CLOUD BASED SECURE ELEMENT IMPLEMENTATION FOR ANDROID HOST CARD EMULATION

WIRAJ GUNASINGHE

UNIVERSITY OF COLOMBO

2018

CLOUD BASED SECURE ELEMENT IMPLEMENTATION FOR ANDROID HOST CARD EMULATION

Wiraj Gunasinghe

Supervisor Name: Dr. Chamath Keppitiyagama

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE IN
INFORMATION SECURITY
SCHOOL OF COMPUTING
UNIVERSITY OF COLOMBO

2018

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Wiraj Gunasinghe
14th July 2018

Acknowledgment

First and foremost I would like to offer my sincerest gratitude to my supervisor, Dr. Chamath Keppitiyagama, for his guidance as my supervisor. And also he introduced me to Near Field Communication(NFC) during one of his lectures. His feedbacks and technical guidance was very helpful to go forward with this thesis.

I also highly appreciated to countless contributor of Open Source programming community for providing many tools and frameworks that I have used to develop this implementation project.

I also express my gratitude to Mr. Dayan Bandula who taught me mobile payment eco system related lessons as Principle Software Architecture during my first job.

I am also thankful to everyone who directly or indirectly give their support in this thesis.

Finally I want to thank my parents, sister and specially my wife for supporting me throughout my masters studies.

DEDICATION

TO MY FAMILY, TEACHERS AND FRIENDS who didn't hesitate to criticize my work at every stage without which I would neither be who I am nor would this work be what it is today.

Contents

Abstract

Mobile banking and mobile payment have evolved tremendously over the last few decades. Currently contactless card payment based on Near Field Communication(NFC) technology emulates physical secure element embedded on mobile device by using Host based Card Emulation(HCE) technology. We provide Cloud based implementation for Android HCE, to show how HCE based mobile payment solutions can be operationally secured using cloud technologies. We present full design and implementation of the security concerns involving with the project. We also provide a comparison of implemented system attributes with Payment Card Industry Data Security Standard(PCIDSS) guidelines to ensure the standard of the proposed system.

List of Tables

List of Figures

List of Symbols

$\oplus$	XOR
$\sum$	Sum

List of Abbreviations

AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
AES	Advanced Encryption Standard
C-APDU	Command APDU
CMAC	MAC according to NIST Special Publication 800-38B
EMV	Europay, MasterCard, and Visa
GSM	Global System for Mobile Communications
HCE	Host Card Emulation
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
MicroSD	Micro Secure Digital
MNO	Mobile Network Operators
NFC	Near Field Communication
OTA	Over-the-air
PCD	Proximity Coupling Device
PCI DSS	Payment Card Industry Data Security Standard
PICC	Proximity Card
QR	Quick Response
R-APDU	Response APDU
RFID	Radio-frequency Identification

SE	Secure Element
SIM	Subscriber Identity Module
SSD	Supplementary Security Domain
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TSM	Trusted Services Manager
UICC	Universal Integrated Circuit Card
UID	Unique Identifier

Chapter 1

Introduction

This chapter discusses the motivation behind this thesis and provides an introduction to the mobile payment systems with associating new technologies. Specifically this chapter provides the background of the research intent that this thesis trying to solve. It also discussed about the methodologies and procedures which will be use to implement this proposed system.

1.1 Security aspects on Mobile Payment

Mobile phone based financial services related to Electronic cash gain huge attention and popularity at the beginning of the mobile era. The banking industry and customers all incur the less flexibility and high handling overhead of Physical cash and begins to adopt virtually cashless mobile banking systems. For financial transactions, security can be considered as most important factor apart from the availability and usability of the solution. Then mobile handset developers includes physically embedded Secure Element(SE) types to store payment credentials in order to mitigate from majority of attack vectors related to mobile eco systems . Along with Near Field Communication(NFC), embedded SE provides high security levels to protect operationally important data but the mobile handset vendor specific access rules on embedded SE prevents other mobile based financial application providers from using embedded SE to protect their own data. In order to solve this limitation Google has introduced Host based Card Emulation(HCE) with their Android 4.4 release and access limitations imposed on embedded SE are bypassed by introducing software based SE implementation on mobile handset. Apart from the intended usage of HCE, this new technology created new set of attack vectors which can affect to the confidential operational data stored in mobile application and most of the mobile payment application vendors reluctant to adopt this new technology with their existing mobile contactless payment applications.

1.1.1 HCE mobile payments

HCE is a software architecture that provide virtual representation of various SE as software representations and provides simple, convenient and flexible approach for many complex mobile payment related design and operational issues. Even though HCE experiences a ground breaking entrance to mobile payment arena, many new players are reluctant to use it for their mobile payment systems. Many vendors prefer embedded SE implementation or Trusted Execution Environment based application deployments because they trust on hardware based solutions even with lot of dependencies. Meantime many vendors use HCE implementations with trusted hardware environments to use as secure storage area to save secure payment data.

1.2 Research intent

The research intent of this thesis is to provide secure, operationally flexible HCE implementation based on cloud environment. Tokenization, cloud security principles and mobile security guidelines are used to achieve the security goals covered with this research.

1.3 Methodology

Implementation methodology is basically consists of moving mobile based SE implementation into cloud architecture and handling relevant security concerns arises with the cloud and mobile environments. Distributed authorization schema is introduced over the cloud components in order to maintain data confidentiality and integrity requirements. Investigating and resolving steps of the existing and new attack vectors are also included into the implementation. To evaluate the compliance and effectiveness of the cloud SE implementation both quantitative and qualitative measurements are used. Baseline PCIDSS guidelines are evaluated over the cloud SE implementation and the results are analyze qualitatively to measure the proposed system compliance. As quantitative measurements, transaction time is taken for several SE operations and compared with mobile HCE implementation values for the same operations.

1.4 Scope of the Thesis

The scope basically covers the cloud based SE implementation for android mobile based HCE application with ISO7816-4 compatible SE.The mitigation steps and procedures of the possible attack vectors available in the proposed system is also included into the scope of this thesis.

1.5 Organization of the Thesis

The rest of this thesis are structured as follows:

Chapter 2 discussed about background context and related works. It provides details about mobile payment systems, NFC technology stack, modular systems and cloud computing related details.
Chapter 3 discussed about overall design of the proposed implementation and related solution components.
Chapter 4 defined the implementation of the concepts. It discussed about the common security flows related to HCE mobile payment systems and proposed mitigation procedures.
Chapter 5 discussed about results of the implementations and comparisons in terms of standard mobile transaction benchmarks.
Chapter 6 discussed the conclusion and possibilities of future work for improve this solution.

Chapter 2

Background Context

2.1 Mobile Payment evolution

Mobile technology is revolutionizing the global banking market along with payment industry since the dawn of 2000. New opportunities, flexible customer support and high scalable solutions are the outcomes of new emerging mobile landscape and traditional customers are tend to use new technology day by day. In earlier days banks has not faced many challenges because of the limitations of the technology and accessibility mobile arena. But due to the huge success of mobile world (See Fig. 2.1), users begin to look for mobile centric services and ultimately many entities had to change their traditional business models to be compatible with mobile environments.

2.1.1 Involvement of banks

Due to high competency service providers are rapidly deploying their new service into traditional industry eco systems and ultimately banks had to integrate their legacy systems with mobile banking systems. After the revolution of Internet and its related services, mobile technology is considered as must have facility for any customer banking solutions in the world. Although mobile banking have evolved from traditional banking to contactless payments, still there are huge technology and migration gaps can be found with legacy systems. Due to the complexity of existing system landscape, legal background, banking regulations and customer awareness there are no single model has been successfully adapted for all major mobile payment deployments.

But banks are already invested huge amount of money on mobile technology in order to maintain efficient end-to-end process and reduce high transactions cost. Then consequently mobile banking adoption among new customers has increased (See Fig. 2.2) and reduction of transaction cost affects to mobile customer retention and total online and offline mobile transactions. Again banks has to increase total transaction cost due to system maintenance but both users and banks found mobile banking is one of the essential channel for day today transactions. Governments also pushes mobile technology related

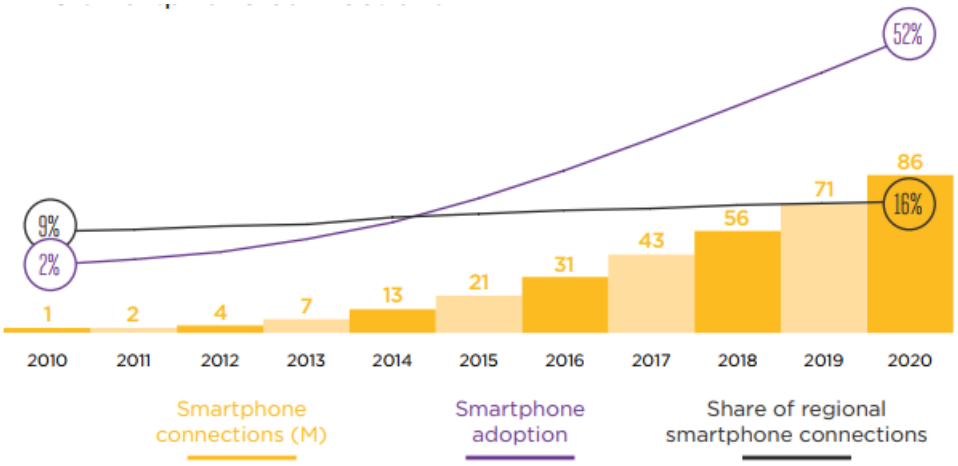

Fig. 2.1. World mobile device usage predictions.[1]

deployment because of the huge mobile device usage all over the world and it clearly addresses large sector of population of each country who does not actively involving with traditional banking facilities. Simple Quick Response(QR) code related money transferring services, SIM centric stored value services, P2P money transactions and bluetooth, NFC related advanced payment methods become very popular among small to large merchants because of the expansion of mobile infrastructure and subscribers.

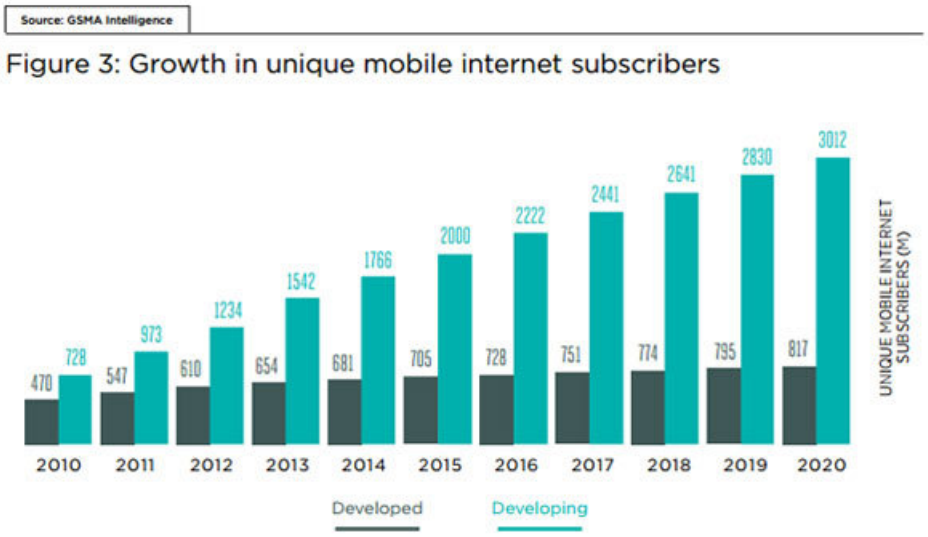

Fig. 2.2. Growth in unique mobile internet subscribers.[2]

Big mobile industry players like Apple, Android and Samsung are working with many inter-industry payment organizations like Visa, MasterCard, EMV and GlobalPlatform to innovate and standardize new technologies and solutions. Alternatively online payment solutions those are not connected with credit card companies like PayPal are also develop there systems in more internet and web centric way in order to address direct Internet and mobile users. Finally brooming mobile payment industry directly market their solutions to audience by promoting potential reduction of transaction cost, high accessibility and efficient service, and so far those factors seems fulfilled in spite of huge designing and

deployment cost factors. The innovation of huge number of new technologies, fast adoption rate, different similar services and lack of coordination among service providers has created confusion for average mobile customer and that could lead to less customer involvement and low rate of new subscriptions.

2.1.2 Contactless Payments

With the technical advancement of Radio frequency identification(RFID) data transmission, smart card industry add additional contactles interface to their mag stripe based cards to improve security and usability and finally contactless payment cards was invented. But card manufactures maintain all interfaces on card to maintain the compatibility with legacy terminal devices. Currently many smart cards have two or multiple payment interfaces implemented on card chip to provide compatibility for different issuers but most of the smart card vendors still using magnetic strip as card interface are migrating to EMV dual interface smart cards by issuing contactless cards in order to increase the usage of smart cards, integrate with new available services and ultimately the final goal of issuing contactless cards is to increase the usage of smartcards of target customers. Contactless payment cards are basically designed to deal with with low-value and high speed transactions like transportation and access management. EMV standardize the contactless payment ecosystems and ensures fast and reliable transactions while maintaining high transaction security levels. Not like magnetic stripe smart cards, contactless smart cards are considered to be intelligent memory storage which does not allows to access resident data without proper identification steps. The chip resides contactless card act as a micro computer which can generate unique transaction cryptograms per transaction with specific cryptographic keys and standards, while card terminal has to have proper agreed process with card micro computer to verify requests. Unlike traditional magnetic strip cads this dynamic verification process helps to protect smart card against conventional attack vectors but consequently it opens new set of advances attack vectors.

2.1.3 Beyond conventional card types

Due to huge mobile based deployments and market share, contactless card products are invented with different physical dimensions and those are easily embedded with mobile devices and other host devices. So nowadays we have contactless stickers, tags with different sizes and shapes. Sometimes technologies like host based card emulation enables user to created completely virtual smart card with any custom standards. These new card types reduces the card manufacturing and maintenance cost and promote usability with wide range of contactless card acceptance devices. Mobile devices with embedded NFC enabled chips changes the direction of mobile payments and users are allowed to emulate their physical smart cards on mobile devices and the same time allows to control their payment instruments with personalized secure domains with specific preferences.

Over-the-air communication capabilities with mobile device, changes the card provisioning domain and ultimately user experience is simplified. This added benefits attracts new users to the existing payment eco systems and then most of the mobile based vendors tend to integrate their existing mobile payment solutions with NFC enabled devices and consequently many new technologies are invented to solve common migration issues. HCE is one of the ground braking alternative to resolve most of the conventional and operational issues arises with legacy deployments.

2.2 Near Field Communication

NFC is short range, bidirectional, proximity coupling technology based on smart card standard ISO14443 that is used to transfer data over a distance of up to 10 cm. ISO 18092 standardize physical and data link layer of NFC peer-to-peer mode and further discussions on NFC protocol is mainly listed on NFC Forum.[1] NFC protocol operates at a standard frequency of 13.56 MHz and can support data exchange rate ranging from 106 kbps to 424 kbps. For mobile payments it helps mobile devices to initiate short range radio connection with other active devices when touching them together. In compared with wireless technologies NFC is considered as simple and robust protocol. Without establishing secure channel or key sharing, NFC transactions can be initialized just by touching a reader or other active NFC device.

2.2.1 NFC communication modes

The devices participating in NFC transactions can be considered either active or passive depending on the communication capabilities of the device. Active devices has its own energy source to generate RF field which can directly turn on the embedded chip on passive device. Small transmitters, tags and NFC cards that can communicate with other NFC devices without the need for a power source of their own can be considered as NFC passive devices. The passive components can only be boot up with related active devices and can not process any information send from outside entities. All NFC device categories support three communication modes:

- In Reader/Writer mode NFC devices behaves as a reader for passive and any other active devices. By using collision avoidance algorithm NFC reader can resolve multiple NFC passive devices in the field and once device is selected, NFC reader can read or write data to the tag.

- When NFC devices exchange data with each other, the communication mode can be defined as peer-to-peer mode and operations of this mode is defined under ISO18092 specifications . Peer-to-peer file sharing, money transactions are the use cases of this mode and Android Beam technology uses this mode for its operations. In order to establish connection client device (peer-to-peer initiator) needs to polling for host (peer-to-peer) target systems.

- Card-emulations mode added after the invention of HCE technology and when a device is active in this mode, it behaves like a contactless smart card and any compatible active device can communicate with the device. Android, Blackberry and Windows devices are can support card emulation mode. When device activated in this mode, any smart card or protocol can be implemented withing the device and subsequently device can initiate service as same as real smart card.

[1] http://www.nfc-forum.org

2.2.2 NFC deployment operational models

According to the real word use case NFC adoption can be categorized under following groups:

- NFC with embedded or external SE in mobile handset that stores tokenized secure payment credentials
- NFC with a trusted execution environment (TEE) can store secure payment details within the secure area of main processor which runs on proprietary operation system. Payment tokens can be stored and used withing the TEE of the mobile device.
- NFC with host card emulation (HCE) can replace the SE in mobile handset and this mode can enable the mobile wallet application to define virtual payment card structure.

2.2.3 NFC protocol standards

NFC protocol is basically standardized under ISO14443 and ISO7816 protocols. These protocols defined the card capabilities and card command standards. Fig. 2.3 shows the NFC protocol stack with mandatory standards.

ISO 7816-4 defines the message format to communicate with between NFC devices. This format is named as APDU and its defines nested TLV formated command standards. Different vendors use different custom APDU command versions for their native communications.

ISO 14443-1 defines the environmental factors that card should sustain without damage to the stored data or the functionality.

ISO 14443-2 standard defines the radio frequency power can signal interface of the NFC device.

ISO 14443-3 defines the initialization process and anti-collision mechanism for NFC Type A and Type B tags.

ISO 14443-4 defines the data transmission protocols involving with application layer functionalities.

2.2.4 Related threats

Standardization and integration of NFC devices into mobile handset is considered as ongoing process, there is a potential of happening security or privacy issues on mobile NFC deployments. Most of the use cases can be avoided by following standard industrial practices but lack mature inter-industry implementation guideline for NFC and mobile environments can still lead to unexpected problems. Following threat categories can be found with mobile NFC deployments.

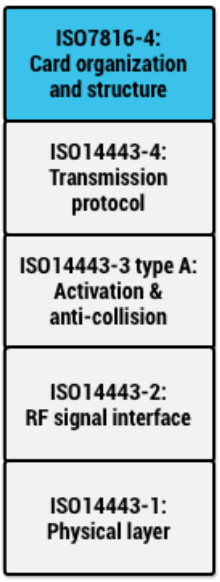

Fig. 2.3. NFC protocol stack. [3]

- Skimming on application ids can affect most of the Global platform based smart cards shows the application directory related to the stores application on card. Then third party can easily detect what kind of applications are already installed in relevant smart card. By using HCE card emulation, this data leakage can be prevented by implementing custom APDU interface to populate data excluding sensitive information.

- Denial of services are applicable for almost all mobile or web components and also applicable for NFC smart cards. Usually just by touching a NFC tag with relevant active devices card begins to respond and this unnecessary read and write cycles[4] can ultimately lead to data corruption on the smart card memory.

- Phishing on smart card data can be initiated using social engineering attack which can replace or modify tag data. Special tag transporters or signatures would be suitable way to mitigate phishing attacks.

2.3 Secure Element

According to the Global Platform specifications[5], SE is defined as: a tamper-resistant platform (typically a one-chip secure micro-controller) capable of securely hosting applications and their confidential and cryptographic data (e.g., key management) in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities. SE can store confidential data like secret keys, cryptograms, PAN, stored values etc. Currently SE is presented on three major formats, namely UICC, embedded SE and Micro-SD card based SE[6]. Most of the SE operation systems are powered by JavaCard platform and applications resides on SE can be written according to the JavaCard programming guidelines.

2.3.1 UICC SIM cards

The Universal Integrated Circuit Card(UICC) SE type is the most commonly available SE implementation deployed to the systems. The mobile operator can directly access to UICC chip card to provision related application and verification data stored in SE. But the mobile OS does not have any access to the UICC and when UICC vendor need to deploy new applications in to SE, both mobile operator and application issuer have agree and operate on same UICC with adequate access controls. In order to control third party access on UICC in secure manner, Trusted Service Manger can be introduced to the mobile eco system.

TSM basically delegate access requirements and connects the trusted application vendors with MNOs. Since the key delegation on UICC and access granting permission handling are still available only to the UICC vendor account, so third parties can not directly create secure domains on UICC without proper access permissions. This limitation affect to the availability and flexibility on UICC application deployments. But this strict access rules enforces the security requirements but mobile device and application vendors again has to trust UICC vendor for their deployments.

Apart from OTA provisioning, UICC can also be accessed by using device resident NFC controller. After the successful UICC to NFC bridge, IPC command interface like AT commands can be used to access UICC through device baseband processor. UICC interface can be connected with NFC controller using Single Wire Protocol (SWP) and here again UICC is only accessible to the baseband processor. Mobile device application processor can not access UICC interface so the complete isolation can be guaranteed.

2.3.2 Embedded SE

Embedded type of secure elements are tightly couples or embedded in mobile device by the device manufacture. Unlike UICC access controls, mobile device vendor have all the access privileges to the secure element. Near Field Communication Wired Interface (NFC-WI) is the recommended protocol to communicate with embedded SE. According

to NFC-WI protocol embedded secure element can communicate with outside entities by using:

- Wired mode
- OFF mode
- Virtual mode

In wired mode, mobile OS can access embedded SE and application installed on mobile OS can access secure element with relevant access keys. In the OFF mode, no communication exists with NFC controller and embedded SE is only used for mobile OS internal usages. In the virtual mode, the embedded SE can be accessed by external active reader and act exactly same as contactless smart card. The default mode of the embedded SE is determined by the mobile device vendor and there are no standards to enforce the availability and accessibility of the embedded SE. Since the mode of embedded SE can only be managed by device vendor, this SE type inherently has practical limitation with mobile payment solutions. As an alternative starting from android build 4.4 google has introduced Host based Card Emulation (HCE) to overcome mobile vendor dependency on embedded SE.

2.4 Trusted Execution Environment

Trusted Execution Environment(TEE) is a secure area of the main processor in the mobile handset that allows to store credentials securely. TEE can facilitate to safe execution of resident NFC applications with the help of its own proprietary operating system. TEE operating system basically segregate the allocated hardware and software resources from main mobile OS by enforcing access controls on secure resources. In TEE operating system, the processor cores are divided to form new virtual cores in order to create normal and trusted environments. Normally this trusted environment can access other low secure portion but low secure portion can not access the trusted environment. Transition between this two environments can be under the supervision of monitor mode and monitor mode can handle which portion need to be activated according to application usage. This isolated nature of TEE can be used to protect the sensitive applications data even if ROM is compromised when the Android operating system is rooted. GlobalPlatform specification defines how applications can securly share TEE, including how to securely communicate between applications running on mobile operating system and TEE (See Fig. 2.4). Even though TEE has built in security architecture, it is still vulnerable to cloning attacks and buffer overflow attacks. But when device unable to facilitate embedded SE for payment application, TEE storage can handle significant secure requirements.

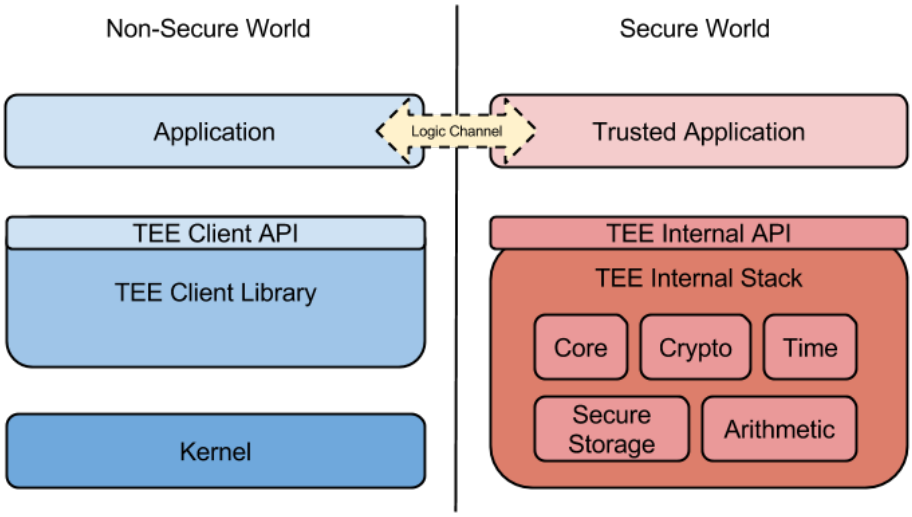

Fig. 2.4. Global Platform TEE. [7]

2.5 Host-based Card Emulation

The term Host Card Emulation(HCE) was introduce in 2012 by SimplyTapp, Inc. to defines the ability of the mobile device to act as virtual secure element. Before that Blackberry research group implemented a smilar process in their handset device Blackberry Bold 990 device in 2011 by defining the technology as Virtual Target Emulation. Later Google has adopted the technology and release it with their Android 4.4 release. Currently both Blackberry and Windows 10 mobile platforms can also emulate HCE related services. Both Visa and Master card currently working on standardizing HCE transactions on their contactless mobile platforms.

Table 2.1
Token management models for popular payment applications.

Service Provider	Token Management	Remarks
Google Android Pay	HCE	-
Apple Pay	Embedded SE	-
Samsung Pay	NFC and HCE	Payment tokens stored in TEE

HCE is a software architecture that provides virtual representation of smart cards using software implementation. HCE is designed according to the card emulation mode of the NFC device. Prior to the introduction of HCE, mobile NFC transactions were mainly coupled with vendor controlled embedded SE. After the introduction of HCE, software based implementations of virtual SE replaces the embedded SE dependency from payment applications. Normally embedded SE used in NFC implementations are beyond the control of mobile wallet service provider because the management of embedded SE secure domains are always reserved to the SE hardware vendors.

As shown in Fig. 2.5 HCE implementation provides alternative routing path for incoming APDU to the user defined android service, but legacy services are also redirected to their relevant embedded SE destinations. According to Fig. 2.6 HCE applications can route their APDU even without having embedded SE. However storing payment credentials and other sensitive data in mobile OS instead of embedded SE is considered less secure and that can make host system very vulnerable to attacks. Additional security measures like payment data tokenization according to EMV Payment Token Specification[8] can adhere to improve data security in HCE application. Implementation of cloud bases SE is one of the viable solution to mitigate the risks involving with Android OS:

- Data and application security in rooted devices
- Payment schema relate issues involving with Card not present(CNP) channels.
- Android OS related vulnerabilities

According to the EMV specification, Token Service Provider(TSP) should generate tokenized PAN of the smart card and mobile application can store the token in device (weather on embedded SE, device memory or TEE). Master card PayPass and Visa Pay-Wave specifications are also amended their specification to facilitate cloud cryptograms to accept as valid payment credentials upon present to terminal device.

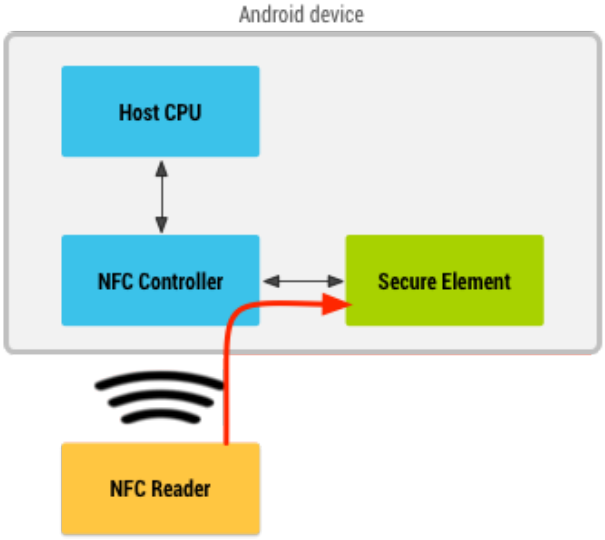

Fig. 2.5. NFC card emulation with a secure element. [3]

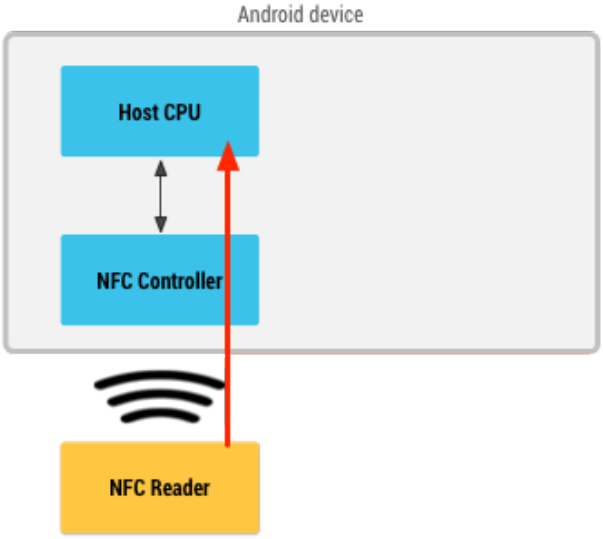

Fig. 2.6. NFC card emulation with host. [3]

2.5.1 Security enforcements

Since HCE implementation not depending on embedded SE, HCE implementations has to completely relies on the Android security model. All the HCE enables applications shares same android sandbox model as like other normal applications but HCE related service component allows two specific security enforcements:

1 - Screen off and Device Lock
HCE applications are only allowed to exchange APDU with external entities if mobile device screen is unlocked. Android enforces security measures to turn off NFC controller and application processor when device screen is off or device is locked. So the HCE related android services can not communicate and this can protect device from unintended interactions and skimming attempts. But this device unlock status which enable to continue with HCE transactions can be altered by simply defining android service attribute `android:requireDeviceUnlock` to false and to prevent this type of alterations, mainly developer can obfuscate the android payment application source to prevent reverse engineering attempts.

2 - Android HCE service permissions
`BIND_NFC_SERVICE` permission allowed only the operation system to communicate with HCE services and ultimately this guaranteed all the APDU commands received to HCE service comes after passing OS filters so ensures that the device NFC controller coupled with external NFC active devices can only access to virtual SE implementation. This permission is used to prevent APDU sniffing from external entities.

3 - Code obfuscation
Code obfuscation can not be considered as direct HCE related security measure, but generally it ensures the confidentiality of the business logic by making code de-compilation and reverse engineering attempts of application artifacts harder. Most of the times android OS can be converted to rooted status by gaining super user privileges, so malicious user can easily decompile android APK files located in `/data/data/` directory in android file system.

4 - Securing HCE storage
Payment credentials can be stored on HCE environment using following locations:

- Cloud based SE
- TEE
- Mobile OS

Mobile host storage is considered as the least secure option compared to TEE and Cloud based storage solutions which can be provided significant protection to HCE transaction credentials. Mobile device storage can only provide application sandbox related isolation but cloud-based SE approach presented with high secure infrastructural solution like the hardware security module(HSM) which is basically used to store master secrets can provide industry level end-to-end security solutions. Due to the expensiveness of the HSM based solution, payment tokens with limited lifespans can be introduced to the HCE

payment environment. This payment tokens can be used to perform off-line transactions considering the availability of the network connection.

2.5.2 HCE security risks

In the HCE model, all the communications are tunnel through Android operating system and hosted system related security issues can be considered as the main attack vectors to HCE deployments. Implementing Multi layer security on HCE mobile payment applications can significantly reduce the possibility of attack.

1 - Android mobile related issues

Device rooting or gaining administrative privileges on device is not difficult on android devices and rooted devices are always susceptible to hacking and exposing android HCE based payment applications in many ways. Bugs, security flows and vulnerabilities of HCE implementation can be identified by attackers after accessing rooted devices. Embedded SE based applications on rooted mobile devices can automatically blocks their transactions and access permission to SE so these security risks are not present on such environments. Sensitive data exposure and malware infections are the most common attack scenarios available on rooted mobile devices. Security issues arises with rooted mobile devices can be controlled by frequently checking rooted status on mobile operating system.

Storing plain payments credentials on the NFC payment device can reveals secure data to third parties. PAN, secret keys can be considered as primary attacking targets, so tokenization on those sensitive information can maintain the confidentiality of the payment credentials.Lack of standards are the next major issue arises with HCE deployments. Only Visa and Mastercard have released their HCE specifications, but apart from EMV tokenization standards not other HCE related standards can be found.

2 - Relay attacks

Both Software and hardware based relay attacks can be considered as the primary attack vector against HCE payment systems. In relay attacks attacker can place relay reader close to the NFC mobile device and a special relay proxy device can be used to emulate as the victim contactless smart card. When relay proxy reader connected with victim device, APDU traffic can be tunnel to legitimate POS terminal through relay proxy. So attacker can analyze the traffic and determine the emulated card structure. Fig. 2.7 shows the components of relay attacks.

3 - Man-in-middle attack

Man-in-middle attack is also possible with HCE payment systems and attacker can place relay between communication channel to sniff the APDU communication and possibly modifies the communication between two parties. In cloud based HCE deployments, at-

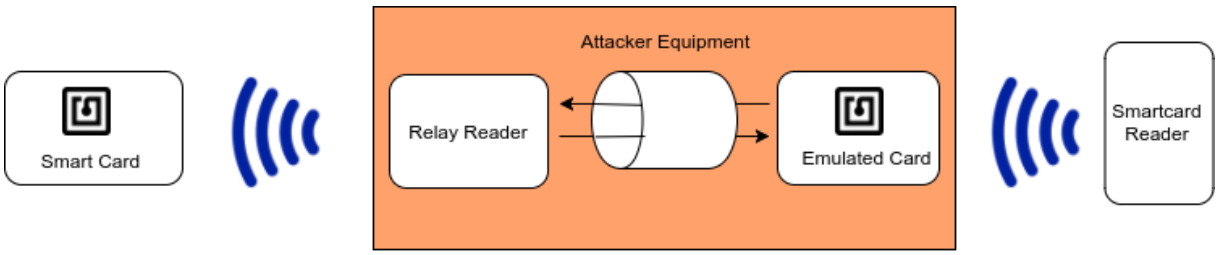

Fig. 2.7. Relay attack.

tacker controlled router can be used to perform man-in-the-middle attacks. This security risk can be easily mitigated by using Transport Layer Security(TLS) protocol with strong cipher suite.

4 - Denial-of-Service Attack

Denial-of-Service (DOS) attack can also be performed on HCE payment application with the help of social engineering. Malicious HCE applications with same AID parameters can be installed on the target device to corrupt routing tables and successful scenarios can collect original APDU commands sent from terminal device. Frequent unnecessary SELECT commands on HCE based target device can drain battery life. Resource intensive malicious application can also be installed on host system to damage battery life and ultimately can affect to the availability of the HCE payment application. If there is no way to identify the legitimate POS terminal from HCE application, attacker can place an additional NFC terminal in close proximity to the actual POS terminal. This malicious terminal can create a AID conflict with HCE application and ultimately payment process can be interrupted.

2.5.3 Summery

Even though the popularity and availability, HCE is still considered as nascent technology that has yet to match standardization, certification and security for large scale deployments. High affordability, flexibility and cost-effectiveness takes HCE into unique stage but still payment system stakeholders needs to come up with universal standards to ensure interoperability.

2.6 Cloud Computing

"I dont need a hard disk in my computer if I can get to the server faster carrying around these non-connected computers is byzantine by comparison."

— Steve Jobs, Co-founder, CEO and Chairman of Apple Inc.

Cloud can be considered as a metaphor for the Internet and cloud computing discuss about the abstraction of infrastructure that used to build cloud based computer services. In recent times cloud computing has become one of most popular topic among technical society and has got lots of attention when it comes to availability and security. Day by day the market share of cloud computing getting increase and the number of applications based on cloud infrastructure also increased significantly. In order to reduce cost and human intervention, modern enterprise tend to utilize cloud services which based on visualization technologies. Efficiency, less human administration, improved utilization, faster deployments and overall financial benefits are the main positive factors behind the popularity of cloud computing.

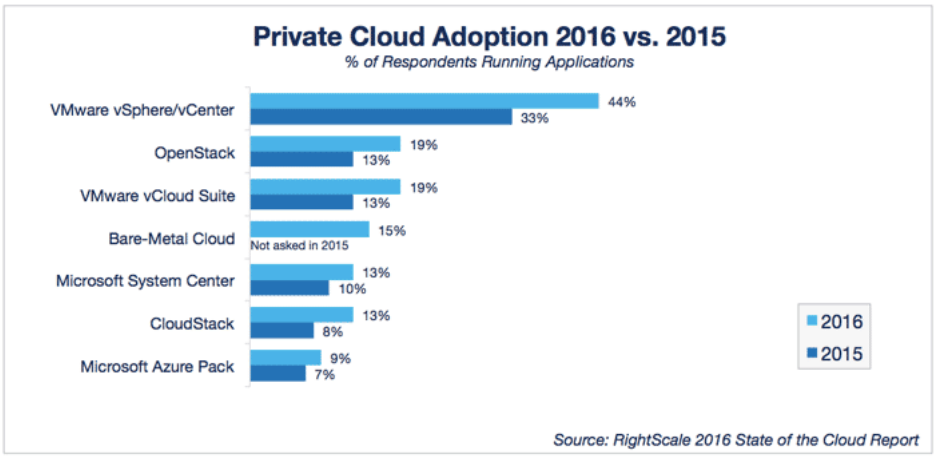

Fig. 2.8. Private Cloud Adoption 2016 vs. 2015. [9]

Unlike the legacy systems, modern day cloud solutions try to maintain mean time to recovery (MTTR) factor very low to improve the recovery time. Most of the times cloud vendor and consumer agrees on service level agreement (SLA) and confidentiality, privacy and access model are depending on that.

2.6.1 Security on Cloud

Cloud platform infrastructures are optimized for cost-efficiency and to cater the high utilization and service availability, vendors has to use cost-effienct hardware that serve as multitenant services running on commodity hardware. The straightforward disadvantage arises with this cost-efficient multitenent model is the difficulty to maintain the tenant isolation this is also tightly coupled with cloud security concerns. Any individual tenant

node in cloud system should be placed in a security sandbox to prevent knowing about any other tenants.

Cloud services can be categorized according to the characteristics and behavior of each service components. In short form this is referred as SPI mode, standing for service, platform and infrastructure.

- Cloud Software as Services
  When consumer can utilize services or applications on the cloud hosted by cloud provider, then cloud service can be categorized as software based service or software as service.

- Cloud Platform as Service
  In this category consumer can deploy applications on cloud platform and cloud service itself provides the basic deployments infrastructure as containers.

- Cloud Infrastructure as Services
  Infrastructure level services provides the processing, deployment, storage and network operations like normal computer where consumer can deploy and run the applications.

Securing the Cloud Components

The base intent of moving to cloud is to save cost on infrastructures and cloud providers also provides cheaper data storage for consumers. The real question arise here is the confidentiality and integrity of saved data on cloud servers and ultimately it became one of the most serious concern related to modern cloud infrastructures. Cloud providers are developing various technologies to maintain those issues but lack of control on hardware to consumer end is again a really complex scenario to solve. Data server physical location, data encryption status and integrity mechanisms on cloud are also considered as proprietary matters and those are only known by cloud providers. By only understanding the Cloud computing mode, security risks can not be identified or foreseen but consumers can standardizer their developments and deployments to ensure the data protection.

Security controls used in both cloud computing and IT environments are not significantly different but level of controls is always fluctuates according to use cases. (Cloud Security Alliance, 2009, p25)

Industry recommended IT auditing framework practices can be apply to maintain the security factor in cloud deployments. Risk management, Compliance, audit, information management, interoperability nature and Legal or Electrical discovery methods can be applied to cloud environments to ensure optimal security levels.

Benefits of cloud computing

Apart from financial advantages, cloud computing mainly popular for the flexibility of application scaling matters. Cloud providers can set up or remove node instantly ac-

cording to consumers demands. Packer filtering, intrusion detection, patch management, general hardening of virtual machines, redundancy on resources, access control with federated identity management are some of advantages of cloud computing over legacy server systems.

2.6.2 Identity delegation

When it comes to cloud complex cloud based enterprise applications, managing identity flow can be identified as a complex yet important task. Because of the complexity of modern enterprise systems, single user access request needs to be send through number of filters sequentially in order to check compatibility of multiple preconditions. After each successful access request which passes all defined filters, identity management system will initialize an authorization chain and further this is used to verify the access rights originator at service destinations. Identity delegation is a concept which is introduced by Active Directory Federation Service (ADFS) that allows set of predefined accounts to impersonate general users. The user account that impersonate the user is called the deligator. The deligator owns the resources and delegate aka client can access service on resource server on behalf of deligator. Identity delegation plays a key role in enterprise security systems.

Deligation workflow can be divided into to two categories according to behavior of the relevant parties such as:

- Direct Deligation
  When deligator directly deligate his permissions on resource server to a deligate, the model can be called as direct deligation.

- Indirect Deligation
  As shown in Fig. 2.9 inderect deligation deligator who owns the resource can deligate necessary permissions to intermediate deligator and then the intermediate deligator deligates to another end user(or again a deligate)

Basic identity delegation work flow scenario is shown in Fig. 2.10 with necessary components.

1 Identity delegation work flow starts with a service request to API endpoint and then the request is redirect back to caller asking for authentications.

2 Then caller need to authenticate himself with Authorization server to get access token.

3 The caller sends a request to the relevant Web server with received token in step 2.

4 Web application contacts Federation Service to get deligation token to access separate entity to do a certain action, which is intended by caller.

5 Federation server sends the token with intended claims about the client and targeted realm.

6 Web server sends a request with deligation token received in step 5 to resource server.

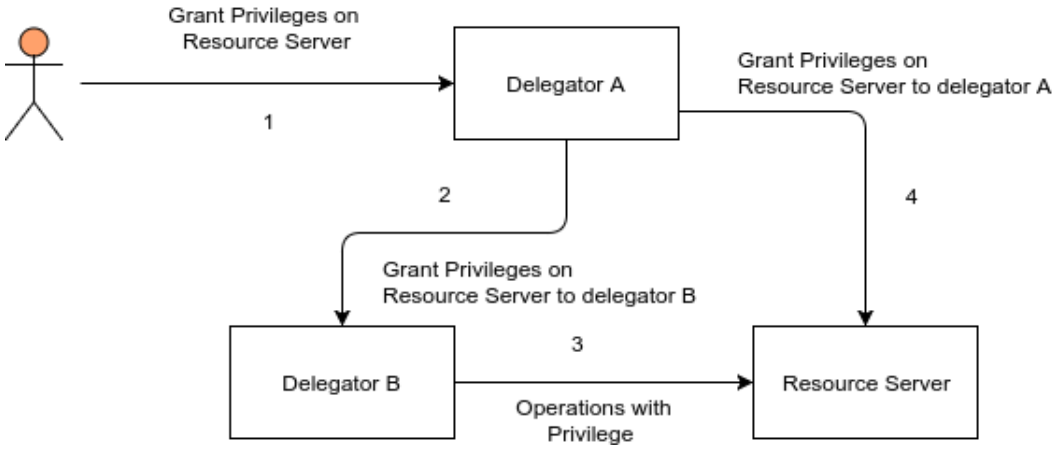

Fig. 2.9. Broker Delegation.

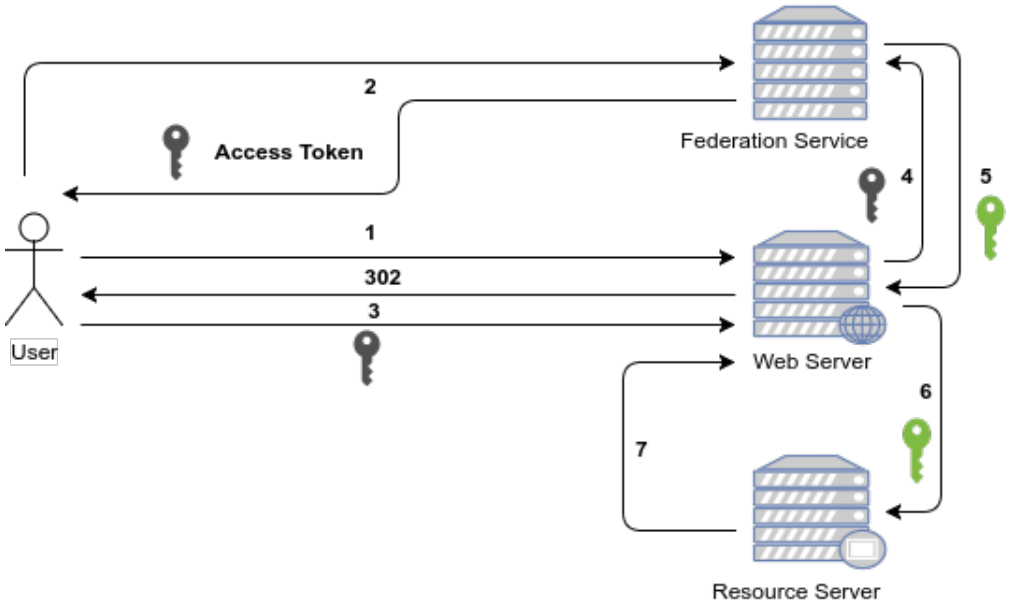

Fig. 2.10. Identity Delegation Work Flow.

7 Resource server can verify the token and send send the intended data back to web server and then to caller.

To handle the security of the token, Federation server can limit the usage and expiration of token. Token revocation services can also be implemented to perform immediate token disposals.

2.6.3 OAuth 2.0

OAuth 2.0 can be considered as de facto standard[10] for access delegation in modern identity management systems. OAuth 2.0 is inherited from its former version OAuth 1.0 with some alteration like OAuth 1.0 signature schema and OAuth 2.0 is considered as highly extensible framework. When using OAuth 2.0 client redirect user to resource service with user identifications but again user is redirected back with token request. Then user can call token endpoint to get OAuth 2.0 token in order to send with each API request to resource servers.

OAuth 2.0 protocol defines 4 types of roles in authorization process.

- Resource Owner
- Resource Server
- Authorization Server
- Client

OAuth 2.0 defines set of grant types to define the behavior of client when obtaining authorization grant from resource owners. According to OAuth 2.0 specification those core grant types are Client Credential grant type, Authorization code grant type, Implicit grant type and Resource owner grant type.

Authorization Code Grant Type is basically suites for web and native mobile applications. Resource owner is initiated the grant flow and then client redirects resource owner to authorization server to get approval. Once the client registered with authorization service, `client_id`, `client_secret` and `redirect_url` is received. When client sends calls for authorization server with above received attribute values, authorization server sends relevant authorization code back to the client via HTTP redirect and it is also visible to the resource owner. Then client exchange the authorization code for and OAuth token by calling OAuth token endpoint. The authorization code received to client should used only one time to get OAuth access token. When authorization server detects multiple usage of same access code, it will revoke all tokens sent with reference to access code.

Client Credential Grant Type itself can become a resource owner. Then client can directly called to authorization server in order to request access token. Unlike other grant types , this grant type doesnt send any `refresh_token` to the client.

Implicit Grant Type authorization flow is initiated by client by redirecting user to authorization server. When requesting access token client has to send `client_id` in the request and then authorization server send the access token to the client.

Resource Owner Grant Type is used when resource owner trust the client application resource owner grant type can be used. Then resource owners access credentials are directly send to the client to initiate authorization request.

2.6.4 JSON Web Tokens

JSON Web Tokens (JWT) defines a standard container structure in JavaScript Object Notation(JSON) to facilitates to transport payload between intended parties. Oauth 2.0 is not defined the access token structure but the token generation process. And the token receiving endpoint looks about the validity and structure of the token. JWT defines the token structure (See Fig. 2.11) which can facilitate both confidentiality and integrity in very efficient manner. When integrating tokens with micro-service architecture, token replication is one of the most critical performance related issue happened in scaling authorization services. Each and every API request against resources servers with token in token embedded in request header has to perform another request to authorization server to check the token validity. So each request has to call another additional request only for verifying token validity for security reasons. This additional overhead can be negatively effected on system architecture and highly performance intensive when it comes to large scale systems.

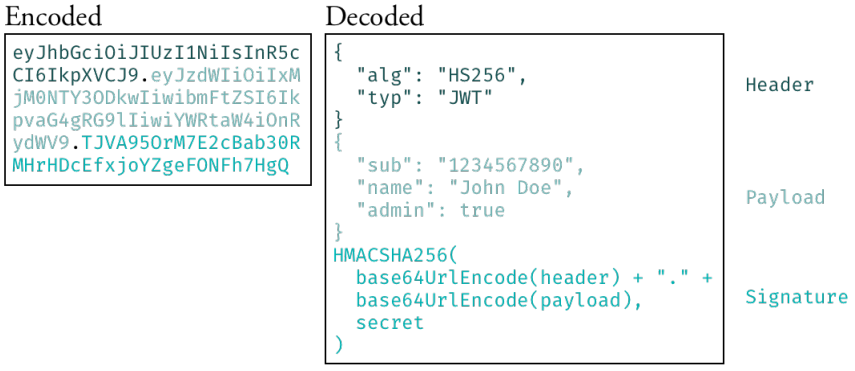

Fig. 2.11. JWT Token structure [11]

JWT token is a self contained token which carries details about OAuth client, user details, claims and signature to verify the content at resource servers. JWT is a base64

url encoded string value which consists of three main sections separated by a period mark.

1 - JOSE Header

JavaScript Object Signing and Encryption(JOSE) header is the first component of the JWT token and it defines the cryptographic operations applied on the JWT token clam set details.

Base 64 url decoded JOSE header contains two mandatory attributes.

```
{"alg":"RS512","kid":"76c029907f2cd361ffe18b4cf2365c396d54f1b6"}
```

2 - JWT Payload

The second part of the JWT token structure is the JWT payload and it contains custom business data of certain token bearer and this payload can further be encrypted if it contain any sensitive information. According to the JWT specification, JWT payload can contain three types of claims:

- Public Claims Claim content is defined by the application that going to consume JWT token and the attribute names should be defined in collision-resistance manner.
- Private Claims This claims are only shared with given identity providers and specific applications.
- Registered Claims Registered in Internet Assigned Numbers Authority (IANA) but JWT doesnt mandate the fields.

3 - JWT Signature

JSON Web Signature (JWS) and JSON Web Encryption (JWE) are derived formats of the JWT tokens and they provides methods to secure JWT payload based on signatures and encryption schema.

2.7 Micro-service Architecture

Service Oriented Architecture (SOA) took the industry in the mid 2000s and many companies adopted this architecture to make their business functions way better than old monolithic systems. But due to the transition and design complexity problems most of the companies refused SOA. Currently microservices architecture is taking the IT systems with the ready-to-go style system component designs available as modular and high scalable sub systems. Microservices architecture promised to deal with some of the problems from large and complex SOA or monolithic business systems. Both SOA and microservices can not be considered as evolutionary patterns on each other but they share some of common characteristics but both can be defined as service based architecture pattern because both are heavily depending on services. They can also be defined as distributed architectures, because the service components are distributed on many locations. Wide variety of transfer protocols needs to be used to communicate with each other service components but the isolated nature guaranteed the loose-coupling of the business logic.

2.7.1 Design Goals

Components withing the mocrosesrvice architecture considered to be more self-contained, loosely coupled. Those characteristics mainly enabled better control over maintenance and provide more responsive nature to each service components.

2.7.2 Modular Systems

Modularity is the practice of encapsulating portions of service components in order to reduce dependency of each other components. This architecture is also encourages to maintain small components over large business components to enable the maintenance and performance goals.

2.7.3 Availability

Service availability and responsiveness are two main goals of adopting microservice architecture which are directly affects with user experience. Service responsiveness is the ability to receive timely responses from service components and the service availability refers to the ability of accepting requests in timely manner. Systems with microservices architecture which unable to provide above availability scenarios can implement busy signal patterns to maintain responsiveness.

2.8 Related Works

EMVCo.[2] has published tokenization framework Specification in 2014 and as extension to they have also publicly release EMV payment tokenization specification v2.0[8]. Mainly later document addressess the usage of payment token use cases in e-commerce. Apart from that mainly Visa and MasterCard contributed to HCE related tokenization deployments with their card specifications.

Thales e-Security Solutions implemented HCE Mobile Payments solution called Thales e-Security HCE based on their Hardware Security Module (HSM)s, and they have used HSM to store payment credentials generated by solution provider. Thalas completely depends on their HSM solutions to handle all data and transaction but proposed solution introduce simple token based approach to build eco system. To ensure end to end security, HSM can be optionally introduced to the proposed solution and client mobile applications totally identified by the generated token value.

Royal Bank of Canada (RBC) is one of the largest card issuer with 7 million issued cards. Early January 2014, they launched its Secure Cloud Platform with NFC sim card on mobile application enabling user to install payment applet in mobile and storing payment credentials in secure cloud solution. Secure cloud send commands to SIM payment application to do transactions. In September 2015, RBC added support for HCE to get rid of SIM card application due to the limited support of MNOs and mobile device models. Then mobile HCE users are allowed to use virtual credit cards with mobile devices that supported HCE. Operationally RBC saved mobile wallet payment credentials in their secure cloud and only cryptograms are saved on mobile HCE application. Commonwealth bank of Australia(CBA)[12], TD bank of canada, Getin Bank in Poland, IGN bank in Netherland accept HCE transactions and they also integrated HCE into their mobile wallets.

Grimiti cloud HCE is also provided similar HCE transaction eco system, but they store card information on mobile device and NFC enabled card terminal used mobile data to continue with transactions.

Above describe most of the systems seems to use HCE mobile application storage to save transaction cryptograms but instead of saving card details for each transaction on mobile devices, proposed system used identification token to continue with transaction. Once the mobile handset identification is done, established secure channel between terminal and TSP via mobile device can be used to send APDU commands. So android mobile device related data security issues can be easily mitigated.

[2] https://www.emvco.com/

Chapter 3

Design and Implementation of the Solution

Design of the system is basically considered on security, efficiency and flexibility concerns. To design end-to-end security system inter industry standards are adopted and equally considered about application security, transport security and storage security all over the system components.

3.1 Overview

This solutions consists of mobile and web components to demonstrate fully secured mobile HCE transactions with cloud implementation of Desfire EV1 SELECT, AUTHENTICATE, CREDIT, DEBIT, COMMIT, ROLLBACK and BALANCE functions. To compare with mobile device based SE implementation transaction norms and standards, we also have implemented mobile based SE too. The main solution components consists of:

- Mobile HCE Application
- Mobile Terminal Application
- Authorization Server
- Cloud Server

Java 1.8 is used as main programming language for Mobile and web applications. Android version 7.0(N2G48H)[1] is used to develop all mobile applications and Spring Boot 1.5.10 GA[2] is used to develop Authorization and cloud server and resulting web artifacts are deployed to Apache Tomcat 8 servlat container. MySQL is chosen as the server database and SQLite is used to persist data on mobile applications. Both web applications are hosted on cloud server hosted by Linode server[3]. Table 3.1 shows Mobile

[1] https://www.android.com/versions/nougat-7-0/
[2] https://docs.spring.io/spring-boot/docs/1.5.10.RELEASE/reference/htmlsingle/
[3] https://www.linode.com/

Terminal and HCE application specification details and Table 3.2 further describe about Authorization and Cloud Server specification.

Table 3.1
Mobile Terminal and HCE application specification.

Application	Android Version	Device Type	Device Vendor	Device RAM
Terminal	7.1.2 Nogut	Google Nexus 5	LG	2 GB
Mobile HCE	7.1.2 Nogut	Nokia 5	HMD Global	3 GB

Cloud hosting server has following configurations:
Vendor : Linode
OS : Ubuntu Linux 16.04 LTS
CPU : 2 Cores
Ram : 4 GB
Storage : 48 GB SSD

Table 3.2
Server Specification.

Server Type	Application Server	Web Framework	Language	Database Server
Authorization	Tomcat 8	Spring Boot	Java 1.8	MySQL 5.7.21
Cloud	Tomcat 8	Spring Boot	Java 1.8	MySQL 5.7.21

Above all hardware specifications consists of commodity hardware versions and cost efficiency and affordability of the solution is also considered. By using high level hardware specifications, existing transaction time and user experience can be improved significantly.

3.2 Solution components

Basic design functions and significance of each component in the proposed solution are discussed under this section. Operational background and security is the main concerns discussed here.

3.2.1 Mobile Terminal

Mobile terminal is developed to simulate PCD and it support DesfireEV1 card operations as listed in Table 3.3. According to Fig. 3.1 mobile terminal only interacts with HCE mobile application and once the card identification is done by HCE controllers, mobile terminal initiate Application Protocol Data Unit (APDU) flow by sending SELECT

command. Then HCE mobile application sends APDU response with data if available. Then according to the funcations mobile terminal continue with rest of the APDU until response APDU from mobile HCE application indicate error status.

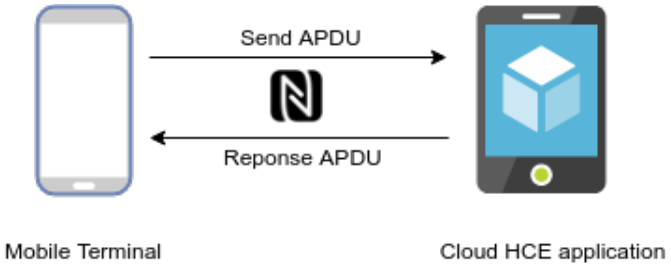

Fig. 3.1. Mobile Terminal.

Moreover all the application status and generated session keys are securely saved in SQLite database and after each and every session they are revoked immediately. Normally for production systems this mobile terminal simulation can be replaced with real terminals those are equipped with well organized secure and efficient functionalities.

3.2.2 HCE Mobile Application

HCE mobile application is the main components of this solutions and it is designed to fulfill two operational modes.

1 - Emulate SE on mobile environment

SE emulation on mobile device is done for the comparisons with cloud based HCE implementations. Cloud implementation and mobile SE implementations shares same module structure on SE emulations and shares same android based HCE configurations.

2 - Connects with Cloud SE implementation

When mobile HCE application connects with cloud SE implementation, mobile HCE logic simply tunnel all APDU commands sent by terminal application to cloud server. From mobile to cloud server, APDU commands are transported with rest API services along with transaction specific token value.

According to the Fig. 3.2, mobile HCE application interacts with all solution components. For every request received from terminal application mobile HCE application has to deal with security integrity and confidentiality of messages. The steps involves in HCE application interactions with each other solution components can be described as follows:

1 - Mobile HCE application request a token from Authorization service by providing client credentials with authentication API request.

2 - After a successful authentication, authorization server grants an access token in JWT format to be used with successive API calls

3 - Mobile terminal sends an APDU to mobile NFC controller.

4 - Mobile HCE application routes the received APDU command through cloud API call

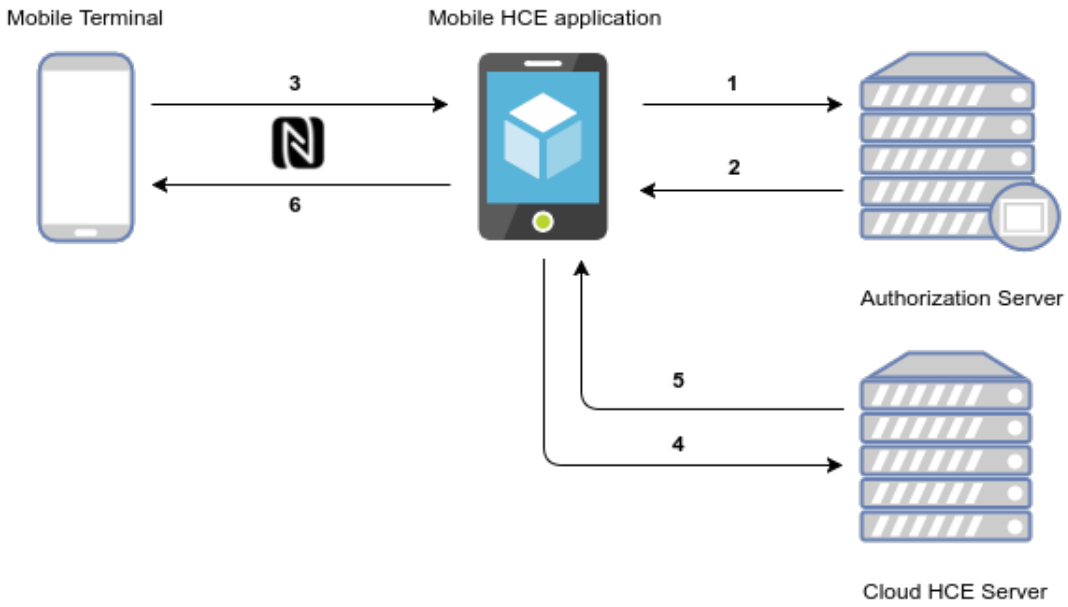

Fig. 3.2. Mobile HCE Application interactions.

secured with SSL along with token received in step 2.
5 - Cloud API server validate the token and then execute APDU command in request payload against cloud SE implementation. Then the result is sent back to the mobile client application.
6 - Mobile client application sends the received APDU response to mobile terminal application.

As same as mobile terminal application, all generated sensitive data within mobile HCE application like session keys and application status values are saved in SQLite database and after each sessions those saved credentials and sensitive data are revoked immediately. Moreover this HCE mobile implementation can be placed on device TEE if application provider is granted access to it. Received access tokens from Authorization server are also kept securely and token revocation can be done according to the usage count and expiration of hte tokens.

3.2.3 Cloud Services

Cloud service components are designed to use REST API in order to communicate with each other components. Security and access rights to each service component is defined under the granted privilege level on access tokens. Following section describe about the basic functionality of each cloud service elements:

Authorization Server

- Provide authentication mechanism and enable authorization for systems resources.
- Create custom tokens with tokenized PAN.

- Handle token revocation mechanism.

Cloud API Server

- Verify API service requests from outside entities for Bearer token.
- Implement cloud SE as a software services.
- Implementation of API endpoint to send and receive APDU.

Modularity, scalability and security are the main design outlines considered for cloud services. Identity delegation related principles are used to manage identity of mobile HCE client with other resource servers while card provisioning and accepting phases.

3.3 APDU Commands

Application Protocol Data Unit(APDU) is defined under ISO 7816-4 specification and used to communicate between smart cards. Usually APDU command is send to target device after successful service discovery and connection establish phases, as hexadecimal byte array. As shown in Fig. 3.3, APDU command consists of multiple standard data components. APDU command can exists without payload data but CLA, INS, P1, P2 fields considered mandatory for ISO 7816-4 implementations. But some vendors like NXP, developed shorter APDU command versions called Native APDU commands to achieve efficient transaction time. Native APDU commands can not integrate with current HCE implementations and ISO 7816-4 commands can be used[13].

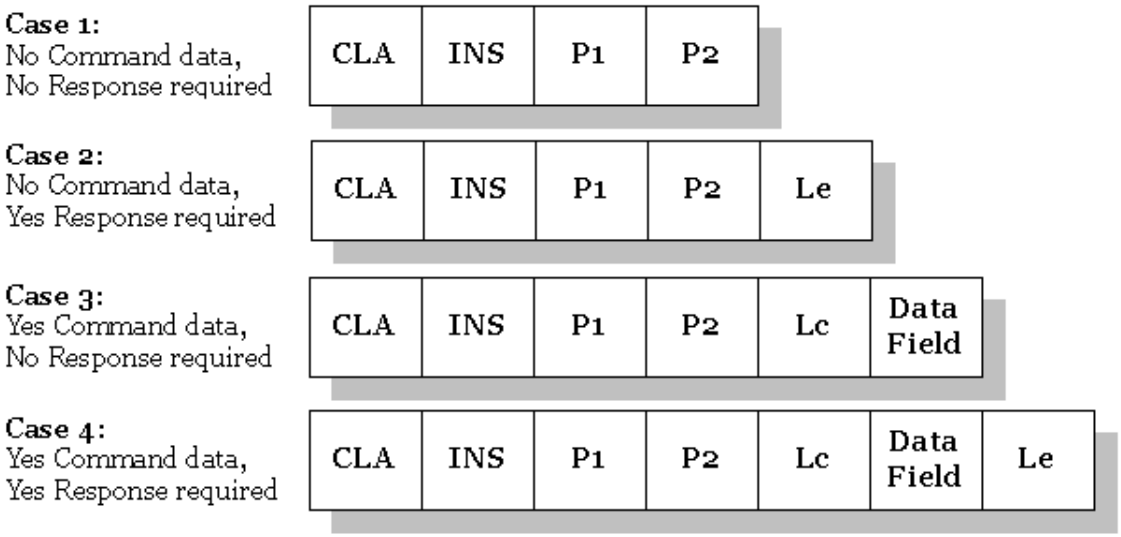

Fig. 3.3. ISO 7816-4 APDU Structure.[14]

Table 3.3 define all APDU commands implemented in both mobile HCE and cloud SE applications. Apart from SELECT APDU, all other commands are secured using AES 128 session encryption. Integrity of APDU commands is handled by using CMAC implementation as described in NIST special publication 800-38B.[15]

Table 3.3
Supported APDU Operations.

Operation	INS	Data	Data Size(bytes)
SELECT	5A	Application Id	3
AUTHENTICATE AES	AA	Key Number	1
CREDIT	0C	Reversed Credit Value	4 - 12
DEBIT	DC	Reversed Debit Value	4 - 12
GET VALUE	6C	File Number	1
COMMIT	C7	-	-
ABORT	A7	-	-

3.3.1 Authentication

Authentication procedure used in this web and mobile components directly connected with the identity delegation mechanisms. Token received after authentication has access key to card details and server components. According to the system design two authentication schemes can be defined as follows:

1 - OAuth and JWT token based user authentications
Authorization server requires delegator user name and password, client user name and password with unique identifier to device for authentication purposes. If the provided inputs are matched with the saved credentials, authorization server issued a token with relevant claims according access level of the user. Generated token is formatted according to JWT standards and token can be used only for limited APDU sessions with cloud services.
2 - Session authentication between PCD and PICC components.
Establishing the session between PCD and PICC is done according to the Fig. 3.4. PICC can have multiple keys stored in memory after initial personalization process. This keys can be selected according to the operation type and security requirements when establishing session with PCD. PCD is always initialized the authentication session creation process and it contains following high level steps:
1 PCD sends authentication initialization command with key identification number which is going to use for authentications.
2 PICC generates a 8 byte long random number and send it to PCD after encrypt with agreed key according to step 1 which is available on PICC memory.
3 PCD received the encrypted random generated by PICC and recover the random number generated by PICC after deciphering it. Then PCD generates new random number and append with the received random number generated by PICC in step 2 after inverting

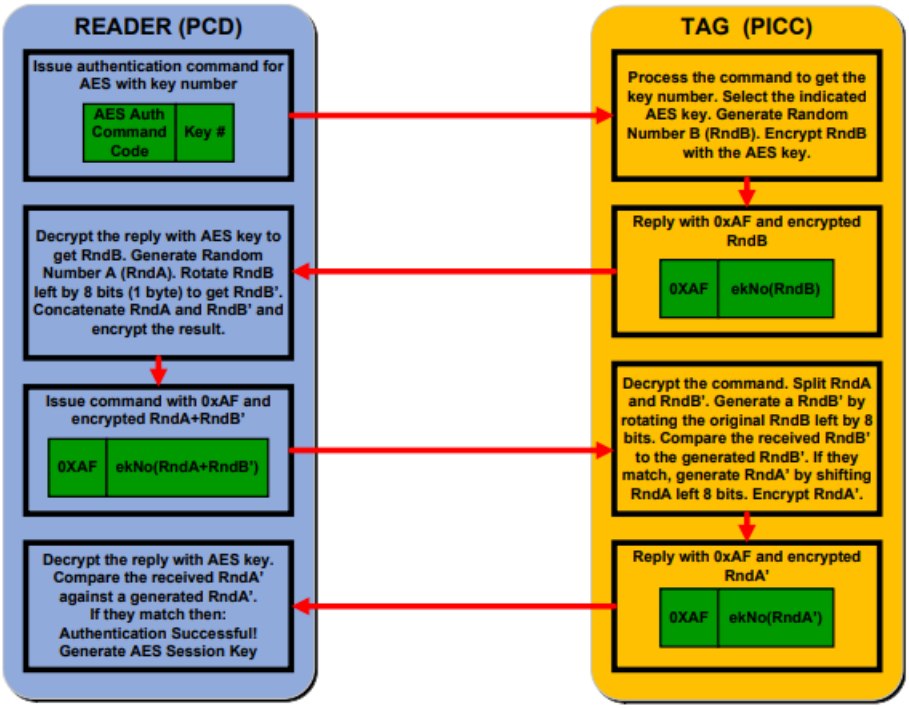

Fig. 3.4. AES authentication with Desfire EV1.[16]

it. Finally the encrypted result is send back to PICC.

4 PICC decrypts the command data and verify the inverted random number with the generated random number in step 2. Then PICC again shift PCD generated random number for 8 bits and the result is send back to the PCD.

5 Once the PCD recovers the random number received in step 4, both parties can verify that they have the same key. Then a new session key is formed on both PCD and PICC using already generated random numbers.

6 The generates session key is used to secure the APDU channel.

Session key generated on ADDU sessions are revoked immediately after the command execution of each session on PICC and PCD implementations.

3.3.2 Data confidentiality

According to the service component design, data confidentiality on HTTP transports can be managed by using SSL/TLS with good cipher suit. Token received after authentication with authorization server is considered as Bearer token and SSL/TSL implementation is required for to protect payment tokens.

Apart from the web components Advanced Encryption Standards(AES) algorithm is used with 16 byte length keys to protect PICC and PCD sessions. Unlike Triple DES algorithm widely used in DesfireEV1 cards, AES is not a Feistel cipher[15].

There are two different algorithms exists for the encryption and decryption of AES sessions. According to Fig. 3.5 AES encryption initiated with Initialization Verctor(IV)

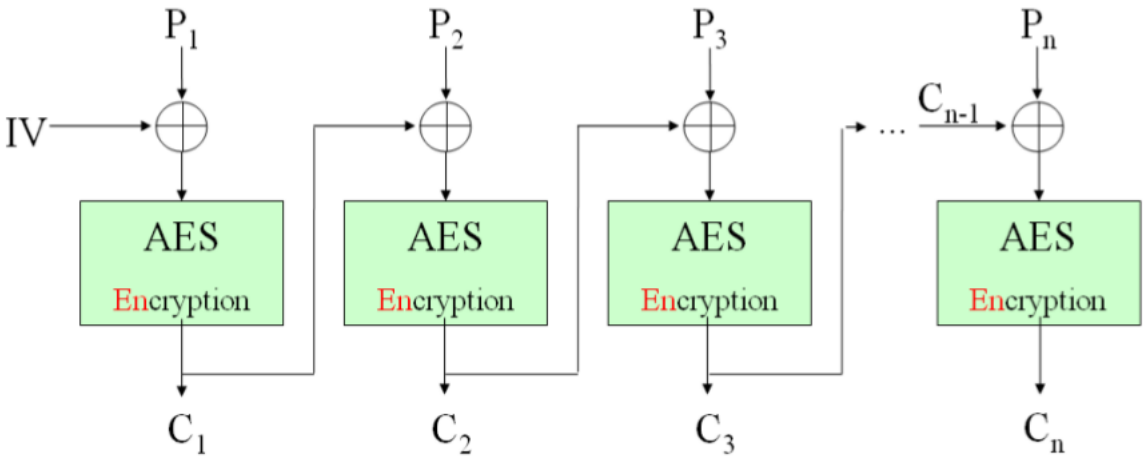

Fig. 3.5. AES enciphering.[17]

of 16 byte all zero values and data block of size 16 bytes, XOR together and result is send into AES algorithm as inputs. So the cipher receive after the operation can be derived as:

$$C = E(P \bigoplus IV)$$

For i number of input blocks, the result would be

$$C_i = E(P_i \bigoplus IV_i)$$

After continuing with same operational cycles with IV values equals to cipher blocks generated in each iteration, the summation of resulting cipher text values can be considered as encrypted value.

$$EN = \sum C_i$$

AES decryption can be found on Figure Fig. 3.6 and it used 16 byte long cipher text blocks as input and plain text data block is generated as the result. AES cipher is used to perform cryptographic operations because it is considered as recommended algorithm apart from TripleDES for many payment solutions.

3.3.3 Data integrity with CMAC

Cyclic Redundancy Check(CRC) algorithms are used to maintain integrity of the data on DesfireEV1 smart cards. For communication with MAC codes, 8 byte CMAC is used to ensure the data integrity and proposed systems used CMAC generation algorithm (See Fig. 3.7) is defined in NIST special Publication 800-38B. This specification defined recommendations for block cipher modes of operations and proposed algorithm provide

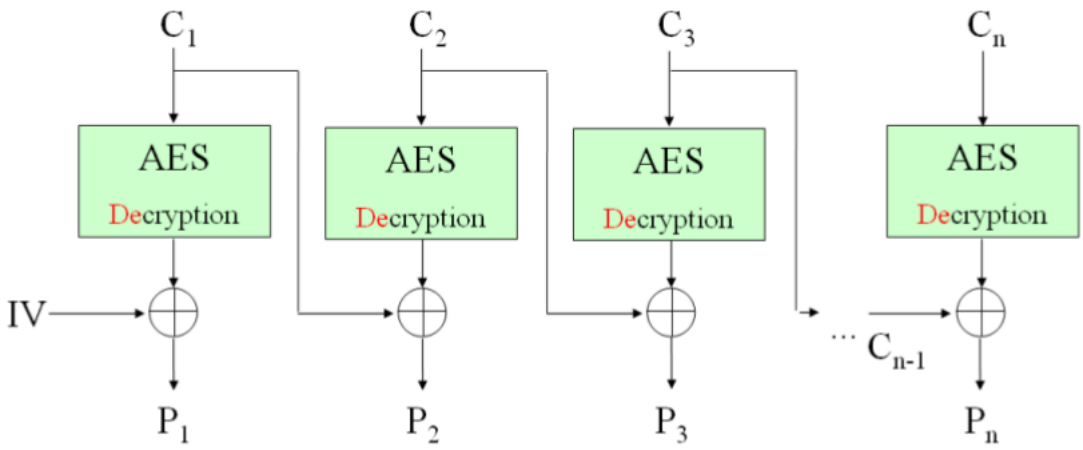

Fig. 3.6. AES deciphering.[17]

assurance of authenticity and ultimately integrity of binary APDU data. Both PCD and PICC components are implemented to use CMAC for ensure integrity of APDU commands.

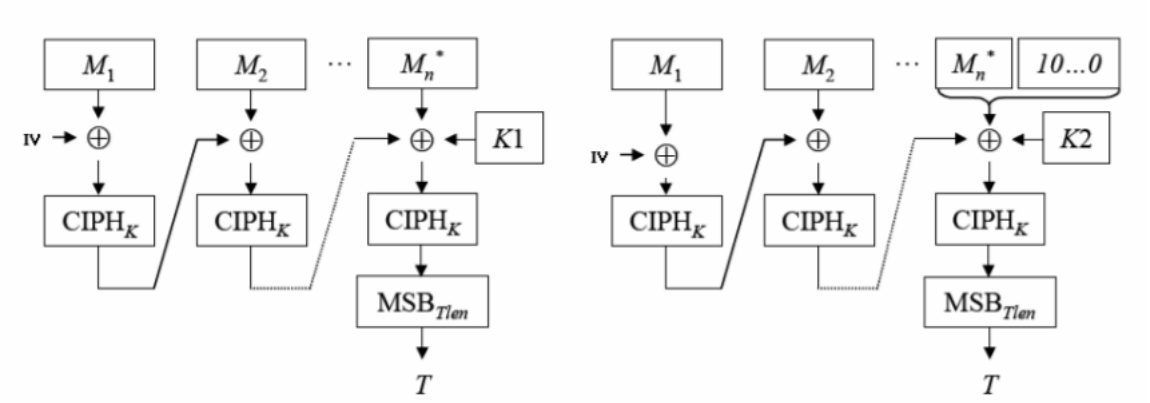

Fig. 3.7. Data integrity with CMAC.[15]

3.4 Authorization server

According to the design authorization server initiate the security flow of the application components. Client can directly connect with authorization server to get Bearer token and further this token can be used for subsequent logins. Authorization server design used Resource Owner Password Credential Grant type as the token issuing work-flow. When using this grant type resource owner(in proposed system cloud SE server) must trust the client application (HCE mobile client) and resource owner password should be shared with client application. But authorization server can manage client authentication separately with another set of credentials and once client succeeded with all inputs, token is issued. Fig. 3.8 describe about the token granting flow as follows:

1 - Resource owner shares the credentials with client.

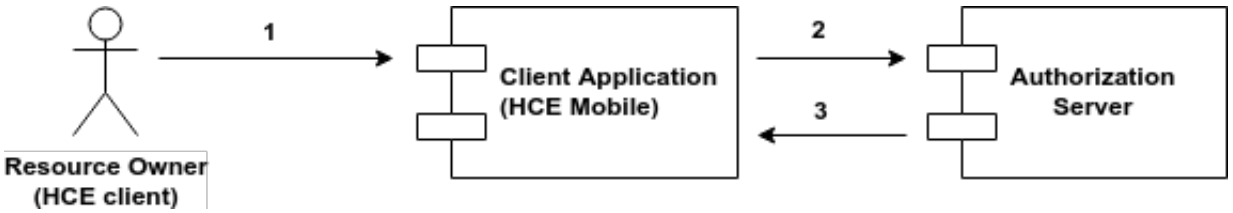

Fig. 3.8. OAuth 2.0 flow with Authorization Server.

2 - Client initiate access token request with authorization server.
3 - Once client is authenticated, new token value is issued to the client.

3.5 Cloud SE server

Cloud HCE server is the host for virtual SE implementation. With the appropriate access levels granted by authorization server only mobile HCE client can be connected with API services provided by cloud SE server. Mobile HCE application is only used to verify and authorize end user and without processing the APDU received from PCD, it just tunnel them to the cloud SE server for the processing. The main functionalities required from cloud SE server are:

1 - Implementation of the cloud SE.
Virtual SE related functions previously implemented on mobile HCE application should be transferred to cloud SE server. This will prevent attacker for directly access SE structure implemented on mobile device. In compared to the physical SE, this approach enables to manage the SE implementation logic without affect to the end consumers in very less time. Finally the handling cost of physical SE can be totally eliminated with proposed cloud implementation.

2 - API interface opened for outside services to interact with virtual SE through APDU commands.
API interface needs to be implemented to receive apdu values from PCD and it should take incoming APDU as parameter.

3 - Verification of RSA512 token signature before each API request.
User should be allowed to access API services hosted on cloud SE service with valid JWT token in request header. The signature of the JWT token can be verified by the public key component on cloud SE server.

4 - Manage the access to the virtual SE for end user.
Enabling or revoking access to the virtual SE can be done on cloud SE service.

5 - Collect and analyze audit logs.
Audit logs for every transactions can be collected and analyzed on cloud SE server. So security flows can be identified and users credentials and account access can be managed accordingly.

3.6 Summery

We have discussed about the high level design, expected functionalities and implementation details of each component on proposed system. Security, availability and scalability are the main design goals of this system.

Chapter 4

Results and Evaluation

4.1 Evaluation Criteria

We can evaluate implementation and operational details of the proposed system qualitatively by comparing with PCIDSS baseline requirements. Quantitative analysis can be performed by comparing transaction time of the cloud and mobile implementations. We can measure and compare the readings and deviation of records to measure the efficiency of the proposed system.

4.2 Results

We have listed transaction time in milliseconds for CREDIT, DEBIT and GET VALUE operations separately for HCE mobile and cloud implementations. Table 4.1 and Table 4.2 shows the received readings.

Table 4.1
Transaction Time for Mobile SE Implementation.

Operation	Reading 1 (ms)	Reading 2 (ms)	Reading 3 (ms)	Mean Value (ms)
CREDIT	376	372	369	372
DEBIT	457	395	381	411
GET VALUE	289	301	292	294

By checking the results of HCE mobile implementation we can have pretty good results in terms of transaction time but security wise underline approach should be improved. Readings taken with cloud SE implementation deviates with the standard transaction time range. Due to the network connection latency and strength, real time calculations on cloud based SE always can not be guaranteed and ultimately cloud implementation may not be able to reach 300ms - 500ms transaction time range as mobile SE ralated readings.

Table 4.2
Transaction Time for Cloud SE Implementation.

Operation	Reading 1 (ms)	Reading 2 (ms)	Reading 3 (ms)	Mean Value (ms)
CREDIT	830	806	790	808
DEBIT	871	831	858	853
GET VALUE	729	739	736	734

Apart from the transaction time based measurements, we can qualitatively measure our implementation against PCIDSS key requirements standards. Table 4.3, Table 4.4, Table 4.5, Table 4.6 shows the comparison results.

Table 4.3
PCIDSS key requirements on build and maintain secure network.

PCIDSS Requirement	Remarks	Result
Install and maintain a firewall configuration to protect cardholders data	This can be done on cloud services both authorization and cloud HCE server can be configured accordingly.	Can implement on existing system
Manage vendor-supplied defaults	This can be configured on the proposed system.	Can implement on existing system
Protect stored cardholder data	Mobile HCE application only received tokenized PAN and so cardholder data can be safely handled only on cloud servers.	This requirement is covered.
Encrypt transmission of cardholder data across open, public network	SSL/TLS connection can be used to secure transport layer security	Can implement on existing system

According to the PCIDSS key requirement analysis, our proposed application compatible with most of the requirements in design and majority of non existing requirement can be easily added as extensions without interrupting current system.

Table 4.4
PCIDSS key requirements on maintain vulnerability management programme.

PCIDSS Requirement	Remarks	Result
Use and regularly update anti-virus software	This can be done on cloud services.	Can implement on existing system
Develop and maintain secure systems and applications	We have implemented many security measures to protect secure payment details	Can be improved according to various methodologies.

Table 4.5
PCIDSS key requirements on implement storing access control measure.

PCIDSS Requirement	Remarks	Result
Restrict access to cardholder data by business need-to-know	Only tokenized pan is sent to mobile device and no other information shared with any components	This requirement is covered.
Assign a unique ID to each person with computer access	This can be done by handling our own hardware servers	With cloud environment we can not achieve this requirement
Restrict physical access to cardholder data	This can be done by handling our own hardware servers	With cloud environment we have to trust vendor

Table 4.6
PCIDSS key requirements on regularly monitoring and testing networks.

PCIDSS Requirement	Remarks	Result
Track and monitor all access to network resources and cardholder data	This feature can be implemented as centralized log management system	Can implement on existing system
Regularly test security systems and processes	Cloud services can be regularly monitored. Using audit logs all incidents can be identified and properly addressed	Can implement on proposed system
Maintain a policy that address information security	This can be done when process get matured	Can implement on existing system

4.3 Summery

In summery implementation of the proposed system shows fairly good readings with respect to the recommended transaction times values. But when it comes to qualitative measurements in compared to PCIDSS key requirements, some features and procedures needs to be implemented to maintain compliance with requirements.

Chapter 5

Future Work and Conclusions

5.1 Overview

Mobile payment is one of the emerging technology that affect many existing businesses. Specially NFC technology resolves many operational issues involved with payment industry and sometimes adoption of cutting edge technologies like HCE, can make existing business process more complex and risky by exposing existing systems to completely new attack vectors. The proposed system consists of:

- transfer mobile environment related security risks involving with HCE to cloud environment in order to handle/mitigate effectively.
- provide solutions for the limitations of embedded SE or TEE based payment solutions and improve flexibility.
- manage newly applicable attack vectors after transferring HCE mobile implementation to cloud based solution.
- solve scalability and availability issues of HCE mobile implementation.
- provide cost effective solutions for the operational issues like physical card management and hardware cost reduction.

5.2 Future Work

After comparing with PCIDSS key guideline we realize following improvements can be done in order to make the system more secure and organize:

- Private cloud can be maintained without utilizing public cloud environments where the data confidentiality and security can not be guaranteed.
- We can implement policy framework to manage the system.

- Access management policies can be converted to more granular levels to ensure high security standards.

Overall we thought by implementing following features can make the proposed system more efficient and secure

- Connection establishing time between mobile and cloud server can be improved.
- Databases can be installed on separate host to make sure high availability and scalability.
- OAuth token received on authentication phase, can equipped with more payment related data and it can be saved in mobile TEE.

5.3 Conclusions

We have proposed a system to operationally secure payment detail when using mobile based HCE deployments. Most of the times vendors tend to use embedded SE or TEE to save sensitive payment details. Due to the security risks involving with Android operating system, we proposed new systems which used almost zero sensitive payment information shared with mobile apart from tokenized PAN of the client. We have measured system performance qualitatively and quantitatively by using established payment card industry norms.

5.3.1 The problem

The problem domain of this thesis is to provide secure, operationally efficient and cost effective alternative for mobile based HCE implementation.

5.3.2 Solution

We have implemented cloud based SE to replace the SE implementation on mobile in order to mitigate mobile based security risks. Even though we move SE implementation from mobile, the basic APDU tunneling is done through mobile. To manage cloud services we have implemented two services used for authorization and cloud SE management operations.

5.3.3 Applications and benifits

The proposed system architecture can be applicable to cloud based mobile payment systems. The described solution components can be improved according to the specific requirements and global platform compliance SE implementations which compatible with ISO/IEC 7816-4 can be implemented on cloud SE server. And also this solution can be

used with any HCE enabled mobile applications without considering the availability of embedded SE or TEE. The proposed system has following benefits:

- minimal use of payment specific confidential data on mobile device. Mobile is only used as identification and command routing device between card terminal and cloud SE server. This will prevent attacker who gain access to mobile device from gathering payment details.
- This system allows remotely update SE implementation on cloud and related security measures without updating customer mobile application or physical smart card(if applicable).
- solve common scalability and availability issues involved with HCE mobile implementations.

5.3.4 Final words

At the end, this thesis attempted to introduce set of approaches to mitigate android HCE payment related risks on the mobile applications by moving SE implementation to cloud environment. Even though the transaction time comparison shows considerable performance improvements with additional network overhead but the evaluation section emphasizes the additional attack vectors applicable after the system migration from mobile to cloud implementation.

Bibliography

[1] *The Mobile Economy*, GSMA, 01 2015, sub-Saharan Africa 2015.

[2] P. Ndichu. (2015) Smarter Utilities for the Developing World. [Online]. Available: https://www.gsma.com/mobilefordevelopment/programme/m4dutilities/smarter-utilities-for-the-developing-world/

[3] Google. (2018) Host-based Card Emulation. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/hce.html

[4] *MIFARE DESFire EV1 contactless multi-application IC*, NXP, 12 2015, version 3.2.

[5] *GlobalPlatform Card Specification 2.2.0.7*, GlobalPlatform, 3 2006, version 2.2.

[6] Marketwired. (2016) SD Association Shows microSD Value for Mobile Payment and Network Security. [Online]. Available: http://www.marketwired.com/press-release/sd-association-shows-microsd-value-for-mobile-payment-and-network-security-2098613.htm

[7] Dafu Lv. (2014) ARM TrustZone Powered Mobile Security. [Online]. Available: http://www.oracle.com/technetwork/java/javacard/javacard1-139251.html

[8] *EMV Payment Tokenisation Specification*, EMVCo, LLC, 09 2017, version 2.0.

[9] RightScale. (2016) Cloud Computing Trends:2016 State of the Cloud Survey. [Online]. Available: http://dafulv.blogspot.com/

[10] Drillster. (2018) Authorization through OAuth 2.0. [Online]. Available: https://www.drillster.com/info/developers/api/oauth

[11] J. T. BLOG. (2017) SECURING YOUR CLOUD-NATIVE MICROSERVICE ARCHITECTURE IN SPRING. [Online]. Available: https://ordina-jworks.github.io/microservices/2017/09/26/Secure-your-architecture-part1.html

[12] C. B. of Australia. (2015) Commonwealth Bank of Australia selects GD HCE payment tech. [Online]. Available: https://www.finextra.com/news/announcement.aspx?pressreleaseid=59139

[13] Google. (2018) Host-based card emulation overview. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/hce

[14] Enrique Ortiz. (2003) An Introduction to Java Card Technology - Part 1. [Online]. Available: http://www.oracle.com/technetwork/java/javacard/javacard1-139251.html

[15] *Recommendation for Block Cipher Modes of Operation:The CMAC Mode for Authentication 800-38B*, NIST, 06 2016, nIST Special Publication 800-38B.

[16] *MIFARE DESFire EV1 AES Authentication With TRF7970A*, Texas Instruments, 12 2014, version 1.0.

[17] *MIFARE DESFire EV1 - Features and Hints*, NXP, 04 2009, rev 03.02 1.0.