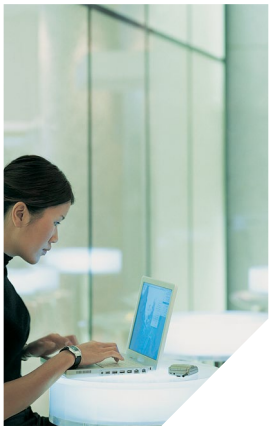
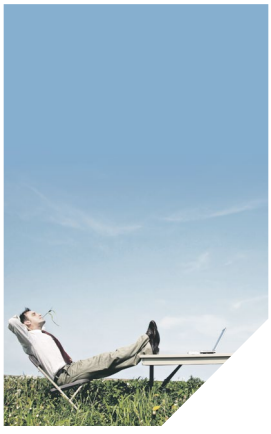
CRYPTOMATHIC

White Paper

# Protect HCE Mobile Applications with Cryptomathic MASC

# 1 Introduction

## 1.1 About HCE and Payment Tokenization

Since the beginning of 2014, some major changes and trends have affected the mobile payment industry, allowing NFC Mobile Payments to be made without the need for a proprietary Secure Element (SE) in the mobile phone. Such changes include:

- Host-Card-Emulation (HCE) is now supported by Google Android mobile devices
- Payment associations announcing support of SE in the Cloud
- More interest in "token" and "tokenization" technologies as a measure to reduce the fraud risks from PAN theft.

As an innovator in the security field, Cryptomathic has developed a cutting-edge solution for mobile payments using HCE and Tokenization.

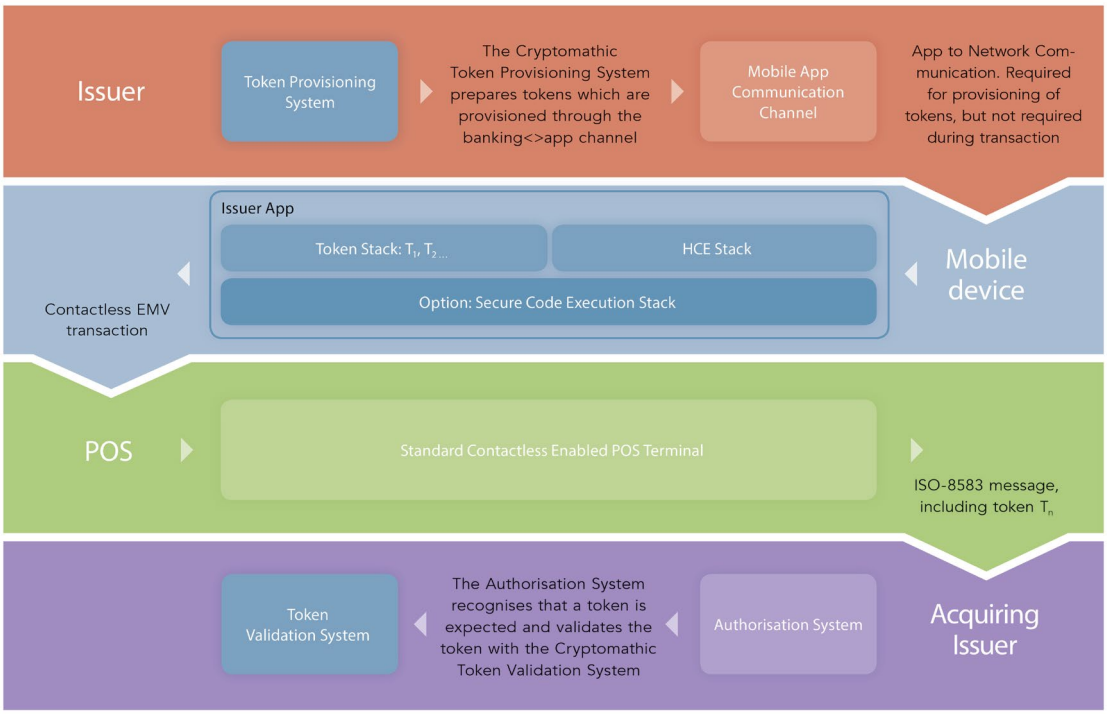
It is based on the use of a "Payment Token" within the EMV transaction. Each payment token can be used only once for a single transaction. These tokens are generated by the back-end and provisioned automatically to the smartphone when requested by the device application. Subject to the configuration, several new tokens may be provisioned and stored/cached locally until used.

## 1.2 Document Scope

Cryptomathic's solution is based on a payment token cached on the smartphone that makes use of HCE. By including risk data in the tokens, forcing the transactions to go online for authorisation[1] and limiting the number of tokens present on the handset at any one time, the solution assists issuers to mitigate the risk of lost/stolen mobile devices. It is also worth noting that mobile phones are usually well guarded by their owners, even more so than cards, where owners typically notice a lost mobile phone within 20 minutes, a lost wallet or purse can take up to 24 hours to be noticed! This results in lower risk levels for lost/stolen smartphones being used in fraudulent payments as the owner is likely to react quickly to a misplaced device.

Mobile Payments using HCE and Tokenization address fraud through a design that allows all aspects of the issuer's fraud management systems to be applied to the transactions, thereby offering reactive measures to attacks. That said, it is common knowledge that an efficient fraud management strategy needs to consist of both reactive and proactive measures.

Efficient fraud management alone is not enough to achieve an adequate level of assurance for issuers. This document describes the additional mobile security protections, based on Cryptomathic's Mobile App Security Core (MASC), that can be used to complement the security of a tokenized HCE solution and offer a proactive defence strategy.




*HCE and Payment Tokenization - Architecture overview*

[1] Fraudsters usually like to cash out quickly. Limiting most contactless payments to low value bearing transactions presents a significant challenge for them to do so.

## 2 Evolving Threat Landscape

### 2.1 Threats for a Contactless Payment App

The threat landscape for mobile security is ever-changing. Let's consider the mobile security threat landscape in the context of contactless NFC payments. Typically, mobile payment application developers need to account for different types of threats and countermeasures, including those referred to in the breakout boxes below.

Even though HCE is currently only available on Android, it is anticipated that it will be ported to other platforms including iOS and Windows Mobile in the near future. It is therefore important to work on a security framework that is not OS dependent to allow for a rapid and agile application management. In addition, the legacy Android key store and iOS KeyChain are by essence an obvious target for attackers who will try to take advantage of their widely documented security vulnerabilities.

#### Reverse Engineering

Hackers use reverse engineering tools in combination with scripting and custom-developed software to exploit the information they gather together with standard OS or protocol weaknesses to launch their attack and cash in. As a consequence, the app developers need to consider hardening the app security design to do as much as possible to make it so that it would require significant investment and innovation for the fraudsters to succeed. There are many ways they can be hindered.

#### Secure Data Storage

One significant benefit of HCE and payment tokens is that it makes the issuer fully independent from the MNOs, thereby solving the complex equation of provisioning Secure Elements to a large retail audience. This, however, implies that other measures are taken to handle the storage of sensitive data and render it cumbersome enough to persuade hackers to fish elsewhere.

#### Mobile Device Binding

When used to conduct payments, it is essential to introduce a strong link between the device used to conduct the transaction and the virtual card account from which the funds will be debited. It is traditionally a challenge for the app developer to find an ID/method to uniquely identify devices and implement a device fingerprinting technique that cannot be reverse engineered easily and ensure that the HCE application can only run on the device to which it was issued.

#### Malware

The mobile device platform has a different security profile to desktop PCs. Mobile operating systems are typically closed so all installed software must be approved by the OS vendor, making them more secure. Nevertheless, malware is present, and it is important to protect against it, but the picture is not as bad as one might think as most malware operates within granted permissions.

#### Secure Communication/TLS Endpoint

It is common to target the authentication mechanism towards the back-end as it is the link to the token provisioning. One can see the concerns caused by **Heartbleed**.

## 2.2 Protection Target: The Need to be Agile

In the chip-card centric payment ecosystem, the payment schemes have defined protection targets to evaluate and certify chip card products. Once issued, the card configuration is locked and post-issuance operations are very limited. In terms of lifespan, the card products are certified for a period of approx. 3 years as illustrated in the list of VISA approved dual interface card products[2].

The move towards a smartphone centric payment solution creates a great disruption in the security management policy, as the old formulas may no longer apply. Payment schemes and issuers have little control on how the devices are used and updated. New devices appear every month, device OSs are frequently updated and issuers keep on introducing new features to their banking applications.

The security profile of the mobile as a protection target changes more rapidly than a conventional payment card.

Typical lifespan of critical components in question:

| Critical Component | Lifespan |
|---|---|
| Conventional payment card | 3-4 years |
| Mobile devices | 18-24 months |
| Mobile operating systems | 6-12 months |
| Mobile banking applications | 1-3 months |

It is worth noting that the design of Cryptomathic's solution shifts the storage of long-term payment card keys to the back-end token generation server, thereby significantly reducing the sensitivity of the data stored on the client side by the mobile banking application. However, it is important to safeguard your app with security mechanisms that are in constant evolution, i.e. changing at the same pace as the mobile banking application itself.

## 2.3 Protection Profile: A New Paradigm in the Mobile Industry

The protection targets for a conventional payment card are not threatened in the same way as they are for a mobile device based payment card. With HCE, the payment card is embedded in an open environment where it coexists with other untrusted applications not under the issuer´s control.

This opens a new security paradigm for issuers of payment cards, which are used to issue payment applications on approved card products that have undergone a thorough evaluation process and which evolve in a controlled environment. This change greatly impacts the risk management policy of the entire ecosystem and architecture.

The tokenization architecture is designed primarily to limit the value of the mobile device in a fraud setting. Specifically aimed at lost/stolen and malware fraud, a new set of protection targets are identified which are different to those found in a conventional payment card.

| Protection Target | Contactless Card | Mobile Virtual Card |
|---|---|---|
| Card keys | Yes | Yes |
| PAN/Card data structures | Yes | Yes |
| Device ID | No | Yes |
| Application ID | No | Yes |
| Tokens | No | Yes |
| Passcode/PIN (on entry) | Yes | Yes |

*Protection Target: Contactless Card vs Virtual Card*

We can see that there is clearly a new protection profile target, and banks need to work hard to protect these elements.

Despite that we are dealing with a more open environment in comparison to a conventional payment card, the ability to change virtual cards instantly at very little expense makes the risk profile much lower even though we are not using physical smart cards.

## 2.4 Creating Consumer Confidence

Consumers' experience of fraud can undermine confidence in security and in larger cases will lead to reputational damage. Customers' attitude towards fraud and accepting fraud protection measures is changing. In general customers are experienced in appreciating and accepting interaction with fraud/risk management systems and will tolerate some disruption.

Times have moved on. The changes caused by the evolving threat landscape have resulted in an environment where banks who wish to provide mobile contactless payments can do so with a software based solution, which is sensible and practical.

[2] http://technologypartner.visa.com/Download.aspx?id=247

# 3 Cryptomathic Mobile App Security Core

Mobile App Security Core (MASC) delivers a foundation to enhance app security and support future technologies, without the need to expend extended time and costs integrating or redeveloping applications to support increasing security requirements. Cryptomathic´s evolutionary security design (see diagram below) ensures that mobile apps and their security framework remain future-proof and resistant against attacks.

Cryptomathic's Mobile App Secure Core (MASC) has been designed with two primary goals in mind:

- Resist theft and abuse of the protection targets
- Disrupt the business model of fraudsters attempting to exploit the protection targets long-term

We invite the reader interested in learning more to read Cryptomathic's white paper **Secure Mobile Transactions – Fact or Fiction?**

## 3.1 Protection Measures

### 3.1.1 Anti-Reverse Engineering

A number of prevention measures to resist reverse engineering of the mobile application are provided:

- Obfuscation of code
- Obfuscation of communications
- Anti-debugging + jailbreak detection

### 3.1.2 Device Fingerprinting

The mobile operating system APIs are used to generate device fingerprints. The device fingerprint is used to support enrolment and login workflows and helps to ensure that the app itself will only run on the mobile device to which it was provisioned. This also ensures that apps cannot either run on smartphone emulators.

### 3.1.3 TLS Endpoint Security

The certificate store is hardened to store only a bank issued root certificate. It implements pinned server certificate verification and has client certificate storage.

CRYPTOMATHIC MOBILE APP SECURITY CORE - FUNCTIONAL OVERVIEW

### 3.1.4 Anti-Malware

Responsible for detecting device rooting/jail-breaking and the presence of malware. It uses software security mechanisms to detect and react to device compromise. It also reacts to the presence of known malware, which may attempt to interfere with the mobile application and harvest data.

### 3.1.5 Secure Storage

Secure Storage protection measures are designed with two objectives:

- Make it difficult to separate data from the application
- Make it difficult to migrate, or copy, data to other devices

All application data is dispersed to make it hard to identify, separate and migrate from the application. The secure storage protections are extremely creative about where it positions the data. Cryptomathic's design uses as many operating system services as possible. On top of this, there are controls in place to prohibit fraudsters from monitoring where the data is accessed; the application is designed to be cautious about the circumstances and times when the application stores and retrieves the data, because it never knows who might be watching.

| | Android Key Store (version 4.4 and later) | MASC Secure Storage |
|---|---|---|
| SCOPE | Key storage and usage | Sensitive data storage including key data and other sensitive credentials<br>The protection profile is much larger. |
| AVAILABILITY OF THE APIs | Standard JCE APIs publicly available in Android 4.4 SDK | Proprietary APIs made available to MASC clients only<br>This makes the education investment much larger for potential attackers. Given that the API is platform independent, it doesn't increase the work for a developer. |
| STORAGE MEDIA | Hardware-backed (Nexus 4, Nexus 7, etc) or software only (Galaxy Nexus, etc) | Software and OS provided hardware can be supported<br>Cryptomathic is also working on leveraging TEE when it is made widely available. |
| STORAGE LOCATION | Static file location easy to spot for an attacker | Dynamic and obfuscated data storage<br>This makes an attacker's job extremely cumbersome and weakens their potential ROI.<br>The attacker will first fail to spot where the relevant data is stored. Secondly, they will not able to find a reverse engineering tool to provide valuable input (Cryptomathic test the majority of tools and engage third parties to pen test as an additional measure). If the attacker were to collect enough information to construct an attack, the attack would be unreliable since the secure storage protection measures change at regular random intervals as Cryptomathic strive to continually evolve MASC to keep one step ahead of the fraudsters. |
| ACCESS | Multi-user capable (using app alias) | Sentinels are implemented to make it very resource intensive for an attacker to be able to retrieve the sensitive data |
| UNDERLYING CRYPTO | Standard Android JCE provider | Cryptomathic PrimeInk proprietary interface (ie not widely distributed) |
| SECURITY VULNERABILITIES | Widely known and advertised | Likely to be kept within closed fraud circles if found. Vulnerabilities are likely to be of little relevance to the wider hacker community. |
| MAINTENANCE | Patches dependent on Google and hadnset vendor | Agile to remain ahead of the game |

*Comparison Between Legacy and MASC Data Storage*

#### 3.1.6 Crypto Engine

There are significant advantages in using a proprietary crypto interface as opposed to an open source crypto service, where behaviour and weaknesses are widely known and referenced by the industry[3].

Based on Cryptomathic's crypto toolkit, PrimeInk, this secure crypto engine provides the AES, RSA, and SHA cryptographic primitives that support the secure operation of the mobile application.

### 3.2 Which Protection Measures Defend Which Protection Targets?

Cryptomathic MASC offers the industry´s most comprehensive set of protection measures to safeguard any mobile payment/banking app against different types of attacks.

The matrix below illustrates how MASC security measures/modules are used to mitigate the risk of attacks to potential targets inside the HCE app.

### 3.3 Long Term Protection - Evolutionary Security Design

Cryptomathic MASC is designed to protect against the ever-changing threat landscape, with protection measures adequate for today and in the future.

Cryptomathic stays one step ahead of the attackers by offering:

- Periodic up-dates of obfuscation code and obfuscation of communications
- Regular updates to the malware database used to identify malware and jailbreak definitions
- Techniques for secure storage and masking of data access are both revised and amended regularly

## 4 Introducing MASC to the Banking App

Cryptomathic MASC is a platform independent security layer featuring different sub-modules, each consisting of thin wrappers implemented in Java for Android, Objective C for iOS and later on .NET for Windows Mobile.

This extension can be integrated with relatively little effort into the HCE and tokenization app SDK used to build your banking app. The MASC deliverables are:

- Client side: The SDK is enhanced with additional Java classes which interface with the different sub-modules of MASC (Anti-Reverse Engineering, Device Fingerprinting, Secure Storage, TLS Endpoint

| Measure \ Target | Card Keys | PAN/Data | Device ID | App ID | Tokens | Passcode |
|---|---|---|---|---|---|---|
| Anti-Reverse Engineering | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Device Fingerprinting | | | ✓ | ✓ | | |
| Secure Storage | ✓ | ✓ | | | ✓ | |
| TLS Endpoint Security | | | ✓ | ✓ | | ✓ |
| Antimalware | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Crypto Engine | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

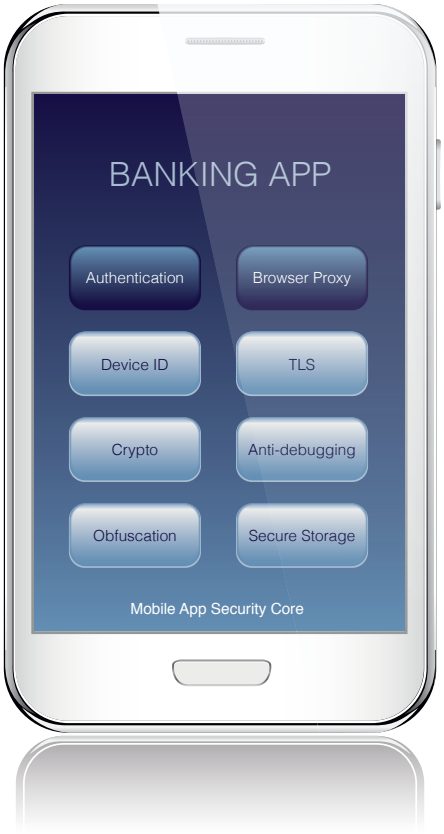
*Targets vs Security Measures for HCE Apps*

[3] http://www.developereconomics.com/app-security-101-list-top-10-vulnerabilities/
http://openjdk.java.net/groups/security/
http://www.cvedetails.com/vulnerability-list/vendor_id-49/product_id-15556/Apple-Iphone-Os.html
http://www.csoonline.com/article/2134479/mobile-security/security-weakness-found-in-ios-7.html

Security, Anti-malware, Crypto Engine). A developer can simply load these classes into the development environment. Regular updates will be provided, but the external commands remain unchanged. Control over the user interface is retained by the developer.

- Back-end: The existing authentication back-end needs to be extended to integrate with existing systems and processes. We recommend that you speak to Cryptomathic for more information on this thread.

### MASC Components




## 5 Conclusion: Make the Fraudsters Look Elsewhere

The comprehensive set of protection measures deployed in an interlocking manner combined with their evolutionary and adaptive security design makes MASC an ideal enhancement to strengthen the HCE solution design and lower the risks inherent in a mobile app deployment project. The framework delivers an effective, multi-layered defence strategy, which places a significant up-front investment barrier to fraudsters with little chance of success. This ultimately persuades them to search elsewhere.

Carefully timed updates to security mechanisms, made popular by the Pay TV industry, who up-date their protection profile the day before a big match, means it's harder for the fraudsters to sell hacks (particularly malware infection) since negotiating the price for an unreliable hack is hard.

We invite the reader interested in learning more to contact us so that we can expand on our secure techniques or the test methodology adopted to defeat attackers, as well as our patent pending security features.

Email us at: **technical_enquiry@cryptomathic.com**
or find our local offices:
**www.cryptomathic.com/contact/cryptomathic-offices**

### Disclaimer



## ABOUT CRYPTOMATHIC

Cryptomathic is one of the world's leading providers of security solutions to businesses across a wide range of industry sectors, including finance, smart card, digital rights management and government. With more than 25 years' experience, Cryptomathic provides customers with systems for e-banking, PKI initiatives, card personalization, ePassport, card issuing and advanced key management utilizing best-of-breed security software and services. Cryptomathic prides itself on its strong technical expertise and unique market knowledge. Together with an established network of partners, Cryptomathic assists companies around the world with building security from requirement specification to implementation and delivery.