OffPAD

Performed by: FITRAYTI AS
September 2018

1 Executive Summary

Mobile devices have become a ubiquitous commodity in today's world. They are also increasingly relied upon to run several security-sensitive applications, such as mobile banking, ticketing solutions, electronic identity and work emails. However, all evidence shows that mobile devices are not trustworthy, as they can be buggy and thus unintentionally subverted by malware, or intentionally backdoored.

As current software-based protection mechanisms can't keep up with advancing threats, hardware-based security primitives emerged as both an alternative and a complementary solution paradigm**.**

OffPAD AS has designed and manufactured a lightweight hardware-based root of trust mobile phone case, which can be universally used with all android based mobile devices. OffPAD can bootstrap a secure trusted execution environment, where security-sensitive applications can run in isolation from the malicious mobile phone. To achieve this goal, OffPAD relies in its architecture on a secure element, and trusted input/output paths to end users and remote service providers.

The goal of this report is to present the reader with a technical foundation upon which they can evaluate the security guarantees the OffPAD offers, and thus decide whether it is a suitable solution for their organization.

This report starts by presenting an overview of the mobile security landscape in which the OffPAD fits. It then moves into introducing the different current paradigms of mobile security. It subsequently discusses the advantages of rooting the security of mobile applications in hardware secure elements. The report then moves into discussing the OffPAD's security vision, guidelines, and implementation details.

2 Table of Abbreviation

API	Application Protocol Interface
ARM	Advanced RISC Machine
CPU	Central Processing Unit
MCU	Microcontroller Unit
NFC	Near Field Communication
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
SE	Secure Element
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TRH	Tamper Resistant Hardware

3 List of Figures

4 Contents

5 Introduction and Background

The last decade has been a pivotal period for mobile devices. Besides the fact that they have grown in popularity and sophistication, **they have also witnessed *a significant increase in the***

number of security sensitive applications they run. Every single day, millions of users put their trust in the security of these devices, when they use them to do their banking and mobile payment for instance. As a matter of fact, the 2016 Bain & Company security report highlighted that as of 2016, ***mobile banking exceeded online banking*** in all the 21 countries in which their study has been performed (Du Toit & Burns, 2016). However, mobile financial services are not the only security sensitive applications one finds on mobile devices. **Other such prevalent applications are mobile-based identities, work emails, personal files, mobile ticketing, and secure communication.**

While these apps come from different industries and have different risk and security profiles, what they all share is the fact that they *handle security and privacy sensitive data which needs to be stored and processed securely* (Hölzl, 2018).

Unfortunately, mobile users' security and privacy is threatened by an ever ***increasing and more complex thereat landscape in mobile devices.*** This is attributed to both malicious software running on the mobile devices, as well as adversaries with physical access. The 2016 Nokia security reported an 83% increase in mobile malware infections just in the second half of the year, While Symantec reported an overall 105% as an overall annual increase compared to 2015 (Nokia Threat Intelligence Laboratories, 2016).

This level of insecurity at the level of mobile devices did not come as a surprise to the security community. It was rather an unfortunate and logical predicted outcome of how fast ***the mobile device industry moved from manufacturing embedded environments with a predefined task defined by pre-loaded software (firmware), to manufacturing what can be referred to as general purpose computing devices.*** The aspects of the latter which had direct implications on the mobile device security are ***the introduction of local and wide area networking to consumer and the widespread open application download*** (Ekberg, 2013)**.**

These mobile general-purpose computing devices operate according to a security architecture where highly privileged system software can compromise the security of applications running on top of it. In other words, it matters little if a service provider spends considerable time and resources developing the perfect application and abiding by all secure development practices. The moment the application starts running on the mobile device and sharing system resources with a host on other unknows applications, it becomes vulnerable to any exploit in that system software***. If you want your application to be secure, then you should assume, trust and hope that the millions of lines of code of privileged system software, such as mobile operating systems, which were written by dozens of developers over the span of many years is perfectly secure and trustworthy. Quite the assumption!***

Hence, end users and application developers should be aware that *the security of a mobile device's privileged system software, is the upper bound of the security of applications running on top of it* (Rutkowska, 2015).

Confronted with such a reality, one naturally wonders about the security state of these mobile privileged system software, namely mobile operating systems.

Simply put, "**mobile phone OSes are untrustworthy.** While in principle they attempt to be more secure than desktop OSes, by preventing modified OSes from booting, by using safer languages, or by sandboxing mechanisms for third-party apps such as capabilities, in practice they are still fraught with vulnerabilities. As a matter of fact, isolation and sandboxing provided by the OS is routinely broken, as was the case of Apple iOS jail-breaking by *clicking a button on a web page.* Furthermore, mobile OSes often share code with open-source OSes such as GNU/Linux, but often lag behind in applying security fixes, meaning that attackers need only look at recent patches to the open-source code to find vulnerabilities in the mobile device's code" (Vasudevan, Owusu, Zhou, Newsome, & McCune, 2012).

Therefore, application developers need isolation and security primitives which do not require trust in the host OS and the rest of the privileged system software.

The absence of such adequate solutions endangers the innovation of many useful security sensitive mobile applications which don't accept to operate in such a risky environment, and this choose to not offer their services to end users on mobile devices. It also puts the user's security and privacy at risk for the already deployed applications.

6 Current Solutions Overview

Having established the importance of finding better security measures on mobile devices to safeguard users' privacy and security using them, this section will investigate the different paradigms and solutions which arose as a response to such a challenge.

6.1 Trust the backend server

One proposed solutions paradigm is to move all security sensitive operations to remote servers and keep the mobile device as a thin client. Such solutions assume that mobile devices are not trustworthy platforms, and that backend servers can be better secured by their service providers (Hölzl, 2018).

However, these solutions suffer several drawbacks:

- When end users accept to offload their security sensitive data and operations to remote third parties, they run the risk of it being compromised and shared with malicious entities. **This greatly increases the risk of security and privacy erosion of end users.**
- Furthermore, even if mobile devices become very light and thin, they would still need to be secure enough to handle input/output interactions with the user at a minimum level.
- Also, such a centralized architecture introduces **a single point of failure and insecurity,** where a breach to a server's provider back end server can expose all its users to compromise.

Finally, **such an architecture goes against the decentralization trend we have been witnessing in the last couple of years.** The latter is characterized using public key cryptography, where the responsibility of safeguarding credentials and security sensitive data falls at the hands of end users.

We thus conclude that while such a paradigm might be useful in certain specific use cases, it certainly shouldn't be pursued as a general efficient solution for securing sensitive mobile applications.

6.2 Security of the end device

To overcome the shortcomings of trusting the back-end server with all end users' security sensitive data and operations, ***pursuing solutions which increase the security and trustworthiness of mobile devices is necessary.*** Such solutions can be either software or hardware based.

6.2.1 Software solutions

"The security of mobile applications is often implemented at the level of privileged system software, e.g. the operating system (sandboxing together with a set of permissions*). Today's mobile OSes provide process-based isolation to protect applications' address spaces and other system resources.* ***However, these mechanisms are circumventable when the OS itself is compromised.*** Another class of attacks which defeats software level protection mechanisms is **stealthy malware, also called advanced persistent threats (APTs), hiding below the operating system,** and which can persist and survive reimaging of mobile devices" (Vasudevan et al., 2012)**.**

Indeed, software-based protection have inherent security weaknesses, as they assume that the operating system, which is a software itself, is secure and can perform such protection mechanisms in a trustworthy manner.

Such a situation where we are trying to protect software using software level solutions unavoidably leads to the paradox of who will be watching the watcher.

The legacy model of software protecting software can't keep up with advancing threats against digital security, safety, and privacy. hardware-enabled security has thus emerged as both an alternative and a complementary solution paradigm (Intel Inc, 1018)**.**

6.2.2 Hardware based solutions

Providing execution environments which are hardware rooted is one way to provide. We examine some candidate isolated execution environments in the rest of this section.

6.2.2.1 Processor based hardware security

One way to instantiate such hardware-based solution is by **using additional hardware functionalities (CPU extensions) on the main CPU.** These security extensions should provide a separate trusted execution environment, referred to as a secure world, that is completely securely isolated from the rest of the non-trusted potentially malicious environment, referred to thereafter as the normal world.

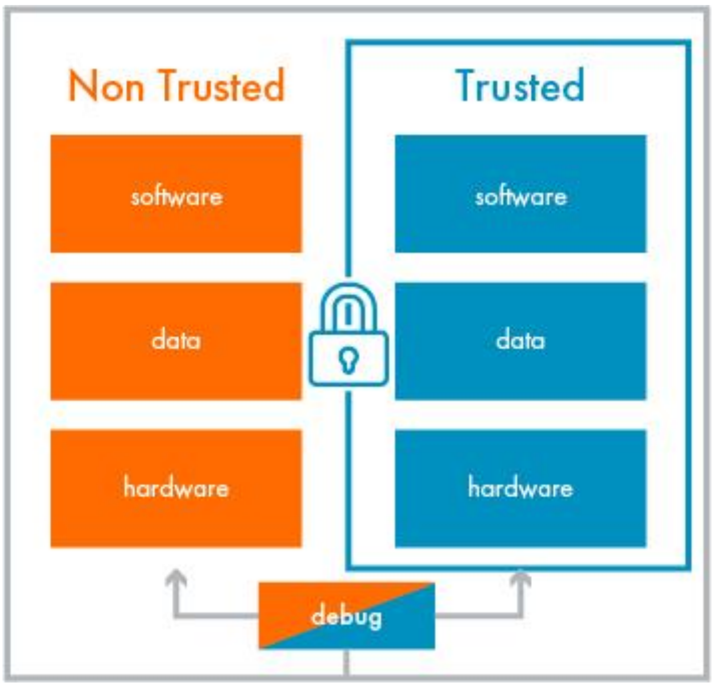


Figure 1: Trusted Execution Environment (Ekberg,2018)

One Such a successfully deployed technology on mobile devices is ARM TrustZone. “It is currently available within both Cortex-A and to Cortex-M based application processors and is commonly used to run trusted boot and a trusted OS to create a Trusted Execution Environment (TEE). Typical use cases include the protection of authentication mechanisms, cryptography, key material and Digital rights management (DRM). Applications that run in the secure world are called Trusted Apps”(Ltd, 2018).

While such processor-based solutions offer great benefits in protecting security sensitive applications, they have several shortcomings. First, the root keys which bootstrap remote trust in software running within them is not under the control of end users. Furthermore, the ARM TrustZone secure world represents one hardware trust domain, where one compromise of any software/application running within it can compromise the security of the rest of the trusted applications. Such a shortcoming is usually dealt with by running a specialized trusted OS such as the ones provided Trustsonic, which then subsequently should be relied upon and trusted to provide secure isolation between the different applications running within the ARM TrustZone secure world. Finally, the fact that they share the same CPU with the rest of the normal world, makes them vulnerable to side channel attacks triggered by potentially malicious software running in the normal world.

***Thus, we can conclude that more isolation in the execution environments of the normal and secure worlds would be a desirable security property, as it would improve the security guarantees of security sensitive mobile applications.* Solutions providing such a property can be bootstrapped using Tamper resistant hardware, which is explored in detail in the following section.**

6.2.2.2 Tamper Resistant Hardware-Based Security

Tamper resistant hardware, also referred to as a secure element, is **"a very specific kind of microcontrollers architecture with security and secret protection in mind"** (Ledger, 2015)**.** They can be defined as "a physically isolated and secure microprocessor chip which can store sensitive data and run secure apps such as payment. It acts as a vault, protecting what's inside the SE (applications and data) from malware attacks that are typical in the host (i.e. the device operating system)" (Gemalto, 2017).

It is essentially "a minimal computing environment on single chip, complete with a CPU, ROM, EEPROM, RAM and I/O port. **Recent cards also come equipped with cryptographic co-processors implementing common algorithms such as DES, AES and RSA. They are capable of running smart card applications (called applets or cardlets) with a certain level of security and features"** (lDAP Wiki, 2017)**.** Secure elements also use various techniques to implement tamper resistance, making it quite hard to extract data by disassembling or analyzing the chip (Elenkov, 2014):

- Resistance against side-channel attacks,
- Physical attacks,
- Tempest requirements.

There exist today three main form factors of secure elements, where each one links to a different business implementation and satisfies a different market need:

- Universal Integrated Circuit Card, otherwise referred to as a SIM Card
- Embedded Secure elements.
- Smart MicroSD.

6.2.2.3 Secure Element Success Stories

As a matter of fact, many industries have been successfully using secure elements as a root of trust for their applications for many years. Such prominent industries are:

- Telecom SIM Cards
- Passports
- Bank Credit cards,
- Mobile payment
- Electronic Identity

This begs the following question: if secure elements provide such obvious security guarantees, have been successfully used, and are available on our mobile device platform, then why don't we see more application developers using them to better secure the applications' security-sensitive data and credentials?

The answer to this critical question is multifaceted, as it spans both business and technical aspects, which are explored in the following sections.

6.2.2.4 The Trouble with The Current Secure Element Ecosystem

6.2.2.4.1 Constrained Resources

One of the downsides of secure element-based solutions is the constraints it imposes on storage and performance capacities. **Compared to a processor-based approach, secure elements have slower data transfer ranges and have access to much less memory**.

6.2.2.4.2 Closed Ecosystem

While the above discussed performance constraints are significant, the closed business ecosystem of secure elements is arguably the biggest obstacle towards their adoption by general purpose mobile applications. This business ecosystem is composed of several software and hardware stakeholders:

- OEM (handset manufacturers),
- Telecommunications providers or carriers,
- Application developers,
- Device owners (IT administration and/or the employee)

Within this ecosystem, secure element manufacturers have always prioritized the security requirements of OEMs and carriers over those of application developers and users. This trust model has been further consolidated by the close historical relationship between OEMs and carriers. The latter have so far been refusing to share access to their secure elements with other applications, even if multiplexing the is perfectly technically feasible. They argue that such an open ecosystem will hinder their security and performance guarantees.

Hence, Today's secure element ecosystem is characterized by a situation where mobile devices are equipped with hardware which can significantly improve the security of many security-sensitive applications, yet such hardware primitives can only be used by a handful of applications belonging to OEMs and telecom providers.

However, it is interesting to note that these same OEMs and carriers which have long argued for keeping secure elements closed, didn't shy away from changing their own rule when it came to be partnering with big third-party developers such as Google with their Google Wallet2.

"Google Wallet2 allows consumers to use their mobile phones as a virtual wallet. The application stores users' payment credentials locally, which are then used to make transactions via near field communication (NFC) with point-of-sale (POS) devices. To store the users' credentials securely, Wallet relies the device's secure Element. Unfortunately, this SE only runs code that is signed by the device manufacturer. *As a result, developers without Google's clout will not be able to leverage these capabilities for their own applications.*

There is evidence that Apple has similar plans for its products; they recently published a patent for an embedded SE with space allocated for both a Universal Subscriber Identity Module (USIM) application and "other" applications.

To remedy this, the industry is still waiting for OEMs and telecom operators to provide a more tightly integrated experience for developers by opening these pre-existing hardware security primitives available through mobile devices' secure elements to application developers" (Vasudevan et al., 2012).

6.2.2.4.3 Online Service Providers' Security Requirements

While having an open ecosystem for secure elements is a pre-requisite and a great step towards providing better security to mobile end users, it's not the whole story.

Indeed, while smart cards have indeed been a golden standard for the banking, payment and telecom industries, **they are not perfectly aligned with the security needs of internet online services.**

As a matter of fact, when you use your credit card to withdraw cash from your bank's ATM or to pay for groceries at a commercially established supermarket chain, you have high insurance that the hardware you are making the payment to is trustworthy. That is when why you enter your credit card pin and clock the ok button after having made a purchase of 100 NOK, you trust that the machine is not billing you 10000 NOK instead.

However, when you are making a purchase online, you need to trust that that your computer's hardware is not compromised and that what you are reading on the display has not been altered by malware. Thus, when interacting with internet service providers, storing passwords or private keys securely on the secure element is not enough, if you are not e sure that your display is showing you the document or the transaction that you are indeed going to sign?

(Gemalto, 2018).

Thus, solutions which want to use secure elements to provide better security for online service providers should also implement adequate measures to ensure trusted input/output channels to end users.

6.2.2.4.4 Lack of Trusted Input/output channels

When an OEM needs to securely communicate with a remote embedded controller, an end-to-end network encrypted channel is enough. Such a communication use case does not involve any interactions with end-users. Therefore, trusted input/output channels from end-users to their keyboard/display are not required.

Clearly, this is not the case for most enterprise applications such as user authentication and secure communication, which require interaction with potentially untrusted endpoint input/output channels.

7 OffPAD security principles

7.1 OffPAD Security Vision

Considering the above presented discussion about the available hardware-based security primitives built into mobile devices, their advantages and their shortcomings**, OffPAD envisions a solution which optimizes the security, usability and business strategic aspects of a hardware-based solutions, which can be used as a root of trust by various security-sensitive general purpose mobile applications.** From this vision, the following high-level properties have emerged:

- The OffPAD is a hardware-based solution architecture around a secure element, in order to leverage their higher isolation and security guarantees. The OffPAD provides a fully physically isolated execution environment, instead of one which is embedded within the mobile phone main processor and is just logically isolated.

- OffPAD is an open ecosystem where application developers are able to leverage the trusted secure execution environment of secure elements.

- OffPAD caters to the specific needs of online service providers, by providing an adequate solution for a trusted user input/output. As highlighted in the previous section, a secure element used with an ATM machine or a groceries shop terminal, does not necessarily required the use of dedicated I/O paths, since the end user has high trust in the these terminals, and believes that when they ask him to pay 100 NOK they are not going to overcharge him. However, this is not the case when the end user is interacting with a remote service provider, and is thus using an untrusted mobile phone an intermediary terminal to make a payment. In such a case, the user has no way to guarantee that a malware within the mobile phone is not going to display 100 NOK while deducting 10000 NOK from their account.

- OffPAD is a universal solution for mobile devices, with has little dependence on the type of processor or hardware implemented by the mobile device.

- OffPAD is designed in a usable form factor, which encourages end users to use it as a hardware trust anchor for their mobile security transactions.

7.2 OffPAD Implementation Security Principles

The above articulated OffPAD vision has been translated into the following implementation guidelines, which were followed during the product development of OffPAD into a functioning prototype.

7.2.1 Isolated Execution

The OffPAD provides both logical and physical isolation from the main application execution environment. This ensures an optimal secrecy and integrity of the security sensitive application' module's code and data at run-time.

7.2.2 Secure Storage

Secure storage ensures the integrity, confidentiality and freshness of the application's confidential data at rest. Today's mobile operating systems rely on software-based access control mechanisms in the form of the file system permissions in order to implement secure storage. However, these mechanisms are vulnerable to an operating system level software vulnerability, or to a physical attack where the storage media is removed, and its contents dumped.

OffPAD thus implements a stronger form of secure storage which is built using a storage location that is physically protected, and with access control implemented independently of the OS.

7.2.3 Remote Attestation

In the context of the OffPAD, we use remote attestation to refer to any mechanism which allows remote parties to achieve data origin integrity guarantees. In other words, a remote party needs to ensure that it is communicating with an application running on a device of type of devices.

While application running within the mobile execution environment can also make use of remote attestation, they are faced with the difficult challenge of the semantic gap. Indeed, if an application is running on top of a general-purpose mobile operating system, its Trusted Computing Base becomes bloated as it encompasses millions of lines of code from the operating system, firmware, device drivers, and any other privileged system software. As of today, it has been proven impossible for remote parties to use a remote attestation which include such a large TCB, as it cannot judge whether the remote attestation quote maps to a good or bad platform configuration. Therefore, it is highly desirable to have a small TCB to usefully leverage the remote attestation property.

The OffPAD provides a remote attestation capability which only includes a measurement of the smaller isolated execution environment code, and of the given security sensitive application. "This allows the attestation to be more meaningful because the isolated execution environment is presumed to be less susceptible to run-time compromise. This is important because the attestation only tells the verifier what code was *loaded*; it would not detect if a run-time exploit overwrote that code with unauthorized code" (Vasudevan et al., 2012)**.**

7.2.4 Secure Provisioning and an Open Ecosystem

Run time security of mobile applications not only requires a trustworthy execution environment, but also a mechanism to securely send data to **specific applications** which are running within **specific devices**. This is referred to as secure provisioning.

In order to implement such a mechanism based on hardware primitives, one can leverage remote attestation properties, to "attest that a public encryption key belongs to a particular software application running on a particular device. The sender can then use that key to protect data to be sent to the target software module on the target device" (Ekberg, 2013).

In order to bootstrap an open ecosystem, OffPAD implements a secure provisioning protocol, as well as makes it available to third party applications which need to leverage the security properties of the OffPAD secure execution environment.

7.2.5 Trusted Path

Trusted path provide integrity and confidentiality guarantees for applications which need to communication with input/output devices. "When used with human-interface devices, this property allows a human user to ascertain precisely the application with which she is currently interacting. With full trusted path support, malicious applications that attempt to spoof legitimate applications by creating identical-looking user interfaces will conceivably become ineffective" (Vasudevan et al., 2012).

OffPAD provides a logically and physically separate trusted input/output paths, which can only be used by the OffPAD secure execution environments, with no interference from the mobile device

7.3 OffPAD Target Attacker Model

A concise description of the OffPAD's attacker model requires the knowledge of the application architecture running on top of it. However, this section presents a generic attacker model, which can be used to analyze the security guarantees of most of applications.

The OffPAD has been designed to protect against a strong adversary.

- Such an adversary is assumed to be able to compromise the security of the mobile device, including its non-privileged as well as privileged software. The OffPAD thus also assumes that mobile device Input/output peripherals are compromised and should not be trusted.
- OffPAD protects against an adversary who has physical access to the mobile device.

Furthermore, the following threats naturally fall outside the OffPAD's threat mode:

- Software vulnerabilities present in the applet software.
- Compromise to any remote service provider back end server.
- Supply chain attack where the OffPAD hardware is compromised.
- Compromise to the provisioning of the issuer security domain keys.

While these threats fall outside of the formal adversarial model OffPAD, the latter still aims to provide adequate mitigation strategies in order to minimize their risk. For instance, OffPAD aims to implement a thorough set of guidelines and reviews to OffPAD application developers. It also aims to provide reference application architectures which guide the design of secure OffPAD applications.

8 OffPAD Architectural Details

In order to implement the above discussed security principles, OffPAD was designed around a customized dual chip architecture, centered around a secure element (SE) and a proxy microcontroller (MCU).

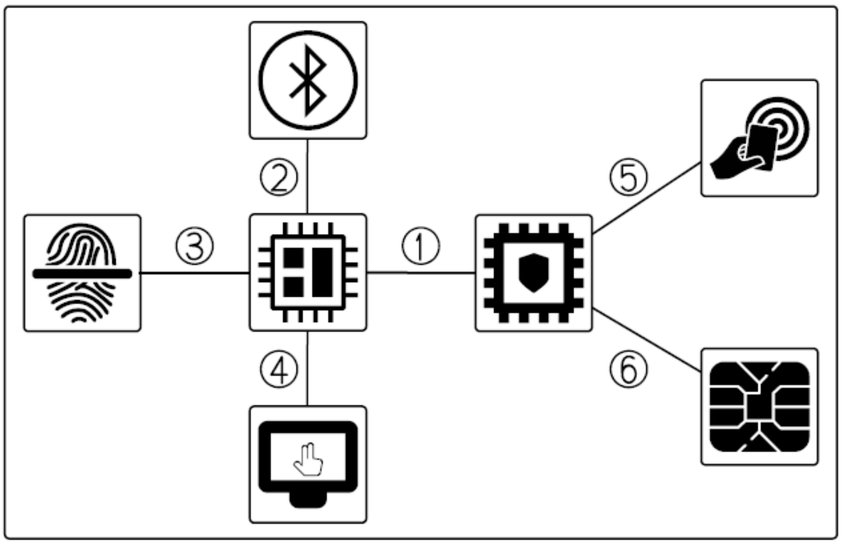


Figure 2: OffPAD High Level Architecture

8.1 Components description

8.1.1 Secure element

OffPAD relies on a secure element to provide an isolated secure execution environment for security sensitive data to be stored and operated on. The implemented SE instantiation is a Gemalto IDCore 10 Java Smart card. According the SE datasheet, The IDCore 10 smartcard "benefits from the latest release of Java Card technology standards. **It is available as an open, multi-application card and. It is a Public Key Java Card (supporting both RSA and elliptic curves) that meets the most advanced security requirements of long-term, multi-application programs, including those being deployed by large global organizations.** IDCore 10 complies with the latest international standards: Java Card 2.2.2 (& 3.0.1 for the elliptic curves algorithms) Global Platform 2.1.1 (amendment A) ISO 7816. It also includes multiple hardware and software countermeasures against various attacks: side channel attacks, invasive attacks, advanced fault attacks and other types of attacks" (Gemalto, 2018)
The SE software is split in 2 parts:

1. A proprietary low-level software that is under NDA.
2. An open source friendly software composing the loaded applications.

8.1.2 The Microcontroller

The secure element has no direct links to the input/output peripherals. However, the OffPAD incorporates an ARM-Cortex M4 processor, STM32f405.

The MCU maintains a direct communication with the SE, as well as the OffPAD peripherals, and is used to a proxy which mediates an indirect communication between the SE and I/O.

The rest of the components that the OffPAD boosts are summarized in the table below:

Component	Model	Description
Secure Fingerprint reader	FPC1020Touch Fingerprint sensor	Can store a minutia securely using a secret key shared with the SE
Secure Display	e-ink 2.5-inch display	-
A 32 MB flash memory		Private and secure storage
NFC Transceiver	NPC 102A2EV	NFC forum certified
microUsb	**USB ORG 2.0(High Speed**	

Table 1: OffPAD Components

Usability is a critical design decision which can either hinder or boost a product’s adoption. Thus, OffPAD decided to implement its product as a phone cover, which will not burden the user with a cumbersome extra device, and which is at the same time large enough to incorporate all the OffPAD components, namely its secure input/output peripherals.

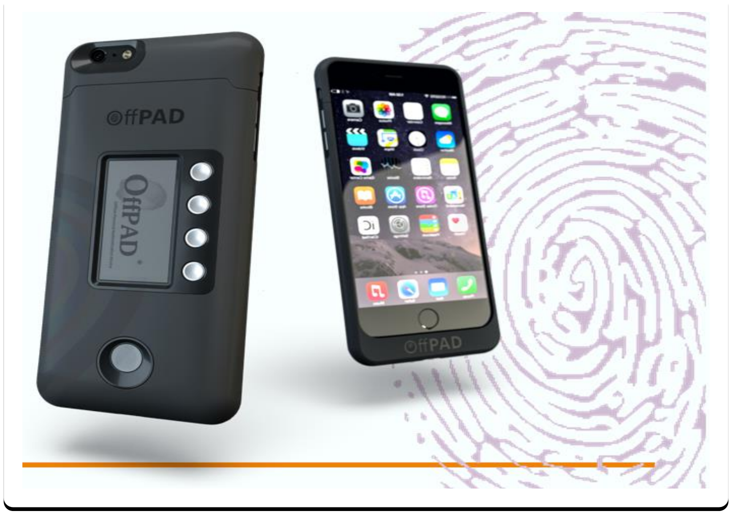


Figure 3: OffPAD Phone Cover Form Factor

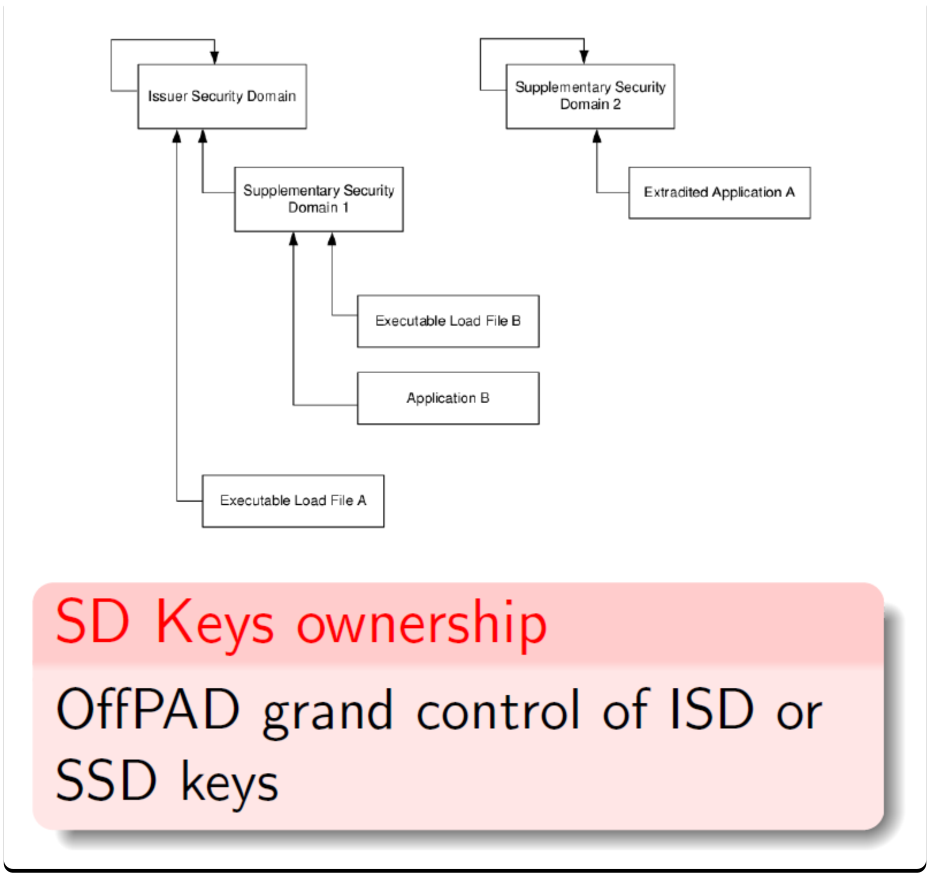
9 OffPAD in Deployment

9.1 Lifecycle

In multiapplication smart cards, each application consists of a separate security domain that isolates its application resources from other applications on the card.” The domain owner controls each security domain, and therefore, hosting an application in a card requires either the creation of a separate domain or the permission of a domain owner. This architecture where the domain owner retains the right to the host application in a security domain is called the issuer-centric card ownership model” (Tamrakar, 2017).

While the OffPAD adopts the issuer centric model, it has decided to also offer interested customers the possibility to control the ownership of the issuer’s security domain (ISD).

Alternatively, the key management itself can be taken over by OffPAD and offered as a premium service to interested customers.



SD Keys ownership

OffPAD grand control of ISD or SSD keys

Figure 4: OffPAD Key Ownership Model

9.2 Enterprise integration

The successful adoption of OffPAD in an enterprise environment requires its integration with the different enterprise device and identify management solutions.

Because OffPAD is implemented around the JavaSmart card, it can adopt industry standards in order to integrate its identity management with tools such as Microsoft Intunes, or Azure Active directory services. Certificate enrollment services will be the preferred method to achieving such a goal.

Successfully integrating OffPAD with different enterprise grade management software will greatly simplify the IT management of OffPAD devices, which will be no different to managing an enterprise-owner computer or smart phone.

10 OffPAD Applications

10.1 OffPAD as a Root of Trust

The way security critical applications are supposed to leverage the high security guarantees offered by the OffPAD should take into consideration its constrained execution environment while designing their architecture.

As a matter of fact, it is important to reiterate the notion that OffPAD is not just a secure storage, but also a hardware-enabled root of trust, where security sensitive modules of distributed application can run securely.

What this means in practice is that applications should preferably adopt a distributed architecture, where their lightweight security sensitive applet runs on the OffPAD, in conjunction with an app running on the host computer.

Further practical guidelines to be considered are as follows:

- The Secure Element has a limited storage capacity. Thus, it is preferred to store encrypted data on the mobile phone.
- "Applications that process substantial amounts of data may not be able to store all of the data on the device at once. There are two effective solutions to this problem:
    - Stream data through the device while the application processes it (for example, it may be encrypting / decrypting the data) or
    - Derive a secret on the device (preferably from the master seed) and use it on the computer to process the data (this is how the PGP app works)" (Ledger, 2017).

10.2 Sample applications

Applications which can leverage the security guarantees of the OffPAD are numerous and varied. In this section, we highlight the applications that the OffPAD is ready today to offer to end users.

- Act as a Cryptography providers: Java/Android, .net, PKCS#11, etc. This will allow application developers to use the OffPAD secure implementation of the cryptographic protocols they require, versus relying on potentially weaker implementation provided by the mobile phone's system software.
- Provide a strong second factor authentication.
- Act as a root of trust for password managers.
- Provide secure storage for data that end user deems to be sensitive.
- Implement a FIDO authenticator able to provide a secure and password less authentication experience to various online service providers which are currently supporting the FIDO protocols, such as Facebook, Gmail and Salesforce.
- Provide a secure platform for Electronic Identity.
- Hold electronic ticketing application
- Provide secure physical access to building

The power of the OffPAD can be greatly amplified if it is used by end users to support multiple security critical applications. It is indeed a power proposition to have one trusted secure device to gain physical access to your building, login to your different online service providers, bootstrap secure communication with other parties, and have a ready

11 References

Arm, L. (2018). TrustZone for Cortex-A – Arm. Retrieved September 3, 2018, from https://www.arm.com/why-arm/technologies/trustzone-for-cortex-a

Du Toit, G., & Burns, M. (2016). *Customer Loyalty in Retail Banking: Global Edition 2016* (Bain Report). Retrieved from https://www.bain.com/insights/customer-loyalty-in-retail-banking-2016/

Ekberg, J.-E. (2013). Securing Software Architectures for Trusted Processor Environments, 92.

Elenkov, N. (2014). *Android Security Internals: An In-Depth Guide to Android's Security Architecture* (1st ed.). San Francisco, CA, USA: No Starch Press.

Gemalto. (2017). What is a Secure Element? - Just Ask Gemalto. Retrieved September 3, 2018, from https://www.justaskgemalto.com/en/what-is-a-secure-element/

Gemalto. (2018). IDCore 30: Java-based smart cards - Product Brief. Retrieved September 3, 2018, from https://safenet.gemalto.com/resources/data-protection/idcore30-smart-card-product-brief/

Hölzl, M. (2018). Applying Smart Cards to Security Critical Mobile Applications, 28.

Intel Inc. (1018). Hardware-Enabled Security. Retrieved September 3, 2018, from https://www.intel.com/content/www/us/en/security/hardware/hardware-security-overview.html

IntrinsicID Inc. (2017, August 2). Pros & Cons of Secure Elements. Retrieved September 3, 2018, from https://www.intrinsic-id.com/pros-cons-secure-elements/

lDAP Wiki. (2017). Ldapwiki: Secure Element. Retrieved September 3, 2018, from https://ldapwiki.com/wiki/Secure%20Element

Ledger. (2015, January 17). Bitcoin security: why smart cards matter. Retrieved September 3, 2018, from https://www.ledger.fr/2015/01/17/bitcoin-security-why-smart-cards-matter/

Ledger. (2017). *Persistent Storage and PIC — Ledger Documentation Hub 2 documentation* (Technical Documentation). Retrieved from https://ledger.readthedocs.io/en/latest/userspace/memory.html

Leroy, X. (2002). Bytecode Verification on Java Smart Cards. *Softw. Pract. Exper.*, *32*(4), 319–340. https://doi.org/10.1002/spe.438

Nokia Threat Intelligence Laboratories. (2016). *Nokia Threat Intelligence Report – 2H 2016.* (Technical Report). Retrieved from https://pages.nokia.com/8859.threat.intelligence.Report-Thank.you.html

Oracle. (2003). Inside Java 2 Platform Security, Second Edition: Architecture, API Design and Implementation. Retrieved September 3, 2018, from https://www.oracle.com/technetwork/java/javaee/gong-135902.html

Rashid, S. (2018). *Breaking the Ledger Security Model* (Technical Report). Retrieved from https://saleemrashid.com/2018/03/20/breaking-ledger-security-model/

Rutkowska, J. (2015). *Intel x86 considered harmful* (Technical Report).

Tamrakar, S. (2017). *Applications of Trusted Execution Environments (TEEs)*. Aalto University. Retrieved from https://aaltodoc.aalto.fi:443/handle/123456789/26624

Vasudevan, A., Owusu, E., Zhou, Z., Newsome, J., & McCune, J. M. (2012). Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me? In S. Katzenbeisser, E. Weippl, L. J. Camp, M. Volkamer, M. Reiter, & X. Zhang

(Eds.), *Trust and Trustworthy Computing* (Vol. 7344, pp. 159–178). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-30921-2_10