ChatGPT

Automotive Software Update Security Mechanisms

Security Controls in SUMS

Standards like ISO 24089:2023 and UNECE Regulation R156 require OEMs to establish a **Software Update Management System (SUMS)** with formal processes and controls. For example, ISO 24089 and R156 mandate “sound processes for secure software distribution and secure software updates” with defined roles and risk management [1]. R156 explicitly requires manufacturers to protect update processes against tampering [2] and to enforce authenticity/integrity of updates [3]. Key SUMS security controls include:

- **Access Control:** Update creation and deployment infrastructure (cloud servers, build systems, etc.) must be tightly controlled. Only authenticated backend agents and personnel may initiate or sign updates. Strong authentication (e.g. multi-factor) and role separation are used to ensure that only authorized individuals or systems can produce or approve update packages [4] [5].
- **Update Authorization and Code Signing:** All update binaries must be cryptographically signed by the OEM or supplier before release. The SUMS workflow typically requires multi-stage approvals (engineering, safety, security) and code-signing with private keys. For compliance R156, vehicles must reject any update that fails signature verification, ensuring only valid, authorized updates are applied [3] [6].
- **Tracking and Audit:** R156 imposes extensive recordkeeping. OEMs must log every software version and its integrity data (hash/signature) for each ECU (RXSWIN) both before and after updates [7] [8]. Each update is documented (purpose, affected functions, pre/post-configurations, etc.) and linked to specific vehicle IDs [7] [9]. This tamper-proof log (often maintained in the cloud and vehicle) enables traceability and compliance audits. NHTSA guidelines similarly advise “tamper-proof logging of all important events” in the update process [10].
- **Key Management:** A robust PKI/KMS scheme underpins SUMS. OEMs use Hardware Security Modules (HSMs) or TPMs to generate and protect private keys, and a centralized Key Management System issues certificates and symmetric keys to suppliers and vehicles [11] [12]. For example, vendors note that an HSM-rooted trust anchor ensures each ECU only installs firmware signed by the OEM [12] [13]. Key rotation and certificate revocation lists are also managed to address key compromise.

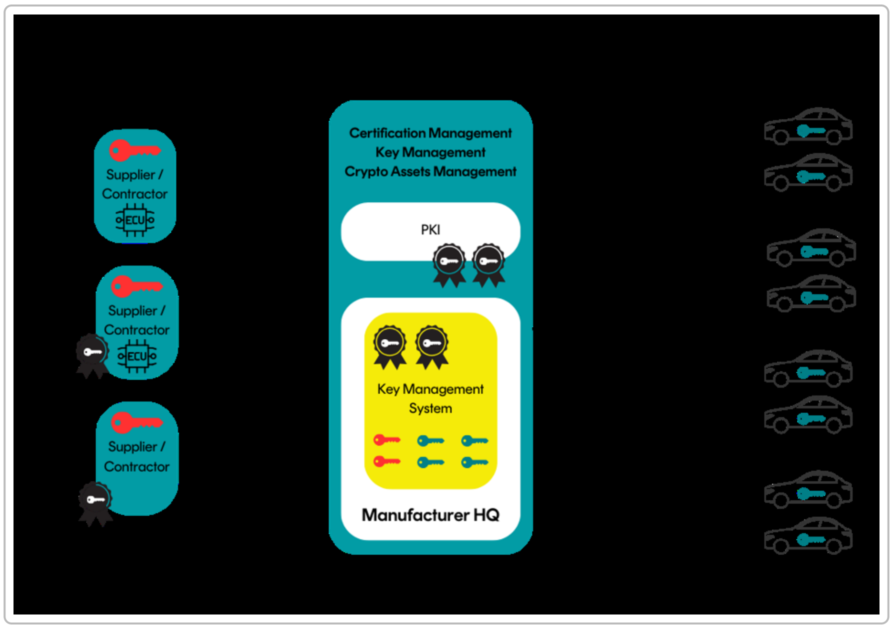

Figure: Centralized PKI and Key Management. Supplier key material and ECU identities are managed by a manufacturer's PKI/KMS (middle), which issues certificates and keys to each vehicle (right) [11] [12].

Secure Update Protocols

Automotive updates are delivered either **wirelessly (OTA)** or via **dealer/service tools (wired)**, using secure transport protocols and embedded update frameworks:

- **Encrypted Transport (TLS/IP):** Updates sent over cellular/Wi-Fi links use standard secure channels. For example, NXP describes that the telematics unit establishes a TLS connection to the OEM server to download update files [4]. All network transactions should be TLS-encrypted with mutual authentication (hostname verification and CA-signed certificates) to thwart man-in-the-middle attacks [14]. Similarly, adaptive AUTOSAR environments use HTTPS or secured TCP/UDP stacks for OTA download.
- **Uptane Framework:** Many OEMs supplement TLS with the Uptane protocol, a vehicle-specific secure update framework. Uptane uses signed metadata with separate roles (Root, Targets, Snapshot, Timestamp) so that even if a repository is compromised, forged images can be detected [15]. Its design mandates redundancy and multiple signers, reducing single points of failure. Uptane's formal "compromise resilience" architecture is increasingly cited in guidance for secure automotive updates [15] [16].
- **Adaptive AUTOSAR OTA:** The AUTOSAR Adaptive standard includes services for updating software components. It supports differential (delta) updates and orchestrates component reloads via an "update master" and diagnostic manager [17]. Elektrobit notes that Adaptive AUTOSAR specifies secure OTA functions (e.g. Update Configuration Manager, diagnostic interfaces) so that only authenticated update binaries are installed [17]. Standardized client-server interfaces (often over DoIP or HTTPS) enable the ECU gateway to receive and redistribute update payloads.
- **In-Vehicle Networks:** Internal bus communications employ secure protocols. For example, CAN traffic can be protected by Secure Onboard Communication (SecOC) which adds message authentication codes (MACs) and timestamps to each CAN frame [18]. Ethernet or LIN networks

similarly use HSM-based encryption/authentication at endpoints. OEMs typically route updates through a central gateway ECU (with the highest trust), which checks versions and signatures before flashing subordinate ECUs [19] [20].

- **Service-Mode Updates:** When vehicles are serviced, technicians may use USB or OBD-II tools. These use the UDS diagnostic protocol with security (e.g. ISO 14229's seed/key or token-based security access). The session is encrypted (often via TLS or proprietary link-layer crypto) and requires authentication by challenge-response, ensuring only authorized tools can write firmware to ECUs (often enforced by the gateway or HSM).

Digital Signatures and Certification

All update images are authenticated by digital signatures and managed via an automotive PKI:

- **Code Signing:** Each update package is signed (typically with ECDSA or RSA) by the OEM or supplier's private key. According to industry guidance, "digital certificates authenticate software updates, ensuring they come from a legitimate source and have not been tampered with" [21]. The vehicle's bootloader or update agent uses the corresponding public key (often stored in hardware) to verify this signature before proceeding.
- **PKI Certificate Chains:** The signing keys are certified by a chain of trust. Typically, an OEM or supplier root CA (protected offline in an HSM) issues intermediate CAs and end-entity certificates for code signing. GlobalSign notes that a robust PKI "enables secure and authenticated software updates for vehicles" [22]. Vehicles keep a trust anchor (root certificate) in secure storage (TPM/HSM or one-time fuses) and use it to validate the full chain. Some OEMs use specialized automotive PKIs (e.g. AutoCrypt PKI) tailored to vehicle identities.
- **ECU/Gateway Verification:** During secure boot or update, each ECU/gateway verifies that any new firmware or software bundle is signed by the appropriate certificate. For example, NXP emphasizes that the gateway runs a "secure boot process… so that only firmware signed by the OEM can be executed" [13]. If signature or certificate validation fails, the update is rejected. UNECE R155/156 and ISO/SAE 21434 effectively mandate this: ECUs "should reject code that isn't properly signed or validated," just as current U.S. and EU regulations require [6] [13].
- **Certificate Management:** The SUMS must also manage certificate lifecycles – issuing and rotating signing certificates, handling expirations, and maintaining revocation lists. For example, if a signing key were compromised, its certificate would be revoked and replaced in the trust store. In practice, vehicle update systems integrate with cloud PKI services or on-premise CAs to automate key provisioning and revocation across fleets.

Integrity Verification and Anti-Rollback

Ensuring update integrity and preventing downgrade attacks involve multiple layers:

- **Cryptographic Hash Checks:** Each update file includes a cryptographic hash (e.g. SHA-256) recorded in the signed metadata. During installation, the ECU recomputes the hash and compares it to the signed value to verify integrity. Best practices call for *both* pre- and post-flash hash checks. As one industry guide notes: "use cryptographic hash functions… to verify the integrity of the update files before and after flashing" [23]. Any mismatch or tampering causes the update to abort.
- **Secure Boot / Bootloader Validation:** ECUs use secure (verifying) bootloaders. On power-up, each boot stage authenticates the next stage's code using stored keys. Elektrobit explains that adaptive

AUTOSAR “protects the entire update process… thanks to a secure boot mechanism” that loads only authenticated components [24]. In other words, even a successfully transferred image will not run unless its signature (or hash in firmware metadata) matches an approved key.

- **Anti-Rollback Mechanisms:** To prevent attackers from reinstalling an older, vulnerable version, OEMs implement anti-rollback checks. Typically this means storing a monotonic version counter or using a hardware-protected “lifecycle” field. NXP’s whitepaper advises securely logging the current firmware version and checking it against any incoming update [19] [25]. If a received update has a lower version number than what is stored, it is rejected. Cryptographic measures (e.g. chaining each new firmware image with the previous version) and secure elements are used to protect this version state. As WolfSSL’s wolfBoot notes, strict version enforcement is critical: it “will refuse to load unauthorized or downgraded firmware” [26].
- **Fail-Safe Recovery:** Additionally, vehicles often use dual-bank (“A/B”) flashing or backup images. If a new update fails integrity checks or crashes, the bootloader can revert to the known-good copy. While not strictly a cryptographic measure, this supports recovery when malicious or faulty updates are detected. All the above cryptographic checks (hashes, signatures, secure boot, versioning) work in concert to make unauthorized or outdated code installation infeasible.

Compliance Note: These practices align with regulatory and standards requirements. UNECE R156 explicitly demands update integrity protection [3] and R155/ISO 21434 mandate that ECUs only run authenticated code [6]. NHTSA’s Cybersecurity Best Practices similarly urge preserving OTA update integrity and designing for compromised-server scenarios [27]. In summary, modern automotive software update security combines a PKI-backed signing infrastructure, end-to-end encrypted channels (TLS or secure frameworks like Uptane), rigorous version/boot integrity checks, and comprehensive logging/audit as prescribed by ISO/UNECE standards [1] [3].

Sources: Industry whitepapers and standards such as ISO 24089, UNECE R156, NHTSA guidelines, and publications by OEMs/suppliers. [1] [2] [8] [15] [4] [28] [21] [13] [25]

---

[1] [5] [11] Key Management in the Automotive Domain
https://www.cryptomathic.com/blog/key-management-in-automotive

[2] [3] [7] [8] [9] unece.org
https://unece.org/sites/default/files/2024-03/R156e%20%282%29.pdf

[4] [13] [19] [25] nxp.com
https://www.nxp.com/docs/en/white-paper/Making-Full-Vehicle-OTA-Updates-Reality-WP.pdf

[6] [26] Meeting Secure Boot Compliance Requirements – wolfSSL
https://www.wolfssl.com/meeting-secure-boot-compliance-requirements/

[10] [12] [14] [27] What is OTA in automotive? Over the air updates explained. - Rambus
https://www.rambus.com/blogs/ota-updates-explained/

[15] [16] Uptane Standard 2.1.0 | Uptane
https://uptane.org/docs/2.1.0/standard/uptane-standard

[17] [18] [20] [24] [28] OTA updates with Adaptive AUTOSAR environment – Elektrobit
https://www.elektrobit.com/blog/ota-updates-with-adaptive-autosar/

[21] [22] Securing the Automotive Industry with Digital Certificates
https://www.globalsign.com/en/blog/Securing-connected-cars-with-digital-certificates

[23] Secure Software Updates on Automotive Embedded Systems: Secure Boot, Secure Flash, and OTA | by Muhammet Kalaycı | Medium
https://medium.com/@mkklyci/secure-software-updates-on-automotive-embedded-systems-secure-boot-secure-update-and-ota-68d856c7933f