

ACE THE FUTURE OF MOBILE CARD PAYMENTS

Whitepaper by Joachim Samuelsson, Crunchfish

Crunchfish introduces App-integrated Card Emulation (ACE) offering the security of Mobile Card Emulation (MCE) without sacrificing the scalability provided by tokenized card payments with Host-based Card Emulation (HCE).

Card payments is a common payment method that banks offer their customers. To enable card payments from a mobile device in a secure way that is scalable to all customers have proven to be difficult. It is either limited to users of select mobile devices or operators (MCE) or alternatively only available for e-commerce or terminal purchases that requires online clearing. The ground-breaking mobile technology that Crunchfish has developed for offline payments is also applicable for mobile card payments.

This whitepaper by Crunchfish introduces App-integrated Card Emulation (ACE) as a novel way of card emulation. The initial section relates ACE to other ways of **Card Emulation (CE)** – Mobile Card Emulation (MCE) using a hardware-based Secure Element (SE) and Host-based Card Emulation (HCE) with tokenized payments. The second section discusses **Implementation Architectures** using hardware-based as well as software-based protection. The third section discusses **Data Integrity** during runtime operation, for stored data at rest and for data in transit. The conclusion includes a **comparison table of MCE, HCE and ACE** that summarizes the points raised in the three sections of this whitepaper.

Card Emulation

An Integrated Circuit Card (ICC) is a physical card that banks issues to their customers for EMV (Europay, MasterCard, VISA) card payments. It is secure and scalable solution that the bank may offer it to any of their customers. The bank controls the ICC as issuer of the payment application to their customer. An ICC have a secure element (SE) where the payment applet executes securely within an isolated environment. Securing a mobile device such that it can make card payments is a well-known challenge in the payment industry. The two emulation options currently available MCE and HCE have significant drawbacks. The secure hardware-

based MCE is not a scalable solution as the user base is limited to select mobile devices or network operators whereas the scalable software-based HCE is not secure unless the use case is limited to only purchases with online clearing. ACE enables, on the other hand, a solution that is uniquely secure and scalable.

Mobile Card Emulation (MCE) in TEE / embedded SE is a secure but not scalable solution

Securing a mobile device such that it can make card payments is a well-known challenge in the payment industry. An ICC have a secure element (SE) where the payment applet executes securely within an isolated environment. Although some mobile devices also come with such Trusted Execution Environments (TEE) or embedded Secure Elements (eSE) emulating a card by Mobile Card Emulation (MCE) is difficult to deploy a payment application to the wide range of handset models or mobile network operators. This is because the TEE or eSE is not controlled by the payment application itself, but instead by the mobile device manufacturer (OEM) or alternatively the mobile network operator (MNO). Whereas it is perfectly fine to pilot MCE on a specific handset model or in partnership with an network operator, MCE is simply not scalable for mass deployment as pointed out in [Thales' whitepaper on Host-based Card Emulation](#). For the very same reason, solutions using TEEs or eSE for offline payments do not scale either as argued in the [whitepaper Ensuring trust in scalable offline solutions by Lipis Advisors and Crunchfish](#) from November 2023.

Host-based Card Emulation (HCE) with SDK in REE is a scalable but not secure solution

To overcome the scalability issues with Mobile Card Emulation, the leading card networks VISA and MasterCard, have suggested an alternative approach to implement mobile card payments based on tokenization and Host-based Card Emulation (HCE). With HCE, the payment credentials are not stored in a TEE or eSE on a mobile device, but instead in a shared repository on a host server at the issuer's data center or in private cloud. To enable contactless transactions, limited tokenized credentials are delivered to the phone in advance. As tokens are handled in the unsecure but flexible Rich Execution Environment (REE) on the mobile device they are exposed to various attacks. Tokenized HCE payment is therefore typically limited to online purchases.



Achieving total security is impossible for any implementation, but integrating security measures make it harder for hackers to infiltrate applications and obtain sensitive data. Security protections typically involve some of the following techniques: code obfuscation, root-detection, anti-tampering, code-integrity detection, anti-debug, anti-instrumentation, hook detection, device binding and white-box cryptography. Unfortunately, when these protection mechanisms are implemented in the REE, an attacker will be able to break the security. [It is only a matter of time as discussed in this whitepaper](#) from 2018. As implementing offline payments securely in mobile devices poses similar challenges as implementing tokenized HCE payments it is beneficial to refer to the [whitepaper Offline payments in smartphones by Crunchfish](#) from February 2024 for a discussion on risks and mitigations for assets handled on the mobile device.

App-integrated Card Emulation (ACE) in TA in virtual SE is a scalable and secure solution

Crunchfish is offering a patent-pending approach to mobile card payments by emulating payment credentials within the payment app itself. An App-integrated Card Emulation Trusted Application (ACE TA) executing in a certified virtual Secure Element (vSE) offers an isolated runtime for cryptographic data operations and Crunchfish proprietary technology protects data at rest with secure storage against roll-back attacks of tokenized payment credentials and risk limits. Scalability is achieved as it is distributed, upgraded, and maintained as an integral part of the payment app itself. ACE TA provides the required security, without sacrificing scalability, which is a unique and major implementation advantage of mobile card payments.

ACE TA provides a much higher level of security than the commonly available software-based application protection where the protection mechanisms are all implemented in the unsecure REE. This is in sharp contrast to the security provided by the vSE where the ACE TA as well as the protection mechanisms are all executing within the vSE. This means that attackers cannot tamper with the protection mechanisms or ACE TA without first breaking the security of the vSE itself.

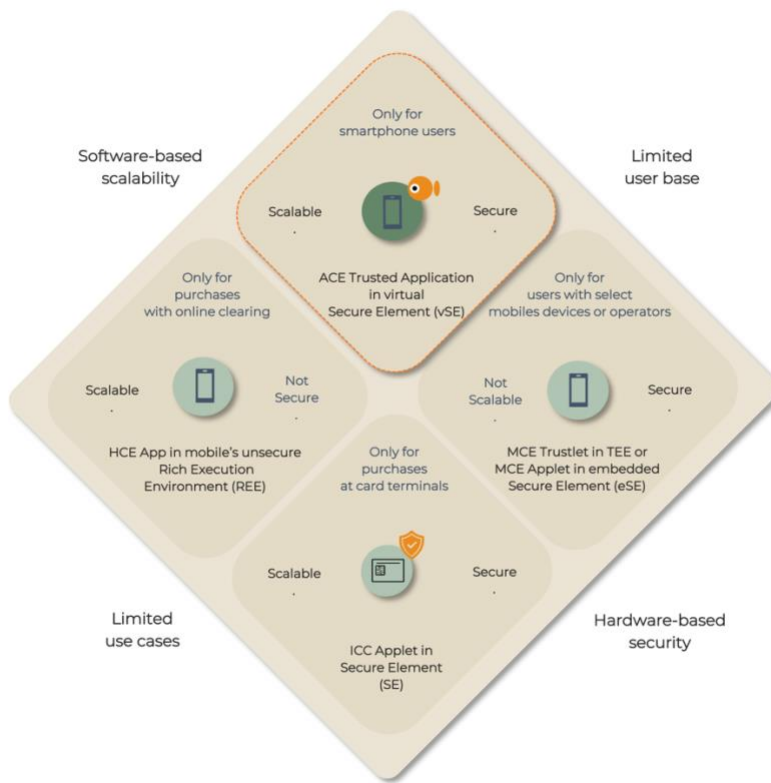


Figure: Overview of implementation architectures and their limitations for mobile card payments.

As noted in a [blog post by Nordic Semiconductor from 2017](#) Apple Pay uses a SE on the iOS mobile device and HCE in a hybrid system. By including the SE on the iPhone and using HCE technology for payments credentials in the cloud, Apple has built a system that mitigates the risks of each technology. Samsung provides a similar hybrid on their Galaxy devices for Samsung Pay.

Apple and Samsung are able to provide such hybrids as they are a payment application provider that can control the security arrangement on their mobile devices as they are also the device manufacturer. On the device fragmented Android market this increased level of security is not possible for Google Pay using hardware-based security. With ACE TA executing within a vSE, on the other hand, this hybrid becomes possible also for Google Pay. Furthermore, ACE TA provides mobile device security that is homogenous across all smartphone devices without any dependencies on mobile devices or OS.

Implementation Architectures

Mobile card payment applications with HCE SDKs are typically executing in the unsecure software-based Rich Execution Environments (REEs), which provide high level of programmability and scalability, but have security issues as the tokenized payment credentials, cryptographic key data, and other EMV assets, such as risk limitations, are exposed to attacks. The use case of tokenized HCE payments is therefore limited to purchases when online clearing is available. To relax this limitation HCE payments must be implemented within a Tamper Resistant Element (TRE) offering an isolated runtime and secure storage for tokenized credentials, cryptographic key data and other EMV assets.

A key consideration for establishing the required additional security with maintained scalability is how this TRE is implemented. It could either be implemented in hardware or software. A TRE implemented in hardware is a TEE or an eSE, whereas a software-based TRE is a vSE.

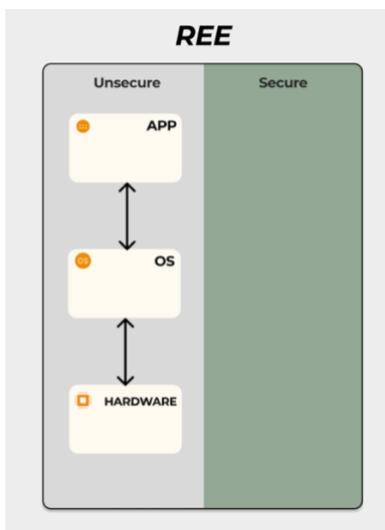


Figure: The Rich Execution Environment provides only a limited security for mobile card payments and the use cases are therefore typically limited to online purchases only.

Hardware-based TEE / eSE

A hardware-based TEE or eSE could either be a device-integrated TRE in the System-on-Chip (SoC) or a standalone SE implemented as an eSE or alternatively as SIM or eSIM. The OEM controls the device-integrated TRE and the eSE as a standalone SE and the MNO controls the SIM or eSIM.

Device-integrated TRE

Device-integrated TREs integrated into the SoC, such as Android Keystore or iOS Keychain, are widely available on smartphones. However, they protect only the cryptographic key data and are therefore not sufficiently secure for mobile card payments as they do not offer an isolated runtime for tokenized payment credentials and other EMV assets. It's easy to attack and bypass the security.

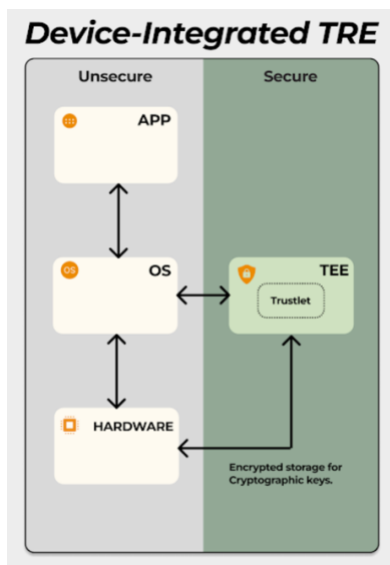


Figure: A device-integrated TRE that only handles cryptographic keys such as Android Keystore or iOS Keychain is not secure enough for mobile card payments as tokenized credentials and other EMV assets are not protected by an isolated runtime. Trustlets on a device-integrated TEE that are able to handle tokenized credentials and other EMV assets are hard to deploy in practice and have also known security issues.

Another common way that allows mobile payment applications to increase their security is to use the device-integrated Trusted Execution Environment (TEE) typically delivered by ARM

Trustzone. Although it is technically possible to write a trustlet, that is a trusted applet, that can handle tokenized payment credentials, cryptographic key data, and other EMV assets on a device-integrated TEE, the device fragmentation in the market makes it difficult to implement in practice. Several challenges emerge that make trustlets on device-integrated TEEs hard to deploy in practice:

- Mobile devices use different TEEs that come with their specific operating systems. This means that it would be necessary to implement multiple versions of trustlets to execute on the variety of TEEs in the market.
- The next and harder challenge is how to provision the trustlets on the market. Provisioning a trustlet on a device-integrated TEE is not suitable for third parties because the distribution ecosystem is not mature. It would be necessary to partner with multiple OEMs to get the trustlet loaded onto many devices to achieve widespread availability in the market.
- The same challenge applies to the maintenance of the trustlet as no standard post-issuance method exists for TEEs.
- Furthermore, there is currently no support for implementing trustlets on a device-integrated TEE on iOS, which also limits the market penetration.

In addition, there are also several known attacks on applications using TEEs [as suggested by this IEEE article](#) from 2020. With significant scalability as well as security issues device-integrated TREs are questionable for mobile card payments.

Standalone TRE

Standalone TREs are another type of hardware-based TREs that can host an applet with an isolated runtime and secure storage for tokenized payment credentials, cryptographic key data and other assets. These are implemented in the mobile device either as an eSE or on a SIM or eSIM. However, provisioning an applet on a mobile device on a standalone TRE poses similar challenges as for device-integrated TREs.

To achieve widespread market availability for a payment application with support for mobile card payments, there is a need for the payment application provider to partner with a sufficiently large number of device manufacturers or possibly mobile operators (if the standalone TRE is implemented on a SIM). There are also additional challenges in manufacturing, distributing, and maintaining the standalone TREs on mobile devices. This creates significant scalability issues to bring mobile card payment applications on hardware-based standalone TREs to market.

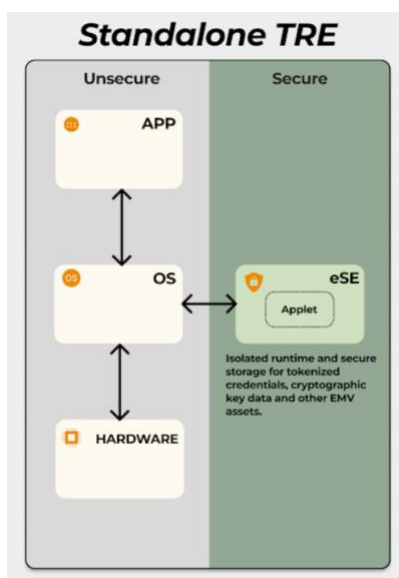


Figure: A standalone TRE that provides an isolated runtime and secure storage for tokenized payment credentials, cryptographic key data and other EMV assets makes mobile card payments secure. However, it is hard to deploy applets on hardware-based, standalone TREs in mobile devices in practice as these embedded SEs / SIMs are controlled by the OEM or alternatively by the MNO. Furthermore, there is limited mobile OS support for interacting with such standalone TRE

Software-based TREs

Software-based TREs on smartphones are based on tamper-resistant, virtual machines, also known as vSE. They offer the required security for mobile card payments, without sacrificing the scalability of applications implemented in the Rich Execution Environment. They provide the required isolation by a virtualized isolated runtime and offer secure encrypted storage for tokenized credentials, cryptographic key data and other assets. Attackers cannot tamper with the TA with ACE on vSE or bypass the protection mechanisms without first breaking the vSE itself.

Weaker software-based protection solutions are implemented in the REE and rely on a combination of code obfuscation and white-box cryptography. This is not sufficiently secure for mobile card payments. The primary weakness of such solutions is that the cryptography and runtime protection mechanisms execute natively in the unsecure mobile OS, which attackers can bypass quite easily.

App-integrated TRE

Software-based TREs are app-integrated vSE. This means that the ACE TA executes within a vSE that is integrated within the payment app. As the app-integrated TRE is an integral part of the payment app, there is no trust gap, and provides a consistent level of security for mobile card payments, independent of the mobile-OS and the mobile device on which it is running.

Another key benefit is that the ACE TA can run securely even on rooted or jailbroken devices as it executes within the secure, software-based TRE that is not affected by an attack on the device by rooting or jailbreaking. The ACE TA can easily be integrated with a payment application with the same flexibility as writing a regular app. It can be deployed, updated and maintained on any smartphone device using the regular distribution ecosystem for apps, i.e. Apple App Store and Google Play. Altogether, this makes the solution uniquely secure and scalable for mobile card payments.

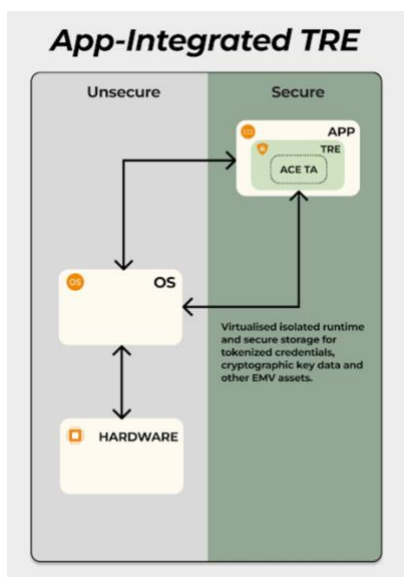


Figure: An App-integrated TRE provides a virtual secure element in which the ACE TA can be implemented to support secure and scalable mobile card payments. It provides an isolated runtime for data in operation by virtualization and Crunchfish proprietary technology provides a secure storage for tokenized payment credentials, cryptographic key data and other EMV assets at rest on the mobile device. It is secure and certainly scalable, as the ACE TA is deployed, updated and maintained with the payment app via Apple's App Stores or Google Play.

A common belief is that [hardware-based TREs are always more secure than software-based TREs](#), because of the clarity of security boundaries. However, due to the separation between the payment app and the hardware-based TRE, there is an inevitable gap in the chain of trust between the two communicating endpoints, the unsecure payment application on one side and the hardware-based TRE in the other end. There are attacks that expose this trust gap by replacing an endpoint with a malicious one or tampering with an endpoint and modifying the behavior during runtime. As the TRE does not have full visibility of the payment app and the mobile OS, it cannot determine the identity of the app or whether the app has been tampered with, and must therefore blindly trust the unsecure OS and the payment app.

Data integrity

Mobile application protection is hard. This section looks at what security is required from a mobile application client perspective. In addition to tamper resistance, the trusted environment must deliver an isolated runtime to protect the integrity of the data during runtime as well as a secure storage for data at rest. This is mainly related to providing protection against rollback attacks.

The standard ways of protecting a payment app and its HCE SDK are insufficient to securely handle tokenized payment credentials, cryptographic key data and other EMV assets in a mobile device. The use case has therefore been limited to purchases involving online clearing. The software-based protection mechanisms come in different shapes and forms, but typically use these types of protections includes code obfuscation, runtime application self-protection (RASP) or app shielding, and white-box cryptography. These protection mechanisms are better than nothing but do not hold back attackers for long.

Securing data during the runtime operation

A payment application requires protection against direct attacks on the application itself and against indirect attacks that may manifest as manipulation of the mobile OS or exploitation of hardware vulnerabilities. To achieve this level of protection requires an isolated runtime in a

Tamper Resistant Element (TRE).

[An article by Fime on Global Platform's webpage gives an overview of such software-based app protection mechanisms](#) explains that *"Payment tokenization converts sensitive payment information into a unique token, which has a limited number of predefined circumstances under which it can be unlocked, rendering the data useless to hackers. Finally, while the use of hardware protection is not required or standard for HCE deployments, some implementations are now utilizing Global Platform-based Trusted Execution Environment (TEE) technologies to add additional security. They provide secure, isolated environments in which to store the "trusted application" itself, its sensitive code, and cryptographic keys."*

[Crunchfish has in April 2024 applied for a patent for ACE](#) protecting a TA solely executing within a secure virtual machine that is embedded within a payment application, the secure virtual machine having a virtualized operating system and providing an isolated runtime for the trusted application to securely handle tokenized payer credentials and cryptographic key data. This novel card emulation is just as scalable as standard HCE implementations in REE as it is also integrated, distributed, and maintained with the payment app itself. ACE TA is also secure since it is executing within a vSE that provides similar protection provided by a trustlet in a TEE or an applet executing in a SE, but with all benefits of being implemented in software.

Crunchfish has pioneered offline payments since 2020 and [decided early to implement its Digital Cash TA within V-OS](#) to be able to deliver an offline payment solution that is both secure and scalable. The same approach can be used for mobile card payments with the only difference that the ACE TA implements the EMV security protocol instead. [V-OS is a patented, common criteria EAL3+ certified and FIPS 140-2 Level 3 validated vSE](#) from the Singaporean mobile security company [V-key](#). Crunchfish released [an interview in 2021 with V-Key's CEO and co-founder regarding Crunchfish Digital Cash Trusted Application for offline payments](#) executing in their Tamper Resistant Virtual Secure Element V-OS. In 2022, Crunchfish invited [V-Key to present their technology at Crunchfish's webinar series](#) Survival of the fittest.

Securing stored data at rest

In addition to protecting payment credentials and other EMV assets during cryptographic operations on the mobile device it also crucial that data is protected when it is at rest as well. MCE using an eSE or SIM offers secure storage of limited size for data within the SE itself whereas a TEE implementation stores data encrypted on the mobile device's filesystem, which also is the case for ACE TA. Encryption in an isolated runtime restricts access to the stored data, but as it is stored on the file system they are exposed to rollback attacks on rooted / jailbroken devices.

Implementing rollback protection is a must to mitigate tampering with the data at rest on a mobile device. A rollback attack, in short, is a way to restore a previous state on the mobile device. In the context of offline payments this could be the balance of an offline wallet, making it possible to double-spend. For mobile card payments this could be in attack on the payment credentials or risk limits.

VISA discusses TEE rollback protection in their [technical paper describing an offline payment system from December 2020](#) which in turn refers to [Western Digital's paper from 2017](#). It relies on *"a replay-protected memory block (RPMB) partition on an eMMC storage (e.g., the phone's persistent storage) is used to store TA's data securely. Any data written on the RPMB is protected against man-in-the-middle replay/rollback attacks using a monotonically-increasing counter (MIC) maintained by a dedicated hardware, known as the RPMB engine. The engine increments the MIC after every write to the RPMB and uses message authentication codes (MACs) to verify the validity of the write command by checking that (1) the counter was increased, and (2) the MAC that was sent by the sender (e.g., the TA) is identical to the MAC that the RPMB engine generated using its latest value of MIC. Finally, every read from RPMB is MAC-checked by the reader (e.g., the TA) using the latest value of MIC maintained by the reader."*

[Crunchfish has a patent-pending for rollback protection using asymmetric cryptography with priority from March 2023](#). Similar to the rollback protection described by VISA and Western Digital it makes use of an anchor point that an attacker cannot tamper with in a rollback attack that may be preferably available on the mobile device in an RPMB or alternatively made available in the backend. The main difference between Crunchfish approach and VISA / Western Digital is

that trust between the anchor point and the Crunchfish is establishing trust by asymmetric cryptography rather than symmetric cryptography with shared secrets. In practice, asymmetric cryptography is much preferred as trust should be implemented between an RPMB anchor point controlled by the OEM and multiple TAs in the market.

The proprietary rollback protection uses a combination of the system state together with ACE TA and payment app data/metadata. The ACE TA validates this data on every operation and updates the rollback data with the new state during and after the ACE TA operation has executed. If the ACE TA detects that a rollback to a previous state it locks the ACE TA. In case an attacker is able to snapshot the entire state of the mobile device including system clocks, working memory, processes, in order to restore it an attacker must reboot the mobile device which can be detected by Crunchfish proprietary rollback protection system. As a conservative measure, the ACE TA may be locked and require an online sync operation with the issuer's backend before it is accessible again. Rollback attacks may also be detected by backend processing. Transactions with embedded monotonically increasing payment IDs will be uploaded during the sync operation and by reconciling with transactions uploaded by the payees it is possible to reveal whether a mobile device has been exposed to a rollback attack.

If a vulnerability is found in the ACE TA or in the vSE it can be patched with a regular software update together with the app. This is advantage of using ACE compared to hardware-based TEE or eSE. The issuer backend may reject request to re-key the certificates if the payment applications are not running the latest HCE TA version. Using short-lived certificates ensure that payment applications that are not updated will not work as the HCE TA certificates expire.

Securing data in transit

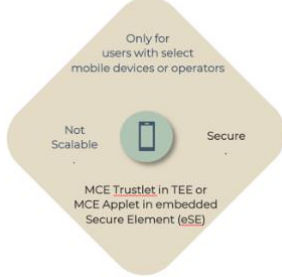
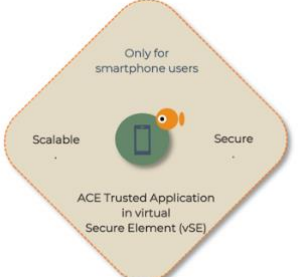
The EMVCo security protocol is secure for ICC, MCE and HCE payments assuming either data integrity during runtime operation and at rest, or alternatively limit the use cases for HCE payments to purchases with online clearing. Crunchfish ACE TA must implement the EMVCo security protocol to interact seamlessly with the card terminals and issuer's backend systems. It is a hybrid that can ensure data integrity during runtime operation and at rest and supports tokenized HCE payments for any user without mobile device or OS dependencies.

Conclusion

Mobile card payments require higher security than what is offered by the standard Rich Execution Environment on smartphones. Mobile Card Emulation using a TEE or eSE on the mobile device offer higher security but lacks scalability as the user base is limited to select OEMs or MNOs. The higher level of security can instead be achieved by implementing an App-integrated Card Emulation Trusted Application (ACE TA) protected by a virtual Secure Element (vSE) that provides a virtual isolated runtime and secure storage for tokenized credentials, cryptographic key data and other EMV assets, e.g. risk rules. Crunchfish is pioneering an App-integrated Card Emulation Trusted Application protected by a virtual Secure Element. As far as Crunchfish is aware, this is the only mobile card payment solution that is secure as well as scalable in the market without any OS, OEM, or MNO dependency.

For more information on how Crunchfish provides both security and scalability for offline payments please refer to these two whitepapers; [“Ensuring trust in scalable offline solutions”](#) from November 2023 in the series [“Enabling offline payments in an online world”](#) offers a discussion on the scalability advantages of using a software-based, virtual SE instead of a hardware-based SE and the whitepaper [“Offline payments for smartphones”](#) from February 2024 provides a discussion about security risks with REE implementations and how to mitigate risks in an implementation in a virtual SE. Although written in the context for offline payments, the insights in these whitepapers are also applicable for mobile card payments.

A follow-up whitepaper is planned that will project opportunities to extend the EMV security protocol with innovations that Crunchfish has originally developed for offline payments that would enrich the EMV payment rail with new features, such as offline payments, privacy, interoperability with other schemes, and quantum-safe offline payments.

Card Emulation	Mobile CE (MCE)	Host-based CE (HCE)	App-integrated CE (ACE)
Overview			
Description	Mobile Card Emulation (MCE) executing in a TEE / eSE is as secure as ICC but not a scalable solution. TEE / eSE is not available on all mobile devices and the TEE / eSE is controlled by the device manufacturer (OEM) or mobile network operator (MNO). No limitation on use cases. Apple Pay and Samsung Pay use a TEE to deliver more secure HCE payments.	Mobile card payments using Host-based Card Emulation (HCE) executing with a HCE SDK in the unsecure Rich Execution Environment (REE). A scalable solution without any dependencies on device manufacturers (OEM) or mobile network operators (MNO). Use cases are limited to e-com or terminal purchases with online clearance.	App-integrated Card Emulation (ACE) executing in a Trusted Application (TA) within a virtual Secure Element (vSE) A secure and scalable solution that may be integrated in payment apps in any smartphone without dependencies on device manufacturers (OEM) or mobile network operators (MNO). No limitation on use cases.
Security environment	TEE / eSE in mobile device	Repository at issuer or cloud	vSE in payment app
CE application	Trustlet in TEE or applet in eSE	HCE SDK in REE	Trusted Application (TA) in vSE
Use case limitations	No limitations	Purchases with online clearing	No limitations
User base limitations	Select OEMs or MNOs only	Smartphone users	Smartphone users
Protection mechanisms	Hardware-based in TEE / eSE	Software-based in REE	Software-based within vSE
Secure on device	Yes	No	Yes
Scalable in market	No	Yes	Yes

Card Emulation	Mobile CE (MCE)	Host-based CE (HCE)	App-integrated CE (ACE)
Implementation architecture			
Tamper Resistant Element	Standalone by hardware	None	App-integrated by software
App protection	No	Yes, by software for app or SDK	Yes, by virtual SE within app
Implementation cost	High	Low	Medium
Provisioning	Hard as TEE / eSE is controlled by OEM or MNO	Easy provisioning together with payment app	Easy provisioning together with payment app
Maintenance	Hard to upgrade firmware	Easy to upgrade with app	Easy to upgrade with app
Trust gap issues	Yes	Yes	No
Rooting / jailbreaking issues	Risk of malware attacks	High risk with many attack vectors	Low risk as ACE on rooted / jailbroken are supported
Data integrity	High	Low, limited use case to purchases with online clearing	High
Isolated runtime	Yes	No	Yes, by virtualization
Rollback protection	Limited storage in SE. Rollback protection typically not provided	No, rollback protection typically not provided	Yes, by Crunchfish's proprietary rollback protection mechanisms
EMVCo security protocol	Yes	Yes	Yes

Table: Comparison table between various modes of card emulation for mobile card payments.