Cyber-Security Training Program

BGSW- SJCE
May 2023

Secure Flashing

Rejin Sathianesan

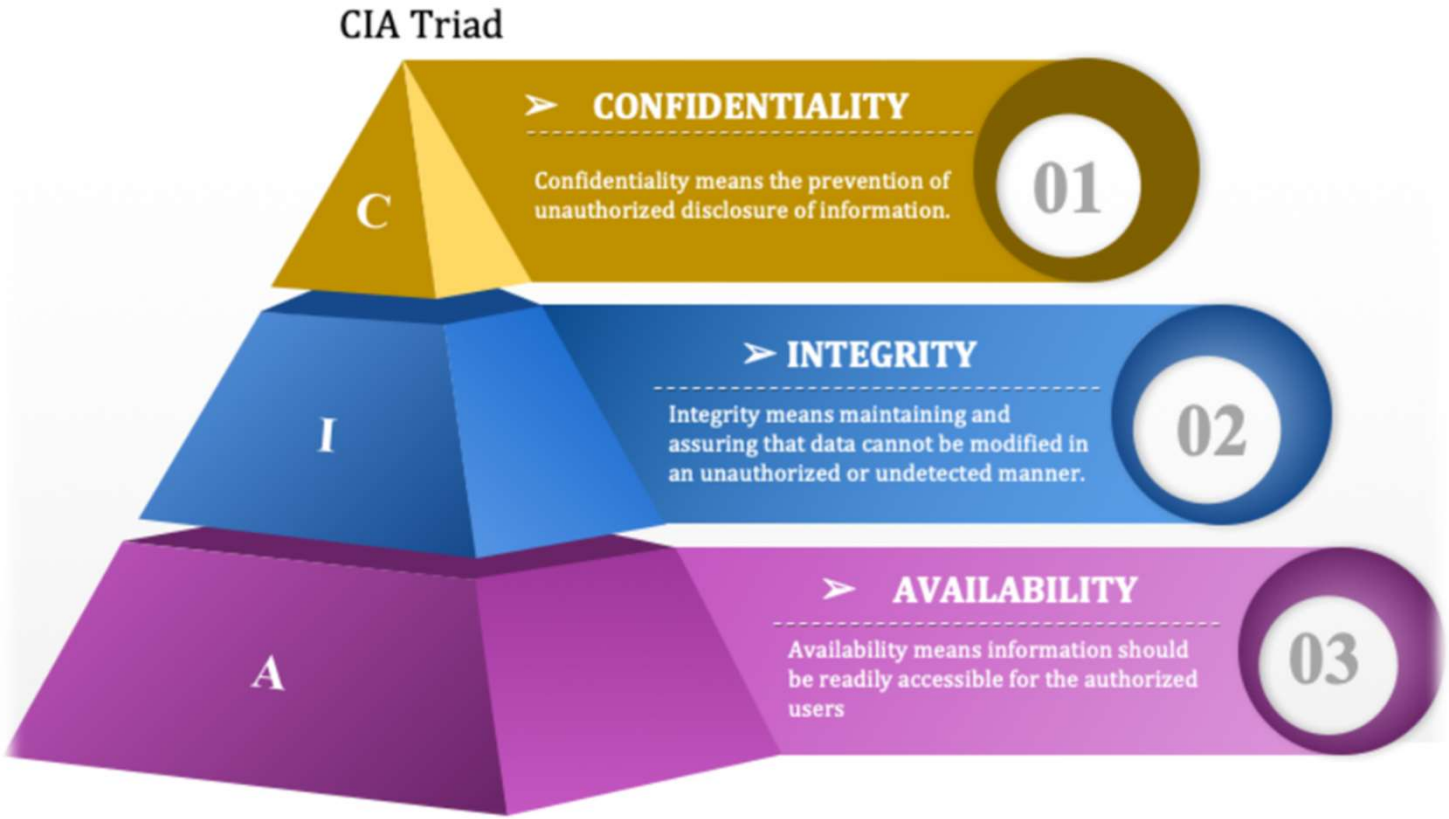
Bosch
Global
Software
Technologies
alt_future

Agenda

- Re-cap CIA Triad
- Why Secure Flashing ?
- What’s required ?
- Goal of Secure flashing
- Automotive use case

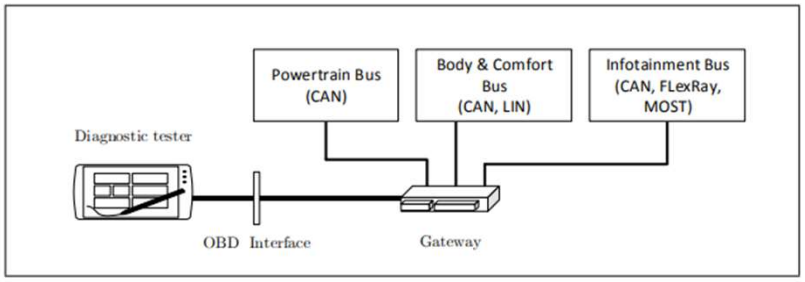
Security Features overview

CIA - RECAP



Secure Flashing

Automotive use case



Secure Flashing is used to prevent the flashing of unauthentic or manipulated firmware images over the available update interfaces. To this end, a digital signature (reference value) of the firmware image is created at the backend.

Example: UDS Flash Programming:

	Start			
1	ReadDataByIdentifier(22)		9	Routine Ctrl-Erase Memory.(31 01 FF 00)
2	DiagnosticSessionControl-extSession(10 03)		10	Request Download (34)
3	Routine Ctrl-Check Prog. Precond.(31 01 02 03)		11	Transfer Data (36)
4	Ctrl DTC Setting – DTC Setting = Off (85 02)		12	Request Transfer Exit (37)
5	Comm. Ctrl –Disable Non.Diag Comm (28 01 01)		13	Routine Ctrl-Check Memory.(31 01 02 02) Routine Ctrl-Check Reprog. Dependencies
6	DiagnosticSessionControl-ProgSession(10 02)		14	ECU Reset (11 01)
7	Security Access Request Seed (27 11) –Transfer Key(27 12)		15	Comm. Ctrl–Enable Non.Diag.Comm. (28 00 00)
8	Write data By Identifier – Write FingerPrint (2E)		16	Ctrl DTC Setting – DTC Setting = On (85 01)
			17	DiagnosticSessionCtrl-DefaultSession(10 01)
				End

ISO – 14229 UDS specification

https://piembsystech.com/security-access-service-identifier-0x27-uds-protocol/

Security Features overview

Why Secure Flashing ?

Security Needs

- An Attacker should not be able to manipulate the firmware.
- Prevent an attacker to analyze the Bosch Software and flash the un-trusted software or prevent third parties from flashing unauthorized software/firmware updates into the ECU.

Protect Assets : Firmware Integrity

- Microcontrollers (RH 850, ST Chorus) – Master/Primary, Microprocessors- Slave/Secondary
- Flash (In case if we use external flash memory)

Security Features overview

What's required ?

- An ECU with initially downloaded complete SW + bootblock: the bootblock “knows” the public key for this SW
- A “new, to be downloaded” SW, doesn’t matter if download by diagnosis, FOTA ….
- Additional condition
  - Vehicle must be at standstill. RPM = 0.
  - The ECUs involved in the update process must be supplied with power.

Security Features overview

Goal of Secure Flashing

Ensure the **authenticity** and **integrity** of software updates with **Digital Signatures**

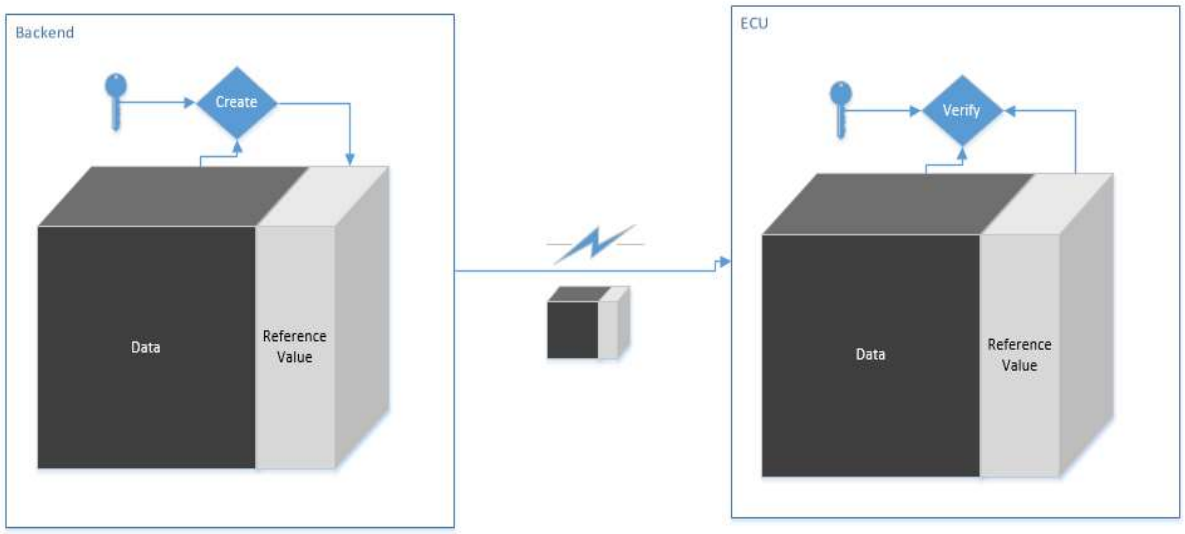
Signature Verification during flashing process using **HSM**

Only **Authenticated** software is flashed and can be activated

Secure Flashing

Definition

Secure Reprogramming, also known as Secure Flashing, is a mechanism to protect the integrity, authenticity and confidentiality of software updates to be flashed in the ECU.



Secure Flashing refers to the prior verification of the software that is to be flashed to an ECU.

Before flashing the software, a check is made to make sure that the software has come from authentic source.

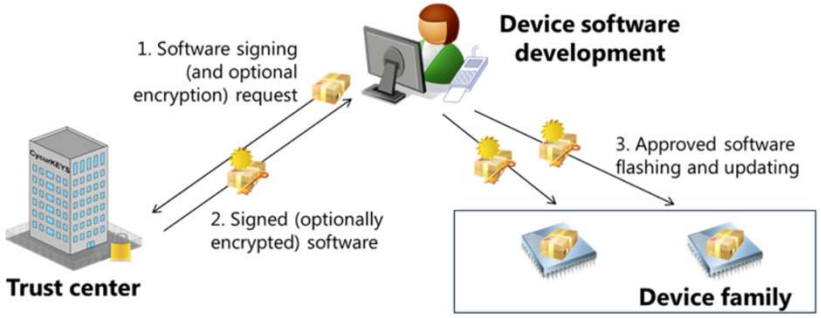
This procedure make use of digital signatures.

The entire digital signature procedure consist of 2 parts. Signing and verification.

Signing happens at trust center and verification happens in the ECU

Secure Flashing

Explanation



Asymmetric cryptographic primitives requiring a private key are used for the creation such that creating a correct digital signature without knowing the private key is practically infeasible.

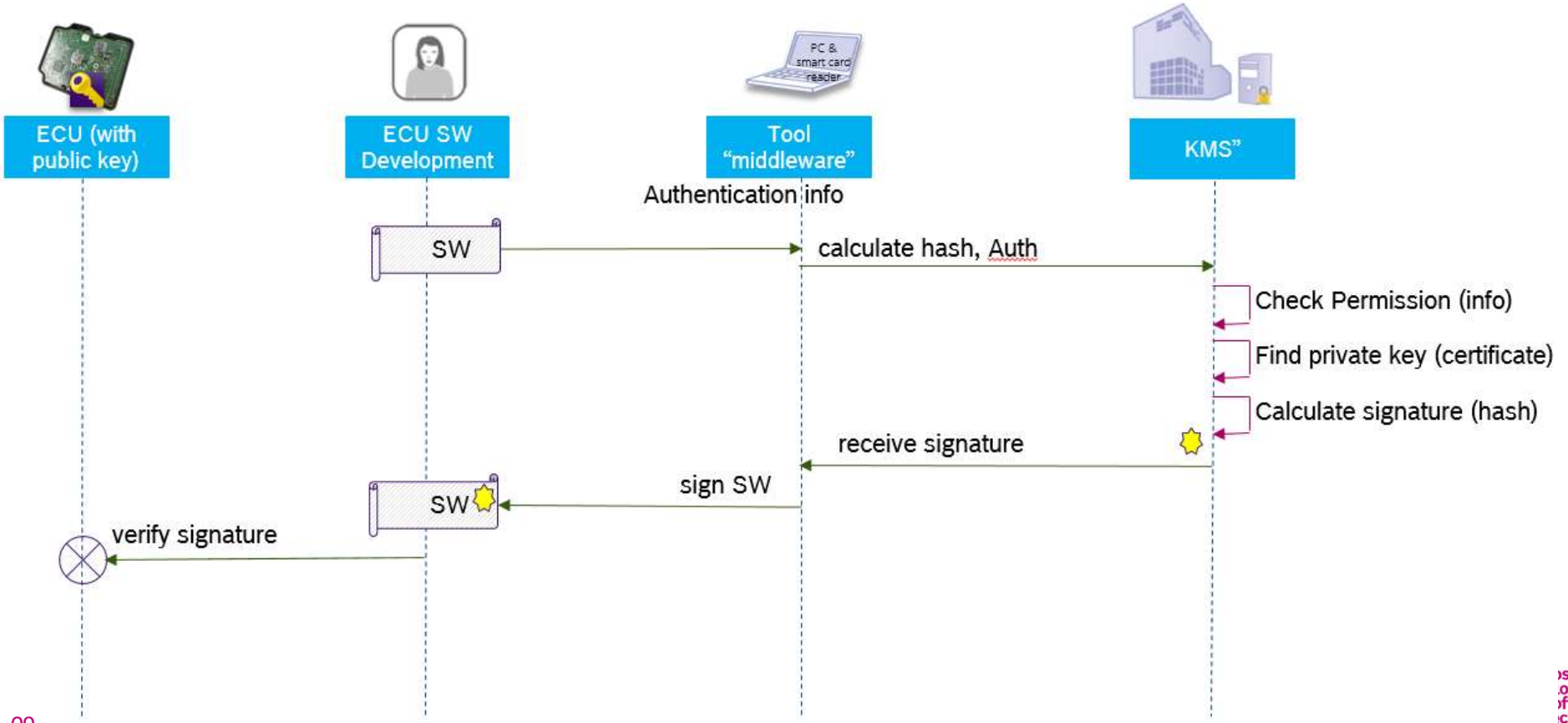
The firmware image and digital signature are thereafter transferred to the ECUs that should be updated. The integrity and authenticity of the received data is verified by checking the received digital signature.

The firmware image can optionally be encrypted to prevent an attacker from acquiring the source code

An attacker not knowing the private key is unable to create a correct digital signature and therefore cannot flash a firmware image created by himself.

Secure Flashing

Explanation



Secure Flashing

Explanation

- The “new” SW is extended by a signature (some kind of “cryptographic checksum”) calculated by its Hash, the private key and a cryptographic function (RSA2048, RSA3072, Elliptic curves…)
  - The signature is generated in a backend or by authenticated users with local smartcard (on local PC)

- After the download of the “new” SW in the ECU, the bootloader uses the cryptographic function with the known corresponding public key on the plain message (or the hash) and calculates a verification signature
  - If the received and the calculated signature values are identical, the bootloader enables the start of the SW.
  - In case of a deviation, the SW will not become operational, ECU-SW will not “leave” the bootloader

Thank you!

Bosch
Global
Software
Technologies
alt_future