Future Generation Computer Systems 80 (2018) 596–612

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Trusted mobile computing: An overview of existing solutions

Mohamed Amine Bouazzouni [a,b,c,*], Emmanuel Conchon [d], Fabrice Peyrard [a,b,c]

[a] University of Toulouse, 2 Rue Camichel, Toulouse, France
[b] INP, 2 Rue Camichel, Toulouse, France
[c] IRIT UMR 5505, 2 Rue Camichel, Toulouse, France
[d] University of Limoges, XLIM, UMR CNRS 7252, 123 avenue Albert Thomas, 87060 Limoges, France

HIGHLIGHTS

- Survey of trusted mobile computing.
- Presentation of hardware and virtualized solutions of secure execution environment.
- User centric models for trusted mobile computing.
- Analyses and discussion of the current and future solutions.
- Installing and testing an open source TEE adapted for UCOM model.

ARTICLE INFO

Article history:
Received 11 September 2015
Received in revised form
23 May 2016
Accepted 25 May 2016
Available online 8 June 2016

Keywords:
Mobile Trusted Computing
Trusted Execution Environment (TEE)
Secure Element (SE)
Trusted Platform Module (TPM)
Virtualization
User Centric Model

ABSTRACT

Nowadays, smartphones are able to process large amounts of data enabling the use of applications for personal or professional use. In these contexts, the smartphone needs to process, store and transfer sensitive data in a secure way. Encryption is a commonly used solution to enforce security but the encryption keys it relies on have also to be securely processed and stored. Several research works have investigated these issues and different solutions have been proposed. They can be classified into two main categories: hardware-based solutions (Secure Elements, Trusted Platform Module and Trusted Execution Environments) and software-based solutions (Virtualization Environments). This paper overviews/surveys these two categories highlighting their pros and cons. Examples of trusted computing applications are then provided for each category. Finally, a discussion is provided about trends and perspectives for trusted mobile computing.

1. Introduction

Over the past years, smartphones have become an important part of our everyday life. According to recent statistics [1], the smartphone market has grown from 20% in 2014 and recently 10% in 2015. Moreover, smartphones usage has changed in recent years with their increase of computing and storage capabilities. Mobile applications used on smartphones become more and more sophisticated targeting new fields of application such as health, entertainment, business productivity and social interactions for instance. These applications can sometimes process sensitive or personal data that require a high level of security and confidentiality. Indeed, applications targeting banking, ticketing, health monitoring applications or even Digital Rights Management (DRM) for multimedia applications have to be secured so that the user confidence will increase. Therefore a need for mobile trusted computing arose as data security has not been considered as mandatory in existing mobile operating systems (OS). For instance, Fahl et al. [2] have proposed in 2012 a study of SSL-based applications in the Android market showing that nearly 8% of them are vulnerable to a well-known Man in the Middle attack.

To address this need of trusted mobile computing, several technical solutions have been provided in the past few years among which are hardware-based solutions like Secure Element (SE), Trusted Platform Module (TPM) and Trusted Execution Environment (TEE). More recently, thanks to the performance of embedded multi-core processors, software-based solutions appear especially for Virtualization Environments (VE).

* Corresponding author at: IRIT UMR 5505, 2 Rue Camichel, Toulouse, France.
E-mail addresses: mohamedamine.bouazzouni@enseeiht.fr (M.A. Bouazzouni), emmanuel.conchon@unilim.fr (E. Conchon), fabrice.peyrard@enseeiht.fr (F. Peyrard).

http://dx.doi.org/10.1016/j.future.2016.05.033
0167-739X/

Secure Element is a popular solution that has been widely deployed in recent smartphones. It is a dedicated component that enables the secure execution and storage of applications with cryptographic capabilities. It can be viewed as a smartcard which executes JavaCard applications in a smartphone. A SE can be embedded into a SIM card issued by a telecom service provider, into an embedded chip issued by the phone manufacturer or even into a removable micro SD card. As a SE can be viewed as an embedded smartcard, it suffers from the same limitations such as the issuance of application. Indeed, to store a new application on the SE, an agreement has to be made with the SE issuer. Moreover, the storage capacity and processing are limited which is a handicap for large-scale deployment.

Another hardware solution is the Trusted Platform Module that has been originally designed for laptop environment. TPM consists of a regular micro-controller with additional cryptographic capabilities that can perform complex cryptographic operations easily. However, the storage capacity is very limited so that only these cryptographic operations can be performed whereas SE can perform every possible operations of an application.

Trusted Execution Environment is another proposition to overcome the deployment issues encountered in SE-based solutions. The TEE is an environment of secure storage and processing with greater storage and processing capacities than the SE. TEE can be split into a hardware part and a software part. The hardware part consists of a processor which isolates the execution of sensitive applications from the regular ones. The software part consists of two separate OS: a Rich OS and a Secure OS isolated from each other. The Secure OS uses a limited instructions set that allows its verification and validation from a security standpoint. The advantage of this solution is that this processor is present on almost all modern smartphones. Since 2009, GlobalPlatform [3] is in charge of the TEE standardization. Based on these standards, ARM, one of the major mobile processor manufacturer, has integrated TEE on most of existing smartphones and enabled its usage. However, several challenges have to be addressed by TEE solutions among which are their ability to deal with physical attacks, the necessity to provide a secure boot to avoid rootkit as well as a secure channel that protects the interactions between the user and the smartphone.

The last solution is Virtualization Environment. It consists of running multiple virtual OS on a single physical phone. The isolation between applications and the secure storage are the main challenges. Adding minimum overhead to the execution is also a major criteria to evaluate the quality of this solution.

Most of these solutions are proprietary and follow the Issuer Centric smart card Owner Model (ICOM) which restricts the access to such technology. To get access to these technologies, the developers have to sign agreements with the hardware manufacturer. This constraint has hindered the development and dissemination of these technologies. For this reason, the need for technologies following the User Centric smart card Owner Model (UCOM) has become a priority. This model allows users to have the full control of their environment. Virtualization seems to be a good candidate for a UCOM model adoption.

In this paper an overview of existing trusted mobile computing solutions will be provided. Complementary to the major surveys proposed by Asokan et al. in [4] and by Vinh et al. in [5], this paper will provide an extended presentation of SE, TEE and VE attempts, and will compare them for trusted mobile computing.

This overview is intended to describe all the hardware and software bricks to consider in the development of secure mobile applications such as mobile payment, transport ticketing, DRM and building secure access for instance. Indeed, many access control systems are now based on RFID technology despite well known cloning weakness [6,7]. These access controls (packing, buildings, offices, hotel, ...) do not always check authentication of the holders of RFID tags. NFC technology which is a more secure evolution of RFID and is increasingly implemented on smartphones has led to design new access control systems. These new systems are based on the concept of a virtual card where a smartphone can be used as a substitute to a regular smart card. Such operation is called dematerialization.[1] This requires more security solutions such as encryption and mutual authentication. Applications and data from these systems must be carried and stored in secure and trusted environments.

In Section 2, a short overview of applications provisioning models will be given. In Section 3, several fields of application of the secure mobile computing will be highlighted. In Section 4, the characteristics of secure environments are presented as well as the three attacker models chosen in this work. Sections 5 and 6 highlight respectively hardware-based and software-solutions for trusted mobile computing before exposing main existing approaches based on these solutions in Section 7. The Section 8 is a comparative analysis of the robustness of Secure Environments against attacker models, then a comparison of Trusted Mobile Computing approach for user-centric solutions. Finally, potential solutions for the upcoming years will be discussed in Section 9.

2. Application provisioning models

Flexibility in application provisioning has become an important factor for the proper development of some technologies. The most spread model is Issuer Centric Smart Card Owner (ICOM). A new model, called UCOM, centered on the user, is discussed in [8,9]. UCOM allows the card issuer to delegate the ownership of the card to the user.

2.1. ICOM model

This model gives the full power to the issuer [10] allowing him to pre-install, install, remove and manage the applications on the device. This model is service oriented. Indeed, every service provider installs its own application(s) on the device and does not share its secure environments with other issuers or with the users. This reason leads to limit the development of such solutions. This model is the most common because it provides the following advantages:

- The card is controlled by the issuer. He handles the applications and their life cycle. This centralized management led to the robust reputation of this model.
- The centralized management is an asset to achieve a high level of security. The issuer is the only one able to enforce the suitable security level for his applications. This applications' code is checked by the issuer allowing the limitation of the spread of malwares.
- Changes on the card are only handled by the issuer. A user cannot modify, install or remove any application on the smartcard without the issuer intervention.
- The card communications are controlled by the issuer. Different protocols can be used such as TLS or SSL. Depending on the sensitivity of the application, a communication model is selected by the issuer.

Fig. 1 shows the usage of smartcards and their life-cycle.

These characteristics enabled smartcard to become a key security element mainly used for the most sensitive services such as bank payments. However this issuer full control leads to many

[1] In the remaining of the paper, the word dematerialization will mean replacing a physical card with a virtual one saved on a smartphone.

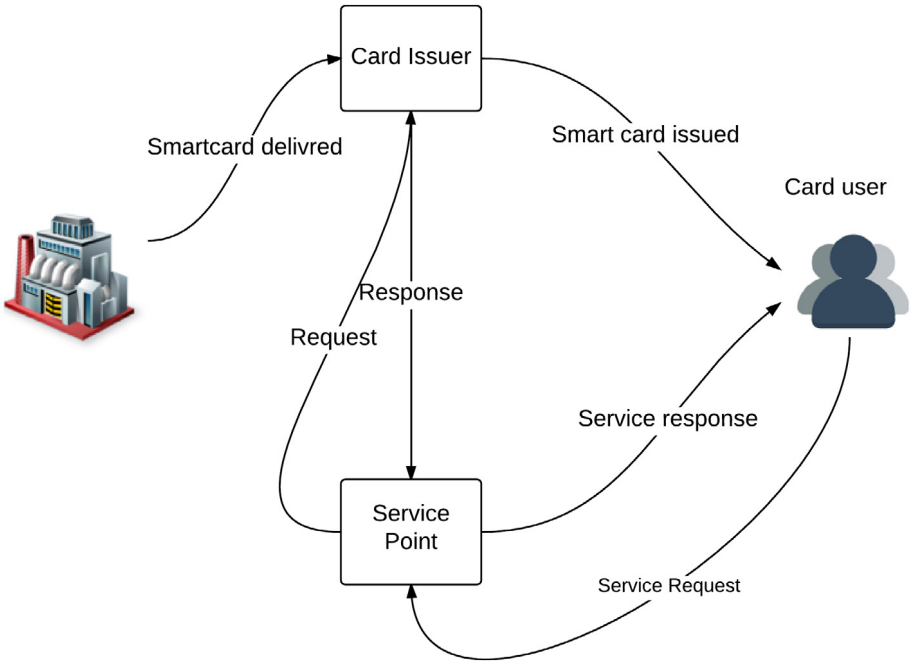

Fig. 1. Smartcard usage and lifecycle.

drawbacks and slows the expansion of this technology. These drawbacks are:

- A card can be used by a single service. With the proliferation of services, the number of cards used in everyday life is growing which is a drawback from the user stand point.
- User access to the card is restricted. All operations including services and application management are the sole responsibility of the manufacturer.
- Deploying new services requires the issuance of a new card. This constraint generates costs and significant delays that penalize the development of these technologies.

The main argument that led to this model is the problem of the card ownership. In the ICOM model, as the issuer owns the card, he is legally responsible for it and for the security of applications and data used on this card. However, with the increase in the number of services offered, it became necessary to find a solution that allows the user to install as many applications as needed. A User Centric Smart Card Ownership Model (UCOM) emerged [10,11]. Fig. 2 shows the UCOM model characteristics.

2.2. UCOM model

In this model, users have the full control of their cards management. Indeed, they are able to install any applications they want on their smartphone. The ownership of the card is transferred from the issuer to the user who is responsible for the operations the card makes.

Moreover with the increased use of the Near Field Communication technology [12], many services appeared to facilitate the life of clients by allowing different smartphones usages including contactless payment for instance. For these services, storing data securely and ensuring the integrity of the execution has become an unavoidable challenge to provide quality services with a high level of security. The most intuitive solution is to exploit the capacity of the Secure Element present on the smartphone. Nevertheless, for now, this technology is not accessible for developers and users. Indeed, chip manufacturer signs agreements with major companies granting them the use of their chip. But for developers and users, getting such agreements is almost impossible. This major limitation has slowed the development of NFC-based services such as secure card emulation for building access control applications.

3. Fields of application and attacker models

In the first part of this section, we will answer the question: In what field do we process secure data in everyday life? We will show that many use cases need secure storage and processing. Finally, we will expose some attackers profiles that can be encountered in the mobile world and what kind of damages they can cause.

3.1. Fields of application

Many applications need to store and process secure data which leads to the development of secure environments. The growing use of these secure applications is linked to smartcards dematerialization. Indeed, the classical contactless cards can now be dematerialized on a NFC-based third party hardware like a smartphone. This dematerialization process has straighten the need for secure processing and storage environments. For instance several solutions may be concerned by this dematerialization process such as:

- Mobile payment: In recent years, every bank has developed its banking application for mobile platform. These applications process sensitive data which require secure environments. The data are transmitted from the smartphone to the payment terminal via the NFC technology. The smartphone has to store the card data to perform the EMV transactions [13]. The storage and the processing are performed in the secure environment to minimize the security flaws [14].
- Transport ticketing: This kind of applications are widely disseminated because they eliminate the classical plastic card. Economically, this solution is very cost effective because it allows to save money by replacing the cards by a smartphone application. The subscription is hold by the smartphone and the user taps his smartphone on the access reader to allow him use public transport. The data transmission uses NFC

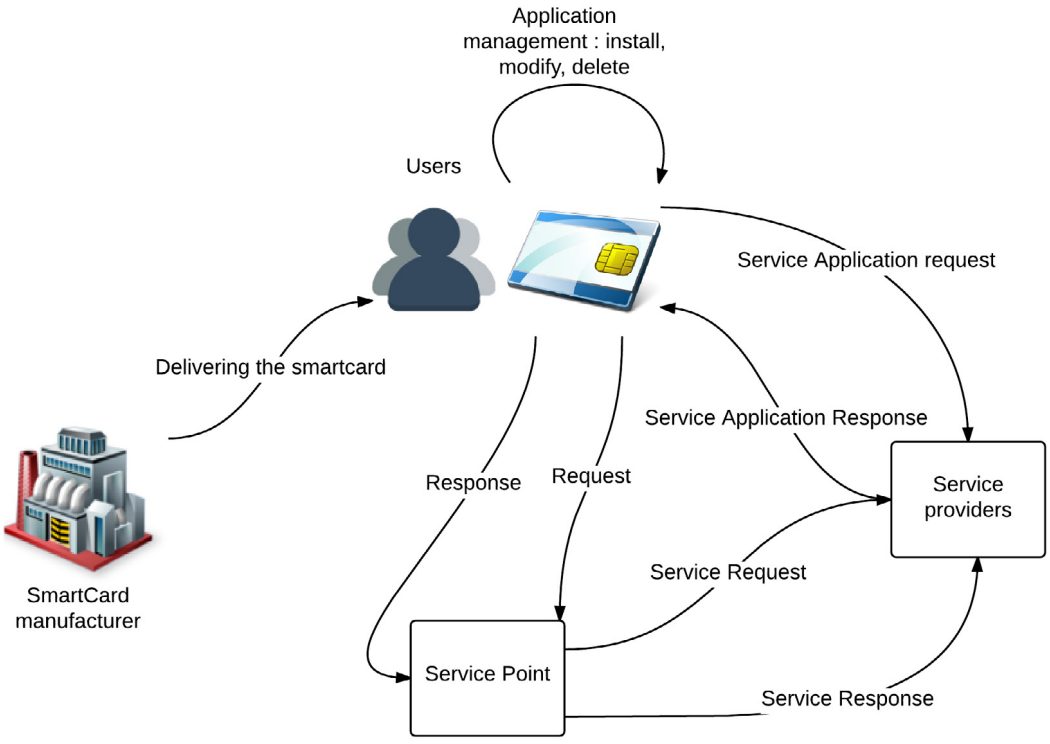

Fig. 2. UCOM model.

technology. Also, the dematerialization allows developers to improve the security level of the deployed solutions. Actually, the computation power of the smartphones facilitates the development of more reliable security mechanisms as those deployed on ordinary cards. The card data must be stored and processed securely in the secure environment to avoid fraud. This technique has also a big advantage which is to facilitate the recovery of access to public transport after the loss of the phone [15].

- Buildings secure access: Some sensitive buildings may need access control. This process can be performed by the classical contactless cards. Some of these cards can have security breaches. For these reasons and others, people prefer to use dematerialized cards in smartphones. Indeed, these smartphones use NFC technology to communicate with the reader. The authentication data must be stored and processed securely.
- DRM: Several Data Right Management solutions exist on computers. Nowadays, DRM emerged on mobile phone platforms. These DRMs need secure storage and secure data processing. Firstly, the License and the cipher key must be stored securely on the phone. On the other hand, the stream is routed to the secure hardware to be processed there. By using TEE, the ciphered stream and the keys are in the secure component and all the process is performed securely. This solution guarantees no data leaks [16].

3.2. Attacker models in mobile computing

In [17], Cooijmans et al. exposed three kinds of attacker that can target a mobile device: Malicious App Attacker, Root Attacker and Intercepting Root Attacker. In this paper, it has been chosen to consider the same three attackers models for the evaluation of trusted mobile computing solutions.

- **Malicious app attacker**: The attack is performed by a malicious installed application. Indeed, the attacker designs the application to intercept sensitive information transmission and processing. The application is able to use all the declared permission to spy on other applications.
- **Root attacker**: This kind of attacker have root credentials and are able to run applications with root permissions. It allows them to inspect the file system. This scenario is increasingly common. Indeed, millions of Android users have rooted their phone.
- **Intercepting root attacker**: This kind of attacker has the same abilities of the root attacker with an access to the input/output operations and the capacity to inspect the device memory.

4. Secure execution environment: Definition and characteristics

A secure environment is an environment that allows the secure storage and execution of applications. This feature is guaranteed by the following requirements:

- Isolated execution: Every application should run independently of other applications. This ensures that a malicious application cannot access sensitive data manipulated by an other secure application. This also implies that the malicious application cannot access the code or data of an application while it is running and also cannot alter the execution.
- Secure storage: This characteristic ensures the integrity and secrecy of all data including the binaries representing the applications to be run. The same security properties should be guaranteed to the application data and cache. The most sensitive data to protect prove to be the passwords, encryption keys and certificates.
- Secure provisioning: This property guarantees the capability to send data to a specific software in the secure environment of a specific device while ensuring the integrity and secrecy of the data exchanged. This feature includes the ability to remotely install sensitive applications and transfer encryption keys or certificates.

An application running in an environment which guarantees these three characteristics is called a secure application in the remaining of the paper.

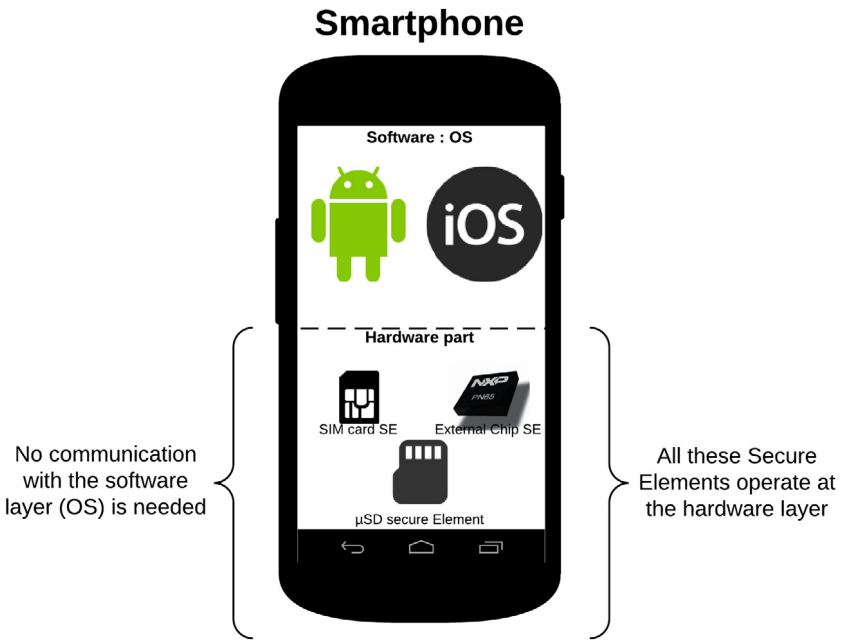

Fig. 3. Types of Secure Elements.

At this point, a legitimate question arises: Does encryption address the security issues in the mobile world? The answer is no. The reason is that encryption is a process which needs an algorithm and a key. The storage of the encryption key is the principal challenge we face. Two of the three attacker models exposed in Section 3.2 are able to either inspect the storage memory or intercept the key during the execution process. The environments presented in next sections provide secure storage and execution solutions to address the issues of secure storage and execution in the mobile world. Moreover, the resistance of these solutions to the attacker models is better than the resistance of encryption.

In this paper, two kinds of secure environments are considered: hardware-based and software-based environments. In the next section, the hardware-based solutions are presented: Secure Elements, Trusted Platform Module and Trusted Execution Environment. In Section 6, software-based solutions are exposed.

5. Hardware-based solutions

Among the existing solutions, the hardware-based ones are the most deployed by manufacturers. Indeed hardware security solutions have the advantage of greatly reducing intrusions and attacks. In addition, low engineering and manufacturing costs of silicon components allow integration in large public terminals. The massive deployment of SIM card incorporating a Secure Element, is the origin of the hardware-based solutions. In this section, SE, TPM and TEE solutions which are three different kind of hardware-based solutions are presented.

5.1. Secure Element (SE)

SE are chips with the same features as a regular smartcard. Indeed, the SE applications are JavaCard applications. These chips are tamper resistant and provide a high level of security [5,18]. We can distinguish three kinds of SE as depicted in Fig. 3: Embedded SE, UICC SE and Micro SD SE [19].

5.1.1. The embedded SE

In this type of SE, the chip is wired to the motherboard of the phone. It is not removable or transferable to another phone. Security in this case is equivalent to the security of a conventional smartcard. Indeed, on an Android phone with NFC, the embedded SE is only wired with NFC controller and cannot communicate with any other part of the phone.

The drawback of this solution is the difficulty to define coherent access rights to the chip. These rights are defined by the chip manufacturer. They can be restrictive and do not allow developers to install their application on the chip. This restriction makes this solution difficult to implement.

5.1.2. The UICC SE

The SIM cards delivered by the telecom service providers can contain a secure environment to store sensitive data and execute secure applications. The advantage of this proposition is that all smartphones can use SIM cards unlike embedded SEs that are not present on all the phones.

The main drawback of this solution is that agreements with telecom service providers are needed. Indeed, secure applications cannot be installed without these agreements. More specifically, secure applications have to be signed by the telecom service providers. Signatures can be verified with the certificate emitted by the telecom service providers. It has to be noted that this signature process is very constraining for applications developers.

5.1.3. The micro SD card SE

This kind of SE has the same security level as the previous SE. Unlike the previous types of SE, this micro SD card is not wired with the motherboard of the phone and does not depend on the telecom service providers. These cards are removable and transportable from one phone to another. This property is simultaneously an advantage and a disadvantage in terms of security. Removability of this card responds to a need for mobility. Indeed, if the user often changes his mobile phone, then it is interesting to keep his sensitive data on a removable media even if there is a risk of card theft.

5.1.4. Comparative analysis

The three kinds of secure elements have the same native security level. However, the manner in which they are connected to the smartphone may induce some differences. The embedded

secure element is connected to the motherboard of the smartphone and is directly wired with the main input/output peripherals. In the other hand, UICC SE and micro SD card SE are external devices and the communication between these peripherals and I/O peripherals is achieved through the OS.

The provisioning model for the SE is based on an ICOM approach which makes this solution reliable for secure application developers due to manufacturers restrictions. Indeed, only the applications signed by the manufacturer are allowed to run in a SE which limits the possibilities of a malicious app attacker and of a root attacker. On the other hand, this is a big restriction for developers and the main reason that hindered the development of this technology.

Google sets up a Secure Element Evaluation Kit for the Android platform called SEEK [20] for Android. This kit provides a standardized smartcard API to access the SE on Android devices.

5.2. Trusted Platform Module (TPM)

As previously mentioned, the TPM consists of a micro-controller with additional cryptographic capabilities. In 1999, the Trusted Computing Group (TCG) [21] was charged, in collaboration with several industry companies to elaborate specifications for a new computing platform in the aim to ensure privacy and enhance security for laptops. Their security mainly relied on data encryption, login with username and passwords and using tokens for authentication like biometrics. However, these elements were subject to many weaknesses: data theft, unauthorized access to laptops, unauthorized access to the network.

- Data theft: Due to the mobility of notebooks, the probability that the terminal is stolen is higher than a desktop one. To avoid theft of sensitive data on the device, the solution was to encrypt them. This solution implies a big weakness: The keys used in the encryption process are stored on the hard disk and are sensitive to tampering attacks. To address this problem, TPM proposes to store securely the encryption keys on the hardware (notebook).
- Unauthorized access to laptops: If anyone can get access to the platform, it should be necessary to limit the impact of attacks that can be perpetrated by malicious users. To limit the accesses, many solutions were implemented including username/password and biometrics authentication. These solutions are still weak to several attacks. The first attack is a dictionary and brute force attacks on passwords [22]. Next, biometrics can be spoofed to mislead the authentication mechanism [23]. The last and major drawback of these solutions is that the credentials are not bound to the platform. To address these drawbacks TPM propose to bind the authentication credentials to the platform and to secure their storage.
- Unauthorized access to the network: A stolen notebook with the right credentials could access the company network and steal sensitive data. Several solutions were deployed to ensure the security of the network access. Among these solutions we can list: Windows network logon and IEEE 802.1x [24]. These solutions have weaknesses [25] which allow the attacker to bypass these security mechanisms. Moreover, the certificates used to perform authentication can be spoofed. To address these problems, the TPM proposes to implement a PKI based method for platform authentication and to perform a hardware protection of the authentication data.

The TPM is a hardware implementation of the TCG specifications designed to provide solutions for the above weaknesses.

5.2.1. TPM architecture and features

The TPM is a secure micro-controller with the addition of cryptographic features [26]. The TPM can be bound to the device using a Low Pin Count (LPC) bus. The TPM is able to perform very complex cryptographic operations from the symmetric encryption to RSA asymmetric encryption [27]. The advantage of using a TPM is that the developer has not to know anything about the implementation of these algorithms. Indeed, these features are integrated to the TPM providing an Application Programming interface (API).

The cryptographic features of a TPM are the following:

- RSA accelerator: An engine module performs RSA encryption/decryption with a maximum key length of 2048 bits. A built-in RSA engine is used during digital signing and key wrapping operations.
- The TPM is capable of computing hash values of pieces of data. The TPM storage and computing capabilities are not sufficient to hash large pieces of data. The main purpose of TPM is the processing of small sensitive data such as encryption keys and certificates.
- Generating pseudo random numbers: The TPM is able to generate pseudo-random number. This feature is very important and useful to generate encryption keys especially for RSA for instance.

5.2.2. TPM platform architecture

Fig. 4 shows the main elements of a TPM platform: The Endorsement Key (EK), Attestation Identity Keys, Certificates and Platform Configuration Registers (PCRs).

1. *The Endorsement Key (EK)*: It consists of a public/private key pair. The size of this pair is 2048 bits. The private key is generated into the TPM and is never used externally. This feature makes the recovery of the key impossible for an attacker. This key is unique for every platform and TPM. The EK generation process differs from a manufacturer to another. The first way to generate the key is the use of a specific TPM command. The second way is to generate the key outside the TPM. The manufacturer seals the key in the TPM during the manufacturing process.
2. *Attestation Identity Keys*: The main purpose of these keys is to perform an authentication with a service provider. This authentication affects the platform and differs from the user authentication.
3. *Certificates*: The TPM stores three kinds of certificates: Endorsement Certificate, Platform Certificate and Conformance Certificate.
   - **Endorsement certificate**: The main purpose of this certificate is to ensure the integrity of the Endorsement Key. This certificate can be provided by the same issuer as the EK but it is not mandatory. The trust on the TPM is based on the fact that the EK is unique for each TPM and the certificate protects the key from identity theft at all time.
   - **Platform certificate**: This certificate is provided by the platform vendor. The main purpose of this certificate is to ensure that all the security components provided with the platform are genuine. This certificate enables the platform trust.
   - **Conformance certificate**: This certificate is provided by a third party evaluation lab or by the platform vendor itself. It certifies that the security properties claimed by the manufacturer are genuine.
4. *Platform Configuration Registers (PCR)*: PCRs are used to bind a critical data to a specific device. This binding process that disables the data reuse on another device relies on the authentication of the device. Specific hardware and/or software configuration information is used by the TPM to calculate a particular value stored in the PCRs. This value is strongly linked to a unique device and is used to authenticate it while an entity tries to access the data. If the authentication failed, the data cannot be accessed.

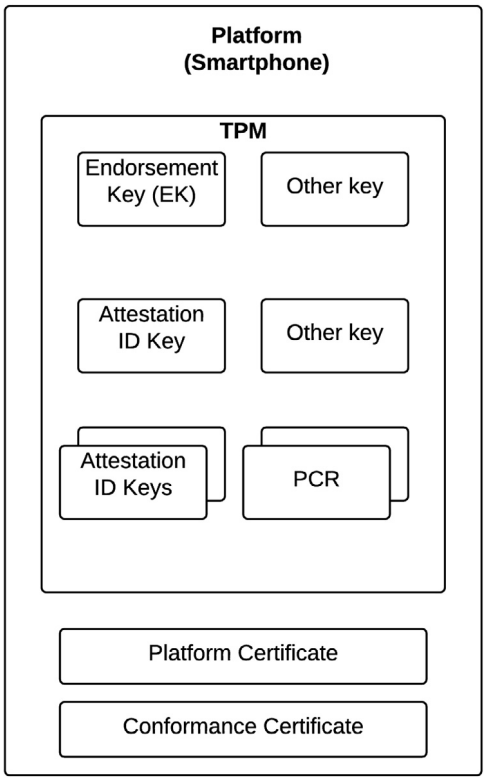

Fig. 4. TPM architecture.

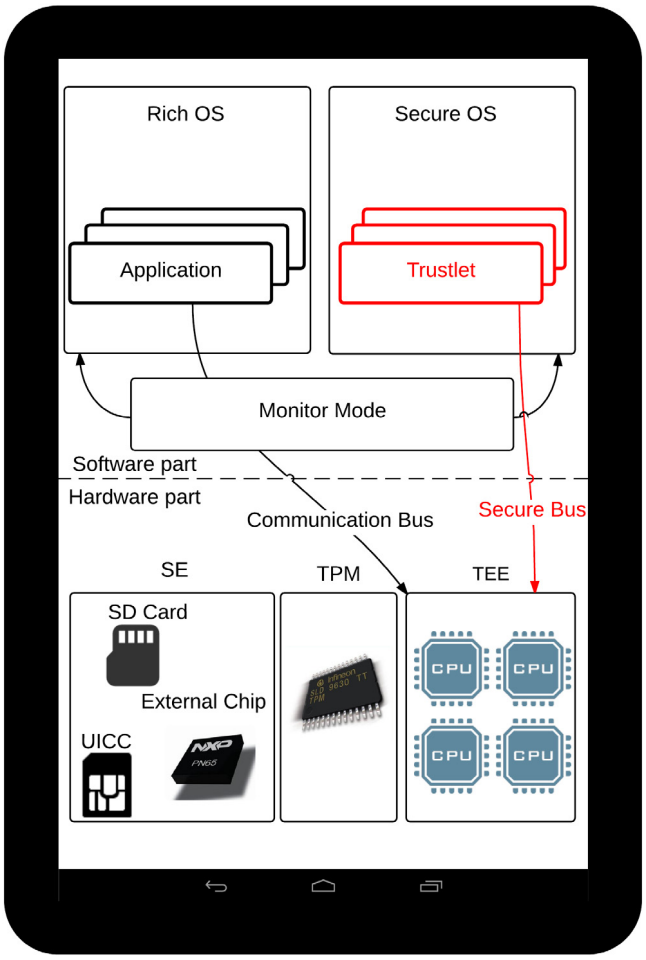

Fig. 5. TEE components overview.

5.2.3. Binding critical data to a platform with TPM

Binding critical data to a device is equivalent to make this data unusable by another platform. This binding is implemented by using specific hardware/software configuration information about the platform. PCRs are used to store the measurements calculated by the TPM.

The critical data are merged with a number of PCR. The result of this operation is encrypted and stored on the TPM. Later, if any critical data has to be accessed, the encrypted data are decrypted and the platform calculates the value of the data and the value of the measurements used in the binding process. This operation allows to extract the critical data from the whole information decrypted.

As a use case, a TPM can be used as a PKI that provides the required keys and certificates for establishing secure communications and signing documents. In the mobile context, TPM can be very useful to ensure the security of a system with heterogeneous stakeholders (device manufacturer, mobile operators, users). Indeed, with TPM mobile specifications, it is possible to have more than one active instance of a TPM in a device.

5.3. Trusted Execution Environment (TEE)

This solution was developed to address the compatibility and performance issues raised by the use of SE. In this section, a detailed description of the TEE technology highlighting its advantages and disadvantages will be provided before making a comparison with the SE.

5.3.1. General architecture

TEEs [3,28,29] are a combination of a hardware and a software parts. Moreover, the system is divided into two execution environments.

- The first environment is the Rich Execution Environment (REE) also called Normal World Execution Environment. This first environment represents the standard OS of the smartphone such as Android for instance. The term Rich describes the extensive features of the OS such as camera management, telephony capabilities and so on. These features significantly increase the attack surface.
- The second environment is the Trusted Execution Environment. It represents the Secure OS responsible for performing sensitive operation such as cryptographic operations. It also has the capability to secure the display and the input by using a secure mode of the buses connecting the processor to the I/O peripherals. OP-TEE and OPEN-TEE presented in Section 5.3.3 are examples of Secure OS.

5.3.2. TEE components and characteristics

The TEE provides a processor zone with a secure OS and other mechanisms to enhance the security of sensitive data processing as depicted in 5. Indeed, the TEE splits the processor in two zones: Normal World for the classical OS applications and the Secure World for the sensitive applications.

The other essential feature is the secure storage. Indeed, in the case of cryptographic operations, the key generation and storage must be secure. The key has to be on the phone but isolated from the user data to enhance its security.

In opposition to the SE, the TEE provides a secure communication channel between the processor and the external peripheral especially input and display. The TEE uses a secure channel for the

Input/Output operations while the SE is relying on the default operations provided by the Rich OS. It is a very important feature because if a malicious application can intercept the input, it can have access to sensitive data like pass phrase. The secure display is also primordial because the user has to be sure that what he sees on the screen is really sent by the secure world. We will discuss how this is implemented in a next section.

A simple example to illustrate the difference between SE and TEE capabilities is the user PIN code input. Indeed, the SE relies on the Android platform to get such sensitive information. Many attacks [30,31] make an attacker able to collect and record the user provided PIN code. For its part, the TEE uses secure I/O operations which makes these attacks unsuccessful.

A secure boot process is also needed. It enables Rich OS and Secure OS integrity checking. The secure process follows the steps below:

1. Read a trusted ROM (locked at manufacturing),
2. Signature and integrity checking of the Secure OS,
3. Setting up the Secure OS which takes the control.

In the TEE world, two heterogeneous systems coexist and therefore it is necessary to set isolation rules between them to avoid data leak. Thus, we need a third mode called Monitor mode. This mode is used to perform context saving and switching between the Rich OS and the Secure OS. This mode is only accessed by the Secure OS to request a context switching when all the operations are performed.

The Secure OS is a limited instruction set OS. This constraint is necessary to reduce the attack surface. It schedules sensitive applications running on it. These applications are called trustlets. Trustlets execute secure instructions like cryptographic operations: key generation, encryption and decryption. The Secure OS manages the resources between all the trustlets. A trustlet is a secure application which runs on the TEE. It must be signed by the chip manufacturer and signature verification is performed before loading on the TEE. Some APIs can be added to manage extra features but it has to be ensured that these features will not introduce security breaches.

So far, the majority of Secure OS used for TEE are proprietary which makes it difficult to get access to test your trustlet. In fact, if a third party wants to get access to a TEE to test its application, this application has to be signed with the private key of the manufacturer. The manufacturers are reluctant to sign third party applications. This is the biggest drawback for TEE expansions. Developers continue to advocate for change access rights to TEE to deploy their applications.

5.3.3. Secure OS overview

At the time of writing, several implementations of Secure OS for TEE are available. We can list Sierra TEE, Genode, Trusted Language Runtime, OP-TEE and TrustKernel T6.

- **Sierra TEE**: According to the OS maker [32], the Sierra TEE OS performs an integrity management process and several scanner checks. Among these scanners, Sierra TEE checks the integrity of the Android file system, the Android OS, processes running and the interrupt table. These checks are performed to ensure the security of the Android execution. It also allows to perform key management, device management, DRM and I/O operations securisation. Indeed, for secure input/output, the peripherals communicate directly with the TEE. In this case, the Android OS cannot intercept passwords and sensitive information from I/O operations.
- **Genode**: Genode is a secure OS with a very low complexity [33]. Its source code is approximately 10,000 lines. This feature is very important as it allows a simple security verification of the OS. Moreover, Genode manages the execution of touchscreen driver and hardware as well as the frame buffer driver. These executions are run in the secure world ensuring secure interactions between the TEE and the user. Genode also controls the textboxes used to collect user passwords and screen used to display secure information from the TEE. This OS ensures that no information leakage occurs.
- **Trusted language runtime**: This OS [34], based on a.NET platform, relies on a multiple trustboxes[2] system. Each trustbox is like a container isolated from the others. The device manufacturer initializes a pair of key (public/private) in a trustbox to allow remote verification and attestation. For now, TLR is not able to communicate securely with users. This allows attackers to intercept, modify and alter the data flowing between the users and the TEE. The I/O operations are managed by the REE.
- **OP-TEE project**: OP-TEE [35] is a secure OS for TEE developed by STMicroelectronics in collaboration with LINARO [36] that is fully compatible with the GlobalPlatform specifications [3]. OP-TEE consists of a client API, a Linux kernel driver and a Secure OS. As mentioned previously in this section, the switch between the Rich OS and the Secure OS is managed by a monitor mode. It has also a multi-core capability. The unique constraint is that only one core can be in the secure world at the same time.
- **TrustKernel T6**: T6 is a secure OS for ARM processors with TrustZone capability. It can run simultaneously the secure OS with one of the multiple Rich OS supported (Android, Linux, etc.) [37]. It provides strong security properties and enhances the ease use. The TrustKernel team provides all the source code and the support. T6 is provided with multiple libraries like LibC and OpenSSL to facilitate the user application development. Finally, T6 is fully compatible with the GlobalPlatform specifications [3].

Like the SE, the TEE is based on an ICOM model since most Secure OS are proprietary. But unlike the SE, we observe an emergence of open sources TEE like OP-TEE. This is a great opportunity to extend the use of this technology and to allow developers to design new architectures based on TEE like those exposed in Section 7.

5.3.4. Comparison between SE, TEE and TPM

Table 1 shows the main differences between SE, TEE and TPM. It can be noted that the SE and the TPM have more physical security features than TEE. However, TEE provides a bigger computation power which enables more complex security models.

Table 1
Comparison between the hardware solutions.

Criteria	SE	TEE	TPM
Tamper resistance	✓		✓
Secure input and display		✓	
High computation power		✓	✓
High storage capacity		✓	
Dependency to manufacturer	✓	✓	✓
Proven security level	✓	✓	✓

6. Software-based solutions

In the previous sections, several trusted mobile computing approaches have been presented. These approaches have in common to rely on a hardware component to ensure trust and

[2] A trustbox is a runtime environment which protects the confidentiality and the integrity of code and datas.

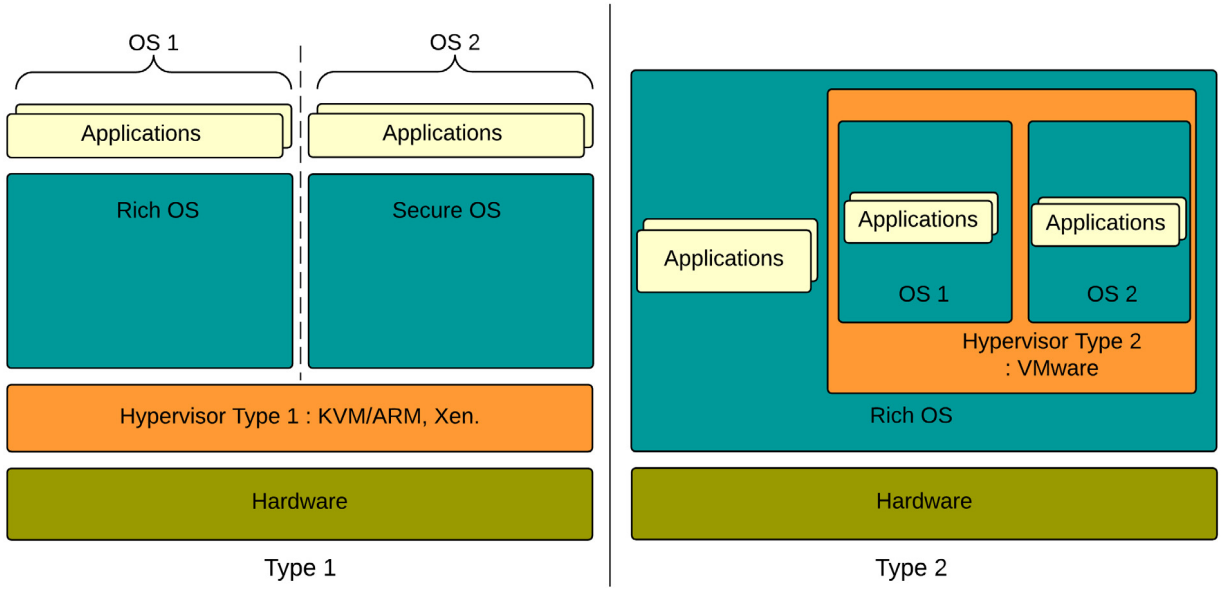

Fig. 6. Virtualization types 1 and 2.

security. Unfortunately, most of these components are proprietary limiting their usage to an ICOM model. For a user-centric perspective and in order to fit to the UCOM deployment model, open approaches are required. One of them consists of using software components instead of hardware ones. More precisely, it consists of virtualizing OSs in order to provide a trusted and secure operating system that enforces the requirements described in Section 6.1. Indeed, the new generation of smartphones is now able to run several virtual OSs on the same device. Each virtual OS has a new and parallel stack with all features of a regular OS. Furthermore, to prevent malicious applications installed on a Rich OS from stealing data in a Secure OS for instance, an isolation has to be set up between the different OSs. As presented in Fig. 6, there are two different kinds of virtualization process at the time of writing. For type 1 virtualization, the Rich OS and the Secure OS are physically independent while in type 2, the Rich OS acts as a host to the Secure OS (which can be viewed as a guest). To achieve this partitioning (this isolation) and to guarantee the security of both host and guest, a dedicated software component is needed: the hypervisor. In the remaining of the section, the overall virtualization process and several existing solutions for trusted mobile computing will be presented.

6.1. Virtualization mechanism design

The aim of such a mechanism is to provide a complete smartphone stack by virtualizing the system at the hardware or software level. To execute the guest OS, a virtual machine is provided. To fulfill its mission, the hypervisor must meet the following requirements:

- **Portability**: The hypervisor has to work on large variety of mobile devices. Some parts of the smartphone are more difficult to virtualize because they may contain specific elements such as dedicated processors or memory for the GSM part. Each processor has its own architecture and several cores. Also, each smartphone has its own set of interfaces and programming model. This implies that the location of the hypervisor is very important. To be compatible with most smartphones, many constraints must be taken into account.
- **Compatibility**: The aim of a hypervisor is to create an abstraction layer between the hardware and the system when the virtual machine is running. The hypervisor has also to provide a solution against compatibility issue. Indeed, many applications are dependent of a specific OS system or a specific library. It is also important to allow the reuse of already developed applications and ecosystem.
- **Security**: To address security issues, the hypervisor must be designed to allow secure application provisioning on the guest OS. The main security mechanism is also a perfect isolation between the virtualized OS and the host.
- **Low complexity**: The hypervisor must be reliable. To ensure this characteristic, we need to reduce the complexity of the hypervisor. This feature allows to have a better safety and also ensures a better maintainability.
- **Performance**: The applications executed in the guest OS should keep the same level of performance as applications in the host OS. For mobile devices, the battery is the only source of energy. In order to monetize the virtualization process, the battery life should not be unreasonably affected.
- **Manageability**: In the virtualized system, it is necessary to be able to install applications, delete, erase the memory including remotely if needed. If a company provides a virtualized smartphone to workers, the IT department should be able to manage the virtualized secure OS remotely.

6.2. Architecture of a virtualized system: VMware proposition

To provide a powerful system with a minimum overhead, VMware proposed a type 2 or hosted hypervisor [38]. This hypervisor is installed on the Android system and can execute an additional virtualized system alongside the host. A hosted hypervisor is also a powerful solution to address the problem of hardware diversity. Indeed, a minimum overhead is needed to be compatible with a large variety of hardware. A user can use its own device with an hypervisor to execute the hosted OS. This solution does not interfere in the smartphone development cycle.

The processor uses the multiplexing time paradigm[3] between the host and the virtual machine. To toggle between the two OS, the processor performs a context switching by giving the control

[3] This paradigm allows the host and the virtual machines to share a physical processor during their execution.

to one of the two machines. For the host, the guest OS execution is similar to a thread execution. The main challenge is to virtualize the processor cores including the specific instructions set for each brand (TI, ARM, Intel), type and the memory management and sharing between the host and the guest. Barr et al. in [38] also identified major issues related to virtualization: the compatibility with instruction sets issue, the memory sharing issue, the platform virtualization issue, the storage issue, the network access issue, the multiple telephony lines issues and the security issues. In the remaining of the section, these different issues will be detailed as well as their respective solutions.

6.2.1. Compatibility with instruction sets issue

The main purpose of this solution is to avoid any changes in the guest code for applications and middleware and to minimize changes in the hardware definition. The complexity of the system is reduced and the portability of the different guests is improved.

6.2.2. Memory sharing issue

The host and the guest share the same memory. On a smartphone, there is two kinds of memory: volatile memory (RAM) and persistent memory (intern storage or SD cards). To address the memory sharing issue, two sub-layer components have to be virtualized: Memory Management Unit (MMU) and the physical SDRAM memory on the device. The virtualization of the MMU provides an abstraction of the addressed memory space. This virtualization includes the processor address translation and memory protection. The Virtual Machine Monitor maintains a mapping between guest physical addresses and host physical addresses. As the guest is running in user mode, a protection is set up to avoid attempts to access privileged registers results. Shadow copies of these privileged registers are maintained and are used as backup copies.

6.2.3. Platform virtualization issue

To virtualize a smartphone with keeping all of its capabilities, it is necessary to virtualize the mobile platform. Indeed, to allow the guest to access the network and several other applications (GPS, VoIP), the guest has to get an access to the physical antennas and hardware.

6.2.4. Storage issue

The host memory is used to store at the same time the hypervisor and the VM images. VM images are too large to be stored on the flash memory, then the best solution is to move them on the SD card. The hypervisor is installed on the built-in flash memory. The challenge is to guarantee a high performance level with integrity checking.

6.2.5. Network access issue

Smartphones use the TCP/IP stack to communicate over wireless networks. In the virtualized device, this stack is emulated by a high level framework called para-virtualized TCP. This framework is acting at the socket system level in opposition to the device interface level. This approach is interesting because it minimizes virtualization overhead and allows more flexibility in the deployment process. This architecture is divided into two modules: the client module and the offload engine. The client module is running on the guest and intercepts at runtime all the network requests made by the guest. The offload engine runs on the host and receives the intercepted requests on the guest. This traffic is tunneled to satisfy the security model requirements in the guest side.

6.2.6. Multiple telephony lines issues

The main feature of a smartphone is to allow people making calls. For a secure virtualized system, we must distinguish between the virtualized phone and the native phone. A first basic solution is to have smartphone supporting two SIM cards. This kind of smartphone exists but it is not the most spread model. Alternatively, we can use SIM cards supporting different lines. Indeed, we can have two phone numbers over one SIM. Another approach consists of providing multiple IMSI (International Mobile Subscriber Identities) to a single SIM card allowing the mobile to appear as a set of devices to the network. Therefore, each virtual machine can hold its own phone number.

6.2.7. Security issues

Designing such a virtualization process has to take in account some security properties. Indeed, the isolation between the two environments, the sensitive data protection and secure communications must reach a high level of security. Moreover, physical threats issues have to be addressed to avoid sensitive data leaks if the smartphone is lost or stolen.

Another significant threat is untrusted downloadable applications and malwares. They can induce undesirable behaviors and may introduce some vulnerabilities exploitable by attackers. With a system based on permissions like Android, the user is an important part of the overall security system. He has the power to accept or deny any rights for an application installation. But users may not be aware about the risks incurred by according some sensitive permissions to applications.

For these reasons, the management of applications on the guest OS has to deal with software attacks. In order to achieve this, good practices are needed such as:

- *Restriction of the host application scope*. Indeed, host applications cannot gain access to the guest data including address book, SMS and all data stored on the guest. The guest network access and telephonic capabilities are also protected.
- *Protecting the guest traffic from interception*. A tunneled traffic is a reasonable solution to address this issue. In order to achieve this, the negotiation phase should not be intercepted by the host.
- *Protecting the data on the SD card*. Data hosted on a SD card can be accessed by any application on the host device as well as by potential attackers. Moreover these SD cards are removable from the host so that physical theft has to be dealt with. Using encryption is a solution to make the data unreadable by unauthorized applications and attackers. However, the security of this solution relies on the privacy of the used encryption key. Once again, the question of the key storage arises. Several solutions have been proposed to address this issue among which is the solution proposed by Cooijmans et al. in [17] presented in Section 7.2.
- *Switching between the two environments should be protected*. A basic solution consists of using passwords. If the guest is idle for a certain period of time corresponding to the session time defined, it should be deactivated.

6.3. KVM/ARM: exploit the virtualization extension of ARM processors

KVM/ARM is the first full system ARM virtualization. It can run an unmodified guest OS on an ARM architecture. Since the version V7 of the ARM architecture, the virtualization support is an optional extension [39]. Thus, the virtualization process can rely on these extensions to build a new kind of hypervisors. ARM introduced a new mode called HYP mode. This HYP mode is strictly more privileged than the other CPU modes. It is invoked for the most sensitive system calls. For these reasons, the hypervisor must at least partially reside in this mode.

The virtual machines (VM) run in the user land or in the kernel land. When these VMs need to execute a sensitive operation, the

HYP mode is invoked and takes the control. Once the operation is finished, a switch back context is performed to return in the caller mode.

To minimize the context switching time and therefore to limit the impact on the system performances, not every system call and memory page fault are managed by the HYP mode. This option contributes to limit the virtualization overhead.

To facilitate hypervisor development process, ARM wanted to reduce control registers available on the HYP mode. We will expose in the following the ARM virtualization extension capabilities.

- **Memory virtualization**: When a VM is running, the physical addresses managed in the VM are called Intermediate Physical Addresses. They are also called guest physical addresses. They need to be translated into physical addresses also called host physical addresses. This translation is performed to make the guest operation executable by the physical processor. The HYP mode is responsible for managing this translation process.
- **Interrupt virtualization**: A Generic Interrupt Controller (GIC) was defined to route the interruptions from devices to CPU and discover the source of the interrupt.
- **Timer virtualization**: A counter was introduced to measure the time passed in real time and a timer for each processor. This counter and timers are used to raise an interrupt to the CPU after a certain period of time. To ensure the isolation between the VMs, the timers used by the hypervisor cannot be managed by any guest OS. Based on the ARM Virtualization extension, Dall et al. proposed an hypervisor in [40]. The specificity of their hypervisor is that it is split into two modes the HYP mode and Kernel mode.

6.3.1. Split-mode virtualization

Running the hypervisor on the HYP mode is both attractive and problematic. Indeed, significant changes had to be made on the Linux Kernel to run it in the HYP mode. Secondly, this hypothesis may affect the native performances of the system. To address this issue, Dall et al. in [40] proposed a *split-mode virtualization*. The hypervisor can run in different CPU modes which make it able to benefit from every mode advantages. The KVM/ARM hypervisor takes advantage from ARM virtualization support and leverages at the same time existing Linux services in kernel mode. This approach minimizes the changes on the Linux Kernel which allows greater community acceptance.

The hypervisor is split into two components: Lowvisor and Highvisor. The lowvisor exploits the ARM virtualization extension available in HYP mode. It is responsible for the following operations:

- The lowvisor loads the correct execution context by setting up the right system configuration.
- The lowvisor switches between the VMs and the host. Its execution in the HYP mode makes it able to perform such switching.
- The lowvisor handles interruptions and system exceptions.

The high visor runs in the kernel mode of the Linux kernel. It can use the standard Linux kernel functions like processes scheduling, memory allocation etc.

6.3.2. CPU virtualization

The KVM/ARM model gives an interface to the VM which is essentially identical to the hardware CPU including a persistent access to state registers of the physical CPU. The processor will stay in the VM context until an event occurs which triggers the context switching via HYP mode.

- Memory virtualization: The memory virtualization is performed by translating the virtual addresses[4] used in the VM for all the memory accesses requested by the VM. The address translation can be only configured in the HYP mode. The high visor manages the accesses to the memory specifically allocated to VM. Any other entity that tries to access other memory zone will cause a page fault. This fault will be managed by the hypervisor. This mechanism ensures the isolation between the VM themselves and between the VM and the host.
- SierraVisor: SierraVisor [41] is a flexible and reliable hypervisor for ARM processors. It supports three modes of operation: Paravirtualization for ARM TrustZone enabled devices, paravirtualization for ARM 11 and Cortex A9 and a hardware virtualization for Cortex-A15. These three separate modes allow the equipment manufacturer to choose the best virtualization solution according to the processor they use.

6.4. Xen project hypervisor

Xen [42] exploited the ARM virtualization extension to build its own hypervisor for ARM processors. This hypervisor resides entirely in the HYP mode which minimizes the context switching costs. It also uses a virtual memory assigned to the virtual machine and a translation is needed to get the physical memory addresses. The Xen hypervisor provides also timer and interrupt management to harmonize the scheduling between the virtual machines and the host.

6.5. Hybrid model for virtualization

Virtualization is based on a hybrid approach. Indeed, in the host side, a UCOM model is adopted while an ICOM model is used for the guest side. This approach is very interesting. It allows the smartphone's owner to get all applications and functionalities he wants in the host side without interfering with the sensitive data on the guest. This solution is ideal for getting rid of the manufacturers constraints that occur on purely hardware solutions.

7. Existing solutions based on trusted platforms

The paradigm of trusted computing has attracted many researchers. In order to illustrate the usefulness of each technology, we expose architectures exploiting secure environments on smartphones to ensure secure storage and execution. The On board Credentials (ObC) solution is a TEE architecture designed to open TEE for application developers and the Secure key storage on Android addresses the secure key storage issue presented in Section 4. The two solutions use a TEE to ensure the security of the credentials in two different OSs: Windows 8 and Android. SGX is a distributed TEE architecture developed by Intel for x86 processors that is promising for the future of trusted mobile computing. The fourth solution, Cells, is an implementation of a type 2 hypervisor (see Section 6.2) with some specificities like the deployment process. The fifth solution consists of a cloud based TPM platform that illustrates the concept of Section 5.2. Finally the two last studies aim to provide examples of open source or virtualized TEE solutions like OP-TEE that we installed and tested.

7.1. On board Credentials (ObC)

ObC [16] is a project that was developed by the Nokia research center aiming to ease the use of hardware TEE with the

[4] Virtual addresses have to be translated into physical addresses to execute data on the processor.

introduction of a new provisioning system for trusted applications. With ObC, an application developer will only need the mobile phone user's agreement to provision a trusted application in the TEE instead of requesting the device manufacturer or the OS provider authorization. Moreover, an API is defined allowing the developer of application to easily develop both the REE part and the TEE part of his application. To achieve this, ObC's TEE architecture relies on two parts: The ObC interpreter and the ObC scheduler.

- The ObC interpreter (aka ObC VM) is a small virtual machine that provides a runtime environment for trusted application that can be viewed as a lightweight OS. This interpreter can run either as a trusted operating system on top of an hardware TEE or as a trusted application in an existing trusted OS of a TEE. In the latter case, the interpreter increases the features of the underlying trusted OS with additional provisioning capabilities and enforced trusted applications executions' isolation. As a counterpart, it can be noted that these additional features reduce the implementation minimality of the trusted OS which could increase its overall attack surface.
- The ObC scheduler is running in the REE (aka the phone's native OS). It aims to manage the trusted application life-cycle and interacts with the ObC interpreter to do so. More specifically, the scheduler is in charge of providing the ObC interpreter the encrypted trusted applications as well as their inputs or their previous states. It must be noted that the trusted applications managed by the scheduler are always encrypted. Finally, the scheduler can also be used to access the cryptographic features offered by the TEE via the interpreter.

ObC can be viewed as an interesting attempt to open proprietary implementations of TEE to application developers and can be viewed as a good solution for the UCOM model. Nonetheless, it has been developed with a focus on Trustzone TEE recommendations that may limit its usage to this architecture and is only available for Nokia devices which is not fully compliant with a user-centric approach. Another limiting point is the overhead induced by the communication between the scheduler and the interpreter. Indeed, this context switching between REE and TEE as well as the trusted applications' encryption/decryption operations performed by the provider in the TEE are costly.

7.2. Secure key storage on Android

In [17], Cooijmans et al. discussed the secure key storage issue on Android. They exposed some attacks scenarios and evaluated the proposed solutions to store the cryptographic key in a secure way. First, they introduced a purely software solution that consists of an Android library existing since the API 1: The Bouncy Castle. Second is the Android keystore available since API 18. They tested this keystore with two processors with TEE capabilities: A Qualcomm and a Texas Instrument MShield.

Their experiments show that no solution is able to totally secure the encryption keys storage. However, solutions with the use of TEE are more reliable and provide a better storage solution. Indeed, they grant a protection against attacks that the purely software solutions cannot fight against.

To conclude, Cooijmans et al. provide some recommendations to increase the security level of the keystores. These recommendations are summarized here:

- Encrypt all the keystore files with a generated key or a user provided key.
- Include the user ID of the application that generated the key pair in the integrity checking of the keystore.

These recommendations do not require any system nor architectural changes.

7.3. Software Guard eXtensions (SGX)

SGX (Software Guard eXtensions) [43–45] is a set of instructions proposed by Intel for its x86 processors that allows the execution of applications (or parts of an application) in a secure container called an Enclave. An Enclave provides a protection against other applications or privileged system software such as the OS, hypervisor or even BIOS. Indeed, SGX holds a memory region called the Processor Reserved Memory (PRM) that holds the Enclave Page Cache (EPC). The EPC stores enclave pages that embed sensitive data as well as SGX data structures. The EPC region is encrypted so that only the associated Enclave is able to access the stored data. The Enclave Page Cache Map (EPCM) that is stored in the processor maintains this association.

The main difference with TPM is that in SGX the whole application does not have to be stored securely. Only private data and the applications code that operates on it has to be stored in the Enclave. To upload these sensitive data, a software attestation mechanism is provided that relies on a cryptographic signature based on a SHA-256 digest. This digest certifies that the identity of the software is verified, before the codes execution, against the identity enforced by a trusted third-party authority.

For prototyping purposes, OpenSGX [46] that is an open source emulator based on QEmu like OP-TEE has been proposed. It can be used to test a secure application before being deployed on a distributed architecture. Indeed, SGX also provides a remote attestation feature that allows an Enclave to check the identity of another Enclave running on a remote host. This original feature allows to consider SGX architecture as a distributed one where several distributed enclaves can cooperate.

But, regards to trusted mobile computing it must be noted that at the time of writing, SGX is still not available on smartphones processors.

7.4. Cells: A virtual mobile smartphone architecture

Cells, proposed by Andrus et al. in [47], is a virtualized platform that allows several virtual Android smartphones to run on the same physical device. Cells guarantees the isolation between the physical smartphone and the virtual phones and between the virtual phones themselves with performances close to native.

Cells does not run multiple instances of the Android OS. Indeed, it is able to run multiple Virtual Phone (VP) on a single instance of the OS. It maximizes the sharing of common read-only code and data to the VP. This approach minimizes the memory usage and therefore minimizes the overhead of running virtual phones on a physical smartphone.

To provide an individual phone service to each virtual phone, Cells uses a VoIP service to overcome the constraint of having as many SIM cards as VP. However, incoming and outgoing calls when using the cellular network are routed to the VoIP service which shift the communication to the right caller ID. Indeed, the VoIP service acts as a proxy between the caller ID and the cellular network.

Andrus et al. claim that they were able to run five virtual phones on the physical smartphone with minimum performance overhead. They also report that Cells is fully compatible with hardware devices with native performance including sensors, cameras, touchscreens, etc. The applications using these devices are run transparently and no modifications are needed.

In order to add a VP to a smartphone, the creation and the configuration are done on a computer. The resulting VP is transferred to the device via a USB port.

An important security mechanism implemented in Cells prevents an attacker who has achieved an exploit to perform a privilege escalation to do the same on the other VP and the physical

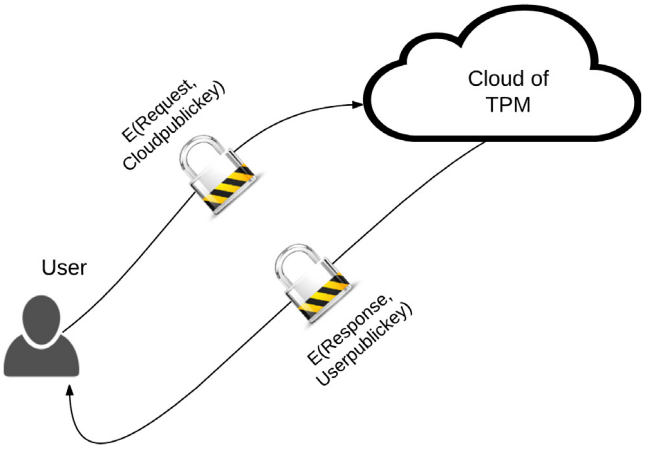

Fig. 7. Communication with a cloud of TPM.

smartphone. This is guaranteed by the complete isolation between the two environments.

Cells leverages the intern memory to allow sharing the memory between the VPs. This mechanism includes read-only and read–write zones. The results of the experiments show that a minimum overhead is added by the deployment of VPs on Nexus 1 and Nexus S. This solution is promising for the deployment of business phones on a personal phone while guaranteeing the security of processing and storage of data.

7.5. A cloud architecture of virtual trusted modules

Liu et al. in [48] proposed a cloud of TPM which can be used by a remote device. It consists of a cluster of physical TPMs. Each service is represented by a virtual port on the cloud. A request/response protocol is deployed to communicate between users and the cloud.

Secure communication protocol is necessary to exchange data securely between users and cloud. Each user and the cloud have a unique public/private key pair. Fig. 7 represents the communication scheme between users and the cloud.

As described in Fig. 7, the user builds a request for the cloud that is an aggregation of the operations' set to execute in the cloud and of his public key. The request is encrypted with the cloud public key and sent on the network. Upon reception, the cloud processes the user operations and generates the corresponding results. These results are encapsulated into a response message encrypted with the received public key. Finally, the encrypted response message is sent back to the user. The overall system performance makes the user believe that a local TPM is under use instead of a cloud of TPM allowing its usage for smartphones that are lacking of a TPM chip.

This cloud-based solution is a good solution to extend the usage of TPM. A similar approach has been proposed by Urien in [49] for a cloud of SE. But, these solutions are only available if the mobile devices have a full connectivity to the cloud which is not always possible neither suitable. Another limitation pointed out by Vinh et al. in [5] is that the cloud has also to be secured in addition to the mobile device. Therefore, the number of potential vulnerabilities is growing with cloud-based solutions.

7.6. OPEN-TEE an open virtual trusted execution environment

OPEN-TEE [50,51], is a complete virtual TEE conforming with the GlobalPlatform specifications. It allows to write TEE applications and to test them on a full software platform without a physical access to a hardware TEE. The objective of the authors was to elaborate an SDK and framework granting the development and the test of trusted applications. This SDK must require the minimum configuration and maintenance as possible.

The authors choose to design their framework as a set of components. The main ones are:

- Base: It consists of a daemon process in the user mode responsible for the configuration and the preparation of the system.
- Manager: It can be considered as the OS of the OPEN-TEE. Its main responsibilities are to manage the connections between applications, to monitor trusted applications states, to share memory between the applications and to provide a secure storage area.
- Launcher: The main purpose of this component is to create and launch trusted applications.

Other components are used such as: Trusted applications process, GlobalPlatform TEE APIs and IPC communication component.

The OPEN-TEE SDK and framework are currently under tests by several organizations in order to validate the results obtained in the study.

7.7. Virtualizing the trusted execution environment

Vahidi et al. in [52] propose to design and implement a hypervisor for the U8500 NovaThor platform [53] which operates in the Secure World inside the CPU (i.e. with TEE capabilities). The main purpose of this proposition is to ensure the isolation between the trustlets themselves and between the trustlets and the Rich OS Applications.

This solution is based on the concept of the virtualization including the TEE part. For their platform, the authors choose to implement a type 1 hypervisor with a very small code and very high performances.

Fig. 8 shows the selected architecture and the identified data flows. Flow number (1) is the invocation by a client application of an API function in the TEE. Flow number (2) represents a call from the TEE secure OS to a Trusted application that performs a specific processing. Finally, flow number (3) is the communication between the Secure OS and the hypervisor that allows the use hardware components when needed.

The security of the overall solutions mainly relies on the security of the hypervisor which is expected to be high. But, as stated by the authors this security has still to be formally verified before formal validation.

7.8. OP-TEE solution

OP-TEE is a Linaro and STMicroelectronics project with the objective to release a totally virtualized and open source TEE. Indeed, in OP-TEE the platform is either virtualized by using Qemu[5] [54] or FVP[6] [55] or consists of a physical development card of ARM called JUNO. For the virtualized platform, a Debian based OS is needed to download and install OP-TEE. The OP-TEE implementation follows the GlobalPlatform standards. We can execute a trusted application on OP-TEE by loading the application in the Rich OS side first. When the application needs to perform a sensitive operation, it uses the API to invoke this operation in the Secure World side (TEE). The TEE sends the response to the REE side for display or other purposes. A debug and web interface are provided to facilitate the management of the platform. Fig. 9 represents our architecture of OP-TEE. We use FVP to virtualize OP-TEE which can be run on the top of a Debian OS.

Fig. 10 represents an execution of a trusted application which requests the hash of '*abc*' with the SHA-1 and SHA-256 algorithm

5 Qemu is a generic and open source machine emulator and virtualizer.

6 Fixed Virtual Platform.

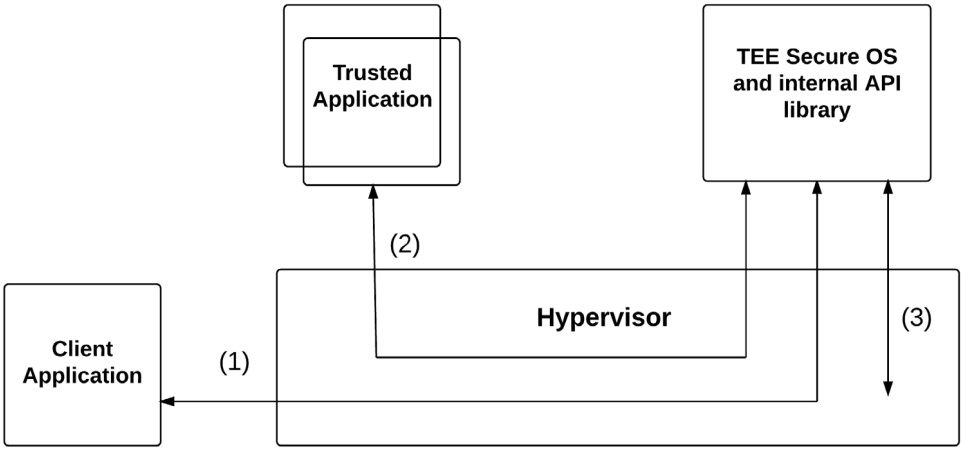

Fig. 8. TEE virtualization architecture and data flows.

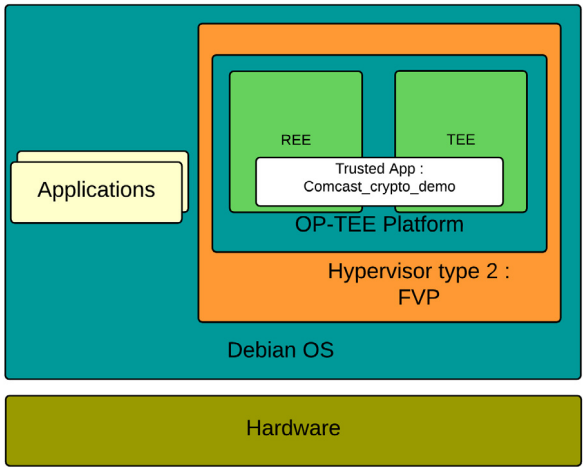

Fig. 9. OP-TEE architecture.

to the TEE. We observe that the REE establishes a communication with the TEE and sends the data and the operation for processing. After that, the TEE answers by sending the result to the REE world to display.

The sequence diagram represented by Fig. 11 gives us more details about the interaction between the Rich OS and the Secure OS in the precedent example.

The OP-TEE platform allows developers to deploy their applications following the UCOM model when the hardware TEE is following an ICOM model.

8. Comparative analysis

8.1. Analysis of the robustness of the secure environments against attacker models

Following the general presentation of existing hardware and software solutions, a comparison of their behavior for each attacker model defined in Section 3.2 will be made. The main results of this comparison are presented in Table 2.

It can be noted that the SE and the TPM have exactly the same behavior against the attacker models. Indeed, the chips are independent from the applications that run on the smartphone. For this reason, the Malicious App Attacker and Root Attacker models are neither able to corrupt nor to intercept the data with these two hardware solutions. On the other hand, the Intercepting Root Attacker controls the smartphone Input/Output mechanisms. In this case, the interaction between the SE or TPM and the user creates a security hole as data sent or received by the user can be intercepted by the attacker.

The main difference between the TEE and the SE and TPM is that the Input/Output process is secured. Then, no data leak can happen by sending or receiving data as explained in Section 5.3. The TEE is able to resist the three attackers.

The virtualization is sensitive to root attacker and intercepting root attacker. Indeed, the guest OS and the host OS are sharing the same physical device. Therefore, if an attacker can be root on the host OS, he can be able to intercept the data sent to the guest. Hence, a data leak is possible and can have consequences on the security and privacy on the guest.

After analyzing these solutions, we can observe that only the TEE is able to resist to all the three attacker models. The physical and the logical environment combined with the secure I/O operations is the best of the four discussed solutions.

8.2. Comparison of trusted mobile computing approach for user-centric solutions

Our study has led us to propose in Table 3 a synthesis of the main differences between the trusted mobile computing solutions presented in the previous sections. It can be noted that the hardware solutions are not sensitive to physical attacks (PA) or logical attacks (LA) while the hybrid solution or pure software are. ICOM provisioning model prevails with an exception for virtualization. This model limits the growth of these technologies. The issuers are not willing to free the access to their security elements to developers.

The overhead induced by the security components is an interesting criteria to classify and prioritize the security solutions. The physical solution (SE, TPM, TEE) does not cause any overhead because they use their own hardware. In the other hand, the virtualization can induce an overhead which is due to the fact that the guest OS and the host OS are sharing the same hardware. Hence, if the performances of the system is an important criteria in the deployment of a secure solution, the hardware-based solutions are more suitable. Another important criteria is the computing and storage capacity. For the SE and TPM, this capacity is limited because the chip does not have a large memory. For the TEE and virtualization, they can have the full power of the smartphone.

We observe that almost all presented technologies follow the ICOM model. It is motivated by the high security level guaranteed

```
root@FVP:/ comcast_crypto_demo
Opening the session to Comcast RDK TA
Invoking a function in Comcast RDK TA
Hash of message: 'abc' is:
a9993e364706816aba3e25717850c26c9cd0d89d
Opening the session to Comcast RDK TA
Invoking a function in Comcast RDK TA
Hash of message: 'abc' is:
ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad
root@FVP:/
```

Fig. 10. TEE SHA-1 and SHA-256 hash of '*abc*'.

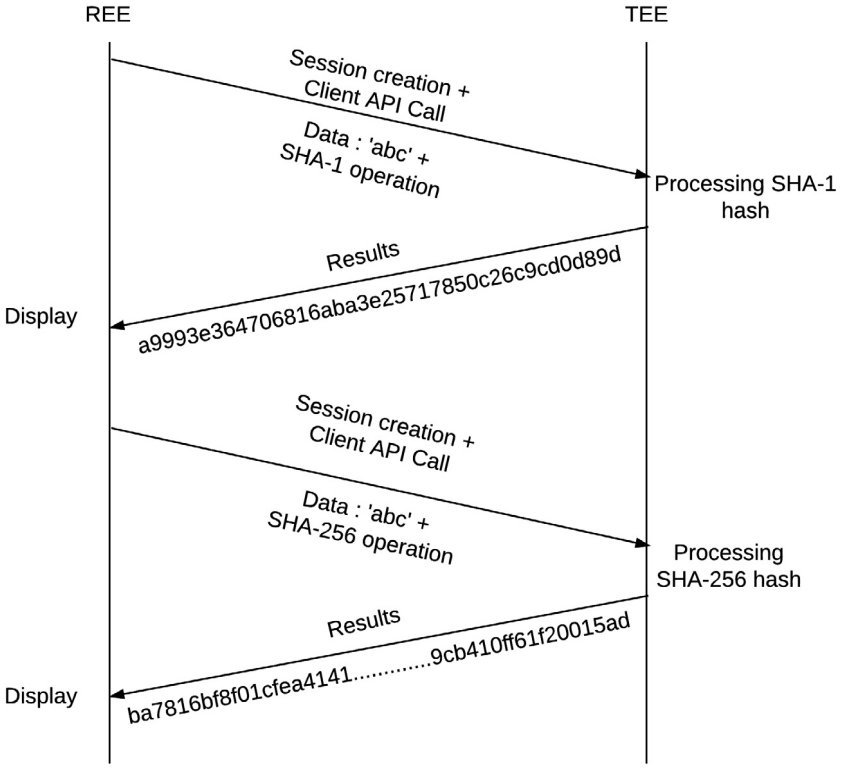

Fig. 11. Trusted application example execution.

Table 2
Resistance of the solutions against the attacker models.

Technologies/attacker model	Malicious app attacker	Root attacker	Intercepting root attacker
Secure Elements (SE)	✓	✓	
Trusted Platform Module (TPM)	✓	✓	
Trusted Environment Execution (TEE)	✓	✓	✓
Virtualization	✓		

Table 3
Comparison of existing secure storage and execution environments.

Solution	Type	Resistance to P. attacks	Resistance to L. attacks	Provisioning model	Overhead	Computing cap.	Storage cap.
Secure elements	Physical chip	No	No	ICOM	0	Limited	Limited
TEE	Both	Yes	Yes	ICOM	0	High	High
TPM	Physical chip	No	No	–	0	Low	Low
Virtualization	Purely software	Yes	Yes	ICOM/UCOM	Minimal	High	High

by the manufacturers. This restriction is harmful for the spread of these technologies and do not consider the user implication in the security process. Therefore, a UCOM approach offering a compromise between the security level and the needs of the users is highly recommended. The virtualization is a very interesting example of a UCOM approach for secure mobile computing with a high level of security and a good user consideration. In addition, the capabilities of embedded multi-core processors will help to ensure a satisfactory level of virtualization both in terms of security and processing speed.

9. Discussion and conclusion

The mobile security and trust paradigms have become very important since the massive use of smartphones for personal and professional activities. In this context, solutions for a secure storage and processing based on above mentioned technologies have emerged but none is fully suitable for user-centric approach. TEE seems to be the most promising in a UCOM perspective as several attempts exist to bypass chip manufacturers' control. The ObC [16] project developed by Nokia research lab and presented in Section 7.1 is a very interesting first attempt for trusted mobile computing that tries to open TEE for application developers. KNOX [56] is a similar security solution developed by Samsung that tries to open Trustzone to ease the trusted applications provisioning through a cloud application market. But, as they are limited to specific phones, they still cannot be viewed as fully user-centric. Indeed, the user still relies on the phone manufacturer's closed ecosystem.

At the time of writing, every trusted mobile computing solution operates in a closed ecosystem. Even the cloud-based solutions presented in Section 7.5 are dependent on a specific technology (TPM or SE for instance). But, as every technology can address a part of the overall security problem on mobile devices, an interoperability framework should be suitable. This lack of interoperability standard is problematic but may be overcome in the upcoming years. For instance, GlobalPlatform [57] is working on a secure routing protocol allowing communication between TEE and SE. It represents an excellent combination to benefit from the high resistance to attacks of the SE and from the high processing capabilities of the TEE.

The communication between TEE and SE (or even TPM) is promising to improve the overall security but is still not fully satisfactory from a user standpoint. Indeed, these solutions are still dependent on devices. Credentials and data may be stored securely but in the event that the device is lost or stolen, data are no more usable. The use of a secure cloud as a backup of credentials and data is then suitable but it has then to propose the same level of security as the device. The secure cloud can also be viewed as a good support for security architectures. Indeed a full user-centric architecture that makes the assumption that the user's device has a SE, a TPM or a TEE in it may limit the overall adoption as not every device on the market disposes of this technology. A cloud based alternative of TPM, SE or TEE is therefore interesting. Moreover, with cloud-based solutions, new secure computing technologies may become available for smartphones allowing even more versatile secure computing architectures. An example of such technology is SGX [43] which is developed by Intel for its x86 processors. As OP-TEE for the TEE, OpenSGX is an open source SGX emulator implemented based on qemu virtualization allowing developers to test their SGX applications. Finally, a protocol suite similar to Transport Layer Security (TLS) [58] that enables the selection of relevant secure computing solutions for a specific trusted application should be investigated.

This survey work led us to the following conclusion; most of these trusted mobile computing solutions focus on virtualization and/or cloud-based implementations to overcome hardware limitations introduced by SE, TPM and TEE. These solutions range from new fully virtualized TEE in a smartphone to cloud-based TPM that can be accessed by smartphones over wireless communications. Nevertheless, at the time of writing, none of these solutions provides a fully satisfactory solution for secure application provisioning in ICOM model as they only focus on a piece of the security threats that can target a secure application. Therefore, they can be considered as useful parts of an overall generic solution that has still to be defined. Indeed, this generic solution will have to mix existing secure execution environments and cloud to enforce the user confidence.

References

[1] Smartphone use growth statistics, March 2016. URL http://www.idc.com/getdoc.jsp?containerId=prUS25641615.
[2] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, M. Smith, Why eve and mallory love android: An analysis of android SSL (in) security, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, ACM, 2012, pp. 50–61.
[3] GlobalPlatform, TEE System Architecture, Technical Report, GlobalPlatform (2011). URL http://www.globalplatform.org/specificationsdevice.asp.
[4] N. Asokan, J.-E. Ekberg, K. Kostiainen, A. Rajan, C.V. Rozas, A.-R. Sadeghi, S. Schulz, C. Wachsmann, Mobile trusted computing, Proc. IEEE 102 (8) (2014) 1189–1206.
[5] T.L. Vinh, S. Bouzefrane, Trusted platforms to secure mobile cloud computing, in: The 16th IEEE International Conference on High Performance Computing and Communications, 2014, pp. 1096–1103.
[6] A. Mitrokotsa, M.R. Rieback, A.S. Tanenbaum, Classifying RFID attacks and defenses, Inf. Syst. Front. 12 (5) (2010) 491–505.
[7] A. Mitrokotsa, M. Beye, P. Peris-Lopez, Security primitive classification of RFID attacks, in: Unique Radio Innovation for the 21st Century, Springer, 2011, pp. 39–63. (Chapter).
[8] R.N. Akram, K. Markantonakis, K. Mayes, A paradigm shift in smart card ownership model, in: International Conference on Computational Science and Its Applications, ICCSA, IEEE, 2010, pp. 191–200.
[9] R.N. Akram, K. Markantonakis, K. Mayes, Trusted platform module for smart cards, in: 6th International Conference on New Technologies, Mobility and Security, NTMS, IEEE, 2014, pp. 1–5.
[10] R.N. Akram, K. Markantonakis, K. Mayes, User centric security model for tamper-resistant devices, in: IEEE 8th International Conference on e-Business Engineering, ICEBE, 2011, pp. 168–177.
[11] R.N. Akram, K. Markantonakis, D. Sauveron, A novel consumer-centric card management architecture and potential security issues, Inform. Sci. 321 (2015) 150–161.
[12] ISO/IEC, NFC Technology Full specification, March 2015. URL https://www.iso.org/obp/ui/#iso:std:53424:en.
[13] EMV, EMV Transactions Full specification, March 2015. URL http://www.emvco.com/specifications.aspx?id=223.
[14] J.-E. Ekberg, K. Kostiainen, N. Asokan, Trusted execution environments on mobile devices, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, 2013, pp. 1497–1498.
[15] S. Tamrakar, J.-E. Ekberg, N. Asokan, Identity verification schemes for public transport ticketing with NFC phones, in: Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing, ACM, 2011, pp. 37–48.
[16] J.E. Ekberg, K. Kostiainen, N. Asokan, The untapped potential of trusted execution environments on mobile devices, IEEE Secur. Privacy 12 (4) (2014) 29–37. http://dx.doi.org/10.1109/MSP.2014.38.
[17] T. Cooijmans, J. de Ruiter, E. Poll, Analysis of secure key storage solutions on Android, in: Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, ACM, 2014, pp. 11–20.
[18] Trusted Labs, EAL7 Certification for a Gemalto smartcard embedded embedded software, March 2015. URL http://trusted-labs.com/trusted-labs-achieves-the-first-security-eal7-certificate-for-gemalto-smart-card-embedded-software/.
[19] Mastercard: Secure Elements, kinds and certification. URL https://mobile.mastercard.com/Partner/MobilePayPass/SefcureElements.
[20] Secure Element Evaluation Kit for the Android platform. URL http://seek-for-android.github.io/.
[21] Trusted Computing Group official website. URL http://www.trustedcomputinggroup.org/.
[22] A.-D. Vu, J.-I. Han, H.-A. Nguyen, Y.-M. Kim, E.-J. Im, A homogeneous parallel brute force cracking algorithm on the GPU, in: 2011 International Conference on ICT Convergence, ICTC, IEEE, 2011, pp. 561–564.
[23] A.K.J. Salil Prabhakar, Sharath Pankanti, Biometric recognition: Security and privacy concerns, IEEE Secur. Priv. 1 (2003) 33–42. http://dx.doi.org/10.1109/MSECP.2003.1193209.
[24] RFC 3580: IEEE 802.1X Remote Authentication Dial In User Service, RADIUS. URL https://tools.ietf.org/html/rfc3580.
[25] Microsoft, How to protect your network from pass the hash attack. URL https://www.microsoft.com/security/sir/strategy/default.aspx#!password_hashes.
[26] T. Nyman, J.-E. Ekberg, N. Asokan, Citizen electronic identities using TPM 2.0, in: Proceedings of the 4th International Workshop on Trustworthy Embedded Devices, ACM, 2014, pp. 37–48.
[27] S. Bajikar, Trusted Platform Module (TPM) Based Security on Notebook pcs-White Paper, Tech. Rep., 2002, URL http://www.ogobin.org/TCPA/Trusted_Platform_Module_White_Paper.pdf.
[28] ARM, ARM Security Technology. Building a Secure System Using TrustZone Technology, Tech. Rep. 2009. URL http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.
[29] M-Shield Mobile Security Technology, Tech. Rep. 2008. URL http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf.
[30] E. Owusu, J. Han, S. Das, A. Perrig, J. Zhang, Accessory: password inference using accelerometers on smartphones, in: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, ACM, 2012, p. 9.
[31] L. Gomez, I. Neamtiu, T. Azim, T. Millstein, Reran: Timing-and touch-sensitive record and replay for Android, in: 2013 35th International Conference on Software Engineering (ICSE), IEEE, 2013, pp. 72–81.

[32] Sierra TEE virtualization system for TrustZone. URL http://www.sierraware.com/open-source-ARM-TrustZone.html.
[33] Genode operating system framework. URL http://genode.org/documentation/articles/trustzone.
[34] N. Santos, H. Raj, S. Saroiu, A. Wolman, Trusted language runtime (TLR): enabling trusted applications on smartphones, in: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, ACM, 2011, pp. 21–26.
[35] OP-TEE official wiki page. URL https://wiki.linaro.org/WorkingGroups/Security/OP-TEE.
[36] LINARO security working group official website. URL https://wiki.linaro.org/WorkingGroups/Security.
[37] T6: The TrustedKernel secure OS for TrustZone processors. URL http://trustkernel.org/wp-content/uploads/2015/03/T6_TEE_datasheet.pdf.
[38] K. Barr, P. Bungale, S. Deasy, V. Gyuris, P. Hung, C. Newell, H. Tuch, B. Zoppis, The VMware mobile virtualization platform: is that a hypervisor in your pocket? ACM SIGOPS Oper. Syst. Rev. 44 (4) (2010) 124–135.
[39] ARM Virtulization Extensions. URL http://www.arm.com/products/processors/technologies/virtualization-extensions.php.
[40] C. Dall, J. Nieh, KVM/ARM: The design and implementation of the Linux ARM Hypervisor, in: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2014, pp. 333–348.
[41] Sierra Hypervisor for ARM processors. URL http://www.sierraware.com/arm_hypervisor.html.
[42] Xen ARM Hypervisor Project. URL http://www.xenproject.org/developers/teams/arm-hypervisor.html.
[43] F. McKeen, I. Alexandrovich, A. Berenzon, C.V. Rozas, H. Shafi, V. Shanbhogue, U.R. Savagaonkar, Innovative instructions and software model for isolated execution, in: Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP, HASP'13, ACM, New York, NY, USA, 2013, pp. 10:1–10:1. http://dx.doi.org/10.1145/2487726.2488368, URL http://doi.acm.org/10.1145/2487726.2488368.
[44] V. Costan, S. Devadas, Intel SGX explained, Vol. 2016, 2016, p. 86. URL http://eprint.iacr.org/2016/086.
[45] I. Anati, S. Gueron, S. Johnson, V. Scarlata, Innovative Technology for CPU Based Attestation and Sealing, in: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP, ACM, New York, NY, USA, 2013.
[46] P. Jain, S. Desai, S. Kim, M.-W. Shih, J. Lee, C. Choi, Y. Shin, T. Kim, B.B. Kang, D. Han, OpenSGX: An Open Platform for SGX Research, in: Proceedings of the Network and Distributed System Security Symposium, 2016.
[47] J. Andrus, C. Dall, A.V. Hof, O. Laadan, J. Nieh, Cells: a virtual mobile smartphone architecture, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, ACM, 2011, pp. 173–187.
[48] D. Liu, J. Lee, J. Jang, S. Nepal, J. Zic, A cloud architecture of virtual trusted platform modules, in: 2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing, EUC, IEEE, 2010, pp. 804–811.
[49] P. Urien, Cloud of Secure Elements(CoSE), Internet Draft, 2014. URL http://tools.ietf.org/html/draft-urien-cfrg-cose-00.html.
[50] B. McGillion, T. Dettenborn, T. Nyman, N. Asokan, Open-tee—an open virtual trusted execution environment, in: Trustcom/BigDataSE/ISPA, 2015 IEEE, Vol. 1, 2015, pp. 400–407. http://dx.doi.org/10.1109/Trustcom.2015.400.
[51] OpenTEE official github page. URL https://github.com/Open-TEE.
[52] A. Vahidi, P. Ekdahl, VETE: Virtualizing the Trusted Execution Environment, Tech. Rep., 2013, URL http://soda.swedish-ict.se/5456/2/20130305b_VETE_final_report.pdf.
[53] U8500 NovaThor specifications and usages. URL http://system-on-a-chip.specout.com/l/352/ST-Ericsson-NovaThor-U8500.
[54] Qemu, Qemu, a generic and open source machine emulator and virtualizer. URL http://wiki.qemu.org/Main_Page.
[55] ARM, Fixed Virtual Platforms, ARM. URL http://www.arm.com/products/tools/models/fast-models/foundation-model.php.
[56] SAMSUNG, Samsung KNOX Security Solution. URL https://www.samsungknox.com/en/system/files/whitepaper/files/Samsung_KNOX_Security_Solution_V1_10_0.pdf.
[57] GlobalPlatform, TEE Secure Element API v1.0 Specification, Tech. Rep., July 2013. URL https://www.globalplatform.org/specificationsdevice.asp.
[58] T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, Proposed Standard, August 2008. URL http://www.ietf.org/rfc/rfc5246.txt.

Mohamed Amine Bouazzouni received the M.S. degree in Information security and cryptology from Limoges university in France. In October 2014, he joined the INP-IRIT laboratory for a 3 years Ph.D. on security and privacy in the mobile world. His research interests include security solutions for dematerialized smartcards on smartphones and secure execution environments on mobile platforms.

Emmanuel Conchon received the M.S. and Ph.D. degrees in Wireless Communication from the "Institut National Polytechnique de Toulouse" (INPT), Toulouse, France in 2002 and 2006, respectively. In September 2008, he joined the Champollion University as an Associate Professor and IRIT (Institut de Recherche en Informatique de Toulouse) as a researcher. Since September 2015, he is an Associate Professor in the University of Limoges. His research interests include security solutions for wireless networks, context-aware systems and middleware solutions for health applications.

Fabrice Peyrard received the Ph.D. degree in Computer Science in 1998 and an accreditation to supervise research in 2008. Currently, he is Associate Professor of computer science and network communications at the University of Toulouse (France). His main research topics of interest are security, privacy and quality of services for ubiquitous computing.