



Faculteit Bedrijf en Organisatie

Verskillende veiligheidsmethodes voor Android HCE (Host-based Card Emulation)

Thibaut Maddelein

Scriptie voorgedragen tot het bekomen van de graad van  
professionele bachelor in de toegepaste informatica

Promotor:  
Leen Vuyge  
Co-promotor:  
Francesco Verheyne

Instelling: CCV Lab

Academiejaar: 2018-2019

Tweede examenperiode



Faculteit Bedrijf en Organisatie

Verskillende veiligheidsmethodes voor Android HCE (Host-based Card Emulation)

Thibaut Maddelein

Scriptie voorgedragen tot het bekomen van de graad van  
professionele bachelor in de toegepaste informatica

Promotor:  
Leen Vuyge  
Co-promotor:  
Francesco Verheye

Instelling: CCV Lab

Academiejaar: 2018-2019

Tweede examenperiode



## Woord vooraf

Het idee om mijn bachelorproef te schrijven over de verschillende beveiligingsmethodes voor een Android host-based card emulation applicatie was een idee van mijn co-promotor Francesco Verheyen. Toen hij met dit voorstel afkwam was ik direct geïntrigeerd om meer te weten te komen over dit onderwerp. Betalingsapplicaties is iets waarmee we bijna dagelijks in contact komen maar niet veel over weten hoe het eigenlijk allemaal in zijn werkt gaat. Ik heb altijd gehoord dat de wereld van Android heel veel mogelijkheden biedt voor de gebruikers en de ontwikkelaars, maar ben er nooit veel mee bezig geweest. Het schrijven van deze bachelorproef was dus de ideale gelegenheid om meer te weten te komen over betalingsapplicaties, beveiliging van data en de wondere wereld van Android. Ik ben ook altijd al gefascineerd geweest door hoe dingen in elkaar zitten en in zijn werk gaan, dus het uitzoeken hoe betalingsapplicaties en de beveiliging van data in elkaar zit was een leuk en leerrijk proces.

Tijdens het schrijven van deze bachelorproef heb ik de kracht van Android ontdekt en heeft me nieuwsgierig gemaakt om hiermee verder te experimenteren en nog meer nieuwe dingen te ontdekken. Door de vrijheid en de kracht van Android was het mogelijk om de host-based card emulation technologie te kunnen ontwikkelen. Door deze nieuwe technologie is het mogelijk om NFC-technologie te gebruiken zonder de nood te hebben aan een Secure Element die beheerd wordt door derden. De loskoppeling van het Secure Element die zorgde voor een hoog niveau van veiligheid betekent dat er een andere manier nodig is om de gegevens te kunnen beveiligen, vandaar ook de nood aan deze bachelor om de verschillende mogelijkheden te onderzoeken.

Ik zou graag nog wat mensen willen bedanken die mij geholpen hebben bij het schrijven van mijn bachelorproef. Eerst en vooral wil ik mijn ouders bedanken voor hun steun doorheen mijn studies, ik zou ook graag mijn vriendin willen bedanken voor alle steun

tijdens mijn studies en voor de hulp bij mijn bachelorproef. Natuurlijk zou ik ook graag mijn promotor Leen Vuyge en mijn co-promotor Francesco Verheyen willen bedanken voor alle feedback en raad die ze mij gegeven tijdens het schrijven van mijn bachelorproef.

Als laatste wil ik ook u, de lezer bedanken voor de interesse in mijn bachelorproef. Ik heb veel bijgeleerd uit het schrijven van deze bachelorproef en ik hoop van harte dat u ook wat kunt bijleren uit dit onderzoek.

## Samenvatting

Door gebruik te maken van Host-based Card Emulation is de nood voor een Secure Element verdwenen voor het simuleren van smartcards. Het Secure Element zorgde ervoor dat de applicaties en de gevoelige data die er werden opgeslagen goed beveiligd waren. Dit niveau van beveiliging wordt bekomen doordat het Secure Element zowel hardware- als softwarematig beveiligd is. Een Secure Element wordt geproduceerd door derden en niet door de producent van het mobiele toestel, wanneer een ontwikkelaar gebruik wil maken van dat Secure Element zal er toestemming moeten verkregen worden van de producent. Wanneer een ontwikkelaar toestemming wil hebben om gebruik te maken van het Secure Element zal er een commerciële overeenkomst moeten getekend worden met de producent en is de ontwikkelaar dus afhankelijk van deze overeenkomst. Omdat men niet meer afhankelijk zou zijn van een derde partij en omdat niet ieder toestel beschikt over een Secure Element was er nood aan een andere oplossing namelijk host-based card emulation. Nu het simuleren van smartcards kan gebeuren zonder een Secure Element worden de applicaties en de gevoelige data opgeslagen op het mobiele toestel. Aangezien alles op het mobiele toestel opgeslagen wordt zijn de applicaties en de gevoelige gegevens vatbaar voor aanvallen vanwege de veiligheidsproblemen die het besturingssysteem met zich meebrengt. Om te voorkomen dat er kan geknoeid worden met de data of met de applicatie is er nood aan andere manieren om de applicatie en de data te beveiligen. Wat de verschillende manieren zijn om een HCE applicatie te beveiligen zal onderzocht worden in deze bachelorproef.

Er zal gekeken worden naar welke mogelijkheden er allemaal beschikbaar zijn om een HCE applicatie te beveiligen en welke de verschillende bedreigingen zijn voor zo'n applicatie. Er zal ook een experiment uitgevoerd worden waarbij drie verschillende beveiligingsmethodes uitgewerkt zullen worden tot een proof-of-concept namelijk biometric factors, encryption en tokenization. Hiermee wordt aangetoond hoe de verschillende methodes geïmplementeerd kunnen worden. Op basis van het experiment zullen de verschillende

onderzoeksvragen beantwoord worden om een duidelijk beeld te geven van de verschillen of gelijkenissen tussen de gekozen beveiligingsmethodes. Er wordt gekeken naar de moeilijkheid om een methode te implementeren, zichtbaarheid van de data, veiligheid van de methode, mogelijkheid tot combinatie en de voordelen van het combineren van verschillende methodes. Aan het begin van dit onderzoek werd er verwacht dat het gebruiken van biometrische factoren de veiligste oplossing zou zijn ook op vlak van user-experience.

Uit het onderzoek is gebleken dat dit inderdaad de veiligste manier is om een applicatie te beveiligen maar enkel de toegang tot de applicatie en niet de beveiliging van de data. Uit het onderzoek kan er ook geconcludeerd worden dat er niet één juiste of beste oplossing is, het kiezen van de beste methode hangt sterk af van de noden van de applicatie en kan dus niet veralgemeend worden. Bij het maken van de keuze van beveiligingsmethode moet er niet enkel gekeken worden naar het niveau van veiligheid die de methode aanbied maar ook naar de kosten die het met zich meebrengt.



# Inhoudsopgave

<b>1</b>	<b>Inleiding .....</b>	<b>13</b>
1.1	Probleemstelling	13
1.2	Onderzoeksvraag	14
1.3	Onderzoeksdoelstelling	14
1.4	Opzet van deze bachelorproef	14
<b>2</b>	<b>Stand van zaken .....</b>	<b>17</b>
2.1	Host-based Card Emulation (HCE)	17
2.2	Near Field Communication (NFC)	18
2.3	Secure Element (SE)	19
2.4	Bedreigingen voor een contactloze betalingsapplicatie	19
2.4.1	Software Reverse Engineering .....	20
2.4.2	Secure Data Storage .....	20

2.4.3	Malware .....	20
2.4.4	Gebruik van een nep Point-of-Sale (POS) .....	21
2.4.5	Brute force aanval .....	22
<b>2.5</b>	<b>Hardware beveiliging vs risicobeperking</b>	<b>23</b>
<b>2.6</b>	<b>Beveiliging HCE</b>	<b>23</b>
<b>2.7</b>	<b>Beveiliging op hardware niveau</b>	<b>24</b>
2.7.1	Biometric Factors .....	24
2.7.2	Secure Element .....	24
<b>2.8</b>	<b>Beveiliging op software niveau</b>	<b>25</b>
2.8.1	White Box Cryptography .....	25
2.8.2	Tamper-Proofed Software .....	25
2.8.3	Device Identity Solutions .....	25
2.8.4	Encryption .....	26
2.8.5	Tokenization .....	26
<b>2.9</b>	<b>Beveiliging op software en hardware niveau</b>	<b>27</b>
2.9.1	Security Frameworks/Trusted Execution Environment .....	27
<b>3</b>	<b>Methodologie .....</b>	<b>29</b>
<b>3.1</b>	<b>Beveiligingsmethodes</b>	<b>29</b>
<b>3.2</b>	<b>Opbouw Host-based Card Emulation applicatie</b>	<b>30</b>
<b>3.3</b>	<b>Implementatie biometric factors</b>	<b>30</b>
<b>3.4</b>	<b>Implementatie encryption</b>	<b>31</b>
<b>3.5</b>	<b>Implementatie tokenization</b>	<b>31</b>
<b>3.6</b>	<b>Onderzoek</b>	<b>32</b>

<b>4</b>	<b>Resultaten</b>	<b>33</b>
<b>4.1</b>	<b>Moeilijkheidsgraad van de implementatie</b>	<b>33</b>
4.1.1	Biometric factors	33
4.1.2	Encryption	34
4.1.3	Tokenization	34
<b>4.2</b>	<b>Zichtbaarheid van de gevoelige gegevens</b>	<b>34</b>
4.2.1	Biometric factors	34
4.2.2	Encryption	35
4.2.3	Tokenization	35
<b>4.3</b>	<b>Veiligheid van de beveiligingsmethode</b>	<b>35</b>
4.3.1	Biometric factors	35
4.3.2	Encryption	35
4.3.3	Tokenization	36
<b>4.4</b>	<b>Combinatie van verschillende beveiligingsmethodes</b>	<b>36</b>
<b>4.5</b>	<b>Voordelen van het combineren van verschillende methodes</b>	<b>37</b>
<b>5</b>	<b>Conclusie</b>	<b>39</b>
<b>A</b>	<b>Onderzoeksvoorstel</b>	<b>41</b>
<b>A.1</b>	<b>Introductie</b>	<b>41</b>
<b>A.2</b>	<b>State-of-the-art</b>	<b>41</b>
<b>A.3</b>	<b>Methodologie</b>	<b>42</b>
<b>A.4</b>	<b>Verwachte resultaten</b>	<b>42</b>
<b>A.5</b>	<b>Verwachte conclusies</b>	<b>42</b>

<b>Bibliografie</b> .....	<b>43</b>
---------------------------	-----------

## Lijst van figuren

2.1	Simulatie van een smart card via SE .....	18
2.2	Flow van een mobiele betalingsapplicatie met tokenisatie .....	27
3.1	Service meta-data in AndroidManifest .....	30
4.1	Representatie van de nodige tijd om een geëncrypteerd wachtwoord te ontcijferen .....	36



# 1. Inleiding

## 1.1 Probleemstelling

Sinds de komst van Android 4.4 (Kit Kat) is het mogelijk om smartcards te simuleren via een nieuwe technologie Host-based card emulation (HCE). HCE zorgt ervoor dat men smartcards die gebruikt worden voor betaalkaarten of loyalty kaarten kunnen simuleren zonder het gebruik van een Secure Element die aanwezig moet zijn in het mobiele toestel. Vroeger kon dit enkel aan de hand van het gebruik van een Secure Element, dit zorgde ervoor dat applicaties en gevoelige data veilig opgeslagen konden worden op het mobiele toestel. Het Secure Element is zowel hardware- als softwarematig beveiligd wat het zeer veilig maakt om gevoelige data in op te slaan, het nadeel en dus ook de nood voor HCE is de moeilijke samenwerking met de producenten van de Secure Elements. Een ontwikkelaar kan niet zomaar gebruikmaken van een Secure Element om zijn gegevens of applicatie in op te slaan, hiervoor moet er toestemming gevraagd worden aan de producent. Wanneer er toestemming gevraagd wordt zal er een commerciële overeenkomst getekend moeten worden. Door het tekenen van zo een overeenkomst is de ontwikkelaar sterk afhankelijk van de producent en zorgt dit voor een sterk verlaagde vrijheid voor de ontwikkelaar. Er bestaan natuurlijk meerdere producenten van Secure Elements, hierdoor zal de ontwikkelaar met meerdere producent een overeenkomst moeten sluiten om een oplossing te kunnen aanbieden voor zoveel mogelijk toestellen. Niet ieder toestel beschikt over een Secure Element wat de nood aan HCE ook verhoogt.

Door de komst van HCE is er geen nood meer aan een Secure Element en is men niet meer afhankelijk van de producent maar hierdoor zijn de applicatie en de gevoelige data niet goed meer beveiligd. Hierdoor kreeg ik de vraag van mijn stagebedrijf CCV Lab om een onderzoek te doen naar de verschillende mogelijkheden om een HCE applicatie te beveiligen. CCV Lab is een bedrijf die betalingsapplicaties, loyalty applicaties en

kassasystemen ontwikkeld. Dit is dus een interessant onderzoek om een goede keuze te kunnen maken welke technologie ze kunnen gebruiken om hun applicaties te beveiligen. Dit geldt niet alleen voor het bedrijf CCV Lab maar ieder bedrijf of ontwikkelaar die bezig is met het ontwikkelen van een betalingsapplicatie aan de hand van HCE technologie. Door de steeds toenemende cyber aanvallen is er ook een grote nood aan beveiligingsmogelijkheden. Hierdoor zijn er tal van mogelijkheden beschikbaar om uw applicaties te beveiligen. Om te weten te komen welke beveiligingsmethode nu de beste is zullen er drie methodes uit gekozen worden waarvan een proof-of-concept uitgewerkt worden. Deze proof-of-concepts zullen dan afgetoetst worden op een aantal criteria om te bepalen wat nu de beste methodes is.

## 1.2 Onderzoeksvraag

In deze bachelorproef zal er onderzocht worden welke de verschillende mogelijkheden zijn om een android HCE applicatie te beveiligen en welken van deze mogelijkheden nu de beste is. Dit zal bepaald worden aan de hand van volgende criteria:

- Moeilijkheidsgraad van de implementatie: Hoe ingewikkeld is het om een bepaalde methode te implementeren in de applicatie. Zijn er extra resources nodig voor de implementatie?
- Zichtbaarheid van de gevoelige gegevens: Zijn de gevoelige gegevens zomaar leesbaar, vanaf wanneer of tot wanneer zijn de gegevens zichtbaar?
- Veiligheid van de beveiligingsmethode: Hoe veilig zijn de methodes effectief, kunnen ze gemakkelijk omzeild worden?
- Combinatie van verschillende beveiligingsmethodes: Kunnen verschillende methodes gecombineerd worden met elkaar, zo ja welke combinaties kunnen er gemaakt worden?
- Voordelen van het combineren van verschillende methodes: Wanneer verschillende methodes met elkaar gecombineerd worden heeft dit dan voordelen of ook nadelen?

## 1.3 Onderzoeksdoelstelling

Op het einde van deze bachelorproef zou er een duidelijk beeld moet zijn van wat de verschillende methodes zijn om een applicatie met host-based card emulation te beveiligen. Daarnaast moet het ook duidelijk zijn wat de voor- en nadelen zijn van de verschillende methodes en wat de beste oplossing is in een specifieke use-case.

## 1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:



In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4 worden de resultaten van het experiment besproken en antwoord gegeven op de onderzoeksvragen.

In Hoofdstuk 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

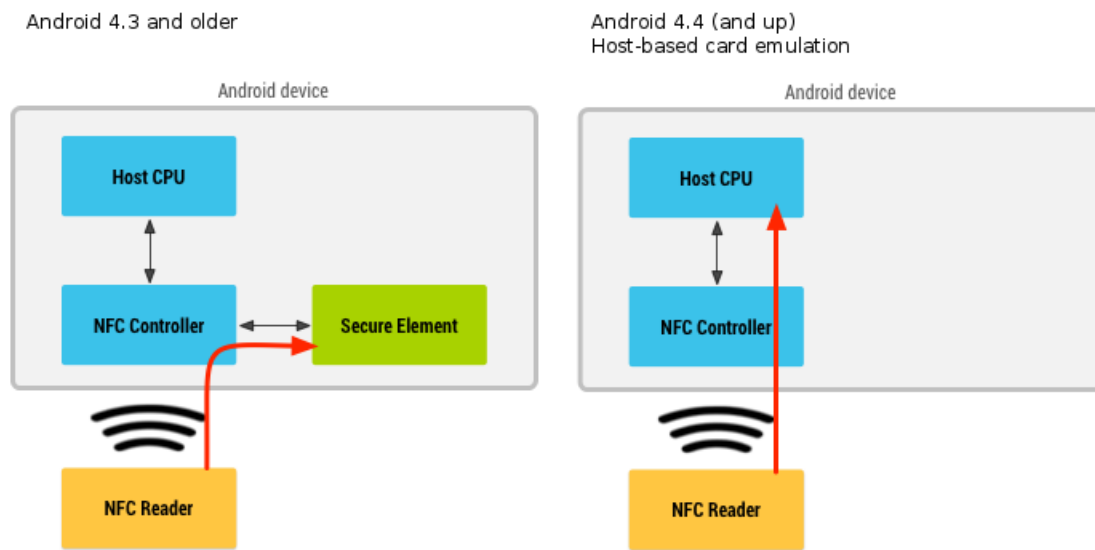


## 2. Stand van zaken

In dit hoofdstuk zal er een stand van zaken gegeven worden van het onderwerp, hoe het vroeger gebeurde en hoe het tegenwoordig gebeurt. Eerst en vooral zal sectie 2.1 wat meer uitleg geven over HCE en hoe het simuleren van draadloze smart cards gebeurt. In sectie 2.2 zal een korte uitleg gegeven worden over NFC technologie. Sectie 2.3 geeft een betere kijk op wat een Secure Element (SE) is. Sectie 2.4 toont een aantal bedreigingen voor contactloze betalingsapplicaties. In sectie 2.5 hoe beveiliging op hardware niveau werkt vs risicobeperking. Ten slotte zal in sectie 2.6 uitgelegd worden welke manieren er bestaan om HCE te beveiligen.

### 2.1 Host-based Card Emulation (HCE)

Sinds de komst van Android 4.4 oftewel Android KitKat introduceerde Google een nieuwe technologie genaamd Host-based Card Emulation (HCE). HCE technologie maakt het mogelijk om Near Field Communication (NFC) technologie te gebruiken zonder de aanwezigheid van een secure element. De nood voor deze technologie is er gekomen door de sterke afhankelijkheid aan de producent die bij het gebruik van een secure element erbij komt. Ook niet ieder mobiel toestel beschikt over een secure element hierdoor was er geen één oplossing voor alle toestellen mogelijk. HCE is beschikbaar vanaf Android 4.4, dit betekent dat meer dan 90% van de actieve android toestellen instaat zijn om gebruik te maken van host-based card emulation technologie. Wanneer het Android toestel zich in card emulation (CE) mode bevindt en tegen een draadloze lezer of point-of-sale (POS) terminal gehouden wordt, heeft het toestel de mogelijkheid om allerlei soorten draadloze smart cards te simuleren. Deze contactloze smart cards worden in vele situaties gebruikt zoals bij draadloos betalen, loyalty systemen, ticketing, toegang tot gebouwen ... (Alliance, 2014).



Figuur 2.1: Simulatie van een smart card via SE

HCE maakt het dus mogelijk voor NFC-apparaten om draadloze smart cards te simuleren. Om gebruik te kunnen maken van HCE heeft Android verschillende libraries en APIs (Application Programming Interface) geïmplementeerd in het besturingssysteem. Deze libraries en APIs worden overschreven door de applicaties die hier gebruik van willen maken en die op de CPU van het apparaat draaien. Deze applicaties kunnen dan APDU (Application Protocol Data Unit) commando's en antwoorden uitwisselen met een NFC POS. Wanneer men vroeger gebruik wilde maken van de NFC technologie kon dit alleen door een Secure Element (SE) die ingebouwd zat in het apparaat zoals een SIM-kaart. De applicaties werden geïnstalleerd op deze SE die dan de APDU's afhandelde om zo draadloze smart cards veilig te kunnen simuleren. De APDU's die verstuurd worden van een NFC-lezer worden opgevangen door de NFC-antenne van het apparaat en wordt doorgegeven via de NFC controller naar het SE en omgekeerd (zie figuur 2.1). Met HCE is het de bedoeling dat de nood van een SE verwijderd wordt uit deze operatie, in plaats van de APDU's door te geven naar het SE worden deze doorgegeven naar de CPU van het apparaat en omgekeerd (zie figuur 2.1) (Alattar, 2014).

## 2.2 Near Field Communication (NFC)

Near Field Communication (NFC) is een technologie die communicatie vanop korte afstanden, meestal nul tot vier centimeter maar kan tot 20 centimeter oplopen, mogelijk maakt. NFC apparaten kunnen actief of passief zijn, wanneer het NFC apparaat actief is dan gebruikt dit apparaat zijn eigen energiebron om zijn radio frequentie te genereren. Een passief NFC apparaat gebruikt de energiebron van een actief NFC apparaat om data te versturen, passieve NFC apparaten kunnen ook enkel maar antwoorden op aanvragen die vanaf een actief NFC apparaat verstuurd wordt. Transacties worden automatisch gestart

door twee NFC apparaten elkaar te laten raken of deze twee dicht bij elkaar te houden (Alattar, 2014).

NFC heeft drie verschillende operating modes: Peer to Peer mode, Reader/Writer mode en Contactless Card Emulation. Peer To Peer mode biedt de mogelijkheid om data te versturen tussen twee NFC apparaten aan snelheden tot 424 Kbit/s. Reader/Writer mode maakt het mogelijk dat twee NFC apparaten gebruikt kunnen worden voor het lezen/schrijven van tags en draadloze smart cards, hierbij is de snelheid van het versturen van de data maar 106 Kbit/s. Contactless Card Emulation laat de NFC apparaten draadloze smart cards of tags simuleren die gelezen of naar geschreven kunnen worden door een NFC lezer (Alattar, 2014).

## 2.3 Secure Element (SE)

Een Secure Element is een hardware element binnenin een Android apparaat. Het doel van het Secure Element is om applicaties en al hun data veilig op te slaan, het Secure Element is hardwarematig beveiligd dus zorgt het voor de ideale plaats om de applicaties in op te slaan. Er bestaan drie verschillende soorten SE's: Universal Integrated Circuit Card (UICC), ingebouwde SE (eSE) en een microSD (Lepojevic, Pavlovic & Radulovic, 2014). UICC is een meer geavanceerde vorm van een gewone SIM-kaart. ESE is een ingebouwde smart card die gesoldeerd is aan het moederbord van het apparaat. Ondanks dat een Secure Element één van de meest veilige manieren is om een applicatie en zijn data in op te slaan is het niet de gemakkelijkste. Niet ieder apparaat is in het bezit van een Secure Element. Dit zorgt er dus voor dat maar een klein aantal apparaten kunnen genieten van de voordelen van een Secure Element, iedere vorm van een Secure Element wordt ook beheerd door een andere producent wat het moeilijker maakt voor applicatieontwikkelaars om het SE te gebruiken. Door deze tekortkomingen kwam de nood voor Host-based Card Emulation technologie en de daarbij horende beveiliging voor HCE.

## 2.4 Bedreigingen voor een contactloze betalingsapplicatie

De mogelijke bedreigingen voor mobiele applicaties veranderen voortdurend. Dit wetende moeten de ontwikkelaars voor mobiele applicaties (in dit geval mobiele betalingsapplicaties) rekening houden met alle verschillende mogelijke bedreigingen en tegen maatregelen. Op de dag van vandaag is Host-based card emulation alleen maar beschikbaar op Android, er wordt verwacht dat Host-based card emulation overgebracht zal worden naar andere besturingssystemen zoals IOS en Windows. Het is dus belangrijk dat er in de toekomst security frameworks ontwikkeld worden die besturingssysteem onafhankelijk zijn, dit zorgt voor snelle en flexibele applicatie management (Cryptomathic, 2014).

### 2.4.1 Software Reverse Engineering

Software Reverse Engineering (SRE) is het analyseren van software systemen in zijn geheel of een deel ervan om zo het ontwerp of implementatie informatie te ontdekken. Een typisch scenario waar SRE vaak gebruikt wordt is bij software waaraan al jaren is gewerkt en informatie over het bedrijf bevat in de geschreven code. Hackers kunnen gebruik maken van de verschillende standaard zwaktes van het besturingssysteem of protocollen samen met reverse engineering tools, scripting en op maat gemaakte software om zo informatie te verzamelen om hun aanval uit te voeren. Software ontwikkelaars overwegen best om hun software zo robuust mogelijk te maken zodat er veel tijd en innovatie nodig is van de hackers om een aanval te doen slagen (Cryptomathic, 2014).

### 2.4.2 Secure Data Storage

Door gebruik te maken van Host-based card emulation en betalingstokens maakt het mogelijk voor de ontwikkelaars om onafhankelijk te werken van de grote bedrijven die de Secure Elements produceren. Dit zorgt er natuurlijk ook voor dat de gevoelige gegevens niet meer opgeslagen kunnen worden in een Secure Element wat het op zijn beurt weer gemakkelijker maakt voor hackers om deze data te verkrijgen. Hierdoor zal er een nieuwe oplossing moeten komen om de gevoelige data ergens veilig te kunnen opslaan waar de hackers niet gemakkelijk aan kunnen raken en ontmoedigd worden om een aanval te plegen (Cryptomathic, 2014).

Secure data storage refereert naar data in rust toestand die opgeslagen wordt op computers, servers, harde schijven, draagbare toestellen zoals externe harde schijven en USB-sticks, online/cloud opslag, network-based storage area network (SAN) of network attached storage (NAS) systeem. Het veilig opslaan van de data op deze systemen kan bekomen worden door data encryption, access control, beveiliging tegen malware, fysieke beveiliging van opslagplaatsen, implementatie van gelaagde opslag beveiliging architectuur.

### 2.4.3 Malware

Mobiele besturingssystemen hebben een heel ander beveiligings profiels in tegenstelling tot desktop besturingssystemen. Alle geïnstalleerde versies van een mobiel besturingssysteem moeten goedgekeurd worden door de producent van het besturingssysteem waardoor ze veiliger zijn. Ondanks de extra veiligheid wordt er vaak malware gevonden op mobiele systemen, de malware werkt altijd binnen de toegestane toestemmingen en kan dus geen al te grote schade aanrichten (Cryptomathic, 2014).

Malware is een algemene naam die gebruikt wordt voor een groot aantal kwaadaardige software varianten zoals virussen, ransomware en spyware. Malware is code dat ontwikkeld is door cyber aanvallers met al doel grote schade aan te richten aan data en systemen of toegang te krijgen tot een netwerk. Malware wordt meestal verspreid in de vorm van een link of bestand via email, de gebruiker moet klikken op de link of het bestand openen om de malware uit te voeren. Zoals eerder vernoemd zijn er verschillende soorten malware:

- **Virus:** Een virus is de meest voorkomende vorm van malware. De kwaadaardige code die de virus met zich meebrengt wordt verborgen in de gewone code. Het virus wordt geactiveerd door een onwetende gebruiker of een geautomatiseerd proces. Een virus kan zich heel snel en ver verspreiden, het kan schade veroorzaken aan de core functionaliteit van het systeem, bestanden onbruikbaar maken en gebruikers uit hun eigen computer sluiten.
- **Worms:** De worm heeft zijn naam gekregen door de manier waar de malware het systeem infecteert. De worm start op een machine en verspreidt zich over het netwerk om andere systemen te infecteren. Dit type malware kan snel een groot deel of zelfs alle systemen binnen een netwerk infecteren.
- **Spyware:** Spyware is ontwikkeld om te spioneren wat een gebruiker doet. De malware draait in de achtergrond van de computer en verzamelt informatie over de gebruiker zonder dat hij er weet van heeft. De informatie die spyware zoal kan verzamelen zijn bank gegevens, wachtwoorden en nog veel meer gevoelige informatie.
- **Trojans:** Trojans doen zich voor als legitieme software, de malware zal zich discreet gedragen en zal achterpoorten creëren voor andere malware zodat ze het systeem kunnen binnendringen.
- **Ransomware:** Ransomware kan een heel netwerk platleggen of een gebruiker uit zijn computer sluiten. De gebruiker moet een hoge dwangsom betalen om terug toegang te krijgen tot het netwerk of de computer. Ransomware wordt veel gebruikt om grote organisaties aan te vallen en grote dwangsommen te ontvangen.

#### 2.4.4 Gebruik van een nep Point-of-Sale (POS)

De point-of-sale heeft een belangrijke rol binnen het uitvoeren van een contactloze transactie en kan daardoor ook gebruikt worden door aanvallers om nieuwe transacties te starten of gevoelige bank data te onderscheppen. Wanneer een contactloze transactie gestart wordt genereert het mobiele toestel een contactloze transactie die verschillende parameters zoals het betalingstoken en verval datum bevat. Het mobiele toestel stuurt de gegevens naar de point-of-sale van de handelaar via de NFC interface. Een valse point-of-sale kan op veel verschillende manieren gecreëerd worden door een aanvalleur die slechte bedoelingen heeft:

- **Externe nep POS:** Het gebruik van een extern toestel om te communiceren met de mobiele toestel van het slachtoffer via het NFC protocol. Wanneer zo een nep POS de communicatie flow correct implementeerd die gebruikt wordt door een echte POS, kan de aanvaller communiceren met het mobiele toestel van het slachtoffer zonder dat de persoon het door heeft dat hij communiceert met een nep POS.
- **Interne nep POS:** Het gebruik van een applicatie die rechtstreeks kan communiceren met de API van de NFC controller via het besturingssysteem. In dit geval is het uitwisselen van verschillende berichten via NFC niet meer nodig en kan de aanvalleur de NFC controller rechtstreeks aansturen via de API.
- **Nep POS applicatie:** Een applicatie met slechte bedoelingen die rechtstreeks communiceert en probeert om interactie te voeren met de wallet. Dit is enkel mogelijk wanneer de wallet een API heeft of calling methods van de interacties heeft.

Een nep point-of-sale kan gebruikt worden om valse transacties aan een hoog tempo te genereren zodat alle payment tokens die lokaal op het mobiele toestel zijn opgeslaan opgebruikt worden. Eens alle payment tokens die lokaal bewaard werden op zijn, worden er nieuwe payment tokens aangevraagd en weer lokaal opgeslagen. Wanneer dit gebeurt via een communicatie kanaal naar de cloud of de backend waar de tokens worden gegenereerd die niet goed beveiligd is, kan een aanvaller belangrijke informatie onderscheppen. Als het gaat om payment tokens die meermaals gebruikt kunnen worden kan de schade veel groter zijn dan wanneer het gaat om eenmalige payment tokens. Wanneer de aanvaller een grote hoeveelheid payment tokens kan bemachtigen die meermaals gebruikt kunnen worden, kan de aanvaller de tokens gebruiken om de gebruiker zijn identiteit of specifieke transacties te bevestigen.

In een recente studie is gebleken dat er een beveiligingsfout zat in een contacloos betalingsprotocol (In Groot-Brittanië). Deze fout in het protocol maakt het mogelijk voor een aanvaller om een transactie uit te voeren van bijvoorbeeld £20 zonder dat de gebruiker zijn pincode moest ingeven. De onderzoekers zijn er zelfs in geslaagd om \$ 999 999,99 over te schrijven van een credit kaart naar de bankrekening van de aanvaller. Deze aanval was gebaseerd op het gebruik van een nep point-of-sale dat draaide op een mobiele terminal, de POS kon ingesteld worden dat hij 999 999,99 eenheden van eender welke valuta kan overschrijven. Deze terminal kan gebruikt worden om transacties te starten bij mobiele toestellen in de buurt zonder dat de gebruiker zijn pincode moet ingeven.

Om het risico te verminderen die gevormd worden door nep point-of-sale, moeten we eerst de beveiliging verhogen van de wallet applicatie op een mobiel toestel. De wallet applicatie moet voorkomen dat er verbinding kan gemaakt worden met de applicatie door ongeautoriseerde applicaties van derden, dit kan gebeuren door het controleren van de oorsprong van de NFC communicaties die binnenkomen (Maurizio Cavallari, 2014).

#### 2.4.5 Brute force aanval

Een brute force aanval wordt gebruik om geëncrypteerde data te ontcijferen. Geëncrypteerde data kan enkel ontcijfert worden met behulp van de sleutel waarmee de data geëncrypteerd is. Een brute force aanval zal proberen de sleutel te raden, dit gebeurt door alle mogelijke combinaties van tekens te proberen. De snelheid waarmee de sleutel geraden kan worden hangt af van de hardware van het systeem waarmee de aanval uitgevoerd wordt. Hoe meer karakters de sleutel bevat die gebruikt is om te encrypteren hoe langer het duurt vooraleer de sleutel geraden kan worden en dus hoe veiliger de encryptie. Door het hashen van data kan een brute force aanval sterk vertraagd worden, hashing zorgt voor extra rekenwerk die nodig is om data te ontcijferen. Door de komst van quantum computers is de kans op het raden van een sleutel extreem hoog. Binnen een aantal decennia zullen de sterkste cryptografische algoritmes die we op de dag van vandaag kennen in een mum van tijd geraden kunnen worden.



## 2.5 Hardware beveiliging vs risicobeperking

Het beveiligingsmodel van een Secure Element implementatie hangt sterk af van de beveiliging op hardware niveau en de tampered-proofed SE chip. De hoge graad van beveiliging en de zekerheid dat een Secure Element biedt laat financiële instellingen toe om betalingsgegevens door te sturen naar het mobiele toestel, juist zoals de betalingsgegevens die opgeslagen worden op een fysieke kaart bij de productie van de kaarten. Eens de gegevens opgeslagen zijn op het Secure Element kunnen ze gebruikt worden om betalingen te verrichten totdat ze vervallen of de gebruiker beslist om ze te verwijderen. Bij een HCE implementatie is het de bedoeling dat dezelfde betalingsgegevens opgeslagen worden binnen de HCE service. Aangezien dat de applicaties die op het besturingssysteem opgeslagen worden veel gevoeliger zijn voor aanvallen dan applicaties die opgeslagen worden op een Secure Element is het risico te hoog om betalingen uit te voeren. Hoe u een HCE applicatie kunt beveiligen wordt verder uitgelegd in sectie 2.6 (Alliance, 2014)

## 2.6 Beveiliging HCE

Met de komst van Host-base Card Emulation sinds Android 4.4 is er geen nood meer aan een Secure Element. HCE brengt wel een aantal beveiligingsrisico's met zich mee. Bij een simulatie van smart cards via SE wordt de applicatie geïnstalleerd op het Secure Element, aangezien het Secure Element hardwarematig beveiligt is tegen fraude is de applicatie automatisch ook beveiligt tegen veiligheidsbedreigingen. Wanneer er gekozen wordt voor een HCE gebaseerde simulatie wordt de applicatie gewoon op het apparaat geïnstalleerd en is deze niet meer beveiligt tegen bedreigingen van andere applicaties die ook op het apparaat staan. Doordat de communicatie tussen de NFC-controller en de HCE-applicatie niet meer beveiligt is kan de communicatie tussen deze twee onderschept worden door andere malware applicaties. Malware applicaties vormen een groot gevaar voor uw Android toestel, deze applicaties kunnen het besturingssysteem aanvallen. Het risico van een aanval op het besturingssysteem wordt ook vergroot door het Android toestel te rooten. De malware applicatie kan ook de mogelijkheid hebben om het toestel zelf te rooten of de gebruiker in de val te lokken om dit te doen (Alliance, 2014).

De naam rooting of rooten is afkomstig van de term root die ook in de Linux wereld gebruikt wordt. Android is gebaseerd op de kernel van Linux, wanneer de gebruiker in Linux (dus ook in android) root rechten heeft, heeft deze "rootgebruiker" alle rechten om acties uit te voeren. Het is dus duidelijk dat het niet handig of veilig is wanneer een gebruiker zomaar alle rechten heeft, dit voorkomt dat de gebruiker bewust of onbewust schade kan aanrichten aan het systeem. Één van de meest voorkomende redenen waarom android gebruikers hun toestel rooten is zodat ze alternatieve kernels kunnen installeren. Door de kernel van een toestel te veranderen kan de snelheid en de accuduur van een toestel enorm te verbeteren, via een alternatieve kernel heeft men ook de mogelijkheid om de kloksnelheid van de processor aan te passen, de kleuren van het scherm of de kracht van de GPU (Graphics Processing Unit) aan te passen.

Om deze bedreigingen tegen te gaan biedt de technologie de dag van vandaag een waaier

aan mogelijke oplossingen die hiervoor ingezet kunnen worden:

- White box cryptography
- Tamper proofed software
- Biometric factors
- Device identity solutions
- Security frameworks/trusted execution environment
- Encryption
- Tokenization
- Bijkomende beveiliging voorzien door een SE

## 2.7 Beveiliging op hardware niveau

### 2.7.1 Biometric Factors

Biometric factors of biometrische factoren kunnen gebruikt worden bij de authenticatie van een gebruiker in Host-based Card Emulation applicaties. Biometrische factoren worden meestal gebruikt in samenwerking met andere authenticatie middelen. Wat biometrische factoren zo aantrekkelijk maakt bij de authenticatie van een gebruiker is voornamelijk de gebruiksvriendelijkheid, zeker in vergelijking met het bijhouden van meerdere wachtwoorden. Er bestaan verschillende soorten biometrische factoren die gebruikt kunnen worden: vingerafdruk, gezichtsherkenning, irisscan en stemherkenning. Deze biometrische factoren zitten al een tijdje geïntegreerd in de meeste laptops en smartphones en kunnen dus gemakkelijk op applicatie niveau gebruikt worden. Een bijkomend probleem is de privacy en beveiliging omtrent de biometrische data die ook in achtting genomen zal moeten worden bij het implementeren van de applicatie (Alliance, 2014).

### 2.7.2 Secure Element

HCE maakt het mogelijk om communicatie met de NFC-controller te laten verlopen zonder een SE, maar direct op de processor van het toestel, HCE specificeert niet waar de data opgeslagen moet worden. De data kan zowel in de cloud als op een SE opgeslagen worden dus een combinatie van de twee is zeker ook mogelijk wanneer een SE aanwezig is in het toestel. Data die op het SE opgeslagen worden zijn beveiligd door het gebruik van cryptografische sleutels. De beveiliging van een SE kan nog versterkt worden door gebruik te maken van een TEE (zie 2.9.1). Het blijft wel nog altijd veiliger de applicatie en de data enkel op een SE uit te voeren en op te slaan (Alliance, 2014).

## 2.8 Beveiliging op software niveau

### 2.8.1 White Box Cryptography

White box cryptography is een manier om keys die in het geheugen van een toestel zitten of in de code zitten te verbergen voor aanvallers. Bij het ontwikkelen van software worden vaak keys gebruikt voor encryptie maar deze kunnen gemakkelijk achterhaald worden in een applicatie via het doorlopen en analyseren van de code van de applicatie. Via white box cryptography wordt de key als het ware omgevormd naar een stuk onleesbare tekst, dit resultaat wordt verkregen door de key samen te smelten met een cipher tekst via een deterministisch algoritme. De key en de cipher tekst vormen een onleesbaar stuk tekst die niet zomaar achterhaalt kan worden. Een deterministisch algoritme is een algoritme waarbij een bepaalde input altijd voor dezelfde output zorgt. Aangezien de keys niet meer leesbaar zijn is het dus ook niet meer mogelijk om de keys zomaar te achterhalen via het doorlopen van de code van de applicatie.

### 2.8.2 Tamper-Proofed Software

Tamper proofing software is een extra beveiligingslaag in de software die ervoor zorgt dat het moeilijker is voor de aanvaller om de code statisch of dynamisch aan te passen of reverse engineering toe te passen op de code. Dit kan gebeuren op verschillende manieren, runtime integrity checking, breakpoint defenses, obfuscation, anti-debug, ... Wanneer er een aanval wordt gedaan en ook effectief wordt waargenomen door de software zal het tamper-proofed systeem een antwoord produceren waardoor de aanvallende software faalt en de aanval dus vermeden wordt (Alliance, 2014).

### 2.8.3 Device Identity Solutions

Device identity solutions gebruiken online diensten voor authenticatie van een toestel die zorgen voor een extra beveiligingslaag voor de Host-based Card Emulation applicaties. Fast Identity Online (FIDO) is een voorbeeld van zo'n onlinedienst. FIDO maakt gebruik van publieke sleutel cryptografie technieken voor online authenticatie, het toestel van de gebruiker creëert een sleutelpaar waarbij de privésleutel bijgehouden wordt en de publieke sleutel geregistreerd wordt bij de online dienst. Het authenticeren van het toestel bij de online service wordt gedaan door middel van de privésleutel die enkel lokaal ontgrendeld kan worden via biometrische factoren of door het ingeven van een PIN-code. FIDO ondersteunt veel verschillende technologieën die naast elkaar gebruik kunnen worden zoals tokenisatie en one-time-password oplossingen. PayPal was één van de eerste die gebruik heeft gemaakt van vingerafdruk verificatie op de Samsung Galaxy S5 met FIDO Ready software (Alliance, 2014).

### 2.8.4 Encryption

Encryptie is een methode die het mogelijk maakt om data onleesbaar te versturen en niet als gewone tekst. Het ophalen van data die in gewone tekst verstuurd wordt is één van de grootste vormen van data breaches in card-present en card-not-present betalingsomgevingen, wanneer kaarten gelezen of de gegevens ingelezen worden in een web-based formulier. Er bestaan twee verschillende soorten encryptie end-to-end encryptie (E2EE) en point-to-point encryptie (P2PE). Deze methodes zorgen ervoor dat de data geëncrypteerd is bij NFC-lezer en het versturen van de data. Voor betalingsapplicaties kan encryptie gecombineerd worden met tokenisatie (zie 2.8.5), hierbij wordt het kaartnummer van een bankkaart geëncrypteerd en dan wordt het geëncrypteerde nummer gebruikt om een token te gaan genereren (Alliance, 2014).

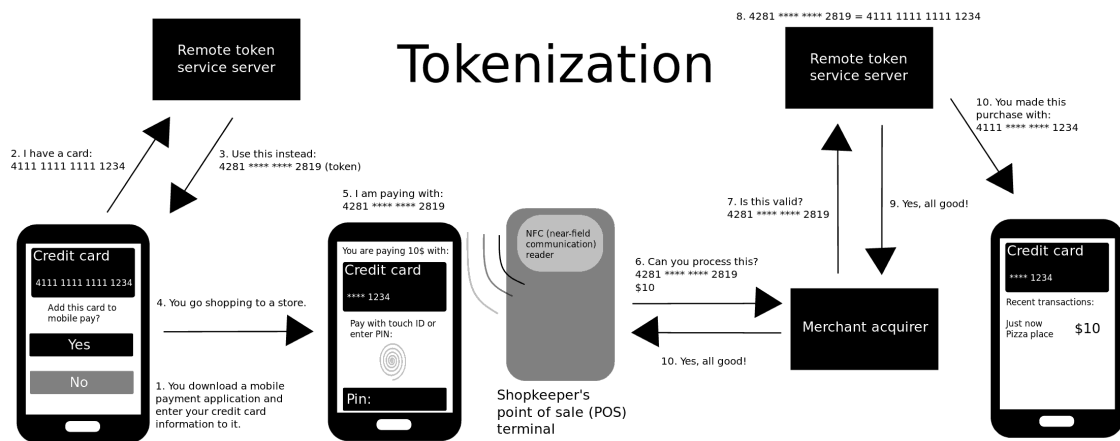
Geëncrypteerde data kan enkel ontcijfert worden met sleutel waarmee de data geëncrypteerd is. Wanneer een de data onderschept kan worden tijdens een aanval kan de data niet ontcijfert worden zonder de sleutel ofwel via een brute force aanval.

- **End-to-end encryptie:** End-to-end encryptie verloopt aan de hand van publieke en privé sleutels. Ieder toestel of gebruiker genereert een privé sleuten en een publieke sleutel, de privé sleutel wordt lokaal bijgehouden op het toestel en de publieke sleutel wordt opgeslagen op een server of kan meegestuurd worden. Wanneer een transactie gestart wordt door de applicatie zullen de gegevens geëncrypteerd worden met de publieke sleuten van de tegenpartij, de gegevens kunnen enkel gedecrypteerd worden via de privé sleutel van de tegenpartij en vice versa.
- **Point-to-point encryptie:** Bij point-to-point encryptie worden de gegevens bij de start van een transactie geëncrypteerd dit gebeurt via een algoritmische berekening. De gegevens worden dan doorgestuurd naar een point-of-sale maar zijn nooit zichtbaar voor de handelaar. Wanneer de gegevens in de beveiligde data zone terechtkomen van de betalingsverwerker worden de geëncrypteerde gegevens doorgestuurd naar de juiste financiële instelling. De financiële instelling zal de gegevens op zijn beurt weer decrypteren en de transactie al dan niet laten doorgaan.

### 2.8.5 Tokenization

Tokenisatie is een methode die een willekeurige waarde teruggeeft van bijvoorbeeld een kaartnummer, rijksregisternummer, ... Tokenisatie op zich wordt al lang gebruikt, maar heeft de laatste jaren meer aandacht gekregen door het toenemende aantal data breaches. Door het lekken van al die data is er een grote nood aan tokenisatie om betalingsgegevens te beschermen tegen fraude en vervalsing. Verschillende grote spelers binnen de beveiligingsindustrie hebben nieuwe standaarden opgesteld voor tokenisatie, één daarvan is EMVCo. De tokenisatie standaarden die uitgebracht zijn door EMVCo voor betalingen kunnen heel domeinspecifiek zijn en bevat cryptogrammen die HCE use cases kunnen isoleren. Door dit toe te passen kan voorkomen worden dat de tokens gebruikt kunnen worden binnen andere betalingskanalen (Alliance, 2014).

Hoe het principe van tokenisatie beter te begrijpen kunt u de flow van een betalingsapplica-



Figuur 2.2: Flow van een mobiele betalingsapplicatie met tokenisatie

tie bekijken op figuur 2.2. Wanneer de gebruiker de betalingsapplicatie download moet zal er gevraagd worden om de bankgegevens in te vullen. Eens de gegevens ingevuld zijn moet het rekeningnummer vervangen worden door een token die aangevraagd wordt aan een token service server, die gegenereerd een token en geeft die terug aan de applicatie. De betalingsapplicatie kan dit token gebruiken om transacties te doen in plaats van het echte rekeningnummer zodat de gevoelige informatie niet onderschept kan worden tijdens de transactie. Een transactie wordt geïnstantieerd via een point-of-sale, de betalingsapplicatie geeft het verkregen token door aan de point-of-sale die het op zijn beurt terug doorgeeft aan de bank. De bank zal dan verifiëren dat het token gekend is in het systeem, als de transactie goedgekeurd is zal dit doorgegeven worden aan de point-of-sale en aan de betalingsapplicatie. Op deze manier worden de gevoelige gegevens nooit gebruikt tijdens een transactie en kunnen ze ook niet onderschept en misbruikt worden door mogelijke aanvallers.

Een mooie use-case voor een HCE applicatie met tokenisatie zou een betalingsapplicatie kunnen zijn voor de smartwatch die in pretparken gebruikt kan worden door kinderen. Stel een pretpark voorziet een x aantal smartwatches met daarop een betalingsapplicatie op, ouders kunnen geld opladen aan een account die gekoppeld is aan de smartwatch. De bezit enkel een token die gekoppeld is aan de gebruiker, wanneer de gebruiker wens te betalen hoeft hij enkel maar de smartwatch tegen een point-of-sale te houden en de betaling is voltooid. Aangezien er met tokenisatie gewerkt wordt kunnen alle transacties veilig verlopen en kan er geen misbruik gemaakt worden.

## 2.9 Beveiliging op software en hardware niveau

### 2.9.1 Security Frameworks/Trusted Execution Environment

Trusted Execution Environment (TEE) is een veilige plaats in de hoofd processor of coprocessor van het toestel waar data kan opgeslagen en verwerkt worden. De bedoeling van een TEE is het uitvoeren van geautoriseerde beveiligingssoftware in een vertrouwde

omgeving. TEE bestaat niet enkel uit software maar ook uit hardware die bescherming bieden tegen aanvallen vanuit het besturingssysteem in het toestel. Gevoelige applicaties die beschermd moeten worden tegen het besturingssysteem van het toestel worden opgeslagen in de TEE en helpt ook bij de controle van toegangsrechten tot de applicaties. TEE heeft zijn eigen besturingssysteem, hierdoor kan de TEE niet aangetast worden wanneer het besturingssysteem van het toestel aangetast is. De TEE kan voor een extra beveiligingslaag zorgen voor HCE applicaties (Alliance, 2014):

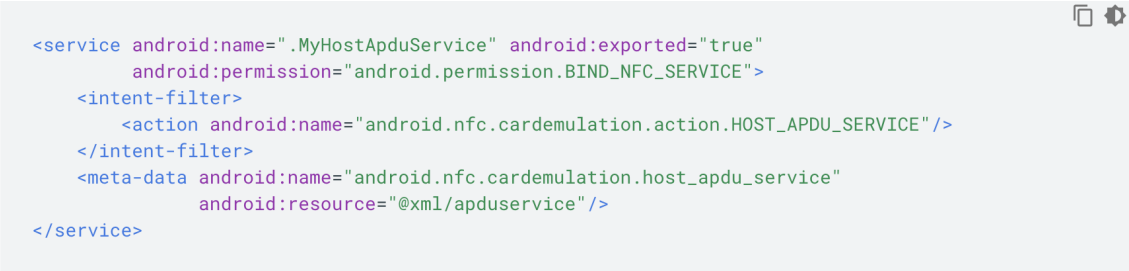
- **PIN/wachtwoord ingave.** De TEE kan extra bescherming aanbieden aan de hand van het invoeren van een PIN-code of een wachtwoord. Bij de TEE is de invoer van de PIN-code of een wachtwoord volledig gescheiden van de invoer van het toestel, hierdoor kan de invoer niet onderschept worden door malware applicaties die zich op het besturingssysteem van het toestel bevinden.
- **Secure storage of credentials.** De TEE implementeert cryptografische operaties binnen de secure execution environment en zorgt voor het veilig opslaan van sleutels. Hierin kunnen dus tokens/sleutels van betalingsapplicaties in opgeslagen worden, zo zijn deze beter beveiligd tegen aanvallen dan wanneer ze opgeslagen worden in het besturingssysteem van het toestel.
- **Secure transfer protocol endpoint.** Het is mogelijk om in de TEE een geëncrypteerd beveiligd kanaal op te zetten aan de kant van de terminal. Dit betekent dat de APDU's geëncrypteerd verstuurd kunnen worden van de terminal en de HCE applicatie via een draadloze interface naar de TEE. Een tweede geëncrypteerd beveiligd kanaal kan opgezet worden tussen de TEE en een Cloud applicatie. Hierdoor zijn de sleutels en data enkel zichtbaar binnen de vertrouwde applicatie, dit biedt een hoger niveau van beveiliging aan dan wanneer de applicatie uitgevoerd zou worden op het besturingssysteem van het toestel.

## 3. Methodologie

In dit hoofdstuk zal er meer uitleg gegeven worden over het opstellen van de verschillende proof of concepts en waarom voor bepaalde implementaties gekozen zijn. Hierbij zal ook besproken worden hoe een basis Host-based Card Emulation applicatie opgebouwd wordt.

### 3.1 Beveiligingsmethodes

Voor dit onderzoek werd er gekozen voor drie verschillende beveiligingsmethodes uit 2.6 die verder uitgewerkt zullen worden als proof-of-concept. De drie gekozen beveiligingsmethodes zijn biometric factors, encryption en tokenization. Door het kiezen voor deze drie beveiligingsmethodes kan er al een mooie vergelijking gemaakt worden tussen één beveiligingsmethode op hardware niveau namelijk biometric factors (in dit geval vingerscan) en twee beveiligingsmethodes op software niveau. Het gebruik van biometrische factoren zorgt ook voor een groot veiligheidsgevoel bij de gebruiker wat het een interessante implementatie maakt naar user-experience toe. Voor het testen van de verschillende gekozen beveiligingsmethodes is er natuurlijk een Android applicatie nodig waarbij het simuleren van een smartcard geïmplementeerd is aan de hand van Host-based Card Emulation. Google heeft een groot aanbod aan voorbeeld applicaties waarin verschillende technieken geïmplementeerd zijn via de best practices, waaronder ook een applicatie waarin HCE geïmplementeerd is die voor dit experiment gebruikt zal worden. In deze standaard applicatie zullen de drie beveiligingsmethodes geïmplementeerd en vergeleken worden.



```
<service android:name=".MyHostApuService" android:exported="true"
    android:permission="android.permission.BIND_NFC_SERVICE">
    <intent-filter>
        <action android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE" />
    </intent-filter>
    <meta-data android:name="android.nfc.cardemulation.host_apdu_service"
        android:resource="@xml/apduservice" />
</service>
```

Figuur 3.1: Service meta-data in AndroidManifest

## 3.2 Opbouw Host-based Card Emulation applicatie

Bij het ontwikkelen van een Host-based Card Emulation applicatie moeten er een paar dingen aangepast worden in de AndroidManifest en er moet een APU service klasse aangemaakt worden. Wanneer een standaard öne activityapplicatie gemaakt is moeten er twee permissies toegevoegd worden in de Android Manifest, `<uses-feature android:name=android.hardware.nfc.hceandroid:required="true"/>` en `<uses-permission android:name=android.permission.NFC/>`, de eerste permissie laat de applicatie toe om hardware HCE technologie te gebruiken en de tweede permissie laat het toe om de NFC technologie van het apparaat te gebruiken. De APU service moet ook gedeclareerd worden in de AndroidManifest zie 3.1. Bij de declaratie van de service wordt er toestemming gegeven aan externe applicaties om verbinding te maken met de service aan de hand van `android:exported="true"`, de permissie `android.permission.BIND_NFC_SERVICE` zorgt ervoor dat enkel externe applicatie met deze permissie verbinding kunnen maken met de service.

Bij het implementeren van de apdu service klasse wordt de klasse uitgebreid met de HostApuService klasse van android. De HostApuService klasse bevat twee abstracte methodes die overschreven worden in de eigen service klasse namelijk `processCommandApu` en `onDeactivated`. De `processCommandApu` methode wordt aangeroepen wanneer een NFC reader een APDU stuurt naar de apdu service en dan zal de service een APDU als antwoord terug sturen. Er moet wel rekening mee gehouden worden dat de `processCommandApu` methode aangeroepen wordt op de main thread van de applicatie, wanneer het toestel dit niet aankan mag de main thread natuurlijk niet geblokkeerd worden. Wanneer de main thread het niet aankan moet de methode null teruggeven en een andere methode `sendApuResponse` die het antwoord op een andere thread terug zal sturen. De `onDeactivated` methode geeft terug of de apdu is teruggestuurd op de main thread of een andere thread wanneer de `processCommandApu` wordt aangeroepen.

## 3.3 Implementatie biometric factors

Één van de beveiligingsmethodes die gekozen werd voor dit onderzoek zijn biometric factors, er zijn natuurlijk meerdere biometrische factoren die gebruikt kunnen worden om de applicatie te beveiligen zoals vingerscan, irisscan, spraak herkenning,... Hier werd



gekozen voor de vingerscan te implementeren omdat dit een van de meest voorkomende beveiligingstechnologieën bij mobiele toestellen. Wanneer we een implementatie doen met vingerscan is het niet voldoende om enkel te vertrouwen op deze technologie of eender welke biometrische factor, het kan altijd voorkomen dat de technologie stopt met werken of de gekozen biometrische factor niet herkend wordt. Wanneer dit voorkomt moet er een back-up zijn, als back-up wordt er gekozen voor de pincode van het toestel te gebruiken ofwel en zelf ingestelde pincode voor de applicatie. Eerst en vooral moet er toestemming gegeven worden aan de applicatie om gebruik te maken van de vingerscan hardware van het mobiele toestel, dit gebeurt in de Android Manifest door middel van een metadata tag user-permission met de inhoud `android:name=android.permission.USE_FINGERPRINT`. Voor de vingerscan implementatie moeten we twee extra klassen aangemaakt worden de `FingerprintUiHelper` klasse en de `FingerprintAuthenticationDialogFragment` klasse. De `FingerprintUiHelper` klasse helpt om de tekst en de iconen te veranderen op de vingerscan authenticatie scherm. `FingerprintAuthenticationDialogFragment` communiceert met de vingerscan API om de gebruiker te authenticeren aan de hand van de vingerscan en als de vinger niet herkend wordt zal de authenticatie gebeuren aan de hand van de pincode. Bij het opstarten van de applicatie zal de vingerscan authenticatie scherm getoond worden vooraleer de gebruiker kan verder gaan en gebruik kan maken van de applicatie.

### 3.4 Implementatie encryption

Bij de implementatie van encryption moeten er drie methodes toegevoegd worden aan de `CardService` die zorgen voor het encrypteren en het decrypteren van data respectievelijk `encrypt`, `decrypt` en `generateKey`. De `generateKey` functie genereert een unieke geheime sleutel die gebruikt wordt om data te encrypteren. De `encrypt` functie maakt gebruik van de `generateKey` functie om een sleutel te genereren die gebruikt zal worden samen met het geïnitieerde Cipher in `ENCRYPT_MODE` om de data te encrypteren. Bij de `decrypt` functie verloopt alles juist hetzelfde als bij de `encrypt` functie het enigste verschil is dat de Cipher geïnitieerd zal worden in `DECRYPT_MODE`. Wanneer de gebruiker zijn accountnummer ingeeft in de applicatie zal het nummer geëncrypteerd worden en opgeslagen worden, wanneer het accountnummer getoond moet worden, zal het geëncrypteerde accountnummer weer gedecrypteerd worden. Wanneer een externe NFC-lezer de functie `processCommandApdu` aangeroep zal de geëncrypteerde data verstuurd worden en gedecrypteerd worden aan de kant van de NFC-lezer. Door de gegevens geëncrypteerd te versturen kan de data niet ontcijfert wanneer de data onderschept wordt door een aanval-ler. De aanval-ler kan de data enkel ontcijferen wanneer die in bezit is van de sleutel die gebruikt werd om de data te encrypteren (overflow, g.d.).

### 3.5 Implementatie tokenization

Tokenization wordt aanzien als een van de meest veilige vormen die geïmplementeerd kan worden in een betalingsapplicatie. Normaal gezien worden betalingstokens gegenereerd aan de kant van de backend waar het token gekoppeld wordt aan het de gebruiker die het

token aanvraagt. Voor het doel van dit onderzoek was de implementatie van een volledig werkende backend overbodig en zal de functionaliteit nagebootst worden in de applicatie. Voor de functionaliteit na te bootsen implementeren we de functie `nextToken` die een nieuwe token genereert die dan doorgestuurd kan worden in plaats van het accountnummer. Iedere keer de `processCommandApu` functie wordt aangeroepen door een externe NFC-lezer wordt de functie `nextToken` aangeroepen en wordt het nieuwe gegenereerde token doorgestuurd. Wanneer er gewerkt wordt met eenmalige tokens en een token wordt onderschept door een aanvaller kan de aanvaller hier niets mee aanvangen aangezien het token maar eenmalig gebruikt kan worden voor een transactie of authenticatie.

### 3.6 Onderzoek

De grote vraag die nu gesteld kan worden is, welke beveiligingsmethode is nu de beste? Maar we moeten ons natuurlijk ook andere vragen stellen, is er maar één methode het veiligst? Zijn er meerdere methodes het veiligst? Kunnen sommige methodes gecombineerd worden? ... Om op al deze vragen een antwoord te kunnen geven zullen we de verschillende gekozen beveiligingsmethodes moeten vergelijken met elkaar. Het vergelijken van deze methodes kan op veel verschillende manieren gebeuren, we kunnen puur kijken naar de veiligheid van de methodes maar eigenlijk moeten we ook kijken naar de moeilijkheidsgraad van de implementatie. Niet iedere applicatie die gebruik wenst te maken van Host-based card emulation heeft een even hoge veiligheidsgraad nodig, zo kan een beveiligingsmethode te gecompliceerd zijn voor die specifieke use-case en heeft dit onderzoek niet veel nut.

Wat zal er in dit onderzoek nu precies onderzocht worden? Er zal gekeken worden naar de moeilijkheidsgraad van de implementatie van de beveiligingsmethode, op welke momenten zijn de gevoelige gegevens zichtbaar, is de geïmplementeerde beveiligingsmethode gemakkelijk te omzeilen, is er een mogelijkheid om verschillende methodes te combineren, zorgt de combinatie van methodes voor extra veiligheid, is het overbodig om sommige methodes te combineren? Eenmaal we al deze vragen onderzocht hebben kan er een conclusie gemaakt worden over de verschillende methodes en kan u een geïnformeerde beslissing maken over welke implementatie het beste past voor uw specifieke use-case.

## 4. Resultaten

In dit hoofdstuk zullen de resultaten van het experiment besproken worden op basis van de criteria uit sectie 3.6. de verschillende criteria zijn:

- Moeilijkheidsgraad van de implementatie
- Zichtbaarheid van de gevoelige gegevens
- Veiligheid van de beveiligingsmethode
- Combinatie van verschillende beveiligingsmethodes
- Voordelen van het combineren van verschillende methodes

### 4.1 Moeilijkheidsgraad van de implementatie

Om de moeilijkheidsgraad van de implementatie van een beveiligingsmethode te bepalen kunnen we niet alleen kijken naar de kennis die nodig is om de methode te implementeren maar ook naar alle resources die nodig zijn om een juist implementatie te voltooien. Het aantal resources die nodig zijn voor een beveiligingsmethode kan vaak de doorslag geven of er gekozen wordt voor de ene methode of de andere.

#### 4.1.1 Biometric factors

Als we kijken naar de implementatie met biometrische factoren moet de gebruiker van de applicatie natuurlijk beschikken over de hardware die het uitlezen van deze biometrische factor kan herkennen. Bijna alle nieuwe mobiele toestellen die beschikbaar zijn voor de consument komen met een vingerscanner. Bij het ontwikkelen van de applicatie kan er gemakkelijk gecontroleerd worden of het mobiele toestel beschikt over een vingerscanner

of over een andere biometrische factor scanner. Wanneer het mobiele toestel niet beschikt over de nodige hardware om de biometrische factoren te herkennen kan er nog altijd gebruik gemaakt worden van een pincode. Eenmaal geweten dat de authenticatie van de gebruiker zal gebeuren via biometrische scanner of de pincode kan de gebruiker geauthenticeerd worden en krijgt de gebruiker al dan niet toegang te de applicatie en de gegevens. Buiten de pure implementatie van het gebruik van biometrische factoren zal er ook nagedacht moeten worden over de opslag en de beveiliging van de biometrische data die verkregen wordt.

#### 4.1.2 Encryption

Bij een implementatie met encryptie van de gegevens is er geen nood voor specifieke hardware van het mobiel toestel zoals bij de biometrische factoren. Voor het encrypteren van de gegevens hebben we enkel een encryptie en een decryptie functie nodig om de gegevens te kunnen encrypteren en te decrypteren. Wanneer de gegevens opgevraagd worden door een point-of-sale zullen de gegevens eerst geëncrypteerd worden en dan verzonden worden naar de point-of-sale. Om de werken tussen de applicatie en de point-of-sales goed te laten werken kan er gebruik gemaakt worden van een oplossing van een derde partij die zorgt voor de juiste implementatie aan beide kanten.

#### 4.1.3 Tokenization

Voor tokenization hebben we heel wat meer resources nodig dan bij encryption en biometrische factoren. Om de veiligheid van tokenisatie te garanderen worden de tokens gegenereerd op de server waar de backend op draait. Wanneer de gebruiker een transactie wil doen met de applicatie gebeurt dit aan de hand van een token die werd verkregen via de server en werd gekoppeld aan de gebruiker. Wanneer er een transactie gebeurt via het token zal aan de point-of-sale kant gecontroleerd worden of het token gekoppeld is aan een gebruiker en zo kan de transactie voltooid worden of juist niet als het token niet meer gekoppeld is aan een gebruiker.

### 4.2 Zichtbaarheid van de gevoelige gegevens

#### 4.2.1 Biometric factors

Bij het gebruik van biometrische factoren om uw applicatie te beveiligen kan je enkel toegang krijgen tot de applicatie wanneer uw vingerafdruk gekend is in het systeem of wanneer je de pincode kent wat een groot voordeel is en een groot veiligheidsgevoel geeft aan de gebruiker van de applicatie. Het grote nadeel is dat de gegevens tijdens een transactie als gewone leesbare tekst doorgestuurd wordt. Wanneer er een aanval gebeurt en de informatie onderschept wordt zijn de gegevens leesbaar voor de aanvaller en kunnen deze misbruikt worden.

### 4.2.2 Encryption

Wanneer je encryptie implementeert in uw applicatie worden de gevoelige gegevens geëncrypteerd verstuurd. Wanneer de informatie toch onderschept wordt door een aanvaller zijn de gegevens onleesbaar voor de aanvaller. De gegevens kunnen enkel ontcijferd worden wanneer men in bezit is van de sleutel die gebruikt is om de gegevens te encrypteren. De gegevens zijn dus zichtbaar bij de invoer ervan of wanneer ze gedecrypteerd worden.

### 4.2.3 Tokenization

Bij tokenisatie worden de gevoelige gegevens vervangen door random gegenereerd token die niet ontcijfert kan worden. Dit token wordt gebruikt in plaats van de gevoelige gegevens en wordt dus doorgestuurd wanneer een transactie gedaan wordt. Aangezien alles via een token verloopt zijn de gegevens enkels zichtbaar bij de ingave, het verschil met encryptie is dat het token niet ontcijferd kan worden.

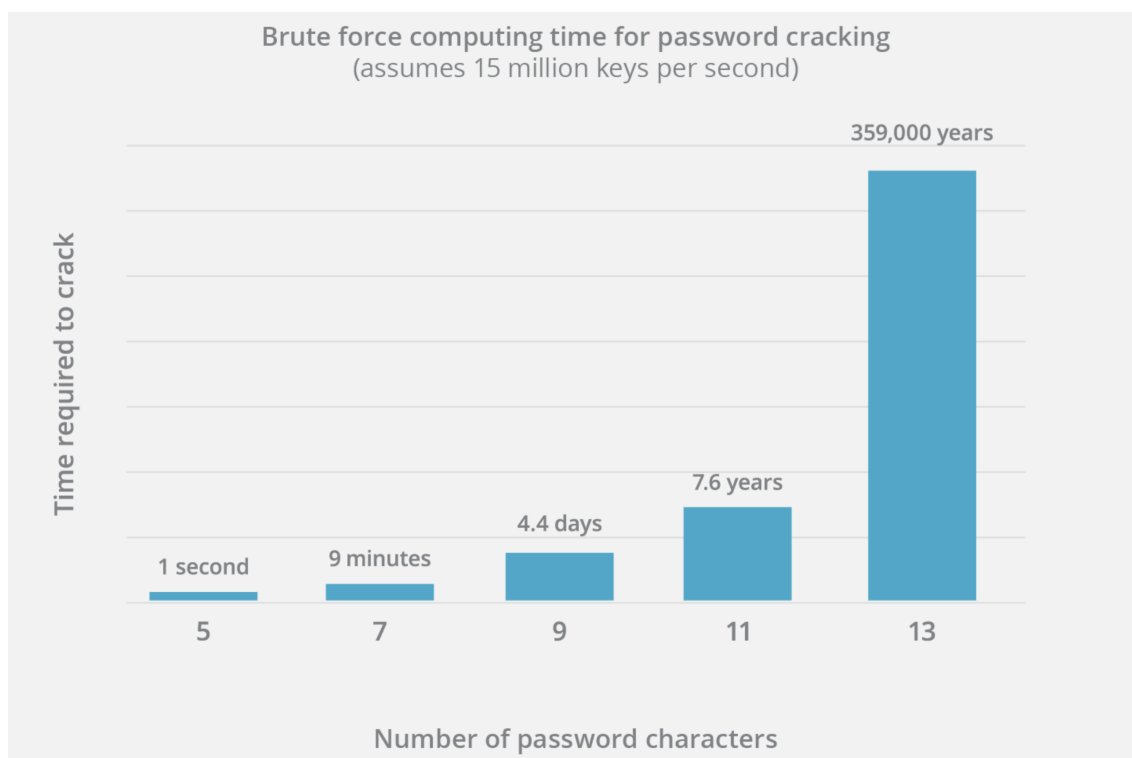
## 4.3 Veiligheid van de beveiligingsmethode

### 4.3.1 Biometric factors

Biometrische factoren zoals gezichtsherkenning, irisscan, spraakherkenning en vingerscan praktisch niet na te bootsen, vingerscan kan omzeild worden als de aanvaller de vingerafdruk van de gebruiker kan bemachtigen. Wanneer men werkt met biometrische factoren is er altijd een back-up met een pincode. Een pincode kan gemakkelijk gekraakt worden via brute force aanvallen (dit is een aanval die alle mogelijke combinaties probeert tot de juiste combinatie gevonden is). Er kan natuurlijk een extra regel voorzien worden dat er maar een bepaald aantal pogingen gedaan mogen worden om de applicatie te ontgrendelen. Zolang de aanvaller niet voorbij deze beveiligingen raakt heeft hij geen toegang tot de gegevens van de gebruiker.

### 4.3.2 Encryption

Geëncrypteerde data kan enkel ontcijfert worden als men in bezit is van de sleutel waarmee de data is geëncrypteerd. Een aanvaller kan proberen om de data te ontcijferen zonder de encryptie sleutel door middel van een brute force aanval. Net zoals bij een brute force aanval voor een pincode zal hier alle mogelijke combinaties geprobeerd worden tot dat de sleutel gevonden is. Hoe langer of groter de sleutel is hoe langer het duurt vooraleer een bruteforce aanval de sleutel kan ontcijferen. In figuur 4.1 ziet u een representatie van hoe lang het duurt om een geëncrypteerd wachtwoord te ontcijferen in vergelijking met het aantal karakters van het wachtwoord (cloudflare, g.d.).



Figuur 4.1: Representatie van de nodige tijd om een geëncrypteerd wachtwoord te ontcijferen

### 4.3.3 Tokenization

Bij tokenisatie wordt er gebruikgemaakt van tokens, dit is een reeks van tekens die willekeurige gegenereerd worden. Aangezien de tokens willekeurig gegenereerd worden en geen gegevens bevatten kunnen ze dus ook niet ontcijfert worden. Wanneer er gebruikt gemaakt wordt van eenmalige tokens kan de aanvaller hier niets mee aanvangen eenmaal ze onderschept zijn. Als het gaat om tokens die meermaals gebruikt kunnen worden kan de aanvaller zich voordoen als de rechtmatige gebruiker van de tokens. De aanvaller kan gebruikmaken van de tokens tot de levensduur van de tokens vervallen zijn of tot wanneer ze uit het systeem verwijderd zijn door het opmerken en het melden van de aanval.

## 4.4 Combinatie van verschillende beveiligingsmethodes

Het is mogelijk om de verschillende beveiligingsmethodes te combineren met elkaar. Er kan gekozen worden om een applicatie met encryptie te combineren met biometrische factoren of tokenisatie te combineren met biometrische factoren. Zo is de toegang tot de betalingsapplicatie beveiligd en worden de gegevens beveiligd verstuurd. Encryptie en tokenisatie kunnen ook perfect gecombineerd worden hierbij worden de gegevens eerst geëncrypteerd en dan verzonden naar de server om daarvoor dan een token te laten genereren. Het is ook mogelijk om alle drie van de gekozen beveiligingsmethodes te combineren met elkaar. Dit zorgt voor beveiligde toegang tot de applicatie, data die

geëncrypteerd verstuurd wordt naar de server om dan een niet ontcijferbare token te genereren waarmee transacties geautoriseerd kunnen worden. Dit onderzoek beperkt zich nu tot deze drie methodes maar het is ook mogelijk om andere methodes uit sectie 2.6 te combineren met elkaar.

## 4.5 Voordelen van het combineren van verschillende methodes

Het combineren van verschillende beveiligingsmethodes zorgt logischerwijs voor extra voordelen namelijk extra veiligheid. Wanneer we enkel een implementatie doen van biometrische factoren is enkel de toegang tot de applicatie beveiligd maar niet het versturen van de gevoelige gegevens. Bij de implementaties met encryptie en tokenisatie is het versturen van de gevoelige gegevens van de gebruiker wel beveiligd maar de toegang tot de applicatie is vrij. Wanneer we biometrische factoren combineren met encryptie ofwel tokenisatie zorgt dit voor een extra beveiligingslaag namelijk de toegang tot de applicatie zelf. Zoals in het in sectie 4.4 al vermeld werd kunnen ook alle drie van deze beveiligingsmethodes gecombineerd worden met elkaar en zorgt dit weer voor een extra beveiligingslaag voor de applicatie. Er moet ook wel rekening mee gehouden worden wanneer men kiest voor een combinatie van verschillende beveiligingsmethodes of dit wel nodig is voor de specifieke use-case, als het gaat om een simpele loyalty applicatie (bijvoorbeeld klantenkaart om punten te sparen) dan is een combinatie van tokenisatie en biometrische factoren misschien overbodig door de hogere kost die erbij komt bij het implementeren van tokenisatie. Het combineren van biometrische factoren met encryptie of tokenisatie kan ook op een andere manier bekeken worden dan puur op de extra veiligheid van de applicatie, het implementeren van biometrische factoren in een applicatie zorgt voor een groter veiligheidsgevoel bij de gebruiker. Dit zorgt voor een grotere user-experience wat op zijn beurt zorgt voor een grotere klanten tevredenheid.





## 5. Conclusie

Om te kunnen concluderen wat nu de beste of veiligste beveiligingsmethodes is om een host-based card emulation applicatie te beveiligen moeten we alles in zijn geheel gaan bekijken. Als u voor het eerst nadenkt over hoe u moet bepalen wat nu de beste methode is om uw applicatie te beveiligen denkt u waarschijnlijk direct aan hoe sterk de methode uw gegevens beveilgd. Maar om tot de beste oplossing te komen moeten we niet enkel naar de graad van veiligheid kijken maar ook naar de kost van implementatie en het opzetten van de beveiligingsmethode.

Dit is ook een belangrijk gegeven om rekening mee te houden want niet iedere ontwikkelaar beschikt over evenveel resources maar niet iedere applicatie heeft een even ingewikkelde of dure implementatie nodig. Elke beveiligingsmethode heeft ook zijn voor en nadelen. Biometrische factoren hebben het voordeel dat de niet gemakkelijk na te bootsen zijn en dat de toegang tot de applicatie beveilgd is, het nadeel is echter dat wanneer er gevoelige data verstuurd moet worden deze niet beveilgd is dus wanneer de data onderschept wordt de gegevens zomaar leesbaar zijn. Bij encryptie is het voordeel dat de gevoelige gegevens geëncrypteerd verstuurd worden en dus niet zomaar leesbaar zijn, de geëncrypeerde data kan enkel ontcijferd worden via de sleutel waarmee de data geëncrypteerd is. Het nadeel van encryptie is dat de toegang tot de applicatie niet beveilgd is. Het grote voordeel bij tokenisatie is dat de gevoelige data vervangen wordt door een onleesbare en ontcijferbaar token maar ook hier is de toegang tot de applicatie niet beveilgd. Nog een nadeel bij tokenisatie is de nood aan een server om de tokens te genereren en te bewaren wat een grote kost kan zijn voor sommige personen of bedrijven.

Gelukkig kunnen de verschillende beveiligingsmethodes gecombineerd worden voor extra veiligheid, zo kan de toegang tot de applicatie en de gevoelige data beveilgd worden. Hieruit kunnen we dus opmaken dat er niet één juiste of beste oplossing is voor alle

gevallen. Voor betalingsapplicaties waar de veiligheid van data van groot belang is zal er best gekeken worden voor een implementatie met encryptie of tokenisatie of een combinatie van beiden. Wanneer het gaat over een kleine loyalty applicatie kan het gebruik van biometrische factoren of encryptie al voldoende zijn. Wanneer u dus beslist om uw HCE applicatie te beveiligen, houd dan zeker rekening met het doel van de applicatie en de kost van implementatie van de beveiligingsmethode.

# A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

## A.1 Introductie

Tegenwoordig worden smart-cards zoals bankkaarten, klantenkaarten, ... maar al te vaak vervangen door digitale versies op smartphones en smartwatches. Dit verloopt via NFC of "Near Field Communication". Hiervoor wordt er normaal gezien een secure-element gebruikt die zorgt voor de beveiliging en de communicatie. Sinds Android4.4 (KitKat) is het mogelijk om dit te doen via HCE of "Host-based card emulation". Eén van de nadelen van deze technologie is het veiligheidsrisico die het met zich meebrengt. Android HCE maakt het gemakkelijker om de NFC technologie te gebruiken omdat er geen nood meer is aan een secure-element, waardoor er dus ook geen commerciële overeenkomst meer moet afgesloten worden met de secure-element verdelers. Het doel van deze paper is om na te gaan hoe men Android HCE op een veilige manier kan gebruiken voor allerlei betalingssystemen.

## A.2 State-of-the-art

Door het gebruik van Android HCE is er een groot veiligheidsrisico. Dit doordat er geen gebruik meer wordt gemaakt van een secure-element die zorgt dat alle gegevens van de gebruiker veilig worden opgeslagen. Volgens de paper van Smart Card Alliance (Alliance, 2014) zijn de twee meest voorkomende manieren om met Android HCE te

werken via de cloud met een token en zonder token. Zonder token werken wordt niet als veilig aanschouwd doordat betalingsgegevens gemakkelijk ontdekt kunnen worden door malware. Het gebruik maken van een token verlaagt de kans niet tot het ontdekken van betalingsgegevens door malware, maar het verlaagt wel de impact van een eventuele ontdekking door het vervangen van statische betalingsgegevens met een token. Nadelen van het gebruiken van Android HCE (Lepojevic e.a., 2014) in vergelijking met een normale Smart Card is dat HCE niet werkt zonder stroom. Wanneer er geen verbinding kan gemaakt worden met de cloud kan HCE de authenticatie niet voltooien.

### A.3 Methodologie

Om de veiligheid te kunnen testen zal er een proof of concept opgesteld worden met drie verschillende beveiligingsmethodes namelijk Tokenisatie, Encryptie en biometrische factoren. Hierbij kan er dan vergeleken worden hoe veilig ze zijn ten opzichte van elkaar en ten opzichte van het gebruiken van een secure-element. Bij het vergelijken van de verschillende methoden zal er gekeken worden naar welke gegevens leesbaar zijn, hoe gemakkelijk het is om deze methoden te omzeilen, ... Er zal gebruikgemaakt worden van OWASP The Mobile Security Testing Guide om zo de verschillende veiligheidsproblemen op te sporen in de applicatie. Verder wordt ook de moeilijkheidsgraad gemeten van iedere implementatie zodat er een besluit kan gemaakt worden welke oplossing bij welke use case het best gebruikt kan worden.

### A.4 Verwachte resultaten

Het resultaat zal bestaan uit verschillende proof of concepts waarbij het gebruik van Android HCE toch op een veilige manier benut kan worden. Op basis van de resultaten zal dan ook beslist kunnen worden welke implementatie het beste is voor welke use case.

### A.5 Verwachte conclusies

Verwacht wordt dat de verschillende implementaties tot beveiligde applicaties leiden waarbij NFC technologie gebruikt kan worden via Android HCE. Alsook dat de applicatie met de biometrische factoren het veiligste zal zijn en hierbij de user experience het beste zal zijn omdat dit ook een groter gevoel van veiligheid geeft.

## Bibliografie

- Alattar, M. (2014). Host-based Card Emulation: development, security, and ecosystem impact analysis.
- Alliance, S. C. (2014). A Smart Card Alliance Mobile & NFC Council White Paper - Host Card Emulation (HCE) 101.
- cloudflare. (g.d.). What is a brute force attack. Verkregen van <https://www.cloudflare.com/learning/bots/brute-force-attack/>
- Cryptomathic. (2014). Protect HCE Mobile Applications with Cryptomathic MASC.
- Lepojevic, B., Pavlovic, B. & Radulovic, A. (2014). Implementing NFC Service Security.
- Maurizio Cavallari, F. T., Luca Adami. (2014). Organisational aspects and anatomy of an attack on NFC/HCE mobile payment systems.
- overflow, S. (g.d.). How to encrypt string values and byte arrays. Verkregen van <https://stackoverflow.com/questions/10562908/encryption-of-strings-works-encryption-of-byte-array-type-does-not-work>