Google

August 2026

High Assurance Digital Credentials on Android

A Framework for Hardware-Backed Security

Executive Summary

This white paper outlines Android's open architecture for supporting high-assurance digital credentials.

We show how complex identity challenges—authenticity, impersonation, and privacy—can be reduced to secure key storage and access control. Within Android's Keystore architecture, the StrongBox specification provides the highest security, backed by a certified, tamper-resistant Secure Element. Finally, we demonstrate how this model provides verifiable cryptographic guarantees that can justify the highest regulatory standards, including the Wallet Secure Cryptographic Application (WSCA) requirements in eIDAS 2.0.

01 An Open Foundation for Digital Identity

A digital credential is more than a digital replica of a physical card; it is a cryptographic proof of identity designed to enable secure, privacy-preserving interactions across both physical and remote environments.

By prioritizing local, hardware-backed security, a digital credential actively improves upon the inherent limitations of legacy physical documents:

- Strict Data Minimization: It replaces the "all-or-nothing" data exposure of a plastic card with strict data minimization. Utilizing modern mechanisms like Selective Disclosure and Zero-Knowledge Proofs (ZKP), users can securely share only what is necessary for a transaction.
- Mitigation of Physical Loss: While a misplaced physical ID instantly exposes a user's sensitive personal data, a digital credential remains cryptographically protected and inaccessible behind the device's lockscreen and hardware authentication barriers.
- Authentic Remote Presentation: It leverages cryptographic signatures to guarantee the authenticity of the identity data presented online, replacing easily spoofed and unreliable image captures of physical cards.

Beyond replacing physical cards, digital credentials face a new operational reality: as autonomous software agents assume a more active role across the digital ecosystem, relying parties need to distinguish genuine human actions from automated processes, while users need guarantees that software cannot silently act on their behalf. To satisfy both demands, the industry requires hardware-rooted architectures capable of anchoring cryptographic presentations to explicit human-in-the-loop confirmation.

Within this evolving landscape, Android's goal is to provide a universal, hardware-backed foundation for digital credentials that is natively integrated into the user's device. "Universal" in this context means providing a flexible, scalable architecture that accommodates a wide spectrum of use cases—from offline proximity sharing to remote online authentication—while simultaneously satisfying the diverse security requirements of different issuers, scaling to meet the highest security standards required for government-issued credentials, and offering developers a single API that abstracts OEM-specific hardware complexities.

This foundation is built on several core pillars: it must be privacy-preserving, capable of offline presentation, secure against advanced local and remote attacks, and cost-effective. Crucially, this capability must be democratized through an open API. Exposing these hardware-backed security features to applications through standard Android APIs prevents ecosystem fragmentation. While it was developed to meet the security criteria of government-issued credentials, this open infrastructure allows any developer—whether in banking or enterprise—to leverage the same high-assurance hardware security for custom credentials. This ensures that any wallet application can generate the hardware-backed attestation required for an issuer to verify trust before provisioning a credential. Consequently, users are never locked into a single, proprietary device brand or vendor-specific application just to manage their digital identity.

However, for digital credential issuers—particularly government issuers—to trust this open ecosystem, the underlying platform is expected to achieve high-assurance independent security certification. Evaluating a mobile platform to ensure it meets these rigorous demands requires a precise understanding of its architecture. The following sections outline the fundamental security goals for digital credentials, and detail how Android's reusable security primitives are engineered to successfully satisfy them. By clarifying this architecture, this paper aims to inform how these systems should be evaluated—providing a framework for scoping high-assurance security certifications that focus strictly on core, verifiable security guarantees without imposing unnecessary architectural constraints.

The digital identity landscape encompasses diverse electronic ID (eID) issuers with varying security goals and operational use cases. On Android, our proposed solution for this highest tier is StrongBox, a dedicated Secure Element (SE) architecture explicitly designed to meet this bar. Rather than a domain-specific implementation, StrongBox is a mature, general-purpose hardware security architecture whose cryptographic isolation is implemented and enforced directly by the SE vendor. By establishing a reliable, cross-OEM standard, StrongBox eliminates the need for wallet providers to develop bespoke, hardware-tied security applets, significantly reducing the friction and cost of high-assurance compliance.

02 Security Foundations for Digital IDs

Before detailing the Android-specific architecture, it is necessary to establish the security requirements of any high-assurance digital credential system.

Securing a digital identity requires solving distinct challenges across the credential lifecycle: guaranteeing data authenticity, preventing impersonation, and preserving user privacy.

Digital credential systems solve these challenges by reducing them to secure key storage and access control. For example, a valid signature over a real-time presentation session proves the use of the possessed device-bound key. If the hardware ensures that only the legitimate user can access that key, the signature also proves user authorization. The following subsections detail how each core security problem is reduced to these key-storage and cryptographic requirements.

2.1. Document Authenticity

In a digital identity framework, the initial challenge is preventing counterfeiting—ensuring an attacker cannot forge or modify the data on the ID. This is a well-understood problem resolved at the application and protocol layers using standard, issuer-signed data formats and models, such as ISO/IEC 18013-5 (mdoc) [1]and IETF SD-JWT VC [2].

Crucially, this anti-counterfeiting measure relies entirely on the issuer's signature generated in the cloud. Once provisioned, the authenticity of the information is decoupled from the device's local hardware security.

2.2. Impersonation Prevention

Preventing impersonation—ensuring an unauthorized actor cannot successfully present an authentic credential—represents the primary device-side security challenge that must be solved. The severity of this threat depends entirely on the presentation context:

- Proximity Presentation: For face-to-face transactions, credential cloning is not a critical threat. A human verifier can easily visually inspect the user's face against the photo contained within the issuer-signed ID.
- Remote Presentation: The risk of impersonation increases drastically in remote transactions, where human visual verification is impossible.

[1] ISO/IEC 18013-5 (mdoc): https://www.iso.org/standard/69084.html
[2] IETF SD-JWT VC: https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/

Securing digital identities against remote impersonation requires a layered approach to risk reduction, explicitly recognizing the distinct boundaries between cryptographic enforcement and human-centric verification. This is critical because local malware can enable impersonation that is virtually undetectable to a remote relying party. To defend against impersonation, the system must incorporate robust user authentication (such as PINs or biometrics) as the definitive way to prove eID holder identity.

2.3. Hardware-Enforced Device Binding

To mitigate the risk of remote impersonation, the foundational security layer must prevent credential cloning. If an attacker can extract the cryptographic keys associated with a digital credential, they can duplicate the identity across arbitrary devices, defeating all remote verification mechanisms.

Device binding is a well-defined technical problem with verifiable cryptographic solutions. The objective is to bind the digital credential to a cryptographic private key generated and permanently retained within secure hardware (such as a Trusted Execution Environment (TEE) or SE). While TEE is globally available on Android, dedicated tamper-resistant SEs provide the highest level of cryptographic key protection available on consumer hardware. By retaining keys inside a certified SE microchip, the architecture guarantees that private key material cannot be copied, exported, or extracted, even in the presence of advanced software exploits or direct physical attacks. For deployments requiring the highest level of security, this paper will focus exclusively on dedicated, tamper-resistant hardware backends for the remainder of the document.

From the issuer's perspective, verifying that the cryptographic key originates strictly from within certified secure hardware successfully solves the device binding challenge. This establishes a robust foundation for remote trust, ensuring that a valid credential presentation can only be presented by the intended physical device.

2.4. Remotely-Verifiable Key Provenance

During the provisioning phase, a government issuer requires verifiable proof of device binding before provisioning can occur. This verification encompasses two distinct cryptographic assurances delivered via key attestation, allowing the issuer to verify the hardware's security posture remotely without relying on the integrity of the wallet application software.

2.4.1. Hardware-Rooted Attestation

To solve the foundational device binding prerequisite, an issuer must have verifiable proof that the private key material is held exclusively within certified secure hardware (proving 2.3 Device Binding). The secure hardware generates a cryptographically signed statement attesting the key's provenance and hardware enforcement policies. This hardware-rooted attestation chains back to a recognized Certificate Authority (CA), verifying the hardware's security posture remotely.

2.4.2. Application Binding

Beyond verifying the hardware backend, an issuer must ensure that credentials are only provisioned to an authentic, untampered wallet application. The architecture must also guarantee that once provisioned, access to the cryptographic key is restricted to the owning wallet application.

To enforce these protections robustly, atomic, native binding of the cryptographic key to the owning application package is essential. While hardware-rooted attestation (Section 2.4.1) verifies hardware provenance, application binding extends the attestation claim to include application package metadata. Note that this binding is only as secure as the operating system itself (detailed in Section 3). This native integration provides the remote issuer with verifiable assurance that the credential key is provisioned to, and exclusively managed by, the authentic wallet application package.

2.5. Privacy and Cryptographic Unlinkability

To prevent surveillance, high-assurance credential architectures must guarantee privacy by design. From a user's perspective, everyday presentations of their digital credentials must remain unlinkable across different relying parties. Even when presenting to the same relying party across multiple sessions, the transactional evidence must not enable correlation—for example, by using an ephemeral key to ensure that cryptographic signatures are mathematically unlinkable, even if the relying party retains all transaction data (independent of user-identifying payload attributes, which are minimized separately via selective disclosure; see Appendix A).

Additionally, where keys are used is critical for transaction privacy. While remote (server-side) credential vaults can securely protect keys, generating signatures requires contacting a remote server during presentation. This network transaction inherently exposes real-time usage patterns—such as the time, date, and IP address of the transaction—to a centralized entity, creating a vector for user tracking. In contrast, device-local hardware-backed vaults generate cryptographic signatures entirely on-device without network dependency, preserving the privacy-preserving properties of local, peer-to-peer credential presentation.

Similarly, establishing hardware-enforced device binding (Section 2.3) and remotely-verifiable key provenance (Section 2.4) must not enable cross-credential correlation or issuer collusion. During the enrollment phase, the issuance of different credentials to the same user must remain uncorrelated. A user must also be allowed to operate multiple independent wallets on the device, with the same strict unlinkability guarantees preserved between different wallet applications.

To satisfy these rigorous privacy goals while providing cryptographic assurance of device binding, the SE can natively support unlinkable device binding. This requires making it statistically difficult for a remote server to determine whether different credentials originate from the same device. The SE can achieve this either by sharing a single attestation private key across a large batch of devices to obscure individual identity, or by generating credential keys that are mathematically random and unlinkable. This prevents both issuers and relying parties from correlating transactions, proving that robust hardware security and user privacy are mutually supportive goals.

NOTE: While this whitepaper focuses primarily on the hardware security architecture of digital credentials, user privacy is equally critical. Appendix A provides a conceptual overview of the different cryptographic privacy-preserving attestation models, including their respective trade-offs and design considerations.

03 Android Keystore Architecture

Modern mobile operating systems like Android already provide robust facilities to allow any wallet application to address these challenges through cryptographic primitives.

Rather than forcing wallet developers to build bespoke, isolated cryptographic stacks, Android's Keystore architecture provides an open, standardized foundation for high-assurance digital identity.

By explicitly separating the application domain from cryptographic execution, Android Keystore natively maps the core requirements of digital identity to verifiable architectural boundaries. Specifically, the underlying hardware backends—with StrongBox positioned as the most secure dedicated implementation—provide secure cryptographic primitives, while the Keystore system service handles logical routing and application isolation at the OS layer.

Rather than relying on a single monolithic mechanism, Android employs a layered defense-in-depth model that distributes specific security and privacy guarantees across specialized tiers. Each layer depends on the integrity and enforcement of the underlying stack to operate reliably. This separation of concerns ensures that a compromise at the higher Android OS layers cannot breach the foundational cryptographic protection of the secure hardware.

Beyond mitigating advanced security threats, this layered architecture addresses a fundamental operational challenge in the mobile ecosystem: hardware diversity. Across distinct device models and diverse silicon architectures, Android Keystore abstracts the underlying physical complexity through open APIs for all applications.

NOTE:
Digital credential protocols (such as ISO/IEC 18013-5 mDL and W3C SD-JWT) execute primarily at the application layer (see Figure 1), delegating generic cryptographic operations to a "Secure Area". Android Keystore provides this Secure Area functionality in the TEE and SE, enabling wallet applications to implement digital credentials in an interoperable way.

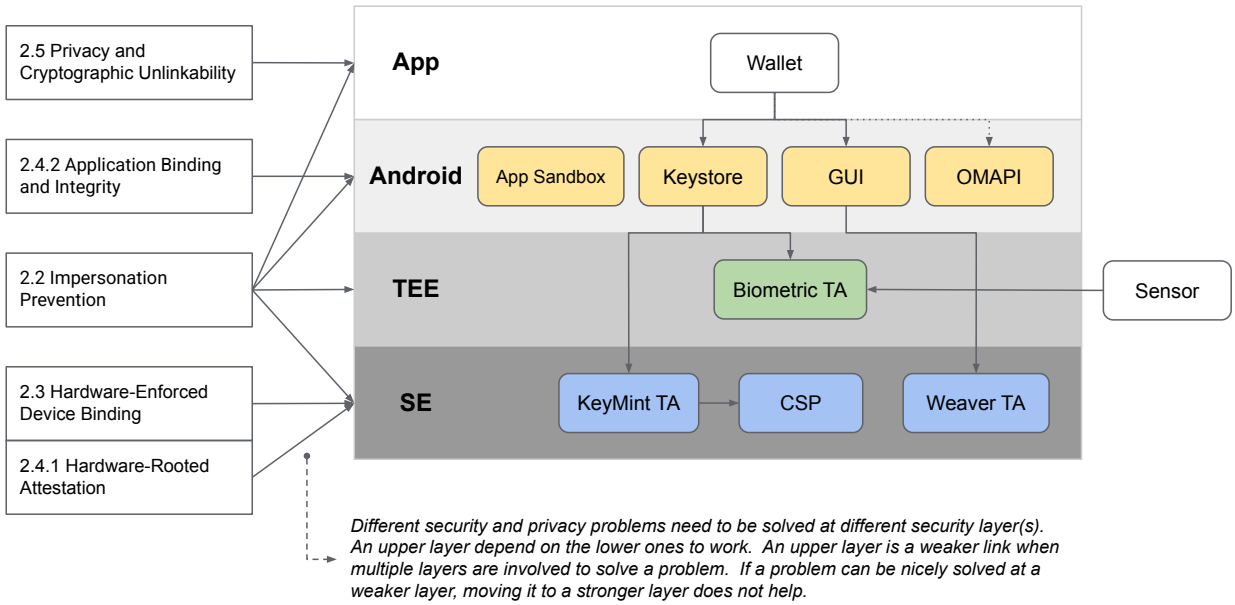

Figure 1: Mapping from generic digital ID problems to solutions in the Android architecture, layered by security level, with distinct component colors representing different provider/trust entities.

3.1. The Keystore Orchestration

Android Keystore manages cryptographic keys for applications. To enforce this boundary at the platform layer, the Keystore system service relies on native OS kernel mechanisms—specifically Linux process isolation and Binder IPC caller identification—to ensure applications can only access their own keys. Through this system service, any application can define security policies on a per-key basis—such as requiring PIN, password, or biometric authentication, or authentication timeouts—to mitigate unauthorized usage (see Section 2.2).
To enforce these policies, Keystore interfaces with the secure hardware through a standardized Hardware Abstraction Layer (HAL). This architecture splits the security model into logical and cryptographic domains. While Keystore manages the storage of encrypted key blobs and routes application requests at the OS layer, the KeyMint HAL standardizes how these requests are passed to the secure hardware. KeyMint then acts as the cryptographic execution boundary, ensuring that raw keys are only decrypted and used within the secure hardware, and that hardware-bound policies—such as cryptographic usage constraints and user authentication requirements—are enforced directly within the secure hardware.

3.2. StrongBox KeyMint: The High-Assurance Keystore Backend

StrongBox is the highest-security hardware backend for Android Keystore, engineered to fulfill the hardware-enforced device binding requirements defined in Section 2.3. While standard Android devices typically run KeyMint inside a TEE on the main processor, StrongBox requires a dedicated, physically isolated SE. Both implementations must meet Android's KeyMint hardware specifications, but StrongBox provides the highest level of cryptographic isolation and tamper resistance.

To achieve this high-assurance designation, the Android Compatibility Definition Document (CDD) mandates that the StrongBox hardware backend must be certified against recognized Secure IC Protection Profiles (such as BSI-CC-PP-0084-2014 or BSI-CC-PP-0117-2022) or evaluated by an accredited laboratory incorporating a High attack potential vulnerability assessment. This dedicated secure microchip operates entirely independently of the main processor, providing physical tamper-resistance and electrical isolation against side-channel or physical glitching attacks. By establishing this definitive physical boundary, StrongBox KeyMint acts as the exclusive authority for cryptographic execution and policy enforcement.

- Key Offloading: To support an unlimited number of credentials despite the Secure Element's memory constraints, KeyMint offloads encrypted keys to the filesystem. When a key is generated, KeyMint wraps (encrypts) the raw key material using a volatile wrapping key derived from a hardware Master Key and application-specific metadata. During a cryptographic operation, KeyMint unwraps the blob into an isolated memory slot for the duration of the task, destroying the plaintext key from memory immediately after.

- Verifiable Key Attestation: KeyMint provides unforgeable cryptographic proof of a key's origin. Upon generation, KeyMint signs an attestation certificate using a device-unique attestation key, proving to remote relying parties that the private key was generated within and is confined to the secure hardware.

- Policy and Authorization Enforcement: KeyMint binds usage policies—such as cryptographic purpose, mandatory user authentication, or authorization timeouts—to the key during generation. For keys requiring authentication, KeyMint gates execution until it verifies the Hash-based Message Authentication Code (HMAC) of a Hardware Authentication Token (HAT). Because this HMAC is verified using a secret shared exclusively between the TEE and the Secure Element, the main operating system cannot forge or bypass these authentication checks.

3.3. Trusted User Authenticators

Android provides hardware-backed user authenticators as a generic platform capability. In the context of digital credentials, these trusted authenticators mitigate the impersonation risks described in Section 2.2 by generating a cryptographically signed HAT upon verifying an authentication factor.
To prevent replay attacks if the Rich OS is compromised, the secure hardware cryptographically binds the HAT to the user's current transaction. KeyMint will not execute operations until it verifies a fresh, transaction-specific token.

3.3.1. Lock Screen Knowledge Factors (LSKF)

LSKF verification relies on hardware-enforced rate limiting to prevent brute-force attacks by an attacker attempting arbitrary inputs through the entry interface. This policy is strictly enforced by a process called Weaver which is inside of the Secure Element. The matching result is translated by the Gatekeeper Trusted Application (TA) executing in the TEE, which mints a HAT to authorize a KeyMint operation in the Secure Element. Crucially, the threat model assumes the attacker has not compromised the Rich OS GUI; if the OS is compromised, an adversary can record and replay the credentials. Because credential entry occurs in software, the Rich OS remains the dominant vulnerability surface in knowledge factor authentication.

NOTE: Launched Android devices with StrongBox achieve hardware-backed LSKF rate limiting via Weaver SE and TEE Gatekeeper HAT translation. Because PIN entry occurs in the Rich OS, removing Gatekeeper from the LSKF trust path hardens the hardware boundary without altering overall end-to-end security assurance.

3.3.2. Biometric Verification

For biometric verification, the physical sensor communicates directly with an isolated secure hardware environment (typically in a TEE) to execute matching. This architecture isolates raw biometric processing from the Rich OS, which merely routes the resulting cryptographically signed HAT to KeyMint to authorize a cryptographic operation, such as generating a signature. Because this hardware-authorized operation provides cryptographic proof of user presence rather than a simple boolean confirmation, it supports wallet applications across both local and cloud-based architectures. For an on-device eID, the authorized key directly signs a presentation session; for a remote cloud HSM, it signs a challenge proving to the remote server that local biometric verification occurred.

3.4. Factory Reset

To guarantee user privacy and unlinkability across device ownership lifecycles (see Section 2.5), the platform implements a factory reset enforced by secure hardware when a device is decommissioned or transferred to a new owner. During a factory reset, the Secure Element destroys the Master Key by rotating it to a newly generated random secret. Because all offloaded credential key blobs are encrypted using keys derived from this Master Key, this destruction instantly renders all existing blobs cryptographically un-decryptable. This achieves a verified cryptographic wipe of all stored credentials, ensuring that no residual key material can be recovered or utilized to track users across different device ownership lifecycles, even in the event of direct physical access to the flash storage.

04 Verifiable Trust: Proving Key Provenance to the Issuer

The security strength of a local hardware boundary is only as valuable as its ability to prove its integrity to a remote relying party.

In the context of secure digital credentials, issuers require cryptographic proof of a key's provenance and isolation properties before provisioning. Android Keystore addresses this challenge through a scalable, privacy-preserving attestation infrastructure designed to operate across a diverse OEM ecosystem.

4.1. Key Attestation as the Foundation of Trust

To satisfy the remotely verifiable key provenance requirements established in Section 2.4, a remote issuer must verify the integrity of the wallet application and the device state before provisioning a credential. Without cryptographic proof of key provenance, an attacker could provision credentials to a simulated environment or a modified wallet application.

Key Attestation provides unforgeable cryptographic proof of key origin and platform state directly from secure hardware. For digital identity provisioning, credential issuers depend on three core properties embedded within the attestation certificate:

- Hardware Key Custody: Verification that the private key resides securely within dedicated StrongBox hardware, providing hardware-enforced protection against secret extraction or software emulation.
- Application Binding: Cryptographic confirmation of the calling application's identity via the ASN.1 DER-encoded AttestationApplicationId. This binds the key directly to the genuine wallet app package name, its version code, and the cryptographic SHA-256 digests of its official developer signing keys.
- Platform Operating Integrity: Evidence derived from Android Verified Boot (AVB) establishing a continuous chain of trust from the Boot ROM, proving the operating system partitions remain genuine.

By evaluating these specific attestation fields, remote issuers can verify the platform security posture before provisioning digital credentials across diverse hardware.

4.2. Remote Key Provisioning (RKP) as the Trust Anchor

While local Key Attestation defines the cryptographic evidence required by issuers, distributing and certifying these keys at global scale requires a dedicated infrastructure. RKP serves as this trust anchor, coordinating a multi-stage provisioning and certification flow that traces the chain of trust from secure hardware to the user application while preserving user privacy (see Section 4.3):

- Trust Anchor Setup: To ensure private keys remain within the secure hardware boundary, each device is provisioned with a Unique Device Secret (UDS) which can be input into a key derivation function to produce a device key pair. The UDS public key is recognized by the RKP server (through manufacturer's public key registration or a certificate chain), establishing the foundational trust anchor.
- On-Device Key Pooling & Remote Certification: The secure hardware locally pregenerates generic attestation key pairs. To certify them, the Secure Element generates a Certificate Signing Request (CSR) containing these keys and signs it with the UDS. An Android system daemon (rkpd) retrieves this CSR and transmits it to the remote RKP server, which validates the signature, issues certificates, and returns them to the device for local pooling.
- Application-Instance Assignment: When an application requests a new credential key, Keystore dynamically assigns a certified attestation key from the pool, scoping it uniquely to that specific application instance. KeyMint uses this assigned key to sign the key attributes, producing the final leaf of the Android-rooted certificate chain while preventing relying parties from linking transactions across different applications on the same device.

A noteworthy architectural property of this design is that the Secure Element and its applets remain completely server-agnostic. The secure hardware contains no hardcoded endpoints, certificates, or network logic pointing to Google or any other remote entity; it merely signs the standard CSR using its local UDS. The routing decision is handled entirely in Android userspace by rkpd.

In standard Google Mobile Services (GMS) deployments, rkpd routes the CSR to Google's RKP server. The RKP server verifies the request against the device identifiers and public keys registered by OEMs and SE vendors during factory provisioning, issuing attestation certificates that chain up to Android's publicly trusted root CA. To provide independent evidence of this operational trust and governance, Google's RKP server infrastructure undergoes independent WebTrust for Certification Authorities audits.

NOTE: Prior to RKP, devices relied on Factory Key Provisioning (FKP), which required distributing secret Android attestation keys from Google to OEMs for injection directly on the factory production line. This legacy approach was operationally complex, exposing the ecosystem to risks of secret key mismanagement during manufacturing. RKP addresses these manufacturing complexities by never transmitting cryptographic secrets outside of the secure hardware that manages them.

Under modern architectural evolutions of RKP, if the RKP server independently validates a signature from a recognized discrete SE vendor's root key, it embeds the validated_attested_entity extension into the attestation certificate. This evolution provides credential issuers with verifiable confirmation that the secure hardware vendor attested to the key provenance directly. Furthermore, this same vendor-backed attestation can be used to prove that the credential key resides within a certified EAL4+/AVA_VAN.5 stack (see Section 5.1 for architectural details).

From an Android ecosystem perspective, this RKP infrastructure functions as a format translator, mapping manufacturer claims directly into X.509 certificates without altering their underlying security properties. This establishes a trust abstraction layer. Rather than forcing third-party developers or remote issuers to manage custom CA certificates and establish bespoke trust relationships with each secure silicon manufacturer, this model decouples credential issuers from the heterogeneous hardware landscape. Issuers evaluate a single, common attestation format, leveraging Android Key Attestation to obtain consistent and verifiable security claims despite vast physical hardware diversity across Android OEMs.

To provide structural clarity, the attestation certificate can be modeled across a clear break-down of security properties:

Attestation Trust Layer	Verified Scope / Asserted Attributes	Custodian & Provenance Source
Top: Runtime Status	Platform bootloader lock state, verified boot state, OS patch level	Android Verified Boot (AVB)
Middle: Device Baseline	Device brand, model parameters, initial hardware specifications	OEM factory production pipeline
Bottom: Hardware Key Provenance	Physical silicon isolation properties, cryptographic execution location	Secure Element vendor silicon factory root (via `validated_attested_entity in modern` RKP) or OEM production root (in standard RKP)

4.3. Universal Privacy and Unlinkability

An attestation framework must provide verifiable security without enabling systemic device profiling. Unlike legacy FKP's reliance on a permanent batch key to achieve ecosystem k-anonymity (where K >= 100,000), RKP enforces privacy dynamically by provisioning attestation certificates scoped strictly per-app to prevent cross-app correlation.

Furthermore, RKP's short certificate lifetimes force keys to rotate regularly (currently every 14 days), preventing long-term correlation profiles while enabling the RKP backend to stop key replenishment for compromised devices. Crucially, while key pool replenishment requires communicating with a remote server, the protocol is explicitly structured to eliminate correlation risks. The RKP backend operates statelessly with respect to user identity; it validates trust anchors and issues certificates without retaining persistent signing records, building long-term device profiles, or tracking usage across or within any applications.

05 Architectural Scoping of the EU AVA_VAN.5 WSCA Boundary

Under the Commission Implementing Regulation (EU) 2024/2981 [1] for the European Digital Identity Wallet (EUDIW), the wallet architecture must meet the Level of Assurance (LoA) High.

While the overall wallet application itself must undergo certification, its core cryptographic capabilities rely on a certified platform foundation—the Wallet Secure Cryptographic Application (WSCA) and Wallet Secure Cryptographic Device (WSCD), which are required under European certification schemes to achieve Common Criteria (ISO/IEC 15408) [2] EAL4+ augmented with AVA_VAN.5. In this section, we examine the platform-level WSCA and WSCD certification boundary. Our goal is to define a practical Target of Evaluation (TOE) centered on core security primitives, avoiding architectural complexity that offers no tangible security benefit.

5.1. The WSCA Core: Security Functional Requirements

The core purpose of the WSCD is to protect cryptographic credentials from unauthorized extraction and clone attacks. Within this scope, the "WSCA Core" defines the specific Security Functional Requirements (SFRs) that must be implemented with high-assurance security properties. These primitives include secure key generation (ensuring keys are generated inside the hardware boundary and never exposed), cryptographic key isolation (preventing physical or logical extraction of keys during execution), and key attestation (generating evidence of key provenance and hardware attributes). These requirements align with the critical security properties in Section 2, ensuring that cryptographic credentials can be securely generated on-device, protected against physical and logical extraction (see Section 2.3), and remotely verified by credential issuers (see Section 2.4).

The most viable path to meet the WSCA/WSCD certification requirements is by deploying a minimal Cryptographic Service Provider (CSP). The CSP provides isolated cryptographic services to its client application (in this case, KeyMint) while strictly preventing any secrets or private keys from leaking to the client. While it is technically possible to certify the entire KeyMint implementation to AVA_VAN.5, doing so would impose a severe long-term recertification and maintenance overhead to the StrongBox vendors that can slow down and add cost to future deployments as the Android platform evolves. To meet EUDIW's high-assurance requirements while avoiding this overhead, one effective approach is to utilize a split-certification model for StrongBox KeyMint. Under this model, the EAL4+/AVA_VAN.5 TOE is confined to the SE hardware and the CSP, which implements the

[1] Commission Implementing Regulation (EU) 2024/2981: https://eur-lex.europa.eu/eli/reg_impl/2024/2981/oj
[2] Common Criteria (ISO/IEC 15408): https://www.commoncriteriaportal.org/

core security primitives—including secure key generation, physical isolation, and RKP attestation (CSR generation and signing). KeyMint acts merely as a client to the CSP, executing high-level policy enforcement outside the core cryptographic boundary. Crucially, the resulting key attestation verifiably reflects that the key was generated and is isolated within the certified EAL4+/AVA_VAN.5 CSP boundary, allowing credential issuers to confirm these properties remotely. By offloading these core primitives to the stable CSP, the rest of the KeyMint implementation is freed from the EAL4+ evaluation burden, allowing the platform to undergo frequent, iterative improvements and updates without the prohibitive cost of recertifying the core cryptographic boundary. A standardized, platform-provided implementation ensures that this core cryptographic engine is integrated directly into the platform's hardware root of trust and subject to unified platform security updates.

5.2. Justifying the Decoupled User Authentication Model

A central challenge in EUDIW certification is determining where user authentication (PIN/pattern or biometrics) must be verified. We argue that, from a security evaluation and threat-model perspective, user authentication must be kept outside the "WSCA Core" boundary. A security chain is only as strong as its weakest link. Verifying user presence and authorization fundamentally requires trusting the physical user interface (GUI) for PIN entry and the biometric sensors and matching pipelines (managed by the TEE) to verify biometrics. The extensive complexity of modern user authentication interfaces and biometric processing stacks has historically made AVA_VAN.5 evaluation infeasible.

The threat models for cryptographic key protection and user authentication are fundamentally distinct:

- Key Protection (WSCD / WSCA Core): The WSCA Core protects credential private keys against physical extraction, electrical fault injection, side-channel leakage, and offline cloning attacks. Defending against these threats requires hardware-enforced silicon isolation and AVA_VAN.5 vulnerability analysis against high attack potential.
- User Authentication & Binding: User authentication protects against live unauthorized presentation or transaction invocation. Live matching is delegated to dedicated authenticators—such as Weaver (implemented in the SE to enforce exponential timeout backoffs against brute-force attacks) or biometric TAs in the TEE—with the option to invalidate auth-bound credentials if new biometrics are enrolled. On devices with multiple user profiles, the platform manages these authenticators as securely for each profile, ensuring credential isolation across accounts.

These authenticators communicate to KeyMint via secure channels (where the trust between components is configured by the OEMs during factory provisioning) using cryptographically signed HATs. Before sending cryptographic requests to the CSP, KeyMint verifies the token's integrity (via HMAC), security level, and freshness, ensuring that the CSP operates according to any specified user authentication condition at the appropriate hardware security level.

This delegated model allows Android Keystore to offer granular flexibility to wallet applications. Apps can configure individual keys with custom authentication requirements—such as requiring user authorization for every single operation (per-transaction binding) or utilizing a brief time-based authorization window to balance security and user experience. While this high-level orchestration and matching execution is vital for operational security, it remains entirely decoupled from the core EAL4+/AVA_VAN.5 WSCA Core cryptographic boundary, keeping the certified TOE minimal and secure.

5.3. Architectural Parity with Remote WSCDs

This split between the core cryptographic boundary (WSCD) and user authentication is not a compromise; it establishes strict architectural parity with widely accepted remote cloud architectures. Under standard eIDAS deployments, remote WSCDs (server-side Cloud HSMs) are certified to AVA_VAN.5 for key protection. However, to authorize a signature execution, these remote HSMs must rely on the user's local, less certified software to capture and route biometrics, PINs, or transaction intent over the network.

Because a remote AVA_VAN.5 HSM architecture delegates user authentication to a local, lower-assurance mobile interface, the same architectural delegation model is equally valid for a local Secure Element. A local SE acting as the "WSCA Core" achieves the same AVA_VAN.5 cryptographic protection while relying on the same TEE and GUI.

06 Conclusion: Empowering the Global Wallet Ecosystem

Preserving user privacy and establishing trust are critical to the widespread adoption of digital identity; privacy-by-design principles are key to addressing the surveillance concerns that prevent many users from transitioning away from physical credentials.

By ensuring that digital identity operates locally and peer-to-peer—mirroring the privacy characteristics of a physical ID card—we prevent centralized tracking of everyday presentations. While meeting the rigorous security requirements of digital identity frameworks presents a complex challenge, it is a fully solvable technical problem that does not compromise user privacy.

On the Android platform, StrongBox provides a production-proven hardware foundation that can be utilized and further enhanced to satisfy high-assurance software certification criteria. This paper defines the minimum necessary security boundary—the WSCA Core—to reduce evaluation and recertification overhead while preserving the platform's overall security posture. By abstracting the complexity of diverse Secure Elements behind the standardized Android Keystore API, this architecture ensures that secure credentials function seamlessly across the entire OEM ecosystem. This canonical approach prevents vendor lock-in, offering users, wallet developers, and issuers alike an interoperable, open, and highly secure environment. Ultimately, this framework demonstrates that public trust, high-assurance security, and user privacy can be successfully harmonized on an open mobile ecosystem.

07 Appendix

Appendix A: Privacy-Preserving Attestation and Unlinkability

In high-security digital credential systems, the requirement for hardware key attestation—proving to remote issuers and relying parties that cryptographic keys reside within certified secure hardware—introduces inherent privacy challenges. Proving key provenance typically requires presenting attestation certificate chains, which can leak static device identifiers or manufacturer batch details, potentially enabling device tracking or correlation.

User privacy in a digital credential system is therefore not solved by any single component; it requires a coordinated defense across the hardware, operating system, wallet application, and presentation protocols. The hardware's primary role is to ensure cryptographic key isolation and prevent the leakage of permanent physical device identifiers into public attestations. The operating system handles application-scoped key allocation and replenishment lifecycles. Meanwhile, transaction-time privacy—such as preventing local readers from tracking the user across multiple visits—is the responsibility of the presentation protocols—such as the ISO/IEC 18013-5 mdoc standard or Selective Disclosure JSON Web Tokens (SD-JWT)—through the use of session-specific ephemeral keys.

As this paper's primary focus is on the device hardware security architecture, this appendix serves to explore the broader privacy design space originating from this key attestation challenge. It outlines how the hardware and platform security primitives described in this paper can support various privacy-preserving models, highlighting the trade-offs of each approach.

Privacy-Preserving Presentation

When credential signatures are generated entirely locally and offline within the secure hardware, the credential issuer is never contacted during verification and receives zero tracking data. To protect the user's identity from the relying party during presentation, protocols leverage data minimization techniques:

- Selective Disclosure: Modern presentation formats (such as ISO/IEC 18013-5 mdoc and SD-JWT) allow the wallet application to disclose only a subset of attributes (e.g., presenting the user's name and portrait, while hiding their residential address) to the verifier, preserving absolute transactional unlinkability.
- Zero-Knowledge Proofs: For attributes requiring conditional verification, the platform can leverage Zero-Knowledge Proofs (ZKPs) (utilizing frameworks like Longfellow ZK [1]). Instead of revealing the raw attribute value, the client generates a ZK proof demonstrating that the attribute satisfies a specific predicate (e.g., proving the user is over 18 without sharing their actual birthdate).

NOTE: Longfellow ZK is an open-source framework designed to run zero-knowledge proofs utilizing only standard cryptographic primitives—such as SHA-256 and ECDSA—allowing ZK verification to be deployed without requiring new algorithms to be recertified in secure hardware. The underlying cryptography is published [2] in IACR Communications in Cryptology, has completed independent third-party security audits, and is actively in standardization at IETF [3] and ETSI.

Privacy-Preserving Enrollment and Issuance

When a user sets up a new digital credential in their wallet, they must obtain cryptographic proof of key provenance to convince the issuer that their keys are securely bound to genuine secure hardware. However, this enrollment process should not allow the issuer to link multiple credentials to the same device or track the user across different apps. Android Keystore resolves this during the enrollment journey through the RKP architecture. Keystore scopes RKP attestation keys strictly per-application package—rotated every 14 days—ensuring different credentials issued to different apps on the same device remain uncorrelated. This design narrows the potential correlation window down to a single 14-day rotation epoch: colluding issuers could only link keys if they are generated within the same application during the same 14-day window. For deployments requiring even stricter privacy guarantees that completely eliminate this temporal window, two alternative design choices can be explored depending on the system's trust model:

[1] Longfellow ZK: https://github.com/google/longfellow-zk
[2] Anonymous credentials from ECDSA: https://eprint.iacr.org/2024/2010
[3] IETF: https://datatracker.ietf.org/doc/draft-google-cfrg-libzk/

Attestation Proxying: If the system permits a model where the wallet provider is trusted with enrollment metadata, the wallet server can act as a proxy. The wallet server verifies the hardware attestation once and issues its own transient certificates for individual credential keys. This completely decouples the device-identifying hardware attestation from the issuer, preventing any issuer correlation.
Decentralized Zero-Knowledge Proofs: If the deployment requirements mandate that the wallet provider must be excluded from the enrollment path entirely (preventing both correlation and centralized tracking by the operator), the client could theoretically verify key provenance via a local zero-knowledge proof of the certification chain. While extending the on-device Longfellow ZK framework to verify full hardware certification chains would require non-trivial research and development, it represents a potential design choice for deployments that require complete independence from the operator.

Appendix B: Local Internal QSCD and Sole Control Parity

B.1. The Missing Piece of eIDAS: Local Internal QSCD

The Qualified Signature Creation Device (QSCD) concept was introduced under the original eIDAS regulation (eIDAS 1.0) to define secure hardware environments capable of creating Qualified Electronic Signatures (QES)—which hold the same legal standing as handwritten signatures. The regulation defines three core architecture types for QSCDs:

1. Remote: Cloud-hosted Hardware Security Modules (HSMs).
2. Local External: Dedicated physical hardware tokens, such as smartcards or USB keys.
3. Local Internal: Secure execution environments integrated directly within consumer devices, such as smartphone secure hardware.

B.2. Architectural Trade-Offs for QTSPs and Users

For Qualified Trust Service Providers (QTSPs) responsible for issuing QES, a local internal QSCD offers a powerful third architectural option alongside remote and local external configurations, each presenting distinct trade-offs:

Remote QSCDs (Cloud HSMs): Like any centralized system, remote solutions depend on network availability and introduce transaction latency. They require QTSPs to operate and scale high-availability cloud HSM infrastructure. To authorize signatures, remote QSCDs rely on network-based Signature Activation Protocols (SAPs)—technically similar to the protocols used by remote WSCA—to verify user intent from the client device.
Local External QSCDs (Smartcards): While highly secure, carrying and tapping physical smartcards introduces significant user friction. This friction limits the practical adoption of QES in everyday consumer scenarios.
Local Internal QSCDs (Smartphone Secure Hardware): This model combines the hardware-level security of dedicated silicon with the low-friction user experience of on-device biometric authentication. It allows QTSPs to issue qualified credentials that can create signatures directly on-device, offering a low-friction user experience, reducing centralized infrastructure overhead, and enabling offline signature capabilities.

Historically, only Remote and Local External solutions have achieved QSCD certification. Local Internal architectures have remained entirely unavailable to the market, primarily because of the lack of defined certification paths and the complexity of certifying consumer hardware platforms to QSCD standards.

B.3. Aligning WSCA/WSCD with Local Internal QSCD

We believe the security narrative and certification framework developed for the WSCA and WSCD under the EUDIW should naturally apply to Local Internal QSCDs.

Crucially, the underlying security stack remains identical: whether a device is used to authorize a remote HSM signature (via a remote WSCA/QSCD SAP) or to generate a signature locally (via a local internal WSCA/QSCD), the system must rely on the user's local device to verify the user's sole control:

- Sole Control Enforcement: The user's explicit intent is captured locally using on-device authentication (such as biometrics or PINs), as detailed in the user authentication discussions in Section 5.2.

- Cryptographic Binding: The authorization token (HAT) is bound to the specific transaction (using AUTH_TIMEOUT = 0 constraints), ensuring that the secure boundary only authorizes the exact signature the user has approved.

Because the same local security stack—protecting biometrics and enforcing transaction-specific authentication—anchors both remote and local internal architectures, a platform that achieves certified WSCA/WSCD status should be recognized as satisfying the technical requirements of a local internal QSCD.

Using the WSCA/WSCD certification boundary as a baseline for Local Internal QSCD allows the industry to align security assurance with consumer usability. Reusing these certified platform primitives can simplify evaluation paths, reduce redundant certification for secure element vendors, and provide a secure, seamless path to QES on personal devices.

Google