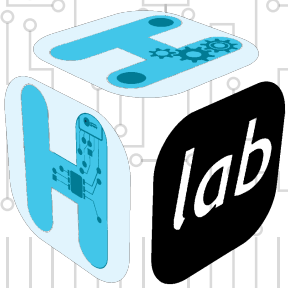
H2LAB-STIG-SECCOMP
08-2026

Embedded systems: a methodology for selecting security components



STIG H^{2}Lab

Information

Created in 2020, H2Lab is a non-profit organization dedicated to designing, developing, and sharing open-hardware and open-source solutions for experimentation around embedded systems. Its main areas of work are:

- Design of hardware platforms for hardware and software analysis
- Development of open alternatives to proprietary, often opaque tools
- Publication of technical guides and recommendations to promote best practices in security, privacy, and sustainability across the ecosystem of connected devices and embedded systems

Support for researchers, teachers, and students through the provision of tools, educational content, and training.
Academic partnerships to encourage sharing of resources and knowledge.

Revision history

Version	Date	Author	Changes
1.0	August 2026	H2Lab	Initial version of the guide

Contents

Information i

1 Glossary and acronyms 1
- 1.1 Glossary . . . 1
- 1.2 List of acronyms . . . 2

2 Introduction and scope 5
- 2.1 Purpose of this guide . . . 5
- 2.2 Intended audience . . . 5
- 2.3 Technical scope and assumptions . . . 5
  - 2.3.1 Equipment type and context . . . 5
  - 2.3.2 Life cycle considered . . . 6
  - 2.3.3 Threat assumptions . . . 6
  - 2.3.4 How to read this guide . . . 6
- 2.4 Threat model, attacker profiles and security objectives . . . 6
  - 2.4.1 Main threats . . . 7
  - 2.4.2 Attacker profiles . . . 7
  - 2.4.3 Security objectives . . . 7
- 2.5 State of the art of hardware security technologies . . . 8
  - 2.5.1 Secure Elements (SE) . . . 8
  - 2.5.2 Hardware Security Engines (HSE) . . . 8
  - 2.5.3 Trusted Platform Modules (TPM) . . . 8
  - 2.5.4 Trusted Execution Environments (TEE) . . . 8
- 2.6 Navigation guide . . . 8

3 Mechanism selection by use case 10
- 3.1 Which mechanism for authenticating the electronic board . . . 10
- 3.2 Which mechanism for authenticating the GPP software at boot . . . 10
- 3.3 Which mechanism for authenticating application transactions . . . 11
- 3.4 Practical selection rule . . . 11
- 3.5 Reinforced selection guidance . . . 11
  - 3.5.1 What these works confirm . . . 11
  - 3.5.2 Weaknesses and trade-offs to take into account . . . 12
  - 3.5.3 Derived decision rules . . . 12
  - 3.5.4 Acceptance checklist before the final choice . . . 13

4 Authenticating the electronic board 14
- 4.1 Threat and risk analysis . . . 14
  - 4.1.1 Mapping of attacks, attackers and controls . . . 15

4.2 Expected security functions . . . 15
4.3 Typical attacks and countermeasures . . . 16
4.3.1 Board cloning and substitution . . . 16
4.3.2 Firmware injection and personalization tampering . . . 16
4.3.3 Physical faults and side channels . . . 16
4.3.4 Replay, maintenance abuse and chain-of-trust bypass . . . 16
4.4 Trust bootstrapping principle . . . 16
4.5 Security specifications for the implementation . . . 17
4.6 Configurable components (FPGA) in the chain of trust . . . 18
4.6.1 The bitstream is a firmware . . . 18
4.6.2 Known attacks and limits of native protections . . . 18
4.6.3 The FPGA as a root-of-trust substrate . . . 18
4.7 Industrial checklist and summary test suite . . . 19
4.7.1 Checklist of industrial-side actions . . . 19
4.7.2 Summary test suite . . . 19

5 Authenticating the general-purpose processor (GPP) software at boot 20
5.1 Threat and risk analysis . . . 20
5.1.1 Mapping of attacks, attackers and controls . . . 20
5.2 Software chain of trust . . . 21
5.3 SE/HSE versus TPM . . . 21
5.4 Security specifications for the implementation . . . 21
5.4.1 Software image verification policy . . . 22
5.4.2 Measurement and attestation of the software state . . . 22
5.4.3 Signing key life cycle . . . 22
5.5 Industrial checklist and summary test suite . . . 22
5.5.1 Checklist of industrial-side actions . . . 22
5.5.2 Summary test suite . . . 22

6 Authenticating application transactions after boot 24
6.1 Threat and risk analysis . . . 24
6.1.1 Mapping of attacks, attackers and controls . . . 24
6.2 Post-boot objective . . . 24
6.3 Role split between SE and TEE . . . 25
6.4 Security specifications for the implementation . . . 25
6.4.1 Services to authenticate after boot . . . 25
6.4.2 Application key usage policy . . . 26
6.4.3 Traceability and audit of cryptographic operations . . . 27
6.4.4 Acceptance criteria and minimal tests . . . 27
6.5 Industrial checklist and summary test suite . . . 28
6.5.1 Checklist of industrial-side actions . . . 28
6.5.2 Summary test suite . . . 28

7 Provisioning secrets and identities in production 30
7.1 Threat and risk analysis . . . 30
7.1.1 Mapping of attacks, attackers and controls . . . 30
7.2 Expected security functions . . . 31
7.3 Manufacturing identities and operational identities . . . 32
7.4 Security specifications for the implementation . . . 33

7.5 Industrial checklist and summary test suite . . . 33
7.5.1 Checklist of industrial-side actions . . . 33
7.5.2 Summary test suite . . . 33

8 Updating firmware in the field (OTA/DFU) 35
8.1 Threat and risk analysis . . . 35
8.1.1 Mapping of attacks, attackers and controls . . . 35
8.2 Recommended update architecture . . . 36
8.2.1 Signed manifest and metadata . . . 36
8.2.2 A/B partitioning and recovery . . . 37
8.2.3 Anti-rollback and consistency with secure boot . . . 37
8.2.4 Protection of the channel and of the local DFU flow . . . 37
8.3 Fleet rollout and governance . . . 37
8.4 Security specifications for the implementation . . . 37
8.5 Industrial checklist and summary test suite . . . 38
8.5.1 Checklist of industrial-side actions . . . 38
8.5.2 Summary test suite . . . 38

9 Attesting, monitoring and retiring devices in operation 39
9.1 Threat and risk analysis . . . 39
9.1.1 Mapping of attacks, attackers and controls . . . 39
9.2 Remote attestation architecture . . . 39
9.2.1 Model and evidence . . . 40
9.2.2 Governance of the reference values . . . 41
9.2.3 Reaction policy . . . 41
9.3 Monitoring and incident response . . . 41
9.3.1 Minimal security telemetry . . . 41
9.3.2 Fleet-scale correlation . . . 41
9.3.3 Incident response . . . 41
9.4 End of life, RMA returns and ownership transfer . . . 42
9.5 Industrial checklist and summary test suite . . . 42
9.5.1 Checklist of industrial-side actions . . . 42
9.5.2 Summary test suite . . . 42

10 Executing cryptographic algorithms on SE, TPM and TEE 44
10.1 Choosing between SE, TPM and TEE . . . 44
10.2 Hardware vs software acceleration . . . 45
10.2.1 Performance/consumption benchmarks: integrated SE vs MCU library . . . 45
10.2.2 Decision criteria for operation placement . . . 45
10.2.3 Impact on battery autonomy . . . 45
10.2.4 Latency introduced by inter-chip exchanges . . . 45
10.3 Natively supported operations . . . 45
10.3.1 Symmetric encryption . . . 46
10.3.2 Asymmetric encryption . . . 46
10.3.3 Hash functions . . . 46
10.3.4 Random number generation . . . 46
10.4 Protecting data at rest . . . 46
10.4.1 Recommended encryption architecture . . . 47
10.4.2 Erasure and end of life . . . 47

10.4.3 Acceptance rules . . . 47
10.5 APIs and programming interfaces . . . 47
10.5.1 GlobalPlatform: APDU commands for Java Card SE . . . 47
10.5.2 TPM 2.0 Library . . . 47
10.5.3 PSA Crypto API (ARM) . . . 47
10.5.4 TrustZone-M: secure calls to a Cortex-M TEE . . . 48
10.6 Software integration patterns . . . 48
10.6.1 Wrapper abstraction layer . . . 48
10.6.2 Error and timeout handling . . . 48
10.6.3 Logging and audit of sensitive cryptographic operations . . . 48
10.7 Practical summary . . . 48
10.8 Industrial checklist and summary test suite . . . 48
10.8.1 Checklist of industrial-side actions . . . 48
10.8.2 Summary test suite . . . 48

A Migration to post-quantum cryptography 50
A.1 Quantum threat and migration urgency . . . 50
A.2 Post-quantum standards . . . 50
A.3 ANSSI roadmap . . . 51
A.4 Cryptoagility and practical implications for embedded systems . . . 51

B Physical interfacing and communication protocols 52
B.1 Bus choice and attack surface . . . 52
B.2 Electrical constraints and physical security . . . 52
B.3 Debug and validation . . . 52

C System integration and operational considerations 53
C.1 Host microcontroller choice and PCB layout . . . 53
C.2 Development chain and regulatory context . . . 53
C.3 Maintenance in operational conditions (MCO) . . . 54

D Certification strategy and normative traceability 55
D.1 Choosing an evaluation scheme . . . 55
D.2 Mapping the security objectives to the frameworks . . . 56
D.3 Building the audit evidence file . . . 56

E Normative references and specifications 58

License 63

1
Glossary and acronyms

1.1 Glossary

Attack surface. Set of physical, logical and organizational interfaces exploitable by an attacker (debug ports, internal buses, update paths, cloud, maintenance), a notion formalized in threat-modeling and risk-management approaches [5, 60, 53].

RoT (Root of Trust). Minimal set of trusted components (hardware and software) on which the whole verification chain depends. The NIST frameworks for firmware protection and platform integrity make it a baseline prerequisite [47, 43].

SE (Secure Element). Secure component, resistant to physical attacks, used to store secrets and execute cryptographic operations under a strict security policy [25, 59, 17].

TPM (Trusted Platform Module). Standardized module for machine identity, integrity measurement (PCRs), sealing and remote attestation [30, 66, 43].

TEE (Trusted Execution Environment). Isolated execution environment within the SoC, separating a secure world from a normal world, with attack surfaces that are mostly micro-architectural and software-based [8, 35, 29].

Attestation. Cryptographic proof mechanism conveying the state of a component or platform to a verifier [63, 43].

Cryptographic agility. Ability to replace algorithms, key sizes and security profiles without re-architecting the product, a key checkpoint for the post-quantum transition [57, 6, 22].

Chain of trust. Sequence of integrity and authenticity verifications from the root of trust up to the application software components.

Secure Boot. Cryptographic verification of every boot stage to prevent the execution of unauthorized code.

Anti-rollback. Mechanism preventing the installation of an older, vulnerable version of a firmware or software component.

Provisioning. Set of security initialization operations performed during manufacturing and deployment (identity, keys, certificates, policies).

Sealing. Cryptographic binding of a secret to a measured platform state (e.g. PCR values), to prevent its reuse outside the expected context.

Side channel. Information leakage through physical quantities (power consumption, timing, radiation, faults) rather than through the intended logical interface.

Fault injection. Deliberate perturbation (glitch, voltage, clock, EMI) aiming to trigger an exploitable faulty behavior.

Remote attestation. Remote verification of a device's state by an external verifier, based on signed evidence and measurements.

HNDL (Harvest Now, Decrypt Later). Attack strategy consisting of collecting encrypted data today in order to decrypt it later with superior cryptanalytic means (notably quantum computers).

Cryptographic hybridation. Combination of a classical mechanism and a post-quantum mechanism (signature or key establishment) so that security is preserved if either scheme is weakened.

TCB (Trusted Computing Base). Hardware and software set that must remain trustworthy to guarantee the security of the system.

TOCTOU (Time Of Check To Time Of Use). Class of vulnerabilities where the verified state differs from the state actually used, due to a change between check and execution.

Post-quantum cryptoagility. Ability to introduce, replace or remove PQC primitives without redeveloping the whole platform.

1.2 List of acronyms

ANSSI. Agence nationale de la sécurité des systèmes d'information (French national cybersecurity agency)

APDU. Application Protocol Data Unit (smart card command/response format)

API. Application Programming Interface

BIOS. Basic Input/Output System

CI/CD. Continuous Integration / Continuous Delivery

CPA. Correlation Power Analysis (side-channel attack)

CRA. Cyber Resilience Act (EU Regulation 2024/2847) [24]

CRQC. Cryptographically Relevant Quantum Computer

DFA. Differential Fault Analysis

DFU. Device Firmware Update

DH. Diffie-Hellman

DPA. Differential Power Analysis (side-channel attack) [34]

DRBG. Deterministic Random Bit Generator

EAL. Evaluation Assurance Level (Common Criteria) [17]

ECC. Elliptic Curve Cryptography

ECDH. Elliptic Curve Diffie-Hellman

ECDSA. Elliptic Curve Digital Signature Algorithm

ENISA. European Union Agency for Cybersecurity

FIPS. Federal Information Processing Standards (e.g. FIPS 140-3, FIPS 203, FIPS 204) [49, 54, 55]

GPP. General Purpose Processor

HSE. Hardware Security Engine

I2C. Inter-Integrated Circuit

IEC. International Electrotechnical Commission

ISO. International Organization for Standardization

JTAG. Joint Test Action Group (debug/test interface)

KDF. Key Derivation Function

MAC. Message Authentication Code

MCU. Microcontroller Unit

MITM. Man-In-The-Middle

ML-DSA. Module-Lattice Digital Signature Algorithm (FIPS 204 standard)

ML-KEM. Module-Lattice Key-Encapsulation Mechanism (FIPS 203 standard)

MPU. Memory Protection Unit

NIST. National Institute of Standards and Technology

OTA. Over-The-Air (remote update)

PCR. Platform Configuration Registers (TPM) [30]

PQC. Post-Quantum Cryptography

PSA. Platform Security Architecture

PUF. Physical Unclonable Function [38]

RBAC. Role-Based Access Control

RNG. Random Number Generator

RSA. Rivest-Shamir-Adleman

RSSI. Responsable de la sécurité des systèmes d'information (chief information security officer)

SBOM. Software Bill of Materials

SCP. GlobalPlatform Secure Channel Protocol [25]

SCP03. Version SCP03 of the GlobalPlatform secure channel protocol

SLH-DSA. Stateless Hash-Based Digital Signature Algorithm (FIPS 205 standard)

SoC. System on Chip

SPI. Serial Peripheral Interface

SSI. Sécurité des systèmes d'information (information systems security)

SWP. Single Wire Protocol

TEE. Trusted Execution Environment

TLS. Transport Layer Security

TOCTOU. Time Of Check To Time Of Use

TPM. Trusted Platform Module

TRNG. True Random Number Generator

UART. Universal Asynchronous Receiver-Transmitter

UEFI. Unified Extensible Firmware Interface

2
Introduction and scope

2.1 Purpose of this guide

This guide provides a complete method for selecting, integrating and auditing hardware security building blocks (SE, TPM, TEE, HSE) in embedded products. The objective is twofold: reduce technical risk (cloning, key extraction, boot compromise, application fraud) and demonstrate compliance with the applicable security frameworks [5, 53, 31, 24].

The document is deliberately operational: each chapter links technical choices to threats observed in the state of the art (side channels, fault injection, micro-architectural attacks, supply-chain attacks), then maps the controls to recognized standards and guides [34, 13, 33, 36, 52, 4].

2.2 Intended audience

This guide is intended for:

- security architects and product architects
- hardware, firmware and embedded software teams
- information security officers (RSSI/CISO) and compliance teams
- technical auditors and evaluators of connected products

It applies to consumer products as well as industrial and critical environments, with the assurance level adapted accordingly (e.g. EAL, FIPS, IEC 62443) [17, 49, 27].

2.3 Technical scope and assumptions

2.3.1 Equipment type and context

The scope covers constrained embedded systems (MCUs, application microprocessors, heterogeneous SoCs), connected objects, edge gateways and industrial equipment with confidentiality, integrity, availability and traceability needs. The targeted profiles include low-power, long-lifetime architectures exposed to the *Harvest Now, Decrypt Later* risk for long-lived data [40, 6].

2.3.2 Life cycle considered

The guide addresses the life cycle end to end:

- design (threats, requirements, architecture)
- industrialization (provisioning, personalization, testing)
- deployment (identity, enrollment, commissioning)
- operation (monitoring, key rotation, incident response)
- maintenance (patching, secure update, decommissioning)

This approach is aligned with the life-cycle security requirements of the NIST and European frameworks [52, 24, 31].

2.3.3 Threat assumptions

The threat assumptions include:

- an opportunistic attacker with local access to the product
- an experienced attacker with laboratory-grade equipment
- a supply-chain attacker (modified firmware, cloned component, compromised CI/CD)
- a remote attacker exploiting software or protocol vulnerabilities

The attack families taken into account are detailed from the literature: DPA/CPA [34, 14], fault injection [13, 12], speculative-execution attacks [33, 36], TEE/SGX attacks [15, 69], replay and bus interception [25, 47].

2.3.4 How to read this guide

The guide can be read along three axes:

- **Architecture:** trust principles and technology choices
- **Integration:** implementation schemes, interfaces, key policies
- **Audit/compliance:** expected evidence, acceptance criteria, normative mappings

2.4 Threat model, attacker profiles and security objectives

The guide relies on an explicit threat model in order to avoid a purely technical reading of hardware security mechanisms. Threats are considered as chained attack scenarios, combining physical access, software compromise, poor secret management and lack of traceability.

2.4.1 Main threats

The threats taken into account are the following:

- **Platform identity theft:** cloning, substitution of boards or components, contamination of the personalization chain
- **Boot and firmware compromise:** injection of unauthorized code, rollback, modification of the chain of trust
- **Extraction of secrets and sensitive assets:** root keys, application secrets, provisioning data, certificates
- **Tampering with critical operations:** command forgery, replay, transaction spoofing, loss of integrity
- **Maintenance and debug abuse:** service mode opening, untraced access to test interfaces, exfiltration through side channels

2.4.2 Attacker profiles

The guide formally distinguishes several attacker profiles:

- **AM-01 – Opportunistic local attacker:** limited physical access, standard tools, primarily aiming at secret extraction or simple bypasses
- **AM-02 – Malicious maintainer or compromised subcontractor:** access to maintenance, provisioning or diagnostic operations
- **AM-03 – Experienced laboratory attacker:** advanced hardware and software means, able to exploit side channels and physical faults
- **AM-04 – Hybrid supply-chain/field attacker:** combination of production-chain compromise and field intervention to obtain durable persistence

2.4.3 Security objectives

Technical choices must be evaluated against explicit security objectives:

- **OS-01 – Confidentiality of critical assets:** prevent exploitable extraction of secrets, keys and sensitive images
- **OS-02 – System integrity:** guarantee that no unauthorized modification can alter execution or the chain of trust
- **OS-03 – Access control to sensitive functions:** restrict boot, maintenance, provisioning and attestation operations to authorized entities
- **OS-04 – Resistance to basic physical attacks:** reduce the feasibility of fault, side-channel or direct-manipulation attacks
- **OS-05 – Traceability and audit:** produce exploitable evidence for monitoring, investigation and compliance

2.5 State of the art of hardware security technologies

2.5.1 Secure Elements (SE)

SEs are designed to protect secrets: anti-tamper, compartmentalized storage, key-usage control and cryptographic operations restricted by policy [25, 59, 17]. They are suited to long-term uses (identity, payment, signature, IoT credentials) when physical extraction is a credible threat.

SE variants (Java Card applets, proprietary firmware, open hardware) mainly differ in the programming model, portability and security governance. Open solutions, for instance architectures based on OpenTitan or industrial approaches such as Tropic Square, increase design auditability and traceability, but require rigorous management of the build chain and formal verification [37, 65, 52].

2.5.2 Hardware Security Engines (HSE)

HSEs are secure blocks integrated into SoCs that offer cryptographic primitives, key storage and low-latency secure boot services. Their main characteristic is SoC integration; their limit is the dependency on the SoC trust domain, with direct exposure to system attacks depending on the chosen architecture [11, 8, 47].

2.5.3 Trusted Platform Modules (TPM)

TPM 2.0 provides a standardized trust anchor to measure the boot state, seal secrets to a platform state and produce verifiable attestation evidence [30, 66, 43]. It is suited to remote compliance and fleet governance, thanks to its measurement (PCR), policy and quote primitives [30, 66].

Its main limitation in constrained embedded systems is the software integration cost (command stack, policy management, attestation server orchestration), as well as the latency of some operations compared with purely local paths [66, 51].

2.5.4 Trusted Execution Environments (TEE)

A TEE isolates sensitive processing inside the general-purpose processor (e.g. TrustZone), which enables security services at low additional hardware cost [8, 35]. This approach addresses part of the software threats, but remains exposed to micro-architectural and configuration attacks if hardening is not applied [33, 36, 69].

2.6 Navigation guide

The following table directs the reader from a need to the chapter that addresses it; each use-case chapter remains readable on its own, with its threat analysis, industrial checklist and test suite.

Your need	Where to go
Choosing between SE, TPM, TEE and HSE according to the threat	Chapter 3
Proving the authenticity of the board and its components (FPGA included)	Chapter 4
Guaranteeing software integrity at startup (secure boot)	Chapter 5
Protecting application commands and transactions	Chapter 6
Securing personalization, keys and identities in the factory	Chapter 7
Updating the fleet without opening a breach (OTA/DFU)	Chapter 8
Attesting, monitoring and retiring deployed devices	Chapter 9
Placing and implementing cryptographic operations	Chapter 10
Anticipating the quantum threat and organizing the PQC migration	Appendix A
Choosing buses and hardening physical interfaces	Appendix B
PCB integration and MCO organization	Appendix C
Preparing certification and regulatory compliance	Appendix D

Table 2.1: Navigation guide: from need to chapter

3
Mechanism selection by use case

3.1 Which mechanism for authenticating the electronic board

Here, the goal is a non-clonable, verifiable **hardware platform identity**, bound to the configuration state of the board. The aim is to ensure that the active components taking part in the chain of trust of the final product are all present, intact and authentic.

It is then necessary, at startup, to ensure that these components are able to authenticate each other, which naturally implies using a hardware root of trust (SE/HSE/ROM) to initialize the secrets and the platform measurements [47, 43]. In addition, each authenticating component must be able to ensure the confidentiality and integrity of the secrets tied to its authentication on the board, so that it cannot be substituted through physical modification of the TCB.

Finally, when protection against hardware patches is required, the authentication chain between components must be resistant to replay attacks.

Chapter 4 details the implementation of this use case.

3.2 Which mechanism for authenticating the GPP software at boot

For this need, the **TPM** is appropriate when the main stake is **measurable proof** of the software state (firmware, bootloader, kernel) through PCRs and remote attestation [30, 43]. Rollback and bootkit attacks can be addressed when measurements are historized and checked on the verifier side [47].

The **TEE** can complement or partially replace this function on some platforms, provided a defined hardware RoT exists (immutable ROM, signature verification, sound key management) [8, 47]. Without this foundation, boot attacks can bypass application-level controls.

The **SE** can store verification keys or sign evidence, but it does not natively observe the whole GPP boot chain without explicit orchestration [25, 43].

Chapter 5 details the implementation of this use case.

3.3 Which mechanism for authenticating application transactions

This case targets the continuous protection of business operations (frame signing, command authentication, application data encryption). The **SE** is appropriate when the key must remain non-extractable despite prolonged physical access [25, 17]. The **TEE** is appropriate for fast local decisions and flow control in secure memory [8, 35].

Replay, spoofing and command-forgery attacks require anti-replay mechanisms (nonces, counters, time windows) and evidential logging [61, 53].

Chapter 6 details the implementation of this use case.

3.4 Practical selection rule

- Prioritize **TPM** for platform state measurement and attestation (compliance and remote verification) [30, 43]
- Prioritize **SE** for application key protection, documented physical resistance and extraction threat cases (profiles AM-03/AM-04) [17, 25, 34]
- Use **TEE** for local execution isolation and performance, applying explicit hardening against micro-architectural attacks [8, 33, 36]
- Design **hybrid** (SE + TPM + TEE) when remote-proof, physical secret-protection and latency requirements coexist [47, 27]

3.5 Reinforced selection guidance

The Eurisko project [19] and the WooKey project [7] provide field experience that can be leveraged to arbitrate between **SE**, **TPM** and **TEE/HSE** in a real product, beyond theoretical comparisons.

3.5.1 What these works confirm

- **Value of an external SE for the root of trust and pre-boot authentication.** The Eurisko project [19] highlights a secure boot chain with a pre-boot authentication mechanism relying on an EAL5+ certified component, used both as a root of trust and as a removable authentication element.
- **Defense in depth through explicit compartmentalization.** The WooKey project [7] shows the value of an explicit split between exposed data modules and trusted modules, with MPU isolation and separation of data and authentication paths.

- **Mandatory secure channel with the token.** The WooKey project [7] enforces mutual SoC-token authentication at session start (ECDH-derived session keys, anti-replay), which documents a reduction of the pre-authentication attack surface and of malicious-token scenarios.
- **Update treated as an attack surface to be covered explicitly.** Both approaches insist on the role of the boot/DFU chain, with anti-rollback and explicit cryptographic controls [19, 7, 47].

3.5.2 Weaknesses and trade-offs to take into account

- **SE alone: secret protection, limited system visibility.** An SE protects keys with physical-resistance mechanisms and usage policies, but does not natively provide full measurement of the GPP software state; without a platform attestation mechanism, the level of proof remains partial [17, 25, 43].
- **TEE/HSE alone: performance and integration, but residual SoC exposure.** The WooKey project [7] recalls that some MCU flash/readout protections are not infallible against faults, which maintains a risk on assets stored on the SoC side if the secret is not externalized.
- **Confidentiality without integrity: a trade-off to make explicit.** The WooKey project [7] documents that data-at-rest protection may favor confidentiality (FDE) without guaranteeing integrity, which must then be handled at a higher layer. This choice must be recorded in the risk analysis and compensated by application-level controls.
- **Complexity of update hardening.** The WooKey project [7] discusses TOCTOU and fault-injection scenarios on signature verification, then introduces a token that remains active during the DFU flow to constrain these scenarios. This approach increases the applied controls, at a high design, test and operations cost.
- **Operational dependency on the token.** An external-token architecture introduces availability, logistics and support constraints (loss, replacement, personalization, recovery), to be anticipated from the architecture phase [19, 7].

3.5.3 Derived decision rules

- **High physical-threat context (AM-03/AM-04):** adopt an **SE + TEE/HSE** approach, with strict separation of trust domains and the root secret kept outside the SoC
- **Remote-proof and fleet-governance priority:** add a **TPM** to benefit from standardized measurements and attestation
- **Cost/energy constrained context without formal remote-attestation requirement: TEE/HSE** may suffice for some uses, subject to explicit hardening of the boot chain, debug interfaces and update mode
- **No explicitly covered data-at-rest integrity requirement:** refuse to go to production until an integrity mechanism (AEAD mode or equivalent application-level protection) is defined and tested

3.5.4 Acceptance checklist before the final choice

- Is startup blocked in the absence of valid pre-boot authentication?
- Do the root keys remain outside the untrusted memory domain of the SoC?
- Is the DFU mode protected against rollback, TOCTOU and plausible simple faults?
- Do critical flows have confidentiality **and** integrity, not just encryption?
- Does the operational plan cover token loss/renewal/revocation and supply-chain incidents?

4
Authenticating the electronic board

4.1 Threat and risk analysis

This use case aims to establish a non-clonable hardware identity for the board and to prove that the active components of the chain of trust are present, intact and authentic from startup. The objective is therefore not only to verify software, but also to prevent the substitution of an entire platform, the modification of the personalization chain and the unauthorized use of field equipment.

The dominant threats are the following:

- board cloning or substitution during production, deployment or in the field;
- modification of components, firmware or personalization parameters;
- fault attacks (voltage/clock glitches, power cuts, execution perturbation) targeting the initial verification or key exchanges;
- side-channel attacks aiming to extract root secrets, signatures or provisioning data;
- replay of authentication messages or sessions, including during maintenance or update phases;
- abuse of service, debug or recovery interfaces, which can create a bypass of the chain of trust.

The risk model must distinguish the critical assets (hardware identity, root keys, usage policies, audit log, platform state) and the realistic attack paths according to the attacker profile, the value of the secret and the maintenance constraints of the product in real environments [5, 53]. In practice, the physical threat and the supply-chain threat must be addressed together, because a board that is authentic at personalization time can become untrustworthy if trust is bypassed later.

Attack scenario	Reference threat	AM profiles	OS objectives	Associated controls
Board cloning or substitution in the field	Platform identity theft	AM-01, AM-04	OS-03, OS-05	Hardware identity, authenticity proof, anti-replay
Unauthorized firmware injection through maintenance or provisioning	Boot and firmware compromise	AM-02, AM-04	OS-02, OS-03	Cryptographic verification at boot, version control, blocking on failure
Extraction of root secrets through faults or side channels	Extraction of secrets and sensitive assets	AM-03	OS-01, OS-04	Secret isolation, protected channels, key-usage policies
Replay of a session or authentication proof	Tampering with critical operations	AM-01, AM-03	OS-02, OS-05	Nonces, monotonic counters, time windows, logging
Abuse of service or debug interfaces	Maintenance and debug abuse	AM-02, AM-03	OS-03, OS-05	Mutual authentication, operation traceability, interface hardening

Table 4.1: Mapping of the attacks of chapter "Authenticating the electronic board" to threats, AM profiles and OS objectives

4.1.1 Mapping of attacks, attackers and controls

4.2 Expected security functions

To address this threat, the architecture must integrate several complementary security functions:

- **Hardware root of trust:** an initially trusted component (SE, HSE, immutable ROM or a combination) must make the trust decision at the very beginning of startup.
- **Mutual authentication of active components:** the MCU, the SE, the TPM or the network elements must prove their identity before exchanging secrets or critical commands.
- **Protection of secrets and channels:** inter-component exchanges must guarantee confidentiality, integrity, anti-replay and origin authentication (SCP, TPM sessions, mutual TLS, session derivation).
- **Anti-replay and anti-rollback:** proofs, platform measurements and updates must be bound to a session context and to a minimal acceptable version.
- **Fault tolerance and fail-secure mode:** any detected anomaly must lead to a lockdown or to traced maintenance, rather than to a transition to an ambiguous operational state.
- **Traceability and audit:** sensitive operations must produce exploitable evidence for monitoring, investigation and compliance.

These functions are not mere add-on mechanisms; they form the foundation for a board authentication that is robust, verifiable and resistant to both software and physical attacks [47, 43, 25].

4.3 Typical attacks and countermeasures

4.3.1 Board cloning and substitution

An attacker may try to replace a board with an unauthorized equivalent, to copy its attributes or to use a substitute component during personalization. Protection relies on a hardware identity bound to a non-clonable root, on certificates or cryptographic proofs and on verification mechanisms embedded from the very first boot. This problem is central in secure card architectures and trusted platforms, where physical resistance and personalization governance are described in industrial specifications and security evaluations [25, 17, 38].

4.3.2 Firmware injection and personalization tampering

The personalization chain, maintenance and updates are prime targets. A modified image can be injected, or an old version can be reinstalled to reintroduce vulnerable behavior. Countermeasures include cryptographic verification at every stage, version control, anti-rollback and separation of personalization and test roles. The firmware-hardening and chain-of-trust literature documents this threat, in particular the firmware resiliency and supply-chain security guides [47, 52, 4, 3]. Feedback from secure platforms and research projects, notably the Eurisko project [19] and the WooKey project [7], completes this analysis.

4.3.3 Physical faults and side channels

Fault attacks (voltage or clock variation, light perturbation) and side-channel attacks (power consumption, electromagnetic emission) aim to falsify a verification or extract a critical secret. The response consists of hardening the verification code, keeping secrets out of untrusted memory, limiting sensitive operations and placing critical computations in a protected trust domain. These attack families are classically described in the academic literature on cryptographic faults and side channels [12, 13, 34, 14]. They also intersect with more recent work on micro-architectural vulnerabilities and speculative execution, which shows that attacks are not limited to the logical or hardware level alone [33, 36, 15, 69].

4.3.4 Replay, maintenance abuse and chain-of-trust bypass

An attacker can re-inject an old proof, hijack a maintenance session or use an unprotected service interface to obtain an unintended trust state. The security mechanism must therefore integrate nonces, monotonic counters, explicit access control and logging of maintenance and attestation operations. The protocols and security architectures associated with attestation, secure sessions and mutual authentication are detailed in the reference specifications and recommendations [61, 63, 30, 25].

4.4 Trust bootstrapping principle

Trust bootstrapping is established when the initial decision is made in a hardware root of trust (SE, HSE, immutable ROM or a combination) before any mutable code. This principle limits the early bypasses typical of bootkit campaigns and reduces the pre-authentication attack window [47, 42].

The recommended sequence is as follows:

- cryptographic verification of the first executable code;
- measurement of the critical stages of the boot chain;
- progressive activation of resources according to a valid state;
- authentication declared only if integrity and provenance remain compliant.

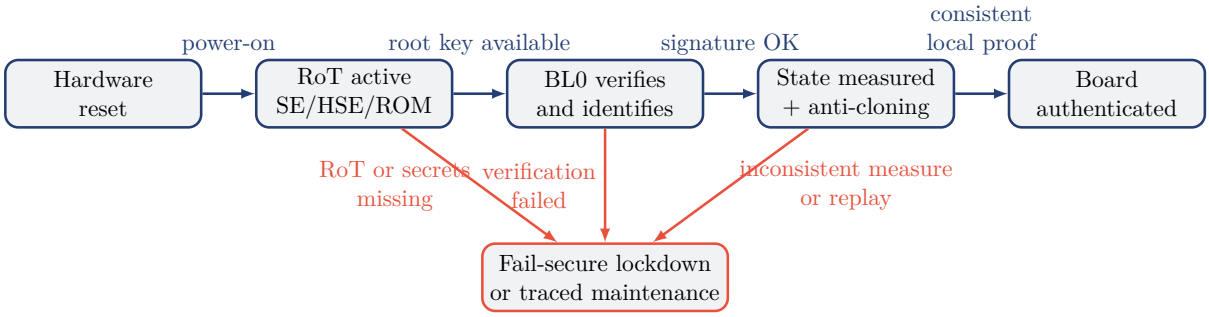


Figure 4.1: State machine of the boot chain for the use case "Authenticating the electronic board"

Summary

The state machine formalizes a boot chain centered on hardware identity: the root of trust first activates the secrets, the first executable code is verified, then the platform measurements are consolidated before the board is declared authenticated. Any loss of the root, verification failure, measurement inconsistency or session anomaly immediately switches to a *fail-secure* lockdown state or to explicitly traced maintenance.

4.5 Security specifications for the implementation

Start the security component first. This constraint reduces the pre-authentication attack window and prevents untrusted code from taking control before trust is established [47].

Mutual authentication of active components. Using a per-device unique key derivation, a PUF or a persistent identity backed by a manufacturing secret reduces cloning, substitution and cross-device replay scenarios [38, 51].

Protection of inter-component exchanges. The channels between MCU, SE, TPM and network elements must ensure confidentiality, integrity, anti-replay and origin authentication (SCP, TPM sessions, mutual TLS) [25, 30, 61].

Secret isolation and role separation. Root secrets must remain outside the untrusted memory domain of the SoC, and provisioning, test and maintenance operations must be separated to limit mass compromises [49, 52, 4].

4.6 Configurable components (FPGA) in the chain of trust

An FPGA is not a security component: by itself it offers neither certified physical resistance, nor key-usage policies, nor standardized attestation, and therefore has no place in the SE/TPM/TEE comparison of the selection chapters. It is however an **active component of the chain of trust** as soon as it is present on the board: its configuration (the bitstream) determines its behavior just like a firmware does, and a substituted bitstream is equivalent to a firmware injection that bypasses all the controls imposed on the GPP.

4.6.1 The bitstream is a firmware

The bitstream must be treated with exactly the same grid as the product's software:

- **Authenticity and integrity:** bitstream signed and verified before loading, using the native mechanisms of the component when they exist (bitstream authentication in recent families) or a verification carried by the board's root of trust;
- **Confidentiality:** bitstream encryption when the design contains intellectual property or elements exploitable by an attacker, with keys managed as root keys (controlled provisioning, non-exportability, compromise plan);
- **Anti-rollback:** bitstream version control when the component supports it, otherwise compensation at the level of the boot chain that loads and verifies the bitstream;
- **Measurement and traceability:** the version and digest of the bitstream take part in the platform measurements and attestation, and the bitstream is included in the SBOM and in the signed build register [47, 52].

Bitstream updates follow the hardened DFU flow of chapter "Updating firmware in the field (OTA/DFU)": signed manifest, reversible installation, re-validation at load time.

4.6.2 Known attacks and limits of native protections

FPGA bitstream protections have a documented history that forbids blind trust: extraction of bitstream encryption keys through power analysis across several component generations [39], and a complete break of the bitstream encryption of Xilinx 7-series devices by the Starbleed attack, not fixable on deployed silicon [20]. The configuration and test interfaces (JTAG, SelectMAP, readback, partial reconfiguration) furthermore constitute bypass paths analogous to MCU debug interfaces and must be disabled or locked in production [47].

The architectural consequence is the same as for the MCU flash protections discussed by the WooKey project [7]: high-value secrets must not reside in the FPGA domain if the threat model includes a laboratory attacker (AM-03); they remain in the SE or HSE, with the FPGA consuming derived, revocable session keys.

4.6.3 The FPGA as a root-of-trust substrate

Conversely, some security-oriented families (native secure boot, side-channel countermeasures, integrated PUF) can serve as the board's trust anchor, and an FPGA makes it possible to

implement auditable security functions: root-of-trust soft cores, cryptographic accelerators, prototyping of open designs such as OpenTitan [37, 38]. This approach is particularly interesting for open architectures, where design auditability compensates for the absence of third-party certification; in return it requires the same build-chain and verification rigor as any trusted component [52], and an explicit evaluation of the physical resistance actually obtained, which generally remains below that of a certified SE [17].

In summary: the FPGA is treated as a **full-fledged firmware carrier** in the threat analysis, checklists and tests of this chapter (component substitution, firmware injection, interface locking), and only substitutes for a certified security component upon a decision argued by the risk analysis.

4.7 Industrial checklist and summary test suite

4.7.1 Checklist of industrial-side actions

ID	Action	Owner	Expected evidence
CARTE-CL-01	Freeze the board's bill of materials and the list of trusted components (SE/HSE/TPM, MCU, boot memory)	Product architecture + industrialization	Signed, versioned BOM
CARTE-CL-02	Define and validate the personalization chain (sites, roles, secrets, separation of duties)	CISO + production	Approved provisioning procedure
CARTE-CL-03	Lock the debug/service interfaces in production mode	Hardware + firmware	Configuration report and lock verification
CARTE-CL-04	Set up the hardware identity and inter-component mutual authentication policy	Security firmware	Design file + authentication tests
CARTE-CL-05	Define the incident management plan (board substitution, secret compromise, suspect batch)	Quality + SOC/CSIRT	Incident playbook and escalation matrix

Table 4.2: Industrial checklist for "Authenticating the electronic board"

4.7.2 Summary test suite

ID	Test	Method	Criticality	Pass criterion
CARTE-T-01	Boot attempt with a substituted component	Integration test + BOM variation	High	Boot refused, audit event emitted
CARTE-T-02	Unauthorized firmware injection at provisioning	Negative bench test	High	Image rejected, no operational activation
CARTE-T-03	Replay of an inter-component authentication proof	Protocol test	Medium	Session refused (invalid nonce/counter)
CARTE-T-04	Fail-secure mode verification on measurement error	Controlled fault injection	High	Switch to traced lockdown/maintenance
CARTE-T-05	Audit of the traceability of sensitive operations	Log review	Medium	100% of refusals with exploitable `reason_code`

Table 4.3: Summary test suite for "Authenticating the electronic board"

5

Authenticating the general-purpose processor (GPP) software at boot

5.1 Threat and risk analysis

Typical attacks target the update chain, the signature mechanisms, the anti-rollback protection and the secure boot configuration. Real-world campaigns show that a CI/CD compromise can distribute firmware that looks legitimate but is malicious [52, 4].

Criticality correlates with remote exposure and with the difficulty of patching field equipment [24, 53].

5.1.1 Mapping of attacks, attackers and controls

Attack scenario	**Reference threat**	**AM profiles**	**OS objectives**	**Associated controls**
Downgrade to an old, vulnerable signed image	Boot and firmware compromise	AM-01, AM-04	OS-02, OS-03	Anti-rollback, minimum version, verification policies
Injection of a modified image into the CI/CD chain	Boot and firmware compromise	AM-04	OS-02, OS-05	Separation of signing roles, state attestation, build traceability
Signature verification bypass through a synchronized fault	Tampering with critical operations	AM-03	OS-02, OS-04	Fault-injection countermeasures, multi-stage verification, fail-secure mode
Chain-of-trust bypass through an active debug interface	Maintenance and debug abuse	AM-02, AM-03	OS-03, OS-05	Debug hardening, logging, maintenance access controls

Table 5.1: Mapping of the attacks of chapter “Authenticating the general-purpose processor (GPP) software at boot” to threats, AM profiles and OS objectives

5.2 Software chain of trust

The chain of trust must implement:

- signature verification at every stage
- cumulative integrity measurement
- anti-rollback version control
- an explicit degraded-mode and recovery policy

This model is consistent with the NIST firmware resiliency recommendations [47, 42].

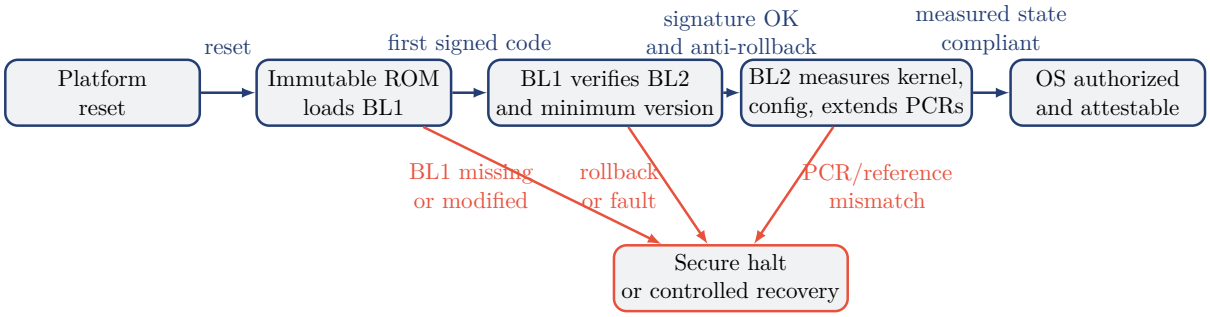


Figure 5.1: State machine of the boot chain for the use case "Authenticating the general-purpose processor (GPP) software at boot"

Summary

This state machine describes the minimal progression of a measured GPP boot: immutable ROM, successive verification of the bootloaders, anti-rollback check, then extension of the measurements before releasing the kernel. The transition to the final state is only allowed if integrity and version remain compliant; otherwise the platform falls into secure halt or controlled recovery.

5.3 SE/HSE versus TPM

With **SE/HSE**, verification is generally local, with limited energy/latency cost. With **TPM**, the state proof is standardized and remotely exploitable [66, 43]. A combined architecture (SE/HSE for local decision, TPM for remote evidence) provides a compromise between these constraints.

5.4 Security specifications for the implementation

5.4.1 Software image verification policy

Define the authorized algorithms, minimum sizes, signing authorities and revocation rules. Avoid obsolete primitives and align the policy with the NIST/ANSSI transitions [50, 1, 51].

5.4.2 Measurement and attestation of the software state

List the measured components (ext. ROM, BL1, BL2, kernel, config) and the attestation scheme. The reference values (golden measurements) must be governed and versioned [43, 30].

5.4.3 Signing key life cycle

Document generation, possible escrow, rotation, revocation and destruction. The compromise of a build key requires a re-issuance and fleet-wide trust recovery plan [52, 51].

5.5 Industrial checklist and summary test suite

5.5.1 Checklist of industrial-side actions

ID	Action	Owner	Expected evidence
GPP-CL-01	Formalize the secure boot policy (algorithms, authorities, revocation, anti-rollback)	Security architecture	Signed, versioned policy
GPP-CL-02	Secure the CI/CD chain of the boot images (role separation, signing, traceability)	DevSecOps	Signed build register + SBOM
GPP-CL-03	Build the reference (golden) measurements and the controlled update procedure	Firmware + operations	Validated, historized golden base
GPP-CL-04	Disable/protect boot bypass paths (debug, unauthorized recovery)	Platform integration	Hardened configuration report
GPP-CL-05	Define the fleet procedure in case of signing key compromise	CISO + operations	Tested rotation/re-issuance plan

Table 5.2: Industrial checklist for “Authenticating the general-purpose processor (GPP) software at boot”

5.5.2 Summary test suite

ID	Test	Method	Criticality	Pass criterion
GPP-T-01	Boot attempt on a signed but rolled-back image	Boot integration test	High	Boot refused (minimum version violated)
GPP-T-02	Injection of an unsigned image into the release pipeline	CI/CD test + signature check	High	Publication blocked and alert raised
GPP-T-03	Verification bypass through a simple fault	Robustness test (fault campaign)	High	No transition to operational state
GPP-T-04	Consistency check between attestation and golden measurements	Attestation test	Medium	Non-compliant device refused
GPP-T-05	Validation of boot decision traceability	Log review	Medium	Complete, exportable `ALLOW/DENY` decisions

Table 5.3: Summary test suite for "Authenticating the general-purpose processor (GPP) software at boot"

6
Authenticating application transactions after boot

6.1 Threat and risk analysis

The dominant threats are transaction replay, application identity spoofing, payload manipulation and local privilege escalation. Side-channel attacks remain applicable if the application keys are heavily used [34, 14, 61].

6.1.1 Mapping of attacks, attackers and controls

Attack scenario	Reference threat	AM profiles	OS objectives	Associated controls
Replay of legitimate commands on the application channel	Tampering with critical operations	AM-01, AM-02	OS-02, OS-03	Nonces/counters, time windows, context verification
Application identity spoofing after local compromise	Platform identity theft	AM-02, AM-04	OS-03, OS-05	Strong component-to-service authentication, periodic attestations
Payload manipulation in transit (local MITM)	Tampering with critical operations	AM-01, AM-02	OS-02, OS-03	End-to-end integrity, MAC/signature, authenticated sessions
Exfiltration or extraction of an application key through a side channel	Extraction of secrets and sensitive assets	AM-03	OS-01, OS-04	Non-exportable key usage, SE/TEE compartmentalization, usage control

Table 6.1: Mapping of the attacks of chapter “Authenticating application transactions after boot” to threats, AM profiles and OS objectives

6.2 Post-boot objective

The objective is to ensure that a transaction is issued by an authorized component, in an authorized state, and that it remains unaltered up to the verifier. This relies on short, contextualized and

auditable cryptographic proofs [63, 53].

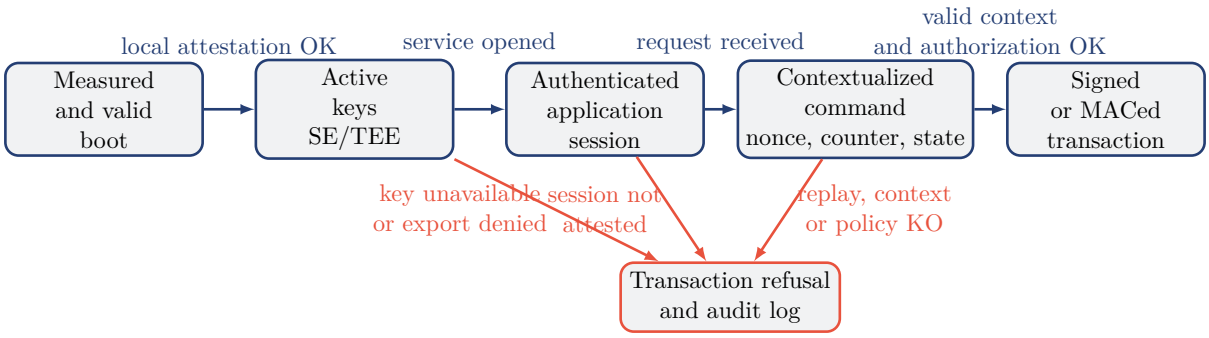


Figure 6.1: State machine linking the trusted boot to the use case “Authenticating application transactions after boot”

Summary

For this use case, the state machine explicitly links the boot chain to transactional authorization: keys are only activated after a valid boot, the application session must remain authenticated, then every command is checked with context, anti-replay and usage policy. Any state divergence, key unavailability or policy violation triggers an immediate refusal together with an audit trail.

6.3 Role split between SE and TEE

SE: storage of root keys, non-exportable signatures, strict usage policies [25, 17].

TEE: execution context verification, dynamic decisions, local acceleration [8, 35].

Recommended pattern: root key in the SE, derived short-lived session key in the TEE, with periodic re-attestation.

6.4 Security specifications for the implementation

6.4.1 Services to authenticate after boot

The implementation must start with an inventory of the services exposed after boot, then a classification by business impact and security impact. The minimal families to cover are:

- actuator commands (open/close, start/stop)
- sensitive telemetry (critical measurements, equipment state, alarms)
- configuration management (network parameters, security thresholds, profiles)
- administration operations (provisioning, key rotation, update policy)

To make the requirements auditable, four levels of application proof are defined:

- **N0 – Low criticality:** channel authentication only (e.g. mutual TLS), without per-message business signature
- **N1 – Standard:** channel authentication + message integrity (MAC) + basic anti-replay (nonce or counter)
- **N2 – Reinforced:** N1 + proof bound to the equipment state (TEE/SE context) + fine-grained authorization per command type
- **N3 – Critical/safety:** N2 + non-exportable signature in SE/TPM + counter-signature or strong server-side validation + immediate evidential log

Example of a recommended initial mapping:

Service	Level	Minimal mechanisms	Example
Critical actuator command	N3	SE signature, monotonic counter, state context verification	Opening command for an industrial valve
Security configuration	N2	MAC + periodic attestation + strict RBAC	Network allowlist change
Sensitive telemetry	N1/N2	MAC, timestamp, anti-replay window	Process measurement upload
Non-sensitive telemetry	N0/N1	Channel authentication + transport integrity	Non-critical battery status

Table 6.2: Example classification of post-boot services and associated proof level

The chosen level must be justified by the risk analysis and linked to the OS-01..OS-05 objectives of the chapter [27, 3, 53].

6.4.2 Application key usage policy

The key policy must be explicit, versioned and enforced by the trusted component (SE/HSE/TPM/TEE depending on the architecture). It covers at least:

- **Ownership and life cycle:** creation, activation, suspension, rotation, revocation, destruction
- **Non-exportability:** root keys and N2/N3 signing keys non-exportable
- **Usage context:** authorized uses (sign, verify, encrypt), service scope, time range, machine state
- **Usage limits:** quotas per period, consecutive-error thresholds, temporary then definitive lockout
- **Key separation:** one key per purpose (authentication, encryption, update, administration) to avoid cross-usage

Example of a concrete rule (N3 service):

- the key `K_cmd_critique` only signs the `ACTUATOR_CTRL` family
- signing allowed only if the local counter is synchronized and the attestation is less than 10 minutes old

- lockout after 5 consecutive verification failures
- forced rotation every 90 days, or immediate in case of incident

Example of a payload structure to sign:

```
{
 "device_id": "A1F4-22",
 "service": "ACTUATOR_CTRL",
 "cmd": "OPEN_VALVE",
 "nonce": "7f9c...",
 "ctr": 184233,
 "ts": "2026-08-09T10:45:12Z",
 "state_ref": "TEE_OK:4d2a"
}
```

Verification must fail if a critical contextual field is missing, if the counter goes backwards, or if the time window is exceeded [51, 53, 61].

6.4.3 Traceability and audit of cryptographic operations

Traceability must make it possible to answer three questions during an audit: *who* requested the operation, *what* was executed, and *in which security context*. The logs must therefore be:

- timestamped (reliable time source)
- bound to the device identity and to the service identity
- linked to the key identifier
- protected against tampering (hash chain or periodic signature)
- exportable to a central collector with a defined retention

Minimal fields of a cryptographic audit event:

- `event_id`, `ts`, `device_id`, `service`, `op_type`
- `key_id`, `policy_version`, `result`, `reason_code`
- `nonce/ctr`, `attestation_ref`, `prev_hash`

Example of a compact trace:

```
2026-08-09T10:45:12Z evt=9c2e dev=A1F4-22 svc=ACTUATOR_CTRL
op=SIGN key=K_cmd_critique policy=v12 res=DENY reason=CTR_ROLLBACK
ctr=184200 att=TEE_OK:4d2a prev=1e88...
```

6.4.4 Acceptance criteria and minimal tests

To avoid declarative requirements, each level N0..N3 must have automatable acceptance tests:

- **Anti-replay:** replay of an already accepted signed message ⇒ systematic refusal
- **Invalid context:** expired attestation or non-compliant state ⇒ refusal for N2/N3

- **Policy robustness:** attempt to use a key outside its scope ⇒ refusal + alert log
- **Traceability:** every `DENY` decision must produce a complete, exportable event
- **Degraded mode:** on connectivity loss, only explicitly locally authorized operations remain active

Example of a simple acceptance rule:

- critical service classified N3
- maximum application false-rejection rate $< 0.1\%$ over a 10 000-request campaign
- 100% of rejections linked to a normalized `reason_code`

This formalization eases alignment with the governance, audit and risk-management practices required by the industrial frameworks [31, 53, 27].

6.5 Industrial checklist and summary test suite

6.5.1 Checklist of industrial-side actions

ID	Action	Owner	Expected evidence
TRX-CL-01	Inventory the post-boot services and classify every flow into a level N0..N3	Product owner + security	Validated service/level matrix
TRX-CL-02	Deploy the application key usage policy (non-export, quotas, context)	PKI/cryptography owner	Versioned policy + enforced rules
TRX-CL-03	Implement unified anti-replay (nonce, counter, time window) on all critical flows	Application team	Architecture file + integration tests
TRX-CL-04	Enable evidential logging of `ALLOW/DENY` operations	Operations + SOC	Audit schema and proof of central export
TRX-CL-05	Define the authorized degraded mode on attestation/connectivity loss	Architecture + business	Operations procedure and continuity rules

Table 6.3: Industrial checklist for “Authenticating application transactions after boot”

6.5.2 Summary test suite

ID	Test	Method	Criticality	Pass criterion
TRX-T-01	Replay of an already accepted signed message	Protocol test	High	Systematic rejection + anti-replay alert
TRX-T-02	Critical command sent with an expired attestation	Negative application test	High	N2/N3 refusal with explicit error code
TRX-T-03	Use of a key outside its service scope	Cryptographic policy test	High	Signature refused + complete audit trail
TRX-T-04	Completeness check of refusal logs	Log review over a campaign	Medium	100% of `DENY` correlatable to a `reason_code`
TRX-T-05	Connectivity loss simulation and degraded mode	Operational resilience test	Medium	Only explicitly authorized services remain active

Table 6.4: Summary test suite for “Authenticating application transactions after boot”

7

Provisioning secrets and identities in production

7.1 Threat and risk analysis

The industrialization phase is the moment of the life cycle where the most critical assets of the product (root keys, identities, security parameters) are handled in the least controlled environment: remote production sites, subcontracting, high throughput, numerous personnel. A compromise at this stage has a mass effect: unlike a field attack targeting a single device, an attack on the personalization chain can affect an entire batch, or even the whole fleet if a shared secret is exposed [4, 52].

The dominant threats are the following:

- exfiltration of secrets during personalization (master keys, diversification data, certificates);
- over-production and cloning: manufacturing of out-of-quota units carrying valid identities;
- substitution of components or firmware in the supply chain;
- persistence of a test mode, debug mode or pre-production firmware after the factory exit;
- compromise of the personalization workstation or server, allowing the injection of attacker-known keys;
- corruption of the identity databases: falsified association between serial number, keys and certificates.

The dominant attacker profile here is the insider or compromised subcontractor (AM-02), possibly combined with a later field intervention (AM-04). The risk analysis must cover every site and every transfer of responsibility, including the logistics between component manufacturing, board assembly and final personalization [5, 4].

7.1.1 Mapping of attacks, attackers and controls

Attack scenario	Reference threat	AM profiles	OS objectives	Associated controls
Master key exfiltration from the personalization site	Extraction of secrets and sensitive assets	AM-02, AM-04	OS-01, OS-05	HSM boundary, encrypted export only, key ceremonies
Over-production of units carrying valid identities	Platform identity theft	AM-02	OS-03, OS-05	Signed quotas, production counters, accounting reconciliation
Injection of firmware or substituted components at assembly	Boot and firmware compromise	AM-02, AM-04	OS-02, OS-03	End-of-line cryptographic verification, provenance control
Compromised personalization workstation injecting known keys	Extraction of secrets and sensitive assets	AM-02	OS-01, OS-02	Secret generation inside the HSM/SE, end-to-end secure channel
Factory exit with active test or debug interfaces	Maintenance and debug abuse	AM-01, AM-02	OS-03, OS-04	End-of-line locking, exit self-test, configuration audit

Table 7.1: Mapping of the attacks of chapter “Provisioning secrets and identities in production” to threats, AM profiles and OS objectives

7.2 Expected security functions

- **Separation of roles and sites:** key generation, personalization, testing and shipping fall under distinct responsibilities; no single actor must be able to produce a complete authentic unit [52, 4].
- **Explicit HSM boundary:** secrets are generated and kept in a factory HSM (or directly in the target SE); any export is encrypted (*key wrapping*), never in clear text, even transiently [49, 51].
- **Per-device diversification:** each unit receives keys derived from a unique identifier (KDF from a master key, or a PUF-derived secret), so that the compromise of one unit does not compromise the batch [51, 38].
- **Production counting and quotas:** the number of issued identities is cryptographically bounded (signed quotas, monotonic counters on the HSM side) and reconciled with the declared physical production.
- **Per-unit traceability:** each personalization produces a signed record (serial number, batch, references of installed keys, test results) exported to a central repository.
- **End-of-line locking:** before shipping, the test and debug interfaces are disabled (fuses, OTP), the production firmware is verified and the device enters a controlled transport state [47, 3].

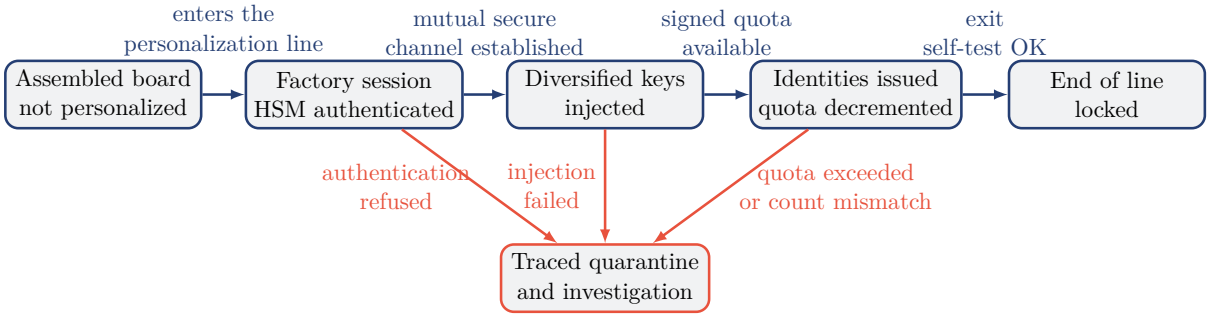


Figure 7.1: State machine of personalization for the use case “Provisioning secrets and identities in production”

Summary

The state machine formalizes factory personalization: the session with the factory HSM must be mutually authenticated before any injection, keys are diversified per device, each issued identity decrements a signed quota, and the line ends with locking and the exit self-test. Any authentication, injection, counting or locking failure sends the unit to traced quarantine, never to shipping.

7.3 Manufacturing identities and operational identities

The recommended model distinguishes two families of identities, along the lines of the IDevID and LDevID identities standardized by IEEE 802.1AR [28]:

- **Manufacturing identity (IDevID-like):** installed in the factory, carried by a certificate issued by a certification authority dedicated to production, it proves the origin and authenticity of the hardware throughout its life. It is immutable and cannot be revoked per unit without a return to the workshop.
- **Operational identity (LDevID-like):** enrolled at deployment from the manufacturing identity, it carries the application rights of the device at its operator. It is renewable and revocable without touching the manufacturing identity.

This separation enables ownership transfer (the operator re-enrolls an operational identity without depending on factory secrets) and targeted revocation: a suspect batch is revoked at the production authority level, without invalidating the rest of the fleet. The factory certification authority must be kept offline, its ceremonies documented, and its compromise treated as a crisis scenario with a re-issuance plan [51, 4].

When the component allows it, a PUF-derived secret can anchor the manufacturing identity without storing a key in non-volatile memory, which reduces the physical extraction surface [38].

7.4 Security specifications for the implementation

Generate secrets as close as possible to where they are used. Generation in the target SE (*on-die*) eliminates secret transport; failing that, generation in the factory HSM with injection through a secure channel (SCP03 or equivalent) is acceptable, clear-text injection never is [25, 49].

Establish an end-to-end secure channel. The channel between the factory HSM and the target component must ensure confidentiality, integrity, anti-replay and mutual authentication, including through the intermediate test equipment, which must be treated as untrusted [25, 61].

Make production accountable. Each identity issuance decrements a signed quota; discrepancies between consumed quotas, declared units and tested units trigger an investigation. This reconciliation is the main defense against over-production [4].

Verify the factory exit state. A final self-test must prove, with a signed record: debug locked, authentic production firmware, identities correctly installed, transport mode active. Any failing unit goes to traced quarantine [47, 3].

7.5 Industrial checklist and summary test suite

7.5.1 Checklist of industrial-side actions

ID	Action	Owner	Expected evidence
PROV-CL-01	Map sites, roles and responsibility transfers of the personalization chain	CISO + industrialization	Validated map + subcontractor contracts
PROV-CL-02	Set up the factory HSM, the key ceremonies and the encrypted-export policy	PKI/cryptography owner	Ceremony minutes, signed policy
PROV-CL-03	Deploy per-device diversification and the dual manufacturing/operational identity	Security firmware + PKI	Design file + test certificates
PROV-CL-04	Activate signed quotas, production counting and periodic reconciliation	Production + quality	Archived reconciliation reports
PROV-CL-05	Formalize end-of-line locking and the factory exit self-test	Hardware + firmware	End-of-line procedure + signed logs

Table 7.2: Industrial checklist for “Provisioning secrets and identities in production”

7.5.2 Summary test suite

ID	Test	Method	Criticality	Pass criterion
PROV-T-01	Clear-text key injection attempt on the line	Negative bench test	High	Injection refused, secret never observable in clear
PROV-T-02	Identity issuance beyond the authorized quota	HSM policy test	High	Issuance blocked, overrun alert raised
PROV-T-03	Replay of a captured personalization session	Protocol test	High	Session refused (factory channel anti-replay)
PROV-T-04	Factory exit state check on a sample	Bench audit + configuration readout	High	Debug locked, transport mode active on 100% of the sample
PROV-T-05	Quota/production/shipping reconciliation on a batch	Security accounting review	Medium	Zero discrepancy, or justified by traced quarantine

Table 7.3: Summary test suite for “Provisioning secrets and identities in production”

8

Updating firmware in the field (OTA/DFU)

8.1 Threat and risk analysis

The update capability is both a regulatory requirement (vulnerability fixing throughout the support period [24]) and one of the most profitable attack surfaces for an attacker: a compromised update mechanism provides persistent code execution, legitimate in appearance and deployable at fleet scale [47, 4].

The dominant threats are the following:

- distribution of a malicious image through a compromised CI/CD chain or update server;
- forced rollback to a signed but vulnerable version;
- desynchronization between verification and installation (TOCTOU): the checked image is not the one actually written or executed [7];
- deliberate or accidental interruption (power cut) leaving the device in an inconsistent or unbootable, exploitable state;
- interception or alteration of the update flow (MITM, truncated image, replayed manifest);
- abuse of the recovery mode or local DFU to load arbitrary code.

8.1.1 Mapping of attacks, attackers and controls

Attack scenario	Reference threat	AM profiles	OS objectives	Associated controls
Malicious image distributed through the update infrastructure	Boot and firmware compromise	AM-04	OS-02, OS-05	Signed manifest, separation of signing roles, build traceability
Rollback to a signed, vulnerable version	Boot and firmware compromise	AM-01, AM-04	OS-02, OS-03	Monotonic anti-rollback counter, minimum version
Alteration between verification and execution (TOCTOU)	Tampering with critical operations	AM-03	OS-02, OS-04	Verification at install and at every boot, atomic write
Replay of an old manifest (freeze attack)	Tampering with critical operations	AM-01, AM-04	OS-02, OS-05	Manifest expiry, verified timestamping
Abuse of recovery mode or local DFU	Maintenance and debug abuse	AM-01, AM-02	OS-03, OS-05	Minimal signed recovery, DFU session authentication, logging

Table 8.1: Mapping of the attacks of chapter “Updating firmware in the field (OTA/DFU)” to threats, AM profiles and OS objectives

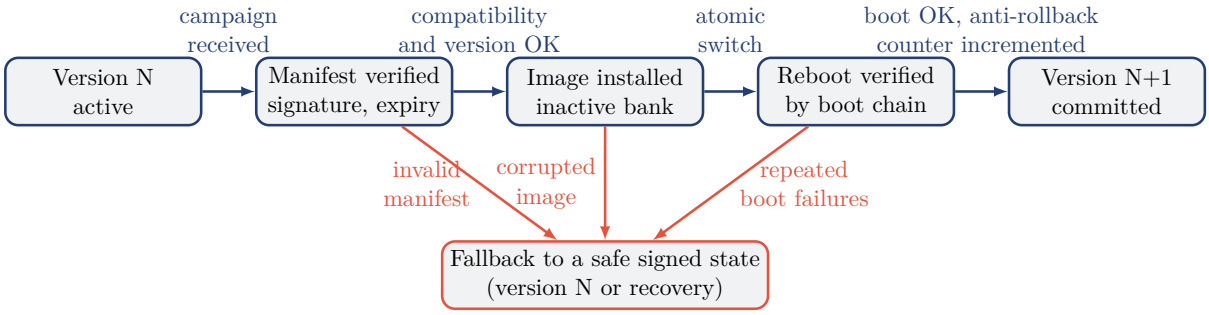


Figure 8.1: State machine of the update for the use case “Updating firmware in the field (OTA/DFU)”

8.2 Recommended update architecture

Summary

The state machine links the OTA campaign to the boot chain: the manifest is verified (signature, compatibility, expiry) before installation into the inactive bank, the switch is atomic, then the new image must be re-validated at startup before the version is committed and the anti-rollback counter incremented. Any anomaly brings the device back to a safe signed state (previous bank or recovery), with no unbootable intermediate state.

8.2.1 Signed manifest and metadata

Every update is described by a signed manifest containing at least: the version, the digests of the images, the hardware target and its compatibility constraints, and an expiry date. Expiry protects against freeze attacks, where an attacker indefinitely replays an old manifest to prevent the fleet from correcting itself. The separation of signing roles (release keys distinct from development

keys, multi-signature thresholds for sensitive operations) follows the model established by TUF and its embedded variant Uptane [64, 68].

8.2.2 A/B partitioning and recovery

Installation is performed in an inactive bank while the active bank remains intact; the switch is atomic and reversible. A boot attempt counter bounds the trials on the new image: on repeated failures, the device automatically returns to the previous bank, then, as a last resort, to a minimal signed recovery. No intermediate state may be unbootable, and no recovery path may accept unsigned code [47].

8.2.3 Anti-rollback and consistency with secure boot

The update mechanism installs, but does not decide: the execution authority remains the boot chain described in chapter "Authenticating the general-purpose processor (GPP) software at boot". The image is therefore verified twice, at installation and at every startup, which closes the TOCTOU scenarios between the two steps [7, 47]. The minimum acceptable version is carried by a monotonic hardware counter, irreversibly incremented when a security update is committed.

8.2.4 Protection of the channel and of the local DFU flow

The OTA channel relies on a mutually authenticated session (mutual TLS or equivalent) [61]; the authenticity of the image however remains carried by the manifest, never by the channel alone, so that security does not depend on intermediate infrastructures (CDN, relays). Local DFU (UART, USB, maintenance JTAG) must require operator authentication and an anti-replay session; the WooKey approach, where an active token constrains the DFU flow, illustrates the hardening achievable when the local threat is high [7, 19].

8.3 Fleet rollout and governance

Rollout must be progressive: a canary population, then growing waves, with success-rate telemetry and automatic campaign stop criteria. For industrial environments, maintenance windows and availability constraints require planning agreed with the operator [27]. The CRA furthermore requires distinguishing security updates from functional evolutions and informing the user [24].

Two crisis scenarios must be prepared in advance: the mass failure of a campaign (fallback and recovery plan) and the compromise of a signing key, which points back to the fleet-scale re-issuance plan defined for secure boot (action GPP-CL-05).

8.4 Security specifications for the implementation

Verify before installing and before executing. A single verification at reception is insufficient: signature and compatibility are checked at installation, then integrity is re-verified by the boot chain at every startup [47].

Never degrade the existing state. The active image remains intact and bootable until the new image is fully validated; writing is tolerant to power cuts (journaling, resume on interruption) and tested as such.

Encrypt the image when confidentiality requires it. Authenticity and integrity are mandatory in all cases; encryption is added when the image contains intellectual property or sensitive parameters, using an authenticated mode [41].

Protect the signing infrastructure. Release keys reside in an HSM, signatures are logged and tied to an identified reproducible build (SBOM, signed register), and key rotation is exercised regularly [52, 51].

8.5 Industrial checklist and summary test suite

8.5.1 Checklist of industrial-side actions

ID	Action	Owner	Expected evidence
OTA-CL-01	Specify the signed manifest (fields, expiry, signing roles and thresholds)	Security architecture	Versioned specification
OTA-CL-02	Implement A/B partitioning with atomic switch and signed recovery	Firmware	Design file + integration tests
OTA-CL-03	Deploy hardware anti-rollback consistent with the secure boot policy	Security firmware	Cross-test report with GPP-T
OTA-CL-04	Set up progressive rollout (canary, waves, automatic stop criteria)	DevSecOps + operations	Validated campaign procedure
OTA-CL-05	Prepare the crisis plans: mass failure and signing key compromise	CISO + operations	Plans tested in exercises

Table 8.2: Industrial checklist for “Updating firmware in the field (OTA/DFU)”

8.5.2 Summary test suite

ID	Test	Method	Criticality	Pass criterion
OTA-T-01	Installation of an image with an expired or replayed manifest	Protocol test	High	Systematic rejection + audit event
OTA-T-02	Power cut at every stage of the installation	Instrumented robustness test	High	Systematic restart on a valid image
OTA-T-03	Rollback attempt to a signed, vulnerable version	Boot integration test	High	Refusal (monotonic counter violated)
OTA-T-04	Image alteration between verification and write (TOCTOU)	Instrumented negative test	High	Detection at boot, no execution of the altered code
OTA-T-05	Canary campaign with a simulated failure rate above the threshold	Fleet governance test	Medium	Automatic stop of the wave + operations alert

Table 8.3: Summary test suite for “Updating firmware in the field (OTA/DFU)”

9

Attesting, monitoring and retiring devices in operation

9.1 Threat and risk analysis

Once deployed, the fleet lives for years in uncontrolled environments. The mechanisms of the previous chapters (measured boot, protected keys, signed updates) only produce value in operation if they are observed: a compromised device that is never queried stays silently compromised. Security maintenance in operational conditions (MCO) therefore covers periodic attestation, monitoring, incident response and the controlled retirement of devices [53, 31, 24].

The dominant threats are the following:

- silent persistence: a compromised device keeps operating without ever being detected, for lack of attestation or exploited telemetry;
- false attestation evidence: replay of an old proof, emulated device, or evidence produced by a compromised component;
- drift of the reference measurements: corrupted or ungoverned golden values cause a malicious state to be accepted, or flood the monitoring with false positives;
- compromise of the verification infrastructure (attestation server, log collector), which blinds the entire fleet;
- secret leakage at end of life: decommissioned, resold or RMA-returned devices with intact secrets and data;
- failing revocation: a retired or stolen device keeps valid identities and accesses.

9.1.1 Mapping of attacks, attackers and controls

9.2 Remote attestation architecture

Attack scenario	Reference threat	AM profiles	OS objectives	Associated controls
Replay or emulation of an attestation proof	Platform identity theft	AM-03, AM-04	OS-02, OS-05	Freshness nonce, attestation key bound to the hardware identity
Persistent, undetected compromise in the field	Boot and firmware compromise	AM-04	OS-02, OS-05	Periodic attestation, security telemetry, fleet correlation
Corruption of the golden values or of the verifier	Tampering with critical operations	AM-02, AM-04	OS-02, OS-03	Signed, versioned golden values, hardened and audited verifier
Secret extraction from a decommissioned or RMA device	Extraction of secrets and sensitive assets	AM-01, AM-03	OS-01, OS-03	Cryptographic erasure, workshop quarantine, revocation
Use of valid identities by a retired or stolen device	Platform identity theft	AM-01, AM-04	OS-03, OS-05	Revocation of operational identities, up-to-date fleet inventory

Table 9.1: Mapping of the attacks of chapter "Attesting, monitoring and retiring devices in operation" to threats, AM profiles and OS objectives

9.2.1 Model and evidence

The architecture follows the RATS model: the device (*attester*) produces signed evidence of its state, a verifier confronts it with the reference values, and the business services (*relying parties*) consume the verdict, never the raw evidence [63]. On a TPM platform, the evidence is a quote of the PCRs covering the measured boot chain of the GPP chapter [30, 43]; on an SE/TEE architecture, equivalent evidence can be built from the measurements consolidated by the root of trust.

Two properties are non-negotiable: **freshness** (verifier nonce included in the evidence, to prevent replay) and **identity binding** (the attestation key is cryptographically bound to the manufacturing identity of the provisioning chapter, to prevent one device from emulating another) [63, 28].

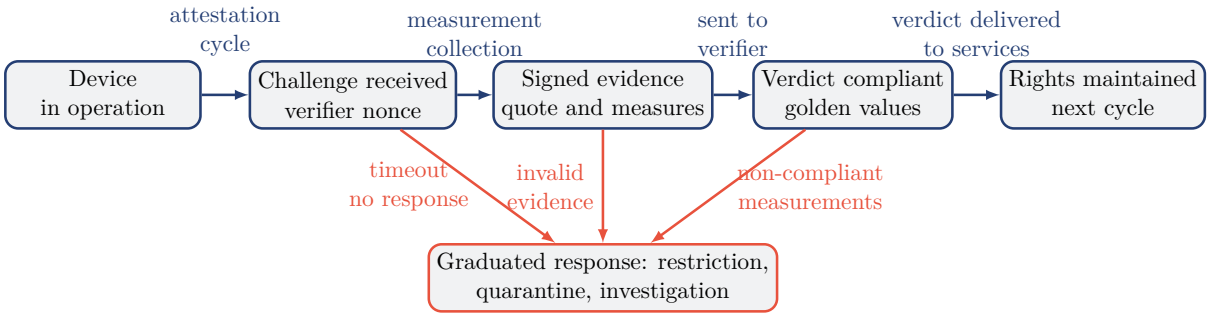


Figure 9.1: State machine of the attestation cycle for the use case "Attesting, monitoring and retiring devices in operation"

Summary

The state machine describes the periodic attestation cycle: the verifier issues a challenge with a nonce, the device produces signed evidence bound to its hardware identity, and the verdict is rendered by comparison with the versioned golden values. Missing, invalid or replayed evidence, like a non-compliant state, triggers the graduated response (rights restriction, quarantine, investigation) rather than a silent rejection.

9.2.2 Governance of the reference values

Golden values are a security asset in their own right: signed, versioned, produced by the CI/CD chain at build time (each release publishes its reference measurements in the signed register), and deployed to the verifier in sync with the OTA campaigns [43, 52]. A firmware update without a reference update produces massive false positives; the reverse, references updated without control, gives a build-chain attacker a way to have a malicious state accepted.

9.2.3 Reaction policy

A non-compliance verdict must trigger a graduated, predefined reaction: restriction of application rights (falling back to levels N0/N1 of the transactions chapter), network quarantine, then investigation. The policy must be written before the first incident, specifying who decides, on which criteria, and how a rehabilitated device rejoins the fleet (full re-attestation after controlled re-provisioning).

9.3 Monitoring and incident response

9.3.1 Minimal security telemetry

The telemetry reported per device must cover at least: boot results and failures, `ALLOW/DENY` decisions of cryptographic operations, attestation verdicts, anti-rollback counter values, and update events (start, success, fallback). These events reuse the audit format defined in the transactions chapter (reliable timestamping, device identity, `reason_code`, chaining) and are exported to the central collector [31, 27].

9.3.2 Fleet-scale correlation

Per-unit analysis is not enough: mass attacks are detected through correlation (anti-replay refusal spike on a batch, grouped attestation failures after an OTA campaign, devices from the same site that stop reporting). The SOC must have the cross-referenced batch/site/version repositories to run these correlations, which implies that the production registers of the provisioning chapter are readable by the monitoring [53, 4].

9.3.3 Incident response

Three playbooks must exist and be exercised: isolated compromised device (quarantine, forensic collection, re-provisioning or retirement), suspect batch (cross-check with the production registers, batch revocation), and compromise of an infrastructure key (switch to the fleet re-issuance plan defined in GPP-CL-05 and OTA-CL-05). The CRA also mandates the notification of actively

exploited vulnerabilities within short deadlines, which assumes an internal qualification chain already in place [24, 53].

9.4 End of life, RMA returns and ownership transfer

Retiring a device is a security operation, not a logistics operation:

- **Secret erasure:** cryptographic erasure (destruction of the encryption key in the SE/HSE) is the reference method for data encrypted at rest; residual secrets in unencrypted memory follow the media sanitization recommendations [44];
- **Revocation:** operational identities (LDevID) are revoked upon inventory exit; the manufacturing identity is kept for traceability (RMA, disputes, post-mortem analysis) [28];
- **RMA returns:** a device returning from the field is treated as untrusted: workshop quarantine, diagnosis on an isolated bench, and full re-personalization before any reinjection into the fleet [4];
- **Ownership transfer:** the new operator re-enrolls its own operational identity after purging the data and policies of the previous one; the process relies on the dual identity of the provisioning chapter, without exposing factory secrets.

These requirements must be defined from the architecture stage: cryptographic erasure is only possible if at-rest encryption was designed for it, and mass revocation is only realistic if the fleet inventory is reliable [24, 3].

9.5 Industrial checklist and summary test suite

9.5.1 Checklist of industrial-side actions

ID	Action	Owner	Expected evidence
MCO-CL-01	Deploy the attestation verifier and the graduated reaction policy	Security architecture + operations	Architecture file + validated policy
MCO-CL-02	Integrate the publication of signed golden values into the CI/CD chain, in sync with OTA	DevSecOps	Versioned reference register
MCO-CL-03	Define the minimal security telemetry and its export to the SOC	Operations + SOC	Telemetry schema + collection proof
MCO-CL-04	Write and exercise the three playbooks (device, batch, infrastructure key)	CISO + SOC/CSIRT	Dated exercise reports
MCO-CL-05	Formalize the end-of-life, RMA and ownership-transfer procedures	Quality + operations	Approved procedures + inventory traceability

Table 9.2: Industrial checklist for “Attesting, monitoring and retiring devices in operation”

9.5.2 Summary test suite

ID	Test	Method	Criticality	Pass criterion
MCO-T-01	Replay of an old attestation proof	Protocol test	High	Proof rejected (invalid nonce) + audit
MCO-T-02	Attestation of a device with modified firmware	End-to-end integration test	High	Non-compliant verdict, graduated reaction applied
MCO-T-03	OTA campaign without golden values update	Negative governance test	Medium	Divergence detected before rollout, campaign blocked
MCO-T-04	Data readout from a device after end-of-life erasure	Workshop media analysis	High	No exploitable data or secret recovered
MCO-T-05	Connection of a device whose identities are revoked	Infrastructure test	High	Access denied on all services + alert

Table 9.3: Summary test suite for "Attesting, monitoring and retiring devices in operation"

10

Executing cryptographic algorithms on SE, TPM and TEE

In an embedded system, executing a cryptographic primitive is not just a matter of choosing a library or a function. In practice it draws the boundary between application code exposed to software attacks and the secrets that must remain controlled, attested and, whenever possible, isolated in a distinct environment. The decision to execute an operation on a Secure Element (SE), a TPM or a TEE therefore depends on the threat model, the criticality of the keys, the attestation needs and the performance and cost constraints [11, 51, 10].

In practice, three motives often justify this offloading: protecting root keys or critical secrets, obtaining reliable attestation of the device state, and reducing the exposure of the main compute core to local attacks or memory leaks. This separation is particularly relevant for industrial products, healthcare equipment and long-lifetime connected devices, where the robustness of the chain of trust and the secure update capability matter as much as the mere execution of an algorithm [27, 31, 24].

10.1 Choosing between SE, TPM and TEE

The choice of a trust architecture must be guided by the required isolation level and by the nature of the operation to execute. An SE is often selected for the protection of root keys and high-value sensitive operations, a TPM for attestation, measurement storage and usage-policy governance, whereas a TEE is appropriate when a subset of secure code should run with a good trade-off between software isolation and performance [25, 30, 9].

In a realistic design, not all cryptographic operations are placed in the same component. Root keys and signature/attestation functions are often delegated to a distinct component, while the host microcontroller keeps session keys, handles or lighter policy elements. This split improves resilience against the compromise of the application core and simplifies updating the cryptographic stack without re-designing the whole product [10, 57].

10.2 Hardware vs software acceleration

Offloading to a secure component generally reduces key exposure and CPU load, but adds IPC/bus latency. The right choice depends on the profile (throughput, energy, secret criticality, BOM cost) [11, 51].

10.2.1 Performance/consumption benchmarks: integrated SE vs MCU library

Compare at least:

- average time and performance of signing, authentication and encryption
- energy per operation
- memory cost (RAM/Flash)
- impact on real-time tasks

These measurements must be interpreted in the light of the integration context: a secure component may cost more in latency and energy per transaction, while providing better secret protection and better attestation capability. For this reason, benchmark campaigns must be run in representative conditions, with realistic loads and recovery scenarios, so as not to wrongly favor a solution that looks too good on paper [52, 11].

10.2.2 Decision criteria for operation placement

A relevant choice often relies on four simple questions: must the key be protected against extraction from the application core? is attestation of the device state needed, or simply fast execution? is the operation security-critical, or can a local weakness be tolerated? finally, must the system allow firmware or cryptographic library updates without full re-design? These questions point to an SE when hardware protection is the priority, to a TPM when attestation and policy are central, and to a TEE when a more flexible trade-off is sought [25, 30, 9, 10].

10.2.3 Impact on battery autonomy

On constrained equipment, the energy/security ratio must be measured: a protected key that is unreachable in sleep mode can hurt QoS. The architecture must therefore separate frequent operations from root operations [3, 11].

10.2.4 Latency introduced by inter-chip exchanges

The impact of the buses (I2C/SPI) on transactional latency must be quantified, especially under load and in case of losses/retransmissions. Timeouts and retries must be bounded to avoid security deadlocks [25, 53].

10.3 Natively supported operations

10.3.1 Symmetric encryption

AES-GCM is to be preferred for combined confidentiality/integrity protection; non-authenticated modes should be restricted to legacy cases with adequate encapsulation. In practice, choosing an authenticated mode is not only a compliance matter: it avoids the classic design mistakes where missing integrity opens the door to stream or message manipulation attacks [41, 50]. This recommendation applies to industrial command exchanges as well as to the local protection of sensitive data at the edge of the device.

Using AES-GCM however comes with one absolute constraint: the uniqueness of the key/IV pair. Nonce reuse destroys both the confidentiality and the authenticity of the mode, and the risk is real in embedded systems, where an IV counter may reset after a power cut. In line with the ANSSI recommendations (RGS, annex B1), IV generation must be either deterministic and persistent (a saved monotonic counter) or random with a bounded encryption volume per key [1, 41].

10.3.2 Asymmetric encryption

ECC is commonly chosen in embedded systems to limit the computational load at an equivalent security level according to the recommended key sizes. Key sizes must follow the transition recommendations, since moving to higher security levels affects message size, consumption and the lifetime of signature or key-exchange mechanisms [51, 1]. In an embedded context, this choice must be aligned with the product's cryptoagility policy and with the firmware update constraints.

10.3.3 Hash functions

SHA-256/384 remain the practical references; SHA-1 is forbidden for security uses. The selection depends on performance and compliance constraints, but also on the expected robustness with regard to data integrity, message signing and firmware authentication [45, 50]. On a constrained microcontroller, it is common to favor a hash function that is lighter in micro-architecture, provided the security level remains consistent with the device's risk profile.

10.3.4 Random number generation

RNG quality conditions cryptographic security: a biased RNG can compromise all mechanisms. In practice this step is often underestimated even though it conditions the security of all subsequent operations, from key generation to the protection of nonces and authentication protocols. It is therefore necessary to require statistical validation and a documented design, typically based on a DRBG fed by a qualified entropy source [46, 48, 18].

10.4 Protecting data at rest

The protection of data at rest is a cross-cutting requirement: the checklist of chapter 3 refuses production release without an integrity mechanism, and the cryptographic erasure of chapter 9 is only possible if at-rest encryption was designed for it. The WooKey feedback recalls the classic trap: full-volume encryption protects confidentiality but not integrity, and an encrypted piece of data can be altered or replayed offline without detection [7].

10.4.1 Recommended encryption architecture

- **Authenticated encryption by default:** AEAD (AES-GCM or equivalent) for application data; if a non-authenticated mode is imposed (legacy FDE), integrity must be carried by a higher layer (application digests, signed journal) and the deviation recorded in the risk analysis [41, 7];
- **Key hierarchy:** non-exportable storage master key in the SE/HSE, volume or file keys derived through a KDF and protected by wrapping; rotation of the wrapping key is then performed by re-wrapping, without fully re-encrypting the media; the proven compromise of a data key however requires re-encrypting the affected data [51, 49];
- **Binding to the platform state:** unlocking the storage keys is only allowed after a valid boot (TPM sealing on PCRs, or SE usage policy), so that compromised firmware cannot mount the volumes [30, 43];
- **Data anti-replay:** a monotonic counter or version reference protects against the offline restoration of an old storage state (data rollback), symmetric to the firmware anti-rollback.

10.4.2 Erasure and end of life

If all sensitive storage is encrypted by design, destroying the master key in the SE/HSE amounts to erasing the data (*crypto-erase*), which makes fleet-scale retirement realistic; media containing unencrypted residues fall under classic sanitization [44]. The associated operational procedures are defined in chapter 9.

10.4.3 Acceptance rules

- direct offline readout of the media ⇒ no exploitable data;
- offline modification of an encrypted block ⇒ detection at mount or read time, traced refusal;
- restoration of an old storage image ⇒ refusal (version reference violated);
- non-compliant boot ⇒ storage keys not unlocked.

10.5 APIs and programming interfaces

10.5.1 GlobalPlatform: APDU commands for Java Card SE

APDUs and SCP03 structure a portable, auditable model for secure operations on an SE [25, 59].

10.5.2 TPM 2.0 Library

TPM stacks offer standard commands (quote, seal, unseal, policy) that ease interoperability with attestation infrastructures [30, 66].

10.5.3 PSA Crypto API (ARM)

PSA eases multi-vendor portability and algorithmic agility by separating the logical API from the hardware implementation [10, 11].

10.5.4 TrustZone-M: secure calls to a Cortex-M TEE

TrustZone-M allows a clean partition between sensitive and non-sensitive code, provided interrupts, memories and peripherals are strictly compartmentalized [9, 67].

10.6 Software integration patterns

10.6.1 Wrapper abstraction layer

Introducing a cryptographic abstraction layer simplifies SE/TPM/TEE substitution and reduces technical debt during post-quantum migration [57, 10].

10.6.2 Error and timeout handling

Secure component errors must never silently degrade to a non-secure mode. Define explicit fail-secure policies [53, 27].

10.6.3 Logging and audit of sensitive cryptographic operations

Log the key identity, the operation, the result, the context and the firmware reference. This traceability is essential for investigation and CRA compliance [24, 31].

10.7 Practical summary

Beyond lists of algorithms and APIs, a sound architecture makes a clear security model, standardized interfaces and a measurement capability coexist. A robust system does not merely pick AES-GCM or ECC: it defines where secrets are protected, how the device state is attested, how timeouts are handled and how operations are made auditable. It is this level of consistency that turns a cryptographic implementation into a durable trusted component, in the sense of the industrial frameworks and current regulatory expectations [11, 25, 30, 9].

10.8 Industrial checklist and summary test suite

10.8.1 Checklist of industrial-side actions

10.8.2 Summary test suite

ID	Action	Owner	Expected evidence
CRYPTO-CL-01	Inventory the primitives in use and ban obsolete algorithms (versioned cryptographic policy)	Security architecture	Signed cryptographic policy [50, 1]
CRYPTO-CL-02	Justify the placement of each operation (SE/TPM/TEE/host) through the threat analysis	Security architecture	Validated operation/component matrix
CRYPTO-CL-03	Implement the cryptographic abstraction layer (PSA or equivalent)	Firmware	Architecture file + code review
CRYPTO-CL-04	Measure performance and energy in representative conditions before freezing the architecture	Firmware + hardware	Archived benchmark report
CRYPTO-CL-05	Qualify the randomness chain (entropy source, DRBG, health tests)	Security firmware	RNG validation file

Table 10.1: Industrial checklist for "Executing cryptographic algorithms on SE, TPM and TEE"

ID	Test	Method	Criticality	Pass criterion
CRYPTO-T-01	Known-answer self-tests (KAT) on all primitives	Automated test at every build	High	100% of KATs compliant, blocking on failure
CRYPTO-T-02	Secure component error or timeout during operation	Controlled failure injection	High	Fail-secure, no silent fallback to an unprotected implementation
CRYPTO-T-03	Export attempt on a key marked non-exportable	Key policy test	High	Export refused + audit event emitted
CRYPTO-T-04	Latency and energy measurement per operation vs allocated budget	Measurement campaign under load	Medium	Budgets met, or documented waiver
CRYPTO-T-05	Statistical validation of the RNG output and of the health tests	Statistical suite (SP 800-90B)	High	No bias detected, health tests active in production [48]

Table 10.2: Summary test suite for "Executing cryptographic algorithms on SE, TPM and TEE"

A
Migration to post-quantum cryptography

A.1 Quantum threat and migration urgency

The security of currently deployed public-key cryptography (RSA, ECDSA, ECDH) relies on the hardness of factoring and of the discrete logarithm. Shor's algorithm, executable on a large-scale quantum computer (CRQC, *Cryptographically Relevant Quantum Computer*), solves both problems efficiently and would render this entire asymmetric cryptography obsolete [62, 6, 57]. The risk is already present through the *Harvest Now, Decrypt Later* (HNDL) attack: an adversary can capture encrypted flows today in order to decrypt them once a CRQC is available [40, 6]. For long-lifetime embedded systems (industrial, healthcare, defense), whose data confidentiality must be guaranteed beyond 5 to 10 years, this threat is immediate and must drive architectural choices from the design stage. Symmetric cryptography is less exposed: Grover's algorithm only provides a quadratic speedup of exhaustive search, and its effect is countered by doubling key sizes (AES-256 instead of AES-128, SHA2-384 or SHA3-384 for hashes) [6, 57].

A.2 Post-quantum standards

Post-quantum cryptography (PQC) refers to algorithms that run on classical computers and whose security is conjectured to hold against an adversary equipped with a quantum computer. NIST finalized in 2024 the first standards resulting from its standardization process [57]:

- **ML-KEM** (FIPS 203, formerly Kyber): key-encapsulation mechanism based on modular euclidean lattices, intended for establishing a confidential channel [54]
- **ML-DSA** (FIPS 204, formerly Dilithium): lattice-based digital signature, general profile [55]
- **SLH-DSA** (FIPS 205, formerly SPHINCS+): hash-tree based signature, mathematical assumptions distinct from lattices and usable without hybridation according to ANSSI

These three algorithms form the concrete migration basis identified by NIST in IR 8547 *Transition to Post-Quantum Cryptography Standards* [57], which also sets a deprecation timeline for classical

algorithms (RSA, ECDSA, DH) starting in 2030 and their prohibition by 2035 at the latest.

A.3 ANSSI roadmap

In its position paper [6], ANSSI defines three transition phases applicable to products certified in France, aligned with the NIST deliverables:

- **Phase 1** (current situation until the publication of the first NIST standards): optional hybridation, post-quantum resistance is an optional *defense in depth*. The hybrid mechanism must guarantee that no pre-quantum security regression is introduced. ANSSI recommends starting this phase now for products whose lifetime extends beyond 2030
- **Phase 2** (from 2026, with NIST standards published): hybridation mandatory for any product claiming quantum resistance. ANSSI will evaluate the selected PQC algorithms and may deliver security visas including a post-quantum security guarantee
- **Phase 3** (probably not before 2030): standalone PQC possible without hybridation, after sufficient cryptanalytic hindsight has accumulated

Hybridation consists of combining a proven classical algorithm and a PQC algorithm: for key establishment, the final secret is derived from both contributions through a KDF; for signatures, both signatures must be valid. The message-size overhead remains small relative to the cost of the PQC primitives themselves.

A.4 Cryptoagility and practical implications for embedded systems

ANSSI and NIST both insist on *cryptoagility*: the ability of a product to substitute its cryptographic primitives without hardware redesign or mass recall [6, 57]. In the context of the embedded systems covered by this guide, this translates into:

- **Primitive abstraction**: do not hard-code an algorithm or key size in business code; expose versioned, negotiable cryptographic profiles (algorithm identifier, size, suite)
- **Secure firmware update**: the secure boot chain described in the previous chapter is the natural vector for updating cryptographic libraries; it must be designed to support library replacements without full product recertification
- **Delegation to an upgradable secure component**: on a constrained MCU, delegating PQC operations to an SE or TPM whose firmware is signed and updatable reduces the risk of hardware redesign
- **HNDL prioritization**: treat first the data whose confidentiality must resist beyond the 5-to-10 year horizon; deploy hybrid mechanisms on these flows before any other optimization concern

NIST/ANSSI normative watch must be organized (a six-month cadence is recommended) to follow the evolution of the deprecation timeline and any corrections to the standardized algorithms [57, 6, 54, 55].

B

Physical interfacing and communication protocols

B.1 Bus choice and attack surface

The buses typically used to connect a host MCU to an SE, TPM or external TEE module are I2C, SPI, UART and SWP. I2C is simple and cheap but exposed to sniffing on a shared bus; SPI offers better throughput and useful predictability for a TPM under sustained activity; UART, convenient for development and provisioning, must be disabled or strictly authenticated in production; SWP is specific to SIM/eUICC contexts and belongs to already-regulated operator ecosystems [25, 30, 3]. In all cases, the absence of logical protection on the bus leaves physical interception possible: the authenticity and confidentiality of critical exchanges must be ensured by application-layer cryptographic protections, regardless of the chosen bus [61, 25].

B.2 Electrical constraints and physical security

Logic levels, timings and ESD protections are not without security implications: physical errors or transients can open windows for fault attacks (voltage glitches, laser injection) or bypass signature verifications [12, 13]. The communication channel must be defended against sniffing, replay, frame injection and local MITM through nonces, monotonic counters and session binding to the component's hardware identity [61, 25, 30]. Low-power optimizations (frequency scaling, wake-on-interrupt, power gating) are compatible with security provided they do not remove re-authentication at wake-up [3, 27].

B.3 Debug and validation

Interface validation must include abuse tests: glitches, voltage variations, synchronization losses, frame fuzzing. The objective is to verify that the system fails safely (*fail-secure*) and that debug interfaces (JTAG, debug UART) are disabled or protected before production release [47, 27].

C

System integration and operational considerations

C.1 Host microcontroller choice and PCB layout

The choice of the host MCU must balance available resources, embedded hardware security primitives (TrustZone-M, OTP flash, key manager), certifiable tooling and long-term support [9, 67, 52]. Every layout decision must be weighed against a concrete threat: placing the SE as close as possible to the MCU reduces trace exposure to probing and EMI injection; filtering the secure component's power supply mitigates glitches; local shielding raises the cost of radiation-measurement attacks; mechanical or electrical anti-tamper measures raise the cost of physical access [34, 13, 17].

C.2 Development chain and regulatory context

The integrated drivers and libraries must be managed in an SSDF-compliant pipeline, with code reviews, SBOM, provenance control and security regression tests [52, 24]. The product must explicitly trace its regulatory obligations: Cyber Resilience Act for connected products placed on the EU market [24], radio-cyber requirements through the RED delegated regulation [23], possible sector requirements (IEC 62443 for OT environments, ISO 27001 for the management system), and an evaluation strategy according to the target market (Common Criteria, FIPS 140-3) [17, 49, 27].

C.3 Maintenance in operational conditions (MCO)

Security MCO must cover security event monitoring, periodic key rotation, signed firmware updates and an incident response plan that can be activated at fleet scale. For large-scale IoT deployments, mass revocation processes (certificates, provisioning keys) must be designed from the initial architecture and not added afterwards [53, 24, 31].

D

Certification strategy and normative traceability

This chapter equips the audit/compliance axis announced in the introduction: choosing an evaluation scheme suited to the target market, mapping the guide's security objectives to the regulatory and normative frameworks, then building the evidence file usable in an audit.

D.1 Choosing an evaluation scheme

The evaluation scheme must be chosen according to the target market, the expected assurance level and the budget, distinguishing the evaluation of the **component** (SE, TPM) from that of the **product** integrating it. A certified component does not certify the product: assurance composes, it does not transfer.

Scheme	**Typical scope**	**Effort**	**Recommended use**
Common Criteria (EAL) [17]	Secure component (SE, smart card)	High	Strong assurance on the component; payment, identity, government markets
CSPN [2]	Finished product, focused scope	Moderate (fixed workload)	First trust level in France; time-boxed black-box evaluation
SESIP / EN 17927 [26]	IoT platform (MCU + RoT + services)	Moderate	Component-to-product reuse; documented bridges to EN 303 645 and the CRA
PSA Certified [11]	MCU root of trust (levels 1 to 3)	Low to moderate	Graduated assurance on microcontroller RoTs
FIPS 140-3 [49]	Cryptographic module	High	North-American markets and federal requirements
ETSI EN 303 645 [21]	Consumer connected device	Low	Minimal consumer baseline; basis of the RED radio-cyber requirements [23]

Table D.1: Overview of the evaluation schemes applicable to secure embedded products

In practice, the most economical strategy is to compose: select an SE or TPM already certified (Common Criteria, FIPS 140-3) to carry the key-protection assurance, then evaluate the integrating product at an appropriate level (CSPN or SESIP), the product evaluation then focusing on integration, the boot chain and the interfaces rather than on the component's physical

resistance [17, 2, 26]. For regulated sectors, the sector schemes come on top: IEC 62443 for industrial [27], ISO/SAE 21434 for automotive [32].

D.2 Mapping the security objectives to the frameworks

The following table links the guide's security objectives OS-01 to OS-05 to the main regulatory and normative requirements. It is the entry point of the compliance file: every control implemented in the technical chapters is attached to an OS objective, hence to an external requirement.

Objective	**CRA [24]**	**IEC 62443-4-2 [27]**	**ISO 27001:2022 [31]**	**NIST CSF 2.0 [53]**
OS-01 Confidentiality of critical assets	Annex I: protection of data confidentiality	CR 4.1 (information confidentiality)	A.8.24 (use of cryptography)	PR.DS
OS-02 System integrity	Annex I: integrity protection, update security	CR 3.4, EDR 3.14 (software and boot integrity)	A.8.9, A.8.19 (configuration, installation)	PR.PS
OS-03 Access control to sensitive functions	Annex I: access control, secure default configuration	CR 1.1, CR 2.1 (authentication, authorization)	A.5.15, A.8.2 (access control, privileges)	PR.AA
OS-04 Resistance to physical attacks	Annex I: attack surface reduction	EDR 3.11 (resistance to physical tampering)	A.7 (physical controls)	PR.IR
OS-05 Traceability and audit	Annex I: logging; vulnerability handling	CR 2.8 to 2.12 (auditable events, timestamps)	A.8.15, A.8.16 (logging, monitoring)	DE.CM

Table D.2: Mapping of objectives OS-01 to OS-05 to the regulatory and normative frameworks

References to the requirements are deliberately given at the family level: the exact breakdown (CRA articles, detailed IEC requirements) depends on the product class and must be established in the project's compliance matrix.

D.3 Building the audit evidence file

The checklists (`*-CL-*`) and test suites (`*-T-*`) of the technical chapters are designed to directly produce the evidence expected in an audit. The minimal file includes:

- the risk analysis and the threat model, kept up to date (EBIOS RM or equivalent) [5];
- the versioned policies: cryptographic policy, key policy, secure boot policy;
- the execution reports of the CARTE-T, GPP-T, TRX-T, PROV-T and CRYPTO-T test suites, with their pass criteria;
- the SBOM and the signed build register of the CI/CD chain [52];
- the operational evidential logs and the vulnerability handling and notification procedure required by the CRA [24];

- for certified products: the component certificates and evaluation reports, with the demonstration that the component's usage assumptions are met in the product.

The consistency between these documents matters as much as their existence: an auditor will check that a control declared in the compliance matrix corresponds to an implemented requirement, tested by an identified campaign, and monitored in operation.

E

Normative references and specifications

- ISO/IEC 11889 (TPM 2.0) [30]
- ISO/IEC 15408 (Common Criteria) [17]
- ISO/IEC 27001:2022 (security management) [31]
- IEC 62443-4-2 (IACS technical requirements) [27]
- NIST SP 800-193 (Platform Firmware Resiliency) [47]
- NIST SP 800-155 (BIOS Integrity Measurement) [43]
- NIST SP 800-57 (Key Management) [51]
- GlobalPlatform Card + SCP [25]
- ANSSI CSPN (first-level security certification) [2]
- SESIP / EN 17927 (IoT platform evaluation) [26]
- ETSI EN 303 645 (consumer IoT baseline) [21]
- IEEE 802.1AR (Secure Device Identity) [28]
- FIPS 140-3, FIPS 203, FIPS 204 [49, 54, 55]
- CRA (EU 2024/2847) and RED Delegated Act (EU 2022/30) [24, 23]

Bibliography

[1] ANSSI. Recommandations de securite relatives aux mecanismes cryptographiques. Guide ANSSI, 2020.

[2] ANSSI. Certification de securite de premier niveau (cspn): referentiel. Schema de certification ANSSI, 2023.

[3] ANSSI. Recommandations de securite pour les objets connectes. Guide ANSSI, 2023.

[4] ANSSI. Securite de la chaine d'approvisionnement numerique. Guide ANSSI, 2023.

[5] ANSSI. Ebios risk manager. Methode de gestion des risques, 2024.

[6] ANSSI. Migration vers la cryptographie post-quantique: recommandations. Note de position ANSSI, 2024.

[7] ANSSI WooKey Project Contributors. Wookey architecture and design rationale. Project documentation, 2019.

[8] Arm. Arm security technology: Building a secure system using trustzone technology. Documentation technique Arm, 2019.

[9] Arm. Armv8-m security extensions: Requirements on development tools. Documentation technique Arm, 2021.

[10] Arm. Psa cryptography api specification. Specification, 2024.

[11] Arm PSA Certified. Psa certified security model and guidance. Program documentation, 2024.

[12] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, 1997.

[13] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, 2001.

[14] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *CHES*, 2004.

[15] Jo Van Bulck et al. Foreshadow: Extracting the keys to the intel sgx kingdom with transient out-of-order execution. In *USENIX Security*, 2018.

[16] CISA, NIST, and NSA. Post-quantum cryptography roadmap and transition guidance. US Government guidance, 2025.

[17] Common Criteria Recognition Arrangement. Common criteria for information technology security evaluation (iso/iec 15408). Norme internationale, 2022.

[18] Debian Security Team. Dsa-1571-1: Predictable random number generator in openssl. Security advisory, 2008.

[19] Arnaud Ebalard, Arnaud Fontaine, David Diallo, Loic Flori, Karim Khalfallah, Mathieu Renard, and Ryad Benadjila. Eurisko : developpement d'une carte electronique securisee. Actes SSTIC 2016, 2016.

[20] Maik Ender, Amir Moradi, and Christof Paar. The unpatchable silicon: A full break of the bitstream encryption of xilinx 7-series fpgas. In *USENIX Security*, 2020.

[21] ETSI. En 303 645: Cyber security for consumer internet of things: Baseline requirements. ETSI European Standard, 2024.

[22] ETSI. Migration strategies and recommendations for quantum-safe cryptography. ETSI technical report, 2024.

[23] European Commission. Commission delegated regulation (eu) 2022/30 supplementing directive 2014/53/eu. Official Journal of the European Union, 2022.

[24] European Union. Regulation (eu) 2024/2847 (cyber resilience act). Official Journal of the European Union, 2024.

[25] GlobalPlatform. Globalplatform card specification v2.3.1. Specification, 2018.

[26] GlobalPlatform. Security evaluation standard for iot platforms (sesip), en 17927. Evaluation methodology, 2023.

[27] IEC. Iec 62443-4-2: Technical security requirements for iacs components. Norme internationale, 2019.

[28] IEEE. Ieee std 802.1ar-2018: Secure device identity. IEEE Standard, 2018.

[29] Intel. Intel software guard extensions (intel sgx) sdk developer reference. Documentation Intel, 2023.

[30] ISO/IEC. Iso/iec 11889: Trusted platform module. Norme internationale, 2015.

[31] ISO/IEC. Iso/iec 27001:2022 information security management systems. Norme internationale, 2022.

[32] ISO/SAE. Road vehicles – cybersecurity engineering (iso/sae 21434). Norme ISO/SAE, 2021.

[33] Paul Kocher et al. Spectre attacks: Exploiting speculative execution. In *IEEE S&P*, 2019.

[34] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, 1999.

[35] Linaro and OP-TEE Contributors. Op-tee documentation. Project documentation, 2025.

[36] Moritz Lipp et al. Meltdown. In *USENIX Security*, 2018.

[37] lowRISC and OpenTitan contributors. Opentitan documentation and hardware specification. Project documentation, https://opentitan.org/, 2025.

[38] Roel Maes. Physically unclonable functions: Constructions, properties and applications. *Springer*, 2013.

[39] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of fpga bitstream encryption against power analysis attacks. In *ACM CCS*, 2011.

[40] Michele Mosca. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy*, 2018.

[41] NIST. Sp 800-38d: Recommendation for block cipher modes of operation: Galois/counter mode (gcm). Special Publication, 2007.

[42] NIST. Sp 800-147: Bios protection guidelines. Special Publication, 2011.

[43] NIST. Sp 800-155: Bios integrity measurement guidelines. Special Publication, 2012.

[44] NIST. Sp 800-88 rev. 1: Guidelines for media sanitization. NIST Special Publication, 2014.

[45] NIST. Fips 180-4: Secure hash standard (shs). Federal Information Processing Standard, 2015.

[46] NIST. Sp 800-90a rev. 1: Recommendation for random number generation using deterministic random bit generators. Special Publication, 2015.

[47] NIST. Sp 800-193: Platform firmware resiliency guidelines. Special Publication, 2018.

[48] NIST. Sp 800-90b: Recommendation for the entropy sources used for random bit generation. Special Publication, 2018.

[49] NIST. Fips 140-3: Security requirements for cryptographic modules. Federal Information Processing Standard, 2019.

[50] NIST. Sp 800-131a rev. 2: Transitioning the use of cryptographic algorithms and key lengths. Special Publication, 2019.

[51] NIST. Sp 800-57 part 1 rev. 5: Recommendation for key management. Special Publication, 2020.

[52] NIST. Sp 800-218: Secure software development framework (ssdf). Special Publication, 2022.

[53] NIST. Cybersecurity framework (csf) 2.0. Framework, 2024.

[54] NIST. Fips 203: Module-lattice-based key-encapsulation mechanism standard. Federal Information Processing Standard, 2024.

[55] NIST. Fips 204: Module-lattice-based digital signature standard. Federal Information Processing Standard, 2024.

[56] NIST. Fips 205: Stateless hash-based digital signature standard. Federal Information Processing Standard, 2024.

[57] NIST. Nist ir 8547: Transition to post-quantum cryptography standards. NIST Interagency Report, 2024.

[58] NIST. Post-quantum cryptography standardization project – status reports. Project documentation, 2025.

[59] Oracle. Java card platform specification, version 3.1. Specification, 2022.

[60] OWASP. Threat modeling cheat sheet. OWASP Cheat Sheet, 2024.

[61] Eric Rescorla. The transport layer security (tls) protocol version 1.3. RFC 8446, 2018.

[62] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, 1994.

[63] Ned Smith, Laurence Lundblade, et al. Remote attestation procedures (rats) architecture. RFC 9334, 2023.

[64] The Update Framework. The update framework specification. Specification, 2024.

[65] Tropic Square. Tropic square. Company website, https://www.tropicsquare.com/, 2025.

[66] Trusted Computing Group. A practical guide to tpm 2.0 and tcg tpm 2.0 library overview. TCG documentation, 2023.

[67] TrustedFirmware.org. Trusted firmware-m documentation. Project documentation, 2025.

[68] Uptane Alliance. Uptane standard for design and implementation. IEEE-ISTO Standard, 2023.

[69] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *IEEE S&P*, 2015.

License