2024-02-01

OFFLINE PAYMENTS FOR SMARTPHONES

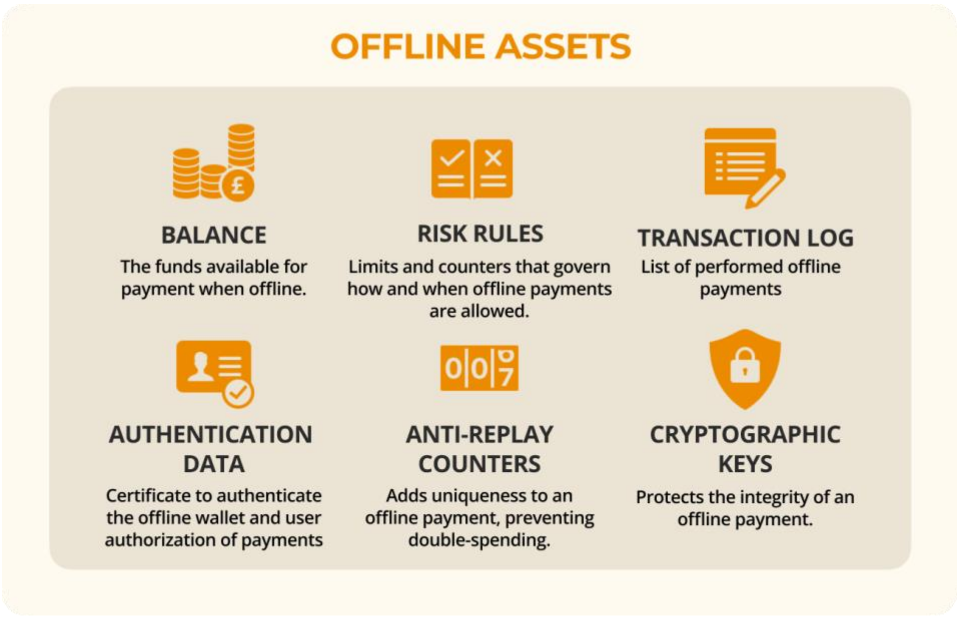
Whitepaper by Kristian Sylwander and Joachim Samuelsson, Crunchfish

In the dynamic landscape of mobile payments, offline payments for smartphones have gained traction[1]. Offline payments allow users to conduct transactions without a real-time connection to a central server, offering convenience but also introducing unique challenges in terms of security and reliability.

How can a payment application add offline payment capabilities? What are the distinct risks of introducing these types of payments? And how can you build trust that enables secure and reliable offline payments? This whitepaper addresses these questions and proposes practical requirements for system operators considering an offline payment implementation.

Understanding Offline Payments

Offline payment refers to the ability of a smartphone to complete transactions without requiring a continuous internet connection. This feature enhances user convenience, especially when network connectivity is intermittent or unavailable. Another benefit of offline payments is their ability to constantly work at the time of payment regardless of the state of the payment service backend. Hence, offline payments can alleviate pressure on the core banking system for low-value transactions.



OFFLINE ASSETS

BALANCE The funds available for payment when offline.

RISK RULES Limits and counters that govern how and when offline payments are allowed.

TRANSACTION LOG List of performed offline payments

AUTHENTICATION DATA Certificate to authenticate the offline wallet and user authorization of payments

ANTI-REPLAY COUNTERS Adds uniqueness to an offline payment, preventing double-spending.

CRYPTOGRAPHIC KEYS Protects the integrity of an offline payment.

Figure 1: A payment application must handle new asset types when enabling offline payment capabilities.

[1] https://www.bis.org/publ/othp64.pdf

Adding offline payment capabilities to a payment application requires handling a local offline balance, representing the funds available offline. But that is not all; there is also a need to add several other important asset types described in Figure 1. Throughout this whitepaper, the above assets are referred to as the offline assets. The payment application must ensure that the offline assets are kept secure both during execution of the application, which we refer to as securing the application runtime, but also at rest, where these offline assets are stored.

The Risks with Offline Payments

While offline payments offer flexibility, they also introduce specific risks that need careful consideration. The immediate risk is the ability to double-spend your balance while being offline. Since the offline assets reside in a device that is in the control of an attacker, it is paramount that the payment application takes the necessary precautions to protect the offline assets and any operation performed on them. There are multiple ways that a potential attacker can take to perform an attack that results in double-spending.

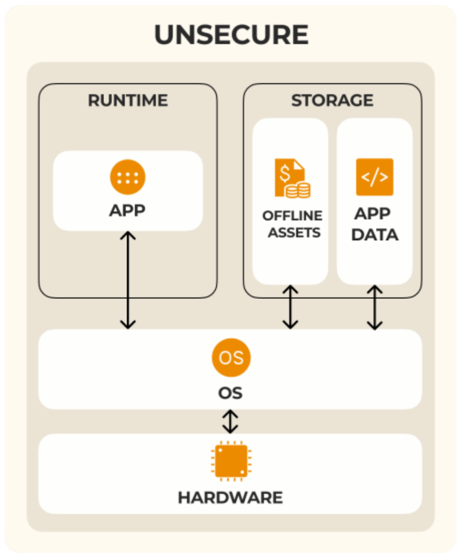


Figure 2: An unsecure offline payment solution without any security mechanisms in place.

Attacking the Bearer Instrument

Rooting (Android) or jailbreaking (iOS) a device is probably the first cause of action when attempting to attack an offline payment solution due to its simplicity and availability[2]. When users root an Android device or jailbreak an iOS device, they gain elevated privileges and access to parts of the operating system that are typically restricted. This allows users to customize their devices extensively, install unauthorized applications, and modify system files such as the

[2] https://www.kaspersky.com/blog/android-root-faq/17135/

offline assets and application data (see Figure 2). One way to approach this issue is not to enable the payment application to run if a device is rooted or jailbroken using root-detection software. However, root detection is like a cat-and-mouse game between the users and manufacturers. Hence, root-detection is always one step behind, and counting on these measures is risky. Often, they are easily circumvented by new rooting software. When designing a solution for offline payments, it is crucial to consider that rooted or jailbroken devices are present in the system.

Attackers can exploit rooting software to gain unauthorized access to application data and encryption keys. Additionally, manipulating the operating system can deceive applications by feeding them faulty information. Furthermore, altering the device's time can alter the application's behavior. All these attacks can lead to a successful double-spending attempt. It is also possible to exploit vulnerabilities in hardware, such as the CPU, which has been proven to leak cryptographic keys[3] [4].

Using the security features of the mobile operating system (OS) leaves the app blindly trusting whatever the OS throws at it. This is significant because changing the OS is relatively easy, and applications rely on sensitive APIs and cryptographic operations.

Attacking the Application Runtime

While performing an offline payment, the application reads the offline assets into memory, performs cryptographic operations, creates a transaction, and modifies the offline assets. Here, an attacker can attempt to alter the code paths to circumvent risk rules, read the sensitive data from memory, and even inject code to bypass security checks and alter values to create a valid offline payment outside the limits of the offline assets.

Common ways to attack the application are various forms of tampering, using hooking software to alter and inject code into an application and decompiling or reverse-engineering the application data. It is essential to embrace the likelihood that the security mechanisms can be circumvented and ask how that would affect the system.

Attacking the Stored Offline Assets

An attack on the offline assets typically means attempting to tamper, copy, or restore the data. If such an attempt is successful, the attacker can, e.g., extract cryptographic keys, increase the offline balance, modify the risk rules, and remove transactions, making it possible to spend money that has already been spent. These attacks can generally be performed by, e.g., copying an application to another device, running multiple instances of the same application, or restoring the state of an older version.

[3] https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-tang.pdf
[4] https://theses.hal.science/tel-03282313/file/TROUCHKINE_2021_archivage.pdf

Mitigating the Risks with Offline Payment

Ensuring trust in offline payments circles around protecting offline assets in various situations. It is possible to attack these offline assets in different ways. Hence, an offline payments solution needs multiple layers of security mechanisms to counter-act, detect, and mitigate these attacks. This section covers the baseline of security mechanisms required to achieve enough trust to conduct offline payments safely.

Mobile application protection is hard, and existing mobile software protections are insufficient for offline payments. They come in different shapes and forms, but all use three types of software protection: Code Obfuscation, Runtime Application Self-Protection (RASP) or App Shielding, and White-box Cryptography. Fundamentally, all these protections are better than nothing but do not suffice for offline payments since they do not hold attackers back for long[5]. A payment application requires protection not only against direct attacks on the application itself but also against indirect attacks that may manifest as manipulation of the OS or exploitation of hardware vulnerabilities.

While performing an offline payment, the application executes code that operates on the offline assets. This is a sensitive operation that requires specific protection. Using standard Android or iOS security for offline payments is not sufficient since the security available is not designed for that purpose. For example, the Android Keystore or iOS Keychain provides an isolated runtime where applications can generate keys and perform cryptographic operations. However, it protects only the use of cryptographic keys and is, therefore, not sufficiently secure for offline payments as they lack a secure runtime for other offline assets. A simple attack is all it takes to bypass the offline balance and the risk limits. Furthermore, it has been proven that Android Keystore, built on top of ARM Trustzone, has vulnerabilities that have given an attacker access to the private keys stored within[6].

Protecting the Application Runtime

A tamper-resistant element (TRE) is a security feature designed to prevent or detect unauthorized access, modification, or compromise of a device. This TRE can provide an isolated secure runtime, which is very difficult to attack from the outside. Here, it is safe to perform operations on all offline assets and generate, store, and operate on cryptographic keys without risking any interference from a potential attacker. A secure, isolated environment is essential for an offline payment solution to ensure its logic remains secure. One way of achieving this isolation is to integrate a virtual TRE in the payment application. Using the foundation of a virtual tamper resistant element (TRE), multiple layers of protection can run through its secure, isolated runtime, securing the protection mechanisms themselves[7]. Attackers cannot tamper with the payment application or bypass the protection mechanisms without first breaking the

[5] https://eprint.iacr.org/2018/098.pdf
[6] https://ieeexplore.ieee.org/document/9152801
[7] https://www.crunchfish.com/wp-content/uploads/2023/11/Lipis_WP6_Crunchfish_Enabling-offline-payments.pdf

TRE itself. Combining multiple protection mechanisms is crucial for achieving the necessary security for the payment application. Examples of protection mechanisms to consider are (but are not limited to) anti-reverse engineering, time locking, anti-cloning, anti-code-lifting, anti-debugging, anti-hooking, anti-code injection, app integrity checks, and emulator/simulator detection. Detecting a threat locks the payment application and requires the backend to unlock it again.

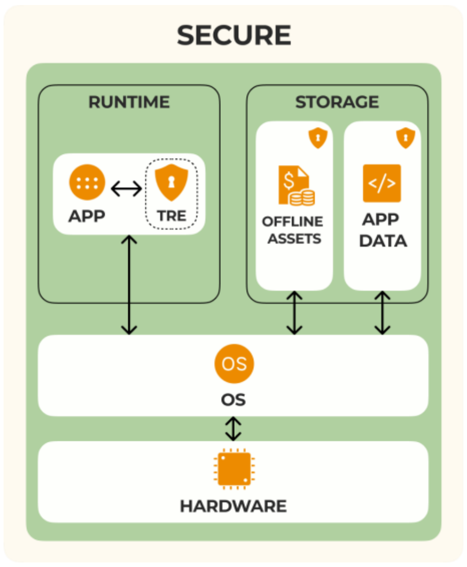


Figure 3: A secure Offline Payment solution that protects the application runtime, the application data, and the offline assets.

The virtual TRE, using a virtualized CPU and encrypted memory, creates a self-contained space within the application, isolating it from the underlying OS. This isolation ensures that the offline payment solution operates within its secure environment, shielded from potential vulnerabilities or changes in the OS. The virtual TRE includes many necessary dependencies and libraries for offline payments. This independence means offline payments are less reliant on external OS components, reducing the risk of vulnerabilities associated with OS updates. Keeping the app self-sufficient makes security updates as simple as updating the payment application. This is key to ensuring scalability in the market, as offline payments are available to all smartphones without any dependencies on devices or mobile operators. Furthermore, the application-bound architecture allows a payment application to run securely on rooted devices, which makes the question of rooted vs non-rooted irrelevant. Root detection mechanisms may still complement an offline solution but should not depend on it.

Protecting the Stored Offline Assets

Securing the payment application is crucial, but we must also protect the offline assets stored on the device. Android Keystore and iOS Keychain store their cryptographic keys encrypted on the device's filesystem. This protects the key itself from being copied to another device, but it does not protect it from attacks happening on the same device.

Instead, storing the offline assets in trusted storage through the virtual TRE is a great way to keep the offline assets in the control of the application while also keeping them encrypted using cryptographic keys. This restricts access to the offline assets, but they are still stored on the file system, leaving them open to attack.

Rollback Protection

Implementing rollback protection is a must-have to prevent tampering with the offline assets stored on the device. A rollback attack, in short, is a way to restore a previous state, usually a balance, of a wallet, making it possible to spend money that has already been spent. This type of attack opens the possibility of double-spending. To counter a rollback, an offline payment solution requires the detection of any tampering with the offline assets and, as a result, locking the offline wallet. Additionally, utilizing a TRE also protects the rollback protection logic from tampering. Unlocking the offline wallet is only possible through an online sync with the backend.

Leveraging Backend Reconciliation

There is increased acknowledgment from the market that backend reconciliation is an important mechanism when securing offline payments. The ECB has indicated that user devices would be expected to go online regularly for offline payments using a future digital euro[8]. By leveraging the backend, the offline payment solution can perform synchronization operations between the offline wallet and the online backend. For example, the online backend can analyze offline transactions and detect if a particular user made a rollback of the offline assets and, if so, enforce a lock the user's offline wallet.

[8]https://www.ecb.europa.eu/paym/digital_euro/investigation/profuse/shared/files/dedocs/ecb.dedocs230113_Annex_1_Digital_euro_market_research.en.pdf

Conclusion

This whitepaper has delved into securing offline payments for smartphones, exploring the inherent risks, particularly the threat of double-spending. The whitepaper emphasizes the importance of keeping offline payments contained in its own environment within the payment application and provides guidelines for what types of security mechanisms make up the foundation of an offline payment solution.

.

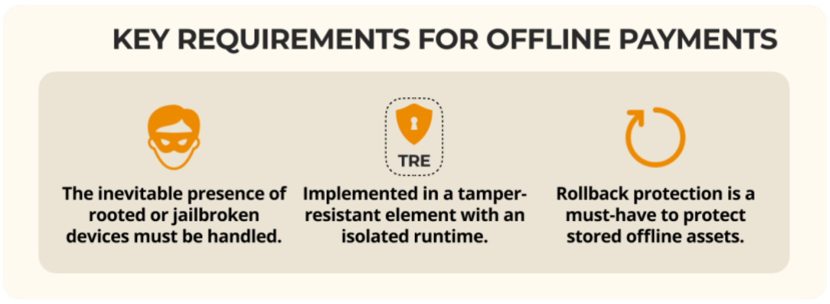

KEY REQUIREMENTS FOR OFFLINE PAYMENTS
TRE
The inevitable presence of rooted or jailbroken devices must be handled.
Implemented in a tamper-resistant element with an isolated runtime.
Rollback protection is a must-have to protect stored offline assets.

Figure 4: The key requirements for offline payments.

Condensing the mitigations down to three key requirements for an offline payment solution:

- The inevitable presence of rooted or jailbroken devices must be handled.
- Implemented in a tamper-resistant element with an isolated runtime.
- Rollback protection is a must-have to protect offline assets.

Following this whitepaper's guidelines and requirements ultimately ensures robust and resilient offline payments for smartphone users.

AN OFFLINE PAYMENT SOLUTION

Crunchfish Digital Cash is an offline payment solution providing a secure environment for offline payments. It includes isolation by a virtualized secure runtime and encrypted storage with rollback protection for all offline assets. As Digital Cash is an integral part of the payment application, shielding it from the OS and hardware provides a consistent level of security for offline payments independent of the device it is running. Another key benefit is that Digital Cash can securely run on rooted or jailbroken devices, as a compromised device does not affect the running of Digital Cash within its virtual TRE. It is also highly scalable, as Digital Cash is deployed and updated with the payment application on all smartphones using app stores. Digital Cash effectively handles the three key requirements suggested by the paper ensuring secure offline payments.