Ecosystems of Trusted Execution Environment on Smartphones - A Potentially Bumpy Road

Assad Umar, Raja Naeem Akram, Keith Mayes, and Konstantinos Markantonakis
Information Security Group, Smart Card Centre, Royal Holloway, University of London,
Egham Hill, Egham, Surrey. United Kingdom
{Assad.Umar.2011, R.N.Akram, Keith.Mayes, K.Markantonakis}@rhul.ac.uk

***Abstract*—The advent of smartphones and the flexibility to have multiple applications serving the user's needs, has started a convergence of different services into a single device. Traditional services provided by mobile phones like voice and text communication became secondary to other domains like Online Social Network (OSN) and entertainment applications on smartphones. A similar trend is also happening for smart card services, in which traditional smart card services like banking, transport-ticketing and access control, are moving to smartphones. This transition from smart cards to smartphone is to a large extent, facilitated by Near Field Communication (NFC) technology which enables a smartphone to emulate a smart card. As the smart card services require a comparatively higher level of security than other applications on the smartphone. Initial proposals for this convergences were focused on secure elements. However, the ownership issues reminiscent of traditional smart card domain became the Achilles' heel. A potential way forward has been proposed by the Google Andriod in the shape of Host Card Emulation (HCE) to allow mobile phone applications to communicate via NFC. However to provide higher-level of security as required by smart card applications, a number of proposals have been put forward including the Trusted Execution Environment (TEE). In this paper, we will look into how the TEE fits into the overall picture of smart card services on a smartphone – provisioned via the HCE. We also analyse the current state of the art of TEE proposal and what potential ecosystem hurdles it might face due to the nature of current trends. Finally, we provide a potential pathway to overcome the ecosystem issues to achieve wide scale deployment, enabling secure services to individual users.**

I. Introduction

Modern mobile phones are more than just a conduit for voice and text communication. They have taken the shape of small mobile computers, where a large number of applications serve their users. These applications on smartphones include traditional communications (voice and text), Online Social Network (OSN), gaming and entertainment. While these applications also require a degree of security and privacy, applications such as healthcare, banking, transport ticketing and access control require higher levels of security and privacy protection. The smartphone on its own cannot provide strong security and privacy protection required by certain applications. To that effect, solutions such as the Secure Element (SE) [1] and Trusted Execution Environment (TEE) [2] were proposed. The SE was proposed as a secure tamper-resistant environment to store and execute traditional smart card services such as contactless banking, transport ticketing and access control.

The Near Field Communication (NFC) [3] and SE seemed to have a natural alliance. While NFC and SE-based smart card services have been trialled in more than 70 countries [4], It has not seen wide scale deployment due to the ecosystem issues related to the control and management of the SEs [5, 6].

The Host Card Emulation (HCE) [7] on the other hand brings in a unique perspective. It enables any application on a smartphone to emulate a smart card and communicate with an external reader using the NFC channel, without the need for an SE. HCE's reliance on software protection provided by the OS means it provides potentially less security than the hardware backed SE, which might be an issue for sensitive applications. For this reason, some security mechanisms have been proposed to counter the risks introduced by HCE. We shall discuss more on these mechanisms later[1].

A TEE provides a secure process, memory and storage protection to an application (or part of an application). During the execution of the application, the TEE ensures isolated execution and integrity protection for the application from any applications running in non-TEE environments on the same device. However, at the moment not many applications are utilising the TEE services and mostly it is due to accessibility issues [8, 9], where an application developer is required to get additional permissions from the Original Equipment Manufacturer (OEM) before its application can execute in TEE environment.

In this paper, we will evaluate how NFC, HCE and TEE are shaping the opportunities that might enable secure services on a smartphone (Section II). We discuss potential options for HCE-NFC-based services and how they can be secured, along with why TEE fares better in comparison (Section III). We also discuss the rationale behind why TEE is considered a potential avenue for HCE-based smart card services. Finally, we evaluate different ecosystems for TEE deployments that might affect the overall acceptability and usability of HCE applications based on TEE. (Section V).

II. Smart Cards Services and Smartphones

In this section, we will discuss how the transition of traditional smart card services to smartphone has invigorated both the smart card and smartphone domains.

[1]These mechanisms include TEE, tokenisation, White-box cryptography and cloud-based HCE.

A. Near Field Communication - An Opportunity

Near Field Communication (NFC) is a low frequency radio wave that operates at 13.56-MHz spectrum and has an operational distance of less than 10 cm. There are 2 types of NFC devices: one that generates the low frequency radio-wave field, and for this they need an internal power supply so are known as an active NFC device. The second type is the device that when it comes close to the NFC radio field, it generates power from the magnetic inductive coupling, and such devices are termed as passive NFC devices. As these devices are dependent on the active devices for the power supply, they do not have any internal power source of their own. An NFC device can act as both an active and passive device as required and this ability gives the NFC-based devices a unique capability as compared to other contactless communication technologies. In addition, NFC devices could operate in three different modes that include reader/writer mode, peer-to-peer and card emulation mode [10]. As this paper is focused on the card emulation mode, in subsequent section we discuss what it entails.

1) Card Emulation: In this mode, NFC devices act as a contactless smart card or card reader in compliance with the ISO 14443 [11]. In fact the external reader is unable to distinguish between a smart card and an NFC device in card emulation mode. This mode is useful, as NFC devices could be utilised in any service sector where traditionally smart cards are already being deployed (i.e., banking, transport and building access, etc.). In this scenario, a mobile phone has a built-in NFC antenna that an SE in the mobile phone can use to emulate the contactless smart card interface. However, with the advent of HCE, any application running on the Android platform can now also access the NFC's features including the card emulation mode. The applications that interact with the external readers either execute in an SE or on general execution platform of the device in the case of HCE. Therefore, protecting the HCE application is paramount and TEE is one of such initiatives to achieve this.

2) Why Not SE?: To break up with the experience of the multi-application smart card initiative in which a centralised card issuer-centric ecosystem was deployed, a different approach was considered for the NFC-based multi-application SE initiative. The architecture proposed by the GSMA was based on an entity termed as Trusted Service Manager (TSM) [12]. The role of the TSM was to provide a link between a Mobile Network Operator (MNO) and other application providers such as Card Issuing Bank (CIB) and transport-ticketing companies. A CIB, for example, can have a relationship with multiple TSMs and so can an MNO, thus providing an inherent scalability in the architecture. However, even after redefining the ecosystem for SEs (i.e., smart cards), they have unfortunately not seen a wide scale deployment.

B. Host-based Card Emulation - An Alternative

Host-based Card Emulation (HCE) offers a radical alternative to NFC card emulation using an SE. HCE was first introduced by Cyanogenmod [13] and more prominently when Google introduced it from Android 4.4 (KitKat) [7] onwards. HCE allows an application running on the OS to emulate a smart card. As the name implies, the 'host' OS has full control of NFC transactions and therefore bypasses the need for an SE. The NFC controller routes messages directly to the application. Therefore, HCE offers more flexibility as compared to the card emulation with an SE, at the expense of security [14]. In HCE, an application emulating a smartcard acts like a typical android application. This means the application relies on the default security features provided by the OS to protect against unauthorised access or modification of data. The default security provided by the OS are typically not enough for sensitive applications such as payment applications.To that effect, a number of mechanisms have been proposed to counter the security threats introduced by HCE, and these will be discussed in III

III. Security Sensitive Services in NFC-based Transactions

In this section, we outline a set of requirements that should be considered when deploying security-sensitive NFC applications such as payment and transit applications. We also provide an overview of mechanisms proposed to counter the security threats introduced by HCE, and in conclusion, we provide a comparison of the proposed mechanisms against the set of requirements outlined.

A. NFC Applications Requirements

1) Security: Although specific requirements may vary from one case to the other, it is important to provide strong security assurances. The security of NFC applications is no different from the fundamentals of information security: confidentiality, integrity, and availability (CIA). Using an NFC transit application, as an example, the transit, payment, and cryptographic data must be safeguarded against unauthorised disclosure and modification both at rest and at run-time, and the transit system should be available for use at all times.
2) Performance: The performance of NFC-based transaction must be significantly fast in order to maintain high throughput and maintain overall user satisfaction; both the payment and the transit industries require that a transaction is completed within the range of 300-500 milliseconds [15]–[20].
3) Flexibility: The use of NFC applications must not introduce rigidity to the ecosystem. For example, the case of card emulation using the SE is rigid because provisioning an application into the SE requires permissions from the 'owner' of the SE, which in most cases is not easy. This means developers and researchers find it very difficult to make use of these services without signing contracts with various stakeholders. Therefore, an effective NFC solution must be flexible and work out-of-the-box.
4) Complexity: The complexity of the ecosystem should be as simple as possible. A complex ecosystem leads to

increased implementation times and added cost. For example, the Trusted Service Manager (TSM) model used to provision applications to the SE increases complexity tremendously. A TSM is a trusted third party whose responsibility it is to facilitate provisioning, and to manage the entire life-cycle of the service. The inclusion of these extra parties makes it more complex and also more expensive because these services are not free.

5) Low Power Mode: In low power mode, the device's OS is shut down due to 'low' battery and therefore appears odd to the user; however, the Power Management Integrated Circuit (PMIC) is still on and can facilitate NFC transactions with help of power generated by the reader in the field.[2]
6) Connectivity: Some NFC applications may require connectivity for every transaction, while some may only require connectivity from time to time, to update credentials. For example, some applications will require access to storage in the cloud for every transaction, while others may use a notion of tokenisation [21] and only require periodic connectivity to request new tokens.
7) Tamper-Resistance: NFC applications require protection against modifications and reverse-engineering by malicious entities. Software tamper-resistant techniques such as obfuscation and other techniques are used by developers to achieve this, although these techniques are usually costly in terms of performance and code size [22]. Other ways involve installing the NFC application and related data into a tamper-resistant physical silicon such as the SE.
8) Interoperability: In the context of this paper, it represents the ability to host NFC applications on different devices from different vendors. This ensures the NFC ecosystem operates in a cohesive and efficient manner.
9) Standardised APIs and Ease of Development: The availability of standardised APIs goes a long way to easing the development process of NFC applications, thereby shortening the overall deployment time. Standardised APIs also ensure that developers adhere to software engineering global best practices, which is beneficial to the overall security fabric.

In subsequent sections, we outline various mechanisms used to manage the security in HCE-based NFC applications to an acceptable level.

B. Cloud-based HCE

One of the ways to manage security in terms of risk of exposure is to store all security sensitive data in the cloud, as well as performing security related executions in the cloud, and necessary data is pushed to the phone just in time for, or during, a transaction. From a feasibility point of view, cloud-based HCE offers a good alternative. However, it poses its own challenges: Network connectivity is needed to perform each transaction, and this may prove difficult to guarantee in some cases such as in transit, where the vehicle is moving, or the train stations are underground. Also, connecting to the cloud for every transaction will introduce latency to the overall transaction, which may be unacceptable for certain cases.

C. White-box Cryptography

Similar to code obfuscation [23], White-Box Cryptography (WBC) is the art of making cryptographic algorithm implementations and related keys and data unintelligible to an attacker, even at run-time. This aims to provide defences against white-box attack scenarios. In white-box attack scenarios, the attacker is assumed to have full access to the implementation of the crypto-algorithm, as well as full control over the device (i.e. system calls, memory locations and registers, etc.). However, WBC has its own weaknesses: for example, previous work in [24, 25] have shown the cryptanalysis of both generic and specific implementations of white-box implementations. In addition, WBC typically require more computing resources in terms of processing and are slower in orders of magnitude [26]. This may call into question its applicability in use-cases such as payments and ticketing where performance is vital.

D. Tokenisation

Payment tokenisation has been adopted by the payment industry as the preferred way of managing the risk of exposure in HCE-based applications, and to that effect, the EMVco has published a specification on the use of tokenisation in the payment ecosystem [21]. This process replaces the payment credential with a short-lived credential referred to as a 'token'. In a situation where the payment device is compromised, the attacker can only access tokens with a limited validity, as opposed to having the real credential. Although tokenisation mitigates the risk to a certain level, it is still leaves gaps with respect to the security of tokens and related data. This is because the tokens themselves still require some level of security, which largely depends on the nature of the tokens (single or multiple use tokens for example). Tokenisation also requires network connectivity to the Token Service Provider (TSP) to receive new tokens from time-to-time.

E. Trusted Execution Environment (TEE)

A TEE is an area of a processor (could also be a dedicated processor) that is 'trusted' to provide secure services such as isolated storage and isolated execution platforms for applications running on the host operating system platform (OS). There are different implementations of TEEs, but the underlying concept is the notion of segregation between the main OS and the trusted side. This effectively removes the host OS of the device from the Trusted Computing Base (TCB)[3] This segregation usually extends to other peripherals of the device. The main focus of this paper is on TEEs in the context of HCE-NFC based mobile applications; therefore, a more detailed discussion on the services a TEE should provide, as well as example implementations, shall be discussed in IV.

[2]Low power mode should not be confused with "battery off mode" where even the PMIC has no power

[3]The TCB is a collection of all the software, firmware, hardware components that are absolutely trusted to enforce security on a device.

F. Secure-HCE Hybrid Element

HCE does not dictate where the application and its data is stored. To that effect, there have been proposals to use an SE to provide security services for HCE applications such as key generation and random number generation This model is referred to as the hybrid model [27]. The tight control on the SE (which gave birth to HCE) means success of this model will depend on exactly what services the 'owners' of the SE are willing to expose to developers and other third-party entities.

TABLE I
COMPARISON BETWEEN TEE AND COMPETING TECHNOLOGIES

Criteria	Cloud-based HCE	White-box Cryptography	Tokenisation	TEE	Secure Element
1) Security	●	●	●	●	●
2) Performance	◑	○	●	●	●
3) Flexibility	●	●	◑	◑	○
4) Complexity	◑	◑	○	◑	○
5) Low Power Mode	○	○	◑	○	●
6) Connectivity	○	●	◑	●	●
7) Tamper-Resistance	○	●	◑	●	●
8) Interoperability	○	○	◑	○	◑
9) Standard APIs & Easy of Deployment	○	○	●	◑	◑

Note: ● if criteria is fully satisfied, ◑ if criteria is partially met and ○ if criteria is not satisfied..

G. Comparison between TEE and Competing Technologies

In terms of security, all technologies provide it to a certain extent, as that is their core functionality. The level of security they provide is beyond the scope of this paper. Their performance is usually acceptable with exception of white-box cryptography, which is slower than textbook implementations of the algorithm used. Cloud-based HCE may also be exposed to performance challenges because it largely depends on the latency introduced from connecting to the cloud.

Only the SE fully satisfies the low power mode criteria; in the case of tokenisation, it will depend on whether the tokens are stored on the SE or on the host OS. In terms of connectivity, all technologies meet the criteria except cloud-based HCE, where connectivity is required for every transaction. Tokenisation only partially meets the connectivity criteria, because even though transactions can be carried out offline, the application will still need connectivity at certain intervals to receive new tokens.

All the technologies provide a notion of tamper-resistance, with the exception of cloud-based HCE. Tokenisation will also depend on implementation decisions: whether tokens are stored in a tamper-resistant storage area.

IV. TRUSTED EXECUTION ENVIRONMENT

In this section, we give a generic set of security services a TEE should provide for mobile applications (including NFC). We also provide an overview of two different implementations of a TEE.

A. TEE Security Services

1) Isolated Execution: A TEE should have the ability to execute a piece of code in complete isolation from the other applications running on the device (including the main OS. This offers *confidentiality* and *integrity* of the code and related data at "run time". For example, a payment application on a mobile device should be able to generate digital signatures, without other applications observing the process or having access to the signing keys.
2) Secure Storage: A TEE should also provide assurances with regards to the integrity, confidentiality and in most cases the freshness of data at rest for the applications. For example, applications such as transit and payment make use of sensitive data such as cryptographic keys, passwords and primary account numbers (PANs) that must be protected against unauthorised access at all times.
3) Remote Attestation: Attestation, means "to vouch for something". *Remote attestation* is the ability to vouch for an entity or its characteristics to a third party. In the context of TEEs, it is used to give assurances to a service that a piece of software or OS is trustworthy. For example, in mobile payments, the issuer can get assurances through remote attestation on the integrity of the banking application running on the device. A remote attestation protocol normally begins with an integrity measurement of the TCB as a whole. This is done by taking a digest of the code, for example by using a hash function, and signing it with a key, stored preferably in a hardware RTS. A third party can verify the signature and compare it against a list of trusted OS hashes.
4) Secure Provisioning: Secure provisioning is the ability to send data to a device or a particular software component running on the device in a secure fashion. This gives service providers (SPs) a way to provision applications and make relevant updates. For example, in payment applications, banks require a secure mechanism to deliver the application to the device of its user. There must be assurances on the integrity of the application as well as the confidentiality of other cryptographic data.
5) Trusted Path: A TEE should also provide a secure input/output mechanism with which users can interact with applications on the device. For example, payment applications often require user verification through PINs or biometrics. The TEE should provide a trusted path between the application and the keypad. This provides protection against malware on the device, which normally might be able to access sensitive data entered by user by logging keystrokes, or by more sophisticated methods such as sniffing data on the physical communication layer.

B. ARM's TrustZone

TrustZone is ARM's security technology solution provided by the more recent ARM processors. TrustZone achieves security by logically separating hardware and software resources into two distinct modes, referred to by ARM as the "secure-mode" and "normal/nonsecure mode". The resources in the secure-mode cannot be accessed by the resources in the normal mode.

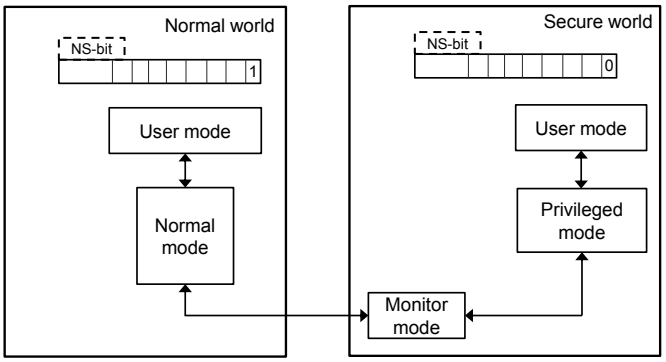

Fig. 1. System Architecture of Trustzone: Showing the Two Worlds [28]

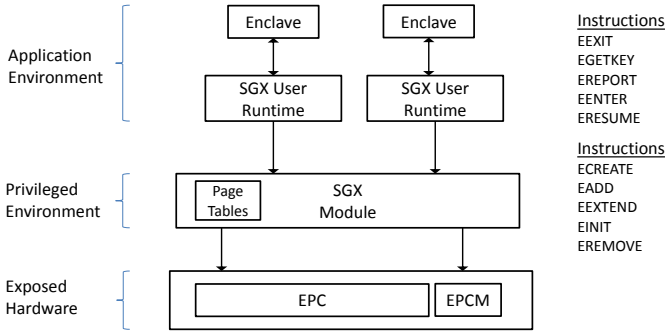

Fig. 2. Hardware and Software Architecture of Intel SGX based on [29]

TrustZone Hardware Architecture

The secure/nonsecure separation is extended to the hardware components, and this is enforced by a hardware logic in the TrustZone-enabled AMBA3 AXI bus fabric [28]. The AMBA3 AXI bus provides an extra *control signal*, known as the *NS bit*, to read and write channels which are on the main system bus. If NS-bit = 0, then the processor is in the secure-mode; otherwise, if NS-bit = 1 it indicates the processor is operating in the nonsecure-mode. This transition between the two modes is controlled by a mechanism referred to as the *monitor mode* as shown in Fig 1.

TrustZone also secures peripherals such as interrupt controllers, timers, and I/O components through the AMBA3 APB peripheral bus. This ensures that the system is monitored by a task that cannot be interrupted by malware. The AMBA3 APB peripheral bus is a low gate-count and low-bandwidth peripheral bus that is connected to the system bus using an AXI-to-APB bridge. This bridge controls access to the peripherals, ensuring only requests with the necessary security status reach the peripherals.

The processor switches between the two modes (context switching) in a *time-sliced* fashion and is controlled by the "monitor mode". For an application running in nonsecure-mode to make use of services offered by another application in the secure-mode, for example to encrypt a document, a special instruction known as the "Secure Monitor Call (SMC)" is used. SMC immediately sets the NS-bit to '0' and then fully transfers control to the secure-mode. On the other hand, switching back from secure-mode to nonsecure-mode is less controlled, and the secure side can directly alter the "Current Processor Status Register". So, essentially, the monitor mode can be seen as providing *gate-keeping* services between the two modes.

C. Intel Software Guard Extensions (SGX)

Intel's SGX are a set of instructions that provide extensions to the Intel architecture processors. These extensions provide a TEE within the computer's untrusted environment. Intel SGX thus aims to provide confidentiality, integrity, and replay protection using *enclaves* [29].

An enclave is part of the applications memory space which is hardware protected. An enclave has a reserved area of memory from which it runs, known as the enclave page cache (EPC). Access to the EPC is protected from processes outside of the enclave. The OS can manage the enclaves, but cannot tamper with the code and data within the enclave itself.

The management of an enclave is carried out through a set of instructions. The *ECREATE* instruction creates an enclave and also sets the base linear address as well as the physical address. After the creation of an enclave, the *EADD* instruction is used to add relevant code and data to the enclave. This adds 4KB of protected data. For integrity protection, the *EEXTEND* instruction is used to measure the contents of the enclave. This measures 256 bytes at a time; therefore, to measure the whole contents of the enclave, the instruction is called 16 times [29]. The enclave is then initialised using the *EINIT* instruction. This sets the INIT attribute to true, which means the code in the enclave is ready for execution.

V. Ecosystem Models for TEE

In this section, we will discuss different ecosystem models for the TEE services. These systems describe the accessibility, business roles, liabilities, and application provisioning architecture that can potentially be categorised into six different facets, which are discussed in the following sections.

Before we go into the discussion of different ecosystems, first we define the individual stakeholders and the terms that refer to them. A device manufacturer is an organisation that develops a handset (both the hardware and software platform). This includes both Android and Apple style of mobile development. An application developer is an organisation that develops an application such as banking, transport-ticketing and access control. Finally, a consumer is the device holder that buys the smartphone and wants to use it for NFC-HCE-based services.

A. Centralised (Device-Manufacturer) Model

In this model, the device manufacturer has full control of the TEE service provisioning. For an application developer that would require access to secure services provided by the TEE, they have to get explicit approval from the device manufacturer to install their application on a smartphone with access to TEE.

This ecosystem is currently being used in limited roll out of the TEE on smartphones. At the core, this is the same ecosystem as the Issuer-Centric Ownership Model in smart cards [30].

A variant of the centralised model is when the access to the TEE is controlled by the device manufacturer, but it allows a trusted third party, referred to as trusted application provisioner (TAP), to provide application provisioning on behalf of the device manufacturer. This is similar to the Trusted Service Manager (TSM) model in NFC-SE, argued that it would enable flexibility and scalability for the application provisioning on SE did not materialised in real world deployment. The TAP facilitates the sharing and usage of the TEE services to achieve this, the TAP would have to have a business relationship with a wide range of device manufacturers and application developers.

B. Security as a Service Model

This is not a full ecosystem that deals with the overall management of the TEE services. However, this provides an alternative which enables third party applications to use secure services running in the secure world of a TEE. The actual underlying model can be either of the above discussed models. In this model, an organisation can negotiate access to the TEE services, and provide Security as a Service (SaaS) features. Third party applications that don't have access to TEE services directly, can then utilise the SaaS to provide security to their applications.

C. Consumer-Centric Model

This model gives full control of the TEE services provisioning to the user. This model is similar to the User Centric Ownership Model for smart cards and a potential consumer-centric model proposed by GlobalPlatform [31]. This model is also similar to the current application provisioning mechanism deployed in the smartphone industry, where a user can install or delete any application as they desire.

The user in this model takes on the role of the TAP and allows the access to the TEE services. They also have the privilege to enrol, evoke and block access of any application that requests the TEE services. From the liability point of view, the applications should manage their own risk mitigation processes and users (or the device manufacturer) might not be liable.

VI. Comparison Between Models

In this section, we set comparison criteria on which we will later compare and contrast different ecosystem proposals for TEE.

A. Comparison Criteria

The rationale for constructing these criteria is to illustrate the positives as well as the shortcomings of individual ecosystem models discussed in previous sections. Economic considerations may decide the choice of model among the competing models, however, this does not mean that it is the best possible model. Nevertheless, the following criteria are based on the potential elements that might play a critical role in successfully bridging the transition of smart card services from specialised hardware (smart card) to smartphones.

1) Market Segmentation: Is the proposed model restrictive to a level that it might create pockets of market access? In market segmentation, certain applications might be only available on particular devices due to the business relationship between application developer and device manufacturer.
2) Scalability: The model enables a wide scale deployment of a number of applications, from heterogeneous application developers, therefore serving applications with varied and potentially changing requirements.
3) Flexibility: The model is flexible in a sense that a small application developer can also gain access to TEE services like a big corporation can.
4) Impartiality: The model does not discriminate any particular or set of application developer(s). Any application developer that abides by the security and privacy policies set by the model is allowed to access TEE services.
5) Consumer-Involvement: Does the consumer (users) have any involvement in either provisioning or evoking the TEE services to individual application developers?
6) Open Provisioning: Any application developer can create an application requiring access to the TEE services and potentially use these services without requiring expansive and potentially long approval processes set by device manufacturers or TAP.
7) Closed Provisioning: For application provisioning of access to the TEE services, a centralised authority such as TAP and/or device manufacturer has to approve it.
8) User Privacy: A set of applications on a smartphone might also signify potential privacy information about a user. Therefore, does the model reveal the set of applications using the TEE services to an external entity including device manufacturers, TAP and/or other application developers?
9) Application Intellectual Property (IP) Protection: The model does not require the application developer to reveal the source code of their application to the entity that provisions the access to the TEE services.

B. Comparison and Discussion

Based on the defined criteria in the previous section, the comparison is shown in Table II.

For the SaaS model, most of the partially met criteria is due to the nature of the model. It does provide flexibility to small application developers to gain access to secure services from TEE, without going through the device manufacturers and/or TAP approval model. For the criteria impartiality we have marked all models except the consumer-centric one as partially meeting the criteria, for the reason that there is a centralised entity and guaranteeing its impartiality would be difficult. However, for the consumer-centric model, we marked it meeting the criteria as full because if a user downloads the

TABLE II
COMPARISON OF DIFFERENT ECOSYSTEMS FOR TEE DEPLOYMENT

Criteria	Centralised Model	Security as Service	Consumer Centric
1) Market Segmentation	◑	◑	●
2) Scalability	◑	◑	●
3) Flexibility	○	●	●
4) Impartiality	◑	◑	●
5) Consumer-Involvement	○	◑	●
6) Open Provisioning	○	◑	●
7) Closed Provisioning	●	◑	○
8) User Privacy	○	○	●
9) Application IP Protection	○	◑	●

Note: ● if criteria is fully satisfied, ◑ if criteria is partially met and ○ if criteria is not satisfied..

application, there is a high probability that he or she would give the permission to use TEE services.

Centralised models do not fare well on the criteria including open provisioning, consumer-involvement, user privacy and applications IP protection. The reason behind this is the process which an application has to go through before getting access to the TEE. It can be argued that App Stores managed by Apple and Google already do so. However, most of these applications might not have proprietary code like banking, transport-ticketing and mobile network operators (soft-SIMs) might have. Such application, along with other high security sensitive applications, might not accept it.

The consumer-centric model meets the highest number of criterion in comparison to all other models discussed in this paper. It might be argued that such a model might introduce security issues, like a malicious application can also run in the TEE. This is possible, but TEE by no means provide an assurance that only nonmalicious application code will execute in it. Such assumptions are based on prior vetting of an application by device manufacturers and/or TAP, by analysing the application source code. This, as discussed before, might not be preferable to a large set of application providers.

C. Future Research Directions

The proposal of the TEE is still in its infancy and there are plenty of open issues that need to be resolved. Some of these are listed below:

- Open Deployment: To build an ecosystem that allows open deployment of the TEE provisions is a challenging task, especially ensuring that access provisioning, revocation and blocking of applications can be achieved in a secure and reliable manner, and, furthermore, to transfer the application binaries that would run in the secure world and proof of hardware (to avoid simulator attack) are an open issue.
- Effect of Faults: High security-embedded devices along with smart cards are targeted using radiations to change the stored values of an applications data during its execution. If TEE was to run secure smart card applications, then similar attacks might target the TEE. Therefore, to ensure that TEE is protected against the effect of faults is important.

VII. CONCLUSION

Applications that were traditionally associated with the smart card devices are in transition from a secure and tamper-resistant device to a feature rich environment like smartphones. This transition was due to the emergence of the NFC technology. To provide high levels of security assurance, similar to the ones required by the smart card applications, initially SE was put forward as the candidate platform. Unfortunately, this is not gaining wide-scale acceptability. In addition to this, HCE enabled any application running on an Android platform the NFC channel, opening the field to any application to emulate a smart card. To provide security to such applications the preferable candidate is TEE. In this paper we have analysed the TEE proposal and its potential deployment models. We listed a set of criteria to show why the TEE is the most preferable candidate to secure the NFC-HCE applications. Furthermore, we have described potential deployment models and comparison criteria, based on the potential of acceptability and wide-scale deployment. This paper showed that there is still a long way to go before TEE might enable smart card applications to run in an NFC-HCE combination. However, the deployment model is most crucial, as similar to the SE, this could either make or break the TEE proposal for NFC-HCE.

REFERENCES

[1] K. Mayes and K. Markantonakis, *Smart Cards, Tokens, Security and Applications*, 1st ed.

[2] GlobalPlatform, "GlobalPlatform Device Technology, TEE System Architecture v1.0," December 2011.

[3] *ISO/IEC 28361: Near Field Communication Wired Interface (NFC-WI)*, International Organization for Standardization (ISO) Std., October 2007.

[4] NFC Trials, Pilots, Tests and Live Services around the World. Online. NFC World.

[5] R. N. Akram, K. Markantonakis, and D. Sauveron, "Collaborative and Ubiquitous Consumer Oriented Trusted Service Manager," in *2014 IEEE 13th Inter. Conf. on Trust, Security and Privacy in Computing and Communications*. IEEE, 2014, pp. 448–456.

[6] ——, "A Novel Consumer-Centric Card Management Architecture and Potential Security Issues, volume = 321, year = 2015," *Information Sciences*, pp. 150 – 161.

[7] A. D. Guide, "Host-based Card Emulation," https://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[8] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The Untapped Potential of Trusted Execution Environments on Mobile Devices, journal =IEEE Security & Privacy, volume = 12, number = 4, issn = 1540-7993, year = 2014, pages = 29-37, publisher = IEEE CS."

[9] T. Nyman, B. McGillion, and N. Asokan, *On Making Emerging Trusted Execution Environments Accessible to Developers*. Cham: Springer, 2015, pp. 58–67.

[10] G. Madlmayr, O. Dillinger, J. Langer, and C. Schaffer, "The benefit of using SIM application toolkit in the context of near field communication applications," in *the Inter. Conf. on the Management of Mobile Business*. DC, USA: IEEE CS, 2007, p. 5.

[11] *ISO/IEC 14443-1: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards*, ISO Std., Rev. 2nd Edition, June 2008.

[12] "Pay-Buy-Mobile: Business Opportunity Analysis," GSM Association, White Paper 1.0, November 2007.

[13] D. Yeager, "Added NFC Reader support for two new tag types: ISO PCD type A and ISO PCD type B," 2012.

[14] A. Umar, K. Mayes, and K. Markantonakis, "Performance Variation in Host-based Card Emulation Compared to a Hardware Security Element, year=2015, pages=1-6, doi=10.1109/MOBISECSERV.2015.7072872, month=Feb,," in *Mobile and Secure Services (MOBISECSERV), 2015*.

[15] "Transit and Contactless Open Payments: An Emerging Approach for Fare Collection," Smart Card Alliance Transportation Council, White Paper, November 2011.

[16] M. Emms, B. Arief, L. Freitas, J. Hannon, and A. van Moorsel, "Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards without the PIN," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 716–726.

[17] "The Future of Ticketing: Paying for Public Transport Journeys Using Visa Cards in the 21st Century," VISA, Whitepaper, January 2013.

[18] "MasterCard Contactless Performance Requirement," MasterCard, Online, March 2014.

[19] "EMV Contactless Specifications for Payment Systems: Book D - EMV Contactless Communication Protocol Specification," EMVCo, LLC, Specification Version 2.6, March 2016.

[20] "Transactions Acceptance Device Guide (TADG)," VISA, Specification Version 3.0, May 2015.

[21] "Payment Tokenisation Specification," Online, EMV, Specification, March 2014.

[22] M. H. Jakubowski, C. W. N. Saw, and R. Venkatesan, *Tamper-Tolerant Software: Modeling and Implementation*. Springer, 2009, pp. 125–139.

[23] G. Wroblewski and G. Wroblewski, "General Method of Program Code Obfuscation," 2002.

[24] W. Michiels, P. Gorissen, and H. D. L. Hollmann, *Cryptanalysis of a Generic Class of White-Box Implementations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 414–428. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04159-4_27

[25] J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen, *Differential Computation Analysis: Hiding Your White-Box Designs is Not Enough*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 215–236. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-53140-2_11

[26] M. Joye, "On White-Box Cryptography," in *Security of Information and Networks (SIN)*. Trafford Publishing, 2008, pp. 7 – 13.

[27] S. C. Alliance, "Host card emulation (hce) 101," Smart Card Alliance, Mobile and NFC Council, Tech. Rep., August 2014.

[28] A. Limited, "ARM Security Technology Building a Secure System using TrustZone Technology," April 2009.

[29] I. Corporation, "Intel Software Guard Extensions Programming Reference," October 2014.

[30] R. N. Akram, K. Markantonakis, and K. Mayes, "A Paradigm Shift in Smart Card Ownership Model," in *ICCSA Workshops*, B. O. Apduhan, O. Gervasi, A. Iglesias, D. Taniar, and M. L. Gavrilova, Eds. IEEE CS, 2010, pp. 191–200.

[31] "A New Model: The Consumer-Centirc Model and How it Applies to the Mobile Ecosystem," GlobalPlatform, Whitepaper, March 2012.