VUB
SECURE EXECUTION FOR EMBEDDED ENVIRONMENTAL MONITORING APPLICATIONS

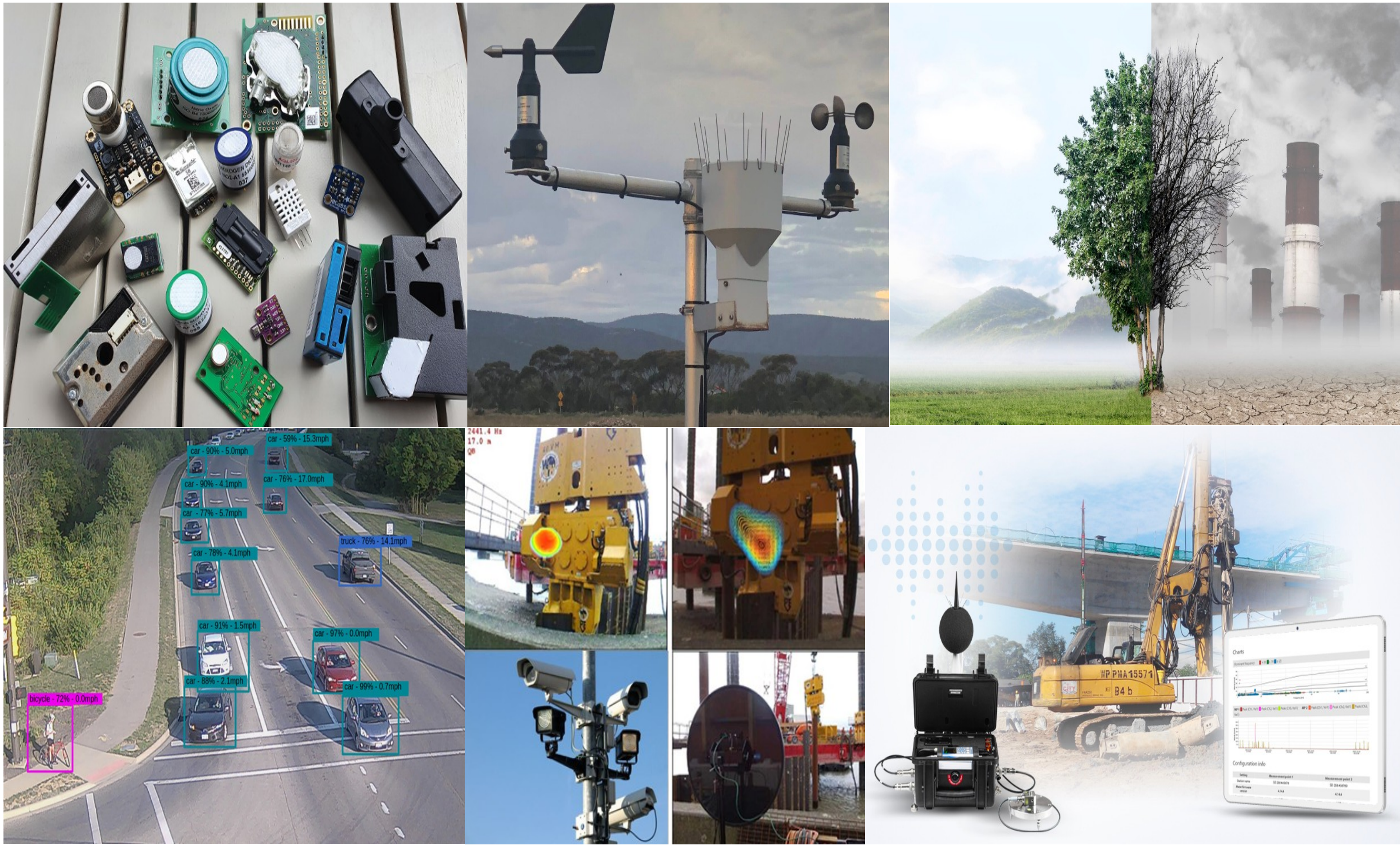
Laurent Segers
Baciu Vlad-Eusebiu
An Braeken
Bruno da Silva
Abdellah Touhafi

29 / 08 / 2024

AGENDA

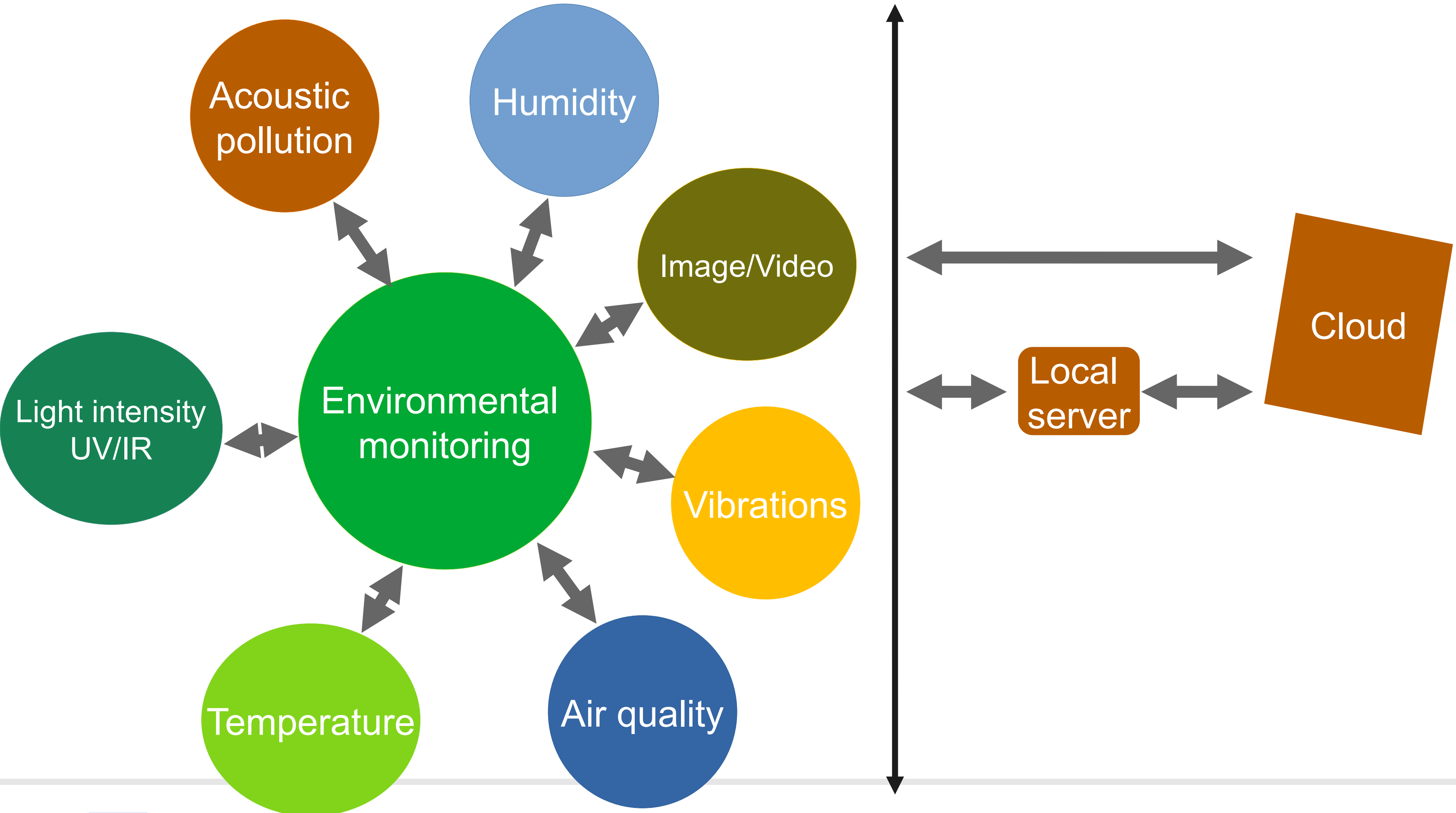
1. Environmental monitoring
2. Embedded security requirements
3. Platform selection
4. Prototype
5. Embedded firmware & considerations
6. Symmetric key renewal
7. Performance analysis
8. Discussions & final notes
9. Follow-up projects

ENVIRONMENTAL MONITORING



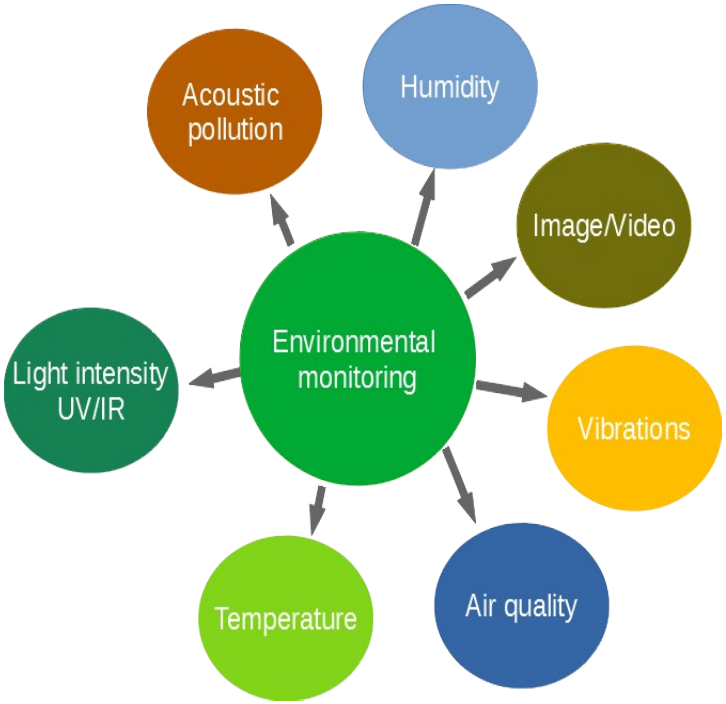
ENVIRONMENTAL MONITORING

TOPOLOGY



VRIJE UNIVERSITEIT BRUSSEL

ENVIRONMENTAL MONITORING



UC1: SECURE LOW-END SENSING

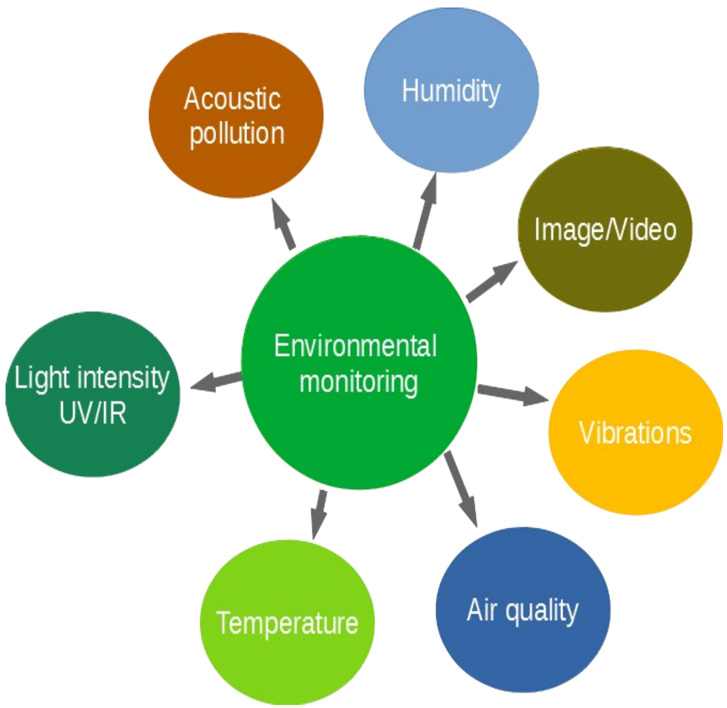
- Limited memory/processing capabilities
- Capable of reading sensors with low update rates (i.e. 1Hz, 10Hz)
- Data integrity & confidentiality of sensor-readouts
- Trusted GPS & RTC
- Lightweight key agreement protocol using PUF

UC2: SECURE HIGH SENSING

- Secure AI modelling

ENVIRONMENTAL MONITORING

UC1: LOW-END



Risks & mitigation

- Moving device to other location
  Location awareness (GPS) can mitigate security risks

- Wireless communication → spoofing, jamming, read-out of data, data alteration
  → *Store jammed data locally until successful retransmission*
  → *Encryption/integrity protection of transmitted data*

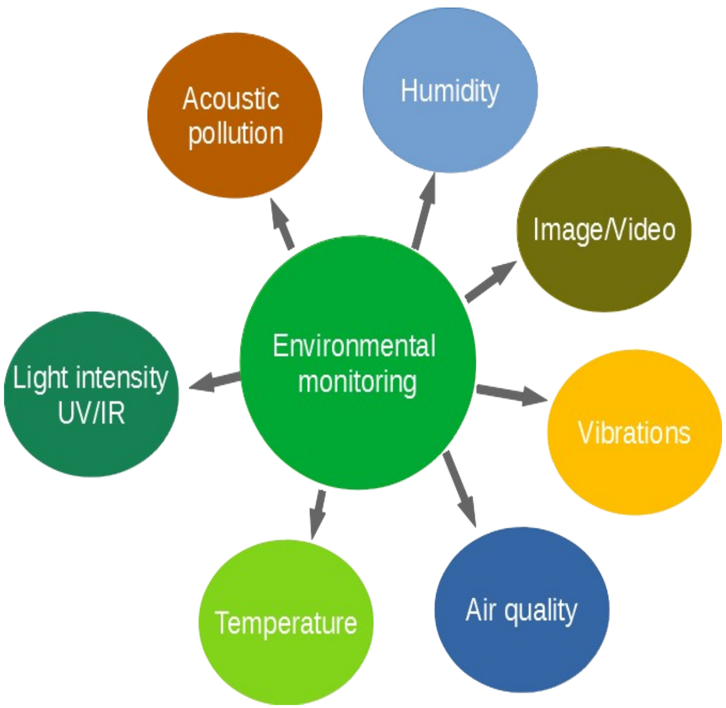
- Modifying/Reading of locally stored data
  Data encryption, data integrity check

- Firmware (mis)configuration
  → *integrity test during attestation*

- Over the air updates compromised with spoofed firmware/configuration
  → *Authentication + encryption of firmware*

ENVIRONMENTAL MONITORING

UC2: HIGH-END SENSING



Risks & mitigation

- Includes low-end risks & mitigations
- Tampering of AI models
  → Secure attestation of AI model

VRIJE UNIVERSITEIT BRUSSEL

EMBEDDED SECURITY REQUIREMENTS

UC1 + UC2: HARDWARE – SILICON SUPPORTED

- Minimal Hardware-based code execution isolation if possible
  → TrustZone
- Basic Root-of-Trust (for some applications)
- Secure boot
- Trusted peripherals (when possible)
- Optimizations for secure storage
- Encrypted + authenticated communication
- Regular key renewal → communication encryption
- Secure over the air updates (if possible)

SELECTION

UC1: LOW/MID-RANGE DEVICES (JUNE 2023)

NXP/Freescale	STMicroelectronics	Microchip
LPC5500-series based on the **single core ARM-Cortex-M33 MCUs**	STM32 based on **ARM-Cortex-M33** (STM32L5 and STM32U5) ultra-low-power MCUs	PIC32CM5164 LS60/LS00 based on **ARM-Cortex M23**
• TrustZone • Energy efficiency, up to 150MHz • SRAM PUF-based RoT • Encrypted images • ~ 4.5€/pc (1000pc)	• TrustZone • Low-power, up to 160MHz • Cryptographic modules integrated • ~7.5€/pc (1000pc)	• TrustZone • Ultra low-power, up to 48MHz • Cryptographic modules integrated • Exist in secure and non-secure variants • ~4€/pc (1000pc)

VRIJE UNIVERSITEIT BRUSSEL

UC2: MID-RANGE DEVICES (CURRENT)

NXP/Freescale	STMicroelectronics	Microchip
MCX N94x and N54x - series based on the **dual core ARM-Cortex-M33 MCUs**	STM32 based on **ARM-Cortex-M33** (STM32H5) High performance MCUs	PIC32CK SG/GC **ARM-Cortex M33**
• TrustZone • High performance, up 150MHz • SRAM PUF-based RoT • Encrypted images • ~ 11€/pc (1000pc)	• TrustZone • High performance, running up to 250MHz • Cryptographic modules integrated • ~8€/pc (1000pc)	• TrustZone • Low-power, up to 120MHz • Cryptographic modules integrated • Exist in secure and non-secure variants • ~10€/pc (1000pc)

Use Case 1

PROTOTYPE

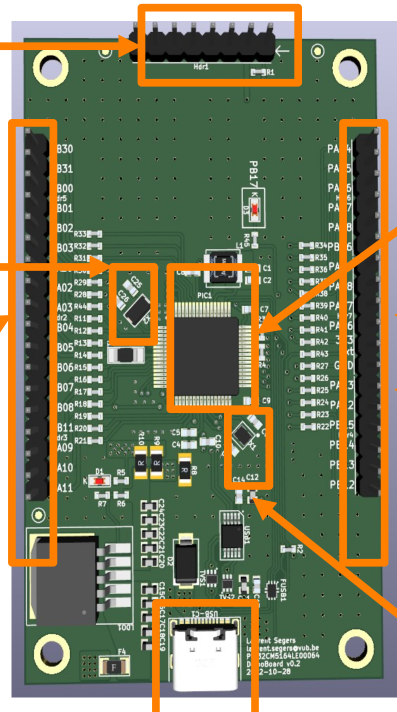
MICROCHIP PIC32CM5164 ARM CORTEX-M23

Custom designed board

Programming header

RTC @ 32kHz

IO (+interruptable IO) Sercom (SPI, I2C, UART)



PIC32CM5164LE00064 (non-secure)

PIC32CM5164LS00064 (secure)

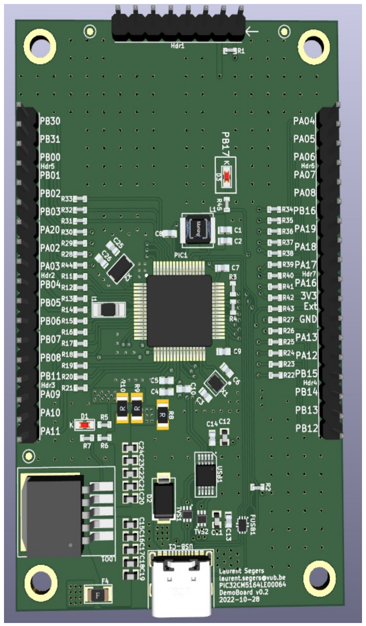
External power

Main crystal @ 32MHz

USB for power over USB + communication to PC

PROTOTYPE

MICROCHIP PIC32CM5164 ARM CORTEX-M23



Based on ARM23 core platform with 512kB flash, 64kB SRAM, 32kB boot ROM

Offers TrustZone (5 regions in flash, 2 regions in data flash and 2 regions in SRAM)

Tamper resistant secure data flash for sensitive data storage

1 TRNG, AES-256/192/128, multiple SHA methods

Secure boot with customizable secure boot public key

Optimized for secure storage + TrustRAM

Up to 8 anti-tamper output IO + secure pin multiplexing to isolate secure communication channels

Unique 128-bit serial number

Separate registers for secure and non-secure application

PROTOTYPE

SENSOR MODULE

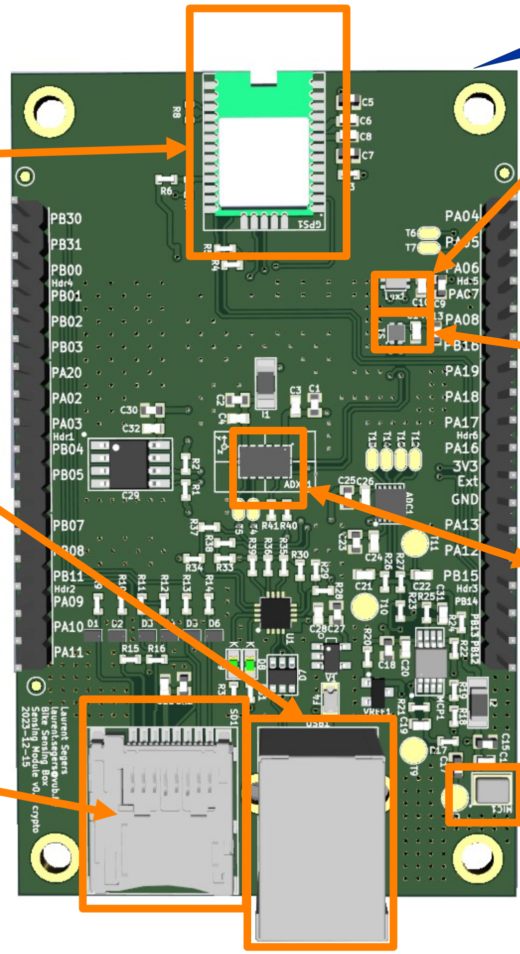
Grouping sensors in secure/non-secure peripherals

GPS (L96-M33)

VEML3328
light sensor (RGB+IR)

SHT41 temperature + humidity sensor

UART to USB communication to PC



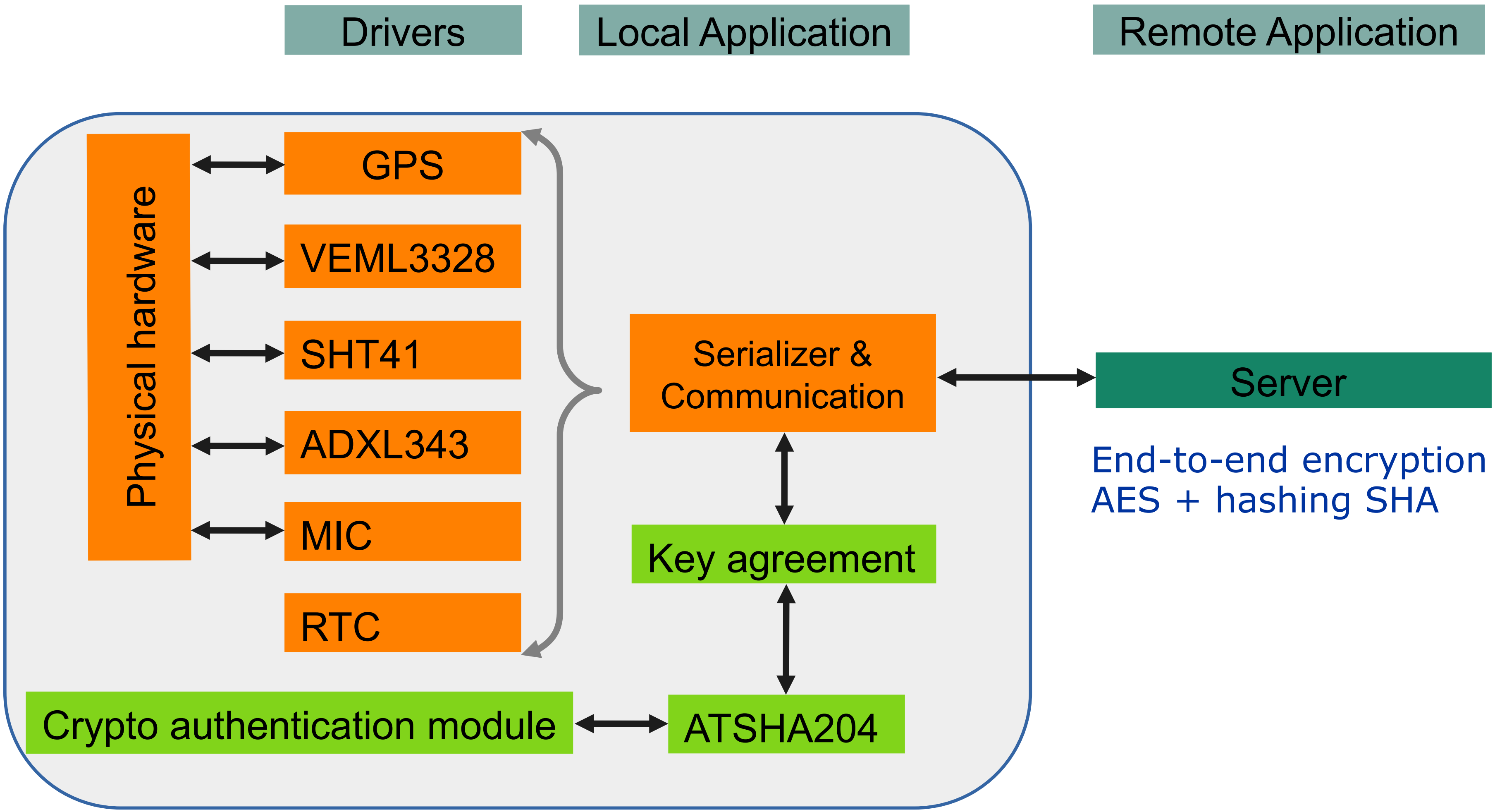
ADXL343 3-axis accelerometer

SD-card for logging

SPU0410LR5H-QB analog microphone + SPI ADC

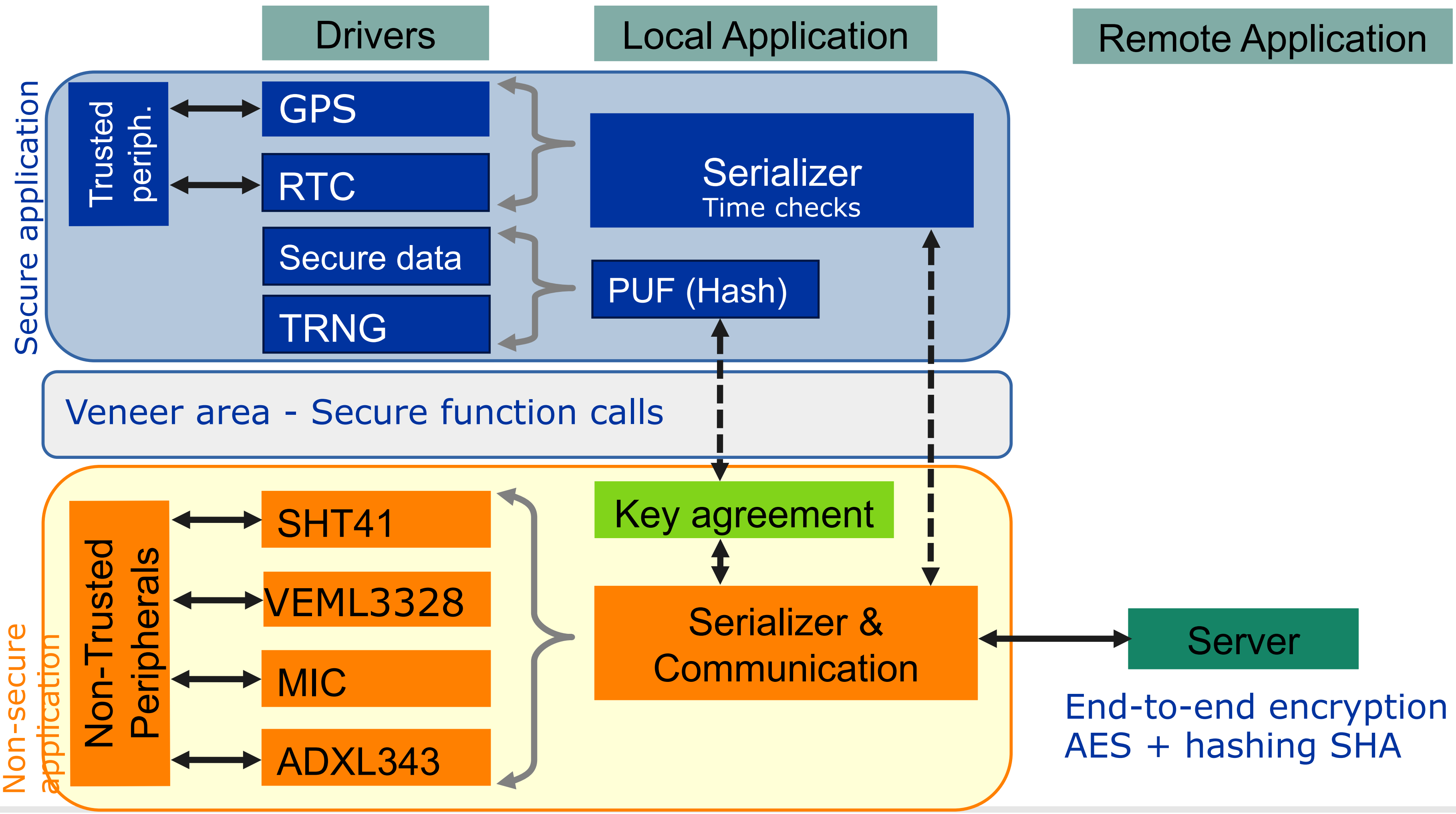
EMBEDDED FIRMWARE (1)

MODULAR APPROACH WITHOUT TRUSTZONE



EMBEDDED FIRMWARE (2)

MODULAR APPROACH WITH TRUSTZONE - PIC32CM5164LS



VRIJE UNIVERSITEIT BRUSSEL

CONSIDERATIONS

PIC MICROCONTROLLERS WITH TRUSTZONE

- One program flow on regular microcontrollers without TrustZone
- TrustZone involves re-thinking application into secure and non-secure code → 2 program flows!
- Special function calls between secure and non-secure code (veneers)
- Pre-compiled libraries (STDIO, Wolfcrypto,...) can not be used in TrustZone
- Only deterministic "C" in TrustZone, C/C++ for regular application
- Hardware peripherals (sensors and communication) bound to secure/non-secure code → double set of hardware registers
- PUF functionality resides in TrustZone

SYMMETRIC KEY RENEWAL

SECRET COMMUNICATION KEY

Secret key might get leaked

→ Secret key can be reverse-engineered via firmware extraction

→ Sensitive data leading to secret key might get compromised

→ Encryption information can be "learned" based on pattern search

→ Firmware errors might leak sensitive information

Need for secret key refresh – Key agreement protocol

Key agreement protocols generate "predictable" new keys
→ Physical Unclonable Function (PUF)?

KEY AGREEMENT PROTOCOL WITH PUF: OUTLINE

1: Registration → 2: Initialisation → 3: Key agreement

PUF → P(input)=output
→ silicon variable but repeatable

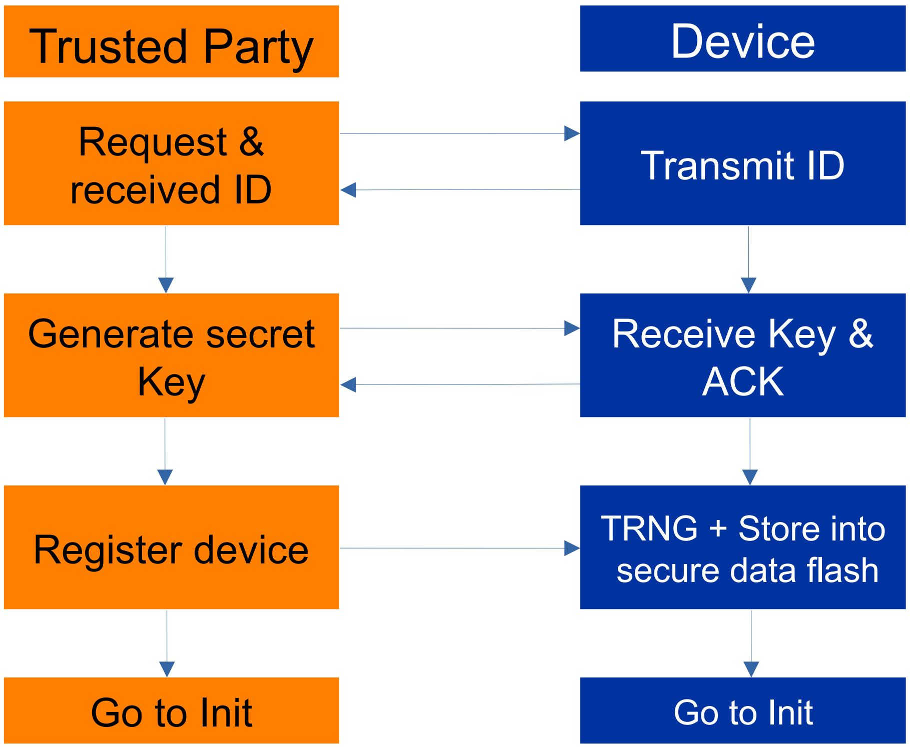
Freescale ARM Cortex M33 has built-in SRAM-based PUF

!!Most MCUs do not have SRAM-based PUF!!

TrustZone + secure data flash:
→ 128-bit TRNG used to mimic PUF (stored in flash)
→ secure data flash erased on intrusion→
→ TRNG + secure data flash accessible via TrustZone

SYMMETRIC KEY RENEWAL

KEY AGREEMENT 1: REGISTRATION



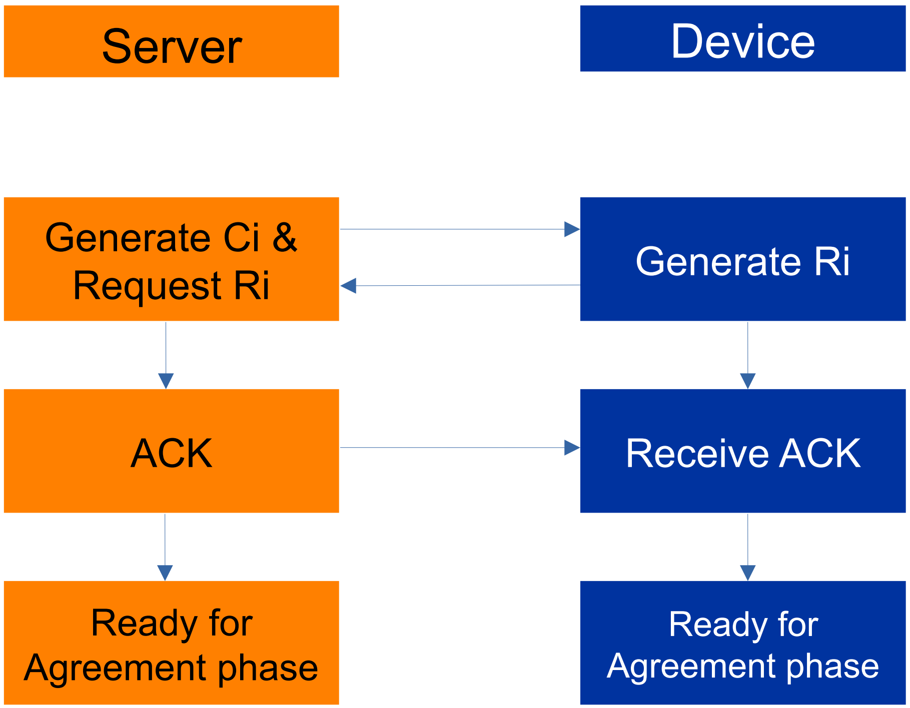
Occurrence: once

Physical connection or via Trusted Third Party

Device initializes the secure data flash:

→ Store 128bit-TRNG value
→ Store secret key

KEY AGREEMENT 2: INITIALISATION



Occurrence: after registration, once

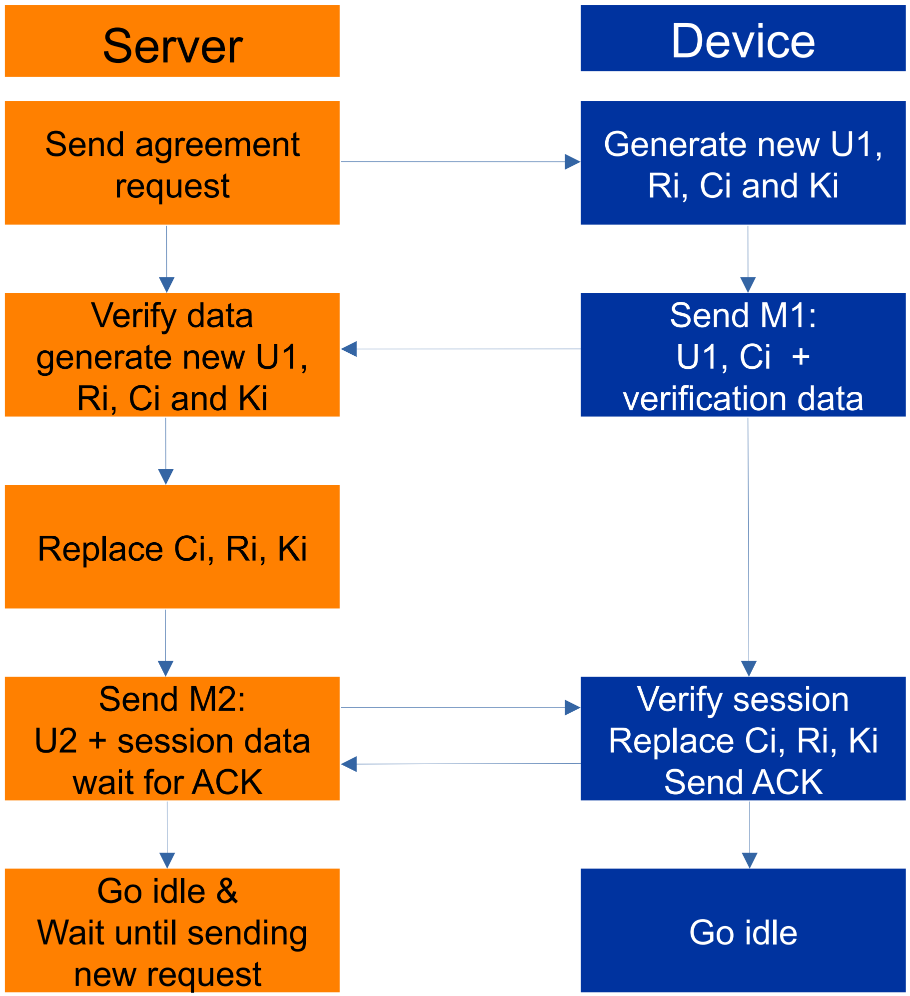
Via secure connection

Hash(Ci|TRNG)=P(Ci)
Hash(Ki|P(Ci)) = Ri
Generated in TrustZone

Ri = response
Ci = challenge

SYMMETRIC KEY RENEWAL

KEY AGREEMENT 3: KEY AGREEMENT/RENEWAL



Occurrence: when required, multiple times

Via secure connection

Ri = response
Ci = challenge
Ki = symmetric secret key

Hash(Ci|TRNG)=P(Ci)
Hash(Ki|P(Ci)) = Ri
Generated in TrustZone

U1, U2: random values

M1, M2: exchange messages

PERFORMANCE & RESOURCE CONSUMPTION

PIC32CM5164LS00064 + TRUSTZONE

Memory allocation

	Non-secure	**Secure (TrustZone)**
Code flash	207 / 256 kB	15 / 254 kB + 2kB veneer functions
RAM	16.64 kB / 32 kB	500B / 32 kB
Data (secure) flash	0 kB	100B / 16 kB

Transferring data from non-secure to secure application

Message length (bytes)	**CPU ticks**	**Time (us)**
16	1184	24.7 ± 2.6
32	1779	37.1 ± 2.3
48	2419	50.4 ± 3.2
64	3101	64.6 ± 4.2
80	3724	77.6 ± 4.2

❗ Including address pointer and array length validity

PERFORMANCE & RESOURCE CONSUMPTION

AES-128 ENCRYPTION + SHA-256 HASHING

Transmission overhead #bytes

→ Data sent in “plain readable” format: ~38-84 bytes per packet
→ Key agreement: up to 140 bytes per packet

AES-128 CBC:

→ encryption + IV: + 17 to 32 bytes
→ SHA-256 hashing: +32 bytes
Total overhead : 49-64 bytes => +- 100% on average

AES-128 GCM:

→ encryption + IV: + 16 bytes
→ GCM tag: +32 bytes
Total overhead : 48 bytes => +- 75% on average

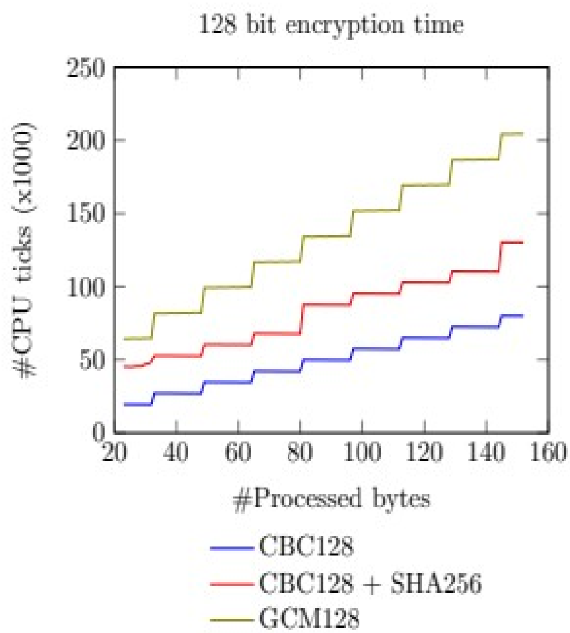
Memory / Flash overhead

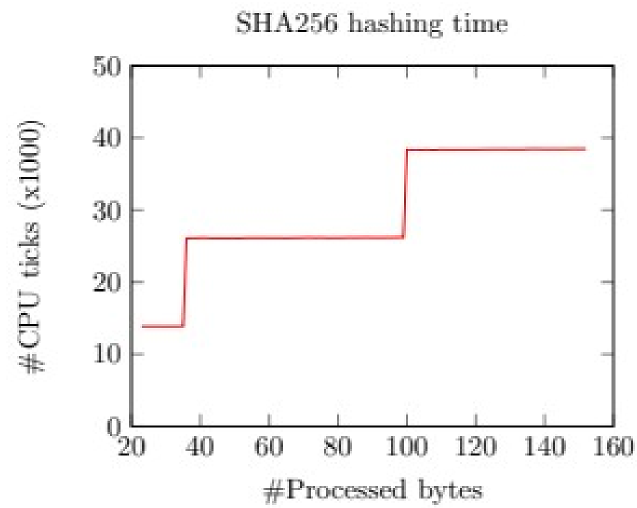
With crypto: 215kB flash / 17.1kB SRAM
Without crypto: 207kB flash / 15.2kB SRAM

PERFORMANCE & RESOURCE CONSUMPTION

AES-128 ENCRYPTION + SHA-256 HASHING

Encryption/hashing time overhead (@48MHz)





AES-128 encryption time increases per block of 16 bytes
SHA-256 hashing increases per block of 64 bytes

<u>Example:</u>
Encryption + hashing of 100 bytes
$T_{required}$ = 1.98 ms (AES-CBC + SHA) → 50kBps
$T_{required}$ = 3.16 ms (AES-GCM) → 31kBps

VRIJE UNIVERSITEIT BRUSSEL

PERFORMANCE & RESOURCE CONSUMPTION

KEY AGREEMENT PROTOCOL

Computation overhead – multiple stages (@48MHz)

Step	CPU ticks	Time (us)
Generate U_1	530	11 ± 0
Hash step 1	18,718	390 ± 1.6
* Generate R_{i+1}	30,500	635 ± 3.2
Transmit M_1	942	19.8 ± 0
Compute session	40,811	850 ± 5.5
Store K_{i+1}	49,865	1034 ± 8.9
Store C_{i+1} & R_{i+1}	59,180	1233 ± 9.0
Total		<5ms

* Including PUF computation in TrustZone

MICROCHIP EMBEDDED TOOL DEVELOPMENT: USER FRIENDLINESS

Device configuration with MPLab X IDE (6.x)
+ Harmony

Code generation of drivers and configuration
→ engineer should focus on applications...

Each new version improves + new features, however...
→ project discrepancies
→ compiler flag discrepancies
→ ~~new project then required~~ → load project dependencies first

Solution/workaround
→ design with harmony/libraries during project creation
→ only update code later on
→ write own drivers on top of CMSIS if possible

SUMMARY

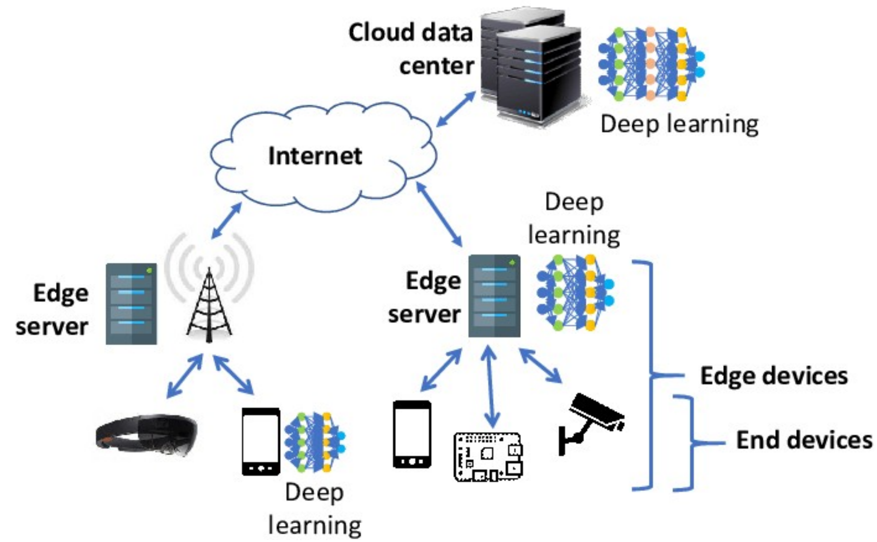
- Low-end Microchip ARM23 (ARMv8 architecture) based platform selected and programmed
- TrustZone and secure remote communication
- Firmware development challenges
- Fine-grained impact analysis of TrustZone and secure communication
- Lightweight key agreement protocol using PUF
- All Sercoms (I2C, SPI, UART) are used + some methods hit MCU processing boundaries
- Limitations of programming tools & resolution

Use Case 2

MACHINE LEARNING ON EDGE AND END DEVICES

CHALLENGES

- IoT devices generate tremendous
- amount of data (order of zettabytes)
- Bottleneck at cloud back-end



Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. Proceedings of the IEEE, 107(8), 1655-1674.

THE CASE FOR EDGE COMPUTING

- Democratization of on-device intelligence
- Reduced latency and increased energy efficiency
- Make endpoint devices more consistent and reliable
- Enhance privacy
- Increasing trend of TinyML

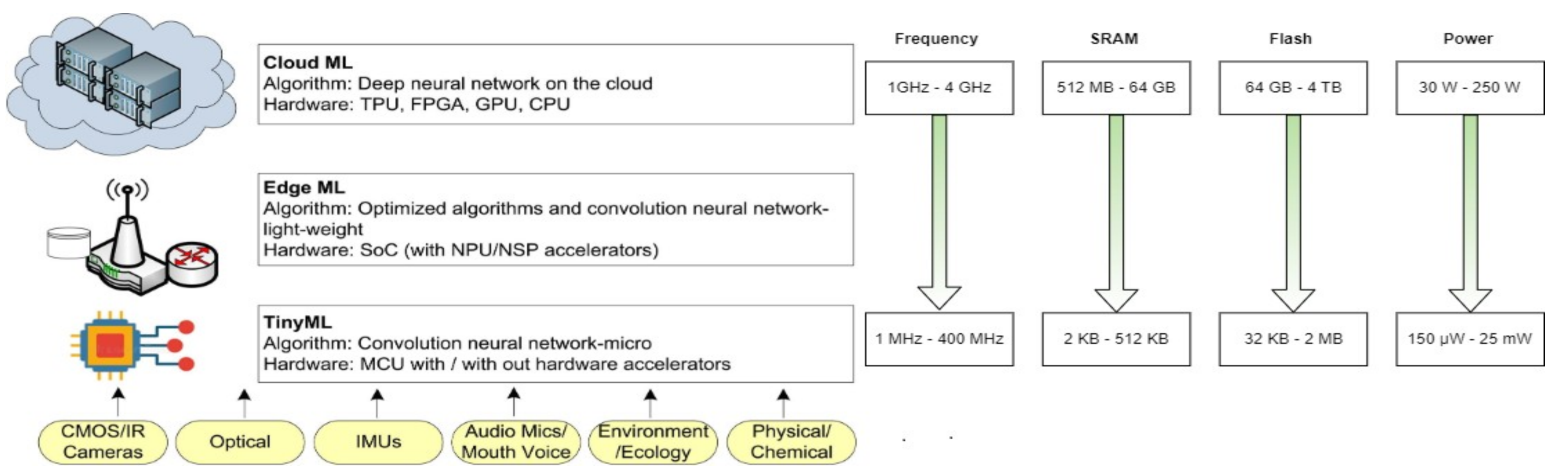
MACHINE LEARNING ON EDGE AND END DEVICES

TINY MACHINE LEARNING

- Combines machine learning with embedded systems
- Bring ML to low-powered devices
- Work on some pretty unimpressive hardware

CHALLENGES

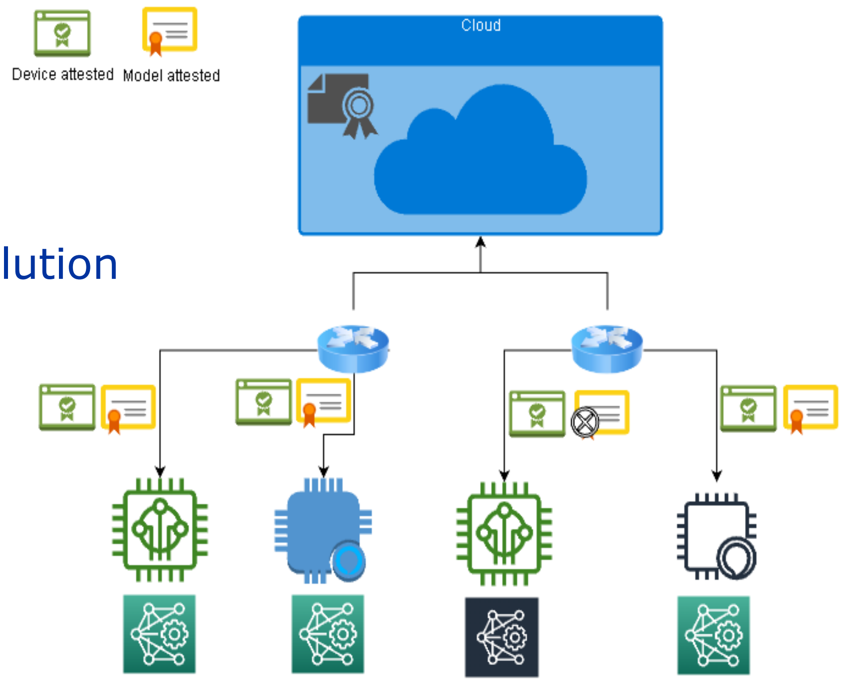
- Power consumption
- Algorithm optimization
- Security



ML MODEL ATTESTATION

MOTIVATION

- Ensure integrity of the ML application
- Prevent unauthorized modifications of the ML model
- Secure model verification
- Ensure device identity and authenticity



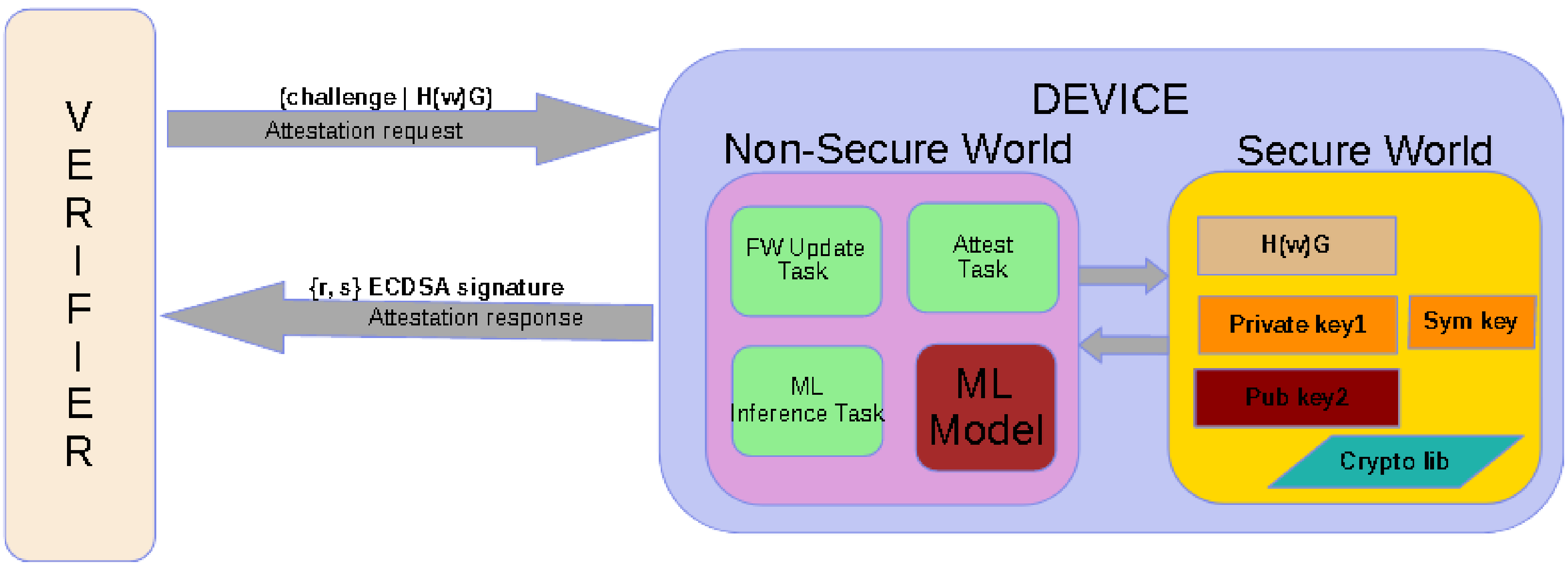
APPLICATIONS

- Environmental monitoring: noise, air pollution
- Network monitoring: malware detection
- traffic analytics
- Health monitoring: wearable activity
- recognition, vital signs measurement

ML MODEL ATTESTATION

ARCHITECTURE

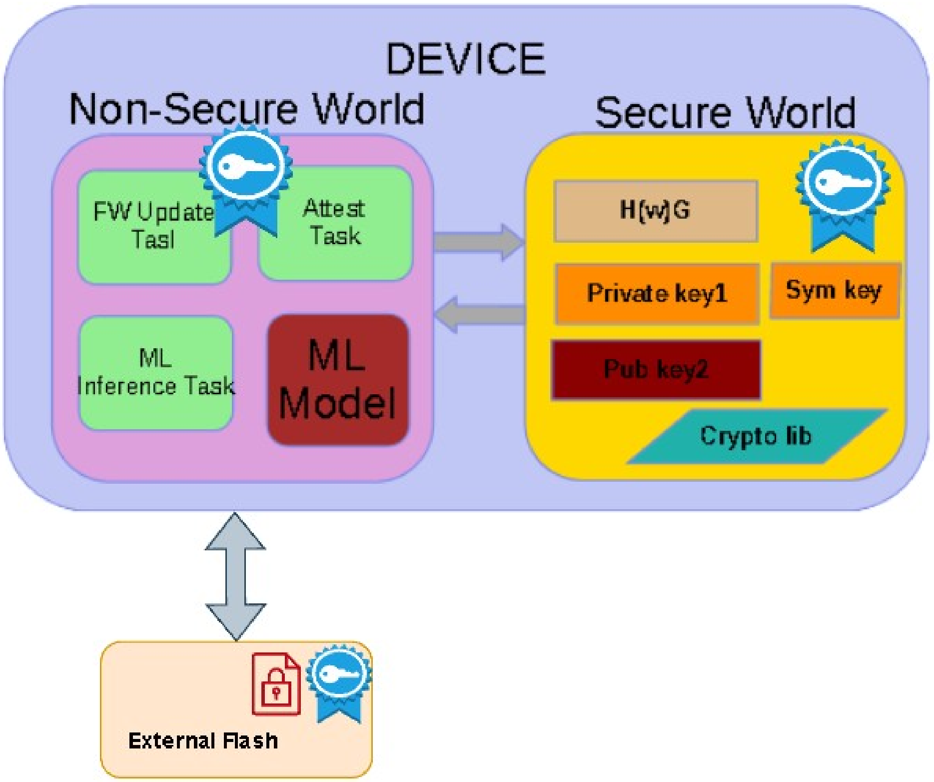
- A verifier wants to check if a device with a known public key is
- running the correct model
- Zero-knowledge proof on both H(w) and the private key
- Device is provisioned with the public-private key pair, H(w)G and the ML model



ML MODEL ATTESTATION

REQUIREMENTS

- Trusted execution environment (TrustZone)
- Secure storage
- On-chip cryptographic engine
- Root-of-trust mechanism
- Minimum 512 kB internal flash,
- 256 kB SRAM
- Mid-range MCU, minimum 200 MHz
- Memory mapped external flash



ML MODEL ATTESTATION

EMBEDDED PLATFORM: STM32H573I-DK

- STM32H573 ARM Cortex M33
- 2Mbytes of flash, 640 Kbytes of SRAM
- 512-Mbit Octo-SPI NOR flash
- Hardware Security Module
    - Two AES coprocessors
    - On-the-fly decryption of external flash
    - Hash hardware accelerator
    - TRNG, SP800-90B compliant
    - Secure data storage
    - Internal/external tamper detection
- Immutable root of trust (ST-iROT)
- STM32Trust ecosystem

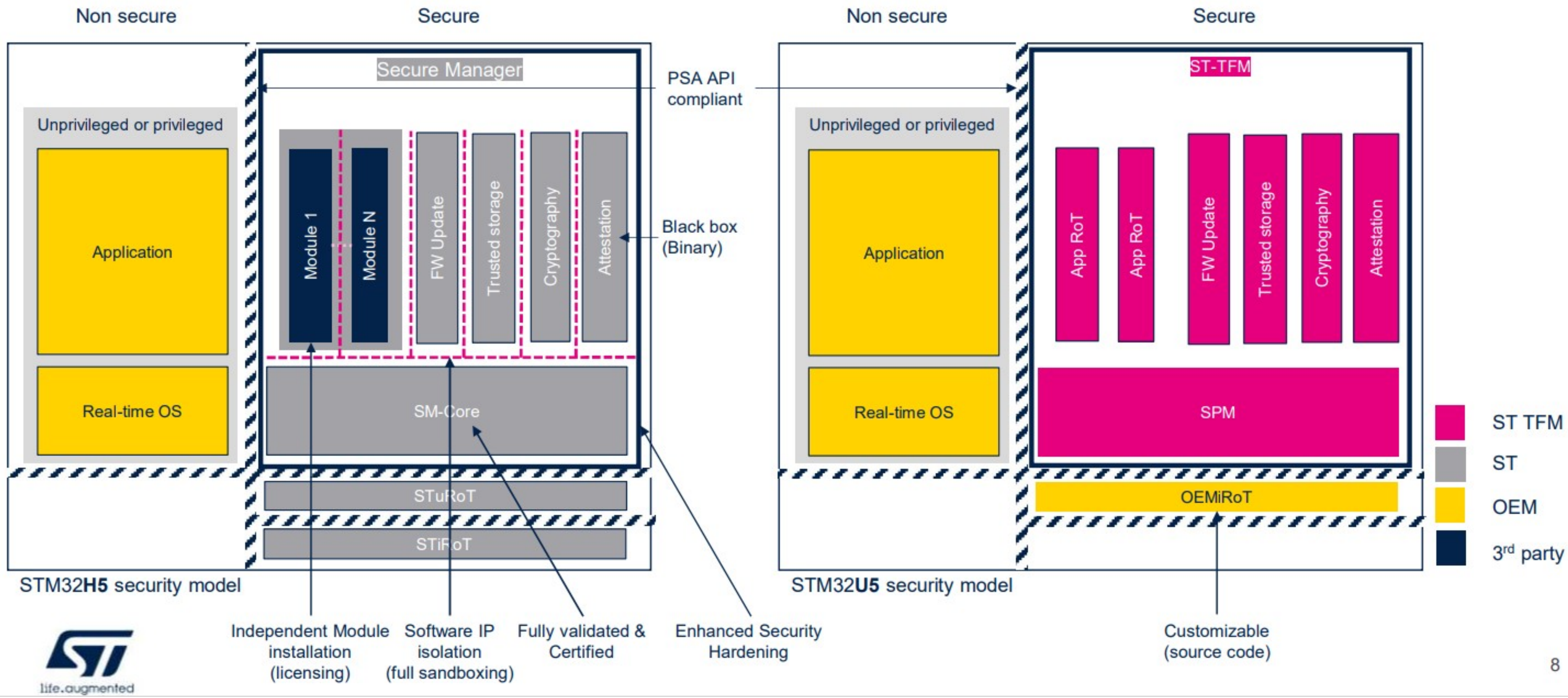
ML MODEL ATTESTATION

STM32TRUST ECOSYSTEM – TRUSTED EXECUTION ENVIRONMENT

ST Secure manager	Trusted Firmware M/A	OPTEE
A security partition manager delivered in binary form.	Open-source software framework	Companion for a non-secure Linux kernel, part of OpenSTLinux
• PSA Level 3 isolation • Secure boot • Cryptography (HW) • Internal trusted storage • Firmware Update • SDK for nonsecure application development • Cortex-M series	• PSA level 2 isolation • Open Source MCUboot • Mbed-crypto (HW/SW) • Secure storage • Firmware Update • Cortex-M or Cortex-A series	• Compliant with GlobalPlatform TEE API specifications • Linux secure boot • (OP-TEE → TF-A → U-Boot → Kernel) • Based on Trusted Firmware A • Cortex-A series

ML MODEL ATTESTATION

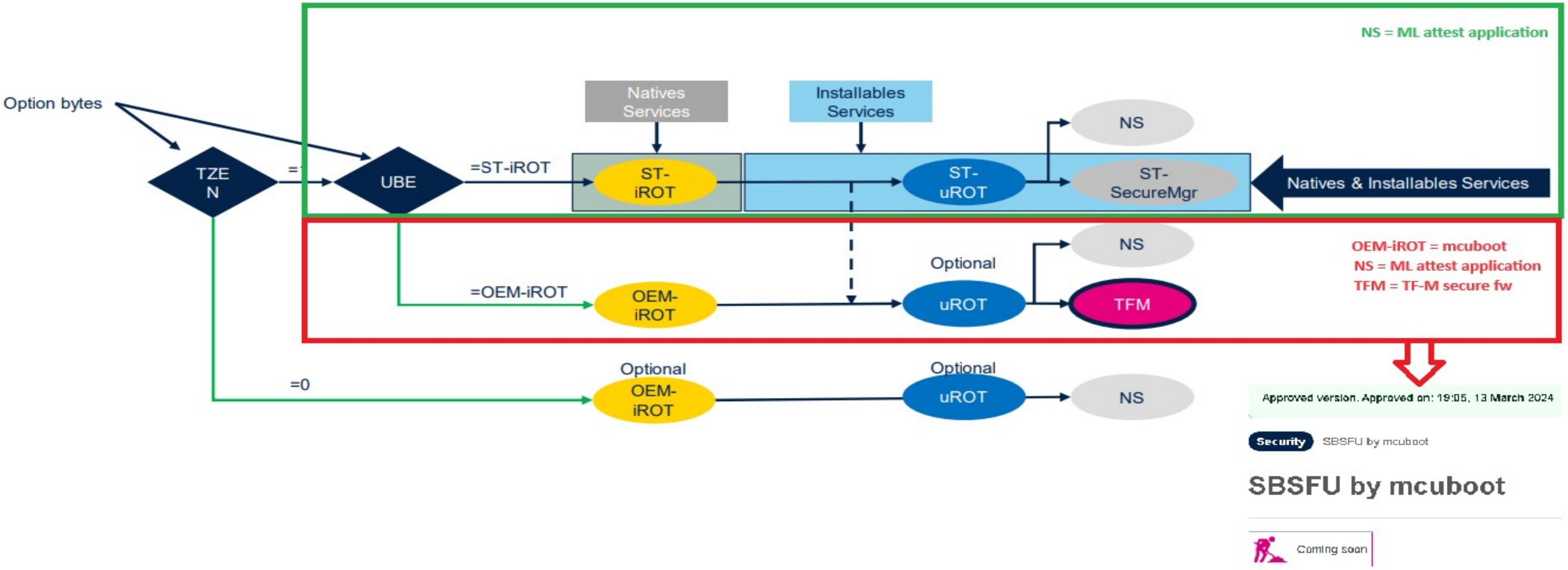
STM32TRUST ECOSYSTEM – SECURE MANAGER VS TF-M



ML MODEL ATTESTATION

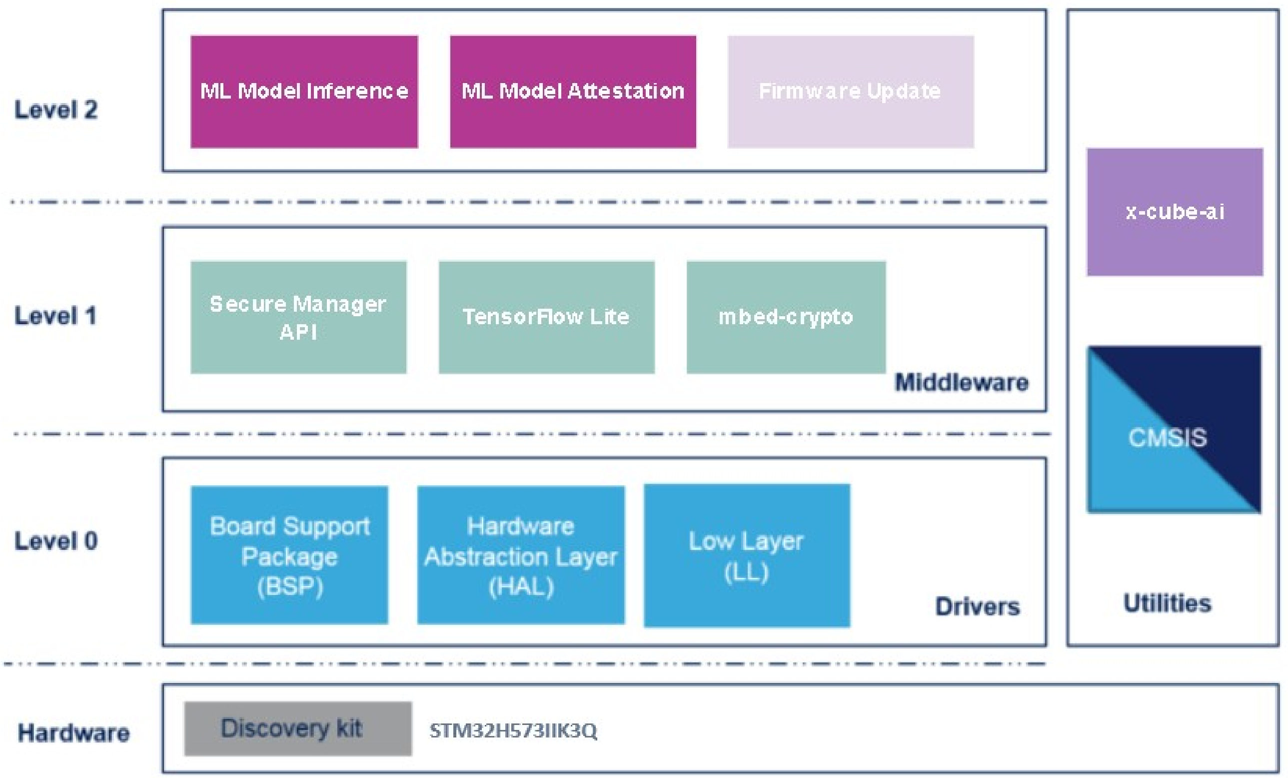
SECURE BOOT FLOW

- Secure Manager solution comes as an encrypted binary that includes the ST-uROT and SM
  - it is easy to provision and further to develop NS applications
  - secure module development kit is not available, under NDA
- TFM solution involves compiling and integrating different images (mcuboot, TFM-core, TFM-secure, TFM-nonsecure, TFM-loader)
  - not so straightforward, no support currently for STM32H5 series
  - flexible secure module development



ML MODEL ATTESTATION

SW ARCHITECTURE



ML MODEL ATTESTATION

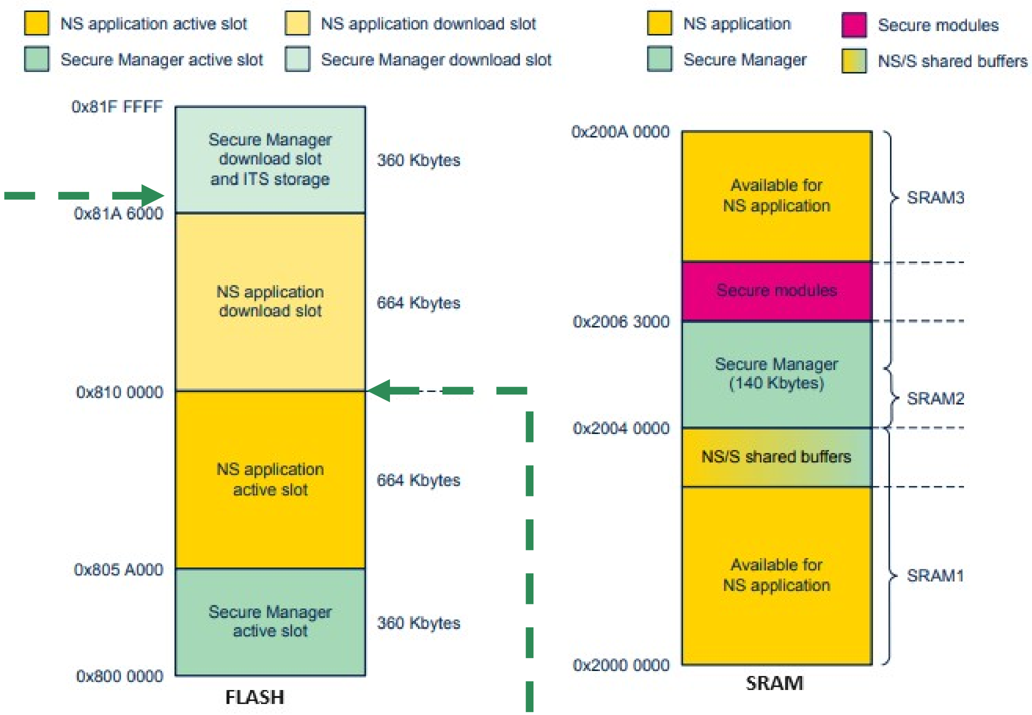
IMPLEMENTATION

- ML Model Training
- TFLite conversion
- Compute H(w)G — Compressed point format
- Create ITS blob — Add private ECC key, SECP_K1, 256b; H(w)G; Blob encrypted with ST key
- RoT Provisioning — Secure boot keys; Factory Internal Trusted Storage; Secure Manager keys; Flash layout; Option Bytes
- Flash encrypted SFI image
- Update application flash layout — Linker file changes; Blob header info

Prebuild

Compile → Encrypt and sign image → Flash blob

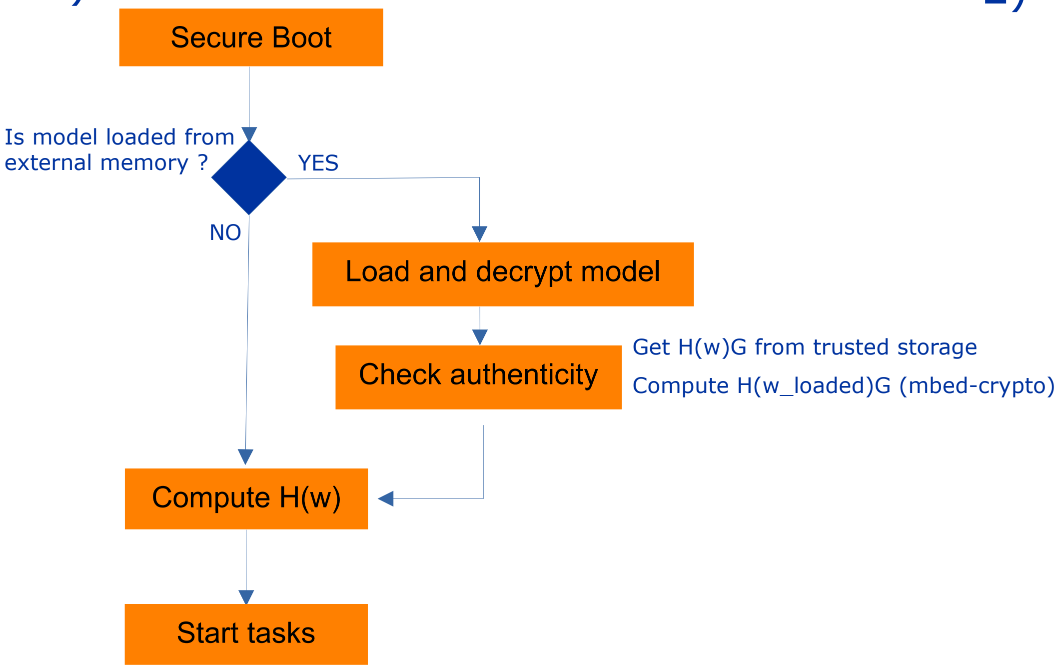
Postbuild

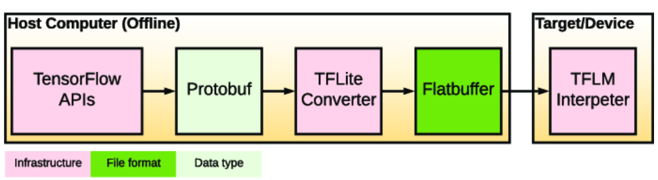


ML MODEL ATTESTATION

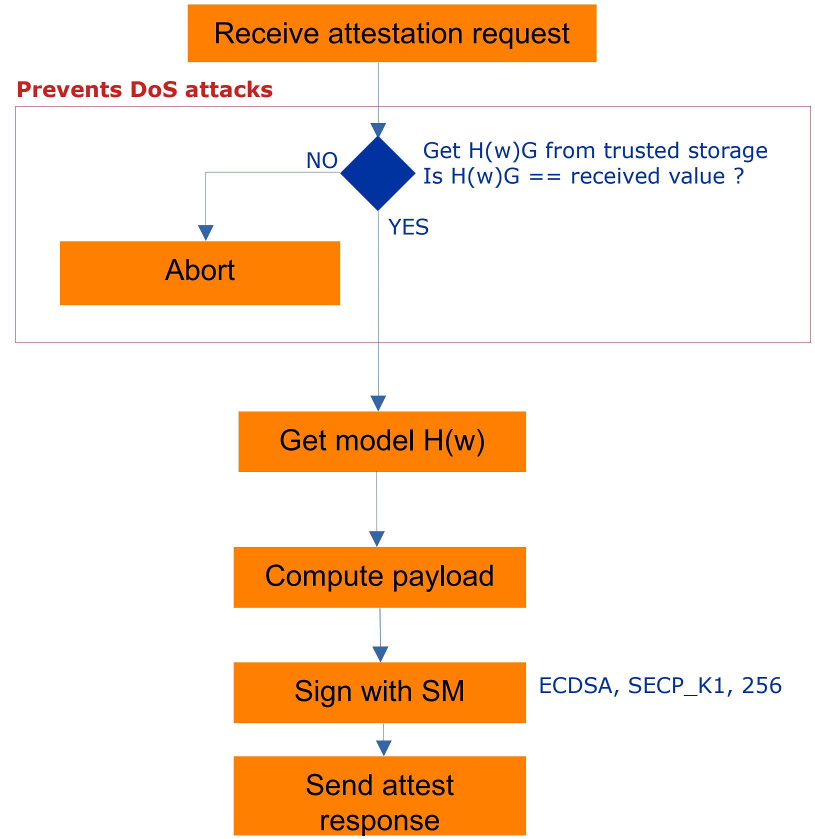
IMPLEMENTATION

1)





2)



ML MODEL ATTESTATION

MEMORY FOOTPRINT

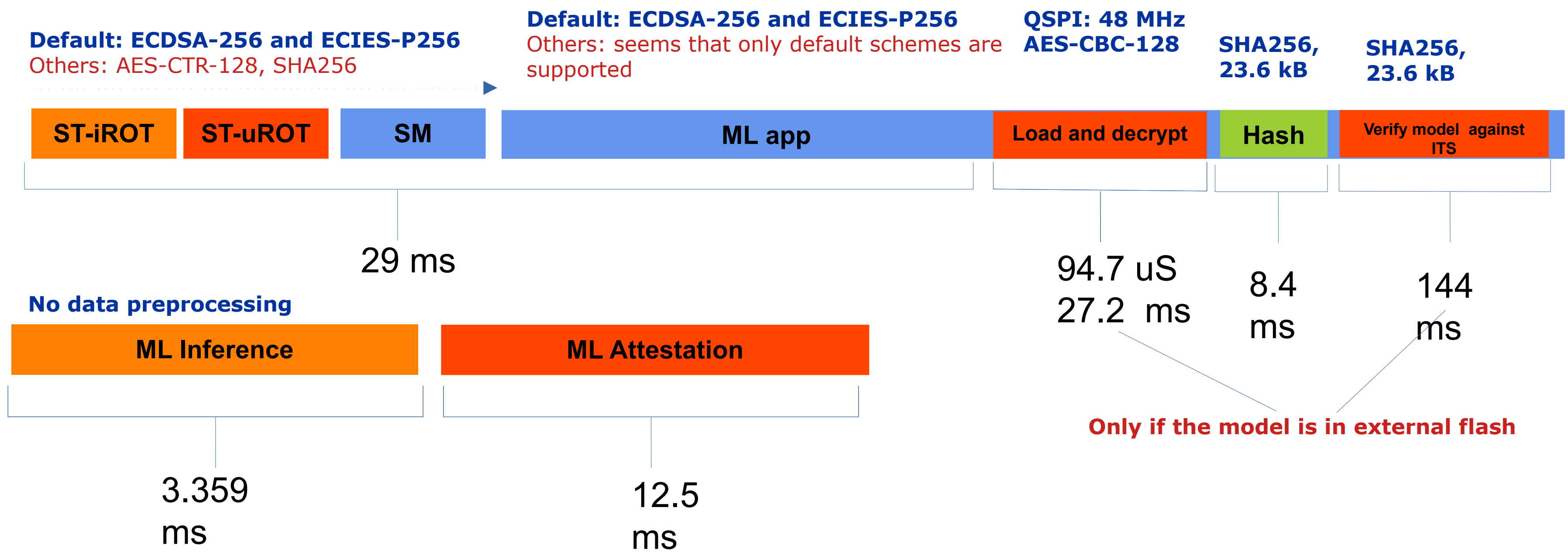
	Non-secure	Secure (SM)
Flash (text + RO data)	358 kB (TFLite 242 kB)	360 kB (1kB ITS included)
SRAM (data, bss)	20.7 kB (tensor area 16 kB)	140 kB + 16 kB NS/S area
MNIST ML Model	23.6 kB (either internal/external flash)	

- 283 kB of internal flash left
- 463 kB of SRAM left

- SM solution is imposing the two firmware slots approach for FW Update
- The TensorFlow Lite binary is ~1MB when all 125+ supported operators are linked (for 32-bit ARM builds), and less than 300KB when using only the operators needed
- For more complex models:
  - push ML model weights to external flash, load and decrypt at runtime
  - split text section between internal and external flash; load at boot in SRAM or leverage XiP

ML MODEL ATTESTATION

PERFORMANCE - TIMING



Non-Secure IRQ	NSPE is executing in thread mode, no secure module running	273 ns
Non-Secure IRQ	SPE is executing in thread mode	4.2 uS

Discussions & Final notes

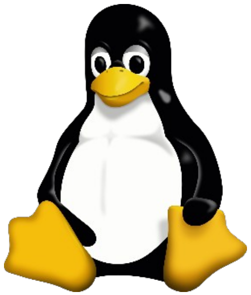
ARMV8 TRENDS

ARMv8 (TrustZone)

Concept 2005-2008

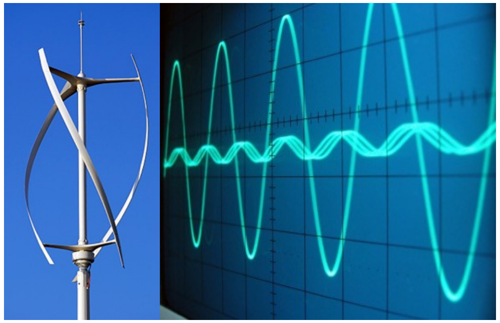
High performance

2012-2014 (64bit)
RPi, IMX (NXP), Sitara (Ti)
600 MHz – 2GHz
Flash + RAM ~ GB



Mid-range

2021-2022 (32bit)
ARM Cortex M33
120-250MHz
Flash + RAM ~ MB



Low-end

2021-2024 (32bit)
ARM Cortex M23
48MHz max
Flash + RAM < 1MB

ARMV8 TRENDS

NEW

Microchip ARM Cortex M23 + M33 (2021-2024)
→ PIC32CM LS00 & PIC32CM LS60 series @48MHz
→ PIC32CK SG01/SG00/GC01/GC00 series @120MHz
→ New devices are added

Microchip MPLAB X support for crypto-authenticator modules
→ ATECC608: secure boot
→ ECC204/6: elliptic curve
→ (AT)SHA204/5/6
→ TA100 and TA101: support for TLS
→ Improved support for secure boot, crypto, TLS, etc.

NEW

Renesas ARM Cortex M23 (December 2023) @48MHz

GigaDevice (China) ARM Cortex M33:
→ GD32F5 series @200MHz, Embedded World Nuremberg April 2024

PLATFORM SELECTION

1: Application type
High, mid or low-end

2: Required peripherals
#Sercoms, ethernet,
wireless, storage, etc,

3: Required security features
Sensitive data? Secure boot

Performance selection shall be similar as before ARMv8

Required peripherals, IO speed, etc. is still very important

MCU selection vs. selection of programming tools

MCU part of family catalog -> upgradability

Secure mechanisms by design!
→ encryption, hashing, secure boot, RoT, etc.

DISCUSSIONS & FINALE NOTES

END PRODUCT (SOFTWARE)

Use case 1 — Open-source framework for TrustZone assisted Hardware (applicable for Microchip devices)

Use case 2 — Solution for trustworthiness on AI model versioning & execution

PUBLICATION

Journal:

Segers, L.; Talebi, B.; da Silva, B.; Touhafi, A.; Braeken, A. Trustworthy Environmental Monitoring Using Hardware-Assisted Security Mechanisms. Sensors 2024, 24, 4720.
https://doi.org/10.3390/s24144720

FOLLOW-UP PROJECTS

ENACT – ENVIRONMENTAL EFFECT ON HEATHCARE AND WELLBEING AND ACTIVE INTERVENTIONS

- **Content:** The overall objective of ENACT is twofold: first, to derive a model assessing the exposomic risk (risk score based on poli-environmental exposures) of hospitalization for acute vascular and non-vascular non-communicable diseases across different populations and locales; second, to translate this risk from population to the individual level, predicting the risk of developing preclinical asymptomatic stages of disease.

- **Contribution from TrustIoT**: Extension of the environmental sensing platform developed in the VUB use cases focus on concentrations and qualitative aspects of pollutants, noise, light (UV) and various types of radiations.

- **Belgian partners:**
  - Vrije Universiteit Brussel
  - UzBrussel
  - Alliance for IoT and Edge Computing Innovation IVZW (AIOTI)

COMBINE-IOT – COMBINING RECENT TRENDS IN

- **Content**:
  - Evaluation of recent communication, ranging and positioning techniques using UWB, BLE and Wi-Fi 6
  - Evaluation of Matter for application interoperability

- **Contribution of TrustedIoT**: selection and use of Trustzone-enabled microcontrollers for secure set-up.

- **User committee**: Lumency, Verhaert, OneSpan, Televic, GemOne, Commeto, Barco, Niko, Citymesh, LSEC, Qorvo, Sealution, Userfull, Callitrix, In The Pocket, etc.

Thank you for your attention

Laurent Segers - laurent.segers@vub.be
Baciu Vlad-Eusebiu - vlad-eusebiu.baciu@vub.be
An Braeken - an.braeken@vub.be
Bruno da Silva - bruno.da.silva@vub.be
Abdellah Touhafi - abdellah.touhafi@vub.be