough to evaluate a device regarding its compliance to eIDAS eID systems and we have to conclude that our approach is not able to gather automatically all information needed for a mobile eID solution with assurance level substantial. Solutions to this problem are discussed in the following section.

7.2 Future Work

To tackle the problems encountered during our research different ideas come to mind. This includes widening the scope and exploring other means to acquire information but also measures that could be taken by manufacturers, the Android Open Source Project or the Common Criteria Portal to improve the availability of information.

It would be possible to do more research about methods to acquire detailed information about the TEE and HSM in a device. This could be done by running code on a device and may require root privileges or flashing a modified firmware to the device. Another approach could be to investigate the source code of the official firmware of a device and look for calls to the secure hardware. This might reveal information about its capabilities and features. The source code for the secure hardware components, however, is usually not available [117] as it is usually delivered in binary blobs. This does not mean that an analysis is impossible, but it is definitely harder than if the source code would be available.

Something that was already discussed by Google is to provide the possibility for App developers to check the certification status of the secure hardware on an Android device. This should be easy to implement for Google as they already check for certifications during the Android CDD procedure. However, the commit[118] that implemented this was not merged into the Android codebase. This should be made up for with a coming version of Android as it is a crucial feature for mobile driver licenses or eIDAS eIDs.

In 2019 Google planned to include such a feature in the IdentityCredential API that was developed to support the storage of driver licenses on mobile phones. A commit that describes the `SecurityHardwareType` Interface is available on xda-developers.com [118].

It proposes three constants for the evaluation of the certification for the IdentityStore implementation. `SECURITY_CERTIFICATION_UNCERTIFIED`, `SECURITY_CERTIFICATION_CC_EAL_4_PLUS`, `SECURITY_CERTIFICATION_CC_EAL_5_PLUS` and `SECURITY_CERTIFICATION_CC_EAL_6_PLUS`. Hence, it would be a good solution for us, but the commit has not been included in the android source code as of the time this thesis was written. [119] The CredentialStore is already being worked on as part of the work for Android 12. Certain functions also mention `HardwareInformation` [120].

Another issue regarding certifications is the machine-readability of common criteria evaluations. The common criteria portal already provides all CC evaluations and offers a machine-readable CSV file that lists the name of the evaluation, the manufacturer of the devices, the assurance levels, protection profiles, certification date, maintenance date and links to the certification report, security target, maintenance report and maintenance security target. This should be expanded to include a list of all device names that a certification is applicable for and under with circumstances the certification is valid, such as the OS version. This would enable an automatic evaluation of a device's certification status.

Another way of solving this problem would be from the manufacturer's side. They should include more information about security properties of smartphones in general but especially more information about the Android Keystore implementation as this information is very sparse at the moment. The manufacturers should state whether the Keystore is backed only by software, by a TEE or by an HSM. Furthermore, they should state if the TEE and the HSM are certified under CC or similar certificates and link the corresponding certification reports on their websites. This would make web scraping for this information possible.

Lastly, a solution to the described problems could also be to lower the requirements for eIDAS compliance. Especially lifting the requirement for a security certification for the HSM comes to mind as it is the only property we could not acquire with the described methods and it can be assumed that one is present if the device shipped with at least Android 9 and offers Strongbox support. If the Bundesamt für Sicherheit in der Informationstechnik (BSI) were to decide an Android device that released with at least Android 9, is sufficient for the requirement of resistance against attackers with moderate attack potential according to eIDAS, this would enable the notification of an eID system that relies on the methods described in this thesis to evaluate the security properties of smartphones.

Acronyms

AESD Android Enterprise Solutions Directory. 25

Android CDD Android Compatibility Definition Document. 23, 24, 32, 33, 35, 36, *Glossary:* Android Compatibility Definition Document

AOSP Android Open Source Project. 8, 23, 27

BSI Bundesamt für Sicherheit in der Informationstechnik. 6, 16, 18, 35, 36, *Glossary:* Bundesamt für Sicherheit in der Informationstechnik

CC Common Criteria; see also: 3.4. 10, 16, 20, 23–25, 31, 32, 34, 35, 37

eID electronic identification. 1, 5–8, 12, 16–18, 20, 21, 28, 30, 31, 33–38, *Glossary:* electronic identification

eIDAS EU regulation on electronic identification and trust services for electronic transactions. 1, 5–9, 16–20, 28, 30, 31, 33–35, 37

eUICC embedded universal integrated circuit card. 8, 36, *Glossary:* embedded universal integrated circuit card

FAR False Acceptance Rate. 19, 21, 28, 34, 36, *Glossary:* False Acceptance Rate

FBE File Based Encryption. 23

HSM Hardware Security Module; see also: 3.1.1. 9–11, 13, 15, 19–22, 24–27, 29, 31–35

IAR Imposter Accept Rate. 28, 36, *Glossary:* Imposter Accept Rate

NFC Near Field Communication. 10

nPA neuer Personalausweis. 36, *Glossary:* neuer Personalausweis

SAR Spoof Acceptance Rate. 28, 36, *Glossary:* Spoof Acceptance Rate

SE Secure Element; see also: 3.1.2. 8, 10, 22, 31

SoC System on a Chip. 22, 27, 31, 32, 36, *Glossary:* System on a Chip

TEE Trusted Execution Environment; see also: 3.1.3. 7, 11–15, 19–24, 27, 29, 31–35

Glossary

Android Compatibility Definition Document List of requirements for a device to be compatible wioth a certain Adnroid version; See: 5.1.3.1. 23, 36

assurance level assurance level of an eID according to eIDAS; see also: 3.5.3. 6, 8, 17–20, 33–35

AVA_VAN.4 Vulnerability Assessment - Methodical vulnerability analysis; see : [59]. 20

Bundesamt für Sicherheit in der Informationstechnik German Federal Office for Information Security. 35, 36

common criteria mode a setting for the Android OS that sets various other security settings, like mandatory key zeroization, blocking bootloader download mode and others in order to comply with CC regulations[121]. 28

electronic identification electronic identification as defined by eIDAS, used for secure authentication. 5, 36

embedded universal integrated circuit card successor of the SIM card with security features. 8, 36

False Acceptance Rate percentage of authentication attempts that are incorrectly accepted when the input is randomly chosen. 19, 36

Imposter Accept Rate percentage of authentication attempts that are incorrectly accepted when the input tries to mimic a know good sample. 28, 36

neuer Personalausweis the new german identity card that also includes an eID. 36

Notification Notification according to eIDAS; see also: 3.5.2. 5

Scrapy is an open-source framework to extract data from websites; See also: `https://scrapy.org/`. 26

Spoof Acceptance Rate percentage of authentication attempts that are incorrectly accepted when an attacker replays a previously recorded genuine sample. 28, 36

System on a Chip A chip that integrates most parts needed for a functioning computer including CPU, memory, I/O port and sometimes a modem. 22, 36

List of Figures

List of Tables

References

[1] *statcounter - GlobalStats*. URL: `https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet`. (accessed: 10.10.2020).

[2] Jacob Poushter et al. "Smartphone ownership and internet usage continues to climb in emerging economies". In: *Pew research center* 22.1 (2016), pp. 1–44.

[3] René Mayrhoferr Daniel R. Thomas Alastair R. Beresford. *Towards a Transparent Database of Android Device Security Attributes*. `https://www.youtube.com/watch?v=zxkbyyl-9b8&t=865s` and `https://www.android-device-security.org/publications/AndroidSecuritySymposium-Intro-20200706.pdf`. July 2020. (accessed: 13.09.2020).

[4] *REGULATION (EU) No 910/2014*. URL: `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910&from=DE`. (accessed: 10.10.2020).

[5] *OPTIMOS 2.0*. [Online; accessed 4. Mar. 2021]. Mar. 2021. URL: `https://www.digitale-technologien.de/DT/Redaktion/DE/Standardartikel/SmartServiceWeltProjekte/Wohnen_Leben/SSWII_Projekt_OPTIMOS_20.html`.

[6] *German eID based on Extended Access Control v2*. [Online; accessed 7. Apr. 2021]. URL: `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/EIDAS/German_eID_Whitepaper.pdf;jsessionid=69B81B5A74224B4C569A4FC65CDA92DE.internet472?__blob=publicationFile&v=1`.

[7] Stefan Krempl. "Gesetzentwurf: Online-Ausweis soll aufs Handy, wird aber teuer". In: *heise online* (Feb. 2021). URL: `https://www.heise.de/news/Gesetzentwurf-Online-Ausweis-soll-aufs-Handy-wird-aber-teuer-5049183.html`.

[8] Matthew Green Maximilian Zinkus Tushar M. Jois. "Data Security on Mobile Devices: Current State of theArt, Open Problems, and Proposed Solutions". In: *ResearchGate* (Jan. 2021). URL: https://securephones.io/main.pdf.

[9] Mohd Shahdi Ahmad et al. "Comparison between android and iOS Operating System in terms of security". In: *2013 8th International Conference on Information Technology in Asia (CITA)* (July 2013), pp. 1–4. DOI: 10.1109/CITA.2013.6637558. URL: https://ieeexplore.ieee.org/abstract/document/6637558.

[10] *Mobile Operating System Market Share Worldwide | StatCounter Global Stats*. [Online; accessed 8. Mar. 2021]. Mar. 2021. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[11] *Secure an Android Device | Android Open Source Project*. [Online; accessed 8. Apr. 2021]. Sept. 2020. URL: https://source.android.com/security.

[12] *Getting Started with iOS App Development | AWS*. [Online; accessed 8. Apr. 2021]. Mar. 2021. URL: https://aws.amazon.com/de/mobile/mobile-application-development/native/ios.

[13] *Technical Guideline TR-03159 Mobile Identities*. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03159/tr-03159.html. (accessed: 30.10.2020).

[14] *Samsung, BSI, Bundesdruckerei und Telekom Security bringen gemeinsam Personalausweis aufs Smartphone*. [Online; accessed 17. Mar. 2021]. Mar. 2021. URL: https://news.samsung.com/de/sicher-und-einfach-identifizieren-mit-dem-smartphone.

[15] Tim Ohlendorf, Wolfgang Studier, and Marian Margraf. "Digitale Identitäten auf dem Smartphone". In: *Datenschutz und Datensicherheit-DuD* 43.1 (2019), pp. 17–22.

[16] *Android Security Database Attributes*. URL: https://www.android-device-security.org/attributes/. (accessed: 07.09.2020).

[17] *Android Device Security Database*. URL: https://www.android-device-security.org/. (accessed: 21.10.2020).

[18] *Uraniborg*. URL: https://github.com/android/security-certification-resources/tree/master/ioXt/uraniborg. (accessed: 07.09.2020).

[19] Billy Lau et al. *Uraniborg's Device Preloaded App Risks Scoring Metrics*. Aug. 2020. URL: https://ins.jku.at/publications/2020/Lau_2020_Uraniborg_Scoring_Whitepaper_20200827.pdf. (accessed: 13.09.2020).

[20] Karsten Nohl and Jakob Lell. *Mind the Gap: Uncovering the Android Patch Gap Through Binary-Only Patch Level Analysis*. URL: https://conference.hitb.org/hitbsecconf2018ams/sessions/mind-the-gap-uncovering-the-android-patch-gap-through-binary-only-patch-level-analysis/. (accessed: 07.09.2020).

[21] *OPTIMOS 2.0 Ein sicheres mobiles Ökosystem*. [Online; accessed 16. Apr. 2021]. URL: https://omnisecure.berlin/wp-content/uploads/os20_Ku%CC%88gler_Dennis-1.pdf.

[22] *Projekt OPTIMOS: Mit Smartphones können ID-Systeme mit Sicherheitsniveau "substantiell" erreicht werden, Bundesdruckerei GmbH, Pressemitteilung - PresseBox.* [Online; accessed 17. Mar. 2021]. Mar. 2021. URL: `https://www.pressebox.de/inaktiv/bundesdruckerei-gmbh/Gesundheits-und-Datenschutz-fuer-die-meisten-Deutschen-gleich-wichtig/boxid/1037390`.

[23] Ulrike Stopka, Gertraud Schäfer, and Andreas Kreisel. "Business and Billing Models for Mobile Services Using Secure Identities". In: *HCI in Mobility, Transport, and Automotive Systems*. Cham, Switzerland: Springer, June 2019, pp. 459–476. ISBN: 978-3-030-22665-7. DOI: `10.1007/978-3-030-22666-4_33`. URL: `https://link.springer.com/chapter/10.1007%2F978-3-030-22666-4_33`.

[24] *Projekte*. [Online; accessed 12. Apr. 2021]. Apr. 2021. URL: `https://www.digitale-technologien.de/DT/Redaktion/DE/Standardartikel/SmartServiceWeltProjekte/Wohnen_Leben/SSWII_Projekt_OPTIMOS_20.html`.

[25] Roger R Dube. *Hardware-based computer security techniques to defeat hackers: From biometrics to quantum cryptography*. eng. 1st ed. Hoboken: WILEY, 2008. ISBN: 0470193395.

[26] Dirk Fox. "Hardware Security Module (HSM)". In: *Datenschutz und Datensicherheit - DuD* 33.9 (Sept. 2009), p. 564. ISSN: 1862-2607. DOI: `10.1007/s11623-009-0145-9`. URL: `https://doi.org/10.1007/s11623-009-0145-9`.

[27] Xiaowen Xin. "Titan M makes Pixel 3 our most secure phone yet". In: *Google* (Oct. 2018). URL: `https://www.blog.google/products/pixel/titan-m-makes-pixel-3-our-most-secure-phone-yet`.

[28] *Secure Element Configuration v2.0 | GPC_GUI_049 - GlobalPlatform*. [Online; accessed 15. Apr. 2021]. May 2019. URL: `https://globalplatform.org/specs-library/secure-element-configuration-v2`.

[29] Martin Schröder. "Sichere Bereitstellung von Identitätstoken auf mobilen Endgeräten". 2012, p. 107. URL: `https://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2013-01/SAR-PR-2013-01_.pdf`.

[30] *ISO/IEC 7810:2019*. [Online; accessed 15. Apr. 2021]. Apr. 2021. URL: `https://www.iso.org/standard/70483.html`.

[31] *ISO/IEC 7816-1:2011*. [Online; accessed 15. Apr. 2021]. Apr. 2021. URL: `https://www.iso.org/standard/54089.html`.

[32] Wolfgang Rankl. *Handbuch der Chipkarten : Aufbau, Funktionsweise, Einsatz von Smart Cards / Wolfgang Rankl ; Wolfgang Effing*. ger. München, 2008.

[33] *Java Card Overview*. [Online; accessed 16. Apr. 2021]. Apr. 2021. URL: `https://www.oracle.com/java/technologies/java-card-tech.html`.

[34] *Common Criteria Certified Products*. URL: `https://www.commoncriteriaportal.org/products/`. (accessed: 05.01.2021).

[35] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. "Trusted execution environment: what it is, and what it is not". In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE. 2015, pp. 57–64.

[36] Javier González. "Operating System Support for Run-Time Security with a Trusted Execution Environment". PhD thesis. Mar. 2015. DOI: 10.13140/RG.2.1.4827.8161.

[37] OMTP. *TEEBiometric System PP-ModuleVersion 0.0.0.11.* URL: http://globalplatform.org/wp-content/uploads/2018/12/PP-MODULE-TEE-BIOMETRIC-SYSTEM-v0.0.0.11_20181128.pdf. (accessed: 03.12.2020).

[38] GlobalPlatform Technology. *ADVANCED TRUSTED ENVIRONMENT: OMTP TR1 v1.1.* URL: https://www.gsma.com/newsroom/wp-content/uploads/2012/03/omtpadvancedtrustedenvironmentomtptr1v11.pdf. (accessed: 03.12.2020).

[39] *Intel® SGX for Dummies (Intel® SGX Design Objectives).* [Online; accessed 13. Apr. 2021]. Sept. 2020. URL: https://software.intel.com/content/www/us/en/develop/blogs/protecting-application-secrets-with-intel-sgx.html.

[40] Sandro Pinto and Nuno Santos. "Demystifying Arm TrustZone: A Comprehensive Survey". In: *ACM Computing Surveys* 51.6 (Jan. 2019), pp. 1–36. ISSN: 0360-0300. DOI: 10.1145/3291047. URL: https://dl.acm.org/doi/abs/10.1145/3291047.

[41] Sandro Pinto and Nuno Santos. "Demystifying Arm TrustZone: A Comprehensive Survey". In: *ACM Comput. Surv.* 51.6 (Jan. 2019). ISSN: 0360-0300. DOI: 10.1145/3291047. URL: https://doi.org/10.1145/3291047.

[42] *Full TrustZone exploit for MSM8974.* URL: https://bits-please.blogspot.com/2015/08/full-trustzone-exploit-for-msm8974.html. (accessed: 03.12.2020).

[43] Di Shen. *An Exploration of ARM TrustZone Technology.* URL: https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android.pdf. (accessed: 03.12.2020).

[44] *Trusty TEE | Android Open Source Project.* [Online; accessed 26. Apr. 2021]. Sept. 2020. URL: https://source.android.com/security/trusty.

[45] *littlekernel.* [Online; accessed 27. Apr. 2021]. Apr. 2021. URL: https://github.com/littlekernel/lk.

[46] *Hardware-backed Keystore | Android Open Source Project.* [Online; accessed 27. Apr. 2021]. Sept. 2020. URL: https://source.android.com/security/keystore.

[47] *BMG MOVI.* [Online; accessed 29. Apr. 2021]. Mar. 2021. URL: https://movi.fokus.fraunhofer.de/androidSecurityFeatures.

[48] René Mayrhofer et al. "The Android Platform Security Model". In: *ArXiv e-prints* (Apr. 2019). eprint: 1904.05572. URL: https://arxiv.org/abs/1904.05572v2.

[49] *KeyGenParameterSpec.Builder | Android Developers.* [Online; accessed 29. Apr. 2021]. Mar. 2021. URL: https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.Builder#setIsStrongBoxBacked(boolean).

[50] *Key and ID Attestation | Android Open Source Project.* [Online; accessed 28. Apr. 2021]. Sept. 2020. URL: https://source.android.com/security/keystore/attestation.

[51] *Authentication | Android Open Source Project*. [Online; accessed 3. May 2021]. Sept. 2020. URL: https://source.android.com/security/authentication.

[52] *Gatekeeper | Android Open Source Project*. [Online; accessed 3. May 2021]. Sept. 2020. URL: https://source.android.com/security/authentication/gatekeeper.

[53] *Common Criteria An Introduction*. [Online; accessed 16. Apr. 2021]. URL: http://www.ia.nato.int/Documents/CC-Intro.pdf.

[54] *Arrangement on the Recognition of Common Criteria Certificates In the field of Information Technology Security*. [Online; accessed 16. Apr. 2021]. July 2014. URL: https://www.commoncriteriaportal.org/files/CCRA%20-%20July%202,%202014%20-%20Ratified%20September%208%202014.pdf.

[55] *Common Criteriafor Information TechnologySecurity Evaluation*. [Online; accessed 16. Apr. 2021]. URL: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf.

[56] *ISO/IEC 17025:2017*. [Online; accessed 19. Apr. 2021]. Apr. 2021. URL: https://webstore.ansi.org/Standards/ISO/ISOIEC170252017.

[57] Stefan Luber. "Was ist Common Criteria?" In: *Security-Insider* (Mar. 2019). URL: https://www.security-insider.de/was-ist-common-criteria-a-677228.

[58] *Licensed Laboratories : CC Portal*. [Online; accessed 19. Apr. 2021]. Apr. 2021. URL: https://www.commoncriteriaportal.org/labs.

[59] *Common Criteriafor Information TechnologySecurity Evaluation*. [Online; accessed 16. Apr. 2021]. URL: https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf.

[60] Felix Bleckmann et al. "Notifizierung von eID-Systemen gemäß eIDAS". In: *Datenschutz und Datensicherheit - DuD* 43.4 (Apr. 2019), pp. 189–193. ISSN: 1862-2607. DOI: 10.1007/s11623-019-1090-x. URL: https://doi.org/10.1007/s11623-019-1090-x.

[61] Governikus KG. *Leitfaden für die öffentliche Verwaltung*. URL: https://www.personalausweisportal.de/SharedDocs/downloads/Webs/PA/DE/informationsmaterial/weiterefuehrendes-material/Leitfaden_eIDAS_Verordnung.pdf?__blob=publicationFile&v=3.

[62] *Commission Implementing Regulation (EU) 2015/1501*. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:JOL_2015_235_R_0001. (accessed: 26.11.2020).

[63] *Commission Implementing Regulation (EU) 2015/1502*. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:JOL_2015_235_R_0002. (accessed: 25.11.2020).

[64] *Guidance for the application of the levels of assurance which support the eIDAS Regulation*. URL: https://ec.europa.eu/cefdigital/wiki/display/EIDCOMMUNITY/Guidance+documents?preview=/40044784/40044786/Guidance%20on%20Levels%20of%20Assurance.docx. (accessed: 26.11.2020).

[65] *Technische Richtlinie TR-03107-1 Elektronische Identitäten und Vertrauensdienste im E-Government*. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03107/TR-03107-1.pdf?__blob=publicationFile&v=4. (accessed: 30.10.2020).

[66] *YubiKey 5C NFC Shop*. URL: https://www.yubico.com/de/product/yubikey-5c-nfc/. (accessed: 30.10.2020).

[67] *ISO/IEC 29115:2013*. [Online; accessed 30. Apr. 2021]. Apr. 2021. URL: https://www.iso.org/standard/45138.html.

[68] *ISO/IEC 18045:2008*. [Online; accessed 30. Apr. 2021]. Apr. 2021. URL: https://www.iso.org/standard/46412.html.

[69] *Common Methodologyfor Information TechnologySecurity Evaluation*. URL: https://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R5.pdf. (accessed: 03.12.2020).

[70] Dr. Dennis Kügler. *Identities Go Mobile - The Future of Electronic Identification*. [Online; accessed 22. Feb. 2021]. Jan. 2020. URL: https://omnisecure.berlin/wp-content/uploads/os20_Ku%CC%88gler_Dennis-1.pdf.

[71] GlobalPlatform Device Committee. *TEEProtection Profile*. [Online; accessed 4. Mar. 2021]. Nov. 2014. URL: https://www.commoncriteriaportal.org/files/ppfiles/anssi-profil_PP-2014_01.pdf.

[72] Carlton Shepherd et al. "Secure and Trusted Execution: Past, Present and Future – A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems". In: *ResearchGate* (Aug. 2016). DOI: 10.1109/TrustCom.2016.0060. URL: https://www.researchgate.net/publication/306039236_Secure_and_Trusted_Execution_Past_Present_and_Future_--_A_Critical_Review_in_the_Context_of_the_Internet_of_Things_and_Cyber-Physical_Systems.

[73] *Google Store Pixel 5*. URL: https://store.google.com/product/pixel_5_specs. (accessed: 05.01.2021).

[74] *Google Pixel Handy – Google Store*. [Online; accessed 7. Apr. 2021]. Apr. 2021. URL: https://store.google.com/de/magazine/compare_pixel.

[75] *Samsung Galaxy S21 Ultra 5G | Samsung DE*. [Online; accessed 7. Apr. 2021]. Mar. 2021. URL: https://www.samsung.com/de/smartphones/galaxy-s21-ultra-5g.

[76] *Knox features on Android | Samsung Knox*. [Online; accessed 1. May 2021]. May 2021. URL: https://www.samsungknox.com/en/knox-features/android.

[77] *Knox features on Android | Samsung Knox*. [Online; accessed 7. Apr. 2021]. Apr. 2021. URL: https://www.samsungknox.com/en/knox-features/android.

[78] *Geräte secured by Knox | Samsung Knox*. [Online; accessed 7. Apr. 2021]. Apr. 2021. URL: https://www.samsungknox.com/de/knox-platform/supported-devices.

[79] *Knox-Zertifizierungen | Samsung Knox*. [Online; accessed 1. May 2021]. May 2021. URL: https://www.samsungknox.com/de/knox-platform/knox-certifications.

[80] *Security Updates Firmware Updates | Samsung Mobile Security*. [Online; accessed 23. Apr. 2021]. Apr. 2021. URL: https://security.samsungmobile.com/securityUpdate.smsb.

[81] *Samsung became the third largest smartphone SoC vendor globally in 2019*. URL: https://www.xda-developers.com/samsung-became-the-third-largest-smartphone-soc-vendor-globally-in-2019/. (accessed: 05.01.2021).

[82] *Qualcomm® Snapdragon™888 5G Mobile Platform*. URL: https://www.qualcomm.com/media/documents/files/qualcomm-snapdragon-888-mobile-platform-product-brief.pdf. (accessed: 05.01.2021).

[83] *MediaTek MT6737T granted GlobalPlatform TEE Security Evaluation Product Certificate*. URL: https://www.mediatek.com/blog/mediatek-mt6737t-granted-globalplatform-tee-security-evaluation-product-certificate. (accessed: 05.01.2021).

[84] *Homepage*. [Online; accessed 24. Mar. 2021]. Mar. 2021. URL: https://www.mediatek.com.

[85] *Android 11 Compatibility Definition | Android Open Source Project*. [Online; accessed 30. Mar. 2021]. Jan. 2021. URL: https://source.android.com/compatibility/android-cdd.

[86] Google. *List of supported Android devices*. URL: https://storage.googleapis.com/play_public/supported_devices.html. (accessed: 18.12.2020).

[87] *Android 8.0 Compatibility Definition | Android Open Source Project*. [Online; accessed 31. Mar. 2021]. Sept. 2020. URL: https://source.android.com/compatibility/8.0/android-8.0-cdd.

[88] *Android 9 Compatibility Definition | Android Open Source Project*. [Online; accessed 30. Mar. 2021]. Sept. 2020. URL: https://source.android.com/compatibility/9/android-9-cdd.

[89] *Android 10 Compatibility Definition | Android Open Source Project*. [Online; accessed 30. Mar. 2021]. Jan. 2021. URL: https://source.android.com/compatibility/10/android-10-cdd.

[90] *GSMArena*. URL: https://www.gsmarena.com/results.php3?idQwerty=0&sColor=&idCardslot=0&idDisplayNotch=0&idBatRemovable=0&sFreeText=&nOrder=0. (accessed: 05.01.2021).

[91] *Frequently Asked Questions - GSMArena.com*. [Online; accessed 24. Mar. 2021]. Mar. 2021. URL: https://www.gsmarena.com/faq.php3.

[92] *Common Criteria Certified Products CSV*. URL: https://www.commoncriteriaportal.org/products/certified_products.csv. (accessed: 05.01.2021).

[93] *Android Enterprise Solutions Directory*. [Online; accessed 24. Mar. 2021]. Mar. 2021. URL: https://androidenterprisepartners.withgoogle.com.

[94] *PDFGrep - a commandline utility to search text in PDF files*. URL: https://pdfgrep.org/. (accessed: 05.01.2021).

[95] *SystemProperties.java*. URL: https://android.googlesource.com/platform/frameworks/base/+/eclair-release/core/java/android/os/SystemProperties.java. (accessed: 02.02.2021).

[96] *kaltura-device-info*. URL: https://bitbucket.org/oF2pks/kaltura-device-info-android/src/master/app/src/main/. (accessed: 02.02.2021).

[97] *Locking/Unlocking the Bootloader | Android Open Source Project*. [Online; accessed 3. Feb. 2021]. Sept. 2020. URL: https://source.android.com/devices/bootloader/locking_unlocking?hl=en.

[98] *Boot Flow | Android Open Source Project*. [Online; accessed 3. Feb. 2021]. Sept. 2020. URL: https://source.android.com/security/verifiedboot/boot-flow.

[99] *Diff - 6879cc1e2ee91f47fa05a01dfbce9dfef7504501 - platform/system/core - Git at Google*. [Online; accessed 5. Feb. 2021]. Feb. 2021. URL: https://android.googlesource.com/platform/system/core/+/6879cc1e2ee91f47fa05a01dfbce9dfef7504501%5E%21.

[100] *File-Based Encryption | Android Open Source Project*. [Online; accessed 31. Mar. 2021]. Mar. 2021. URL: https://source.android.com/security/encryption/file-based.

[101] *Build | Android Developers*. [Online; accessed 5. Feb. 2021]. Dec. 2020. URL: https://developer.android.com/reference/android/os/Build.

[102] *FingerprintManager | Android Developers*. [Online; accessed 5. Feb. 2021]. Sept. 2020. URL: https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.

[103] *BiometricManager | Android Developers*. [Online; accessed 5. Feb. 2021]. Sept. 2020. URL: https://developer.android.com/reference/android/hardware/biometrics/BiometricManager.

[104] *BiometricManager | Android Developers*. [Online; accessed 5. Feb. 2021]. Sept. 2020. URL: https://developer.android.com/reference/android/hardware/biometrics/BiometricManager#canAuthenticate(int).

[105] *Measuring Biometric Unlock Security | Android Open Source Project*. [Online; accessed 5. Feb. 2021]. Oct. 2020. URL: https://source.android.com/security/biometric/measure.

[106] *PackageManager | Android Developers*. [Online; accessed 5. Feb. 2021]. Sept. 2020. URL: https://developer.android.com/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String).

[107] *DevicePolicyManager | Android Developers*. [Online; accessed 9. Feb. 2021]. Sept. 2020. URL: https://developer.android.com/reference/android/app/admin/DevicePolicyManager.

[108] *PackageManager | Android Developers*. [Online; accessed 9. Feb. 2021]. Dec. 2020. URL: https://developer.android.com/reference/android/content/pm/PackageManager.

[109] *KeyInfo | Android Developers*. [Online; accessed 27. Mar. 2021]. Mar. 2021. URL: https://developer.android.com/reference/android/security/keystore/KeyInfo#isInsideSecureHardware().

[110] *KeyGenParameterSpec.Builder | Android Developers*. [Online; accessed 27. Mar. 2021]. Mar. 2021. URL: https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.Builder#setIsStrongBoxBacked(boolean).

[111] *KeyInfo | Android Developers*. [Online; accessed 27. Mar. 2021]. Mar. 2021. URL: https://developer.android.com/reference/android/security/keystore/KeyInfo#getSecurityLevel().

[112] *MediaDrm | Android Developers*. [Online; accessed 10. Feb. 2021]. Sept. 2020. URL: https://developer.android.com/reference/android/media/MediaDrm.

[113] *Validation ReportGooglePixel Phones on Android 11.0*. [Online; accessed 29. Mar. 2021]. Feb. 2021. URL: https://www.commoncriteriaportal.org/files/epfiles/st_vid11124-vr.pdf.

[114] *Qualcomm Snapdragon 855 Becomes First Mobile SoC to Receive Smart Card Equivalent Security Certification | Qualcomm*. [Online; accessed 29. Mar. 2021]. June 2019. URL: https://www.qualcomm.com/news/releases/2019/06/25/qualcomm-snapdragon-855-becomes-first-mobile-soc-receive-smart-card.

[115] *Samsung Elevates Data Protection for Mobile Devices with New Security Chip Solution*. [Online; accessed 29. Mar. 2021]. Mar. 2021. URL: https://news.samsung.com/global/samsung-elevates-data-protection-for-mobile-devices-with-new-security-chip-solution.

[116] *Samsung bringt den Personalausweis auf das Galaxy S20*. [Online; accessed 29. Mar. 2021]. Mar. 2021. URL: https://news.samsung.com/de/samsung-bringt-den-personalausweis-auf-das-galaxy-s20.

[117] *A mysterious bug in the firmware of Google's Titan M chip (CVE-2019-9465)*. [Online; accessed 22. Feb. 2021]. Feb. 2020. URL: https://alexbakker.me/post/mysterious-google-titan-m-bug-cve-2019-9465.html.

[118] *Google is working on securely storing Digital Driver's Licenses in Android*. [Online; accessed 10. Feb. 2021]. Mar. 2019. URL: https://www.xda-developers.com/google-android-digital-drivers-license.

[119] *Android Code Search*. [Online; accessed 10. Feb. 2021]. Feb. 2021. URL: https://cs.android.com/search?q=SECURITY%5C_CERTIFICATION.

[120] *_/android/platform/system/security - Android Code Search*. [Online; accessed 10. Feb. 2021]. Feb. 2021. URL: https://cs.android.com/android/_/android/platform/system/security/+/472e6c8e18c800b0b650bd8697e17ce65c1f3608.

[121] *Comomn Criteria Mode | Knox Platform for Enterprise White Paper*. [Online; accessed 27. Mar. 2021]. Mar. 2021. URL: https://docs.samsungknox.com/admin/whitepaper/kpe/common-criteria-mode.htm.