

The Trust Gap:

Mobile Payment Apps and the Boundaries of Secure Hardware

Ben Thompson

Senior Product Specialist

Licel



20 YEARS OF SECURING PAYMENT DATA

PCI Security Standards Council

Licel solutions protect organizations around the world from fraud, abuse, and data theft, securing the mobile channel connecting them to more than 500 million customers.

Licel



DexProtector

Enterprise mobile application protection; obfuscation, encryption, RASP, & telemetry



Licel vTEE

Logical isolation & protected cryptography for mobile payment & identity operations



Alice Threat Intelligence

Security & anti-fraud platform; mobile device attestation & monitoring

EMVCo SBMP evaluated & approved for 6 consecutive years

1 in 4 MPoC-certified products integrate Licel solutions

20 YEARS OF SECURING PAYMENT DATA

Agenda



1. **Introduction**
    - 1.1 Mobile Payment Apps
    - 1.2 SoftPOS & MPoC
    - 1.3 What is secure hardware, exactly?
2. **Secure Hardware on COTS Devices**
    - 2.1 How do mobile payment apps use secure hardware?
    - 2.2 Limitations: Availability, Performance, Security
3. **The Trust Gap**
    - 3.1 Data before it’s encrypted
    - 3.2 Cryptographic downgrades
    - 3.3 Keys & secrets outside hardware operations
4. **Summing Up**

1. Introduction

Mobile Payment Apps; SoftPOS & MPoC; Secure Hardware



1.1 Mobile Payment Apps (& SDKs)

- Third-party apps, typically installed through public or enterprise app stores, and normally running in the rich execution environment (REE) using platform APIs.
- Excludes pre-installed & system apps, TEE Trusted Applications, and Secure Element applets that execute directly on secure hardware.
- Core payment functionality may be supplied by an independently developed SDK integrated into the host app.
- SoftPOS main example for this talk, but same principles apply to HCE wallets and apps supporting tokenised payments; similar requirements for protecting payment card credentials and data.

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council

1.2 SoftPOS & MPoC

- SoftPOS: accepting payments on a merchant's general-purpose phone or tablet, typically using native NFC interface; optionally touchscreen for PIN entry.
- PCI MPoC: Mobile Payments on COTS (Commercial Off-the-Shelf Devices). Standard builds on CPoC for contactless acceptance and SPoC for PIN entry.
- Focus on client-side MPoC Application and SDK components; how those components interact with COTS devices' secure hardware.

References are to PCI Mobile Payments on COTS Security and Test Requirements v 1.1



1.3 What is secure hardware, exactly?

- **Hardware isolation:** Trusted Execution Environments (TEEs), Secure Elements, Secure Enclave, StrongBox, TPMs; memory and access controls enforced below the app/OS layer.
- **Restricted interfaces:** third-party apps typically request specific operations through platform APIs; they do not receive unrestricted access to secure memory, keys or execution.
- **Non-exportability:** app gets a key reference and asks the secure component to perform an operation rather than receiving the key itself.
- **Android:** Android Keystore / KeyMint can be backed by a TEE or StrongBox.
- **iOS:** Keychain, Security and CryptoKit can use Secure Enclave-backed keys; payment Secure Element is separate.

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council

2. Secure Hardware on COTS Devices

Uses and Limitations



2.1 How do mobile payment apps use secure hardware?

- **Cryptographic operations**: encrypt account data and PIN blocks, generate MACs or signatures.
- **Create non-exportable device-bound keys** for encryption, signing, key wrapping and app/device binding.
- **Support trust decisions**: use hardware-backed key attestation and device-bound keys for authentication and inputs to server-side monitoring.
- **Not the whole workflow**: kernel logic, PIN entry and handling, transaction preparation etc. happen outside secure hardware; some data inevitably in plaintext outside secure hardware.

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council

2.2 Limitations: Availability, Performance, Security

- **Availability varies:** not every COTS device exposes the same hardware-backed keystore capabilities.
- **Performance constraints:** hardware-backed operations can be slow.
- **Algorithm support limited:** Android KeyMint exposes certain primitives; StrongBox supports fewer algorithms/key sizes. iOS Secure Enclave keychain items are limited to 256-bit elliptic curve private key
- **Unsupported protocols & operations are pushed back into the untrusted REE.** Payment protocols often require multi-step operations, custom key hierarchies and/or algorithms that hardware and platform APIs do not support.
- **Keys stay protected while data is exposed.** App/SDK handles plaintext before and after hardware-backed cryptographic operations. Ordinary app process is not a secure execution environment.

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council

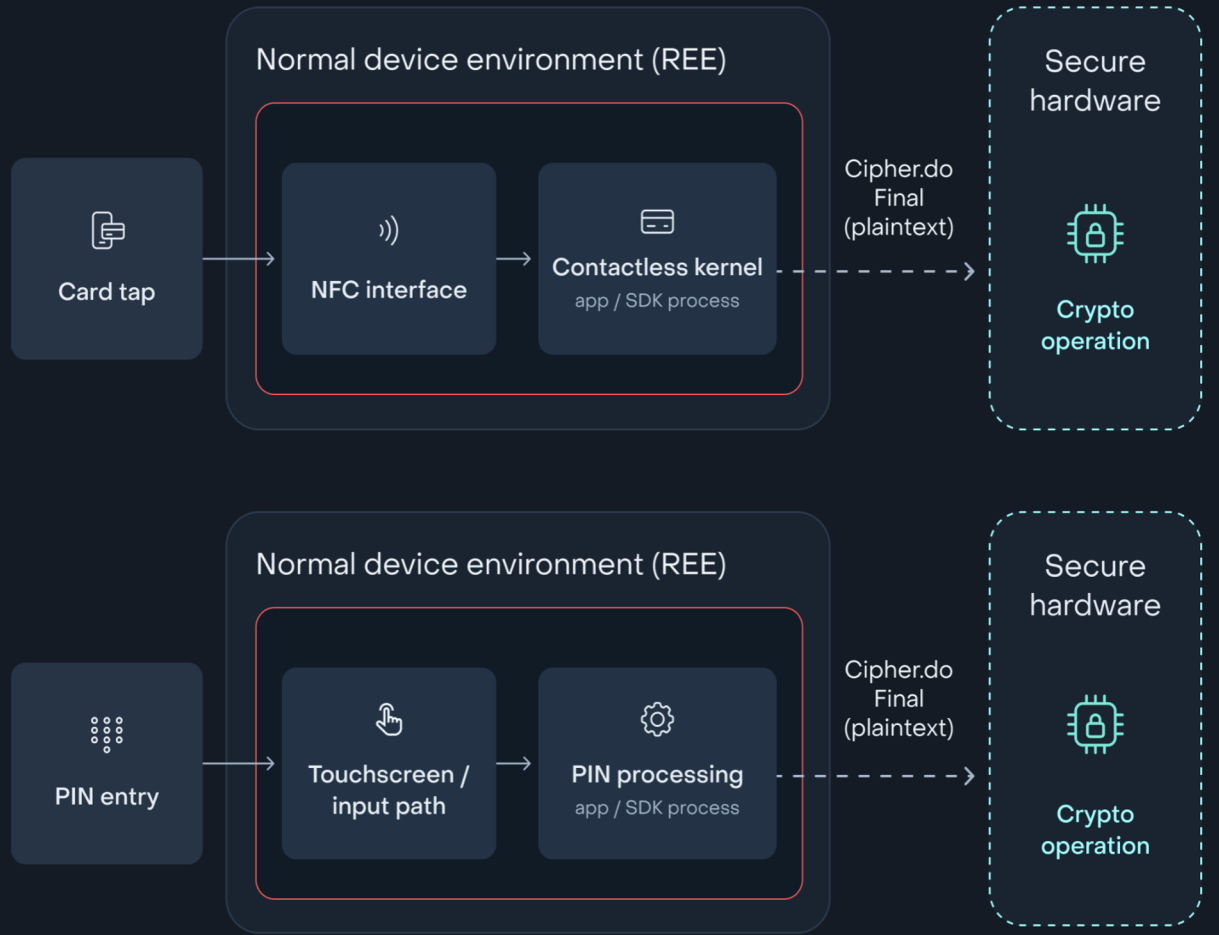
3. The Trust Gap

Why Software Protections Remain Essential



3.1 Attacks may target data *before* it's encrypted

- Classic attack against payment apps: steal payment data before encryption.
- Card data and PIN input pass through device interfaces, OS components, and app process before they can be encrypted.
- Compromised app or runtime can hook NFC callbacks, input handling, kernel functions, PIN formatting, temporary buffers, etc.
- Additional risk to PIN from malware, accessibility service abuse, overlays, screen sharing.



3.1a Protecting data *before* it's encrypted

Security design

- Encrypt account data at earliest possible point.
- Clear plaintext sensitive assets immediately after use; minimise lifetime in memory.

  (Relevant MPoC Security Requirements & Guidance: 1B-1.4; 1D-1.2)

Software protection & attestation

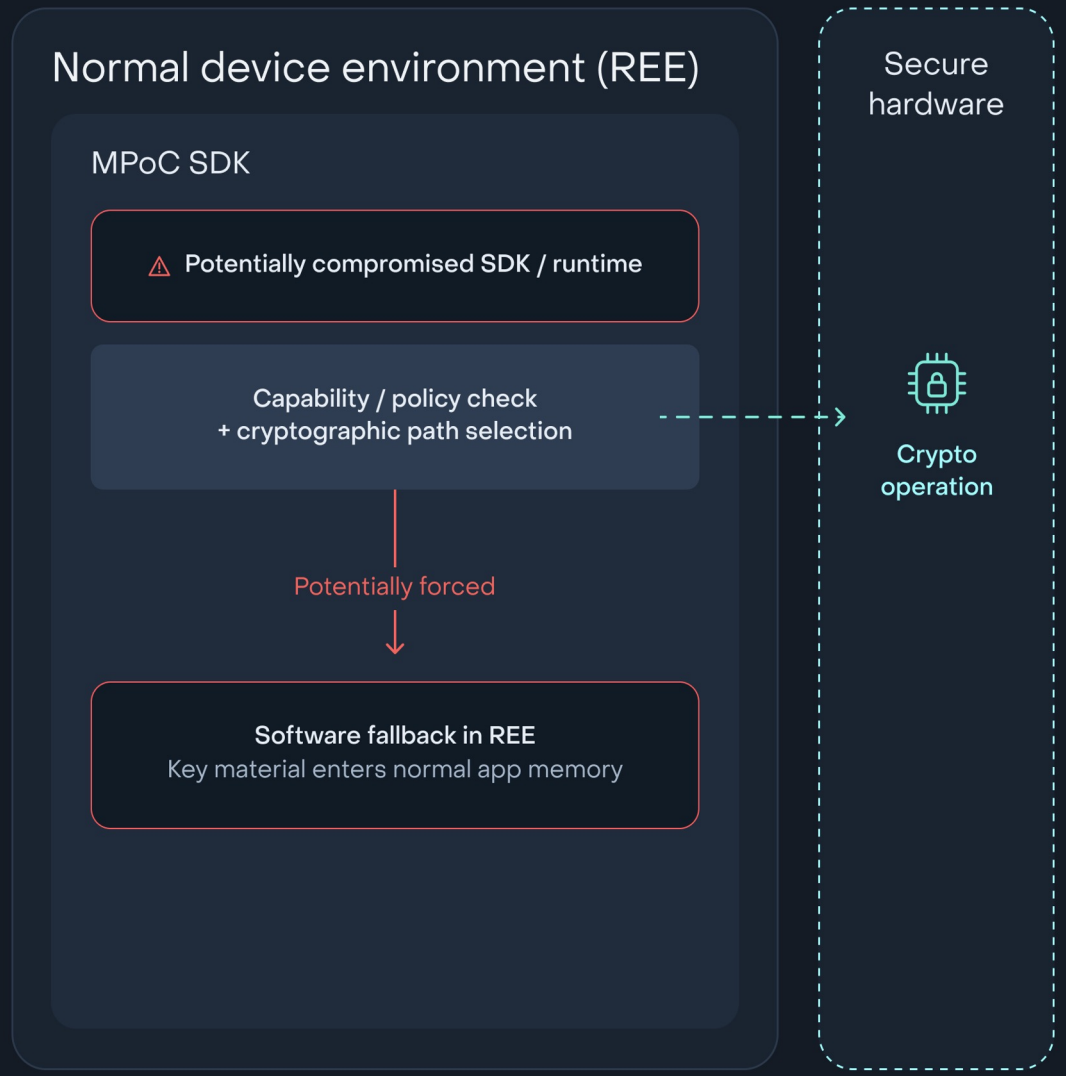
- App/SDK integrity, anti-tampering, anti-hooking, root/jailbreak detection, extended app & device attestation, malware/accessibility/overlay risk signals.
- Security decision **before exposure**: if device/app condition is unacceptable, restrict execution before starting NFC acceptance or presenting PIN UI.
- Protections & attestation do not make REE trusted; they decide whether it's safe enough to expose sensitive asset, then protect that asset while it exists in clear.

  (Relevant MPoC Security Requirements & Guidance: 1B-1.5; 1B-1.14; 1C-1.3; 1E-1.6; 1E-1.7)

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council

3.2 Attacks may force cryptographic downgrades

- Hardware-backed support varies based on algorithm, key size, operation, device.
- App may choose another path when preferred hardware operation is unavailable.
- Modified app, instrumented runtime or controlled device may falsify capability checks, force errors, suppress evidence, tamper with code that selects the fallback.
- Risk: device/installation authentication key, account data derivation key, PIN encryption key or transport/KEK may be exposed to REE.



3.2a Preventing (uncontrolled) cryptographic downgrades

Security design

- Define approved boundary for each crypto operation.
- Specify whether secure hardware is **required**, **preferred with an approved fallback**, or **not used** for that workflow.
- Define fallback paths (e.g. StrongBox to TEE, hardware to vTEE/software-protected cryptography, or reject device).
- Prevent exposure of keys and secrets through unprotected REE cryptography.

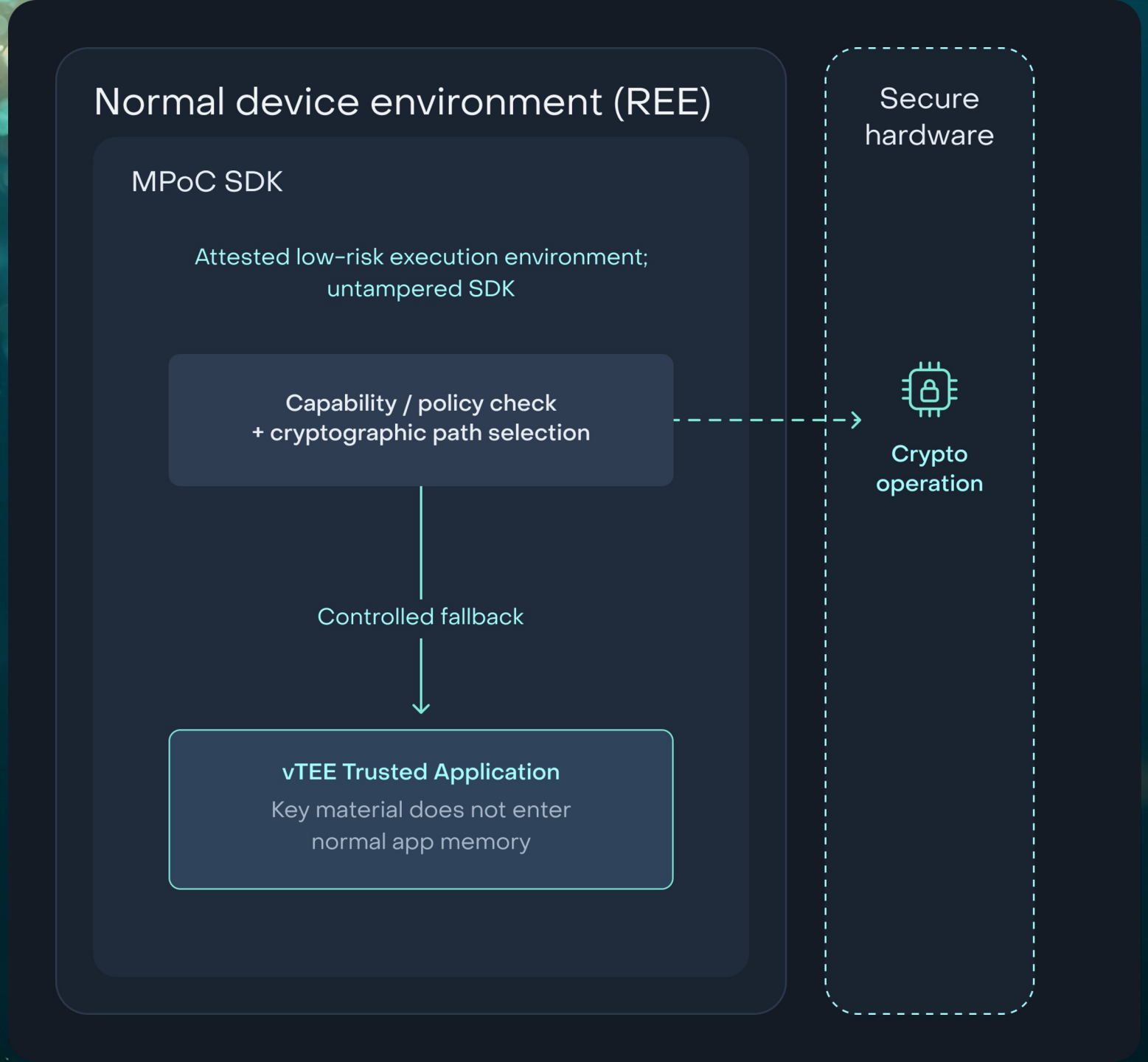
  (Relevant MPoC Security Requirements & Guidance: 1B-1.1; 1B-1.3; 1B-2 guidance)

Software protection, attestation, & software-protected cryptography

- Protect capability checks, error handling, fallback selection against tampering.
- For operations with inconsistent hardware support / no hardware support / unsuitable performance, prefer robust software-protected cryptography implementation.

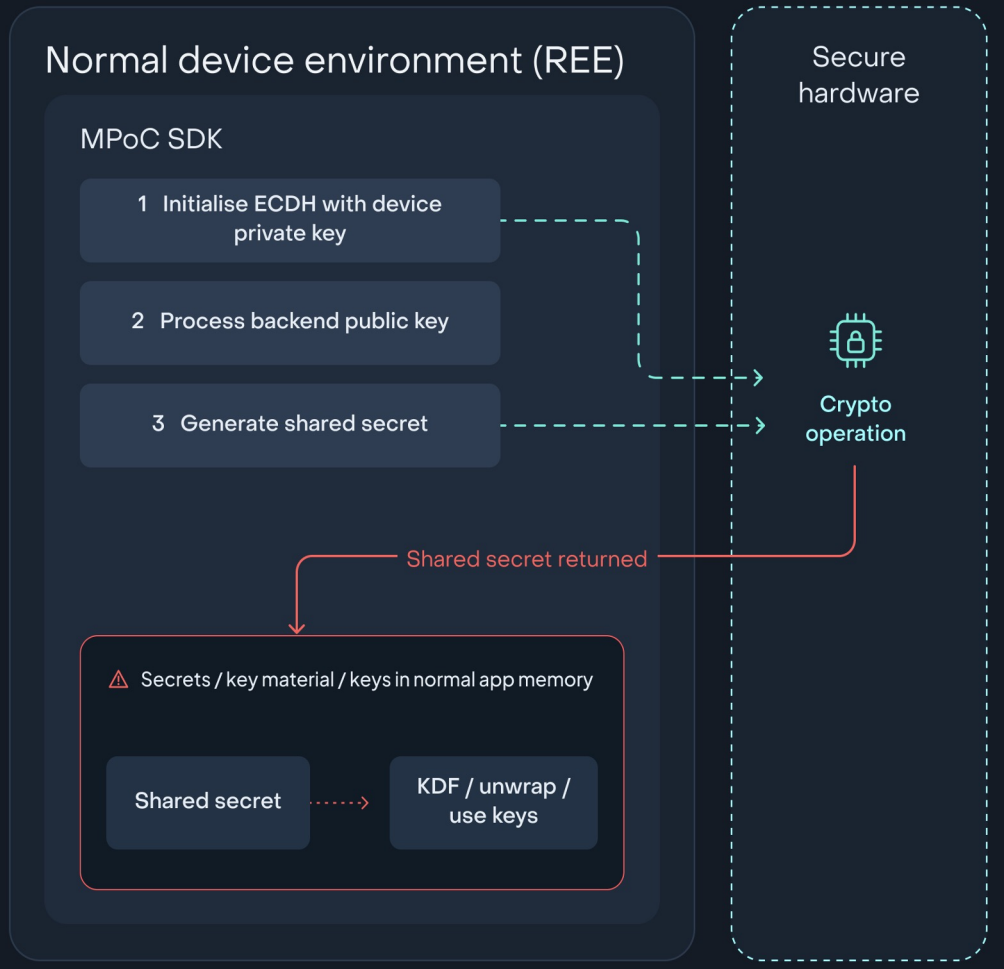
  (Relevant MPoC Security Requirements & Guidance: 1B-1.5; 1B-1.14; 1B-2 guidance; 1C-1.3; 1C-3.2)

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council


Normal device environment (REE)
MPoC SDK
Attested low-risk execution environment;
untampered SDK
Capability / policy check
+ cryptographic path selection
Controlled fallback
vTEE Trusted Application
Key material does not enter
normal app memory
Secure
hardware
Crypto
operation

3.3 Attacks may extract keys & secrets *outside* hardware operations

- Platform keystores expose fixed set of cryptographic primitives; some payment workflows are unsupported or only partly.
- Hardware API may protect only one primitive while returning useful intermediate result to the REE; any multi-stage payment cryptographic workflow can expose useful working material between protected primitives.
- Examples: ECDH shared secret; account data derivation state; PIN encryption key; KEK/transport key; unwrapped working key; DUKPT transaction key.
- Attacker can target useful downstream secrets.



3.3a Protecting keys & secrets *outside* hardware operations

Security design

- Use hardware-backed cryptography where it reliably provides the required operation and an appropriate protection boundary.
- Use single-purpose keys and one-way derivation.
- Use transaction-unique keys and forward secrecy for account data encryption keys.

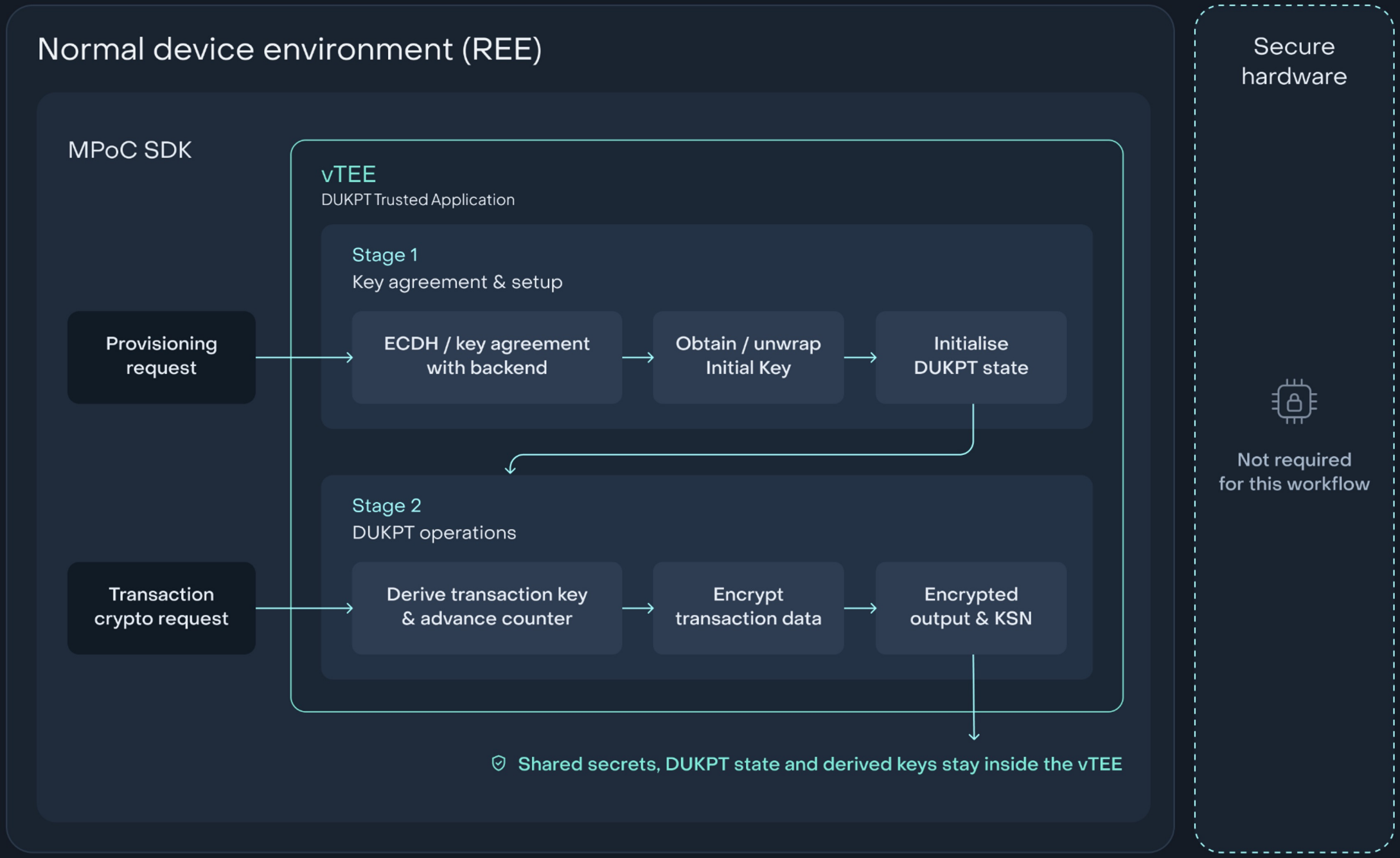
  (Relevant MPoC Security Requirements & Guidance: 1A-3.4; 1A-3.5; 1A-4.11; 1A-4.12)

Software-protected cryptography

- Where platform hardware cannot secure the complete workflow, use appropriate software-protected cryptography implementation.

  (Relevant MPoC Security Requirements & Guidance: 1B-2.1; 1B-2.2; 1B-2.3; 1B-2.7; 1B-2 guidance)
- White-box crypto: protect particular key/algorithm implementations.
- vTEE: protect complete multi-stage key-management and cryptographic workflow, keeping intermediate keys and state out of ordinary application memory.

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council


Normal device environment (REE)
MPoC SDK
vTEE
DUKPT Trusted Application
Stage 1
Key agreement & setup
Provisioning request
ECDH / key agreement with backend
Obtain / unwrap Initial Key
Initialise DUKPT state
Stage 2
DUKPT operations
Transaction crypto request
Derive transaction key & advance counter
Encrypt transaction data
Encrypted output & KSN
Shared secrets, DUKPT state and derived keys stay inside the vTEE
Secure hardware
Not required for this workflow

4. Summing Up

Secure hardware protects cryptographic operations up to a boundary, not the whole transaction.

Protect data before it's encrypted
Minimise cleartext exposure; software protection & attestation: obfuscation, RASP, integrity, anti-tampering, anti-hooking, root/jailbreak detection, device/app attestation.

Prevent uncontrolled cryptographic downgrades
Define approved hardware, fallback, reject paths; protect capability checks, key provisioning, attestation evidence, and fallback logic.

Protect keys & secrets outside hardware operations
Use software-protected cryptography where hardware cannot contain the workflow; use a vTEE to protect multi-stage cryptography and intermediate secrets.

20 YEARS OF SECURING PAYMENT DATA
PCI Security Standards Council