Aalto University

UCI

TECHNISCHE UNIVERSITÄT DARMSTADT

TRUSTONIC

ASSURED: Architecture for Secure Software Update of Realistic Embedded Devices

N. Asokan[1], Thomas Nyman[1,2], Norrathep Rattanavipanon[3], Ahmad-Reza Sadeghi[4], Gene Tsudik[3]

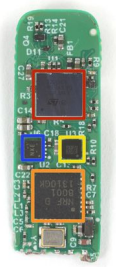
[1]Aalto University, [2] Trustonic, [3] University of California, Irvine, [4] Technische Universität Darmstadt, Darmstadt

Software Update Landscape in IoT

- **Adoption of security technology in IoT is lagging**
  - >80% of respondents in 2017 IoT Survey do not use Over-the-Air updates!
- **Even Secure by design devices need software updates**
  - Assumptions may change after deployment
  - New features may be introduced
- **Existing update mechanisms not suitable for tiny devices**
  - Solutions for microcontroller class devices ad hoc, often insecure

Eclipse Foundation IoT Developer Trends Survey 2017

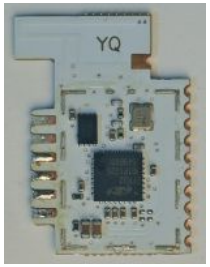
Internet of Resource Constrained Things



ARM Cortex-M3
@ 32 MHz

Wireless-enabled wearable activity tracker

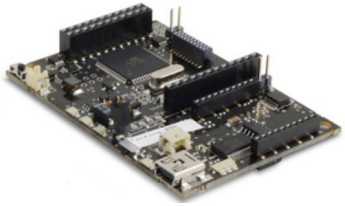
https://en.wikipedia.org/wiki/Fitbit (MorePix)
https://www.ifixit.com/Teardown/Fitbit+Flex+Teardown/16050



ARM Cortex-M4
@ 40MHz

Remote-controlled consumer smart lighting platform

http://www.ikea.com/se/sv/catalog/categories/departments/lighting/36812/
https://www.heise.de/make/artikel/Das-steckt-in-Ikea-Tradfri-3597295.html



ATmega1281
@ 14.74 MHz

Wireless vehicle-presence sensor with 7 to 10 years of battery life

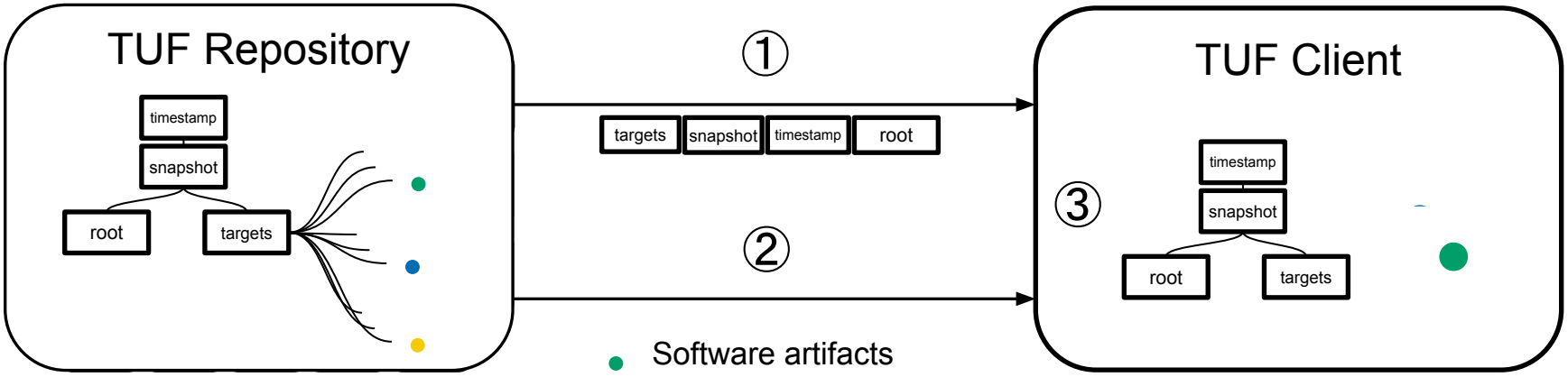
http://embedded-computing.com/articles/sensor-enabled-nodes-support-the-iot-for-smart-buildings-and-smart-transport/
http://www.libelium.com/products/waspmote/hardware/

Challenges for IoT software update

- **Unreasonable expectations of Device capabilities**
  - Update policy decisions deferred fully to **Device**
  - Liberal use of cryptography infeasible for tiny **Device**
- **Direct interaction between Device and OEM**
  - Hinders broadcast deployment of updates
- **Lack of ability to validate correct update installation**

Example: The Update Framework (TUF)

General purpose update framework for software packages



① Fetch repository metadata

② Fetch software artifacts

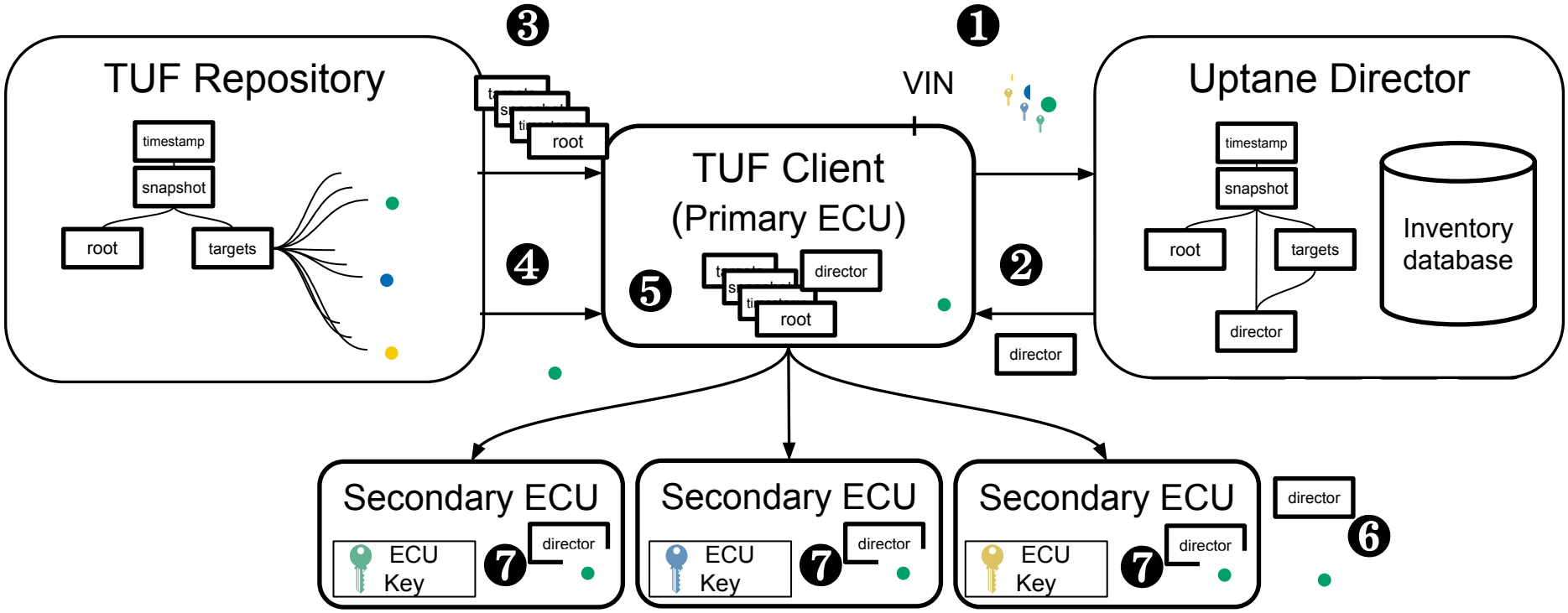
③ Verify all metadata and software artifacts

Used by companies such as Docker, DigitalOcean, Cloudflare, and VMware

J. Samuel et al. "Survivable key compromise in software update systems" in ACM CCS 2010

Example: Uptane

Variant of TUF for automotive ECU updates



❶ Report *version manifest*

❷ Sign and send *director metadata*

❸ Fetch repository metadata

❹ Fetch software artifacts

❺ Verify all metadata and software artifacts

❻ Broadcast to all ECUs

❼ Verify director metadata

[1] K. Kuppusamy et al. "Uptane: Securing software updates for automobiles" in Escar Europe, 2016

Stakeholders

- **Original Equipment Manufacturer (*OEM*)**
  - Produces devices, firmware and subsequent updates
  - Can securely provision cryptographic keys during manufacturing
- **Software Distributor (*Distributor*)**
  - Provides infrastructure and logistics for update distribution
- **Domain Controller (*Controller*)**
  - Responsible for configuration, upkeep and operation of end devices
  - Administrative domain may be physical proximity or organizational
- **Connected Device (*Device*)**
  - End-device that receives the updates
  - Strict resource limitations in memory, storage and processing power

Objectives

1. ***End-to-end security***
   - Update originates from ***OEM*** and is intended for ***Device***
2. ***Update Authorization***
   - Update authorized by ***OEM*** and ***Controller***
3. ***Attestation of Update Installation***
   - Verifiable proof of whether update succeeded or not

4. ***Protect code & secret keys***
5. ***Minimize burden for Device***

ASSURED Overview

Extends update scheme with *Constraints* and *Authorizations*

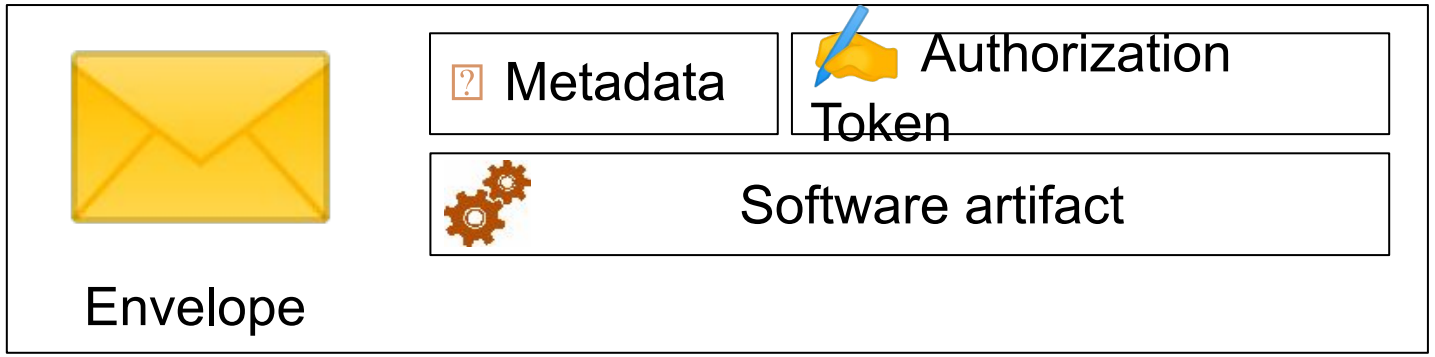
- ***Constraints* allow limiting update to set of devices**
  - e.g. device model and/or unique device identifier

- ***Authorizations* validated by *Device* before applying update**
  - encode *Constraints*, cryptographic hash and size of software
  - signed by OEM or Controller

$$Auth_{SA} = [hash(SA), size(SA), C, Sig(K_{OEM}, hash(SA)||size(SA)||C)]$$

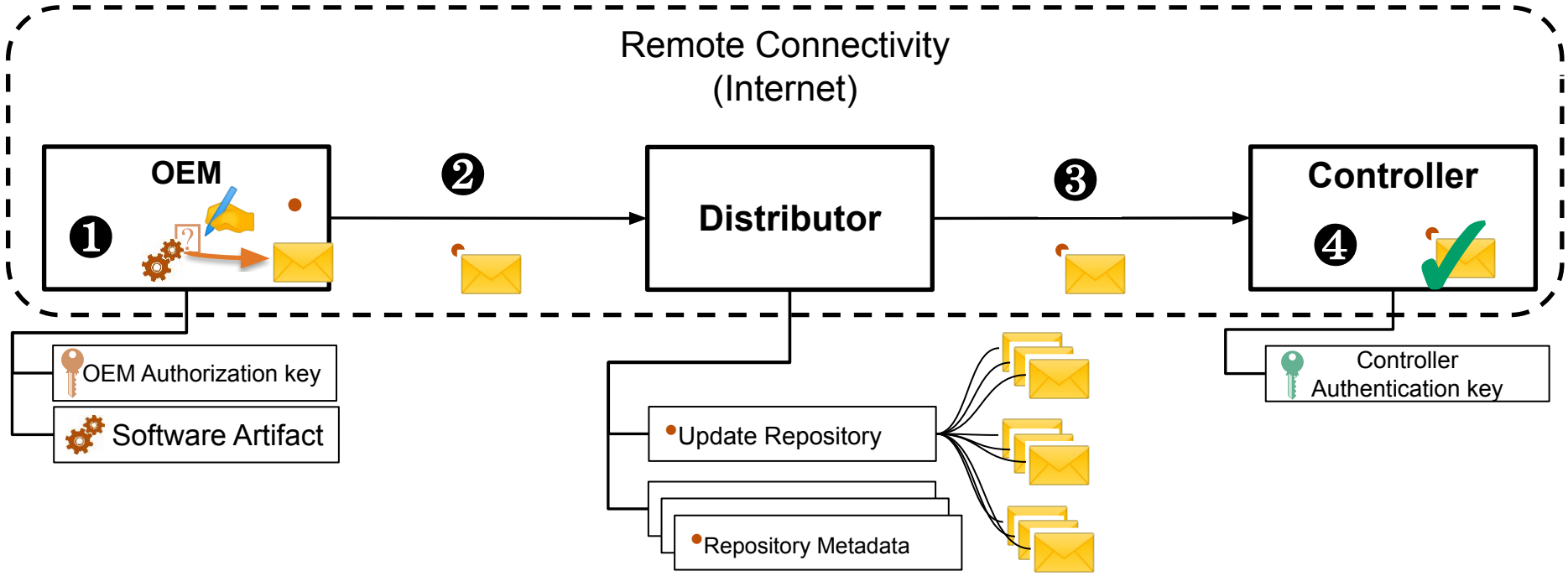
ASSURED Envelopes

Everything needed by *Device* to decide whether to install update



ASSURED Sequence of Events

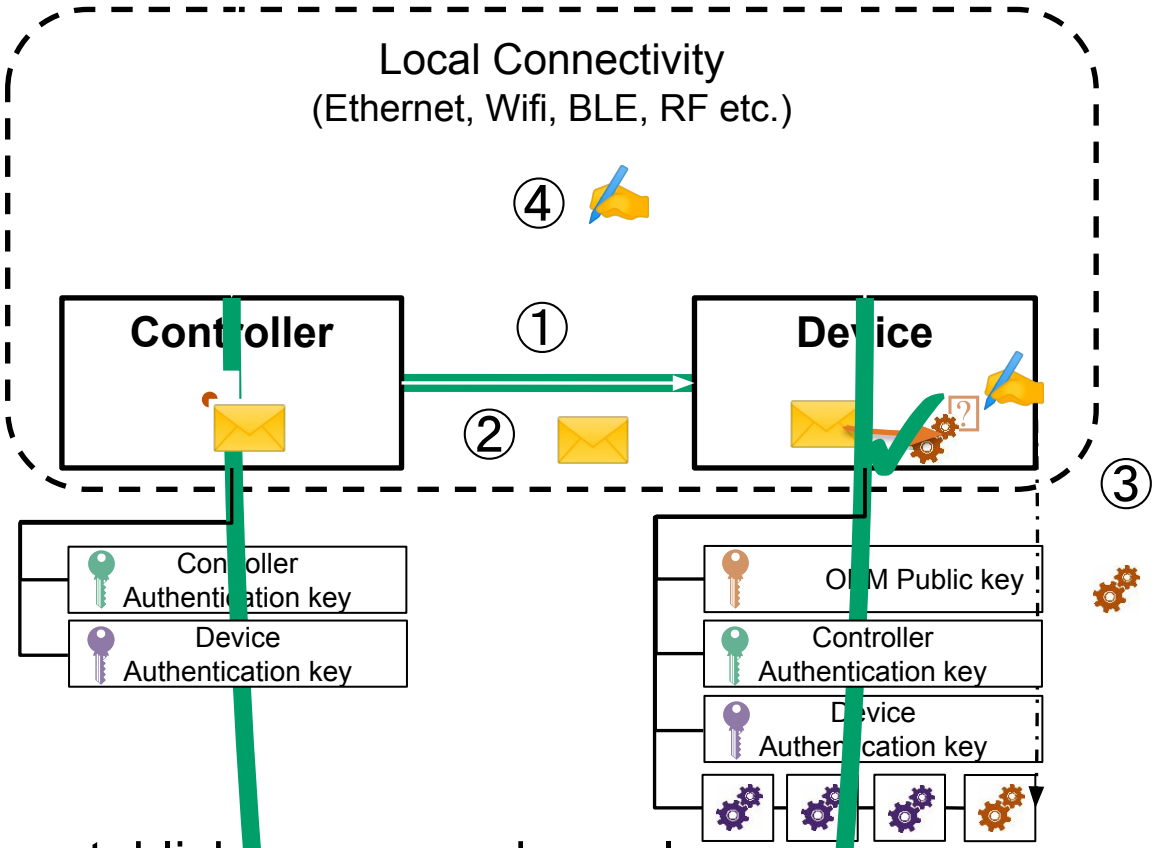
OEM → Distributor → Controller



1. **OEM** creates *Authorization*, *Repository Metadata* • and *Envelope*
2. **OEM** uploads *Repository Metadata* • and *Envelope* to **Distributor**
3. **Controller** fetches latest *Repository Metadata* • and *Envelope(s)*
3. **Controller** validates *Repository Metadata* • and *Envelope(s)*

ASSURED: Sequence of Events

Controller → Device



① **Controller** establishes secure channel

② **Controller** transmits *Envelope*

③ **Device** validates *Authorization* and installs Software Artifact

④ **Controller** attests **Device** state

ASSURED PoC Implementation

ASSURED PoCs implemented on two commodity platforms

Platform	I.MX6-SabreLite	V2M-MPS2+
Processor	Cortex-A9	Cortex-M23
Root of Trust	ROM Bootloader	TrustZone Bootloader
Isolated execution	seL4 Microkernel	TrustZone-M
Signature Algorithm	ED25519	ED25519
Attestation	HYDRA	Binary attestation

TrustZone technology for ARMv8-M Architecture Version 1.1

K. Eldefrawy et al. "HYDRA: hybrid design for remote attestation (using a formally verified microkernel), WiSec 2017

Evaluation: Meeting Security Objectives

1. ***End-to-end security***
   - *Authorization token* incl. *Constraints* and hash of **Software Artifact** signed by **OEM** and validated by **Device**

2. ***Update Authorization***
   - **OEM** authorization through *Authorization token*
   - ***Controller*** authorization either through secure channel or *Authorization token* signed by **Controller**

3. ***Attestation of Update Installation***
   - Provides verifiable proof of whether update succeeded or not

Evaluation: Verification Time

I.MX6-SabreLite @ 800MHz

	TUF	ASSURED		
		Expl. Auth.	Impl. Auth.	Total
Verification Time (ms)	**14.57**	2.46	0.1	**2.56**
Metadata Size (bytes)	940	136	52	188

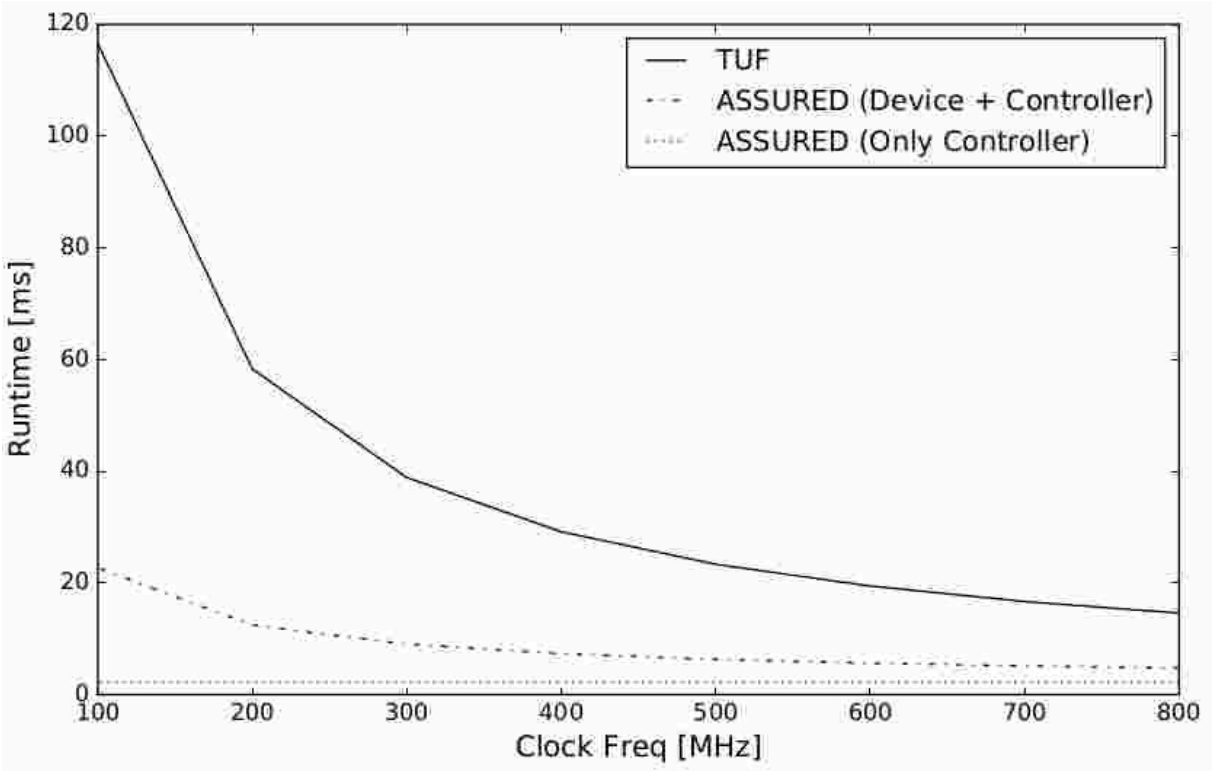
82% reduction in verification time compared to TUF

V2M-MPS2+ (Cortex-M23) @ 25MHz

	TUF	ASSURED		
		Expl. Auth.	Impl. Auth.	Total
Verification Time (ms)	**10723**	1816	*n/a*	**1816**
Metadata Size (bytes)	940	136	*n/a*	136

83% reduction in verification time compared to TUF

Evaluation: Verification Time (cont.)



Comparison of verification time on I.MX6-SabreLite for variable clock frequencies.

Summary

Identified essential roles in *IoT Update Ecosystem*

- Show why existing secure update methods inadequate for IoT

Secure Firmware Update Framework: ASSURED

Proof-of-Concept implementations for two commodity platforms

- I.MX6-SabreLite (seL4), Cortex-M23 (TrustZone-M)

SELIoT: SEcuring Lifecycle of Internet of Things

https://ssg.aalto.fi/research/projects/seliot-project/

Backup Slides

Internet of Resource Constrained Things

Wireless-enabled wearable activity tracker

https://en.wikipedia.org/wiki/Fitbit *(MorePix)*

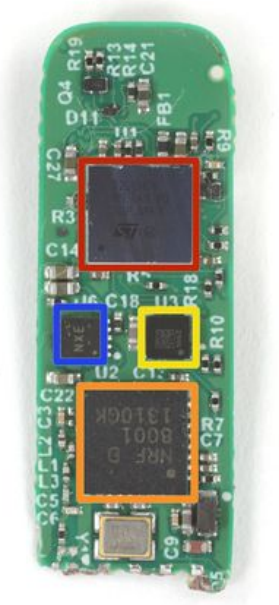
Remote-controlled consumer smart lighting platform

http://www.ikea.com/se/sv/catalog/categories/departments/lighting/36812/

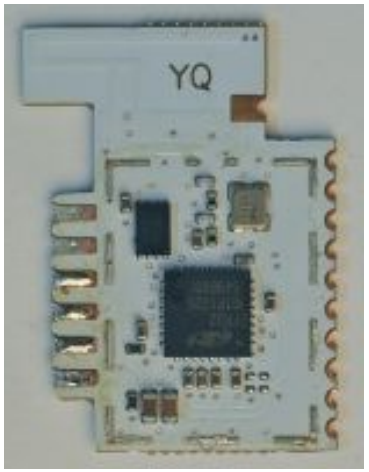
Wireless vehicle-presence sensor with 7 to 10 years of battery life

http://embedded-computing.com/articles/sensor-enabled-nodes-support-the-iot-for-smart-buildings-and-smart-transport/

Workhorses for small IoT devices

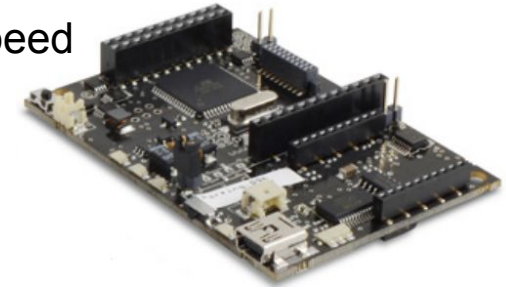


ARM Cortex-M3
Up to **32 MHz** Clock Speed
Up to **16 kB** RAM
Up to **4kB** EEPROM
Up to **128 kB** Flash
Bluetooth LE



ARM Cortex-M4 + Floating Point Unit
Up to **40 MHz** Clock Speed
Up to **256 kB** RAM
Up to **1024 kB** Flash
ZigBee and **Thread** Radio (6LoWPAN)
Hardware Crypto Accelerator w/
AES-256/128, ECC, SHA-1, SHA-2

ATmega1281
14.74 MHz Clock Speed
8 kB SRAM
4 kB EEPROM
128 kB Flash
ZigBee (external)



https://www.ifixit.com/Teardown/Fitbit+Flex+Teardown/16050
http://www.st.com/en/microcontrollers/stm32l151c6.html
http://www.libelium.com/products/waspmote/hardware/

http://www.libelium.com/v11-files/documentation/waspmote/smart-parking-sensor-board_eng.pdf
https://www.heise.de/make/artikel/Das-steckt-in-Ikea-Tradfri-3597295.html
https://www.silabs.com/products/wireless/mesh-networking/efr32mg-mighty-gecko-zigbee-thread-soc

Characteristics of a *secure-by-design* IoT system

- **Root-of-Trust based in hardware**
  - foundation from which trust in integrity and security can be established
  - immutable except by authorized entities

- **Crypto-acceleration**
  - For securing remote communications

- **Protection of security critical components**
  - A Secure boot loader
  - Secret keys
  - Flash programming support
  - High value assets (personally identifiable information, authorization keys etc.)

TrustZone-M

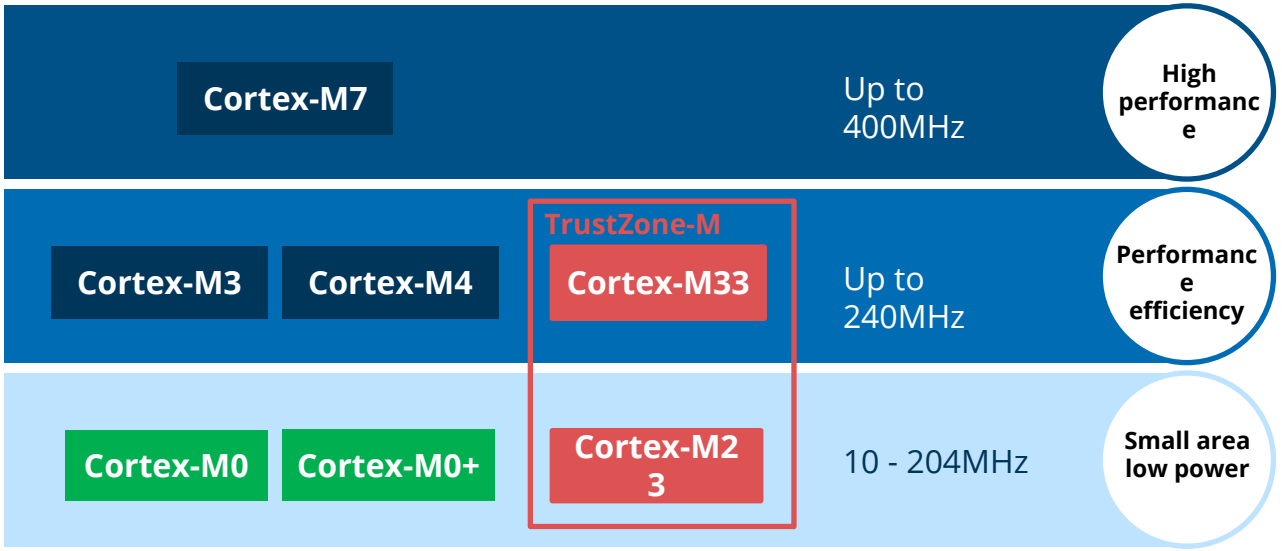
Security through isolation for IoT devices

- **Hardware-enforced isolation between Trusted and Non-trusted software**
  - Enabled by secure architectures such as **SMART**, **TrustLite**, **TyTan** and **TrustZone-M**

- **Trusted and Non-trusted software can interact, but Non-trusted code only access critical resources through APIs in Secure software**
  - access to Secure services subject to authentication
  - reduces attack surface of critical assets
  - vulnerabilities in Non-trusted software do not compromise entire system

SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. ISOC Symposium on Network and Distributed System Security (NDSS). 2012.
TrustLite: a Security Architecture for Tiny Embedded Devices. European Conference on Computer Systems (EuroSys). 2014.
TyTAN: Tiny trust anchor for tiny devices. Design Automation Conference (DAC). 2015
TrustZone technology for ARMv8-M Architecture Version 1.1

Working horses for small IoT devices



TrustZone-M: chip-level hardware security architecture for ARM MCUs

Design characteristics for TrustZone-M

- **Non-secure software accesses Secure APIs with standard function calls**
  - non-secure code can only call Secure functions using valid entry points
- **Secure software can call Non-secure functions**
  - allows protected middleware on Secure side to access Non-secure side device driver
- **Non-secure interrupts requests served while Secure code executes**
  - interrupt handlers remain programmable in C, minimal impact on interrupt latency
  - Non-secure interrupt handlers are prevented from snooping Secure operation data
- **Processor starts in Secure state by default**
  - enables root-of-trust implementations such as Secure boot

Design characteristics for TrustZone-M (cont.)

- **Low switching overhead in cross security domain calls**
  - only one extra instruction (`SG`) when calling from the Non-secure to the Secure domain
  - only a few extra clock cycles when calling from the Secure state to Non-secure functions

- **No separate register banks for Secure and Non-secure states**
  - significant for energy efficiency and processor area