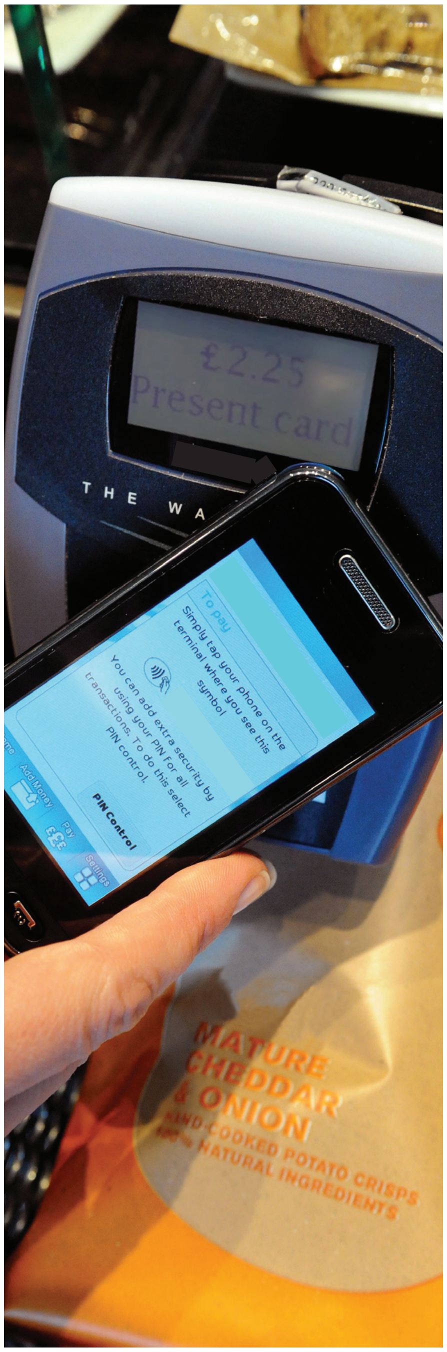
HCE security implications

Analyzing the security aspects of HCE

January 8th, 2014

HCE security implications

About the authors: Thom Janssen is Managing Consultant with UL Transaction Security (UL). Prior to joining UL Thom has worked 8 years as a Business Consultant Mobile Networks. In this role he has helped leading Mobile Network Operators in a wide variety of projects, such as technology strategy, next-generation network trials and network quality measurement and improvement. At UL he focuses on mobile payments.
Mark Zandstra is currently finalizing his master thesis at UL with a focus on the comparison between HCE and SE-based NFC solutions. After obtaining his master's degree in Computing Science, he will start as a Technical Consultant in the Mobile & Payment Practice of UL.


£2.25
Present card
To pay
Simply tap your phone on the terminal where you see this symbol
You can add extra security by using your PIN for all transactions. To do this select PIN control
PIN Control
Add Money
Pay
Settings
MATURE CHEDDAR & ONION
HAND COOKED POTATO CRISPS
100% NATURAL INGREDIENTS

On October 31st 2013, Google introduced the latest version of its mobile Operating System, Android 4.4 KitKat. KitKat included a new Near Field Communication (NFC) feature: Host-based Card Emulation (HCE). HCE has garnered quite some attention in the NFC and mobile payment industry, because it opens up the possibility to perform NFC card emulation without using the Secure Element (SE) in Mobile handsets. Industry attention is further increased by a first major implementation of HCE, by Google for an SE-less NFC payment system in its Google Wallet offering.

UL believes that HCE may accelerate the introduction of NFC services, because it provides an optional more-simple-but-less-secure way to provide an NFC card emulation service. It has great added value for Service Providers (SPs) that can accept a reduced level of security in exchange for an improvement of other factors such as time to market, development costs and the need to cooperate with other parties. These SPs must however be fully aware of the security risks caused by the lack of the hardware-based security as provided by the SE.

1. NFC for card emulation

NFC is a short range wireless technology that allows communication between two devices over short distances of up to ten centimeters. Based on this technology, devices like mobile phones are able to communicate wirelessly with each other. NFC combined with enhanced security mechanisms enables the use of "virtualized" smartcards on mobile devices such as handsets. The security of NFC card emulation is traditionally based on the use of an SE, i.e. a tamper-resistant chip inside the handset in which the card emulation solution can perform cryptography and store its sensitive data in a secure and trusted environment.

NFC is now supported on many types of handsets, and forecasts state that 500 million NFC-enabled handsets and other devices will be on the market in 2014[1].

Due to the secure capabilities of SE-based card emulation using NFC, many parties have become aware of this technology as an enabler for "mobile wallet" functionalities: payment, loyalty, couponing, ID, transit and access control, all combined inside the mobile handset of the end-user[2]. The SE can be present in three form-factors; UICC (i.e. the SIM card), embedded SE (separate hardware chip on handset) or a secure SD-card. The SD-card option has proven difficult because it is often provisioned by a single SP. A user wanting to use emulated cards from multiple SPs must switch SD-cards. The other SE types – UICC and embedded SE – are typically not controlled by the SPs. This means that these SPs must interact with the issuers of the SE, which has proven to add significant complexity to the development and provisioning of NFC card emulation services. HCE promises to change that.

2. HCE technical functionality

NFC has three operation modes: Reader/Writer Mode for reading/writing data from/to a tag, Peer-to-Peer Mode for communication between two devices and Card Emulation Mode for emulating a smartcard. To enhance security, Card emulation makes use of an SE. The NFC Controller, a chip inside the mobile device, makes routing decisions based on the NFC modes. The first two modes (Reader/Writer and Peer-to-Peer) are routed to the host CPU, while Card Emulation Mode is routed to an SE. Android KitKat's HCE changes this[3]. It allows that commands in Card Emulation Mode can be routed to an HCE service on the host CPU. As shown in the Figure below this is optional – it remains possible to still route commands in Card Emulation Mode to an SE.

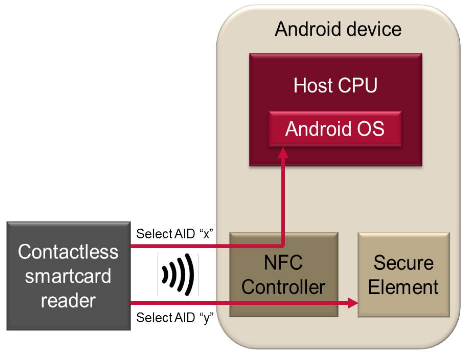


Figure 1: Android operating with both SE-based and Host-based Card Emulation

Both HCE services and off-host (i.e. SE-based) services must register the Application ID (AID) that they would like to handle in their corresponding Android manifest, which is fixed at the time of installation. In Android KitKat, the default route for NFC services is to the host. In order for SE-based NFC services to be found, the Android OS registers their AIDs in a routing table on the NFC controller. When a terminal selects an AID the communication will be routed by default to the host or – in case the AID is present in the routing table – to the SE. As a consequence, existing SE implementations will have to be adapted when a handset is upgraded to Android KitKat in order to remain accessible[4].

Android KitKat defines two categories for NFC services: Payment and Other. Payment services – both HCE as SE-based – should register in the payment category. In the Android settings menu there is a Tap & Pay setting where a default payment application can be selected. This setting is independent from default payment selection settings that exist within wallet applications. Avoiding conflicts in the default payment setting requires special attention from developers.

HCE introduces new security risks, which we will discuss in Section 3. In addition to the security concerns, we identify a set of potential functionality issues caused by HCE:

1. Existing SE implementations will have to be registered in the NFC controller's routing table when a handset is upgraded to KitKat, in order to be found by the NFC controller.

2. The routing table on the NFC controller can be modified from the Android OS domain. This introduces a Denial-of-Service threat in case the routing of existing NFC services can be changed by a malware application. This risk may well be limited to rooted devices, as legitimate applications require explicit user interaction to change these settings. See Section 3 for a discussion about rooted devices and the associated risks.

3. An SP using an SE-based NFC solution can allow transactions to be made while inactive, switched off or even without battery power, as long as no user-input is required. While this option exists, some

1. Forecast by ABI Research, March 2013. 2. See other UL white papers on this topic, e.g. "How to Win the Wallet War?" 3. Note that RIM's Blackberry already implemented HCE in 2011. This paper focuses however on HCE as implemented in Android KitKat because it currently dominates the market for NFC-enabled handsets. 4. Note that not all NFC chips support AID routing. In case AID routing is not supported, HCE will be available, and the NFC Controller will route all Card Emulation Mode requests by a terminal to the SE.

SPs of current mobile payment implementations choose to avoid it; for security reasons they require that a device is activated and unlocked. If an NFC service uses HCE to run on the host processor, the option to perform transactions with a switched-off device is no longer possible as the host must be activated (although no unlocking is required).

4. The Tap & Pay settings may lead to confusion for the end-user. It introduces a second location to select a default payment application, next to any other payment applications (such as wallets) that are installed on the handset. The Android API includes a function to check whether the running application is the default payment application. Developers can use this function to prevent confusion regarding the default payment application.

5. Most NFC solutions that are using MIFARE (currently in use by a wide variety of services such as transit and access control) or Calypso cannot work as HCE service[5] in current combinations of Android software and NFC hardware This may be addressed in future Android releases and/or NFC hardware versions.

3. HCE-related security risks compared to SE-based NFC

Android KitKat supplies the new communication channel from the contactless card reader to the host CPU, which enables HCE. In HCE communication always passes through the Android OS. This provides basic security measures (for instance by running each application in its own "sandbox" which prevents that it can access data from any other application).These basic security features are however lost when a handset is rooted. Rooting is the process of allowing users of handsets, tablets, and other devices to attain privileged control, e.g. become a super-user.

We see three ways in which this introduces security risks which are not present in SE-based NFC services:

1. The user can root the device. As a consequence, the user can access all information stored in applications, including sensitive information such as payment credentials. Typically, in payment and transit applications the SP wants to prevent such user access, for instance because it implies malware could also access this data. Estimates are that only a small minority of Android handsets is rooted – but this minority still adds up to millions of devices.

2. Malware could emerge that can root the device. For previous Android versions, Android exploits have emerged that root the phone from a malware application. While these exploits had a limited reach (the malware was not available from official download channels), it is a potential risk that has to be considered. It has proven to be difficult to fix an identified exploit in Android due to the long Android update process: new Android versions typically take a long time to reach the majority of handsets, while a substantial

5. Whether SE-based implementations of these services will work on an Android KitKat handset is unknown to us at this point in time.

set of handset types are not updated to the new version at all. For example, estimates are that currently around 24% of Android devices are still running Android 2.3.3-2.3.7 [6], a version available since February 2011. If an exploit would appear in the future, it may therefore take a long time for a fix to be installed on all devices.

3. In a situation where a handset is lost or stolen, a malicious user can root the device or access device memory by connecting it to another device. This malicious user can then gain access to all information stored within the application. This introduces severe risks. For instance, the malicious user can use the sensitive information in his own payment application to conduct fraudulent payments.

The basic security features of Android offer limited security, which can be circumvented relatively easily by rooting the device.

4. Mitigation techniques for HCE-related security risks

Mitigating the HCE-related security risks can be done in two different ways. One is providing a more secure location for storing sensitive data and the other is applying security mechanisms to make the location more secure.

4.1 Locations to store sensitive data

The HCE service runs within the Android OS. An SP may require a more secure location to store credentials, generate and process the communication and perform cryptography. We identify four basic location options, which have a different balance between risk mitigation and associated costs. These options are illustrated in Figure 2.

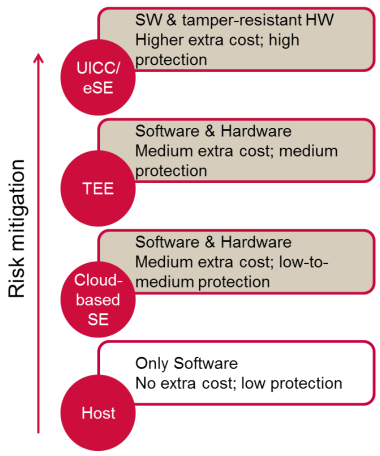

SW & tamper-resistant HW
Higher extra cost; high protection
UICC/ eSE
Software & Hardware
Medium extra cost; medium protection
TEE
Software & Hardware
Medium extra cost; low-to-medium protection
Cloud-based SE
Only Software
No extra cost; low protection
Host
Risk mitigation

Figure 2: Storage options for sensitive data in HCE services

4.1.1 Host

This is the basic approach, i.e. storing and processing occurs within the application running on Android OS on the host. Apart from Android security mechanisms such as sandboxing, no additional security is added in terms of location for storage and processing.

4.1.2 Cloud-based SE

In this approach, storage and processing of the sensitive data is done in a server somewhere in the "cloud", to which the NFC device can make a connection. This connection is therefore essential to activate the NFC service. Of course, an internet connection might not always be available and the speed of a mobile connection might cause latency issues. A mobile payment transaction must be completed within narrow time limits, for example 400 ms for MasterCard *PayPass* M/Chip or 170 ms for *PayPass* magstripe[7].

Cloud authentication challenges

Today's mobile banking applications face a similar challenge with authentication to the cloud as the card emulation approaches described in this white paper. Yet, many mobile banking applications do not require an SE on the mobile phone and therefore accept the security offered by cloud-based solutions. There are however important differences between *mobile banking* and *mobile payments* which can explain the different security requirements. For instance, mobile payments require that transactions are possible to a very wide range of counterparties (merchants) without too much hassle with real-time verification mechanisms etc. In mobile banking applications typically transactions can be limited to a subset of bank accounts, triggering additional security mechanisms for transfers above a certain limit or to an unknown bank account. Also, in money transfers done in mobile banking the liability for a security breach often lies with the party that also suffers the damage of the breach. This party can make his own risk assessment and security design of his mobile banking service, to a large extent independent of scheme regulations. This is in contrast to mobile payments, where multiple parties play a role in the chain (four-corner model) which means that parties must make arrangements for who is liable in which case. This creates the need for certification, which puts requirements on a mobile payment service. These requirements must be taken into account when a party designs the security of its mobile payment service.

6. Statistics from http://developer.android.com, December 2013.
7. Mobile MasterCard *PayPass* Application Note #6.

A real-time calculation on a Cloud-based SE cannot guarantee this kind of transaction speed. Therefore, Cloud-based SE solutions typically include a tokenization mechanism to allow transactions up to a certain number and value. Google Wallet for instance deploys a form of "tokenization". Section 4.2.3 explains the concept of tokenization.

A fundamental issue with any Cloud-based SE is how to enable the handset to securely identify itself to the cloud. If credentials to the Cloud-based SE are stored inside the HCE service then this severely limits the extra security which can be supplied by the Cloud-based SE solution. This problem could be solved by requiring user interaction for accessing the cloud, which would in turn negatively impact the user experience. Another solution could be to use the hardware SE to authenticate towards the Cloud-based SE.

4.1.3 TEE

The Trusted Execution Environment is a separate execution environment that runs alongside the OS and provides security services to that environment.

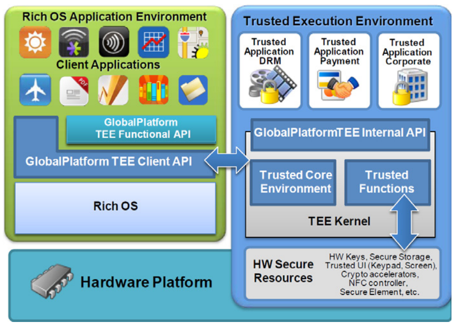

Rich OS Application Environment
Client Applications
GlobalPlatform TEE Functional API
GlobalPlatform TEE Client API
Rich OS
Trusted Execution Environment
Trusted Application DRM
Trusted Application Payment
Trusted Application Corporate
GlobalPlatformTEE Internal API
Trusted Core Environment
Trusted Functions
TEE Kernel
HW Secure Resources
HW Keys, Secure Storage, Trusted UI (Keypad, Screen), Crypto accelerators, NFC controller, Secure Element, etc.
Hardware Platform

Figure 3: TEE architecture (Source: GlobalPlatform Inc.)

As illustrated in Figure 3, the TEE isolates access to its hardware and software security resources from the OS and its applications. The TEE runs its own separate OS and therefore is not compromised when the main OS is rooted. In that way, the TEE can be used to provide a higher level of security than the basic approach described in Section 4.1.1. It does not reach the security level provided by an SE because it does not have the SE's tamper-resistance.

Note that TEE standardization is not yet finalized.

4.1.4 UICC or embedded SE

This option offers the most advanced form of security on the Android device. It is questionable whether this option in combination with HCE really makes sense to an SP, because it seems to provide no additional advantages over traditional, SE-based NFC. It adds complexity in the routing through the Android OS where a direct link to the SE is available.

4.2 Security mechanisms

A wide range of mechanisms exists to make applications more secure. We list a selection of these below. In principle, these mechanisms can be applied to the four locations specified above and combined with each other to provide increased risk mitigation. Obviously, enhancing the protection typically leads to extra steps for the user to execute and/or developer to implement. There is a trade-off between security, end-user convenience and costs that the SP should consider.

4.2.1 User and hardware verification

Payment transactions can be made more secure by verification of the user and/or the hardware that is used in the transaction. Typical verification mechanisms to enhance security include the verification of:

• What the user *knows* (Username/password combinations, PIN, etc.)
• What the user *has*. For example Device ID, smartcard reader, sticker, etc.
• How the user *behaves*. For instance if a payment is done in geographically distant places very quickly after one another, such payment transactions could be denied.
• Biometrics. The use of biometrics for user authentication receives increasing attention, for instance by use of finger-print-scans, voice- and facial-recognition, iris-scans, etc.

4.2.2 Transaction constraints

To limit the impact of potential security breaches, transactions could be limited in various ways, e.g.:

• Only online transactions (Check transaction parameters in systems from the issuing bank)
• Only allowing low value and/or limited number of transactions per timeframe
• Country limitations

We note that such transaction constraints cannot be adapted by a malicious user in a fake app. These transaction constraints are signed by the issuer with a key that is not present on the card (and therefore not present in the app). The constraints can therefore not be manipulated.

4.2.3 Tokenization

In the scope of mobile payments, Tokenization is often used as a mechanism to overcome timing issues of Cloud-based SE solutions. This means that the tokens need to be stored in the application, where they are still at risk. However, the use of a token can be restricted in how they are used with a specific merchant, device, number of transactions or category of transactions. In most cases a token can be used to authenticate only a limited number of times. When the tokens are used new tokens will need to be retrieved. In this way the risk is limited compared to the case where all payment details are stored inside the application. For this tokenization risk to remain limited it should only be possible to retrieve tokens under certain requirements (like user verification or pushing to the device from a different environment).

The payment schemes MasterCard, Visa and American Express are standardizing a tokenization mechanism for online and mobile payments[8]. MasterCard has announced a first release for mid-2014[9].

4.2.4 Android OS checks

An Android application is able to verify system settings and can detect whether a device is rooted. Given the risks associated with rooting of a device we would recommend an HCE service to check for this kind of settings (developer options and root access) and take appropriate action as soon as those settings are detected.

4.2.5 White-box cryptography

White-box cryptography means that the key is obfuscated by storing it within code of the cryptographic algorithm. The aim is that the key cannot be retrieved even if the original source code is available. Thus, in card emulation white-box cryptography can be used to hide sensitive data within the card emulation application.

A drawback of white-box cryptography in the context of card emulation is that the distribution of NFC card emulation applications becomes more complex and costly. The code of each application needs to be unique as the key is hidden inside the code. This would require dynamic loading of source code into the application at the moment of personalization.

A further drawback is that the application's performance may be impacted by the obfuscation.

5 Potential HCE impact on NFC ecosystem

HCE might accelerate the introduction of NFC services, because it provides an alternative, more-simple-but-less-secure way to provide an NFC card emulation service. In this way, it has great added value for SPs that can accept a reduced level of security in exchange for an improvement of other factors such as time to market, development costs and the need to cooperate with other parties. In these cases, HCE would make life for SPs considerably easier and could eliminate the role of Secure Element Issuers. The role of the Trusted Service Manager may also change significantly with HCE, from the personalization of an applet on the SE to personalization of an HCE service.

Thus we can conclude that HCE would have a substantial impact on the NFC ecosystem.

Note however that SPs must be fully aware of the security risks caused by not using the hardware-based security provided by the SE. Let's consider the example of an open-loop mobile payment service that allows for high-value payments and aims for widespread adaptation. While in those cases the advantages of using HCE would certainly be welcome to an SP, using HCE also opens up the threat for fraudulent exploitation of this application, as described in this paper. The sheer value potentially flowing through the payment application once the solution sees mass market adoption makes the potential impact of a security breach very high. Whether the identified vulnerabilities are actually exploited depends to a great extent on the "business case" for a hack, which obviously requires thorough analysis as part of a risk assessment. With these considerations in mind, the risks associated with HCE could be considered too high for this example.

For open-loop payment services specifically the response of payment schemes plays an important role. Using HCE Google has already introduced an SE-less payment application, but standardization (EMV) and approval from payment schemes (see box text) would be required for HCE to become a widely accepted solution for the full scope of

8. https://newsroom.mastercard.com/press-releases/mastercard-visa-and-american-express-propose-new-global-standard-to-make-online-and-mobile-shopping-simpler-and-safer/
9. MasterCard Global Operations Bulletin No. 12, 2 December 2013

open-loop payment services that we are used to from the physical banking cards. It is uncertain at this point how major payment SPs (banks) as well as payment schemes respond to HCE.

On the other hand, for services such as low-value closed-loop payment the advantages introduced by HCE may outweigh the risks.

It is important to realize that a major security breach of HCE-based payments could impact the sector as a whole. Such a breach could negatively influence the perception by consumers of mobile payments and NFC card emulation in general, regardless of whether the breach was limited to HCE security risks only or could also occur in SE-based card emulation solutions.

6 Conclusion

Certainly HCE is an important development in the world of NFC on mobile handsets. It "democratizes" NFC Card Emulation, as it no longer requires access to an SE and thereby introduces an optional degree of freedom for SPs. The access to the SE is often identified as a major challenge in the business plan of SPs. At the same time, any SP planning to go the HCE route will have to consider the security implications of by-passing the security provided by the SE. Android OS by itself is not a secure location to store sensitive data. Risks can be reduced by possible countermeasures such as devaluating stored data (e.g. by making it valid for a single transaction only) and/or storing data in other locations.

For some services SPs may consider the security risks caused by the by-passing of the SE's hardware-based security too high. For instance for high-value open-loop payment systems this could be the case. For other services – such as low-value closed-loop payment systems – these risks may be acceptable. Note that for open-loop payment services approval from payment schemes would be required for HCE to become a widely accepted solution.

Overall, UL believes that HCE will accelerate the introduction of NFC services by providing an alternative, more-simple-but-less-secure way to provide an NFC card emulation service.

Certification

The card networks such as MasterCard and Visa have prescribed clear implementation guidelines with which card issuers must comply. The presence and use of an SE is generally a key requirement in these guidelines. For example, MasterCard state the following in their Mobile *PayPass* Issuer Implementation Guide[10]: "A mandatory requirement of the Mobile *PayPass* program is that the Mobile *PayPass* application on the mobile handset must be stored and executed within an approved SE" and "The mobile phone itself is not considered a secure personal device and therefore any payment processing must occur within the SE." Visa has equivalent certification requirements for payWave.

10. Version 10 July 2013

Contact details

UL Transaction Security
info@ul-ts.com
www.ul-ts.com