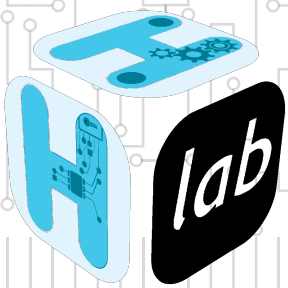
H2LAB-STIG-MAJ-001
07-2026

Systèmes embarqués : guide de la mise à jour sécurisée



STIG H²Lab

Informations

Créée en 2020, H2Lab est une association loi 1901 dont l'objectif est de concevoir, développer et diffuser des solutions open-hardware et open-source dédiées à l'expérimentation autour des systèmes embarqués.
Ses principaux axes de travail sont :

- Conception de plateformes matérielles pour l'analyse hardware et software
- Développement d'alternatives ouvertes aux outils propriétaires, souvent opaques
- Publication de guides techniques et de recommandations pour promouvoir les bonnespratiques en matière de sécurité, de confidentialité et de durabilité dans l'écosystèmedes objets connectés et des systèmes embarqués

Soutien aux chercheurs, enseignants et étudiants via la mise à disposition d'outils, de contenus pédagogiques et de formations.
Partenariats académiques pour encourager la mutualisation des ressources et des savoirs.

Table des matières

1
Glossaire et acronymes

1.1 Glossaire

Ancre de confiance. Donnée immuable (clef publique ou hash) utilisée pour établir la chaîne de vérification du boot et des mises à jour.

Anti-rollback. Mécanisme empêchant l'installation d'une version antérieure, potentiellement vulnérable, même si elle est correctement signée.

Banque active. Emplacement firmware actuellement execute au boot.

Banque inactive. Emplacement firmware cible de la prochaine mise à jour.

Boot vector / table des vecteurs. Table associant interruptions/exceptions a leurs gestionnaires ; son intégrité conditionne la maîtrise du flux d'execution.

Fail-closed. Comportement de sécurité par défaut en cas d'erreur : refuser plutôt qu'accepter.

GoT (Global Offset Table). Table de pointeurs/offsets resolus au chargement, permettant de rediriger des references globales sans réécrire tout le code.

Machine d'états de boot. Automate pilotant les transitions *pending/confirmed/rollback/recovery.*

Manifeste de mise à jour. Métadonnées signées décrivant version, compatibilité, contraintes d'installation, empreintes et politiques associées.

PI-lite. Variante pragmatique du code position-independent, limitant la relocation aux blocs critiques pour contenir le coût RAM/Flash/performance.

Power-fail safe. Propriété d'un mécanisme résilient aux coupures d'alimentation en cours d'opération critique.

Secure boot. Vérification cryptographique de l'intégrité et de l'authenticité des étages logiciels avant exécution.

TOCTOU. *Time Of Check To Time Of Use* : incohérence entre l'objet vérifié et l'objet effectivement utilisé.

Wear-leveling. Répartition des écritures sur la NVM pour réduire l'usure locale et prolonger la durée de vie mémoire.

1.2 Liste des acronymes

AES-CTR. *Advanced Encryption Standard* en mode compteur.

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

AM. Profil d'attaquant formalisé dans ce guide (AM-01 à AM-05).

ANSSI. Agence nationale de la sécurité des systèmes d'information.

APDU. *Application Protocol Data Unit*.

API. *Application Programming Interface*.

APT. *Advanced Persistent Threat*.

BL2. *Boot Loader stage 2* (étage de démarrage secondaire).

BLE. *Bluetooth Low Energy*.

CA. *Certificate Authority* (autorité de certification).

CAP. Capacité technique associée à un profil d'attaquant (CAP-01 à CAP-08).

CBOR. *Concise Binary Object Representation*.

CI/CD. *Continuous Intégration / Continuous Delivery*.

COSE. *CBOR Object Signing and Encryption*.

CRA. *Cyber Résilience 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* (électromagnétique).

EMFI. *ElectroMagnetic Fault Injection*.

ENC. *Encryption* (usage de clef de chiffrement).

ETSI. *European Telecommunications Standards Institute*.

EU. *European Union* (Union européenne).

FIPS. *Federal Information Processing Standards*.

FS. Fonction de sécurité requise dans ce guide (FS-01 à 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. Identifiant.

IEC. *International Electrotechnical Commission.*

IP. *Internet Protocol.*

IV. *Initialization Vector.*

JTAG. *Joint Test Action Group* (interface de debogage).

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* (mode PMP RISC-V).

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

NIST. *National Institute of Standards and Technology.*

NISTIR. *NIST Interagency/Internal Report.*

NOR. Famille de mémoire Flash NOR.

NVM. *Non-Volatile Memory.*

OCSP. *Online Certificate Status Protocol.*

OS. Objectif de sécurité dans ce guide (OS-01 à OS-08).

OTA. *Over-The-Air* (mise à jour distante).

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. Niveau de protection lecture STM32 le plus strict (niveau 2).

RIoT. *Robust Internet of Things* identity architecture.

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

RMA. *Return Merchandise Authorization* (retour maintenance/usine).

ROM. *Read-Only Memory.*

RoT. *Root of Trust.*

RV32. Famille RISC-V 32 bits.

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* (mode PMP RISC-V).

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 et périmètre

2.1 Objectif du guide

Ce document fournit une architecture de référence pour la mise à jour sécurisée de microcontrôleurs dans des contextes contraints : faible RAM, Flash interne avec granularité d'effacement élevée, absence d'abstraction mémoire complète (pas de MMU, MPU parfois minimaliste), et exigences fortes de continuité de service. L'objectif est de rendre l'architecture de mise à jour :

- résistante aux attaques modernes sur la chaîne d'update ;
- operationnellement robuste (coupure d'alimentation, reset intempestif, image partielle) ;
- compatible avec des plateformes ayant ou non le support dual-bank NVM.

2.2 Public visé

Le guide cible les architectes embarques, équipes cyber produit, responsables plateforme firmware, équipes PKI/DevSecOps et évaluateurs de conformité (IEC 62443, ETSI EN 303 645, NISTIR 8259A, PSA Certified).

2.3 Périmètre technique et hypothèses

2.3.1 Type d'équipement cible

Le périmètre couvre principalement :

- MCU Cortex-M/RISC-V classes IoT et industriel ;
- stockage firmware en Flash interne ou NOR externe executee en XIP ;
- boot initial depuis ROM constructeur puis chaîne de boot secondaire de type MCUboot, TF-M BL2 ou équivalent [21, 46, 6].

2.3.2 Architectures couvertes : Cortex-M et RV32

Le document couvre explicitement deux familles de cibles :

- **Arm Cortex-M** : gestion du vecteur de boot via table des vecteurs et VTOR, protection memory-mapped avec MPU et, selon SKU, cloisonnement complémentaire via TrustZone-M [18].
- **RISC-V RV32** : modèle privilège `M/S/U`, routage des exceptions via `mtvec`, et contrôle d'accès mémoire par **PMP** (modes TOR, NAPOT, NA4) configuré par le boot secure en M-mode [40, 18].

Sur RV32, l'équivalent pratique d'une politique "secure boot + anti-tamper software" repose sur le triptyque : ancre immuable, politique PMP verrouillée (bit `L`) et isolation stricte des métadonnées d'update.

2.3.3 Contraintes d'un MCU sans abstraction mémoire

Sur MCU sans abstraction mémoire avancée :

- le code est souvent lié à des adresses fixes (vecteurs d'interruption, sections absolues) ;
- la relocation dynamique est limitée voire absente ;
- la protection de zones critiques repose sur options de sécurité matérielle (RDP, PCROP, secure boot fuse, TrustZone-M) plutôt que sur isolation virtuelle.

Cela impose un travail fort sur le linker script, la table des vecteurs (VTOR), et les contraintes d'appel inter-banques si l'on vise un comportement proche du code position-independent.

2.3.4 Rappels techniques indispensables

Wear-leveling. Les marqueurs de cycle de vie d'update ne doivent pas écrire en place sur la même cellule Flash. Il faut journaliser en anneau, utiliser des séquence numbers monotones, et valider chaque enregistrement via CRC/ECC.

PIC / PI-lite. L'approche PIC complete reste coûteuse sur MCU contraints. Le compromis industriel consiste souvent à rendre relocalisables uniquement les sections critiques (trampoline de boot, stubs d'interruption, pointeurs de services) et à limiter les références absolues.

Vecteur de boot. Sur Cortex-M, la relocation du vecteur est centralisée via VTOR. Sur RV32, l'équivalent se fait via `mtvec` avec distinction entre mode direct et vectored ; la transition bootloader→application doit garantir la cohérence des gestionnaires d'exception.

2.3.5 Hypothèses d'exploitation et de maintenance

Le système est supposé opérationnel 24/7, avec mises à jour périodiques en flotte. Les menaces incluent attaquant distant, attaquant local avec accès debug, et attaquant physique avec capacités de fault injection [38, 26].

3

Contexte cyber et justification de l'architecture A/B

3.1 Menaces ciblées

Altération du firmware. L'attaque de base consiste à injecter un binaire non autorisé ou à modifier un artefact légitime en transit. Les enseignements de la sécurisation de dépôts logiciels (TUF/Uptane) montrent que la signature seule ne suffit pas sans métadonnées cohérentes, expiration, délégation de rôles et anti-rejeu [44, 47].

Downgrade et rejeu de mise à jour. Un firmware ancien mais correctement signe peut reintroduire une CVE corrigee. La mitigation repose sur compteur monotone en matériel (OTP/eFuse) ou politique de version minimale imposable dans le *boot vérifier* [24, 25].

Compromission de la chaîne de distribution. Les incidents de type supply chain (ex. SolarWinds) imposent d'isoler les rôles de signature, de journaliser toute emission et de disposer de mecanismes de révocation opérables en mode dégradé [31, 9].

Retours d'experience d'attaques connues. Les cas suivants structurent l'état de l'art defenseur. Au-dela du nom de l'incident, l'ingenierie de sécurité doit expliciter la chronologie, le type d'attaquant, l'actif vise et le gain opérationnel espere. Cette lecture facilite le passage d'un "retour d'experience" vers des exigences testables sur la chaîne de mise à jour.

Risque de déni de service lors d'une mise à jour. Le DoS peut être intentionnel (image invalide répétée, payload surdimensionné) ou accidentel (coupure énergie). L'A/B est principalement choisi pour convertir un échec d'update en retour arrière contrôle plutôt qu'en mise hors service irréversible.

3.2 Modèle d'attaquant et capacités techniques

Le modèle de menace est formalisé avec deux dimensions :

Cas (date)	Attaquant	Attaque	Attaquée	Gain associé
Stuxnet (2010)	Acteur étatique hautement capacitaire (attribution publique convergente)	Chaîne logicielle et automates industriels	Environnement ICS/SCADA et processus physiques	Sabotage discret, persistance, effet cinétique via code malveillant signé ou perçu comme légitime [15]
Jeep Uconnect (2015)	Chercheurs offensifs, scénario transposable à un attaquant distant opportuniste	Surface distante infotainment/télématique, pivot réseau vers bus internes	Véhicule connecté et fonctions critiques	Prise de contrôle à distance de fonctions du véhicule, démonstration d'impact sécurité/sûreté [23]
TRITON/TRISIS (2017)	Groupe avancé cible OT (attribué publiquement)	Compromission de postes d'ingénierie puis de composants de sûreté	SIS/automates de sécurité industrielle	Neutralisation de barrières de sûreté, augmentation du potentiel d'incident majeur [10]
SolarWinds (2020)	Acteur APT sur la chaîne d'approvisionnement logicielle	Corruption de la chaîne d'intégration/publication et diffusion logicielle compromise	Éditeur, intégrateurs, clients finaux	Distribution massive de portes dérobées via mécanisme de mise à jour légitime [9]
Fault injection MCU (2017–2024)	Attaquant local/physique avec laboratoire	Glitch tension/horloge, EMFI, bypass contrôles debug/boot	Microcontrôleurs en possession attaquant	Extraction secrets, contournement secure boot/readout protection, préparation d'attaque à grande échelle [37, 5, 26]

Table 3.1 – Contexte des attaques de référence

- **Profils AM-*** : position de l'attaquant dans le système (distant, local, physique).
- **Capacités CAP-*** : moyens techniques effectivement maîtrisables.

ID	Profil	Surface principale	Hypothese de capacités
AM-01	Distant opportuniste	API OTA, réseau IP, backend edge	CAP-01, CAP-02, CAP-03
AM-02	Distant avancé (APT)	Supply chain logicielle, CI/CD, PKI opérationnelle	CAP-01 à CAP-05
AM-03	Local mainteneur malveillant	Port debug, interfaces de maintenance, UART/SWD/JTAG	CAP-02, CAP-04, CAP-06
AM-04	Physique en laboratoire	PCB, alimentation, horloge, EM, extraction mémoire	CAP-06, CAP-07, CAP-08
AM-05	Interne écosystème (insider)	Outils de publication, dépôt d'artefacts, politiques de clef	CAP-03, CAP-04, CAP-05

Table 3.2 – Profils d'attaquants considérés

3.3 Objectifs de sécurité associés au double banque

Les objectifs de sécurité sont normalisés ci-dessous sous un nommage `OS-*`.

ID	Capacite	Description opérationnelle
CAP-01	Injection/rejeu réseau	Injection de payload OTA, rejeu d'artefacts anciens, MITM partiel.
CAP-02	Exploitation logicielle	RCE applicative, elevation de privilège logicielle, TOCTOU sur update agent.
CAP-03	Compromission supply-chain	Substitution d'artefact, corruption de chaîne d'intégration/-signature, dépôt de paquets compromis.
CAP-04	Vol/abus de secrets	Usage illegitime de clef de publication, jeton CI, certificats internes.
CAP-05	Sabotage opérationnel	Blocage de campagne, diffusion de politique de révocation incohérente, DoS contrôle plan de déploiement.
CAP-06	Accès debug local	Extraction Flash/RAM, patch en exécution, contournement de contrôles s'ils ne sont pas verrouillés.
CAP-07	Injection de faute physique	Voltage/clock glitch, EMFI, perturbation durant vérifications critiques de boot.
CAP-08	Reverse engineering avancé	Extraction firmware, analyse diff binaire, construction de payload adapté cible.

TABLE 3.3 – Capacités techniques du modèle de menace

Fonctions de sécurité requises (FS-*). Les fonctions de sécurité implementables sont identifiees sous un nommage `FS-*` et reliees aux objectifs `OS-*`.

Traçabilité menace → objectifs → fonctions. La matrice ci-dessous explicite la chaîne de justification de sécurité.

Continuité de service et de sécurité. Une banque connue valide doit rester disponible tant que la candidate n'est pas confirmée. Cette règle évite qu'un échec transitoire transforme une campagne de mise à jour en indisponibilité massive, et elle impose un pilotage explicite des états de boot et des seuils de tentative.

Intégrité logicielle avant exécution. Le saut vers la banque candidate ne doit intervenir qu'après un contrôle cryptographique complet : signature, compatibilité cible, fenêtre de validité et politique anti-retour. Cette vérification doit être de refus par défaut et auditable pour réduire les ambiguïtés d'acceptation.

Récupérabilité en cas d'échec de mise à jour. Le design doit être tolérant aux coupures d'alimentation : toute coupure entre écriture, marquage *pending* et bascule doit converger vers un état récupérable déterministe. Les métadonnées de transition et les compteurs de tentative doivent donc être redondants et cohérents.

Traçabilité et auditabilité des changements. Chaque transition d'état de boot (candidate reçue, validée, testée, confirmée, rollback) doit être journalisée dans des métadonnées redondantes avec CRC/ECC. Cette piste d'audit alimente à la fois l'investigation incident et les preuves de conformité.

Journal d'événements de mise à jour (exigence complète). En plus des flags techniques de boot, l'équipement doit maintenir un journal d'événements dédié à la mise à jour, transmissible via un **canal tiers** (télémétrie, agent SOC, collecte atelier), explicitement hors du canal de transport de la MAJ. L'objectif est d'éviter qu'une perturbation du canal MAJ masque les signaux de détection, d'audit ou de preuve de conformité.

Contenu minimal par événement :

- identifiant événement normalisé (ex : `DL_START`, `SIG_FAIL`, `ROLLBACK_DONE`) ;
- identifiants de contexte (équipement, campagne, version cible, banque A/B) ;
- horodatage local (ou compteur monotone si horloge absente), code résultat et raison d'échec ;

ID	Objectif	Critere de vérification
OS-01	Authenticité de l'image	Toute image active est signée par une clef autorisée non révoquée.
OS-02	Intégrité pre-execution	Hash et manifeste vérifiés avant tout saut vers slot candidat.
OS-03	Anti-rollback	Version installée v telle que $v \geq v_{min}$, avec état monotone local.
OS-04	Disponibilité en échec d'update	Coupure pendant update n'entraîne pas de bricking irrécoverable.
OS-05	Isolation privilégiée démarrage/exécution	Politiques MPU/PMP interdisent écriture en exécution des zones RoT/métadonnées.
OS-06	Resistance aux attaques physiques de base	Détection/réponse au glitch simple et verrouillage debug production.
OS-07	Tracabilite forensique	Toute transition d'état de boot est journalisée et horodatable.
OS-08	Agilité cryptographique	Rotation/révocation de clefs sans immobiliser la flotte.

TABLE 3.4 – Objectifs de sécurité normalisés pour OTA MCU

- empreinte courte des artefacts (hash tronqué), sans divulguer de secret ;
- niveau de gravité et action appliquée (nouvel essai, quarantaine, retour arrière, reprise).

Contraintes d'implémentation à respecter :

- **Taille limitée MCU** : journal en anneau borné (budget explicite Flash/RAM), avec politique d'éviction déterministe et conservation prioritaire des événements critiques ;
- **Écriture robuste** : en ajout seul, enregistrements atomiques, numéros de séquence, CRC/ECC et reprise power-fail safe pour éviter la corruption silencieuse ;
- **Droits d'accès stricts** : écriture réservée au bootloader/agent MAJ privilegie, lecture restreinte (maintenance authentifiee, SOC), aucune suppression arbitraire par l'application ;
- **Intégrité et authenticité** : protection anti-altération locale (MAC/signature de lot, chainage d'enregistrements, ancre locale) afin de détecter une falsification post-incident ;
- **Confidentialité et minimisation** : pas de clef, secret, jeton ou charge utile brute dans le journal ; les données personnelles/eventuelles doivent être minimisées et pseudonymisables ;
- **Canal tiers indépendant** : export asynchrone et bufferisé vers un canal distinct du protocole MAJ (ex : MQTT/TLS, syslog sécurisé, canal usine), avec accusés de réception et retry ;
- **étention et exploitabilité** : fenêtre de rétention dimensionnée pour l'analyse forensique, normalisation des codes d'événements, et corrélation possible avec backend de campagne.

Critere de verification associe. Un audit doit pouvoir reconstruire, sur une période donnee, la chronologie *telechargement* → *verification crypto* → *activation* → *confirmation/rollback*, meme en cas de coupure réseau sur le canal MAJ principal.

3.4 Limites intrinsèques de l'approche A/B

L'A/B ne corrige pas :

- l'absence d'ancre de confiance immuable ;

ID	Fonction de sécurité	Exigence d'implémentation
FS-01	Vérification cryptographique stricte	Signature manifeste+image obligatoire, magasin de confiance versionné, refus par défaut.
FS-02	Contrôle anti-rollback	Compteur monotone (OTP/eFuse prioritaire), version minimale état local.
FS-03	Gestion transactionnelle métadonnées	Double copie, numéros de séquence, CRC/ECC, écriture en ajout seul tolérante aux coupures.
FS-04	Isolation mémoire privilégiée	Politiques MPU/PMP verrouillées, séparation démarrage/exécution, métadonnées M-only ou équivalent.
FS-05	Verrouillage debug production	Lifecycle verrouillé, RMA contrôlée, journal d'ouverture maintenance.
FS-06	Détection et réponse aux fautes	Redondance de contrôles, watchdog, vérification cohérente avant saut, chemin de reprise.
FS-07	Gouvernance des clefs de publication	HSM/KMS, séparation des rôles, rotation, révocation d'urgence testée.
FS-08	Traçabilité forensique OTA	Journal immuable transitions A/B, identifiant campagne, raisons d'échec normalisées.
FS-09	Contrôle de compatibilité cible	Vérif produit/board/revision HW, dépendances inter-images, politique d'upgrade explicite.

TABLE 3.5 – Fonctions de sécurité requises

- la compromission de la clef de signature de publication ;
- les attaques physiques avancées si aucune mitigation hardware n'existe.

Il ne remplace pas non plus une gouvernance PKI et une CI/CD sécurisée.

Scénario	Capacités dominantes	Objectifs OS	Fonctions FS requises
Injection firmware malveillant distant	AM-01 + CAP-01 / CAP-02	OS-01, OS-02, OS-08	FS-01, FS-07, FS-09
Downgrade et rejeu d'image signée	AM-01 / AM-03 + CAP-01 / CAP-06	OS-03, OS-07	FS-02, FS-03, FS-08
Compromission de la chaîne de publication	AM-02 / AM-05 + CAP-03 / CAP-04	OS-01, OS-08	FS-01, FS-07, FS-08
Corruption métadonnées état A/B	AM-03 / AM-04 + CAP-06 / CAP-07	OS-04, OS-07	FS-03, FS-04, FS-08
Bypass secure boot par fault injection	AM-04 + CAP-07 / CAP-08	OS-05, OS-06	FS-04, FS-05, FS-06
DoS de campagne OTA	AM-01 / AM-05 + CAP-01 / CAP-05	OS-04, OS-07	FS-03, FS-08, FS-09

TABLE 3.6 – Traçabilité menaces-objectifs-fonctions

4

Principe de l'architecture logicielle double banque (A/B)

4.1 Vue d'ensemble et principes de fonctionnement

Une implémentation robuste suit une chaîne en trois étages :

1. ROM constructeur (immutable) : vérification minimale de l'étage suivant.
2. Bootloader de sécurité (modifiable sur le terrain ou non) : politique de mise à jour, vérification d'image, anti-rollback, sélection de banque.
3. Application : acquittement de bon démarrage et supervision en exécution.

Ce modèle est cohérent avec PSA RoT et DICE/RIoT pour l'identité dérivée depuis le boot [39, 45, 22].

4.2 Partitionnement mémoire

Banque active (A) et banque passive (B). Topologie recommandée :

- région bootloader dédiée, protégée en écriture en production ;
- deux slots firmware de taille égale (A et B) si possible ;
- zone métadonnées persistantes distincte des slots, avec double copie.

Cette séparation réduit le risque de corruption croisée entre code exécutable et état de décision du boot.

Zone bootloader et métadonnées persistantes. Les métadonnées doivent inclure version, état de confirmation, compteur de tentatives, hash image, niveau de sécurité minimal requis et dernier motif d'échec. Elles doivent être écrites avec un protocole transactionnel simple pour conserver une lecture fiable en cas de coupure d'alimentation.

Stockage d'état de boot et drapeaux de validation. Éviter les drapeaux mono-bit non protégés. Préférer une structure redondante (double enregistrement + numéro de séquence + CRC), écrite selon un protocole en ajout seul pour limiter l'usure Flash. Ce choix simplifie la détection des états invalides et rend explicite la dernière transition de boot valide.

4.3 Cycle de vie d'une mise à jour A/B

éléchargement et stockage en banque inactive. L'image est reçue en blocs vérifiés (taille adaptée au secteur erase) avec vérification incrémentale du hash pour éviter une seconde lecture complète quand la RAM est contrainte. La chaîne de traitement doit aussi vérifier les bornes d'écriture et la cohérence globale avant de déclarer la banque candidate complète.

Vérification cryptographique pre-activation. Le vérifier valide au minimum :

- signature(s) du manifeste et de l'image ;
- identifiant produit/board et contraintes HW ;
- monotonicité de version ;
- politique de compatibilité des données persistantes.

La politique d'acceptation doit être identique entre backend et équipement pour éviter des divergences de décision en production.

Bascule atomique contrôlée. La bascule ne doit pas réécrire de gros volumes en fenêtre critique. On préfère une sélection par pointeur/flag de slot actif, idéalement dans une zone dédiée résiliente au power-loss. L'opération doit rester courte, déterministe et facilement testable en campagne de robustesse.

Confirmation post-boot et rollback. La nouvelle image est marquée *pending*. L'application doit émettre une confirmation après contrôles de santé (pilotes, connectivité, watchdog). Sans confirmation dans la fenêtre N demarrages, retour arrière automatique. Les criteres de confirmation doivent être stables dans le temps, afin de comparer les campagnes et d'identifier rapidement une dérive de qualité.

4.4 Machine d'états de démarrage sécurisée

Les états minimaux : *VALID A*, *TEST_B*, *CONFIRMED_B*, *ROLLBACK_A*, *RECOVERY*. Chaque transition est conditionnée par preuves locales (signature valide, flag pending, compteur essais, état watchdog).

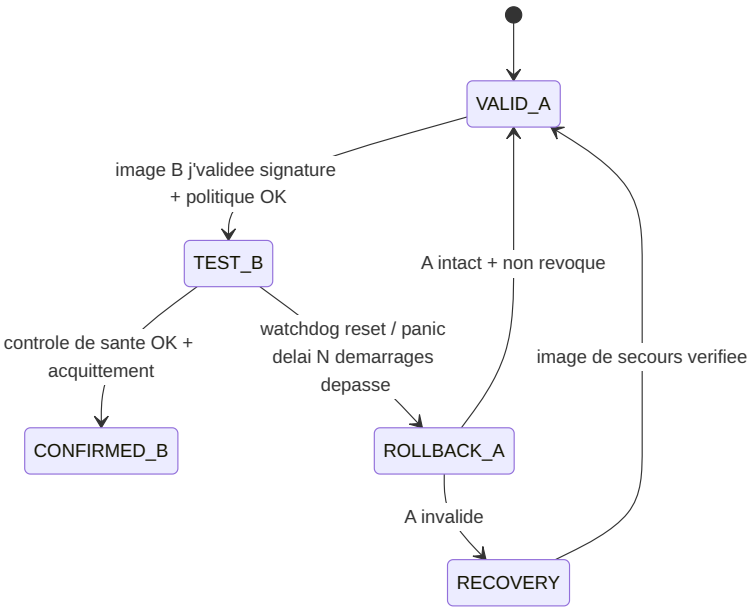


FIGURE 4.1 – Automate d'états de la mise à jour A/B

Modèle	**Principe**	**Avantages**	**Limites / préconditions**
A/B natif double banque	Deux banques NVM execute-in-place, sélection par drapeau de démarrage	Retour arrière rapide, peu de copie en exécution	Nécessite support NVM double banque et placement correct des vecteurs
A/B emulé (banque unique + zone de préparation externe)	Image candidate en NOR externe, copie ou XIP avant activation	Compatible MCU sans double banque interne	Surface d'attaque bus externe, coût nomenclature, latence
Permutation par secteurs (MCUboot)	Échange progressif active/inactive avec zone tampon	Fonctionne sur Flash unique	Complexité métadonnées, usure accrue, fenêtre de corruption
Écrasement seul + image de secours	Écrasement slot actif avec image de secours minimale	Empreinte mémoire réduite	Risque de briquage élevé sans alimentation stable

TABLE 4.1 – Comparatif des modèles architecturaux de mise à jour firmware

5

Contraintes spécifiques des MCU sans abstraction mémoire

5.1 Contraintes matérielles

Granularité Flash, alignement et atomicité d'écriture. Le couple *erase-size/program-size* conditionne directement la sûreté de l'update. Sur de nombreux MCU, l'effacement est page/secteur (de 1 à 128 KiB) alors que la programmation est par mots ou lignes. Ce décalage impose une stratégie transactionnelle stricte :

- ne jamais réécrire en place une structure critique (métadonnées, compteur, drapeau de boot) ;
- écrire un nouvel enregistrement complet puis valider via marqueur final ;
- valider cohérence et intégrité par CRC/ECC avant prise en compte.

Usure NVM et budget de cycles en exploitation. Les cellules Flash internes se situent typiquement entre 10k et 100k cycles ; certains profils industriels imposent des marges sévères en température/rétention. Les prérequis minimaux sont :

- journal *en ajout seul* pour états de boot et anti-rollback ;
- nivellement d'usure explicite (anneau de slots, rotation des pages, collecte bornée) ;
- budget d'endurance calculé sur la durée de vie produit (cycles/jour x années) ;
- tests d'endurance et de coupure d'alimentation corrélés [43, 36].

Prérequis matériels de sécurité souvent oubliés. Pour qu'un design A/B soit defendable, il faut en pratique :

- un ancrage immuable (ROM, OTP, eFuse) pour la clef racine ou son hash ;
- un mécanisme de verrouillage debug production avec procedure RMA distincte ;
- une source d'alimentation dimensionnée pour fenetres d'écriture critiques (ou supercap) ;
- un watchdog independant et configure des les premieres etapes de boot.

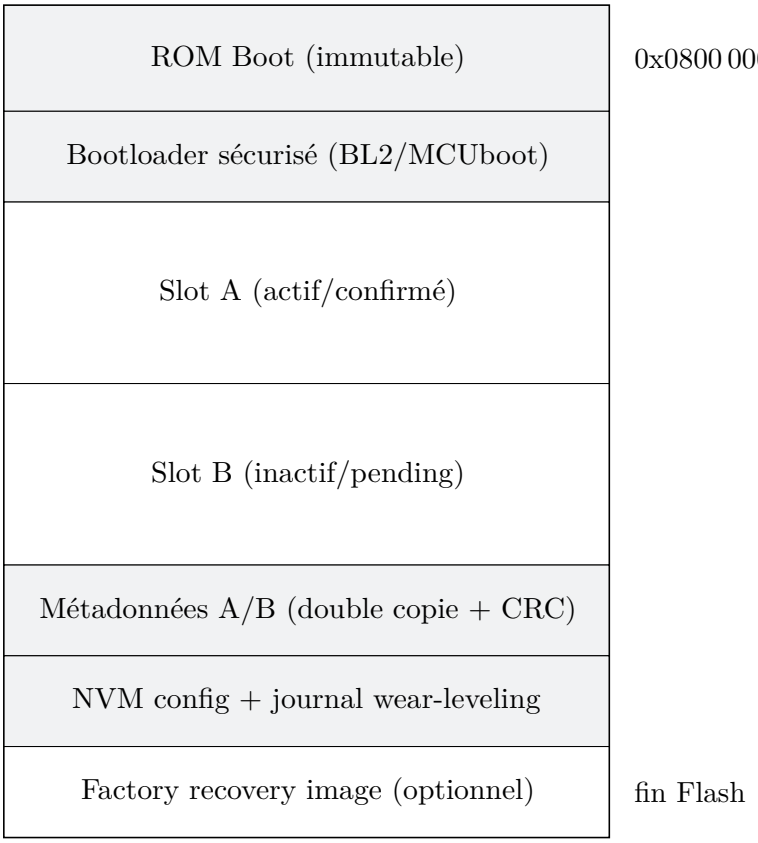


FIGURE 5.1 – Exemple de cartographie mémoire A/B sur MCU Flash interne

RAM limitée et impact sur la vérification cryptographique. La vérification d'une signature post-quantique peut depasser les budgets RAM des MCU d'entree de gamme. Les strategies viables sont :

- hash streaming + vérification signature sur manifeste compact ;
- vérification hors ligne de blocs et preuve Merkle résumée ;
- approche hybride classique + PQC réservée à certains profils produit.

Temps de démarrage et fenêtre de disponibilité. Un boot long augmente la surface DoS. Le vérifier doit être borné temporellement, avec watchdog, seuil de tentatives défini et chemin de sortie déterministe vers image connue valide. Il faut définir un *budget boot sécurité* mesurable (P95/P99) et des seuils d'alarme SOC.

5.2 Contraintes logicielles

Adressage fixe, vecteur de boot et PIC. Sans abstraction mémoire, l'application est souvent linkée pour une base précise. Pour comprendre le problème, il faut distinguer trois notions :

- **Code adresse absolue** : le compilateur/linker émettent des références vers des adresses fixes (ex. fonction à `0x08020000`, donnée à `0x20001000`).
- **PIC** (*Position-Independent Code*) : le code privilégie des calculs relatifs (PC-relatif), de sorte qu'il puisse être placé à plusieurs bases.

- **PIE** (*Position-Independent Executable*) : binaire complet relocalisable au chargement ; sur MCU, on emploie rarement un PIE complet, faute d'un chargeur riche comme sur OS généraliste.

Dans un schéma de **mise à jour incrémentale dynamique** (fonctions ajoutées, retirées ou remappées à chaud), le PIE devient nécessaire : la liste des fonctions n'est plus totalement figée à la production. L'exécution doit alors gérer un mappage dynamique en RAM, avec un chargeur capable de construire/mettre à jour une GoT cohérente au chargement pour résoudre les symboles et pointeurs globaux.

Concrètement, quand le compilateur génère des accès absolus, le firmware devient couplé à **une seule adresse de slot**. Si ce même binaire est copié dans l'autre banque, les sauts et accès données peuvent pointer vers la mauvaise zone, ce qui rend le double-banque instable ou inutilisable. C'est la raison principale d'incompatibilité entre firmware strictement non-PIC et architecture A/B.

Le rôle de la table GOT (*Global Offset Table*) est d'éviter de figer les adresses globales dans chaque instruction, en centralisant des offsets/pointeurs ajustables. Sur microcontrôleur, une GOT complète peut coûter en taille Flash, RAM et latence ; on retient donc souvent un compromis.

Pour supporter A/B, trois stratégies existent :

- deux compilations distinctes (A et B) avec adresses fixes ;
- compilation unique relocalisable partiellement (position-independent code/PIC) ;
- trampoline minimal + VTOR dynamique vers table vecteurs du slot choisi.

Le PIC complet sur Cortex-M reste coûteux en taille/performance. En pratique, un compromis *PI-lite* (sections critiques relocalisables, appels absolus limites) est souvent retenu.

Prérequis de conception associés :

- scripts d'édition de liens versionnés et testés en continu (A/B, secure/non-secure, debug/-prod) ;
- invariants explicites sur vector table, gestionnaires de faute et init bas niveau ;
- tests de non-régression sur migration de map mémoire entre versions.

Spécificités RV32 avec PMP. Sur RV32, la robustesse de l'update dépend fortement de la qualité du partitionnement PMP [18]. Les exigences minimales sont :

- région ROM/boot configurée en execute-only ou read-execute, verrouillée (`L`=1) ;
- région métadonnées A/B accessible en lecture seule hors M-mode ;
- interdiction en exécution de toute écriture sur les plages contenant clé publique, politique anti-rollback et code de vérification ;
- ré-initialisation explicite de `pmpcfg/pmpaddr` à chaque transition de privilège.

En complément, les prérequis d'audit doivent inclure :

- preuve de couverture des plages PMP (pas de trou exécutable non intentionnel) ;
- vérification des transitions `M`→`S/U` et des chemins d'exception `mtvec` ;
- test de résistance à l'abus d'`ecall`/interruptions sur frontière de privilège.

Dépendances aux drivers bas niveau. Les pilotes horloge, contrôleur Flash, MPU/SAU et verrouillage debug doivent être compatibles avec les deux banques et initialisés de façon déterministe, sinon le retour arrière peut échouer avant même l'initialisation série.

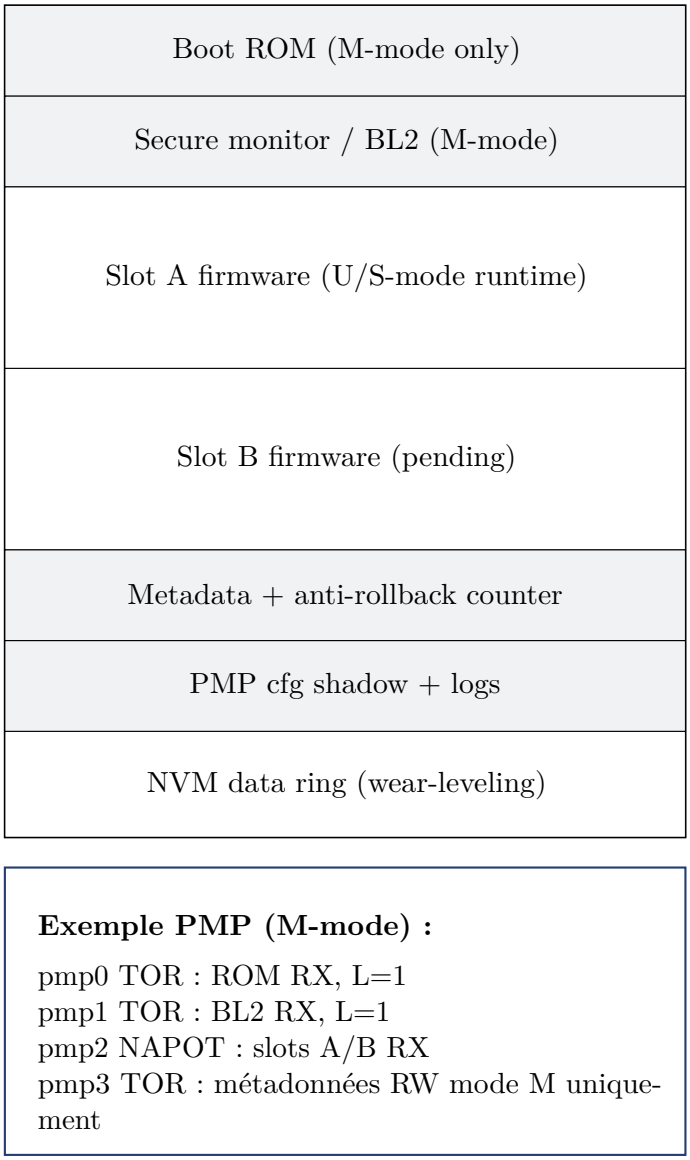


FIGURE 5.2 – Exemple de cartographie mémoire RV32 avec partitionnement PMP

Compatibilité inter-banques et migration de données. L'état applicatif (NVM de configuration) doit être versionné. Une migration irréversible doit être differee jusqu'a confirmation du nouveau firmware ; sinon retour arrière impossible.

Prérequis de migration :

- schéma versionné avec transformateurs `vN`→`vN+1` et `vN+1`→`vN` quand possible ;
- marqueur de migration atomique distinct du marqueur de confirmation de boot ;
- garde-fous de compatibilité croisée application/bootloader/manifeste.

5.3 Contraintes opérationnelles

Perte d'alimentation en cours d'update. Toute écriture critique doit être atomisée en transactions idempotentes. Interdire les modifications non journalisées de métadonnées.

Mises à jour partielles interdites ou strictement encadrées. Les deltas binaires économisent la bande passante mais augmentent les risques de désynchronisation de base image. Leur usage doit être réservé à des fleets strictement inventoriées.

Conditions minimales d'acceptation d'un delta :

- fingerprint explicite de l'image de base ;
- vérification post-application par hash complet de l'image reconstruite ;
- repli automatique vers image complète en cas d'écart.

Gestion des versions et anti-retour. Le mécanisme anti-rollback doit être conçu selon le hardware disponible :

- compteur OTP/eFuse monotone (idéal) ;
- compteur dans zone Flash protégée avec anti-tearing et vérification croisée ;
- politique serveur stricte si aucune racine matérielle (niveau de confiance inférieur).

Un prérequis fondamental est la définition d'une politique de versionnement cohérente :

- sémantique de version unique (bootloader, app, composants secondaires) ;
- fenêtres d'upgrade autorisées ;
- règles d'exception documentées (correctif critique, retour arrière d'urgence).

Contrainte HW	**Option disponible**	**Impact sécurité**	**Recommandation**
NVM double banque interne	Oui	Activation atomique simplifiée	Prioriser A/B natif, confirmation différée
NVM double banque interne	Non	Plus de copie/permutation, risque de coupure d'alimentation	Ajouter une zone de préparation externe ou un mode de reprise robuste
Stockage immuable de secrets	OTP/eFuse	Ancre fiable, anti-rollback fort	Stocker hash clef racine + compteur monotone
Stockage immuable de secrets	Aucun	Exposition forte au reflash physique	Compenser par boitier, verrouillage debug, attestation fréquente
Protection debug	Verrou irreversible (RDP2, lifecycle)	Reduit extraction de clefs et dump firmware	Activer en production avec procedure RMA dédiée

Table 5.1 – Contraintes matérielles structurantes pour la sécurité OTA

6

Exigences de sécurité sur le processus de mise à jour

6.1 Prérequis organisationnels

Les prérequis organisationnels doivent être explicites et auditables.

PR-ORG-01 – Séparation des rôles et circuit d'approbation. Appliquer le principe de double contrôle sur la signature de publication : compilation, approbation sécurité, signature, publication, chacun avec compte/service distinct et journal d'action.

PR-ORG-02 – Gouvernance des clefs et secrets. Conserver les clefs de signature en HSM/KMS, imposer des politiques d'usage (MFA, quorum, rotation, expiration), et exécuter des exercices de compromission de clef.

PR-ORG-03 – Evidence et auditabilite. Conserver preuves de compilation reproductible, empreintes d'artefacts, journaux de signature, SBOM et métadonnées de campagne [42, 31].

6.2 Prérequis techniques côté équipement

PR-DEV-01 – Root of Trust immuable. La clef racine de vérification doit être ancrée dans ROM, OTP ou eFuse. Toute mise à jour de cette ancre doit passer par un chemin exceptionnel, authentifié et journalisé.

PR-DEV-02 – Isolation des privilèges (Cortex-M vs RV32). Pour satisfaire OS-05 :

- sur Cortex-M : protection MPU stricte des zones boot + métadonnées, et politique d'exécution distincte entre gestionnaires critiques et application ;
- sur RV32 : PMP défini en M-mode avec verrouillage, et application exécutée en mode moindre privilège avec appels de service médiés.

Cadre de référence recommande pour les objectifs/contre-mesures MPU-PMP : [18].

PR-DEV-03 – Boot sécurisé et vérification d'image. Le boot vérifier doit être minimal, auditable, et résistant aux fautes simples : vérification redondante, contrôles de flot, et contrôles cohérents avant saut.

PR-DEV-04 – Protection anti-rollback. L'anti-retour doit combiner version manifeste et état local monotone. Sans état local, un attaquant peut rejouer un ancien binaire signé.

PR-DEV-05 – Rollback sûr et déterministe. Rollback uniquement vers une image précédemment confirmée et non révoquée.

PR-DEV-06 – Robustesse énergie et watchdog. Le système doit définir : points de reprise après coupure, timeout de confirmation, et stratégie watchdog en phase d'activation.

6.3 Prérequis techniques côté backend

PR-BE-01 – CI/CD de signature maîtrisée. Chaine d'intégration durcie : provenance, SBOM, scan de dépendances, signatures de compilation et artefacts de publication.

PR-BE-02 – Distribution authentifiée des artefacts. TLS mutuel pour canaux critiques, pinning des certificats backend, et signatures applicatives indépendantes du transport.

PR-BE-03 – Révocation et rotation de clefs operables. Prevoir révocation d'une clef de publication compromise sans immobiliser la flotte (liste de clefs autorisées versionnée dans manifeste et politique de transition).

6.4 Critères d'acceptation d'une mise à jour

Une mise à jour n'est promouvable que si :

- vérification cryptographique complète réussie ;
- compatibilité hardware déclarée et validée ;
- anti-rollback satisfait ;
- test de santé post-boot concluant ;
- télémétrie de confirmation reçue dans la fenêtre attendue.

7

Chaîne de confiance cryptographique de la mise à jour

7.1 Objectifs cryptographiques

Authenticité. Garantir que l'image provient d'une entité habilitée.

Intégrité. Garantir l'absence d'altération des payloads et métadonnées.

Fraîcheur et anti-rejeu. Garantir qu'un artefact ancien ne peut pas être réinstallé hors politique.

Non-répudiation. Conserver des preuves de signature auditables, sans exposer les clefs privées.

Referentiels crypto à appliquer. La politique cryptographique doit être alignée sur les référentiels suivants :

- **NIST** : FIPS 180-4, FIPS 186-5, SP 800-57 (gestion de cycle de vie des clefs), SP 800-193 (résilience firmware plateforme), FIPS 203/204/205 pour transition PQC [27, 32, 29, 28, 34, 33, 35] ;
- **ANSSI** : recommandations sur mécanismes cryptographiques et politiques de gestion des clefs/certificats en contexte national/réglementé [1, 3, 17] ;
- **Europe** : exigences de sécurité produit issues du Cybersecurity Act et du Cyber Résilience Act (CRA), à décliner dans les contrôles OTA et la gouvernance de vuln [12, 13].

7.2 Format d'image et manifeste signé

L'état de l'art embarque converge vers SUIT CBOR + COSE pour décrire et vérifier les mises à jour firmware [24, 25, 41].

Exigences minimales du manifeste. Le manifeste doit inclure au minimum :

- identifiants produit, revision PCB, SKU, contraintes hardware ;
- version, monotonicité, politique d'upgrade et fenêtre de validité ;
- empreintes de tous composants (app, monde sécurisé, pile radio, bootloader secondaire) ;
- contraintes de dépendance inter-images et niveau de sécurité minimal requis.

7.3 Algorithmes recommandés

Hash et KDF. SHA-256/384 reste la base pragmatique. Pour dérivées de clefs locales, HKDF est recommandé avec contexte explicite et séparation de domaine [27, 20].

Signatures classiques (pre-PQC). Ed25519 offre un bon compromis sur MCU ; ECDSA P-256 reste dominant pour compatibilité industrielle et certification [32, 7].

Chiffrement des payloads (optionnel). Le chiffrement OTA n'est pas un substitut à la signature. Il est pertinent pour protection IP/confidentialité en transit et au repos (AES-GCM par exemple), au prix d'une complexité supplémentaire de provisionning/rotation des clefs.

7.4 Gestion des clefs

Ancrage racine. Stocker le hash de la clef racine (ou certificat racine compact) en zone immutable.

élégation de signature firmware. Utiliser des clefs intermédiaires à durée de vie courte, avec politique de délégation explicite.

Rotation, expiration, révocation. Concevoir des manifests de transition autorisant simultanement ancienne et nouvelle clef sur une fenêtre limitée, puis retrait force de l'ancienne.

Prérequis de gouvernance clef/certificat.

- inventaire des clefs actives et de leur usage exact ;
- politique de rotation périodique et rotation d'urgence ;
- preuve d'exercice de révocation au moins annuelle ;
- processus de décommission sécurisé des clefs obsolètes.

7.5 Exemple concret : mode de mise à jour sécurisé du projet WooKey

Le projet WooKey (ANSSI) est un exemple intéressant car il combine un transport USB DFU standard avec une surcouche cryptographique et opérationnelle spécifique au produit [4]. La chaîne de mise à jour ne se limite pas à une vérification tardive de signature : la plateforme

maintient une authentification forte de l'opérateur via jeton externe, et implique activement le jeton DFU dans la dérivation de clefs de session pendant l'écriture des blocs firmware.

Dans cette architecture, le format de mise à jour inclut un en-tête signé (ECDSA), des métadonnées de version, un IV et une protection HMAC de l'en-tête ; le corps est chiffré par blocs (AES-CTR) pour réduire la malléabilité exploitable en cas de faute. Ce point est important sur MCU à RAM limitée, ou l'image est souvent écrite en Flash avant vérification finale de signature. WooKey ajoute également une défense en profondeur sur le mode DFU (isolation en tâches et séparation des rôles), ainsi qu'un mécanisme flip-flop double banque avec anti-rollback strict au boot [4].

8

Protocoles pour la mise à jour de microcontrôleurs

8.1 Objectif et frontières d'un protocole de mise à jour

Un protocole de mise à jour définit surtout *comment* transférer et piloter une image (état, blocs, reprise, statut), mais ne garantit pas automatiquement *qui* est autorisé ni *quoi* est légitime. En pratique, il faut superposer une politique cryptographique complète : authenticité du producteur, intégrité de l'image, anti-rejeu, anti-rollback, et journalisation exploitable.

8.2 USB DFU : standard de transport et d'orchestration

USB DFU (Device Firmware Upgrade) standardise une machine d'états et un jeu de requêtes de contrôle pour le téléversement et la gestion du firmware (`DFU_DNLOAD`, `DFU_UPLOAD`, `DFU_GETSTATUS`, `DFU_CLRSTATUS`, `DFU_ABORT`) [48]. Son intérêt industriel est la compatibilité avec des outils existants et des flux de maintenance atelier.

Limites natives de DFU. Le protocole DFU n'impose pas, à lui seul, la signature du firmware, la confidentialité du payload, ni une politique anti-rollback. Il ne définit pas non plus une gouvernance de clefs ou une traçabilité forensique complète. Sans surcouche sécurité, DFU reste donc un mécanisme de transport/commande exposé aux risques de rejeu, d'image non autorisée ou de compromission de poste hôte.

Exemple WooKey au-dessus de DFU. WooKey conserve l'interopérabilité DFU, mais ajoute une couche cryptographique applicative : authentification forte, dérivation de clefs de session via jeton, chiffrement par blocs, signature ECDSA, HMAC de l'en-tête et vérification de cohérence au boot. Cette approche illustre une bonne pratique générale : garder un protocole standard de transport, tout en déplaçant les garanties de sécurité dans un cadre cryptographique explicite [4, 48].

L'automate ci-dessous reprend les états DFU de la classe USB-IF (branche application et branche DFU) et les transitions opérationnelles principales utilisées par les outils de mise à jour (DETACH, DNLOAD, UPLOAD, GETSTATUS, ABORT, CLRSTATUS, reset) [48].

8.3 SCP03 (GlobalPlatform) : canal sécurisé pour commandes sensibles

SCP03 est un protocole de canal sécurisé défini par GlobalPlatform pour protéger des commandes APDU vers un composant sécurisé (SE/eSE/UICC) [16]. Il s'appuie sur des clefs statiques de domaine (typiquement ENC/MAC/DEK), un échange d'alea hote-carte, puis une dérivation de clefs de session. Selon le niveau de sécurité configuré, les commandes peuvent être protégées par MAC (C-MAC), chiffrées (C-ENC) et les réponses authentifiées (R-MAC).

Propriétés utiles pour l'update MCU. SCP03 apporte confidentiality, integrity et protection anti-rejeu sur le canal de commande. Il est particulièrement pertinent pour provisionner des secrets, piloter des états de sécurité, ou encapsuler des operations critiques (activation d'image, rotation de clef, ou déploiement de clefs de déchiffrement firmware).

Limites de périmètre. SCP03 n'est pas, à lui seul, un protocole de distribution de firmware de bout en bout : il ne remplace pas une spécification de manifeste, de dépendances logicielles, de politique anti-rollback produit, ni la logique A/B de récupération. Il doit être intégré dans une architecture plus large (secure boot, métadonnées de version, état transactionnel, télémétrie).

8.4 Comparatif DFU et SCP03

Option	**Point fort**	**Limite principale**	**Usage recommande**
USB DFU natif	Interopérabilité et outillage large	Pas de garanties crypto produit natives	Maintenance atelier, prototypage, base de transport
DFU + surcouche crypto (ex. WooKey)	Compatibilité + sécurité appliquée au firmware	Complexité d'intégration (format, clefs, token, états)	Produits exposés, exigences de résilience élevées
SCP03 (GlobalPlatform)	Canal commande fortement protégé (MAC/chiffrement/anti-rejeu)	Ne couvre pas seul la logique complète OTA/A-B	Provisioning sécurisé, commandes critiques, pilotage secrets

Table 8.1 – Positionnement de protocoles de mise à jour et canaux sécurisés

8.5 Recommandations d'intégration pour MCU

Pour une implémentation robuste, la combinaison suivante est recommandée :

- protocole de transport standard (DFU, HTTP(S), BLE OTA) pour l'opérabilité ;
- manifeste signé (SUIT/COSE) pour l'éligibilité cryptographique ;
- canal commande durci (SCP03 ou équivalent) pour opérations sensibles ;
- machine d'états A/B transactionnelle pour la récupérabilité ;
- télémétrie sécurité et traçabilité pour exploitation SOC et conformité.

Rappel d'adéquation protocole/interface. Le choix du protocole standard doit rester cohérent avec l'interface de communication effectivement disponible sur l'équipement. Par exemple, HTTP ou TFTP sont adaptés à des piles réseau de type IP/Ethernet (ou Wi-Fi), mais ne constituent pas le bon choix en l'état sur des bus série bas débit comme SPI ou I2C, qui requièrent en général un protocole de transport plus léger ou une passerelle adaptée.

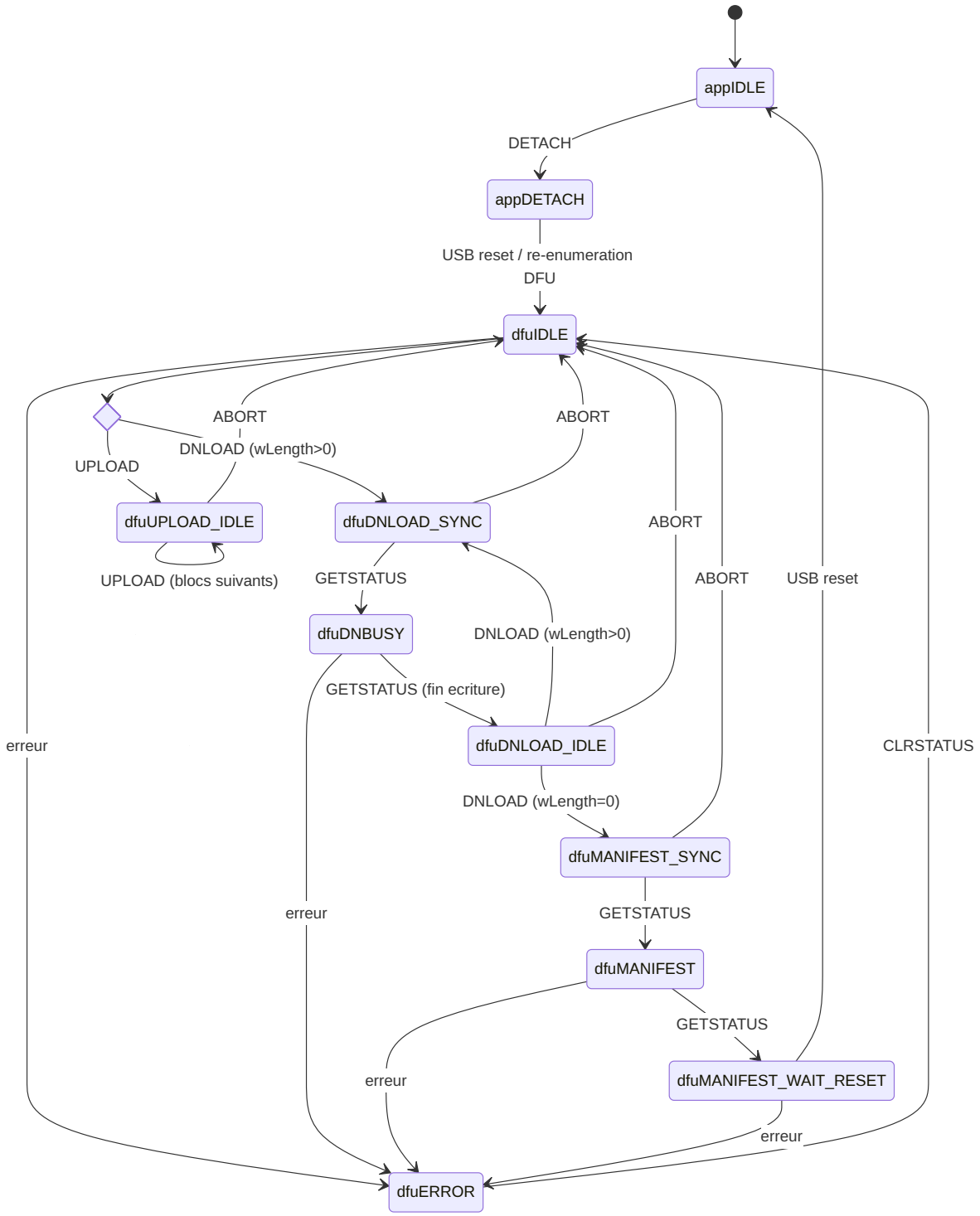


FIGURE 8.1 – Automate d'états du protocole USB DFU (classe DFU USB-IF)

9

PQC et stratégie de transition crypto-agile

9.1 Justification de l'anticipation PQC

Le risque *harvest now, decrypt/forge later* concerne surtout les actifs longue durée. Pour firmware critique en cycle de vie 10-20 ans, une feuille de route PQC est pertinente [30, 34, 33, 35].

9.2 Cas d'usage PQC dans la chaîne de mise à jour

Signatures de firmware post-quantiques. ML-DSA (Dilithium) est le candidat principal côté standardisation, mais ses tailles de clef et de signature peuvent pénaliser les MCU les plus contraints. Une stratégie réaliste consiste à réserver son usage initial aux lignes de produits disposant de marge mémoire suffisante.

Echanges de clefs post-quantiques (si applicable). Si le canal de provisioning impose un échange de secret, ML-KEM (Kyber) peut être introduit en priorité sur passerelles ou modules plus capacitaires. Sur MCU stricts, une architecture hybride déléguant une partie des opérations à un composant adjacent est souvent plus tenable industriellement.

9.3 Approche hybride classique + PQC

Double signature et politique de vérification. L'approche recommandée combine une signature classique obligatoire et une signature PQC expérimentale, puis inverse progressivement les priorités selon la maturité écosystème. Cette cohabitation doit être explicite dans le manifeste et dans les règles backend.

Impacts mémoire, performance et latence boot. Le coût principal est l'empreinte mémoire et le temps de vérification au boot. Il est souvent préférable de vérifier la double signature pendant la phase d'activation, puis de conserver un chemin de redemarrage nominal plus léger lorsque l'image est déjà confirmée.

Plan de migration progressive et retour arrière maîtrise. Définir des paliers (pilote, cohabitation, généralisation, decommission classique) et associer à chacun des critères de succès, des seuils d'alerte et un mécanisme de retour arrière testable en conditions proches du réel.

9.4 Exigences de crypto-agilité

Versionnement des suites cryptographiques. Le manifeste doit porter un identifiant de suite crypto versionné, stable et interprétable sur toute la durée de vie du produit.

Négociation/politique côté backend et équipement. Le backend ne doit jamais imposer une suite non supportée localement ; l'équipement expose un profil de capacité signé. Cette négociation doit être déterministe et rejetée en cas d'incohérence de profil.

Gestion de l'obsolescence algorithmique. Prévoir une date de fin de vie des algorithmes et des campagnes de migration forcée pour éviter les flottes en dette cryptographique durable.

10
Enrôlement des équipements et hiérarchie de CA

10.1 Modèle PKI industriel

CA racine hors-ligne. Conserver hors-ligne la racine de confiance et limiter son usage aux émissions exceptionnelles. Cette discipline réduit fortement l'impact d'une compromission opérationnelle sur l'ensemble de la PKI.

CA intermédiaires par usage (production, test, RMA). Séparer strictement les domaines pour éviter la contamination croisée. Les politiques de cycle de vie et les droits d'émission doivent être indépendants entre environnements.

Autorités d'enregistrement et contrôle d'identité. L'AR doit lier l'identité matérielle (serial, lot, SKU) à l'identité cryptographique, avec preuves de vérification conservées pour audit et investigation.

10.2 Schéma de référence : hiérarchie CA, enrôlement et dérivation de clef

Intérêt des sub-CA. Les sub-CA segmentent les risques et les responsabilités. Une compromission dans un domaine (par exemple RMA) ne doit pas permettre de signer des firmwares de production. La rotation et la révocation sont aussi plus ciblables, sans ré-émission massive de toute la flotte ni impact direct sur la racine hors-ligne.

Mode enrôlement (PKI) versus mode injection d'un secret. Le mode enrôlement attribue une identité cryptographique propre à chaque équipement (certificat, traçabilité, révocation granulaire). A l'inverse, l'injection d'un secret statique partagé simplifie parfois la fabrication, mais augmente le rayon d'impact en cas de fuite : une seule compromission peut affecter un lot complet, voire une famille de produits.

Usage de la dérivation de clef. La dérivation (HKDF ou équivalent) permet de produire des clefs distinctes par usage (attestation, session update, canal de commande) depuis un secret racine matériel et un contexte (identifiant équipement, version, compteur/nonce). Cela limite la

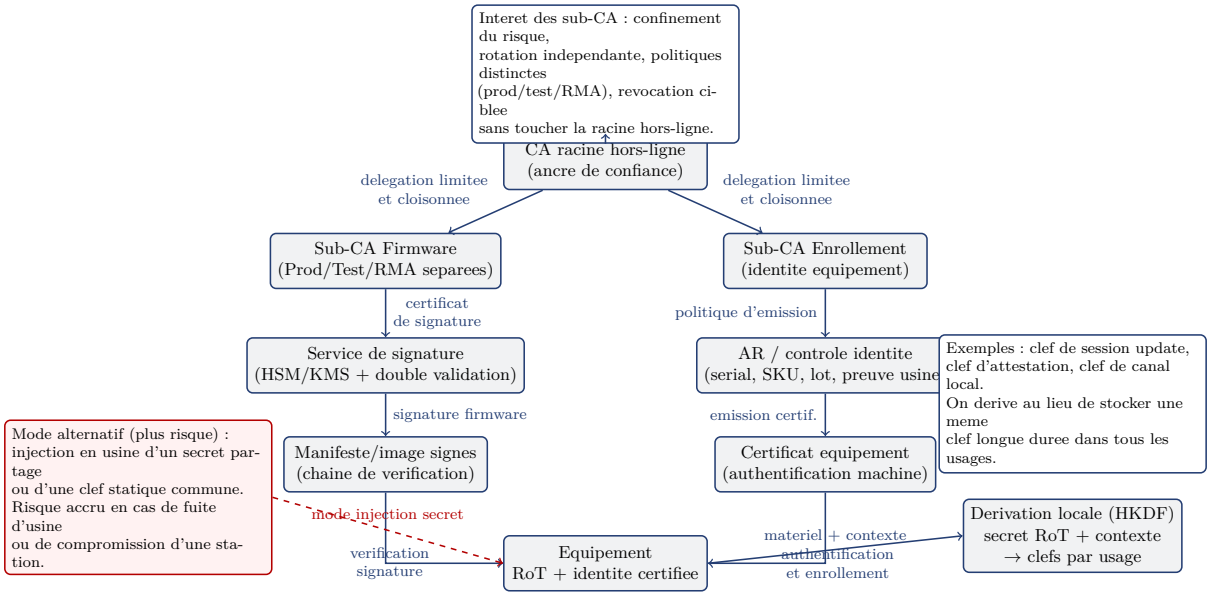


FIGURE 10.1 – Hiérarchie CA type pour signature firmware et enrôlement des équipements

réutilisation de clefs longues durées, réduit les mouvements latéraux et facilite la rotation logique sans reprovisioning complet.

10.3 Processus d'enrôlement initial

Injection d'identité en fabrication. Réaliser l'injection en environnement sécurisé, avec traces d'opération scellées et preuves de personnalisation. La chaîne d'approvisionnement doit pouvoir démontrer qui a injecté quoi, quand et sous quelle autorisation.

Provisioning de certificats et ancres de confiance. Préférer des certificats courts et renouvelables, avec un profil embarqué compact adapté aux contraintes mémoire du MCU.

Personnalisation sécurisée par lot ou par unité. Le mode unitaire augmente la traçabilité mais a un coût industriel supérieur ; le mode par lot demande des contrôles compensatoires plus stricts pour conserver le même niveau de confiance.

10.4 Cycle de vie des certificats

Renouvellement et rotation. Le renouvellement doit être anticipé avant expiration pour éviter les flottes en dette cryptographique. La rotation d'urgence doit être testée régulièrement avec des scénarios de dégradation réalistes.

Révocation (CRL/OCSP ou mécanisme embarqué équivalent). Sur MCU hors-ligne intermittent, la révocation peut être diffusée via une politique incluse dans les manifests de campagne. L'objectif est d'atteindre une couverture de révocation mesurable, même avec une connectivité intermittente.

Gestion des équipements compromis ou décommissionnés. Basculer en mode quarantaine, couper l'accès aux services sensibles et marquer l'identité comme retirée afin d'empêcher toute réutilisation non autorisée.

11
Attestation et preuves de conformité logicielle

11.1 Objectifs de l'attestation

Preuve d'intégrité du firmware actif. L'attestation doit démontrer qu'une empreinte mesurée correspond à un firmware approuvé et à une configuration connue. Cette preuve devient exploitable seulement si elle est comparée à une référence maîtrisée côté service de confiance.

Preuve de version et d'état de sécurité. Le rapport doit inclure au minimum le niveau de patch, le statut debug et la politique secure boot active, afin de supporter une décision d'accès explicite et reproductible.

11.2 Attestation locale et distante

Mesures au boot (secure measurements). Mesurer bootloader, application et configuration de sécurité dans une chaîne de confiance, avec un format stable dans le temps pour faciliter l'automatisation de la vérification.

Rapport d'attestation signé. Signer le rapport via une clef d'attestation dérivée du RoT (DICE/RIoT) et associer un nonce de fraîcheur [45, 22, 8]. La structure du rapport doit rester sobre pour être compatible avec des terminaux contraints.

Vérification côté service de confiance. Le service valide la chaîne de certificats, la fraîcheur du nonce et la politique d'acceptation avant d'autoriser l'accès à des ressources sensibles.

11.3 Lien entre attestation et décision opérationnelle

Autorisation d'accès réseau conditionnelle. L'accès aux API critiques doit être conditionné à une attestation conforme et récente, selon une politique de risque explicite.

Déclenchement de remédiation ou de quarantaine. En cas de non-conformité, appliquer une remédiation graduelle (réduction de privilèges, mise à jour forcée, isolement) afin de limiter

l'impact opérationnel sans perdre le contrôle sécurité.

12

Exploitation sécurisée de l'A/B en conditions réelles

12.1 Stratégies de déploiement

Canary, vagues progressives, fenêtre de maintenance. Les campagnes doivent commencer sur un échantillon représentatif puis progresser par vagues avec seuils d'arrêt automatique. Ce mode de déploiement limite l'ampleur d'un incident et accélère sa détection.

Critères de promotion d'une version. Promouvoir une version uniquement si le taux de boot confirmé est stable, sans rollback anormal et sans incident sécurité sur la fenêtre d'observation définie.

Gestion de flotte hétérogène. Segmenter par SKU, révision carte, région radio et capacités crypto pour éviter des décisions globales inadaptées à certains segments techniques.

12.2 Supervision et télémétrie de sécurité

Indicateurs clefs (succès boot, rollback, échecs vérification). Collecter au minimum :

- taux de vérification cryptographique en échec ;
- taux de rollback automatique ;
- latence moyenne activation→confirmation ;
- distribution des raisons d'échec.

Ces indicateurs doivent être historisés et comparés par segment de flotte afin de détecter rapidement une dérive locale.

Détection d'anomalies et alerting SOC. Des patterns de rejet signature massifs peuvent indiquer une tentative de campagne malveillante. Les règles d'alerte doivent distinguer les défauts de qualité des signaux d'attaque pour réduire le bruit opérationnel.

12.3 Réponse à incident liée à la mise à jour

Blocage de campagne et révocation d'urgence. Prévoir un kill-switch de campagne et une liste de révocation diffusée rapidement. Les conditions de déclenchement et de levée du blocage doivent être documentées à l'avance.

Forensic et collecte de preuves. Conserver manifests, journaux de boot, identifiant campagne et preuves de signature pour investigation post-incident. La valeur probante dépend de la qualité de l'horodatage et de l'intégrité des traces.

13
Vérification, validation et conformité

13.1 Exigences de test A/B

Tests de robustesse (coupure d'alimentation, corruption Flash). Injecter des coupures à chaque étape critique et vérifier l'absence de bricking. Les résultats doivent être reproductibles sur plusieurs lots matériels.

Tests de sécurité (signature invalide, rollback force, rejeu). Executer une batterie negative systematique incluant fault injection contrôlée, afin de valider le comportement fail-closed dans les scenarios adverses.

Tests de performance (temps de boot, temps d'update). Définir des budgets maximaux et valider sous conditions thermiques et tension dégradées, pour garantir une marge opérationnelle suffisante en environnement industriel.

13.2 Critères de sécurité avant mise en production

La mise en production requiert :

- audit code bootloader ;
- preuve de tests de non-régression sécurité ;
- validation des procédures de révocation/rotation ;
- simulation de compromission de clef de publication.

13.3 Traçabilité normative et exigences réglementaires

Une correspondance explicite doit être maintenue avec ETSI EN 303 645, NISTIR 8259A, IEC 62443 et recommandations ANSSI pour embarque/IoT [11, 14, 19, 2].

14

Recommandations d'implémentation et anti-patterns

14.1 Bonnes pratiques de conception

- Bootloader minimal, revue de code, dépendances réduites.
- étadonnées transactionnelles avec CRC/ECC et séquence numbers.
- Séparation stricte des clefs (développement, préprod, production).
- Politique anti-rollback ancrée matériellement.
- Télémétrie sécurisée orientée détection précoce.

14.2 Erreurs fréquentes à éviter

Confiance implicite dans le transport réseau. TLS sans signature applicative expose a des compromis d'infrastructure et ne suffit pas a garantir la legitimite d'une image firmware.

Absence de politique de révocation exploitable. Sans révocation opérationnelle, la compromission d'une clef devient rapidement catastrophique, car la flotte ne peut pas être assainie dans un delai acceptable.

Rollback non maîtrise. Autoriser un downgrade libre annule les efforts de correction de vulnérabilités et reintroduit des versions connues comme faibles.

Couplage excessif entre bootloader et application. Un couplage fort complique les migrations de format manifeste, ralentit les correctifs et augmente le risque de régression lors des évolutions de plateforme.

14.3 Checklist opérationnelle de sécurité

Contrôle	**Statut cible**	**Preuve attendue**
Ancre de confiance immutable (OTP/eFuse/ROM)	Obligatoire	Dossier hardware + lecture fuse en production
Vérification signature + anti-rollback au boot	Obligatoire	Rapport de tests négatifs et logs de refus
Mécanisme rollback automatique power-fail safe	Obligatoire	Campagne de tests coupure alimentation
Révocation et rotation de clefs opérables	Obligatoire	Procédure exercée + compte-rendu d'exercice
Télémétrie sécurité update centralisée	Fortement recommandée	Tableau de bord SOC + rétention des événements
Préparation migration PQC (mode hybride)	Recommandée	Étude d'impact RAM/Flash/latence + roadmap

Table 14.1 – Checklist minimale de sécurisation de la mise à jour MCU

15

Synthèse des attaques modernes sur la mise à jour MCU

15.1 État de l'art des attaques

Les familles d'attaques les plus pertinentes aujourd'hui sont :

- **Chaine d'approvisionnement** : compromission d'artefacts ou de chaîne d'intégration ;
- **Downgrade/replay** : rejeu d'images légitimes mais vulnérables ;
- **Fault injection** : glitch tension/horloge, EMFI pour contourner contrôles ;
- **Abus de debug** : reactivation SWD/JTAG, dump firmware, patch en exécution ;
- **TOCTOU** : vérification sur objet différent de l'objet exécuté ;
- **Desynchronisation métadonnées** : corruption des flags de confirmation.

Mapping attaque-vers-objectif OS. Pour guider l'ingénierie de vérification :

- injection firmware malveillant → OS-01, OS-02, OS-08 ;
- downgrade/replay → OS-03, OS-07 ;
- glitch/fault injection → OS-05, OS-06 ;
- corruption métadonnées/power-loss → OS-04, OS-07 ;
- compromission clef de publication → OS-01, OS-08.

15.2 Matrice attaques vs contrôles

Attaque	Vecteur	Impact	Contrôles prioritaires
Injection firmware malveillant	Chaine d'intégration/dépôt compromise	Exécution de code arbitraire	Signature hors ligne, séparation des rôles, provenance de compilation, révocation rapide
Downgrade d'image signée	Rejeu local/distant	Réouverture CVE connues	Compteur monotone OTP/eFuse, politique version minimale
Bypass check par glitch	Accès physique, fault injection	Secure boot contourne	Redondance contrôles, détecteurs de faute, boîtier anti-tamper, lockdown debug
Corruption métadonnées boot	Coupure ou attaque écriture Flash	Boot indéfini/DoS	Journalisation en ajout seul, double copie + CRC, machine d'états stricte
Compromission clef de publication	Fuite secrets SI	Prise de contrôle flotte	HSM/KMS, rotation urgente, magasin de confiance versionné, attestations

TABLE 15.1 – Matrice synthèse des attaques sur OTA MCU et contre-mesures

16
Conclusion

16.1 Synthèse des garanties apportées par le double banque

L'architecture A/B apporte une résilience opérationnelle forte face aux échecs d'update et une base saine pour appliquer les contrôles cryptographiques modernes.

16.2 Conditions de maintien de la sécurité dans le temps

La sécurité reste conditionnée à la gouvernance des clefs, à la discipline de publication, et à la capacité à réactualiser les politiques anti-rollback et de révocation.

16.3 Feuille de route de durcissement (incluant PQC)

Priorités recommandées :

1. Ancrage matériel fort (OTP/eFuse) + rollback robustes.
2. Standardisation manifeste SUIT/COSE + télémétrie sécurité exploitable SOC.
3. Introduction hybride classique/PQC sur familles de produits capacitaires.
4. Généralisation attestation distante pour pilotage risque flotte.

A

Annexe A – Exemple de machine d'états de boot

États minimaux suggérés :

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

Transitions critiques :

- pending → confirmed uniquement après preuve de santé applicative ;
- pending → rollback sur délai dépassé, réarmement watchdog, ou assertion sécurité ;
- rollback → recovery si banque précédente invalide.

B

Annexe B – Exigences minimales de manifeste signé

Le manifeste doit contenir au minimum :

- identifiant produit et révision hardware ;
- version firmware et politique anti-rollback ;
- empreinte cryptographique image ;
- clef(s) de vérification autorisées et période de validité ;
- dépendances inter-composants et préconditions d'installation.

C

Annexe C – Politique type de gestion de clefs et certificats

Politique type :

- racine offline, intermédiaires online segregés ;
- rotation périodique semestrielle ou annuelle selon criticité ;
- exercice de révocation au moins trimestriel ;
- journalisation immuable des émissions et retraits.

D

Annexe D – Matrice de menaces et contrôles associés

La matrice complète doit relier chaque menace a :

- contrôles de prévention ;
- contrôles de détection ;
- contrôles de remédiation ;
- preuve de test associée et propriétaire de contrôle.

Bibliographie

[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.

Licence