IDOLS WITH FEET OF CLAY: ON THE SECURITY OF BOOTLOADERS AND FIRMWARE UPDATERS FOR THE IOT

Lionel Morel | CEA / LIST / DACLE
Damien Couroussé | CEA / LIST / DACLE

NEWCAS – München, Germany – 2019-06-24

BFUs – BOOTLOADERS AND FIRMWARE UPDATERS

Bootloader ?

- Everything that is between the system reset/startup and the startup of the ‘User Application’.
- Also supports the capability to upgrade parts or all of its *firmware*
  - The bootloader component may not be included in this understanding of *firmware*
- Achille’s heel of the whole system: if you control the boot process or the upgrade process, you control the ~~world~~ platform.

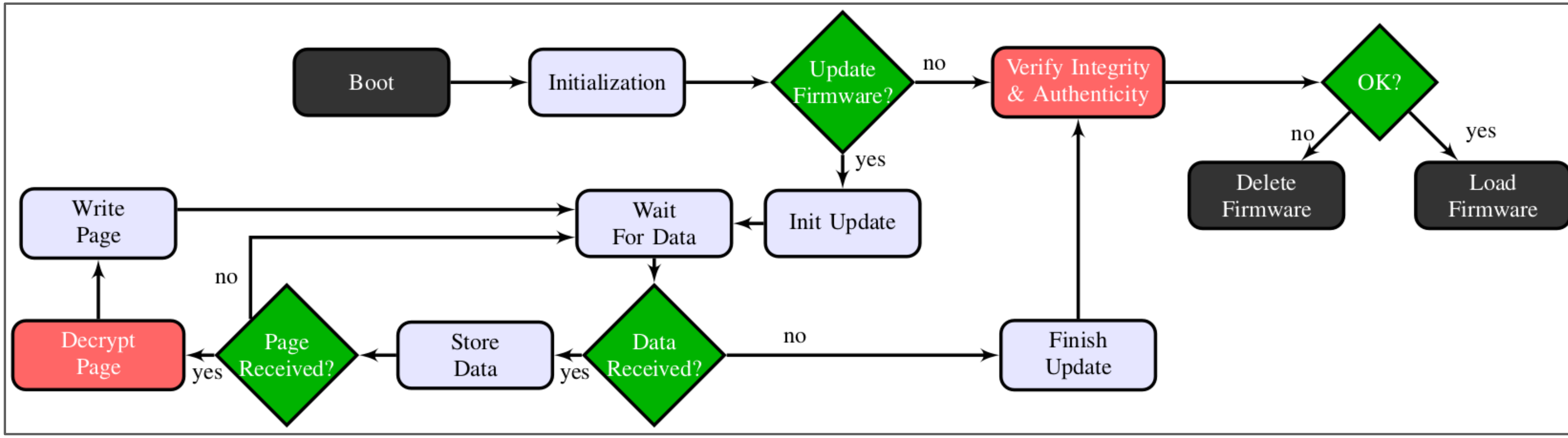
Security properties to support

- **Confidentiality** → encryption functions, usually symmetric
- **Integrity**
  - Of the device → requires hardware support (“*anti tampering*”)
  - Of the firmare → CRC, hash functions, MAC, digital signature
- **Authenticity** → MAC, digital signature

Our credo: BFUs provide a good case to study the security of Embedded/IoT systems

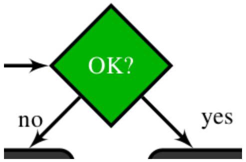
- Logical security: exploits of buffer overflows, ROPs, memory dumps, etc.
- Hardware security, mainly side-channel and fault-injection attacks, reverse engineering
- BFUs integrate cryptography
- But you can target all the glue code around the crypto components!
- A good case study to demonstrate the scalability of analysis tools

GENERIC ARCHITECTURE OF A BFU



Cryptographic functions: implemented in SW, dedicated HW IPs, or SW+specific processor instructions

“System” components: implemented in SW, mostly HW-dependant and/or supported by dedicated HW (e.g. DMA for data movement)

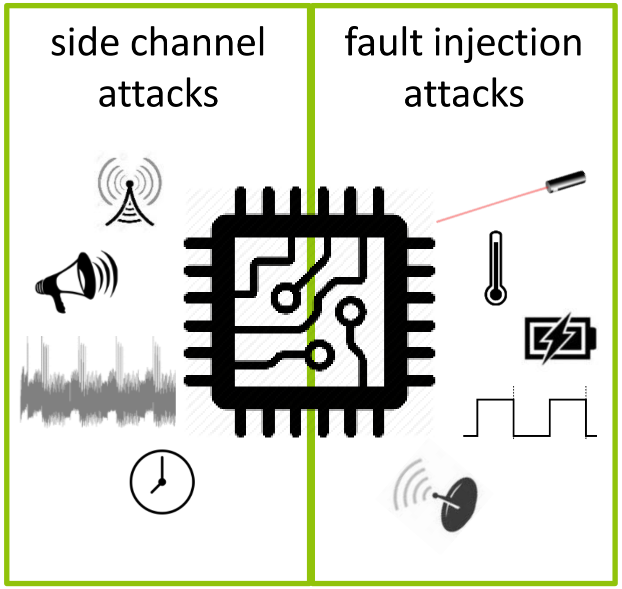


Control logic: implemented in SW

PHYSICAL ATTACKS

A major threat against secure embedded systems

- The most effective attacks against crypto-systems
- Relevant against many parts of CPS/IoT: bootloaders, firmware upgrade, etc.
- Recently used to leverage software vulnerabilities [1]

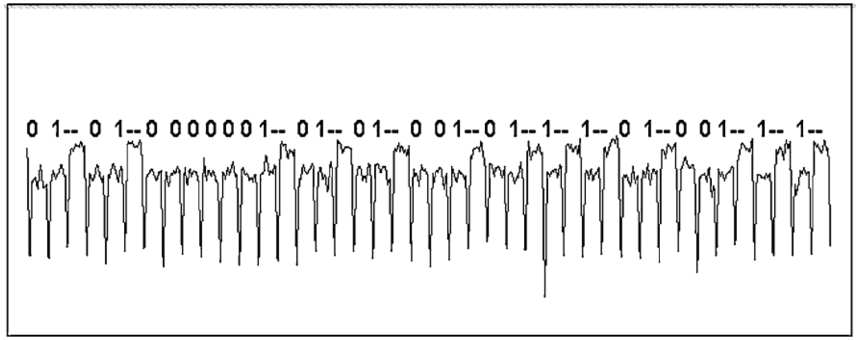


In practice,

- An attacker mostly uses logical attacks if the target is unprotected (e.g. typical IoT devices): buffer overflows, ROP, protocol vulnerabilities, etc.
- All high security products embed countermeasures against side-channel and fault injection attacks. E.g. Smart Cards, payTV, military-grade devices.
  - Using a combination of hardware *and* software countermeasures
- Tools for Side-channel and fault injection are getting really affordable

[1] A. Cui and R. Housley, 'BADFET: Defeating Modern Secure Boot Using Second-Order Pulsed Electromagnetic Fault Injection', presented at the WOOT, 2017.

EXPLOITATION OF SIDE-CHANNEL INFORMATION LEAKAGE

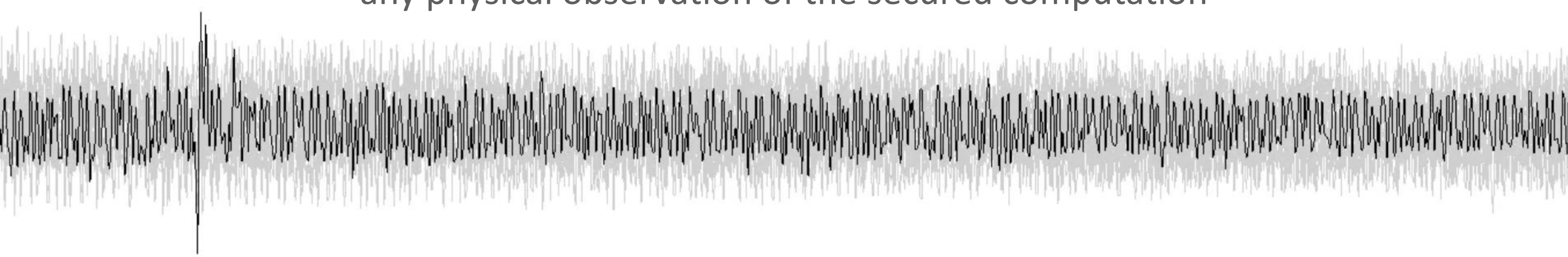


Simple power analysis (SPA)

SPA leaks from an RSA implementation

P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, 'Introduction to differential power analysis', Journal of Cryptographic Engineering, vol. 1, no. 1, pp. 5–27, 2011.

Correlation Power/EM Analysis (CPA/CEMA) – Can be generalised to any physical observation of the secured computation



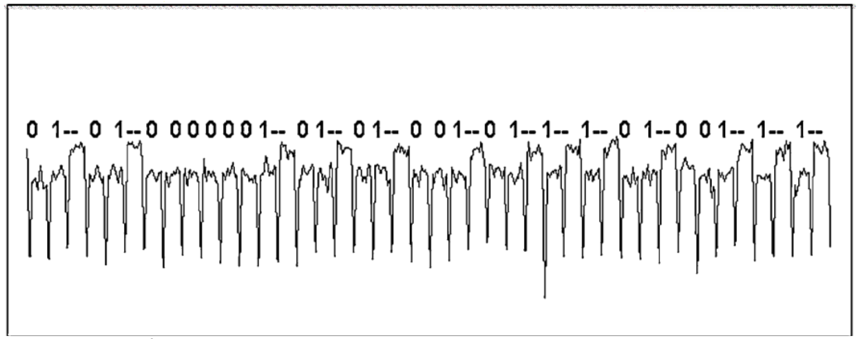
Key found!

- AES, unprotected implementation
- EM traces
- Attack on the output of the 1^{st} SBOX

#265

After the encryption of 4240 Bytes of data!

EXPLOITATION OF SIDE-CHANNEL INFORMATION LEAKAGE

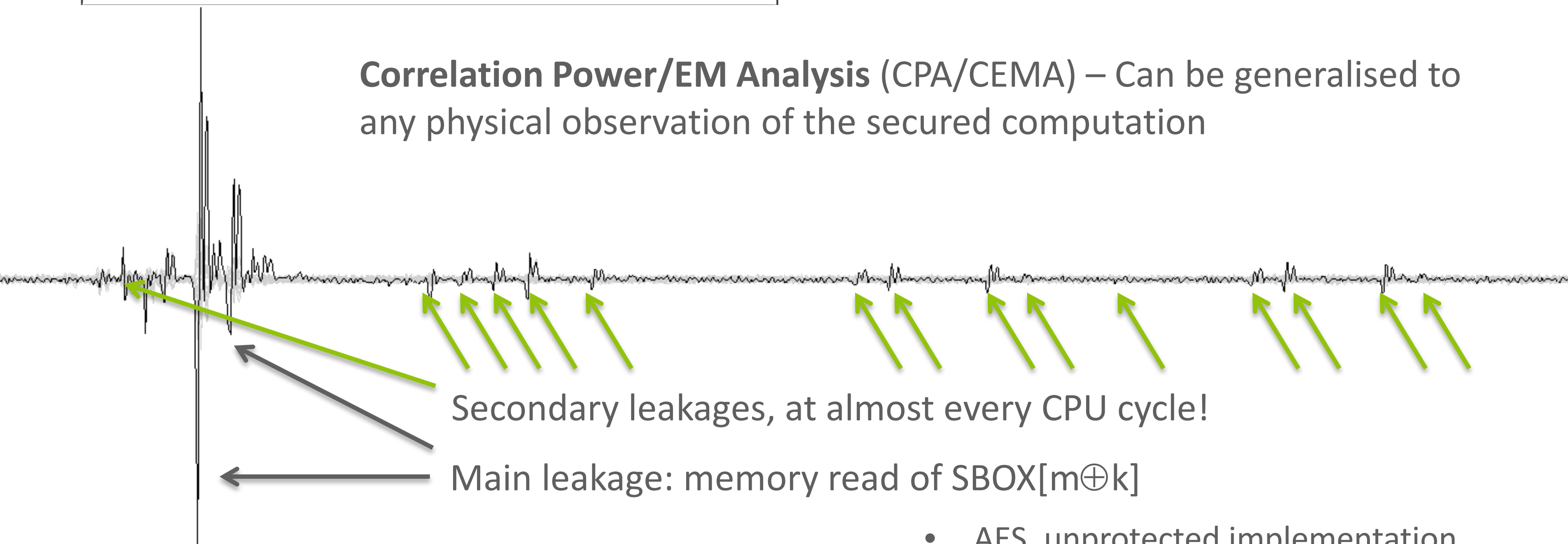


Simple power analysis (SPA)

SPA leaks from an RSA implementation

P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, ‘Introduction to differential power analysis’, Journal of Cryptographic Engineering, vol. 1, no. 1, pp. 5–27, 2011.

Correlation Power/EM Analysis (CPA/CEMA) – Can be generalised to any physical observation of the secured computation



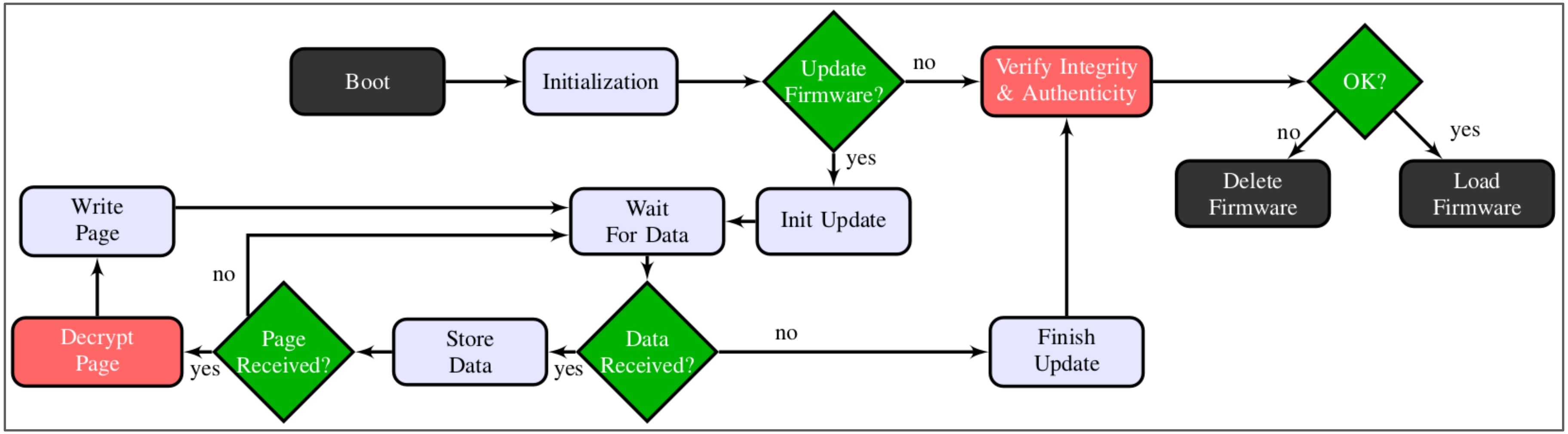
Secondary leakages, at almost every CPU cycle!

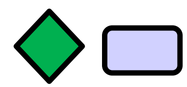
Main leakage: memory read of SBOX[m⊕k]

- AES, unprotected implementation
- EM traces
- Attack on the output of the 1st SBOX

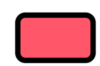
#121278

SIDE-CHANNEL ATTACKS: APPLICATION TO BFUS



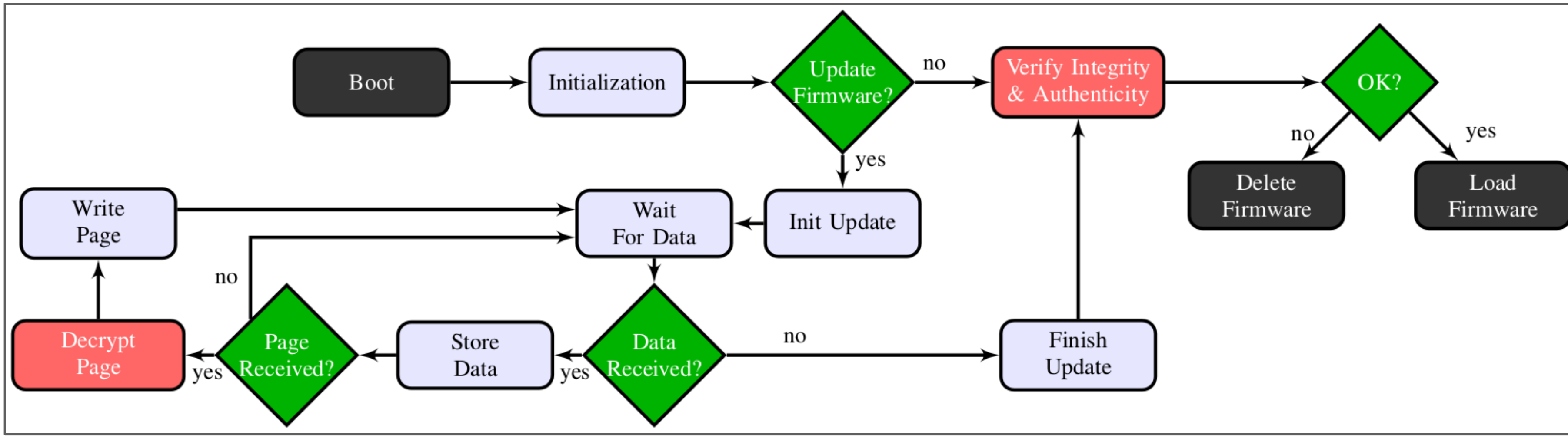


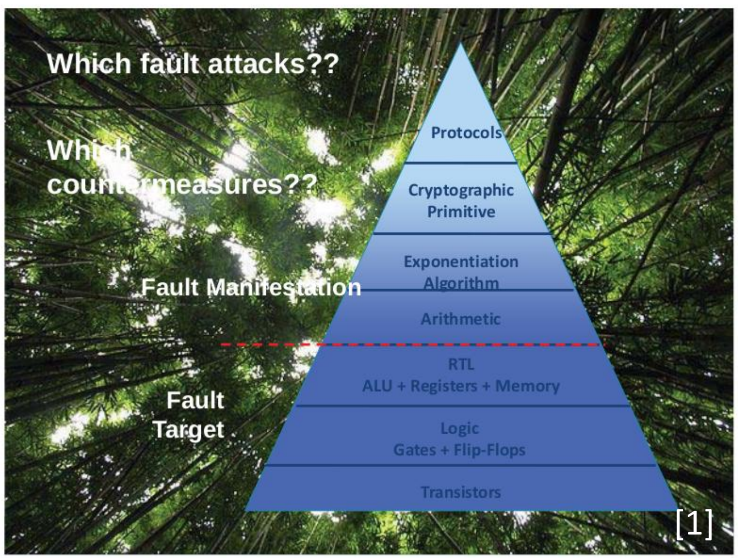
1. Inspection of single side-channel traces
   - Reverse-engineering, e.g., identification of the program structure



2. CPA/CEMA
   - Recovery of secret data, e.g. cipher keys
   - Reverse-engineering of lookup tables

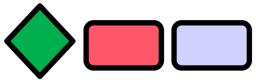
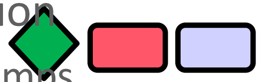
FAULT INJECTION ATTACKS: APPLICATION TO BFUS





[1] I. Polian, M. Joye, I. Verbauwhede, M. Witteman, and J. Heyszl, 'Controlled fault injection: wishful thinking, thoughtful engineering or just luck ?', FDTC, 2017.

Fault models, at the Instruction Set Architecture (ISA) level:

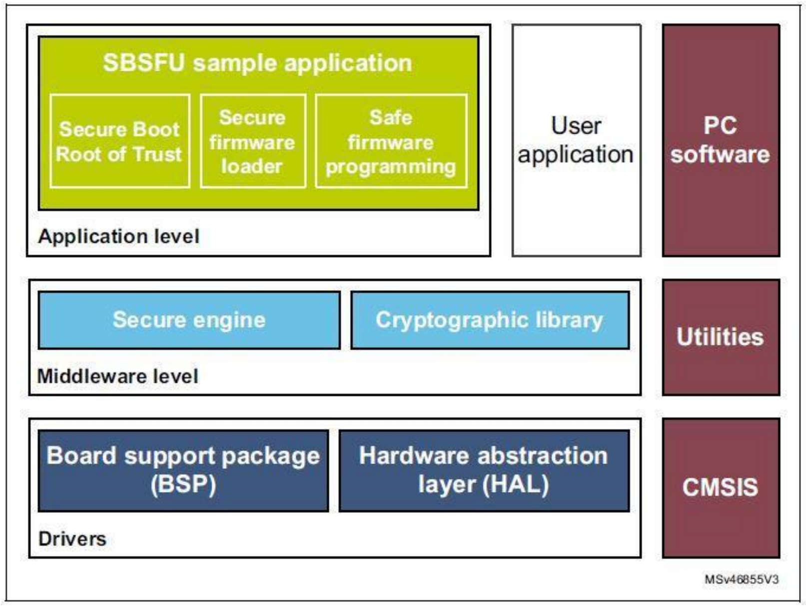
1. Data alteration, down to the bit level. 
   - ROM / RAM, processor registers
   - Bit flip, bit stuck-at
   - Typically: modification of loop counters, crypto data, opcode corruption.
2. Instruction skip, instruction modification 
   - Typically: NOP execution, arbitrary jumps
3. Modification of the control flow, e.g., test inversion 

IoT security: 2 types of product families

1. Integrates AES-256 → clearly not enough
2. Secured bootloaders → Atmel, MicroChip, STMicroelectronics, etc.

Secured bootloader: provides a secured Chain-of-Trust (CoT) encompassing a full boot sequence.

- The SW boot component can be considered as part of the product. Immutable in memory, usually not upgradable.
- Provides a Secure Enclave
  - Secured storage with limited capacity. Usually only for a few encryption keys.
  - Secured execution context with limited processing capability
  - Strong isolation from the User / Application execution domain.
- Only the User/Client app is upgradable
- Does not protect the User Application. i.e., securing the User / Application execution domain is still up to the application developer



Example: X-CUBE-SBSFU

Many interesting initiatives in the RISC-V ecosystem.

- e.g. wolfBoot https://www.wolfssl.com/products/wolfboot

IDOLS WITH FEET OF CLAY: ON THE SECURITY OF BOOTLOADERS AND FIRMWARE UPDATERS FOR THE IOT

Lionel Morel | CEA / LIST / DACLE
Damien Couroussé | CEA / LIST / DACLE

damien.courousse@cea.fr

leti
Centre de Grenoble
17 rue des Martyrs
38054 Grenoble Cedex

list
Centre de Saclay
Nano-Innov PC 172
91191 Gif sur Yvette Cedex