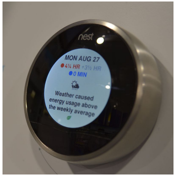
Sancus 2.0: Open-Source Trusted Computing for the IoT

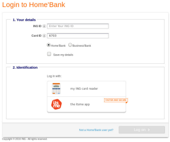
Jan Tobias Mühlberg
jantobias.muehlberg@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

FOSDEM, Brussels, February 2018

Joint work with Job Noorman, Jo Van Bulck, Frank Piessens, Pieter Maene, Ingrid Verbauwhede and many others.

Security

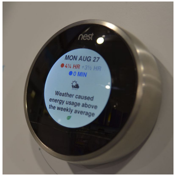




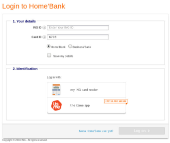
Source of images 1, 2, 3: `https://en.wikipedia.org/`

Security

1. **Understand the system.**
   - Context, hardware, software, data, users, use cases, etc.










Source of images 1, 2, 3: `https://en.wikipedia.org/`

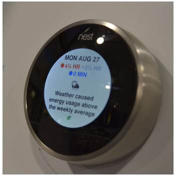
Security

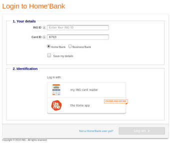
❶ **Understand the system.**

- Context, hardware, software, data, users, use cases, etc.

❷ **Understand the security requirements.**

- Requirements are not features!
- “Only authenticated users can do X. Two-factor authentication is required for all users. All X are logged, detailing time, user and properties of X.”

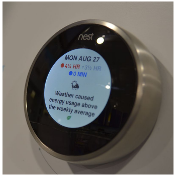


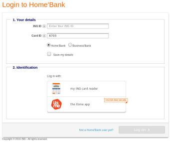


Source of images 1, 2, 3: `https://en.wikipedia.org/`

Security

1. **Understand the system.**
   - Context, hardware, software, data, users, use cases, etc.
2. **Understand the security requirements.**
   - Requirements are not features!
   - “Only authenticated users can do X. Two-factor authentication is required for all users. All X are logged, detailing time, user and properties of X.”
3. **Understand the attacker.**
   - “Attackers can listen to all communication, can drop, reorder or replay messages, may compromise Y% of the system, can’t break crypto.”

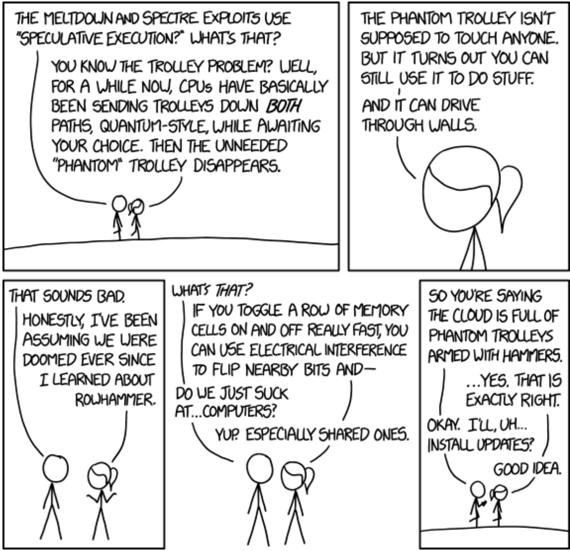




Source of images 1, 2, 3: `https://en.wikipedia.org/`

Security

“New zero-day vulnerability: In addition to rowhammer, it turns out lots of servers are vulnerable to regular hammers, too.”

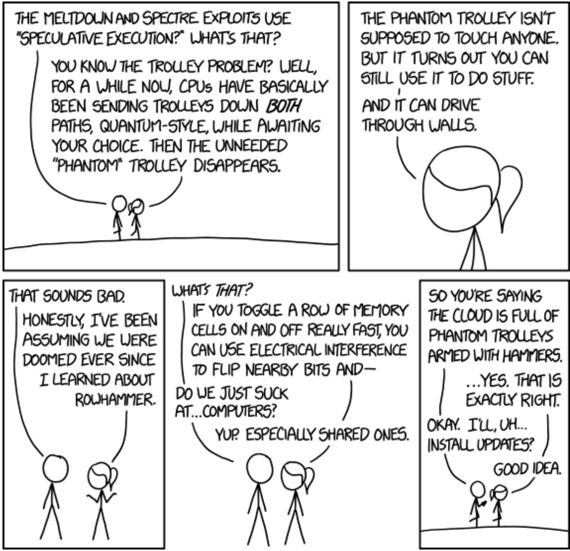


Source: `https://xkcd.com/1938/`

Security

"New zero-day vulnerability: In addition to rowhammer, it turns out lots of servers are vulnerable to regular hammers, too."

1. **Understand the system.**
2. **Understand the security requirements.**
3. **Understand the attacker.**

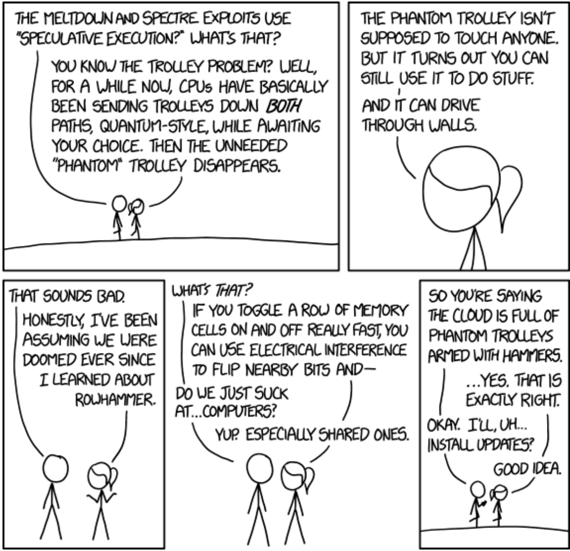


Source: `https://xkcd.com/1938/`

Security

"New zero-day vulnerability: In addition to rowhammer, it turns out lots of servers are vulnerable to regular hammers, too."

1. **Understand the system.**
2. **Understand the security requirements.**
3. **Understand the attacker.**
4. **Understand and embrace change!**
   - Discovery of vulnerabilities
   - Different understanding of the system
   - New (functional|security) requirements
   - New attacks, different attackers



Source: `https://xkcd.com/1938/`

Trusted Computing

Source: `https://en.wikipedia.org/wiki/Trusted_Computing`

Trusted Computing

According to the *Trusted Computing Group*

Protect computing infrastructure at end points;

Hardware extensions to enforce specific behaviour and to provide cryptographic capabilities, protecting against unauthorised change and attacks

Source: `https://en.wikipedia.org/wiki/Trusted_Computing`

Trusted Computing

According to the *Trusted Computing Group*

Protect computing infrastructure at end points;
Hardware extensions to enforce specific behaviour and to provide cryptographic capabilities, protecting against unauthorised change and attacks

- **Endorsement Key**, EK Certificate, Platform Certificate: Unique private key that never leaves the hardware, authenticate device identity
- **Memory curtaining:** provide isolation of sensitive areas of memory
- **Sealed storage:** Bind data to specific device or software
- **Remote attestation:** authenticate hardware and software configuration to a remote host
- **Trusted third party** as an intermediary to provide (ano|pseudo)nymity

Source: `https://en.wikipedia.org/wiki/Trusted_Computing`

Trusted Computing

According to the *Trusted Computing Group*
Protect computing infrastructure at end points;
Hardware extensions to enforce specific behaviour and to provide cryptographic capabilities, protecting against unauthorised change and attacks

- **Endorsement Key**, EK Certificate, Platform Certificate: Unique private key that never leaves the hardware, authenticate device identity
- **Memory curtaining:** provide isolation of sensitive areas of memory
- **Sealed storage:** Bind data to specific device or software
- **Remote attestation:** authenticate hardware and software configuration to a remote host
- **Trusted third party** as an intermediary to provide (ano|pseudo)nymity

In practice: different architectures, subset of the above features, additions such as "enclaved" execution, memory encryption or secure I/O capabilities

Source: `https://en.wikipedia.org/wiki/Trusted_Computing`

Trusted Computing

According to the *Trusted Compu*
Protect computing infrastructure at
Hardware extensions to enforce spe
capabilities, protecting against unau

- **Endorsement Key**, EK Certific
  that never leaves the hardware
- **Memory curtaining:** provide is
- **Sealed storage:** Bind data to s
- **Remote attestation:** authentic
  remote host
- **Trusted third party** as an inter

In practice: different architectures,
as "enclaved" execution, memory e

Possible Applications

Digital rights management [edit]

Trusted Computing would allow companies to create a digital rights management (DRM
though not impossible. An example is downloading a music file. Sealed storage could b
with an unauthorized player or computer. Remote attestation could be used to authoriz
record company's rules. The music would be played from curtained memory, which wo
copy of the file while it is playing, and secure I/O would prevent capturing what is being
system would require either manipulation of the computer's hardware, capturing the a
recording device or a microphone, or breaking the security of the system.

New business models for use of software (services) over Internet may be boosted by the
one could base a business model on renting programs for a specific time periods or "pa
download a music file which could only be played a certain number of times before it b
only within a certain time period.

Preventing cheating in online games [edit]

Trusted Computing could be used to combat cheating in online games. Some players m
advantages in the game; remote attestation, secure I/O and memory curtaining could b
a server were running an unmodified copy of the software.[18]

Verification of remote computation for grid computing [edit]

Trusted Computing could be used to guarantee participants in a grid computing system
they claim to be instead of forging them. This would allow large scale simulations to be
redundant computations to guarantee malicious hosts are not undermining the results

Source: `https://en.wikipedia.org/wiki/Trusted_Computing`

Trusted Computing

According to *Richard Stallman*

Treacherous Computing: "The technical idea underlying treacherous computing is that the computer includes a digital encryption and signature device, and the keys are kept secret from you. Proprietary programs will use this device to control which other programs you can run, which documents or data you can access, and what programs you can pass them to. These programs will continually download new authorisation rules through the Internet, and impose those rules automatically on your work."

Source: `https://www.gnu.org/philosophy/can-you-trust.html`

Trusted Computing

According to *Richard Stallman*

Treacherous Computing: “The technical idea underlying treacherous computing is that the computer includes a digital encryption and signature device, and the keys are kept secret from you. Proprietary programs will use this device to control which other programs you can run, which documents or data you can access, and what programs you can pass them to. These programs will continually download new authorisation rules through the Internet, and impose those rules automatically on your work.”

In the light of recent incidents...

- **Buggy software:** think of OpenSSL’s Heartbleed in an enclave
- **Side channels:** timing, caching, speculative execution, etc.
- **Buggy system:** CPUs, peripherals, firmware (Broadpwn, Intel ME, Meltdown)
- **Malicious intent:** Backdoors, ransomware, etc.

Source: `https://www.gnu.org/philosophy/can-you-trust.html`

Trusted Computing (and why Sancus?)

Good design practice for trusted computing?
Good use cases for trusted computing?

- non-invasive, understandable, measurably secure
- stuff that matters: critical applications, critical infrastructure, embedded

Source: `https://twitter.com/MelissaKaulfuss/status/804209991510937600?s=09`

Trusted Computing (and why Sancus?)

Good design practice for trusted computing?
Good use cases for trusted computing?

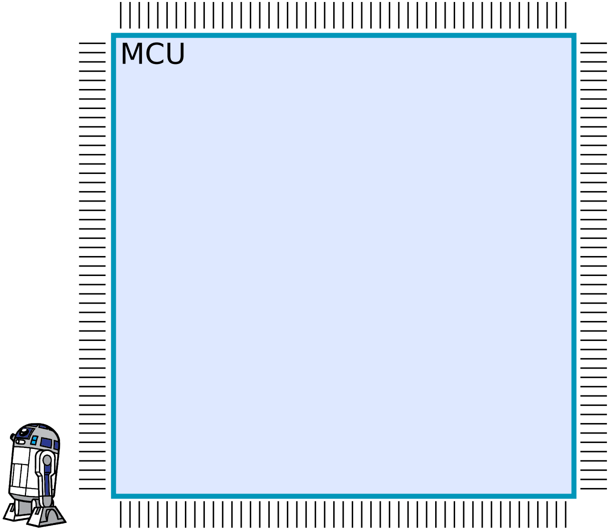
- non-invasive, understandable, measurably secure
- stuff that matters: critical applications, critical infrastructure, embedded

Don't restrict the user but enable them, convince them to trust.
Build to validate, invite to crutinize: hardware and software.
Build upon well-understood OSS building blocks: hardware, crypto, compilers, OS, libs
Divide and conquer: memory curtaining and isolation make validation easier

Source: `https://twitter.com/MelissaKaulfuss/status/804209915109376000?s=09`

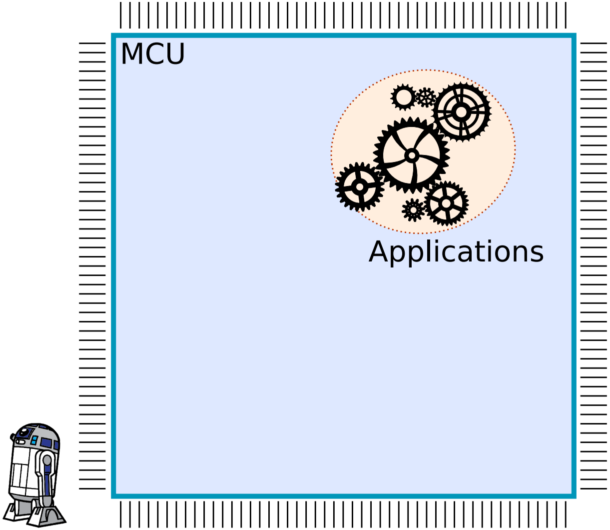
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality



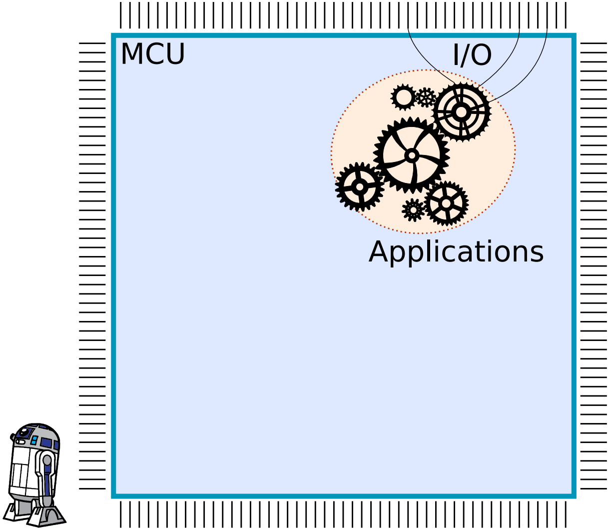
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality



Isolation and Attestation on Light-Weight MCUs

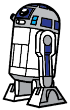
Many microcontrollers feature little security functionality

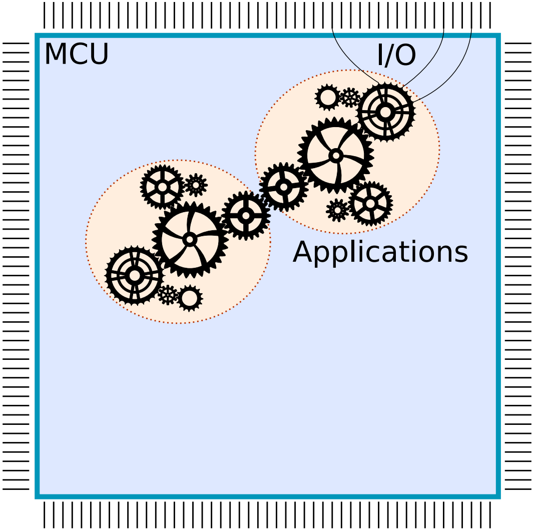


Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space

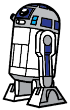


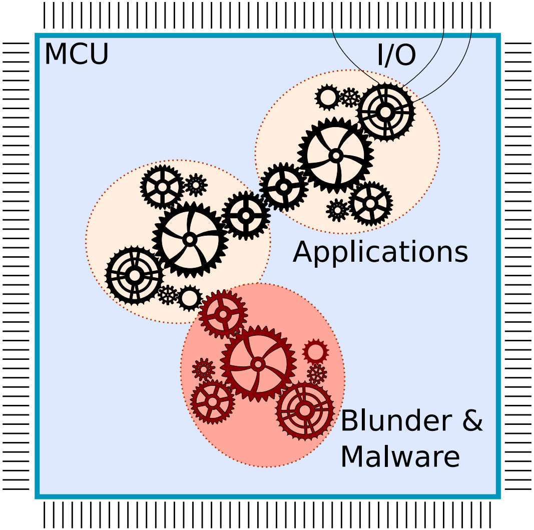


Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space
- Boundaries between applications are not enforced

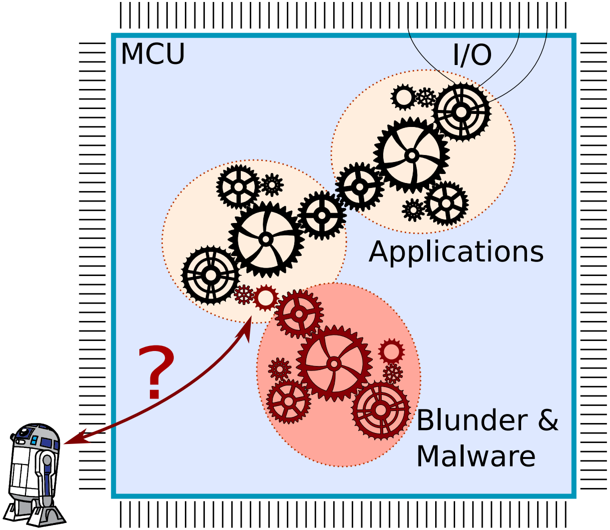




Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space
- Boundaries between applications are not enforced
- Integrity? Confidentiality? Authenticity?



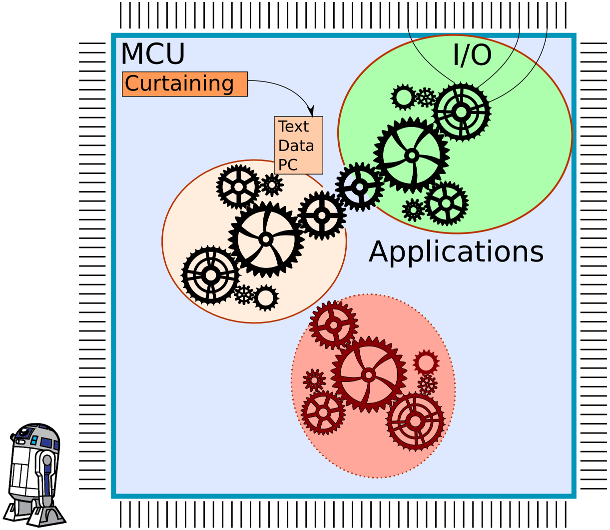
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space
- Boundaries between applications are not enforced
- Integrity? Confidentiality? Authenticity?

Trusted Computing aims to fix that:

- Strong isolation, restrictive interfaces, exclusive I/O



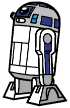
Isolation and Attestation on Light-Weight MCUs

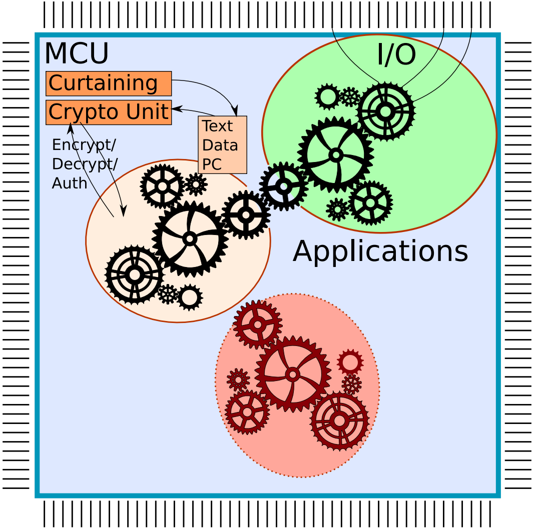
Many microcontrollers feature little security functionality

- Applications share address space
- Boundaries between applications are not enforced
- Integrity? Confidentiality? Authenticity?

Trusted Computing aims to fix that:

- Strong isolation, restrictive interfaces, exclusive I/O





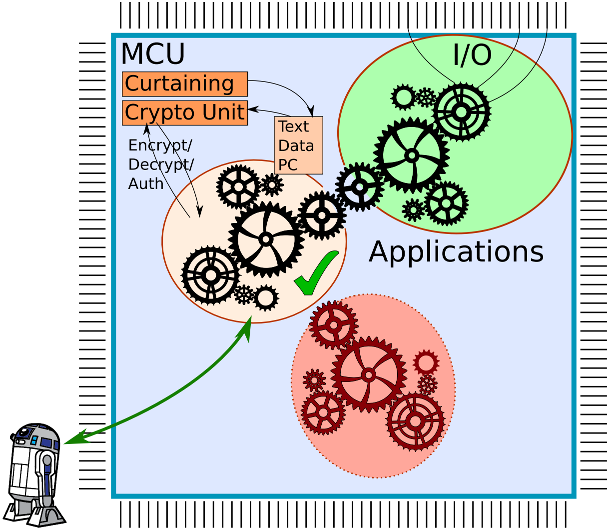
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space
- Boundaries between applications are not enforced
- Integrity? Confidentiality? Authenticity?

Trusted Computing aims to fix that:

- Strong isolation, restrictive interfaces, exclusive I/O
- Built-in cryptography and (remote) attestation



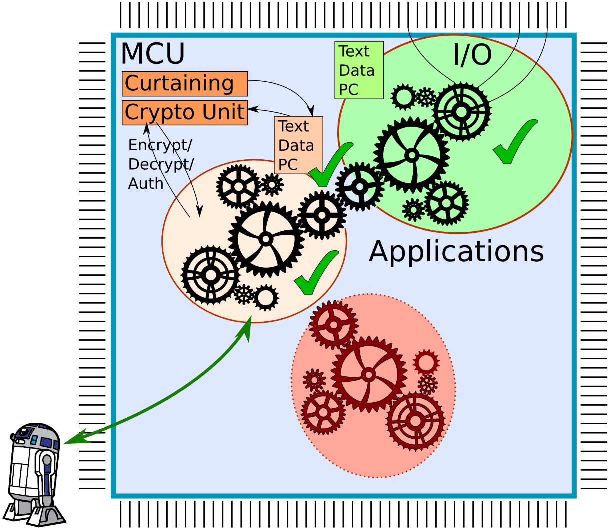
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space
- Boundaries between applications are not enforced
- Integrity? Confidentiality? Authenticity?

Trusted Computing aims to fix that:

- Strong isolation, restrictive interfaces, exclusive I/O
- Built-in cryptography and (remote) attestation



Comparing Hardware-Based Trusted Computing Architectures

	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance	Memory Protection	Lightweight	Coprocessor	HW-Only TCB	Preemption	Dynamic Layout	Upgradeable TCB	Backwards Compatibility	Open-Source	Academic	Target ISA
AEGIS	●	●	●	●	●	○	●	○	○	●	●	●	○	●	○	●	–
TPM	○	●	●	○	●	–	◐	○	●	●	–	–	○	●	○	○	–
TXT	●	●	●	●	●	●	◐	○	●	●	○	●	○	●	○	○	x86_64
TrustZone	●	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	ARM
Bastion	●	○	●	●	●	○	●	○	○	○	●	●	●	●	○	●	UltraSPARC
SMART	○	●	○	●	○	–	○	●	○	○	–	–	○	●	○	●	AVR/MSP430
Sancus 1.0	●	●	○	●	○	●	○	●	○	●	○	○	○	●	●	●	MSP430
Soteria	●	●	○	●	●	●	○	●	○	●	○	○	○	●	●	●	MSP430
Sancus 2.0	●	●	○	●	●	●	○	●	○	●	◐	○	○	●	●	●	MSP430
SecureBlue++	●	○	●	●	●	○	●	○	○	●	●	●	○	●	○	○	POWER
SGX	●	●	●	●	●	○	●	○	○	○	●	●	●	●	○	○	x86_64
Iso-X	●	●	○	●	○	○	●	○	○	○	●	●	●	●	○	●	OpenRISC
TrustLite	●	●	○	○	○	●	○	●	○	○	●	●	●	●	○	●	Siskiyou Peak
TyTAN	●	●	●	●	○	●	○	●	○	○	●	●	●	●	○	●	Siskiyou Peak
Sanctum	●	●	●	●	●	●	○	○	○	○	●	●	●	●	◐	●	RISC-V

● = Yes; ◐ = Partial; ○ = No; – = Not Applicable

Adapted from "Hardware-Based Trusted Computing Architectures for Isolation and Attestation", Maene et al., IEEE Transactions on Computers, 2017. [MGdC+17]

Sancus: Strong and Light-Weight Embedded Security [NVBM+17]

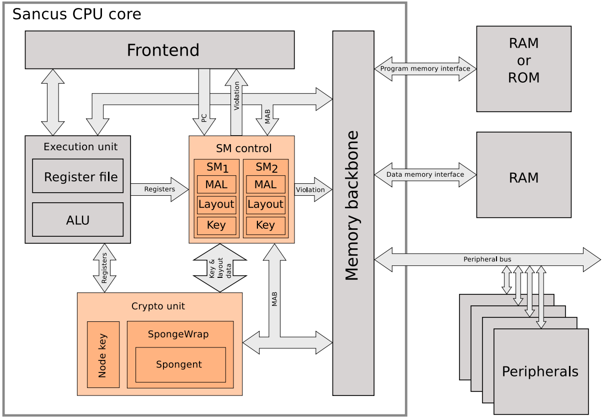
Extends openMSP430 with strong security primitives

- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges

Efficient

- Modular, $\leq$ 2 kLUTs
- Authentication in μs
- $+$ 6% power consumption

Cryptographic key hierarchy for software attestation



Isolated components are typically very small ($<$ 1kLOC)

Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

Sancus: Strong and Light-Weight Embedded Security [NVBM$^+$17]

Extends openMSP430 with strong security primitives

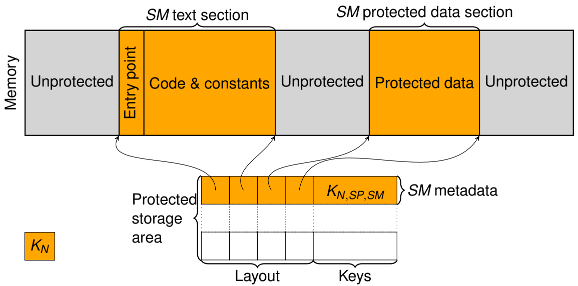
- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges

Efficient

- Modular, $\leq$ 2 kLUTs
- Authentication in μs
- $+$ 6% power consumption

Cryptographic key hierarchy for software attestation

N = Node; SP = Software Provider / Deployer
SM = protected Software Module



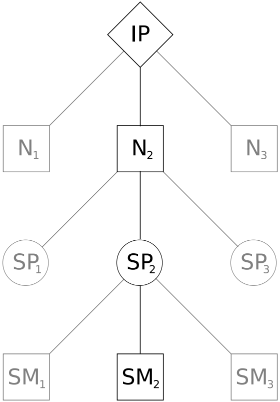
Isolated components are typically very small ($<$ 1kLOC)

Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

Attestation and Communication with Sancus

Ability to use $K_{N,SP,SM}$ proves the integrity and isolation of SM deployed by SP on N

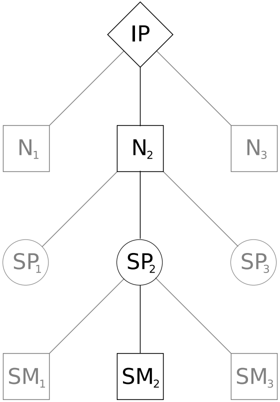
- Only N and SP can compute $K_{N,SP,SM}$
  N knows K_N and SP knows K_{SP}
- $K_{N,SP,SM}$ on N is computed after enabling isolation
  No isolation, no key; no integrity, wrong key
- Only SM on N is allowed to use $K_{N,SP,SM}$
  Through special instructions



Attestation and Communication with Sancus

Ability to use $K_{N,SP,SM}$ proves the integrity and isolation of *SM* deployed by *SP* on *N*

- Only *N* and *SP* can compute $K_{N,SP,SM}$
  N knows K_N and *SP* knows K_{SP}
- $K_{N,SP,SM}$ on *N* is computed after enabling isolation
  No isolation, no key; no integrity, wrong key
- Only *SM* on *N* is allowed to use $K_{N,SP,SM}$
  Through special instructions



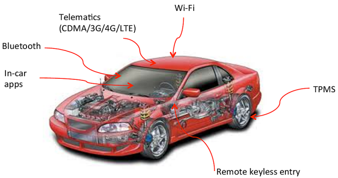
Remote attestation and secure communication by Authenticated Encryption with Associated Data

- Confidentiality, integrity and authenticity
- Encrypt and decrypt instructions use $K_{N,SP,SM}$ of the calling SM
- Associated Data can be used for nonces to get freshness

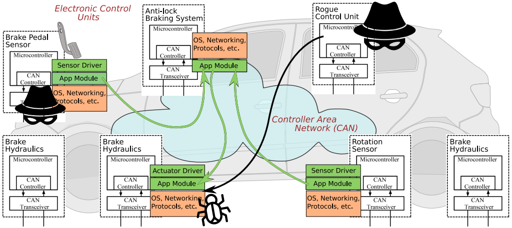
Secure Automotive Computing with Sancus [BMP17]

Modern cars can be hacked!

- Network of more than 50 ECUs
- Multiple communication networks
- Remote entry points
- Limited built-in security mechanisms



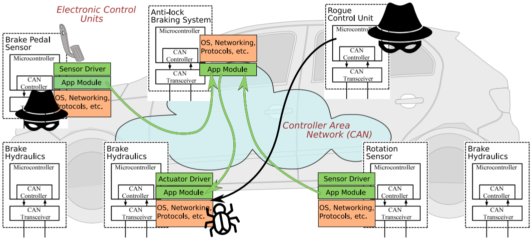
Miller & Valasek, "Remote exploitation of an unaltered passenger vehicle", 2015



Sancus brings strong security for embedded control systems:

- Message authentication
- Trusted Computing: software component isolation and cryptography
- Strong software security
- Applicable in automotive, ICS, IoT, . . .

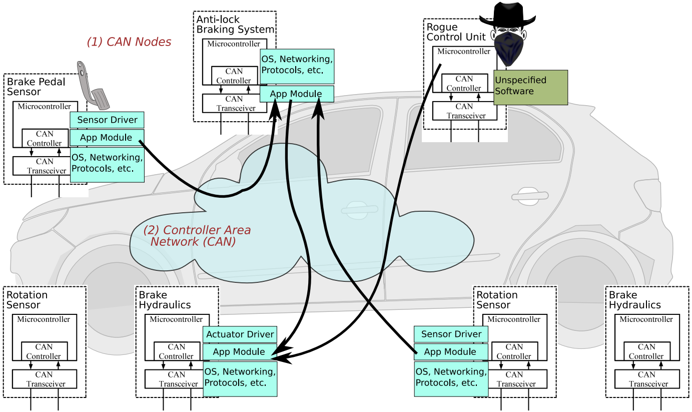
Secure Automotive Computing with Sancus [BMP17]



VulCAN: Generic design to exploit light-weight TC in CAN-based control networks; `https://distrinet.cs.kuleuven.be/software/vulcan/`

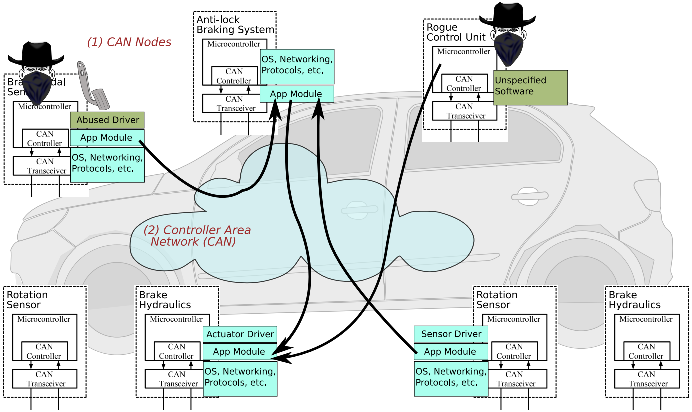
Implementation: based on Sancus [NVBM+17]; we implement, strengthen and evaluate authentication protocols, vatiCAN [NR16] and LeiA [RG16]

Attacking the CAN



Complex bus system with many ECUs and gateways to other communication systems; no protection against message injection or replay attacks.
→ Message Authentication; specified in AUTOSAR, proposals: vatiCAN, LeiA; no efficient and cost-effective implementations yet

Attacking CAN Message Authentication

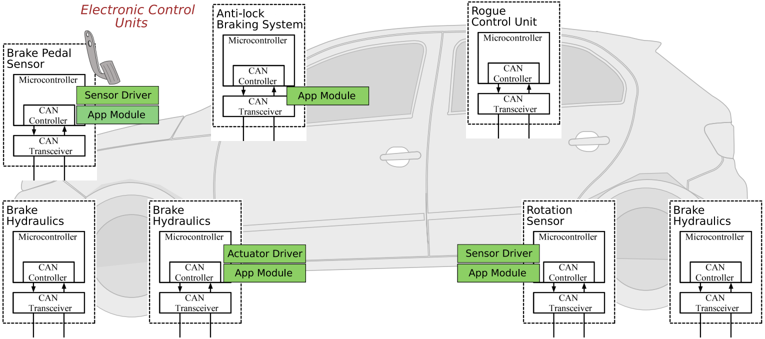


What about Software Security?

Lack of security mechanisms on light-weight ECUs leverages software vulnerabilities: attackers may be able to bypass encryption and authentication.

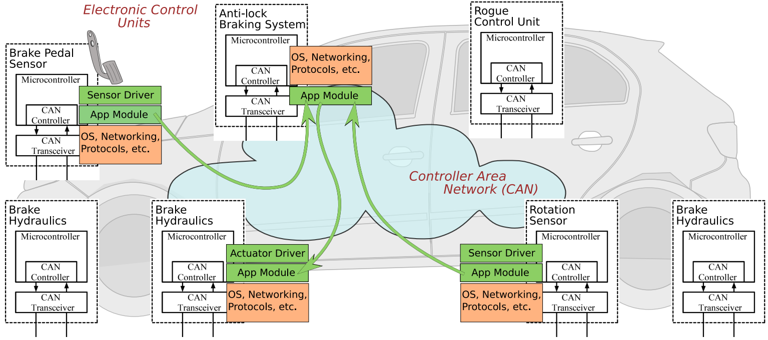
→ Software Component Authentication & Isolation

Vulcanising Distributed Automotive Applications



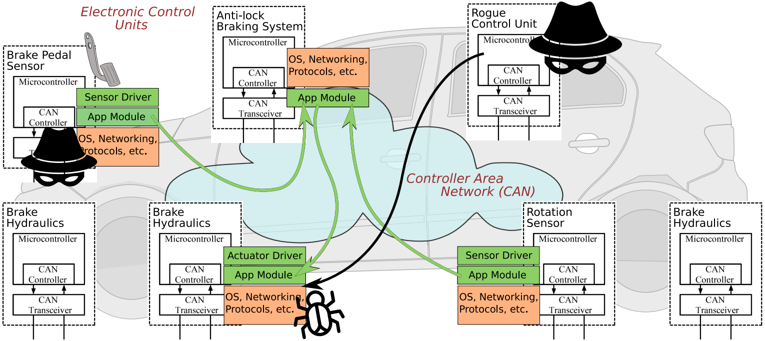
- Critical application components in enclaves: software isolation + attestation

Vulcanising Distributed Automotive Applications



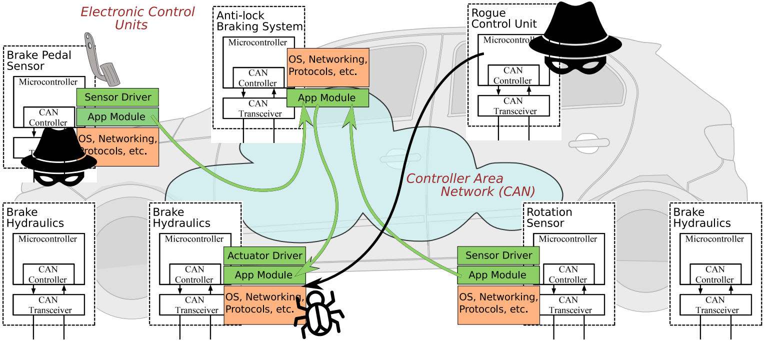
- Critical application components in enclaves: software isolation + attestation
- Authenticated CAN messages over untrusted system software/network

Vulcanising Distributed Automotive Applications



- Critical application components in enclaves: software isolation + attestation
- Authenticated CAN messages over untrusted system software/network
- Rogue ECUs, software attackers and errors in untrusted code cannot interfere with security, but may harm availability

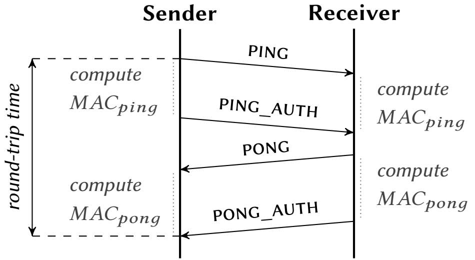
Vulcanising Distributed Automotive Applications



- Critical application components in enclaves: software isolation + attestation
- Authenticated CAN messages over untrusted system software/network
- Rogue ECUs, software attackers and errors in untrusted code cannot interfere with security, but may harm availability
- Infrastructure support: Trusted Computing, **Sancus**

Performance Evaluation: Round-Trip Time Experiment

Scenario	Cycles	Time	Overhead
Legacy	20,250	1.01 ms	–
vatiCAN (extrapolated)	121,992	6.10 ms	502%
Sancus+vatiCAN unprotected	35,236	1.76 ms	74%
Sancus+vatiCAN protected	36,375	1.82 ms	80%
Sancus+LEIA unprotected	42,929	2.15 ms	112%
Sancus+LEIA protected	43,624	2.18 ms	115%



- Hardware-level crypto: +400% performance gain
- Modest ~5% performance impact for software isolation

Authentic Execution of Distributed Event-Driven Applications

"Authentic Execution of Distributed Event-Driven Applications with a Small TCB", Noorman et al., STM 2017. [NMP17]

Summary

Security

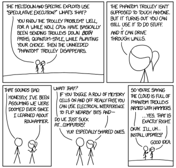
1. Understand the system
2. Understand the security requirements
3. Understand the attacker
4. Understand and embrace change

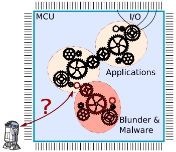
Trusted Computing

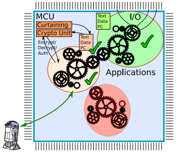
1. Strong security for distributed applications
2. Requires correct hardware and software
3. High potential for invasive use

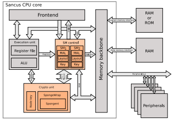
Sancus

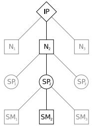
1. The Open-Source Trusted Computing Architecture
2. Built upon openMSP430 16-bit MCU, applications in IoT and embedded control systems
3. Research prototype under active development!

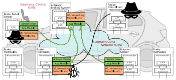












DistriNet

Ongoing Work

IoT Trust Assessment: secure inspection SW

Secure I/O: trusted Paths between sensors and actuators on distributed nodes

Programming Models: authenticity and integrity for event-driven distributed apps

Integration, toolchain and hardware maturity: ext. application scenarios, involve SGX and TrustZone, compiler fixes

Attacks and Mitigation: side channels

Availability and Real-Time: to control reactive safety-critical components in, e.g. automotive, avionic and medical domains

Safe Languages and Formal Verification: guarantee safe operation and absence of vulnerabilities in hardware and software

Thank you!

Thank you! Questions?

https://distrinet.cs.kuleuven.be/software/sancus/

References I

J. V. Bulck, J. T. Mühlberg, and F. Piessens.
Efficient component authentication and software isolation for automotive control networks.
In *ACSAC '17*, pp. 225–237. ACM, 2017.

P. Maene, J. Gotzfried, R. de Clercq, T. Muller, F. Freiling, and I. Verbauwhede.
Hardware-based trusted computing architectures for isolation and attestation.
IEEE Transactions on Computers, PP(99):1–1, 2017.

C. Miller and C. Valasek.
Remote exploitation of an unaltered passenger vehicle.
Black Hat USA, 2015.

J. Noorman, J. T. Mühlberg, and F. Piessens.
Authentic execution of distributed event-driven applications with a small TCB.
In *STM '17*, vol. 10547 of *LNCS*, pp. 55–71, Heidelberg, 2017. Springer.

S. Nürnberger and C. Rossow.
– vatiCAN – Vetted, Authenticated CAN Bus, pp. 106–124.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

J. Noorman, J. Van Bulck, J. T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, and F. Freiling.
Sancus 2.0: A low-cost security architecture for IoT devices.
ACM Transactions on Privacy and Security (TOPS), 20:7:1–7:33, 2017.

A.-I. Radu and F. D. Garcia.
LeiA: A Lightweight Authentication Protocol for CAN, pp. 283–300.
Springer International Publishing, Cham, 2016.