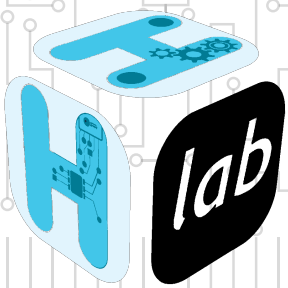
H2LAB-STIG-MAJ-001
07-2026

Embedded Systems: Guide to Secure Updates



STIG H²Lab

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 bestpractices in security, privacy, and sustainability across the ecosystemof 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.

Contents

1
Glossary and acronyms

1.1 Glossary

Trust anchor. Immutable data (public key or hash) used to establish the boot and update verification chain.

Anti-rollback. Mechanism preventing the installation of an earlier version, potentially vulnerable, even if properly signed.

Active bank. Firmware location currently executed at boot.

Bank inactive. Target firmware location of the next update.

Boot vector / vector table. Table associating interrupts/exceptions with their managers; its integrity conditions the control of the execution flow.

Fail-closed. Default security behavior in case of error: reject rather than accept.

GoT (Global Offset Table). Table of pointers/offsets resolved upon loading, allowing to redirect global references without rewriting all the code.

Boot state machine. Automaton controlling transitions *pending/confirmed/rollback/recovery*.

Update manifest. Signed metadata describing version, compatibility, constraints installation requirements, footprints and associated policies.

PI-lite. Pragmatic variant of the position-independent code, limiting relocation to critical blocks to contain the RAM/Flash/performance cost.

Power-fail safe. Property of a mechanism resilient to power failures during critical operation.

Secure boot. Cryptographic verification of integrity and authenticity software stages before execution.

TOCTOU. *Time Of Check To Time Of Use*: inconsistency between checked object and the object actually used.

Wear-leveling. Distributing writes to the NVM to reduce local wear and extend memory life.

1.2 List of acronyms

AES-CTR. *Advanced Encryption Standard* in counter mode.

AES-GCM. *Advanced Encryption Standard* in Galois/Counter mode.

AM. Attacker profile formalized in this guide (AM-01 to AM-05).

ANSSI. National agency for information systems security.

APDU. *Application Protocol Data Unit*.

API. *Application Programming Interface*.

APT. *Advanced Persistent Threat*.

BL2. *Boot Loader stage 2* (secondary boot stage).

BLE. *Bluetooth Low Energy*.

CA. *Certificate Authority*.

CAP. Technical capacity associated with an attacker profile (CAP-01 to CAP-08).

CBOR. *Concise Binary Object Representation*.

CI/CD. *Continuous Integration / Continuous Delivery*.

COSE. *CBOR Object Signing and Encryption*.

CRA. *Cyber Resilience Act* (Regulation (EU) 2024/2847).

CRC. *Cyclic Redundancy Check*.

CRL. *Certificate Revocation List*.

CVE. *Common Vulnerabilities and Exposures*.

DEK. *Data Encryption Key*.

DICE. *Device Identifier Composition Engine*.

DFU. *Device Firmware Upgrade*.

ECC. *Error Correcting Code*.

ECDSA. *Elliptic Curve Digital Signature Algorithm*.

EM. *Electromagnetic*.

EMFI. *ElectroMagnetic Fault Injection*.

ENC. *Encryption* (use of encryption key).

ETSI. *European Telecommunications Standards Institute*.

EU. *European Union*.

FIPS. *Federal Information Processing Standards.*

FS. Security feature required in this guide (FS-01 to FS-09).

GOT. *Global Offset Table.*

HKDF. *HMAC-based Key Derivation Function.*

HMAC. *Hash-based Message Authentication Code.*

HTTP. *HyperText Transfer Protocol.*

HW. *Hardware.*

HSM. *Hardware Security Module.*

ICS. *Industrial Control System.*

ID. Identifier.

IEC. *International Electrotechnical Commission.*

IP. *Internet Protocol.*

IV. *Initialization Vector.*

JTAG. *Joint Test Action Group* (debug interface).

KDF. *Key Derivation Function.*

KMS. *Key Management Service.*

MAC. *Message Authentication Code.*

MCU. *Microcontroller Unit.*

MFA. *Multi-Factor Authentication.*

MITM. *Man-In-The-Middle.*

ML-DSA. *Module-Lattice Digital Signature Algorithm.*

ML-KEM. *Module-Lattice Key Encapsulation Mechanism.*

MMIO. *Memory-Mapped Input/Output.*

MMU. *Memory Management Unit.*

MPU. *Memory Protection Unit.*

MQTT. *Message Queuing Telemetry Transport.*

NA4. *Naturally Aligned 4-byte region* (PMP RISC-V mode).

NAPOT. *Naturally Aligned Power-Of-Two* (PMP RISC-V mode).

NIST. *National Institute of Standards and Technology.*

NISTIR. *NIST Interagency/Internal Report.*

NOR. NOR Flash memory family.

NVM. *Non-Volatile Memory.*

OCSP. *Online Certificate Status Protocol.*

OS. Security objective in this guide (OS-01 to OS-08).

OTA. *Over-The-Air* (remote update).

OTP. *One-Time Programmable.*

PC. *Program Counter.*

PCB. *Printed Circuit Board.*

PCROP. *Proprietary Code Read-Out Protection.*

PI. *Position-Independent.*

PIC. *Position-Independent Code.*

PIE. *Position-Independent Executable.*

PKI. *Public Key Infrastructure.*

PMP. *Physical Memory Protection* (RISC-V).

PQC. *Post-Quantum Cryptography.*

PSA. *Platform Security Architecture.*

RAM. *Random Access Memory.*

RCE. *Remote Code Execution.*

RDP. *Readout Protection.*

RDP2. Strictest STM32 reading protection level (level 2).

RIoT. *Robust Internet of Things* identity architecture.

RISC-V. *Reduced Instruction Set Computer V.*

RMA. *Return Merchandise Authorization* (maintenance/factory return).

ROM. *Read-Only Memory.*

RoT. *Root of Trust.*

RV32. 32-bit RISC-V family.

SAU. *Security Attribution Unit* (TrustZone-M).

SBOM. *Software Bill of Materials.*

SCADA. *Supervisory Control And Data Acquisition.*

SCP03. *Secure Channel Protocol 03* (GlobalPlatform).

SE. *Secure Element.*

SHA-256. *Secure Hash Algorithm 256-bit.*

SIS. *Safety Instrumented System.*

SKU. *Stock Keeping Unit.*

SOC. *Security Operations Center.*

SP. *Special Publication* (NIST SP).

SUIT. *Software Updates for Internet of Things.*

SWD. *Serial Wire Debug.*

TF-M. *Trusted Firmware-M.*

TLS. *Transport Layer Security.*

TOCTOU. *Time Of Check To Time Of Use.*

TOR. *Top Of Range* (PMP RISC-V mode).

TUF. *The Update Framework.*

UART. *Universal Asynchronous Receiver-Transmitter.*

UICC. *Universal Integrated Circuit Card.*

USB. *Universal Serial Bus.*

USB-IF. *USB Implementers Forum.*

VTOR. *Vector Table Offset Register* (Cortex-M).

XIP. *eXecute In Place.*

2
Introduction and scope

2.1 Guide Objective

This document provides a reference architecture for securely updating microcontrollers in constrained contexts: low RAM, internal Flash with high erasure granularity, absence of complete memory abstraction (no MMU, sometimes minimalist MPU), and strong requirements continuity of service. The goal is to make the architecture update:

- resistant to modern attacks on the update chain;
- operationally robust (power failure, unexpected reset, partial image);
- compatible with platforms with or without dual-bank NVM support.

2.2 Target audience

The guide targets embedded architects, cyber product teams, firmware platform managers, PKI/DevSecOps teams and compliance assessors (IEC 62443, ETSI EN 303 645, NISTIR 8259A, PSA Certified).

2.3 Technical scope and assumptions

2.3.1 Target Equipment Type

The scope mainly covers:

- MCU Cortex-M/RISC-V IoT and industrial classes;
- firmware storage in internal Flash or external NOR executed in XIP;
- initial boot from manufacturer ROM then secondary boot chain type MCUboot, TF-M BL2 or equivalent [21, 46, 6].

2.3.2 Architectures covered: Cortex-M and RV32

The document explicitly covers two families of targets:

- **Arm Cortex-M**: management of the boot vector via vector table and VTOR, memory-mapped protection with MPU and, depending on SKU, additional partitioning via TrustZone-M [18].
- **RISC-V RV32**: privilege model `M/S/U`, exception routing via `mtvec`, and memory access control by **PMP** (TOR, NAPOT, NA4 modes) configured by secure boot in M-mode [40, 18].

On RV32, the practical equivalent of a "secure boot + anti-tamper software" policy relies on the triptych: immutable anchor, locked PMP policy (bit `L`) and strict isolation of update metadata.

2.3.3 Constraints of an MCU without memory abstraction

On MCU without advanced memory abstraction:

- code is often tied to fixed addresses (interrupt vectors, absolute sections);
- dynamic relocation is limited or even absent;
- the protection of critical areas is based on hardware security options (RDP, PCROP, secure boot fuse, TrustZone-M) rather than virtual isolation.

This requires a lot of work on the linker script, the vector table (VTOR), and the constraints inter-bank calls if we aim for behavior close to the position-independent code.

2.3.4 Essential technical reminders

Wear-leveling. Update lifecycle markers should not write in place on the same Flash cell. You must log in a ring, use monotonous sequence numbers, and validate each record via CRC/ECC.

PIC / PI-lite. The complete PIC approach remains expensive on constrained MCUs. The industrial compromise often consists of making only critical sections relocatable. (boot trampoline, interrupt stubs, service pointers) and to limit absolute references.

Boot vector. On Cortex-M, vector relocation is centralized via VTOR. On RV32, the equivalent is done via `mtvec` with distinction between direct and vectored mode; the bootloader→application transition must ensure consistency of exception handlers.

2.3.5 Operation and maintenance assumptions

The system is supposed to be operational 24/7, with periodic updates to the fleet. Threats include remote attacker, local attacker with debug access, and physical attacker with fault injection capabilities [38, 26].

3

Cyber context and rationale for A/B architecture

3.1 Targeted threats

Firmware tampering. The basic attack consists of injecting a binary not authorize or modify a legitimate artifact in transit. The teachings of the security of software deposits (TUF/Uptane) show that the signature alone is not enough without consistent metadata, expiration, role delegation and anti-replay [44, 47].

Downgrade and update replay. Old firmware but correctly sign can reintroduce a corrected CVE. Mitigation is based on monotonic counter in hardware (OTP/eFuse) or minimum version policy enforceable in the *boot check* [24, 25].

Compromise of the distribution chain. Supply type incidents chain (e.g. SolarWinds) require isolating signing roles, logging any emission and to have revocation mechanisms operable in degraded mode [31, 9].

Feedback from known attacks. The following cases structure the state of the art defender. Beyond the name of the incident, security engineering must explain the chronology, the type of attacker, the targeted asset and the gain operational hopefully. This reading facilitates the passage of "feedback" towards testable requirements on the update chain.

Risk of denial of service during an update. DoS can be intentional (invalid image repeated, oversized payload) or accidental (power cut). A/B is mainly chosen to convert a miss of update in backtracking control rather than in irreversible deactivation.

3.2 Attacker model and technical capabilities

The threat model is formalized with two dimensions:

- **AM-* Profiles**: position of the attacker in the system (remote, local, physical).
- **Capacities CAP-***: technical means that can be effectively controlled.

Case (date)	Attacker	Attack	Attack	Associate gain
Stuxnet (2010)	Highly capable state actor (convergent public attribution)	Software chain and industrial automatons	ICS/SCADA environment and physical processes	Discrete sabotage, persistence, kinetic effect via malicious code signed or perceived as legitimate [15]
Jeep Uconnect (2015)	Offensive researchers, scenario transposable to an opportunistic remote attacker	Remote surface infotainment/telematics, network pivot to internal buses	Connected vehicle and critical functions	Remote control of vehicle functions, security/security impact demonstration [23]
TRITON/TRISIS (2017)	Advanced group OT target (publicly attributed)	Compromise of engineering positions then security components	SIS/industrial security automation systems	Neutralization of safety barriers, increase in major incident potential [10]
SolarWinds (2020)	APT player in the software supply chain	Corruption of the compromised software integration/publication and distribution chain	Publisher, integrators, end customers	Mass distribution of backdoors via legitimate update mechanism [9]
Fault injection MCU (2017–2024)	Local/physical attacker with laboratory	Voltage/clock glitch, EMFI, bypass debug/boot controls	Microcontrollers in possession attacker	Extraction secrets, secure boot/readout protection bypass, large-scale attack preparation [37, 5, 26]

Table 3.1: Context of reference attacks

3.3 Security objectives associated with dual banking

The security objectives are standardized below under a name `OS-*`.

Required security features (FS-*). Security features implementables are identified under a name `FS-*` and linked to the `OS-*` objectives.

Traceability threat → objectives → functions. The matrix below explains the security justification chain.

Continuity of service and security. A valid known bank must remain available until the candidate is confirmed. This rule prevents a failure transient transforms an update campaign into massive unavailability, and it imposes explicit control of boot states and attempt thresholds.

Software integrity before execution. The jump to the candidate bank does not must only take place after a complete cryptographic check: signature, compatibility target, validity window and anti-return policy. This verification must be default and auditable refusal to reduce acceptance ambiguities.

Recoverability in case of update failure. The design must be tolerant to power interruptions: any interruption between writing, marking *pending* and toggle must converge to a deterministic recoverable state. Transition metadata and attempt counters must therefore be redundant and consistent.

Traceability and auditability of changes. Each state transition boot (candidate received, validated, tested, confirmed, rollback) must be logged in redundant metadata with CRC/ECC. This audit trail feeds both incident investigation and proof of compliance.

ID	Profile	Main surface	Capacity hypothesis
AM-01	Opportunistic Remote	OTA API, IP network, edge backend	CAP-01, CAP-02, CAP-03
AM-02	Remote advance (APT)	Supply chain software, CI/CD, operational PKI	CAP-01 to CAP-05
AM-03	Local malicious maintainer	Port debug, maintenance interfaces, UART/SWD/JTAG	CAP-02, CAP-04, CAP-06
AM-04	Laboratory physics	PCB, power supply, clock, EM, memory extraction	CAP-06, CAP-07, CAP-08
AM-05	Internal ecosystem (insider)	Publication tools, artifact repository, key policies	CAP-03, CAP-04, CAP-05

Table 3.2: Attacker profiles considered

ID	Capacity	Operational Description
CAP-01	Network injection/replay	OTA payload injection, old artifact replay, partial MITM.
CAP-02	Software exploitation	application RCE, elevation of software privilege, TOCTOU on update agent.
CAP-03	Supply-chain compromise	Artifact substitution, integration/signature chain corruption, compromised packet submission.
CAP-04	Theft/abuse of secrets	Illegitimate use of publication key, CI token, internal certificates.
CAP-05	Operational sabotage	Campaign blocking, dissemination of inconsistent revocation policy, DoS control deployment plan.
CAP-06	Local debug access	Flash/RAM extraction, patch in execution, bypass of controls if they are not locked.
CAP-07	Physical fault injection	Voltage/clock glitch, EMFI, disruption during critical boot checks.
CAP-08	Reverse engineering advance	Firmware extraction, binary diff analysis, payload construction adapted to target.

Table 3.3: Threat Model Technical Capabilities

Update event log (full requirement). In addition to technical boot flags, the equipment must maintain an event log dedicated to updating, transmitted via a **third-party channel** (telemetry, SOC agent, workshop collection), explicitly outside the MAJ transport channel. The objective is to prevent a disturbance of the MAJ channel from masking the signals of detection, audit or proof of compliance.

Minimum content per event:

- standardized event identifier (ex: `DL_START`, `SIG_FAIL`, `ROLLBACK_DONE`);
- context identifiers (equipment, campaign, target version, bank A/B);
- local timestamp (or monotonic counter if clock absent), result code and failure reason;
- short hash of artifacts (truncated hash), without disclosing any secrets;
- severity level and action applied (retry, quarantine, rollback, restart).

Implementation constraints to respect:

- **MCU size limit**: terminal ring log (explicit Flash/RAM budget), with deterministic eviction policy and priority conservation of critical events;

ID	Objective	Verification criteria
OS-01	Authenticity of the image	Any active image is signed by an authorized, non-revoked key.
OS-02	Pre-execution integrity	Hash and manifest checked before any jump to candidate slot.
OS-03	Anti-rollback	Installed version v such as $v \geq v_{min}$, with local monotonic state.
OS-04	Availability in failed update	Outage during update does not result in irrecoverable bricking.
OS-05	Privileged startup/execution isolation	MPU/PMP policies prohibit writing of RoT/metadata zones during execution.
OS-06	Resistance to basic physical attacks	Simple glitch detection/response and production debug lock.
OS-07	Forensic traceability	Any boot state transition is logged and timestamped.
OS-08	Cryptographic agility	Key rotation/revocation without immobilizing the fleet.

Table 3.4: Standardized Security Goals for OTA MCU

- **Robust writing**: append only, atomic records, sequence numbers, CRC/ECC and power-fail safe recovery to avoid silent corruption;
- **Strict access rights**: writing reserved for the privileged bootloader/MAJ agent, restricted reading (authenticated maintenance, SOC), no arbitrary deletion by the application;
- **Integrity and authenticity**: local anti-tampering protection (MAC/batch signature, chaining of records, local anchor) in order to detect post-incident falsification;
- **Privacy and minimization**: no key, secret, token or raw payload in the log; personal/-possible data must be minimized and pseudonymisable;
- **Independent third-party channel**: asynchronous export and buffering to a separate channel the MAJ protocol (eg: MQTT/TLS, secure syslog, factory channel), with acknowledgments of receipt and retry;
- **Retention and exploitability**: retention window sized for forensic analysis, standardization of event codes, and possible correlation with campaign backend.

Associated verification criterion. An audit must be able to reconstruct, over a given period, the chronology *download* → *verification crypto* → *activation* → *confirmation/rollback*, even in case of outage network on the main MAJ channel.

3.4 Intrinsic limitations of the A/B approach

A/B does not correct:

- the absence of an immutable trust anchor;
- compromise of the publication signature key;
- advanced physical attacks if no hardware mitigation exists.

It also does not replace PKI governance and secure CI/CD.

ID	Security function	Implementation requirement
FS-01	Strict cryptographic verification	Mandatory manifest signature+image, versioned trust store, refusal by default.
FS-02	Anti-rollback control	Monotonic counter (OTP/eFuse priority), minimum local state version.
FS-03	Metadata transactional management	Double copy, sequence numbers, CRC/ECC, write-only addition tolerant to cuts.
FS-04	Privileged memory isolation	Locked MPU/PMP policies, startup/execution separation, M-only metadata or equivalent.
FS-05	Production debug lock	Lifecycle locks, RMA controlled, maintenance opening log.
FS-06	Fault detection and response	Control redundancy, watchdog, consistent check before jump, recovery path.
FS-07	Governance of publication keys	HSM/KMS, separation of roles, rotation, emergency revocation tested.
FS-08	OTA forensic traceability	Immutable log A/B transitions, campaign identifier, standardized failure reasons.
FS-09	Target compatibility check	HW product/board/revision check, inter-image dependencies, explicit upgrade policy.

Table 3.5: Required security features

Scenario	Dominant Capabilities	OS Objectives	Required FS Functions
Remote malicious firmware injection	AM-01 + CAP-01 / CAP-02	OS-01, OS-02, OS-08	FS-01, FS-07, FS-09
Downgrade and replay of signed image	AM-01 / AM-03 + CAP-01 / CAP-06	OS-03, OS-07	FS-02, FS-03, FS-08
Release chain compromise	AM-02/AM-05 + CAP-03/CAP-04	OS-01, OS-08	FS-01, FS-07, FS-08
Corruption metadata state A/B	AM-03 / AM-04 + CAP-06 / CAP-07	OS-04, OS-07	FS-03, FS-04, FS-08
Bypass secure boot by fault injection	AM-04 + CAP-07 / CAP-08	OS-05, OS-06	FS-04, FS-05, FS-06
OTA	AM-01 / AM-05 + CAP-01 / CAP-05	OS-04, OS-07	FS-03, FS-08, FS-09 campaign DoS

Table 3.6: Threats-objectives-functions traceability

4

Principle of dual bank (A/B) software architecture

4.1 Overview and operating principles

A robust implementation follows a three-step chain:

1. ROM manufacturer (immutable): minimum check of the next stage.
2. Security bootloader (field editable or not): update policy, image verification, anti-rollback, bank selection.
3. Application: acknowledgment of good start and supervision during execution.

This model is consistent with PSA RoT and DICE/RIOT for identity derived from boot [39, 45, 22].

4.2 Memory partitioning

Active bank (A) and passive bank (B). Recommended topology:

- dedicated bootloader region, write-protected in production;
- two firmware slots of equal size (A and B) if possible;
- persistent metadata area distinct from slots, with double copy.

This separation reduces the risk of cross-corruption between executable code and boot decision state.

Bootloader area and persistent metadata. Metadata must include version, confirmation status, attempt counter, image hash, minimum required security level and final reason for failure. They must be written with a simple transactional protocol to maintain reliable reading in the event of a power failure.

Boot state storage and validation flags. Avoid flags unprotected single-bit. Prefer a redundant structure (double recording + sequence number + CRC), written according to an add-only protocol to limit Flash wear. This choice simplifies the detection of invalid states and makes the last valid boot transition explicit.

4.3 A/B update lifecycle

Downloading and storing in inactive bank. The image is received in verified blocks (size adapted to the erase sector) with incremental verification hash to avoid a second complete read when RAM is constrained. The processing chain must also check the write bounds and global consistency before declaring the candidate bank complete.

Pre-activation cryptographic verification. Verification is valid at least:

- manifest and image signature(s);
- product/board identifier and HW constraints;
- version monotonicity;
- persistent data compatibility policy.

The acceptance policy must be identical between backend and equipment to avoid decision divergences in production.

Atomic controlled flip-flop. The flip-flop should not rewrite large volumes in critical window. We prefer a selection by active slot pointer/flag, ideally in a zone dedicated to power-loss. The operation must remain short, deterministic and easily testable in the field robustness.

Post-boot confirmation and rollback. The new image is marked *pending*. The application must issue a confirmation after health checks (drivers, connectivity, watchdog). Without confirmation in the window N starts, automatic reverse. The confirmation criteria must be stable over time, in order to compare campaigns and quickly identify quality drift.

4.4 Safe Boot State Machine

Minimum states: *VALID A*, *TEST_B*, *CONFIRMED_B*, *ROLLBACK_A*, *RECOVERY*. Each transition is conditioned by local proofs (valid signature, flag pending, test counter, watchdog status).

Model	**Principle**	**Advantages**	**Limits / preconditions**
Native dual bank A/B	Two banks NVM execute-in-place, selection by start flag	Fast rollback, little copying in execution	Requires dual bank NVM support and correct vector placement
A/B emule (single bank + external preparation area)	Candidate image in external NOR, copy or XIP before activation	MCU compatible without internal double bank	External bus attack surface, BOM cost, latency
Sector swap (MCUboot)	Progressive active/inactive swap with buffer zone	Works on single flash	Metadata complexity, increased wear, corruption window
Overwrite only + backup image	Active slot overwrite with minimal backup image	Reduced memory footprint	High risk of bricking without stable power supply

Table 4.1: Comparison of firmware update architectural models

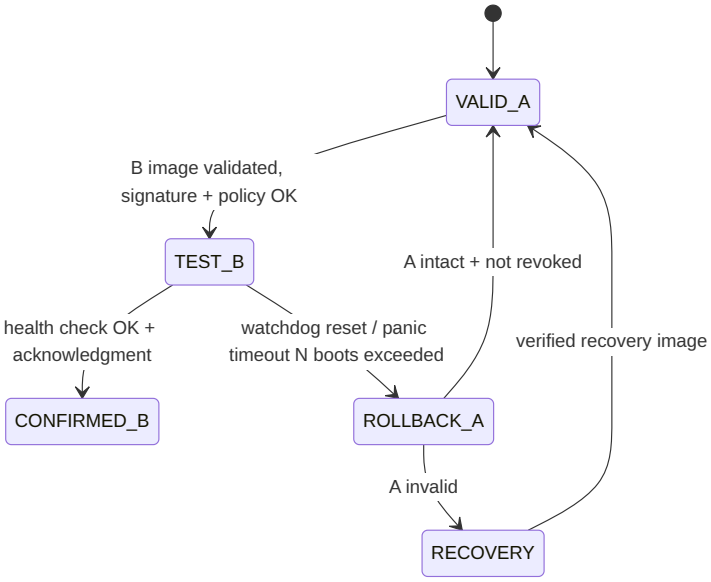


Figure 4.1: A/B update state machine

5

Specific constraints of MCUs without memory abstraction

5.1 Hardware constraints

Flash granularity, alignment and write atomicity. The pair *erase-size/program-size* directly determines the safety of the update. On many MCUs, erasing is page/sector (from 1 to 128 KiB) while programming is by words or lines. This gap imposes a strict transactional strategy:

- never rewrite a critical structure in place (metadata, counter, boot flag);
- write a new complete record then validate via final marker;
- validate consistency and integrity by CRC/ECC before taking into account.

NVM wear and operating cycle budget. Internal Flash cells are typically between 10k and 100k cycles; certain profiles industrial companies impose severe temperature/retention margins. The minimum prerequisites are:

- log *append only* for boot and anti-rollback states;
- explicit wear leveling (slot ring, page rotation, bounded collection);
- endurance budget calculated on the product lifespan (cycles/day x years);
- correlated power outage and endurance tests [43, 36].

Security hardware prerequisites often forgotten. For an A/B design to be defensible, it is necessary in practice:

- an immutable anchor (ROM, OTP, eFuse) for the root key or its hash;
- a production debug lock mechanism with separate RMA procedure;
- a power source sized for critical write windows (or supercap);
- an independent watchdog and configured from the first boot steps.

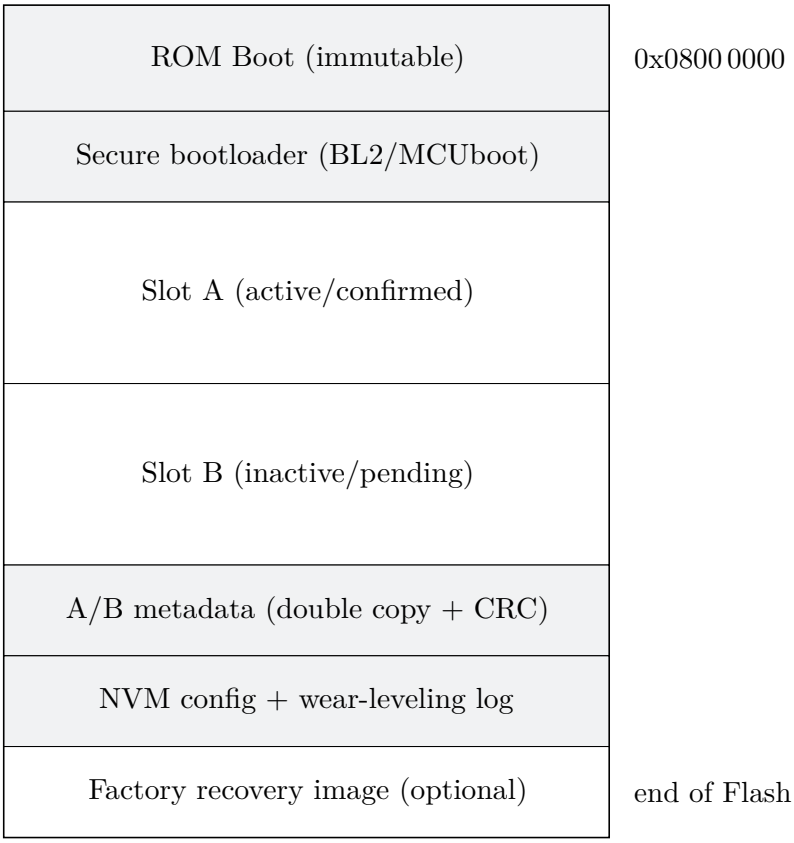


Figure 5.1: Example of A/B memory mapping on internal Flash MCU

Limited RAM and impact on cryptographic verification. Verifying a post-quantum signature can exceed RAM budgets of input MCUs of range. Viable strategies are:

- hash streaming + signature verification on compact manifest;
- offline block verification and summarized Merkle proof;
- classic hybrid approach + PQC reserved for certain product profiles.

Startup time and availability window. A long boot increases the DoS area. Checking it must be limited in time, with watchdog, retries threshold set and deterministic output path to valid known image. It is necessary to define a measurable *boot security budget* (P95/P99) and SOC alarm thresholds.

5.2 Software constraints

Fixed addressing, boot vector and PIC. Without memory abstraction, the application is often linked for a specific base. To understand the problem, we must distinguish three notions:

- **Absolute address code**: the compiler/linker emits references to fixed addresses (e.g. function to `0x08020000`, given to `0x20001000`).
- **PIC** (*Position-Independent Code*): the preferred code for relative calculations (PC-relative), so that it can be placed in several bases.

- **PIE** (*Position-Independent Executable*): full binary relocatable upon loading; on MCU, we rarely use a complete PIE, due to a rich loader like on a general OS.

In a **dynamic incremental update** scheme (added functions, removed or hot remapped), the PIE becomes necessary: the list of functions I am no longer completely stuck in production. The execution must then manage a dynamic mapping in RAM, with a loader capable of building/updating a Consistent GoT on loading to resolve symbols and global pointers.

Concretely, when the compiler generates absolute accesses, the firmware becomes couples to **a single slot address**. If this same binary is copied into the other bank, jumps and data accesses may point to the wrong area, making it the dual-bank unstable or unusable. This is the main reason for incompatibility between strictly non-PIC firmware and A/B architecture.

The role of the GOT table (*Global Offset Table*) is to avoid freezing the global addresses in each instruction, by centralizing offsets/pointers adjustable. On a microcontroller, a complete GOT can cost in Flash size, RAM and latency; we therefore often reach a compromise.

To support A/B, three strategies exist:

- two separate compilations (A and B) with fixed addresses;
- single compilation partially relocatable (position-independent code/PIC);
- minimal trampoline + dynamic VTOR to vector table of the chosen slot.

The full PIC on Cortex-M remains expensive in size/performance. In practice, a compromise *PI-lite* (relocatable critical sections, absolute limit calls) is often retained.

Related design requirements:

- versioned and continuously tested linker scripts (A/B, secure/non-secure, debug/prod);
- explicit invariants on vector table, fault handlers and low level init;
- non-regression tests on memory map migration between versions.

RV32 specifics with PMP. On RV32, the robustness of the update strongly depends on the quality of the PMP partitioning [18]. The minimum requirements are:

- region ROM/boot configured as execute-only or read-execute, locked (`L`=1);
- A/B metadata region read-only outside M-mode;
- ban in execution of any writing on the ranges containing public key, anti-rollback policy and verification code;
- explicit re-initialization of `pmpcfg/pmpaddr` at each privilege transition.

In addition, audit prerequisites must include:

- proof of coverage of PMP ranges (no unintentional executable hole);
- checking transitions `M`→`S/U` and exception paths `mtvec`;
- test for resistance to ecall abuse/interrupts on privilege border.

Low level driver dependencies. Clock, Flash controller, MPU/SAU and debug lock drivers must be compatible with both banks and initialized deterministically, otherwise backtracking may fail even before serial initialization.

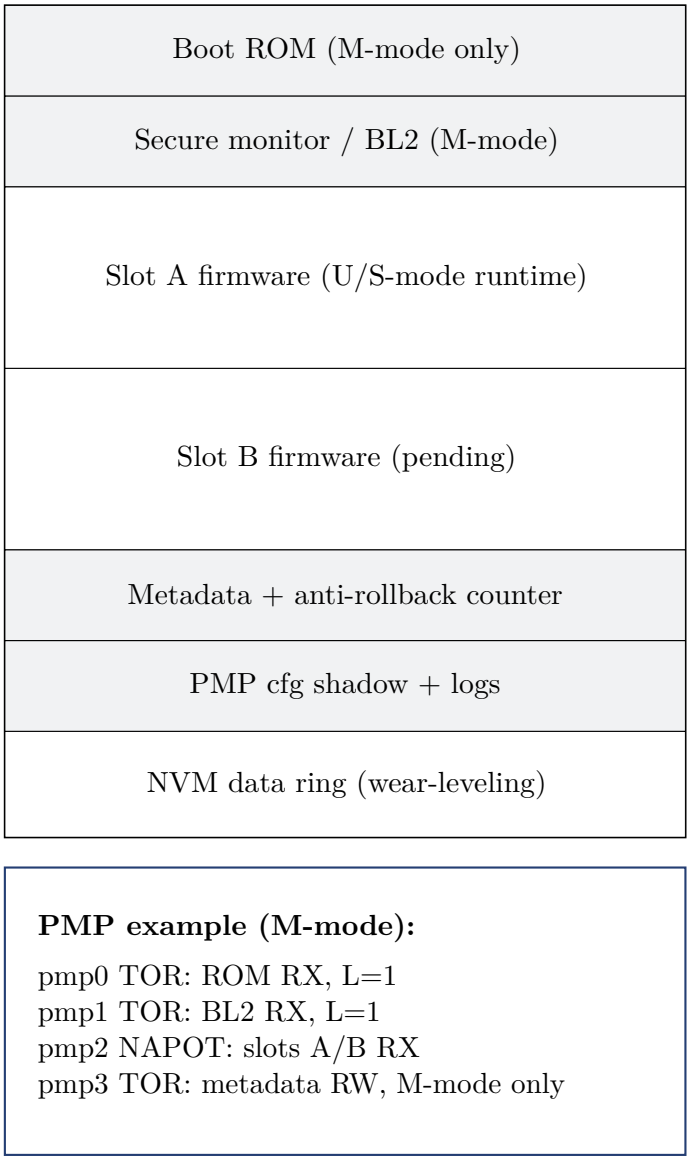


Figure 5.2: Example of RV32 memory mapping with PMP partitioning

Inter-bank compatibility and data migration. The application state (NVM configuration) must be versioned. An irreversible migration must be deferred until confirmation of the new firmware; otherwise backtracking is impossible.

Migration prerequisites:

- versioned scheme with transformers `vN`→`vN+1` and `vN+1`→`vN` when possible;
- atomic migration marker distinct from boot confirmation marker;
- application/bootloader/manifest cross-compatibility guardrails.

5.3 Operational constraints

Power loss during update. Any critical writing must be atomized into idempotent transactions. Ban the Unlogged changes to metadata.

Partial updates prohibited or strictly controlled. Binary deltas save bandwidth but increase the risk of desynchronization basic image. Their use must be reserved for strictly inventoried fleets.

Minimum conditions for accepting a delta:

- explicit fingerprint of the base image;
- post-application verification by full hash of the reconstructed image;
- automatic fallback to full image in case of deviation.

Version management and anti-rollback. The anti-rollback mechanism must be designed according to the available hardware:

- monotonic OTP/eFuse counter (ideal);
- counter in protected Flash zone with anti-tearing and cross-verification;
- strict server policy if no hardware root (lower trust level).

A fundamental prerequisite is the definition of a consistent versioning policy:

- single version semantics (bootloader, app, secondary components);
- upgrade windows allowed;
- documented exception rules (critical patch, emergency rollback).

HW constraint	**Option available**	**Security impact**	**Recommendation**
Internal dual bank NVM	Yes	Simplified atomic activation	Native A/B prioritization, delayed confirmation
Internal dual bank NVM	No	No more copying/swapping, risk of power loss	Add external staging area or robust recovery mode
Immutable storage of secrets	OTP/eFuse	Reliable anchor, strong anti-rollback	Store hash root key + monotonic counter
Immutable storage of secrets	None	High exposure to physical reflash	Compensate by box, debug lock, frequent attestation
Debug protection	Irreversible lock (RDP2, lifecycle)	Reduced key extraction and firmware dump	Activate in production with dedicated RMA procedure

Table 5.1: Structuring hardware constraints for OTA security

6

Security requirements on the update process

6.1 Organizational prerequisites

Organizational prerequisites must be explicit and auditable.

PR-ORG-01 – Separation of roles and approval circuit. Apply the principle of double control on the publication signature: compilation, security approval, signature, publication, each with separate account/department and action log.

PR-ORG-02 – Governance of keys and secrets. Keep signature keys in HSM/KMS, impose usage policies (MFA, quorum, rotation, expiration), and perform key compromise exercises.

PR-ORG-03 – Evidence and auditability. Maintain proof of reproducible compilation, artifact fingerprints, signature logs, SBOM and campaign metadata [42, 31].

6.2 Technical prerequisites on equipment side

PR-DEV-01 – Immutable Root of Trust. The verification root key must be anchored in ROM, OTP or eFuse. Any updates of this anchor must go through an exceptional path, authenticate and log.

PR-DEV-02 – Privilege isolation (Cortex-M vs RV32). To satisfy OS-05:

- on Cortex-M: strict MPU protection of boot zones + metadata, and policy separate execution between critical managers and application;
- on RV32: PMP defined in M-mode with locking, and application executed in mode least privilege with media service calls.

Recommended frame of reference for MPU-PMP objectives/countermeasures: [18].

PR-DEV-03 – Secure boot and image verification. The boot check must be minimal, auditable, and resistant to simple faults: redundant checking, flow checks, and consistent pre-jump checks.

PR-DEV-04 – Anti-rollback protection. Anti-fallback must combine manifest version and monotonic local state. Without local state, an attacker can replay an old signed binary.

PR-DEV-05 – Rollback on and deterministic. Rollback only to a previously confirmed and not revoked image.

PR-DEV-06 – Energy robustness and watchdog. The system must define: recovery points after interruption, confirmation timeout, and watchdog strategy in the activation phase.

6.3 Technical prerequisites on the backend side

PR-BE-01 – Mastered signature CI/CD. Hardened integration chain: provenance, SBOM, dependency scanning, compilation signatures and publication artifacts.

PR-BE-02 – Authenticated distribution of artifacts. Mutual TLS for critical channels, pinning of backend certificates, and application signatures independent of transport.

PR-BE-03 – Revocation and rotation of operable keys. Plan to revoke a compromised publication key without immobilizing the fleet (list of authorized keys versioned in manifest and transition policy).

6.4 Update acceptance criteria

An update is only promotable if:

- full cryptographic verification passed;
- hardware compatibility declared and validated;
- anti-rollback satisfied;
- post-boot health test successful;
- confirmation telemetry received within expected window.

7

Update cryptographic trust chain

7.1 Cryptographic Objectives

Authenticity. Guarantee that the image comes from an authorized entity.

Integrity. Guarantee the absence of alteration of payloads and metadata.

Freshness and anti-replay. Ensuring that an ancient artifact cannot be reinstalled outside of policy.

Non-repudiation. Maintain auditable signature proofs, without exposing private keys.

Crypto benchmarks to apply. The cryptographic policy must be aligned with the following benchmarks:

- **NIST**: FIPS 180-4, FIPS 186-5, SP 800-57 (key lifecycle management), SP 800-193 (platform firmware resiliency), FIPS 203/204/205 for PQC transition [27, 32, 29, 28, 34, 33, 35];
- **ANSSI**: recommendations on cryptographic mechanisms and management policies keys/certificates in a national/regulated context [1, 3, 17];
- **Europe**: product security requirements from the Cybersecurity Act and the Cyber Resilience Act (CRA), to decline in OTA controls and governance of vulnerability [12, 13].

7.2 Image format and signed manifest

State of the art on board converges on SUIT CBOR + COSE to describe and verify firmware updates [24, 25, 41].

Minimum manifest requirements. The manifest must include at least:

- product identifiers, PCB revision, SKU, hardware constraints;
- version, monotonicity, upgrade policy and validity window;
- fingerprints of all components (app, secure world, radio stack, secondary bootloader);
- inter-image dependency constraints and minimum required security level.

7.3 Recommended algorithms

Hash and KDF. SHA-256/384 remains the pragmatic basis. For local key derivatives, HKDF is recommended with explicit context and domain separation [27, 20].

Classic signatures (pre-PQC). Ed25519 offers a good compromise on MCU; ECDSA P-256 remains dominant for compatibility industrial and certification [32, 7].

Payload encryption (optional). OTA encryption is not a substitute for signing. It is relevant for protection IP/confidentiality in transit and at rest (AES-GCM for example), at the cost of complexity additional key provisioning/rotation.

7.4 Key management

Root anchor. Store the hash of the root key (or compact root certificate) in an immutable zone.

Firmware signature delegation. Use intermediate keys with a short lifespan, with an explicit delegation policy.

Rotate, expire, revoke. Design transition manifests authorizing old and new keys simultaneously on a limited window, then forcefully removing the old one.

Governance key/certificate prerequisites.

- inventory of active keys and their exact use;
- periodic rotation policy and emergency rotation;
- proof of at least annual revocation exercise;
- secure decommissioning process for obsolete keys.

7.5 Concrete example: secure update mode of the WooKey project

The WooKey project (ANSSI) is an interesting example because it combines a transport Standard USB DFU with a specific cryptographic and operational overlay to the product [4]. The update chain is not limited has late signature verification: the platform maintains authentication operator via external token, and actively involves the DFU token in deriving session keys while writing firmware blocks.

In this architecture, the update format includes a signed header (ECDSA), version metadata, IV and HMAC header protection; the body is block-encrypted (AES-CTR) to reduce usable malleability in case of fault. This point is important on MCU with limited RAM, where the image is often written in Flash before final signature verification. WooKey adds also defense in depth in DFU mode (isolation in spots and separation of roles), as well as a double bank flip-flop mechanism with anti-rollback strict at boot [4].

8

Protocols for updating microcontrollers

8.1 Objective and boundaries of an update protocol

An update protocol above all defines *how* to transfer and manage an image (status, blocks, recovery, status), but does not automatically guarantee *who* is authorized nor *what* is legitimate. In practice, it is necessary superimpose a complete cryptographic policy: authenticity of the producer, image integrity, anti-replay, anti-rollback, and actionable logging.

8.2 USB DFU: transport and orchestration standard

USB DFU (Device Firmware Upgrade) standardizes a state machine and a set of control requests for firmware upload and management (`DFU_DNLOAD`, `DFU_UPLOAD`, `DFU_GETSTATUS`, `DFU_CLRSTATUS`, `DFU_ABORT`) [48]. Its industrial interest is compatibility with existing tools and flows workshop maintenance.

DFU native limits. The DFU protocol alone does not impose the signature of the firmware, the confidentiality of the payload, nor a policy anti-rollback. It also does not define key governance or complete forensic traceability. Without a security overlay, DFU therefore remains a transport/control mechanism exposed to risks of replay, unauthorized image or host station compromise.

WooKey example on top of DFU. WooKey retains DFU interoperability, but adds an application cryptographic layer: strong authentication, derivation of session keys via token, block encryption, ECDSA signature, HMAC of the header and consistency check at boot. This approach illustrates a good general practice: keep a standard transport protocol, while moving security guarantees into an explicit cryptographic framework [4, 48].

The automaton below includes the DFU states of the USB-IF class (application branch and DFU branch) and the main operational transitions used by the update tools (DETACH,

DNLOAD, UPLOAD, GETSTATUS, ABORT, CLRSTATUS, reset) [48].

8.3 SCP03 (GlobalPlatform): secure channel for sensitive commands

SCP03 is a secure channel protocol defined by GlobalPlatform to protect APDU commands to a secure component (SE/eSE/UICC) [16]. It relies on static domain keys (typically ENC/-MAC/DEK), a host-card random exchange, then a derivation of session keys. According to the security level configured, commands can be protected by MAC (C-MAC), encrypted (C-ENC) and authenticated responses (R-MAC).

Useful properties for the MCU update. SCP03 provides confidentiality, integrity and anti-replay protection on the command channel. It is particularly relevant for provisioning secrets, managing security states, or encapsulate critical operations (image activation, key rotation, or deployment of firmware decryption keys).

Perimeter limits. SCP03 is not, in itself, a protocol of end-to-end firmware distribution: it does not replace a specification manifest, software dependencies, product anti-rollback policy, nor A/B recovery logic. It must be integrated into an architecture broader (secure boot, version metadata, transactional state, telemetry).

8.4 DFU and SCP03 comparison

Option	Highlight	Main limit	Recommended use
Native USB DFU	Interoperability and wide tooling	No native crypto product guarantees	Workshop maintenance, prototyping, transport base
DFU + crypto overlay (e.g. WooKey)	Compatibility + security applied to firmware	Integration complexity (format, keys, token, states)	Exposed products, high resilience requirements
SCP03 (GlobalPlatform)	Strongly protected command channel (MAC/encryption/anti-replay)	Does not cover complete OTA/A-B logic	Secure provisioning, critical commands, secret piloting

Table 8.1: Positioning update protocols and secure channels

8.5 Integration Recommendations for MCU

For a robust implementation, the following combination is recommended:

- standard transport protocol (DFU, HTTP(S), BLE OTA) for operability;
- signed manifest (SUIT/COSE) for cryptographic eligibility;
- hardened control channel (SCP03 or equivalent) for sensitive operations;

- transactional A/B state machine for recoverability;
- security and traceability telemetry for SOC operation and compliance.

Reminder of protocol/interface adequacy. The choice of the standard protocol must remain consistent with the communication interface actually available on the equipment. For example, HTTP or TFTP are suitable for network stacks of IP/Ethernet (or Wi-Fi) type, but are not the right choice as it stands on low-speed serial buses like SPI or I2C, which generally require a protocol lighter transport or a suitable gangway.

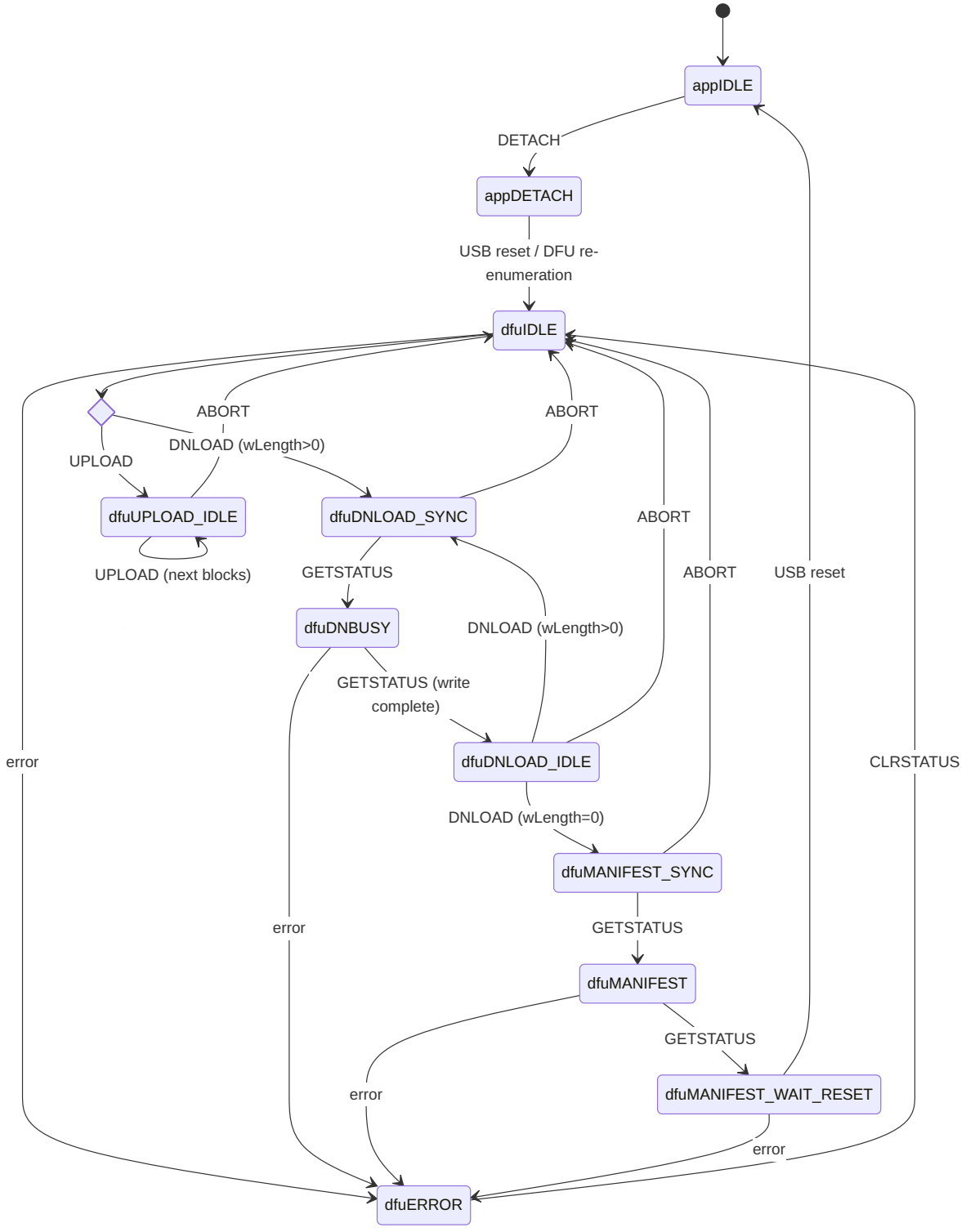


Figure 8.1: USB DFU protocol state machine (DFU USB-IF class)

9

PQC and crypto-agile transition strategy

9.1 PQC anticipation justification

The risk *harvest now, decrypt/forge later* mainly concerns long-term assets. For critical firmware with a 10-20 year life cycle, a PQC roadmap is relevant [30, 34, 33, 35].

9.2 PQC use case in the update chain

Post-quantum firmware signatures. ML-DSA (Dilithium) is the candidate main standardization side, but its key and signature sizes can penalize the most constrained MCUs. A realistic strategy is to reserve its initial use for product lines with sufficient memory margin.

Post-quantum key exchanges (if applicable). If the provisioning channel imposes an exchange of secrets, ML-KEM (Kyber) can be introduced as a priority on gateways or more capacity modules. On strict MCU, an architecture hybrid delegating part of the operations to an adjacent component is often more industrially tenable.

9.3 Classic hybrid approach + PQC

Double signature and verification policy. The recommended approach combines a mandatory classic signature and an experimental PQC signature, then gradually reverses priorities according to ecosystem maturity. This cohabitation must be explicit in the manifest and in the backend rules.

Memory, performance and boot latency impacts. The main cost is memory footprint and boot verification time. It is often preferable to verify the double signature during the activation phase, then to keep a lighter nominal reboot path when the image is already confirmed.

Progressive migration plan and rollback control. Define levels (pilot, cohabitation, generalization, classic decommission) and associate each has success criteria, alert thresholds and a feedback mechanism rear testable in conditions close to real.

9.4 Crypto-agility requirements

Version of cryptographic suites. The manifest must carry a versioned crypto suite identifier, stable and interpretable across the entire product lifespan.

Negotiation/policy on the backend and equipment side. The backend should not never impose a suite not supported locally; the equipment exposes a profile capacity sign. This negotiation must be deterministic and rejected in the event profile inconsistency.

Management of algorithmic obsolescence. Plan an end of life date algorithms and forced migration campaigns to avoid fleets in sustainable crypto debt.

10
Equipment enrollment and CA hierarchy

10.1 Industrial PKI model

Offline root CA. Keep the root of trust offline and limit its use to exceptional emissions. This discipline greatly reduces the impact of an operational compromise on the entire PKI.

Intermediate turnover by use (production, test, RMA). Separate strictly areas to avoid cross contamination. Life cycle policies and emission rights must be independent between environments.

Registration authorities and identity check. The AR must bind the hardware identity (serial, batch, SKU) has the cryptographic identity, with verification evidence retained for audit and investigation.

10.2 Reference schema: CA hierarchy, enrollment and key derivation

Interest of sub-CAs. Sub-CAs segment risks and responsibilities. A compromise in a domain (for example RMA) should not allow signing production firmware. Rotation and revocation are also more targetable, without massive re-issuance of the entire fleet or direct impact on the offline root.

Enrollment mode (PKI) versus secret injection mode. Enrollment mode assigns a specific cryptographic identity to each piece of equipment (certificate, traceability, granular revocation). Conversely, injecting a static secret sharing sometimes simplifies manufacturing, but increases the impact radius in the event of leak: a single compromise can affect an entire batch, or even a family of products.

Use of key derivation. Derivation (HKDF or equivalent) allows you to produce distinct keys by use (attestation, session update, command channel) from a hardware root secret and a context (equipment identifier, version, counter/nonce). This limits the reuse of long-term keys, reduces Lateral movements and facilitates logical rotation without complete reprovisioning.

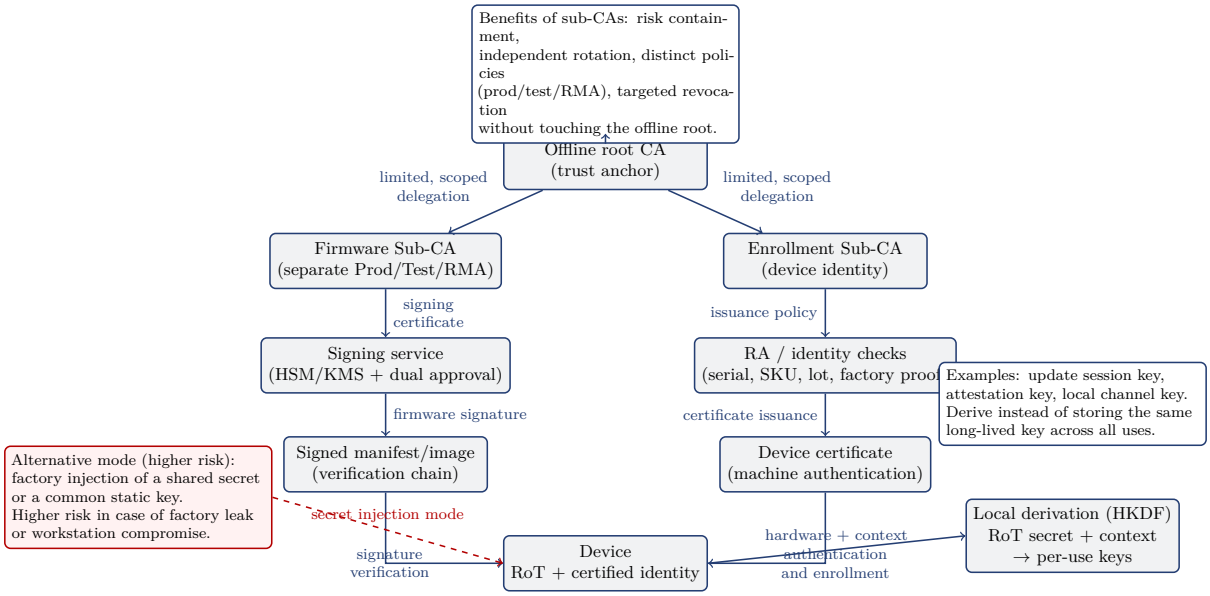


Figure 10.1: Typical CA hierarchy for firmware signature and equipment enrollment

10.3 Initial Enrollment Process

Identity injection in manufacturing. Carry out the injection in the environment secure, with sealed operation traces and proof of personalization. The supply chain must be able to demonstrate who injected what, when and under what authorization.

Provisioning of certificates and trust anchors. Prefer short and renewable certificates, with a compact on-board profile adapted to the memory constraints of the MCU.

Secure customization by batch or by unit. Single mode increases traceability but has a higher industrial cost; batch mode calls for stricter compensatory controls to maintain the same level of trust.

10.4 Certificate lifecycle

Renewal and rotation. Renewal must be anticipated before expiration to avoid crypto debt floats. The emergency rotation must be tested regularly with realistic degradation scenarios.

Revocation (CRL/OCSP or equivalent on-board mechanism). On MCU offline intermittent, the revocation may be broadcast via a policy included in the campaign manifestos. The goal is to achieve revocation coverage measurable, even with intermittent connectivity.

Management of compromised or decommissioned equipment. Switch to mode quarantine, cut off access to sensitive services and mark identity as removed to prevent unauthorized reuse.

11
Certification and proof of software conformity

11.1 Objectives of the attestation

Proof of integrity of the active firmware. The attestation must demonstrate that a measured footprint corresponds to approved firmware and a known configuration. This proof becomes usable only if it is compared to a reference mastered on the trust service side.

Proof of version and security status. The report must include at a minimum the patch level, debug status and active secure boot policy, in order to support an explicit and reproducible access decision.

11.2 Local and remote attestation

Boot measurements (secure measurements). Measure bootloader, application and security configuration in a chain of trust, with a stable format over time to facilitate verification automation.

Signed attestation report. Sign the report via an attestation key derived from the RoT (DICE/RIoT) and associate a freshness nonce [45, 22, 8]. The structure of the report must remain sober in order to be compatible with constrained terminals.

Trust service side verification. The service validates the trust chain certificates, freshness of nonce and acceptance policy before authorizing access to sensitive resources.

11.3 Link between certification and operational decision

Conditional network access permission. Access to critical APIs must be conditional on a compliant and recent certificate, according to a policy explicit risk.

triggering remediation or quarantine. In the event of non-compliance, apply gradual remediation (reduction of privileges, forced update, isolation) in order to limit the operational

impact without losing safety control.

12
Secure operation of A/B in real conditions

12.1 Deployment strategies

Canary, rolling waves, maintenance window. Campaigns must start with a representative sample then progress in waves with thresholds automatic shutdown. This deployment mode limits the scale of an incident and accelerates its detection.

Criteria for promoting a version. Promote a version only if the confirmed boot rate is stable, without abnormal rollback and without incident security on the defined observation window.

Heterogeneous fleet management. Segment by SKU, card revision, region radio and crypto capabilities to avoid global decisions unsuitable for certain technical segments.

12.2 Security monitoring and telemetry

Key indicators (boot successes, rollbacks, verification failures). Collect at least:

- cryptographic verification rate failed;
- automatic rollback rate;
- average latency activation→confirmation;
- distribution of failure reasons.

These indicators must be recorded and compared by fleet segment in order to quickly detect a local drift.

Anomaly detection and SOC alerting. Signature rejection patterns Massive attacks may indicate an attempted malicious campaign. Alert rules must distinguish quality defects from attack signals to reduce the operational noise.

12.3 Update-related incident response

Campaign blocking and emergency revocation. Provide a kill switch campaign and a revocation list distributed quickly. Triggering conditions and lifting of the blockage must be documented in advance.

Forensic and evidence collection. Keep manifests, boot logs, campaign identifier and signature proofs for post-incident investigation. The probative value depends on the quality of the timestamp and the integrity of the traces.

13
Verification, validation and compliance

13.1 A/B Testing Requirements

Robustness tests (power failure, Flash corruption). Inject cuts at each critical stage and check the absence of bricking. The results must be reproducible on several material batches.

Security tests (invalid signature, force rollback, replay). Execute a systematic negative battery including controlled fault injection, in order to validate fail-closed behavior in adverse scenarios.

Performance tests (boot time, update time). Define budgets maximum and validate under degraded thermal and voltage conditions, to guarantee a sufficient operating margin in an industrial environment.

13.2 Security criteria before production

Putting into production requires:

- audit code bootloader;
- proof of security non-regression testing;
- validation of revocation/rotation procedures;
- publication key compromise simulation.

13.3 Normative traceability and regulatory requirements

An explicit correspondence must be maintained with ETSI EN 303 645, NISTIR 8259A, IEC 62443 and ANSSI recommendations for embedded/IoT [11, 14, 19, 2].

14

Implementation recommendations and anti-patterns

14.1 Best Design Practices

- Minimal bootloader, code review, reduced dependencies.
- Transactional metadata with CRC/ECC and sequence numbers.
- Strict separation of keys (development, preprod, production).
- Hardware-anchored anti-rollback policy.
- Secure telemetry oriented to early detection.

14.2 Common mistakes to avoid

Implicit trust in network transport. Unsigned TLS application is exposed to infrastructure compromises and is not sufficient to guarantee the legitimacy of a firmware image.

No usable revocation policy. Without revocation operational, the compromise of a key quickly becomes catastrophic, because the fleet cannot be cleaned up within an acceptable time frame.

Rollback not controlled. Allowing a free downgrade cancels the efforts fixes vulnerabilities and reintroduces versions known to be weak.

Excessive coupling between bootloader and application. Strong coupling complicates manifest format migrations, slows down patching, and increases the risk of regression during platform developments.

14.3 Operational security checklist

Control	Target status	Proof expected
Immutable trust anchor (OTP/eFuse/ROM)	Mandatory	Hardware folder + fuse reading in production
Signature verification + anti-rollback at boot	Mandatory	Negative test report and refusal logs
Automatic power-fail safe rollback mechanism	Mandatory	Power cut test campaign
Revocation and rotation of operable keys	Mandatory	Procedure exercised + exercise report
Security telemetry centralized update	Highly recommended	SOC dashboard + event retention
PQC migration preparation (hybrid mode)	Recommendations	RAM/Flash/latency impact study + roadmap

Table 14.1: Minimum MCU update security checklist

15
Summary of modern attacks on the MCU update

15.1 State of the art of attacks

The most relevant attack families today are:

- **Supply chain**: compromise of artifacts or integration chain;
- **Downgrade/replay**: replay of legitimate but vulnerable images;
- **Fault injection**: voltage/clock glitch, EMFI to bypass controls;
- **Debug abuse**: SWD/JTAG reactivation, firmware dump, patch in execution;
- **TOCTOU**: verification on object different from the executed object;
- **Metadata desynchronization**: corruption of confirmation flags.

OS attack-to-objective mapping. To guide verification engineering:

- malicious firmware injection → OS-01, OS-02, OS-08;
- downgrade/replay → OS-03, OS-07;
- glitch/fault injection → OS-05, OS-06;
- metadata corruption/power-loss → OS-04, OS-07;
- compromise of publication key → OS-01, OS-08.

15.2 Attacks vs. controls matrix

Attack	Vector	Impact	Priority Controls
Malicious firmware injection	Compromised integration/repository chain	Arbitrary code execution	Offline signing, separation of roles, compilation provenance, rapid revocation
Signed image downgrade	Local/remote replay	Reopening known CVEs	OTP/eFuse monotonic counter, minimum version policy
Bypass check by glitch	Physical access, fault injection	Secure boot bypass	Redundancy controls, fault detectors, anti-tamper box, locking debug
Metadata corruption boot	Cut or write attack Flash	Undefined boot/DoS	Logging in append only, double copy + CRC, strict state machine
Publication key compromise	IS secret leak	Fleet takeover	HSM/KMS, urgent rotation, versioned trust store, certificates

Table 15.1: Synthesis matrix of attacks on OTA MCU and countermeasures

16
Conclusion

16.1 Summary of the guarantees provided by the double bank

The A/B architecture provides strong operational resilience in the face of update failures and a sound basis for applying modern cryptographic controls.

16.2 Conditions for maintaining security over time

Security remains conditional on key governance, publication discipline, and has the ability to update anti-rollback and revocation policies.

16.3 Hardening Roadmap (including PQC)

Recommended priorities:

1. Strong hardware anchoring (OTP/eFuse) + robust rollback.
2. Manifest SUIT/COSE standardization + SOC exploitable security telemetry.
3. Classic/PQC hybrid introduction on capacitive product families.
4. Generalization of remote certification for fleet risk management.

A

Appendix A – Boot state machine example

Suggested minimum states:

- `BOOT_A_CONFIRMED`
- `BOOT_B_PENDING`
- `BOOT_B_CONFIRMED`
- `ROLLBACK_TO_A`
- `RECOVERY_MODE`

Critical transitions:

- pending → confirmed only after proof of application health;
- pending → rollback on timeout, watchdog reset, or security assertion;
- rollback → recovery if previous bank invalid.

B

Appendix B – Minimum Signed Manifesto Requirements

The manifest must contain at least:

- product identifier and hardware revision;
- firmware version and anti-rollback policy;
- cryptographic fingerprint image;
- authorized verification key(s) and validity period;
- cross-component dependencies and installation preconditions.

C

Annex C – Standard key and certificate management policy

Typical policy:

- offline root, segregated online intermediates;
- periodic half-yearly or annual rotation depending on criticality;
- revocation exercise at least quarterly;
- immutable logging of emissions and withdrawals.

D

Appendix D – Threat Matrix and Associated Controls

The complete matrix must link each threat to:

- prevention controls;
- detection controls;
- remediation controls;
- associated test proof and control owner.

Bibliography

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

[2] ANSSI. Recommandations de securite relatives a l'iot et aux systemes embarques. Guides et recommandations ANSSI, 2023.

[3] ANSSI. Referentiel general de securite (rgs). Reglementation identite et confiance numerique, 2025. https://cyber.gouv.fr/reglementation/reglementation-identite-confiance-numerique/securite-echanges-voie-electronique/referentiel-general-de-securite/, accessed 2026-07-04.

[4] ANSSI WooKey Project. Wookey architecture: Cryptography, dfu mode and flip-flop mechanism. Project documentation, 2019. https://wookey-project.github.io/architecture.html, accessed 2026-07-04.

[5] Anvil Secure. Glitching stm32 read-out protection with voltage fault injection. Technical report, Anvil Secure, 2024.

[6] Arm Ltd. Arm platform security architecture firmware framework. https://developer.arm.com/architectures/security-architectures/platform-security-architecture, 2023. Accessed 2026-07-04.

[7] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2011.

[8] H. Birkholz et al. Entity attestation token (eat). IETF RFC 9711, 2024.

[9] CISA and NCSC. Solarwinds and sunburst: Lessons for software supply chain security. Public advisories and post-incident reports, 2021.

[10] Dragos Inc. Trisis: Malware that attacks safety instrumented systems. Technical report, Dragos, 2017.

[11] ETSI. Cyber security for consumer internet of things: Baseline requirements. ETSI EN 303 645, 2020.

[12] European Union. Regulation (eu) 2019/881 (cybersecurity act). Official Journal of the European Union, 2019.

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

[14] M. Fagan, K. Megas, K. Scarfone, and M. Smith. Iot device cybersecurity capability core baseline. NISTIR 8259A, 2020.

[15] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32.stuxnet dossier. Technical report, Symantec, 2011.

[16] GlobalPlatform. Secure channel protocol '03' – amendment d v1.2. GlobalPlatform Secure Element specification, 2020. Specification listing: https://globalplatform.org/specs-library/secure-channel-protocol-03-amendment-d-v1-2/, accessed 2026-07-04.

[17] Gouvernement francais. Instruction interministerielle no 901. Instruction interministerielle relative a la protection des systemes d'information sensibles, 2013.

[18] Hardware Hacking Laboratory. H2lab-stig-mpu-001: Guide technique de protection memoire physique pour systemes embarques. STIG H2LAB, 2026. https://blog.h2lab.org/stigs/nouveau-guide-technique-protection-memoire-physique/, accessed 2026-07-04.

[19] IEC. Industrial communication networks – network and system security. IEC 62443 series, 2021.

[20] H. Krawczyk and P. Eronen. Hmac-based extract-and-expand key derivation function (hkdf). IETF RFC 5869, 2010.

[21] MCUboot Project. Mcuboot secure bootloader. https://docs.mcuboot.com/, 2026. Accessed 2026-07-04.

[22] Microsoft Research and collaborators. Fido device onboard and riot-style derived device identity. Whitepapers and architecture notes, 2017.

[23] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. In *Black Hat USA*, 2015.

[24] B. Moran, H. Tschofenig, et al. A firmware update architecture for internet of things devices. IETF RFC 9019, 2021.

[25] B. Moran, H. Tschofenig, et al. A concise binary object representation (cbor)-based serialization format for suit manifests. IETF RFC 9124, 2023.

[26] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz. Voltage glitching attacks on embedded systems. In *FDTC*, 2014.

[27] NIST. Secure hash standard (shs). FIPS PUB 180-4, 2015.

[28] NIST. Platform firmware resiliency guidelines. NIST SP 800-193, 2018.

[29] NIST. Recommendation for key management, part 1: General. NIST SP 800-57 Part 1 Rev. 5, 2020.

[30] NIST. Nist announces first four quantum-resistant cryptographic algorithms. NIST News Release, 2022.

[31] NIST. Secure software development framework (ssdf) version 1.1. NIST SP 800-218, 2022.

[32] NIST. Digital signature standard (dss). FIPS PUB 186-5, 2023.

[33] NIST. Module-lattice-based digital signature standard (ml-dsa). FIPS 204, 2024.

[34] NIST. Module-lattice-based key-encapsulation mechanism standard (ml-kem). FIPS 203, 2024.

[35] NIST. Stateless hash-based digital signature standard (slh-dsa). FIPS 205, 2024.

[36] NXP Semiconductors. Secure ota update on mcu platforms. Application Notes and Security Whitepapers, 2023.

[37] Timo Obermaier and Stefan Tatschner. Shedding too much light on a microcontroller's firmware protection. In *USENIX WOOT*, 2017.

[38] Colin O'Flynn et al. Security analysis of fault injection attacks against secure boot mechanisms. Selected embedded security conference papers, 2020.

[39] PSA Certified. Psa certified: Root of trust. https://www.psacertified.org/, 2024. Accessed 2026-07-04.

[40] RISC-V International. The risc-v instruction set manual, volume ii: Privileged architecture. Ratified specification, 2024.

[41] J. Schaad. Cbor object signing and encryption (cose): Structures and process. IETF RFC 9052, 2022.

[42] SLSA Contributors. Slsa v1.0 specification. https://slsa.dev/spec/v1.0/, 2023. Accessed 2026-07-04.

[43] STMicroelectronics. Eeprom emulation techniques and flash endurance on stm32. Application Note AN4894, 2023.

[44] The Update Framework Authors. The update framework (tuf) specification. https://theupdateframework.github.io/specification/latest/, 2024. Accessed 2026-07-04.

[45] Trusted Computing Group. Tcg dice attestation architecture. TCG Specification, 2018.

[46] TrustedFirmware.org. Trusted firmware-m documentation. https://trustedfirmware-m.readthedocs.io/, 2026. Accessed 2026-07-04.

[47] Uptane Community. Uptane standard for design and implementation. https://uptane.org/, 2024. Accessed 2026-07-04.

[48] USB Implementers Forum. Universal serial bus device class specification for device firmware upgrade, version 1.1. USB-IF specification, 2004. https://www.usb.org/sites/default/files/DFU_1.1.pdf, accessed 2026-07-04.

License