Android Key attestation

In Android/eSE/TSM ecosystems, how can we ensure a certain key is generated by a trusted party?

Short answer: The key itself can tell who generated it. We need cryptographic evidence tied to a root we trust. In practice, that means either

1. a hardware-backed attestation that says “this key was generated here, with these properties,” or
2. a certificate chain where our trusted party signs the public key and asserts how/where it was generated.

In Android/eSE/TSM ecosystems, both models exist, but they’re achieved through different rails.

Below is a practical breakdown for Android-based devices, distinguishing between TEE/StrongBox (Android Keystore) and Secure Elements (eSE/UICC/SD card) managed by a TSM.

# 1) If the key is in Android Keystore (TEE or StrongBox)

* Mechanism: Android Key Attestation
* What it gives us:
  + A certificate chain over the key’s public key, ending in an OEM/Google hardware attestation root.
  + An attestation record signed by hardware that includes:
    - origin = GENERATED or IMPORTED (ensures not imported if we require on-device generation)
    - hardware/security level (TEE vs StrongBox)
    - key characteristics (algorithms, sizes, purposes, user auth, rollback resistance, etc.)
    - a nonce/challenge (freshness, anti-replay)
    - device identity fields (depending on keymaster version and settings)
* How to ensure “generated by a trusted party”:
  + Define “trusted party” as “the device’s hardware TEE/StrongBox under attestation keys we trust.” Pin the attestation root(s).
  + Require origin=GENERATED and reject IMPORTED.
  + Verify the full chain and the attestation extensions server-side.
* When to use: We need proof the key was created inside device hardware (not by a remote HSM/TSM) and bound to that device/security level.

# 2) If the key is in a Secure Element (eSE/UICC/SD) managed via a TSM

There isn’t a single Android-wide attestation API for arbitrary eSE keys. Assurance relies on GlobalPlatform (GP) mechanisms, the TSM trust model, and/or an application-level PKI.

Common, robust patterns:

## A) On-card key generation + certificate from a trusted CA

* Flow:
  1. TSM (or service provider) opens a GP Secure Channel (SCP02/03 or SCP11) to the applet’s Security Domain on the SE. Only a party that already holds the SD’s keys (Issuer or delegated SP) can do this.
  2. Applet generates the key pair on-card (private key never leaves the SE).
  3. Applet produces a CSR (or returns the public key) with identifying info/policies.
  4. A trusted CA (owned by your trusted party) signs the public key, issuing an X.509 certificate that states policy (e.g., “key was generated on-card under SP X”).
  5. The cert is stored on the SE or in backend; relying parties validate this chain to the CA we trust.
* What we can assert:
  + The key pair was generated on the SE (not injected).
  + The trusted party (our CA) vouches for it via the certificate.
* How we verify:
  + Validate the certificate chain to our trusted CA.
  + Check certificate policy OIDs/constraints that encode “on-card generation” and applet identity/version.
* Pros: Strong, portable proof anchored in our CA. Private key stays in hardware.
* Cons: Requires applet support for CSR and our PKI/CA process.

## B) Trusted party (TSM/HSM) generates and injects the key under GP secure wrapping

* Flow:
  1. The “trusted party” generates the key in its HSM.
  2. It injects the key into the SE over a GP Secure Channel using PUT KEY or proprietary secure import, wrapped/encrypted under SD transport keys.
  3. The public key is certified by the same trusted CA (or signed metadata) stating “generated by TSM HSM.”
* What we can assert:
  + The key was generated by that trusted HSM/TSM and delivered securely to the SE (no plaintext exposure outside secured environments).
* How we verify:
  + Again, via the certificate chain/policy that says generation took place in that HSM.
  + Operationally, we rely on TSM audit/logs and the fact that only entities with SD keys could import.
* Pros: Lets the trusted party control key generation centrally.
* Cons: Private key existed outside the SE (albeit inside an HSM); some regulations prefer pure on-card generation.

## C) GlobalPlatform Delegated Management and DAP signatures (provenance of who did what)

* Use GP Delegated Management (DAP) and associated signatures to prove that only a specific service provider (or TSM) loaded an applet or executed management commands on the SE.
* While DAP proves who authorized/installed content, it does not by itself prove key origin. Combine with (A) or (B) and our PKI policies to get clear provenance.

## D) SE attestation (where supported)

* Some SE vendors/applet stacks support attestation of keys (similar in spirit to Android Key Attestation, but not standardized across all eSEs).
* If available, we can obtain an attestation statement signed by an SE vendor/issuer root that claims “this key was generated inside this SE/applet.”
* Verify the attestation chain to a root we trust.

# Bringing it together

How to “ensure a certain key is generated by a trusted party”

* Define exactly what “trusted party” means in our context:
  + The device’s secure hardware (TEE/StrongBox) itself? Use Android Key Attestation and require origin=GENERATED.
  + The TSM/HSM? Have it sign the public key (certificate) asserting generation in its HSM; or rely on GP logs plus certificate policy.
  + The eSE itself? Use on-card keygen plus either
    1. a CA certificate stating on-card generation, or
    2. an SE vendor/issuer attestation if supported.
* Require a verifiable artifact:
  + For Keystore: hardware attestation chain.
  + For eSE: X.509 certificate chain with explicit policy OIDs, or SE attestation evidence, chained to a root we pin.
* Pin the trust anchors:
  + OEM/Google attestation roots for Android Keystore.
  + Your organization’s CA or the SE issuer’s CA for eSE/app-level PKI.
* Reject imports if we require on-device/on-card generation:
  + Keystore: check origin=GENERATED.
  + eSE: policy in cert must state on-card generation; don’t accept keys without that policy.
* Maintain operational controls:
  + Use GP SCP03/SCP11 for provisioning and rotate SD keys.
  + Keep TSM actions in an HSM with audit logs.
  + Use DAP/Delegated Management to restrict who can load/manage applets.

# Example decision matrix

* We want “device-generated and hardware-bound” for app login: Use Android Keystore + Attestation, verify origin=GENERATED, pin roots.
* We want “issuer-controlled keys in an eSE applet” (e.g., payments, transit): Use TSM to trigger on-card keygen, get a CSR, issue a cert from issuer CA with policy “on-card gen,” distribute the cert. Relying parties validate to issuer CA.
* We want “TSM-generated keys, provably from our HSM”: Generate in HSM, inject under GP secure channel, issue a cert with policy “HSM-generated,” and pin that CA.

# Key takeaways

* The guarantee comes from cryptographic evidence tied to a root we trust, not from the key itself.
* In Android Keystore, that evidence is hardware attestation.
* In Secure Element/TSM workflows, that evidence is typically an X.509 certificate (and/or SE attestation) plus GlobalPlatform-controlled provisioning paths.
* Encode our requirements (on-card vs HSM generation, algorithms, key usage) in verifiable policy and reject anything that lacks the right evidence.