Universal Library of Innovative Research and Studies Research Article

ISSN: 3065-0003 | Volume 3, Issue 1

Open Access | PP: 101-109

DOI: https://doi.org/10.70315/uloap.ulirs.2026.0301014

The Use of Hardware Security Mechanisms (Secure Enclave) for Secure Storage of Cryptographic Keys on Mobile Devices

Pankiv Oleg

Senior IOS Developer, Myseum.Inc, Bayonne.

Abstract

This article examines the evolution of hardware mechanisms for protecting cryptographic keys in mobile platforms, tracing the progression from ARM TrustZone-based Trusted Execution Environments (TEE) to physically isolated Secure Elements. The relevance of the work is determined by the shift of attack vectors toward microarchitectural components, the emergence of new vulnerabilities, and the limited adoption of StrongBox/Secure Enclave in applications. The objective of the study is to conduct a comparative analysis of the SEP, Knox Vault, and Titan M2 architectures, to assess their resilience to side-channel and fault-injection attacks, and to evaluate the impact of the choice of key-storage environment on the latency and energy profile of cryptographic operations. The scientific contribution lies in combining a detailed architectural survey with empirical data from the KeyDroid project and in formulating a multi-layer model for distributing keys between SE, TEE, and REE, explicitly accounting for preparation for post-quantum migration. It is demonstrated that logical isolation via TrustZone is insufficient. In contrast, physically segregated enclaves provide a more robust root of trust but require transitioning to ECC and asynchronous usage patterns for hardware-backed keystores. The article is intended for researchers and practitioners in mobile security, platform architects, and application developers working with sensitive data.

Keywords: *Secure Enclave, Trusted Execution Environment (TEE), ARM TrustZone, Cryptographic Performance, Mobile Security.*

INTRODUCTION

The modern smartphone has ceased to be merely a communication device. It has transformed into a universal digital wallet, an authentication token (FIDO2/WebAuthn), a repository of medical data, and a storage medium for digital car keys [1]. This consolidation of sensitive information into a single carrier has shifted the underlying threat model. Whereas application-level attacks (malware) predominated in the past, the development of OS-level protection mechanisms (such as sandboxes and permission controls) has shifted the attack vector toward privileged software and hardware layers [2].

The traditional security model, which relies on access-control mechanisms within the operating system (the Rich Execution Environment, REE), has proven inadequate in the face of kernel-level zero-day vulnerabilities [3]. Once an adversary obtains superuser privileges (root/jailbreak), all software-based protection mechanisms are effectively neutralized, and access to encryption keys residing in process memory is obtained. The industry's response was the introduction of the concept of Trusted Execution Environments (TEE), the most widely known implementation of which is ARM TrustZone. TEE provides logical isolation between a secure and an everyday world, but still relies on the same processing cores, caches, and power rails [4].

Research over the last five years (2020–2025) has convincingly demonstrated that logical isolation alone is insufficient [5]. Shared microarchitectural resources enable side-channel attacks that extract secret keys by analyzing cache access latency (Cache Timing Attacks) or by manipulating supply voltage (Fault Injection). In response, leading vendors have migrated to architectures based on physically isolated security subsystems, Secure Enclaves, and Secure Elements [6].

The central problem arises from a dichotomy between the need for maximum data protection and stringent constraints on performance and energy efficiency in mobile devices. Further, physically isolated security coprocessors (Secure Elements) on devices like the Google Titan M2 or Samsung Knox Vault Processor have considerably lower throughput than the central application processor (AP). Performing crypto operations on Secure Elements can degrade

Citation: Pankiv Oleg, "The Use of Hardware Security Mechanisms (Secure Enclave) for Secure Storage of Cryptographic Keys on Mobile Devices", Universal Library of Innovative Research and Studies, 2026; 3(1): 101-109. DOI: https://doi.org/10.70315/uloap.ulirs.2026.0301014.

application performance, significantly slow down devices, and increase battery consumption.

This work is essential because it highlights the trade-off between security and performance when choosing an architecture for key storage. Furthermore, the emergence of new attack vectors targeting the microarchitecture of Apple Silicon and ARM in 2024–2025 necessitates a reassessment of the robustness of existing solutions [7]. In the scientific discourse, there is a noticeable deficit of comprehensive studies that combine hardware-architecture analysis with empirical performance data from the Android StrongBox and iOS CryptoKit APIs [6].

The study aims to compare the effectiveness of contemporary hardware mechanisms for protecting keys on mobile platforms, identify their vulnerabilities to next-generation attacks, and assess their impact on application software performance.

To achieve this aim, the following tasks are addressed:

Systematize architectural differences between TEE implementations (based on TrustZone) and dedicated Secure Elements (Apple SEP, Knox Vault, Titan M2), with emphasis on memory-isolation mechanisms and power-domain management.

Analyze the threat landscape of 2021–2025, including attacks on the power subsystem (CLKSCREW, Plundervolt) and speculative execution (SLAP, FLOP), and evaluate the effectiveness of hardware countermeasures.

Present empirical data on the performance (latency, energy consumption) of cryptographic primitives in different key-storage environments, based on large-scale studies of the Android application ecosystem.

Formulate integration guidelines for hardware cryptography from the developer's perspective, minimizing UX impact while maintaining a high security level.

The work synthesizes performance data for the Android StrongBox subsystem based on the Titan M2 (RISC-V) chip, comparing it with traditional TEE implementations, and relies on measurements conducted in 2024–2025 within the KeyDroid project. In addition, the influence of the latest Apple Silicon microarchitectural vulnerabilities (SLAP/FLOP) on trust in isolated enclaves is analyzed, an issue that has not previously been examined in conjunction with performance.

MATERIALS AND METHODS

The study takes the form of a systematic review with elements of comparative analysis of technical specifications and interpretation of empirical data. The methodology is built on the triangulation of data from academic sources, vendor technical documentation, and reports from independent security laboratories.

For the comparative analysis of security architectures (Apple SEP, Samsung Knox Vault, Google Titan M2), a functional decomposition method was applied. Each system was examined through the lens of a set of key components. First, the compute core was analyzed, including the architecture type (ARM, RISC-V, or proprietary) and the presence and organization of dedicated caches. Second, memory-isolation mechanisms were evaluated, including memory-encryption technologies (e.g., the Memory Protection Engine), replay-attack protection, and MMU subsystem control.

Special attention was paid to the enclave operating system, including the use of microkernel-based designs (such as L4 or OpenSK) and the resulting attack surface of the software stack. The interaction of the enclave with peripheral devices was also analyzed, including integration with biometric sensors, NFC controllers, and display subsystems.

The performance assessment relies on secondary analysis of data obtained from instrumental measurements on real consumer devices, including smartphones of the Google Pixel, Samsung Galaxy S, and Apple iPhone families. The key performance metrics considered are the latency of cryptographic operations for key generation (KeyGen), signing (Sign), and encryption (Encrypt) for RSA-2048, ECC P-256, and AES-GCM-256. Additionally, energy-consumption metrics are taken into account, expressed either in joules or indirectly via the duration of processor active-state retention (wake locks), allowing comparisons of overheads across different implementations under comparable conditions.

A threat analysis was carried out using the STRIDE threat modeling method, which enables a systematic evaluation of the resilience of the studied architectures to various attack classes. Particular focus was placed on Fault Injection attacks that induce power or frequency glitches, as well as on Side-Channel attacks, in which an adversary derives information from power-consumption patterns or the timing characteristics of operations. The empirical and theoretical basis for the assessment consisted of academic publications describing the CLKSCREW, Plundervolt, TruSpy, and iLeakage attacks, which provided a foundation for reproducing and documenting the compromise vectors [16–18]. The work is based on the analysis of more than 150 sources, from which the most relevant 20 publications for the period 2018–2025 were selected.

RESULTS AND DISCUSSION

Comparative Analysis of Hardware Security Architectures

The evolution of protection mechanisms for mobile devices has progressed from software obfuscation to full-fledged hardware segregation. This section details the technical aspects of contemporary Secure Enclave implementations.

Limitations of the ARM TrustZone Architecture

ARM TrustZone technology, which underpins most TEE implementations on Android platforms (such as Qualcomm QSEE), employs a time-multiplexing approach to resource sharing. The processor operates in two states: Secure World and Normal (Non-secure) World. Switching between them is performed via the special SMC (Secure Monitor Call) instruction [4].

Despite logical memory isolation via the TrustZone Address Space Controller (TZASC), both worlds share the same physical components: cache hierarchy (L1/L2/L3), branch predictors, and, crucially, the power subsystem [8]. This creates fundamental attack vectors. Studies have shown that an adversary in the Normal World can analyze residual cache state after code execution in the Secure World or manipulate processor voltage to induce faults in cryptographic computations [9]. Recognition of these risks has driven the emergence of designs with physical separation.

Apple Secure Enclave (SEP)

Apple Secure Enclave (SEP) is a dedicated security subsystem integrated into the system-on-chip (SoC) of the Apple A and Apple M families. Architecturally, SEP is based on a customized core (presumably ARM-based; earlier versions referenced an L4 microkernel architecture) [10], operating under its own SEPOS operating system. This OS is logically and functionally separated from iOS and macOS, meaning that it executes independently of the primary software stack.

Memory isolation is configured so that SEP does not directly access shared system RAM. Instead, it uses a dedicated region of physical memory (TZ0), protected by a hardware Memory Protection Engine that provides transparent encryption and authentication of data. Memory-encryption keys are generated at each system boot (ephemeral keys). They are cryptographically bound to the device's hardware UID, making it impossible to read the memory contents even if the chip is physically extracted [10]. The Secure Enclave Processor is a separate compute core that is invisible to the central processor's task scheduler, thereby reducing the risk of time-slicing-related attacks.

A critically important architectural feature is the integration with biometric sensors: Touch ID and Face ID sensors are connected directly to SEP via I2C/SPI buses, bypassing the central processor. Biometric image processing and template matching are performed inside the enclave, and the operating system receives only a cryptographic confirmation token [10].

Samsung Knox Vault: Multi-layer Protection

Samsung Knox Vault represents an evolution of the TrustZone approach toward tighter, including physical, isolation of critical components. Along with Knox Vault Processor, which runs sensitive applications on a separate processor core from the AP (Application Processor), Knox Vault also includes Knox Vault Storage, which is separate from the UFS-based storage subsystem and uses a separate non-volatile memory, Secure Flash. This organization mitigates attacks targeting the file system and memory controller, as data processed by Knox Vault does not intersect with the standard storage path.

The Knox Vault architecture incorporates physical tamper-resistance sensors, including detectors for abnormal temperature, voltage, laser exposure, and tampering with the device enclosure. It wipes content when triggers are received instead, making it impossible to steal the key material under physical access conditions. It includes Samsung Weaver, a hardware implementation of password-guessing protection. To counter brute-force dictionary attacks, Weaver exponentially increases the delay between successive authentication attempts, making such attacks impractical even with access to the file system.

Google Titan M2 and Android StrongBox

Google employs a discrete security-chip strategy (Discrete Secure Element) in its Pixel device line and is concurrently promoting the StrongBox standard for the broader Android ecosystem. The hardware's Titan M2 component is based on the open-source RISC-V architecture [12], and the use of an open architecture is seen as a calculated decision aimed at enhancing the security implementation's auditability and transparency. Titan M2 also includes its own SRAM, embedded Flash memory, and cryptographic accelerators, which enable it to run sensitive workloads entirely within its own boundary.

On the software side, Android StrongBox KeyMint uses a Hardware Abstraction Layer (HAL) interface that defines an SE as a separate processor, memory, and True Random Number Generator (TRNG). StrongBox provides a higher isolation level than a TEE (Trusted Execution Environment), while being constrained by considerably more limited computational resources.

From the standpoint of the trusted-boot chain, Titan M2 acts as the central component of the Verified Boot mechanism: it verifies the bootloader signature before any code is executed on the central processor, helping to prevent Evil Maid attacks and attempts to persist malware at the firmware level [12]. Protection against insider threats is implemented by restricting Titan M2 firmware updates to those bearing a valid Google cryptographic signature; at the same time, even Google cannot technically extract user data without knowledge of the password due to a hardware rate-limiting mechanism on authentication attempts [13].

Comparative Architectural Characteristics

A consolidated table highlighting key differences between the examined approaches is presented below (see table 1).

Table1. Comparative analysis of hardware security architectures for mobile devices

Characteristic	ARM TrustZone (Generic TEE)	Apple Secure Enclave (SEP)	Samsung Knox Vault	Google Titan M2 (StrongBox)
Implementation type	Logical isolation (Secure World)	Dedicated subsystem in SoC	Dedicated subsystem + external memory	Discrete chip (RISC-V)
Compute core	Shared with AP (context switching)	Dedicated core (Apple custom)	Dedicated processor (Secure Processor)	Dedicated core (RISC-V)
Enclave OS	QSEE, Kinibi, OP-TEE, etc.	L4 microkernel (SEPOS)	Samsung proprietary	Google proprietary / OpenSK
Memory isolation	Logical (TZASC), shared RAM	Physically dedicated encrypted region	Physically separate memory chip	On-chip internal memory (SRAM/Flash)
Resistance to side-channel attacks	Low (shared cache, power)	High (dedicated power/clock)	High (physical isolation)	Very high (discrete chip)
Protection against physical attacks	Limited	Medium (memory encryption)	High (tamper and laser sensors)	High (resistant to probing)
Primary API	Android Keystore (TEE)	CryptoKit / Keychain	Android Keystore / Knox SDK	Android Keystore (StrongBox)

The data in the table underscore a fundamental industrial shift: abandoning shared resources in favor of complete physical isolation is a direct response to the impossibility of eliminating microarchitectural vulnerabilities in modern high-performance processors solely through software.

Empirical Analysis of Performance and Energy Efficiency

The transition to physically isolated security elements (Secure Elements, SE) inevitably entails performance degradation. Security microcontrollers (Knox Vault Processor, Titan M2) are optimized for robustness and energy efficiency rather than raw computational speed. Moreover, interaction between the central processor (AP) and the SE occurs via relatively slow serial interfaces (SPI, I2C) or via mailbox mechanisms, which introduce additional latency due to data marshalling and context switching.

Analysis of Cryptographic Operation Latency

The KeyDroid study [6], conducted in 2024–2025 on Google Pixel devices (from Pixel 3 to Pixel 8), provides detailed performance metrics for various levels of hardware support (Software, TEE, StrongBox). The most significant performance penalties are observed in asymmetric-key operations, which are computationally intensive.

There is significant variance in the execution time to generate the RSA-2048 key across isolation levels. For example, a Software Keystore key is generated in 0.21 seconds. TEE-based operations generate in 1.93 seconds for an amount of time. When Secure Element (StrongBox/Titan M2) runs the code, the slow operation requires 9.22 seconds. This blocks the UI thread for almost 10 seconds, causing regular interactive apps to display an Application Not Responding error message. The operating system terminates applications after a long response time, forcing developers to use background services like WorkManager to generate the keys. This results in added complexity to the application's architecture and more complicated asynchronous operations.

For ECC key generation on curve P-256, the picture is more favorable. Elliptic-curve operations are significantly more efficient than RSA-2048, and although the latency gap between TEE and StrongBox persists, the absolute delays are much lower. This makes ECC a more attractive choice for mobile platforms [14], especially in scenarios that require balancing strong cryptographic assurance with acceptable user-interface responsiveness.

RSA-2048 key generation exhibits a clearly expressed dependence of timing characteristics on the selected level of hardware–software isolation. In Software mode (Software Keystore) it takes about 0.21 seconds, in TEE (TEE Keystore) it takes about 1.93 seconds, in Secure Element (StrongBox/ Titan M2 Keystore) it takes about 9.22 seconds. This is an issue in case of interactive use-case where blocking the UI thread for the duration of nearly 10 seconds is enough to give the user an ANR (Application Not Responding) error. This forces developers to move key generation into a background service such as WorkManager, complicating the architecture and the interaction between components.

For ECC key generation on the P-256 curve, a more favorable picture is observed. Operations on elliptic curves are substantially more efficient than RSA-2048, and although the relative difference in execution time between TEE and StrongBox persists, the absolute latency values are significantly lower. As a result, ECC is regarded as the more suitable choice for mobile platforms [14], particularly in use cases where a high level of cryptographic strength must be maintained alongside acceptable user-interface responsiveness.

For illustrative purposes, Figure 1 presents a diagram constructed from the empirical data of the KeyDroid study, highlighting the magnitude of the latency problem.

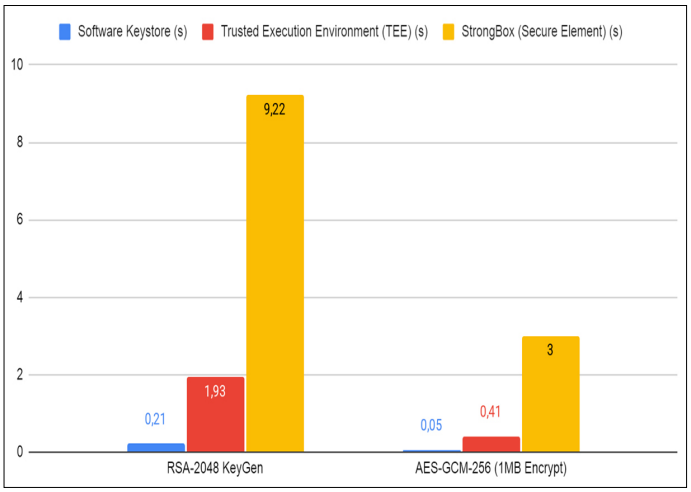

Figure 1. Comparative diagram of cryptographic operation delays in different key storage environments (adapted from [6])

Energy Consumption and Impact on Battery Life

The energy-efficiency dimension of hardware-backed cryptography in mobile devices is often underestimated, despite its decisive role in determining the practical viability of particular solutions. Security microcontrollers such as Titan M2 and Knox Vault are ultra-low-voltage components and, in isolation, consume only milliwatts of power, so their direct contribution to the device's overall energy profile is relatively modest.

The primary issue lies not in the energy cost of the Secure Element itself, but in the systemic overhead incurred at the central processor (AP) level. During long-running operations, such as RSA key generation in StrongBox, which lasts on the order of 9 seconds, the central processor is forced to remain in an active state by a wake-lock mechanism while waiting for the computation to complete and the result to be delivered. This prevents the AP from entering deep-sleep states, leading to a noticeable increase in system-wide power consumption [6]. Against this backdrop, studies have shown that switching from RSA to elliptic-curve schemes (ECC) reduces energy consumption by approximately an order of magnitude, which becomes particularly important when relatively slow Secure Elements are used [1].

Adoption Statistics (Adoption Gap)

Despite the availability of robust hardware security mechanisms, actual usage of StrongBox and Secure Enclave in application scenarios remains limited. An analysis of 490,000 applications from Google Play in 2024 revealed that 56.3% of applications handling sensitive data completely ignore Android Keystore, favoring purely software cryptography or even storing key material in SharedPreferences, which is manifestly incompatible with secure-storage requirements [6]. Only 5.03% of applications explicitly request StrongBox usage via the setIsStrongBoxBacked(true) API, thereby deliberately binding the key lifecycle to a dedicated Secure Element.

These statistics indicate a persistent trend: developers routinely trade security for implementation simplicity and higher performance, attempting to avoid architectural complications associated with asynchronous interactions with a relatively slow hardware module. As a result, the available hardware root of trust remains underutilized, and a substantial proportion of applications relies on vulnerable or suboptimal key-storage and key-processing models.

Threat Analysis and Architectural Resilience

This section synthesizes data on attack vectors and the effectiveness of mitigation measures across the considered architectures.

Attacks on the Power Subsystem (Fault Injection)

The most powerful class of software-based fault-injection attacks occurs when an opponent can force a specific hardware fault at the processor's voltage or frequency level, without requiring physical access to the computer's hardware. This intrusion is accomplished by using privileged access to the computer's system management interfaces. Programmatic access to the relevant drivers and control registers is sufficient.

The CLKSCREW mechanism demonstrates that dynamic voltage and frequency scaling (DVFS) drivers are accessible from the operating-system kernel in many systems [16]. If an adversary obtains root privileges, it can reduce the voltage or increase the frequency of the central processor at critical moments during cryptographic operations in TrustZone. This may cause single-bit faults (bit flips) and result in malformed digital signatures. In the case of RSA/CRT, such a faulty signature can be exploited to recover the private key using the classical Bellcore attack model. Similar strategies have been implemented in Plundervolt and VoltPillager attacks, which target Intel SGX and other TEE implementations [17].

From a hardware-protection standpoint, Knox Vault and Titan M2 architectures were explicitly designed with this class of attacks in mind through physical separation of power domains. Voltage regulators for the central processor are not coupled to the Secure Element power rails, so DVFS manipulations on the AP cannot induce controlled faults within the SE. Additionally, Knox Vault incorporates active glitch detectors that trigger a chip reset when characteristic signatures of voltage glitches are detected. Taken together, these measures make the practical realization of software-driven fault-injection attacks against such solutions infeasible [11].

Microarchitectural Side-Channel Attacks

Although speculative execution is the most crucial technique for the high throughput of modern general-purpose processors, it has also given rise to an entire class of vulnerabilities. Most notably, Spectre and Meltdown, which exploit microarchitectural side channels such as cache effects, branch predictors, and prefetches that are not part of the instruction set architecture.

In the context of TrustZone, this creates a fundamental problem: security-critical code runs on the same pipeline as potentially malicious code in the everyday world. This co-residence enables cache-timing attacks such as TruSpy [18]. In such attacks, an adversary analyzes access latency to shared cache lines to infer the behavior of protected code and, by extension, its secret data.

Around 2025, new threats against Apple Silicon (M and A series) were disclosed, referred to as SLAP and FLOP [19]. These attacks exploit data-memory–dependent prefetchers and, under certain conditions, allow them to read portions of Safari's memory. Critically, the Secure Enclave Processor (SEP) in this architecture is a separate core with its own memory subsystem, which is not implemented as part of the main CPU's speculative execution pipeline, making cryptographic keys and other sensitive artifacts stored in SEP inaccessible to SLAP and FLOP. This empirically reinforces the thesis that physical isolation provides a more robust barrier than purely logical segregation within a single execution pipeline, particularly under the pervasive presence of speculative-execution vulnerabilities.

Biometric Integration and Transaction Protection

Safety guarantees for transactions, e.g., in Apple Pay or Samsung Pay. An example would be if the attacker owned the operating system kernel and changed the payment parameters. He must change what the user sees so that when the user sends $1,000, he actually sends $ 10. They are dangerous because they might appear legitimate to the user, and the interaction seems correct.

This is reduced in Secure Enclave and Knox Vault via the Trusted UI concept, in which the transaction-confirmation screen is created and drawn by the security subsystem directly in a framebuffer to which the operating system has no write access, preventing it from overwriting the display with incorrect information.

The biometric sensor is also connected directly to the Secure Element, bypassing the central processor. Taken together, this creates a protected tunnel from the fingerprint or facial recognition sensor to the banking server, within which the compromised OS is excluded from the trust chain and cannot influence the transaction parameters confirmed by the user.

Risks in the Post-quantum Era

Existing hardware enclaves such as SEP, Titan, and Knox include accelerators for classical cryptographic algorithms, including RSA, ECC, and AES. However, they lack native support for post-quantum cryptography (PQC), specifically lattice-based schemes such as CRYSTALS-Kyber and CRYSTALS-Dilithium.

The key problem is the inability to retrofit already-deployed hardware: physical accelerators cannot be updated post hoc. Implementing PQC algorithms purely in software on relatively weak RISC-V or Cortex-M cores used inside Secure Elements will lead to even greater execution delays than the already observed ~9 seconds for RSA key generation. In aggregate, this creates a risk of accelerated obsolescence for the installed device base within the Harvest Now, Decrypt Later threat model [20], underscoring the need to design new generations of security chips that natively support post-quantum algorithms and include specialized PQC accelerators.

A consolidated analysis of attack classes and the robustness of the considered architectures is provided in Table 2.

Table 2. Consolidated analysis of attack classes and robustness of the considered architectures

Aspect / Dimension	Power-supply fault injection attacks	Microarchitectural side-channel attacks	Transaction / UI manipulation attacks	Post-quantum era risks (PQC)
Threat class	Attacks on the power subsystem (Fault Injection)	Microarchitectural side-channel attacks	Transaction parameter tampering and UI attacks	Long-term cryptographic obsolescence in the post-quantum era
Example attacks	CLKSCREW, Plundervolt, VoltPillager	Spectre, Meltdown, TruSpy, SLAP, FLOP	Substitution of payment amount/recipient in mobile payments	Harvest Now, Decrypt Later
Main vector/ preconditions	Software access to DVFS, ability to control the voltage/frequency of the main CPU	Speculative execution, shared caches, branch predictors, data prefetchers	Compromised OS with the ability to modify data rendered to the user	Lack of hardware support for PQC, limited performance of embedded cores for software PQC implementations
TrustZone (TEE on a shared core)	High vulnerability: TEE shares power domain and DVFS with the everyday world; faults in the AP can induce errors in protected code	Elevated vulnerability: shares pipeline and caches with untrusted code; cache-timing and related attacks (e.g., TruSpy) are feasible	Depends on implementation: without a strictly isolated Trusted UI, the OS can potentially influence displayed transaction parameters	Depends on cryptographic stack: software PQC on the central processor may introduce significant latency and increased resource consumption

Secure Enclave (SEP)	Low vulnerability: separate core and power domain; no direct coupling to AP DVFS	High robustness: separate core and memory subsystem; SEP is not involved in speculative execution of the main CPU	Low vulnerability: Trusted UI is rendered into a dedicated framebuffer controlled by SEP; OS cannot modify the confirmation screen	Classical crypto accelerators (RSA/ECC/AES) but no native PQC; software PQC on a relatively weak core significantly slows operations
Knox Vault / Titan M2	Very low vulnerability: physically separated power rails, integrated glitch detectors, and forced chip reset on anomaly detection	High robustness: secure element is physically separate, does not share cache or pipeline with the everyday world	Low vulnerability: analogous Trusted UI; biometrics and rendering of confirmation screens are encapsulated in a secure tunnel	Similar situation: classical accelerators (RSA/ECC/AES) without PQC; software PQC leads to notable delays and risk of accelerated device obsolescence
Residual risk assessment	Low for dedicated secure elements; significant for TrustZone-only designs	Medium for TrustZone; low for dedicated secure elements	Critical for systems without Trusted UI; low when SE-centric Trusted UI is present	Medium-to-high long-term risk for all current generations of secure elements

Recommendations

Based on the analysis, protection of key material should be organized as a multi-layered construct, with each layer explicitly mapped to its corresponding risk class and latency budget. Long-term identification and root keys should be strictly bound to a hardware root of trust: StrongBox, Secure Enclave, or Knox Vault must be treated as the only permissible loci for their generation and storage, with no copies in REE or TEE memory. At the same time, session and derived keys should be offloaded to faster domains, such as a TrustZone-based TEE or even high-level cryptographic libraries, while strictly limiting their lifetime, the volume of processed data, and the set of operations. This differentiation reduces the potential scope of catastrophic damage in the event of application-level compromise without turning every cryptographic operation into a multi-second dialog with a slow Secure Microcontroller.

Application architecture must be designed from the outset under the assumption that any operation involving hardware-protected keys is inherently slow and potentially unstable in terms of timing. Key generation, long signature chains, and operations involving StrongBox or SEP must be moved unconditionally to background threads or dedicated components isolated from the UI thread. This implies the pervasive use of asynchrony primitives, coroutines, Future/Promise constructs, reactive streams, job queues, as well as precise SLA specifications for cryptographic operations: timeouts, retry policies, and degradation mechanisms in case the hardware module is unavailable. In practice, a hardware-backed keystore should be treated as a remote service with non-zero latency rather than a conventional synchronous function call.

From an algorithmic perspective, it is justified to deliberately orient the cryptographic stack around elliptic-curve schemes and modern symmetric primitives with favorable performance profiles on constrained cores. Continued use of RSA in mobile scenarios should be viewed as technical debt: even with hardware accelerators, latency and energy costs remain significantly higher than for ECC P-256 and comparable schemes. In practice, the choice should shift toward ECC for authentication and session key establishment, and, looking ahead, toward hybrid schemes combining classical and post-quantum cryptography. When designing new protocols, it is essential to ensure that cryptographic algorithms can be swapped without migrating the entire application context, that is, to externalize cryptographic logic into explicitly versioned, configurable layers.

A critical organizational step is the introduction of mandatory hardware security checks at early stages of application initialization. For all operations involving sensitive objects, explicit policies should be defined: only if StrongBox/SEP is available, TEE acceptable, prohibited without hardware-backed storage. The results of isHardwareBacked, isStrongBoxBacked, and similar functions should be treated as critical security indicators. When working in a non-secure environment, applications should inform the user that security is lower than ideal or disable the use of security-sensitive features (e.g., payment token storage or long-term session key storage). The result is a set of rules for developers and end-users: a hardware root of trust is no longer just a marketing phrase, but a requirement.

Use platform-level Trusted UI and hardware-anchored biometric mechanisms wherever possible in payment, biometric, and signing flows for legally relevant documents. Instead of custom confirmation screens, which inevitably fall under the control of a potentially compromised OS, system-provided dialogs and authentication flows should be used, where UI rendering and biometric processing occur within the Secure Element. Any attempt to secretly enhance UX via custom interfaces wrapped around payment or cryptographic operations effectively reintroduces the classic

class of transaction parameter-substitution attacks and must be treated as an architectural flaw.

Finally, systematic preparation for post-quantum transformation is required, treating current Secure Elements as a finite resource. In long-lived systems, security must not be rigidly tied solely to specific hardware primitives, and the architecture must provide a path for gradual transition to PQC: reserving fallback, albeit slower, software implementations; designing protocols with hybrid keys and certificates; and embedding mechanisms for identity migration without compromising the integrity of trust chains. Where justified by the business model, it is already advisable to classify data by temporal sensitivity: information retained over decades should be encrypted in such a way that its security does not depend exclusively on classical algorithms implemented in the current generation of SEP or Titan, whereas short-lived artifacts may continue to rely on existing accelerators to ensure acceptable performance and energy characteristics.

CONCLUSION

The results collectively confirm the initial hypothesis that protection models relying solely on logical isolation within ARM TrustZone and related TEE designs have inherent limitations. A systematic analysis of the Apple Secure Enclave, Samsung Knox Vault, and Google Titan M2 architectures demonstrates a qualitative shift in the industry from shared microarchitectural resources to physically segregated security subsystems with dedicated cores, memory, and power domains. Against the backdrop of documented side-channel and fault-injection attacks (CLKSCREW, Plundervolt, TruSpy, SLAP/FLOP), such Secure Elements, augmented by Trusted UI mechanisms and hardware biometrics, provide a substantially more resilient root of trust for key storage and for securing payment and authentication scenarios than traditional TrustZone-based solutions that share caches, pipelines, and DVFS domains with untrusted code. This provides both empirical and conceptual grounding for the conclusion that physical isolation of the critical security perimeter is the only adequate response to the evolution of low-level attacks.

At the same time, performance and energy-efficiency analysis reveal a fundamental dichotomy between the level of hardware protection and the acceptability of latency for application scenarios. Migration to StrongBox and discrete Secure Elements entails an order-of-magnitude increase in asymmetric operation latency (up to ~9 seconds for RSA-2048 key generation on Titan M2), leading to UI thread blockage, prolonged active-state retention of the central processor, and, consequently, higher energy consumption. This forces the relocation of operations involving hardware-backed keys to asynchronous execution paths and encourages reliance on more efficient ECC schemes, relegating RSA to the role of technical debt on mobile platforms. Empirical data on the minimal real-world adoption of StrongBox and Keystore in the Android ecosystem show that developers systematically sacrifice the hardware root of trust in favor of architectural simplicity and minimal latency, perpetuating vulnerable storage models (including SharedPreferences) and turning the presence of Secure Enclave/StrongBox into a formal rather than an operational security feature. The multi-level key-role separation scheme proposed in this work (long-term keys confined to SE/SEP/Knox Vault, session keys in faster domains) and the requirement to conceptualize cryptographic operations as interactions with a remote service together form a practical methodological framework for overcoming this adoption gap.

Finally, consideration of post-quantum cryptography perspectives exposes long-term risks of accelerated obsolescence for the current generation of Secure Elements, which are architecturally tailored to classical RSA/ECC primitives and lack native PQC support. The impossibility of updating existing accelerators, coupled with the high computational cost of lattice-based algorithms on lightweight SE cores, amplifies the threat of Harvest Now, Decrypt Later scenarios and necessitates security stacks, protocols, and key-management policies that are explicitly designed with migration in mind. In this context, the recommendations formulated in the article, from normative requirements for hardware support (StrongBox/SEP/Knox Vault) for critical features and strict enforcement of Trusted UI usage to the design of hybrid, versioned cryptographic stacks that support phased PQC transition, constitute a coherent, scientifically grounded agenda for the evolution of mobile platforms. Thus, the study not only captures the current state of hardware security mechanisms and their vulnerabilities but also delineates the direction for further development of Secure Enclave architectures and related ecosystems from both technological and applied perspectives.

REFERENCES

1. Hopalı E, Vayvay Ö, Kalender ZT, Turhan D, Aysuna C. How Do Mobile Wallets Improve Sustainability in Payment Services? A Comprehensive Literature Review. Sustainability. 2022 Dec 9; 14(24): 16541.
2. Muñoz A. Cracking the Core: Hardware Vulnerabilities in Android Devices Unveiled. Electronics [Internet]. 2024 Oct 31; 13(21): 4269. Available from: https://www.mdpi.com/2079-9292/13/21/4269
3. Lindenmeier C, Hammer A, Gruber J, Röckl J, Freiling F. Key extraction-based lawful access to encrypted data: Taxonomy and survey. Forensic Science International Digital Investigation. 2024 Sep 1; 50: 301796.
4. Paju A, Javed MA, Nurmi J, Savimäki J, McGillion B, Brumley BB. SoK: A Systematic Review of TEE Usage for Developing Trusted Applications. ARES '23: Proceedings of the 18th International Conference on Availability, Reliability and Security. 2023 Aug 29; 1–15.

5. Gu C, Zhang Y, Abu-Ghazaleh N. I Know What You Sync: Covert and Side Channel Attacks on File Systems via syncfs. arXiv. 2024 Nov 16.

6. Blessing J, Anderson RJ, Beresford AR. KeyDroid: A Large-Scale Analysis of Secure Key Storage in Android Apps. arXiv. 2025 Jul 10.

7. Noseda M, Zimmerli L, Schläpfer T, Rüst A. Performance Analysis of Secure Elements for IoT. IoT. 2021 Dec 21; 3(1): 1–28.

8. Wang KC. ARM TrustZone and Secure Operating Systems. In: Embedded and Real-Time Operating Systems. Springer; 2023. p. 793–845.

9. Li X, Tyagi A. Cross-World Covert Channel on ARM Trustzone through PMU. Sensors. 2022 Sep 28; 22(19): 7354.

10. Apple. Secure Enclave [Internet]. Apple. 2024 [cited 2025 Nov 9]. Available from: https://support.apple.com/en-en/guide/security/sec59b0b31ff/web

11. Samsung Knox. Knox Vault [Internet]. Samsung Knox. 2025 [cited 2025 Nov 10]. Available from: https://docs.samsungknox.com/admin/fundamentals/whitepaper/samsung-knox-mobile-security/system-security/knox-vault/

12. Google. Pixel 6: Setting a new standard for mobile security [Internet]. Google Security Blog. 2021 [cited 2025 Nov 11]. Available from: https://security.googleblog.com/2021/10/pixel-6-setting-new-standard-for-mobile.html

13. Google Cloud. Titan hardware chip [Internet]. Google Cloud. 2025 [cited 2025 Nov 12]. Available from: https://docs.cloud.google.com/docs/security/titan-hardware-chip

14. Mahardika MY, Triayudi A. Comparative Analysis of Performance of the Encryption and Decryption Times of Cryptography. SAGA: Journal of Technology and Information System [Internet]. 2025 Jan 18 [cited 2025 Jun 3]; 2(3): 304–10. Available from: https://www.researchgate.net/publication/388147669_Comparative_Analysis_of_Performance_of_the_Encryption_and_Decryption_Times_of_Cryptography

15. Adeniyi AE, Jimoh RG, Awotunde JB. A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security. Computers & Electrical Engineering. 2024 May 28; 118: 109330.

16. Muñoz A, Ríos R, Román R, López J. A survey on the (in) security of trusted execution environments. Computers & Security. 2023 Jun;129:103180.

17. Joy A, Soh B, Zhang Z, Parameswaran S, Jayasinghe D. Physical and Software Based Fault Injection Attacks Against TEEs in Mobile Devices: A Systemisation of Knowledge. arXiv. 2024 Nov 22.

18. Guo P, Yan Y, Zhu C, Wang J. Research on Arm TrustZone and Understanding the Security Vulnerability in Its Cache Architecture. Lecture Notes in Computer Science. 2021; 12382: 200–13.

19. Chen B, Wang Y, Shome P, Fletcher CW, Kohlbrenner D, Paccagnella R, et al. GoFetch: breaking constant-time cryptographic implementations using data memory-dependent prefetchers. SEC '24: Proceedings of the 33rd USENIX Conference on Security Symposium [Internet]. 2024 [cited 2025 Nov 17]; 1117–34. Available from: https://dl.acm.org/doi/10.5555/3698900.3698963

20. Benitez C. Mapping Quantum Threats: An Engineering Inventory of Cryptographic Dependencies. arXiv. 2025 Sep 29.