Building Distributed Enclave Applications with Sancus and SGX

Jan Tobias Mühlberg and **Jo Van Bulck**
<jantobias.muehlberg|jo.vanbulck>@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

DSN 2018 @ Luxembourg, 25th of June 2018

https://distrinet.cs.kuleuven.be/software/sancus/tutorial.php
https://github.com/sancus-pma/tutorial-dsn18

DistriNet

Building Distributed Enclave Applications with Sancus and SGX

Jan Tobias Mühlberg and **Jo Van Bulck**
<jantobias.muehlberg|jo.vanbulck>@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

DSN 2018 @ Luxembourg, 25th of June 2018

Lecture 1: Authentic Execution on Heterogeneous Trusted Computing infrastructure

DistriNet

Security (in the IoT)

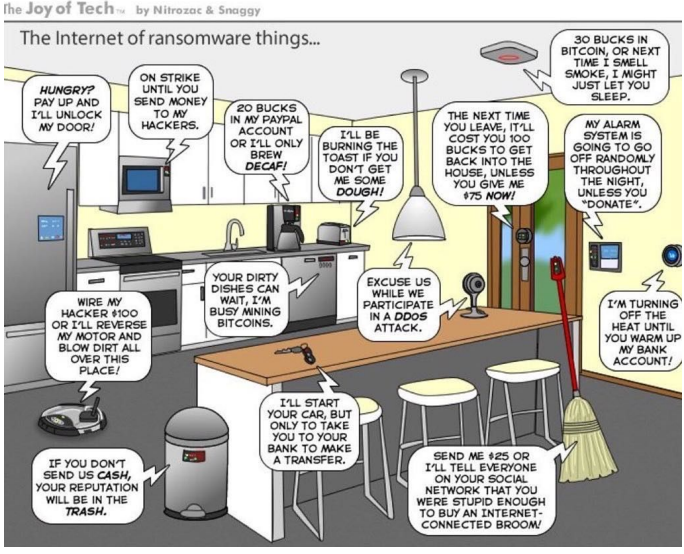
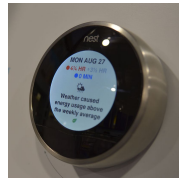


Image source: `http://www.joyoftech.com/joyoftech/joyarchives/2340.html`

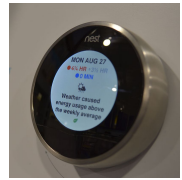
Security (more in general)



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

Security (more in general)

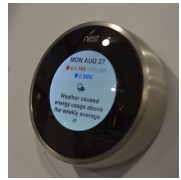
1. **Understand the system.**
   - Context, hardware, software, data, users, use cases, etc.



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

Security (more in general)

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.”



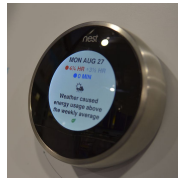




Source of images 1, 2, 3: `https://en.wikipedia.org/`

Security (more in general)

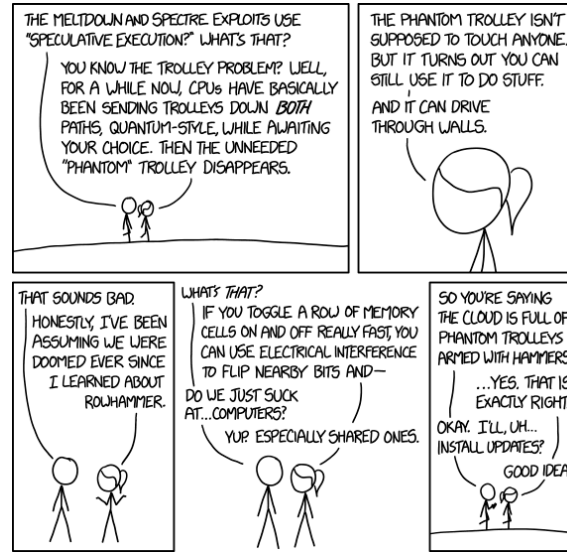
1. **Understand the system.**
   - Context, hardware, software, data, users, use cases, etc.
2. **Understand the security requirements.**
   - Requirements are not features!
   - “Only authenticated users can do X. Two-factor authentication is required for all users. All X are logged, detailing time, 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."

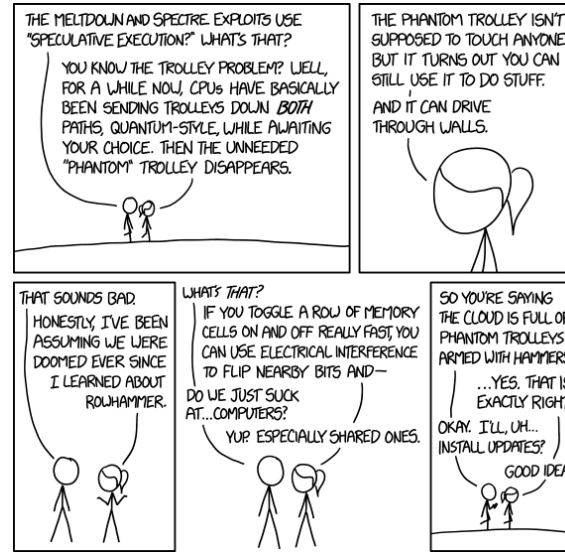


Source: https://xkcd.com/1938/

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.**

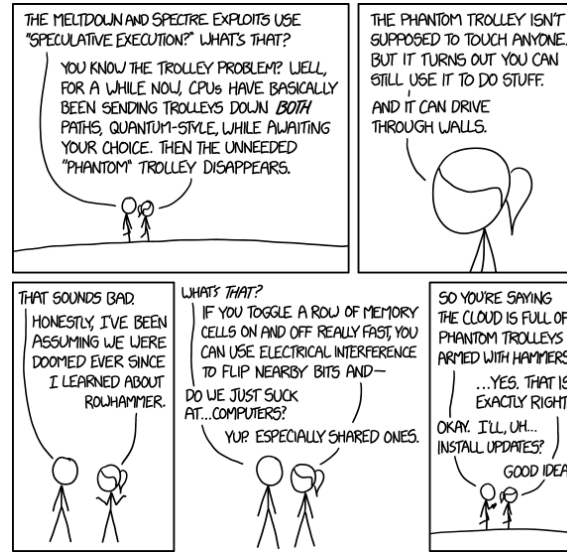


Source: https://xkcd.com/1938/

Security

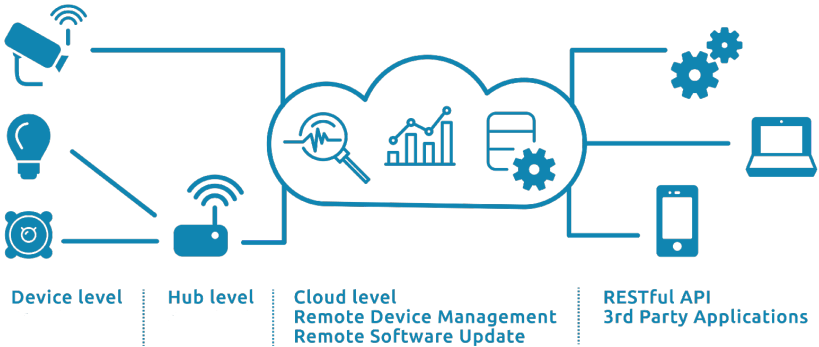
"New zero-day vulnerability: In addition to rowhammer, it turns out lots of servers are vulnerable to regular hammers, too."

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/

Security

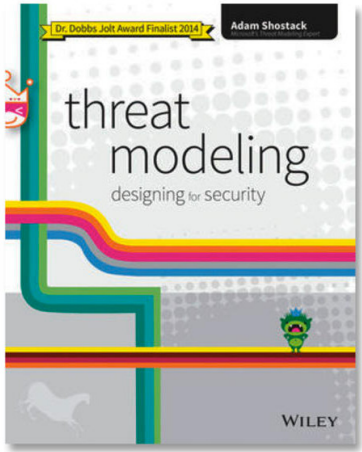


Understanding can be really difficult! What stake holders are involved? What are their objectives and abilities? What hardware and software is involved? Software quality? Data flows? Security requirements and guarantees?

Image source: `https://medium.com/connected-news/iot-foundation-what-is-an-iot-platform-c37c5e72d4a0`

Security

Read a good book!
“Threat Modeling: Designing for Security”
Adam Shostack, Wiley, 2014



Security

Read a good book!

“Threat Modeling: Designing for Security”
Adam Shostack, Wiley, 2014

Learn about security tech!

Today: The awesomeness of Trusted Computing, and how to build secure distributed applications.

Sancus 2.0: A Low-Cost Security Architecture for IoT Devices

JOB NOORMAN, JO VAN BULCK, JAN TOBIAS MÜHLBERG, and FRANK PIESSENS, iMinds-DistriNet, KU Leuven
PIETER MAENE, BART PRENEEL, and INGRID VERBAUWHEDE, iMinds-COSIC, KU Leuven
JOHANNES GÖTZFRIED, TILO MÜLLER, and FELIX FREILING, FAU Erlangen-Nürnberg

The Sancus security architecture for networked embedded devices was proposed in 2013 at the USENIX Security conference. It supports remote (even third-party) software installation on devices while maintaining strong security guarantees. More specifically, Sancus can remotely attest to a software provider that a specific software module is running uncompromised, and can provide a secure communication channel between software modules and software providers. Software modules can securely maintain local state, and can securely interact with other software modules that they choose to trust.

Over the past three years, significant experience has been gained with applications of Sancus, and several extensions of the architecture have been investigated – both by the original designers as well as by independent researchers. Informed by these additional research results, this journal version of the Sancus paper describes an improved design and implementation, supporting additional security guarantees (such as confidential deployment) and a more efficient cryptographic core.

We describe the design of Sancus 2.0 (without relying on any prior knowledge of Sancus), and develop and evaluate a prototype FPGA implementation. The prototype extends an MSP430 processor with hardware support for the memory access control and cryptographic functionality required to run Sancus. We report on our experience with using Sancus in a variety of application scenarios, and discuss some important avenues of ongoing and future work.

CCS Concepts: •**Security and privacy → Embedded systems security;** •**Computer systems organization →** *Embedded systems*; Sensor networks; Sensors and actuators;

Additional Key Words and Phrases: Protected Module Architectures, Embedded systems security, Trusted computing, Software security engineering

ACM Reference format:
Job Noorman, Jo Van Bulck, Jan Tobias Mühlberg, Frank Piessens, Pieter Maene, Bart Preneel, Ingrid Verbauwhede, Johannes Götzfried, Tilo Müller, and Felix Freiling. 0000. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. *ACM Trans. Priv. Sec.* 0, 0, Article 0 (0000), 33 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

This work has been supported in part by the Intel Lab's University Research Office. This research is also partially funded by the Research Fund KU Leuven, and by the Research Foundation - Flanders (FWO). This work was partially supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89). Job Noorman, Jo Van Bulck, and Pieter Maene are supported by a doctoral grant of the Research Foundation - Flanders (FWO).

Author's addresses: {Job.Noorman, Jo.VanBulck, JanTobias.Muehlberg, Frank.Piessens}@cs.kuleuven.be, {Pieter.Maene, Bart.Preneel, Ingrid.Verbauwhede}@esat.kuleuven.be, {Johannes.Goetzfried, Tilo.Mueller, Felix.Freiling}@cs.fau.de.

DOI: 10.1145/nnnnnnn.nnnnnnn

ACM Transactions on Privacy and Security, Vol. 0, No. 0, Article 0, Publication date: 0000.

Security

Read a good book!
“Threat Modeling: Designing for Security”
Adam Shostack, Wiley, 2014

Learn about security tech!
Today: The awesomeness of Trusted Computing,
and how to build secure distributed applications.

Be prepared…
…to reconsider Trusted Computing once
it is broken or something better comes along!

Sancus 2.0: A Low-Cost Security Architecture for IoT Devices

JOB NOORMAN, JO VAN BULCK, JAN TOBIAS MÜHLBERG, and FRANK PIESSENS, iMinds-DistriNet, KU Leuven
PIETER MAENE, BART PRENEEL, and INGRID VERBAUWHEDE, iMinds-COSIC, KU Leuven
JOHANNES GÖTZFRIED, TILO MÜLLER, and FELIX FREILING, FAU Erlangen-Nürnberg

The Sancus security architecture for networked embedded devices was proposed in 2013 at the USENIX Security conference. It supports remote (even third-party) software installation on devices while maintaining strong security guarantees. More specifically, Sancus can remotely attest to a software provider that a specific software module is running uncompromised, and can provide a secure communication channel between software modules and software providers. Software modules can securely maintain local state, and can securely interact with other software modules that they choose to trust.

Over the past three years, significant experience has been gained with applications of Sancus, and several extensions of the architecture have been investigated – both by the original designers as well as by independent researchers. Informed by these additional research results, this journal version of the Sancus paper describes an improved design and implementation, supporting additional security guarantees (such as confidential deployment) and a more efficient cryptographic core.

We describe the design of Sancus 2.0 (without relying on any prior knowledge of Sancus), and develop and evaluate a prototype FPGA implementation. The prototype extends an MSP430 processor with hardware support for the memory access control and cryptographic functionality required to run Sancus. We report on our experience with using Sancus in a variety of application scenarios, and discuss some important avenues of ongoing and future work.

CCS Concepts: •**Security and privacy → Embedded systems security;** •**Computer systems organization →** *Embedded systems*; Sensor networks; Sensors and actuators;

Additional Key Words and Phrases: Protected Module Architectures, Embedded systems security, Trusted computing, Software security engineering

ACM Reference format:
Job Noorman, Jo Van Bulck, Jan Tobias Mühlberg, Frank Piessens, Pieter Maene, Bart Preneel, Ingrid Verbauwhede, Johannes Götzfried, Tilo Müller, and Felix Freiling. 0000. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. *ACM Trans. Priv. Sec.* 0, 0, Article 0 (0000), 33 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

This work has been supported in part by the Intel Lab's University Research Office. This research is also partially funded by the Research Fund KU Leuven, and by the Research Foundation - Flanders (FWO). This work was partially supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89). Job Noorman, Jo Van Bulck, and Pieter Maene are supported by a doctoral grant of the Research Foundation - Flanders (FWO).

Author's addresses: {Job.Noorman, Jo.VanBulck, JanTobias.Muehlberg, Frank.Piessens}@cs.kuleuven.be, {Pieter.Maene, Bart.Preneel, Ingrid.Verbauwhede}@esat.kuleuven.be, {Johannes.Goetzfried, Tilo.Mueller, Felix.Freiling}@cs.fau.de.

DOI: 10.1145/nnnnnnn.nnnnnnn

ACM Transactions on Privacy and Security, Vol. 0, No. 0, Article 0. Publication date: 0000.

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

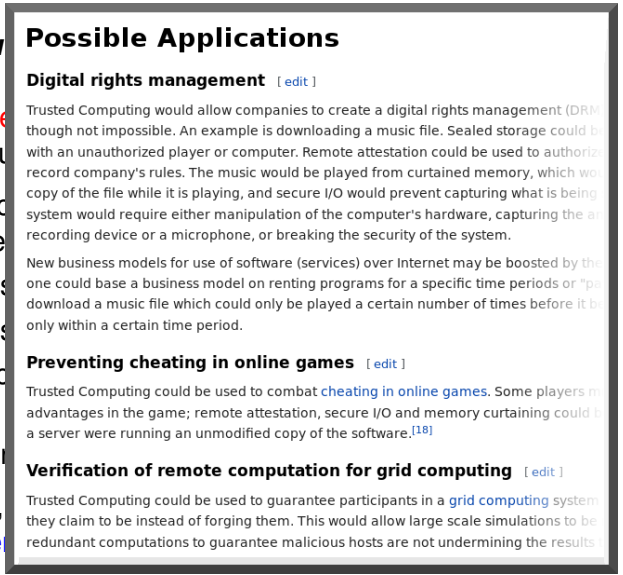
Source: `https://en.wikipedia.org/wiki/Trusted_Computing`

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 th
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 h
a server were running an unmodified copy of the software.[18]

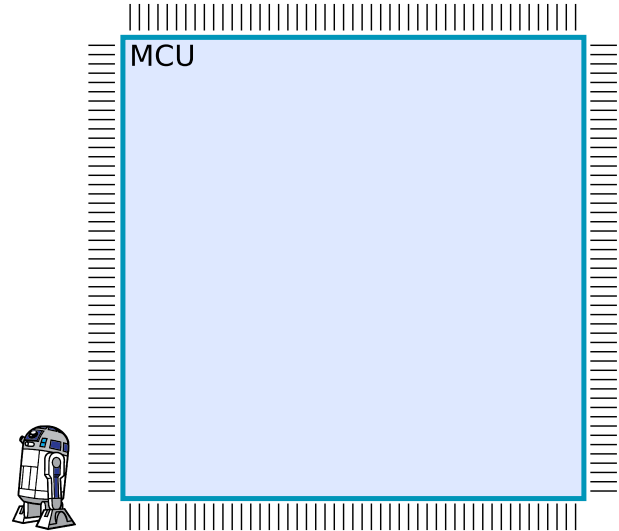
Verification of remote computation for grid computing [edit]

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`

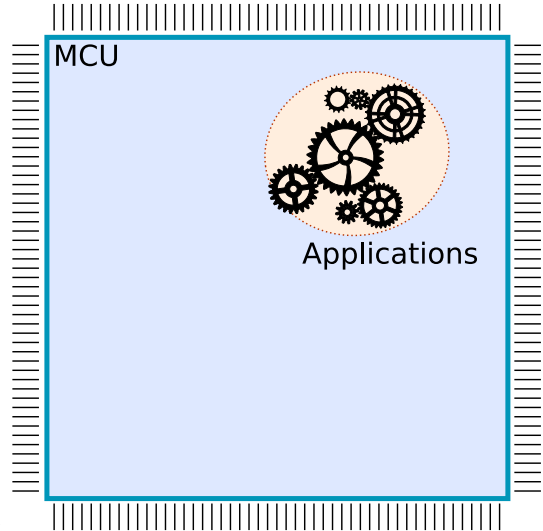
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality



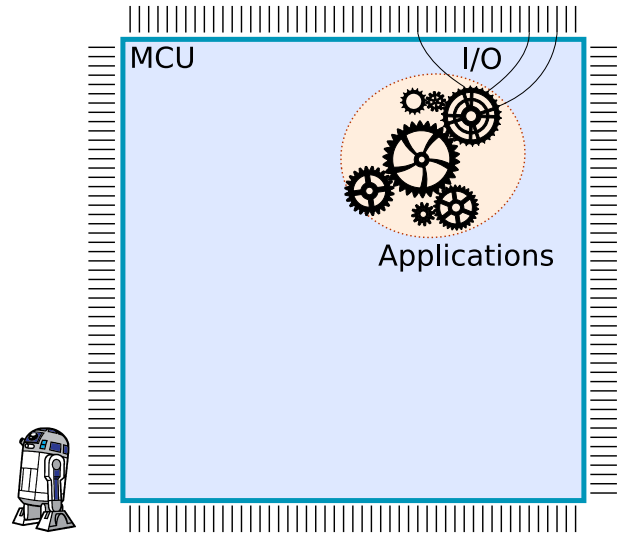
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality



Isolation and Attestation on Light-Weight MCUs

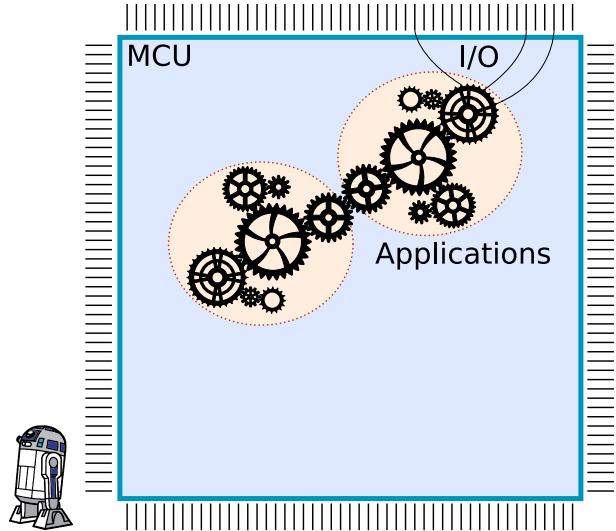
Many microcontrollers feature little security functionality



Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

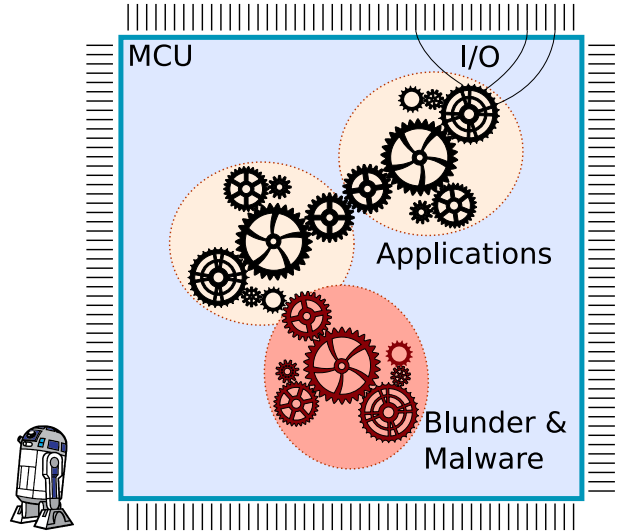
- Applications share address space



Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

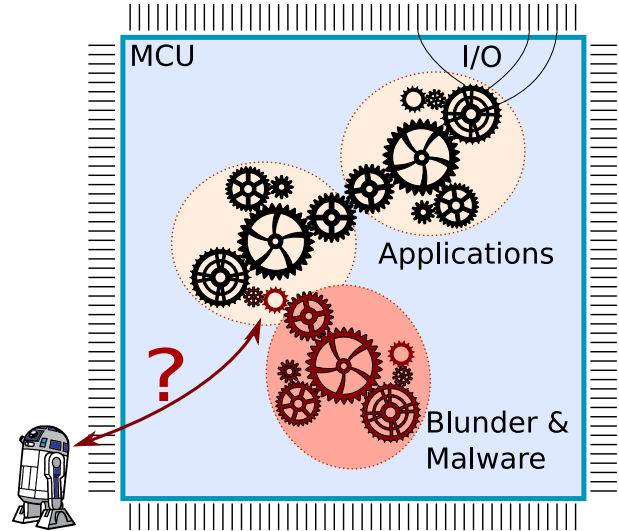
- Applications share address space
- Boundaries between applications are not enforced



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?



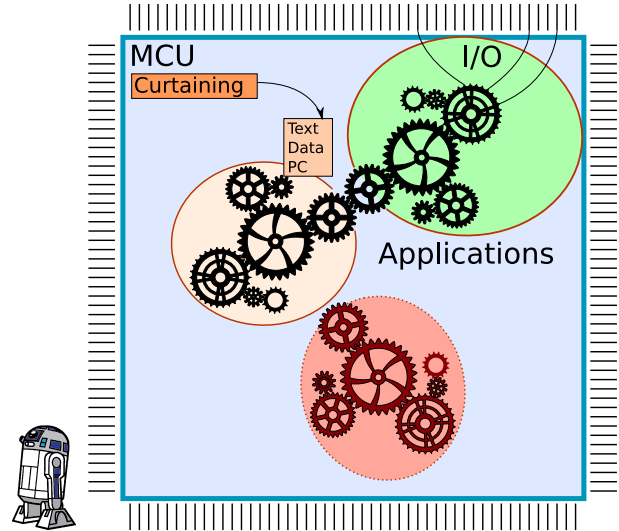
Isolation and Attestation on Light-Weight MCUs

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



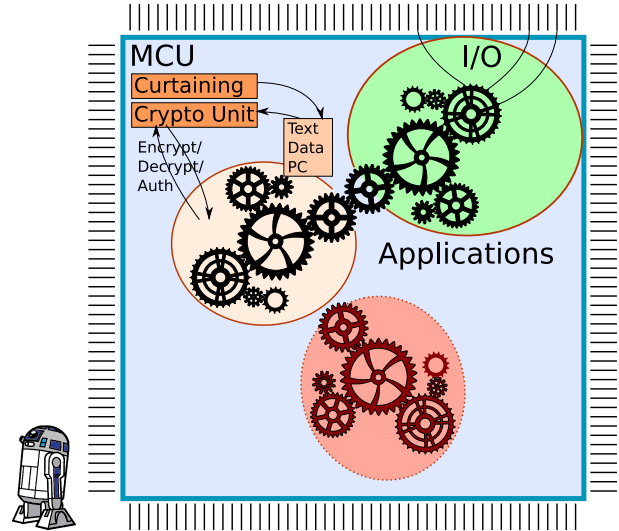
Isolation and Attestation on Light-Weight MCUs

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



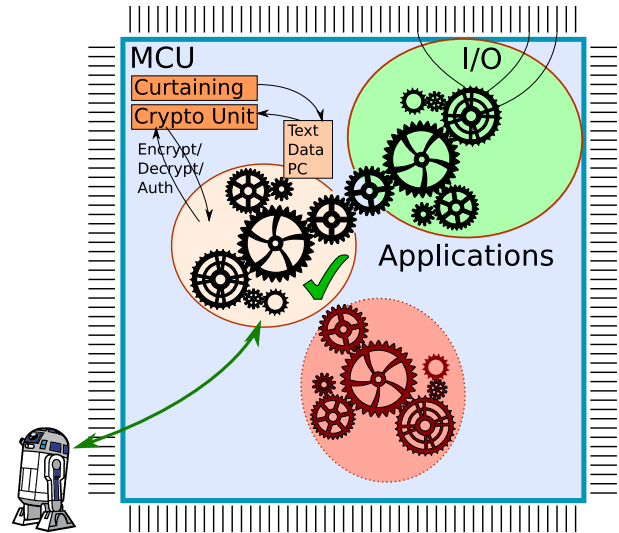
Isolation and Attestation on Light-Weight MCUs

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



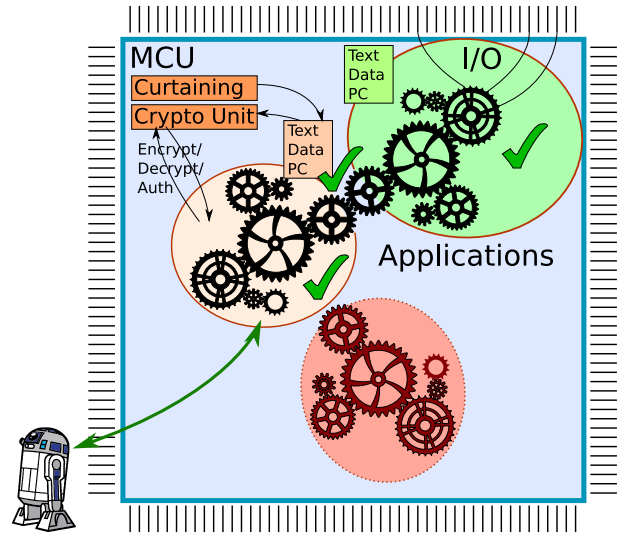
Isolation and Attestation on Light-Weight MCUs

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



Isolation and Attestation on Light-Weight MCUs

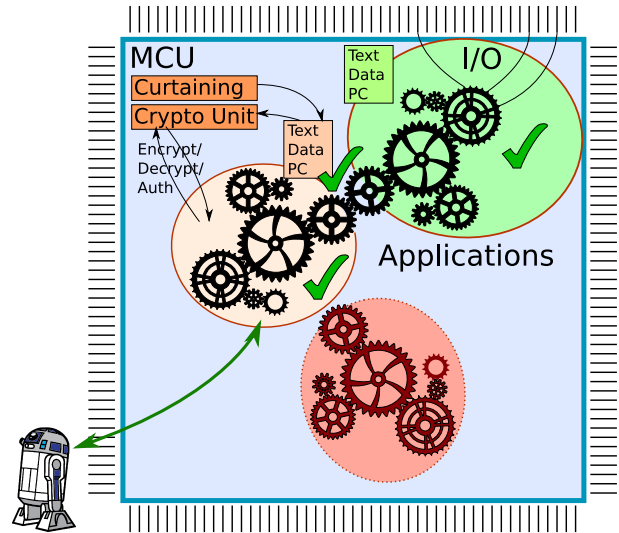
Many microcontrollers feature little security functionality

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

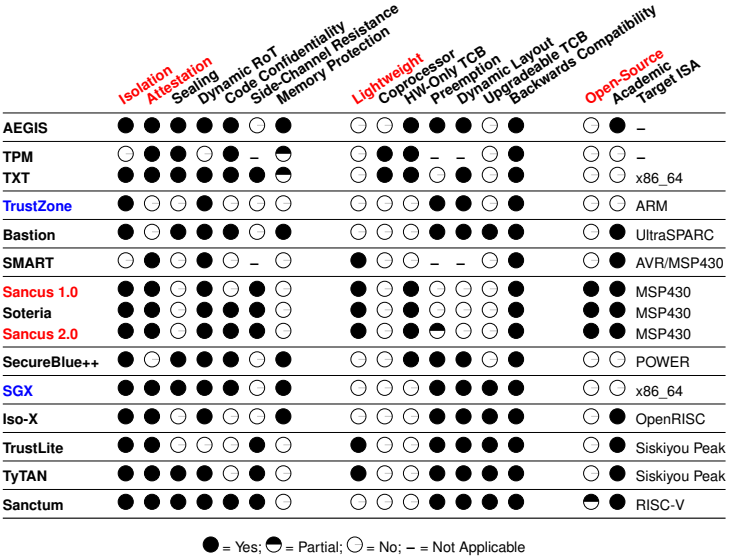
Trusted Computing aims to fix that

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

→ **"Authentic Execution"** [NMP17]



Comparing Hardware-Based Trusted Computing Architectures



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]

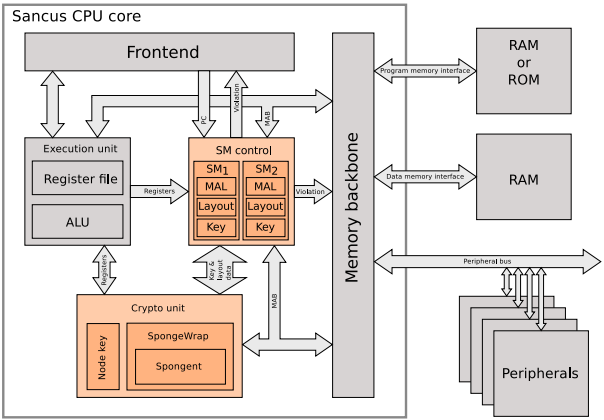
Extends openMSP430 with strong security primitives

- 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

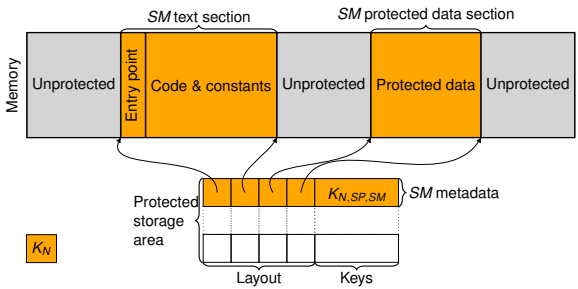
- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges

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



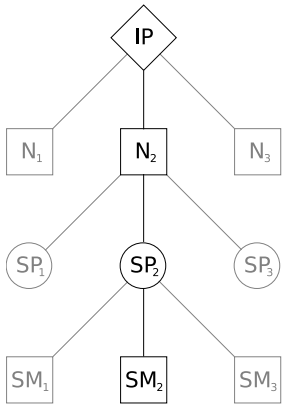
Isolated components are typically very small ($<$ 1kLOC)

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*

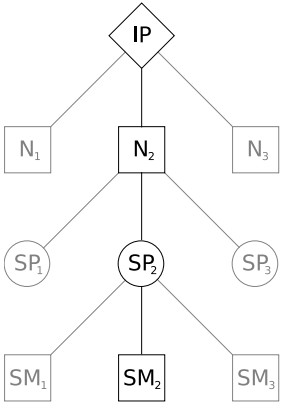
- Only *N* and *SP* can compute $K_{N,SP,SM}$
  N knows K_N and *SP* knows K_{SP}
- $K_{N,SP,SM}$ on *N* is computed after enabling isolation
  No isolation, no key; no integrity, wrong key
- Only *SM* on *N* is allowed to use $K_{N,SP,SM}$
  Through special instructions



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



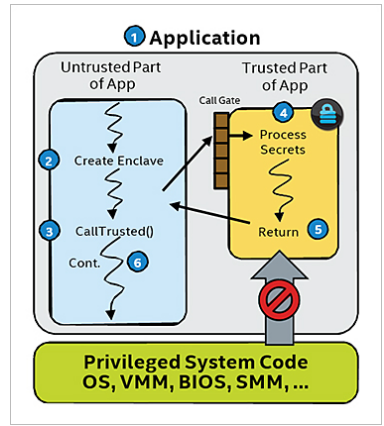
Remote attestation and secure communication by Authenticated Encryption with Associated Data

- 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

A Brief Intro to Intel SGX

Intel Software Guard Extensions

- For server & desktop market
- Features memory encryption and asymmetric crypto
- No secure I/O, side-channels explicitly out of scope
- Secure “production enclaves” require licensing



Enclave Lifecycle

1. App built with trusted and untrusted parts
2. App runs and creates the enclave
3. Execution transitions to the enclave
4. Enclave sees all process data in clear; external access to enclave data is denied
5. Trusted function returns enclave data
6. Application continues normal execution

Source: `https://software.intel.com/en-us/sgx/details`

Authentic Execution of Distributed Applications [NMP17]

Authentic Execution of Distributed Applications [NMP17]

Authentic Execution

- We can explain every observed output event based on a trace of actual input events and the (source) code of the application, in the presence of network and software attackers

Authentic Execution of Distributed Applications [NMP17]

Authentic Execution

- We can explain every observed output event based on a trace of actual input events and the (source) code of the application, in the presence of network and software attackers

. . . modulo availability and confidentiality, but . . .

Authentic Execution of Distributed Applications [NMP17]

Authentic Execution

- We can explain every observed output event based on a trace of actual input events and the (source) code of the application, in the presence of network and software attackers

. . . modulo availability and confidentiality, but . . .

Distributed Event-Driven Applications

- Application modules execute on heterogeneous distributed Trusted Computing infrastructure
- Modules communicate via secure (integrity protected) channels
- Physical events enter or leave the application through secure I/O channels
- Multiple distrusting applications share the infrastructure

Authentic Execution of Distributed Applications [NMP17]

Authentic Execution

- We can explain every observed output event based on a trace of actual input events and the (source) code of the application, in the presence of network and software attackers

. . . modulo availability and confidentiality, but . . .

Distributed Event-Driven Applications

- Application modules execute on heterogeneous distributed Trusted Computing infrastructure
- Modules communicate via secure (integrity protected) channels
- Physical events enter or leave the application through secure I/O channels
- Multiple distrusting applications share the infrastructure

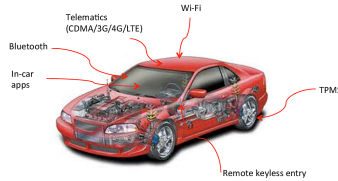
With a small (run-time) Trusted Computing Base

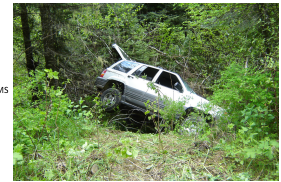
- Code that is not part of an application cannot interfere with that application w.r.t. security properties

Secure Automotive Computing with Sancus [VBMP17]

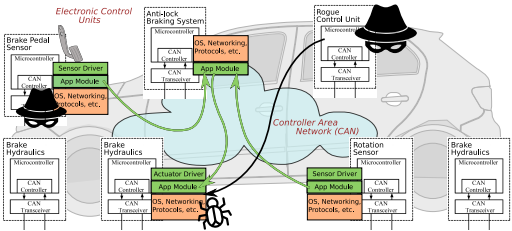
Modern cars can be hacked!

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





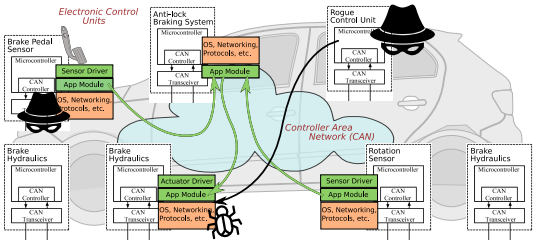
Miller & Valasek, “Remote exploitation of an unaltered passenger vehicle”, 2015



Sancus brings strong security for embedded control systems:

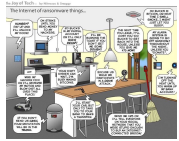
- Message authentication
- Trusted Computing: software component isolation and cryptography
- Strong software security
- Applicable in automotive, ICS, IoT, . . .

Secure Automotive Computing with Sancus [VBMP17]

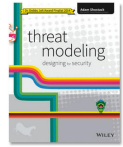


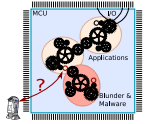
"VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks", Van Bulck et al., ACSAC 2017. [VBMP17]

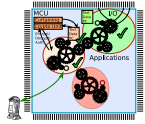
Summary

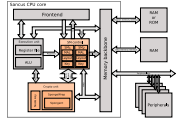

The Internet of ransomware things...


Device level
Hub level
Cloud level
Remote Device Management
Remote Software Update
RESTful API
3rd Party Applications


threat
modeling
designing for security
WILEY

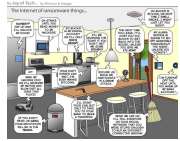

MCU
I/O
Applications
Blunder &
Malware

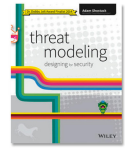

MCU
I/O
Applications

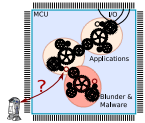

Sancus CPU core
Frontend
Memory Backbone
RAM
ROM
RAM
Peripherals

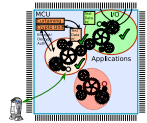
Summary

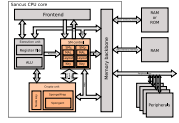
Security: understand, understand, understand. . .









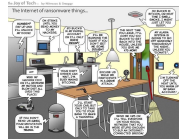


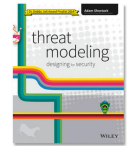
Summary

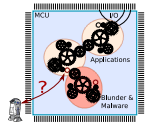
Security: understand, understand, understand. . .

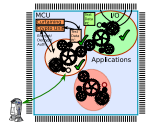
Trusted Computing & Authentic Execution

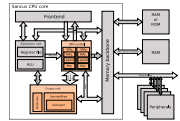
- Software isolation and attestation
- Strong security for distributed applications
- Requires correct hardware and software











Summary

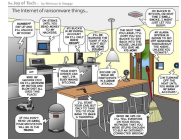
Security: understand, understand, understand. . .

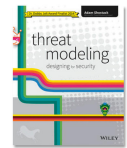
Trusted Computing & Authentic Execution

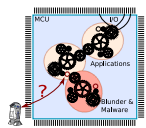
- Software isolation and attestation
- Strong security for distributed applications
- Requires correct hardware and software

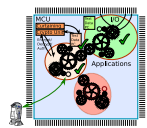
Intel SGX:

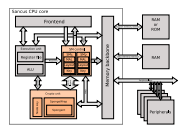
- Trusted execution on server & desktop
- Public key crypto and memory encryption
- Secure “production enclaves” require licensing











Summary

Security: understand, understand, understand. . .

Trusted Computing & Authentic Execution

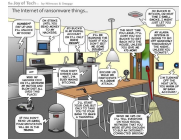
- Software isolation and attestation
- Strong security for distributed applications
- Requires correct hardware and software

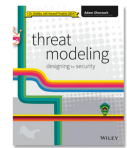
Intel SGX:

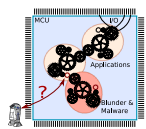
- Trusted execution on server & desktop
- Public key crypto and memory encryption
- Secure “production enclaves” require licensing

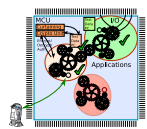
Sancus

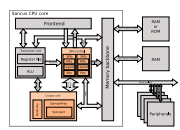
- The Open-Source Trusted Computing Architecture
- Built upon openMSP430 16-bit MCU, applications in IoT and embedded control systems
- Research prototype under active development!



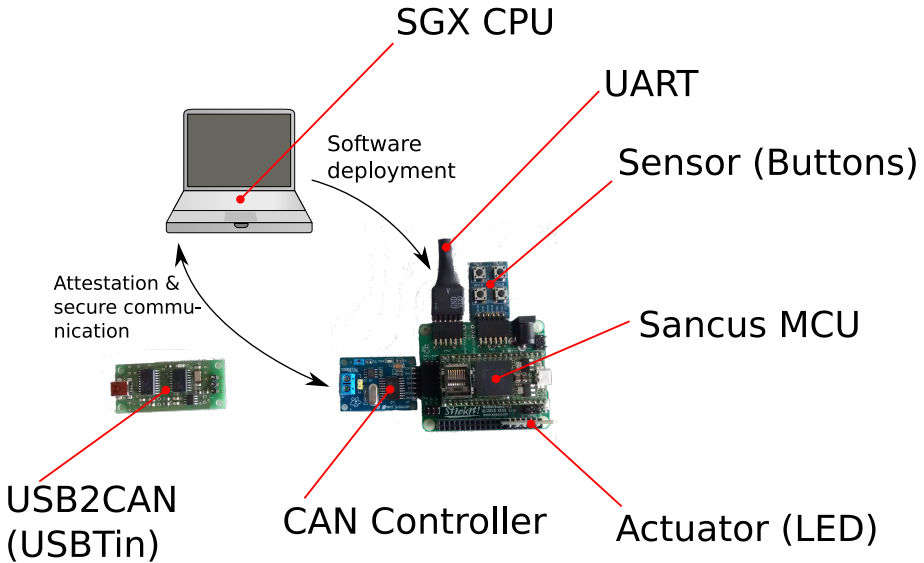




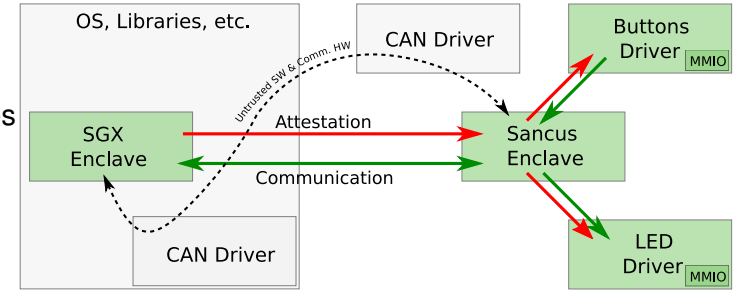




Tutorial Overview – Hardware Setup



Tutorial Overview – Practicals



Sancus Enclave

- Attest and interact with button/LED driver enclaves
- Send and receive auth'ed CAN messages

SGX Enclave

- Attest Sancus enclave
- Receive and verify CAN messages
- Processing and send authenticated CAN messages

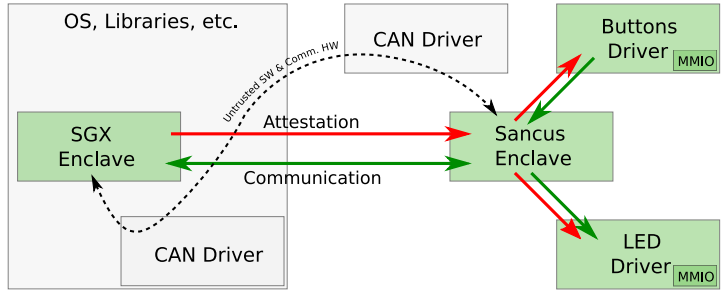
Untrusted Components

- CAN & USB hardware and software
- OS, networks, software deployment, support libraries
- But: We do trust compilation on the host!

Tutorial Overview – Learning Outcomes

Programming Enclaves

- Remote attestation
- ECALLs and OCALLs
- Untrusted pointers
- Secure random numbers
- Local attestation
- Secure I/O



Tricky bits

- Sanitising untrusted pointers
- Information leakage and side channels
- Freshness and non-repudiation: nonces and session keys
- Attesting SGX enclaves – what is the root of trust?

Concepts

- Authentic Execution: end-to-end security for distributed applications on heterogeneous Protected Module Architecture

Thank you!

Questions?

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

Building Distributed Enclave Applications with Sancus and SGX

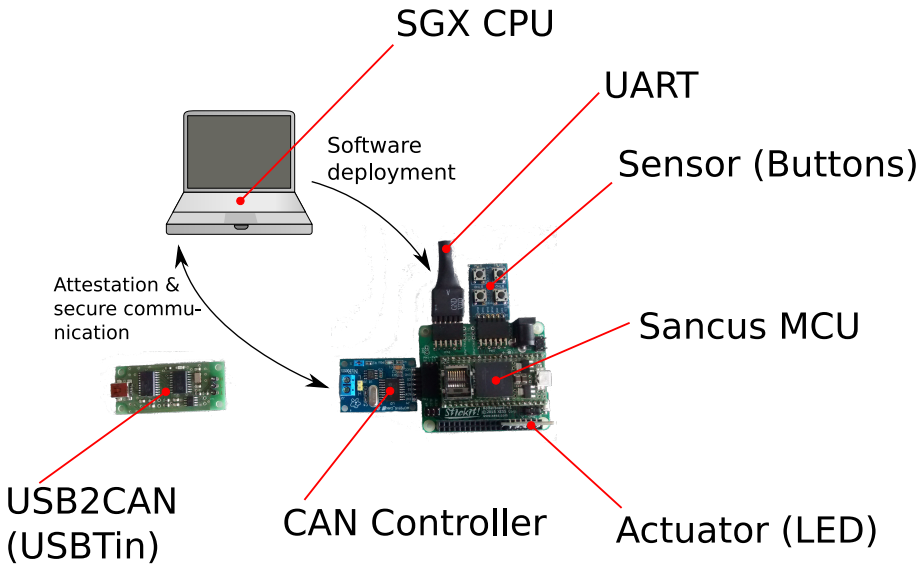
Jan Tobias Mühlberg and **Jo Van Bulck**
<jantobias.muehlberg|jo.vanbulck>@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

DSN 2018 @ Luxembourg, 25th of June 2018

Exercise 1: Basic Enclaves, Deployment, Attestation and Communication

DistriNet

Practicalities



Practicalities

SGX Machine

- What's the laptop's IP?
- Access via `ssh` (`scp|sshfs`): user & password: 'dsnN' with $00 \leq N \leq 09$
- UARTs to connect to Sancus MCUs:
  `/dev/ttyUSBX` – upload programs; $X = 2N$, c.f. project `Makefile`
  `/dev/ttyUSBX+1` – debug output; `screen /dev/ttyUSB1 115200`
- CAN networks: `slcanN` or `/dev/ttyACMN`; try `candump slcan0`
- Development environments:
  - SGX & Sancus SDKs are installed
  - `git clone https://github.com/sancus-pma/tutorial-dsn18.git`
  - No IDEs! Use a text editor or edit via `sshfs` on your laptop.

Practicalities

https://github.com/sancus-pma/tutorial.../app/sancus/000-blinking-led

Sancus: Check the Setup

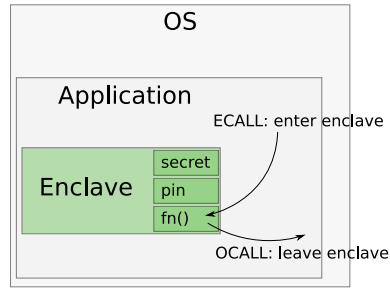
- `ssh dsnN@SGXLaptop`; you'll need three shells
- `git clone https://github.com/sancus-pma/tutorial-dsn18.git`
- `cd tutorial-dsn18`
- `git submodule init ; git submodule update`
- `cd app/sancus/000-blinking-led/`
- 1st shell: `screen /dev/ttyUSBX+1 115200`
- 2nd shell: `candump slcanN`
- 3rd shell: `FPGA_DEV=/dev/ttyUSBX make load`

Understanding Enclaves

https://github.com/sancus-pma/tutorial.../app/sgx/000-pin

A First SGX Enclave

- Enclave stores secret; secret is returned upon PIN validation
- `cd tutorial-dsn18/app/sgx/000-pin`
- `main.c` – create, use and destroy enclave
- `Enclave/encl.edl` – enclave interface
- `Enclave/encl.c` – enclave implementation
- `Makefile` – build script
- `make run`



Break!

Coffee!

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

Building Distributed Enclave Applications with Sancus and SGX

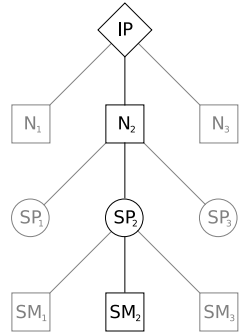
Jan Tobias Mühlberg and **Jo Van Bulck**
<jantobias.muehlberg|jo.vanbulck>@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

DSN 2018 @ Luxembourg, 25th of June 2018

Exercise 2: Attestation, Secure I/O, Authentic Execution in a Distributed Setting

DistriNet

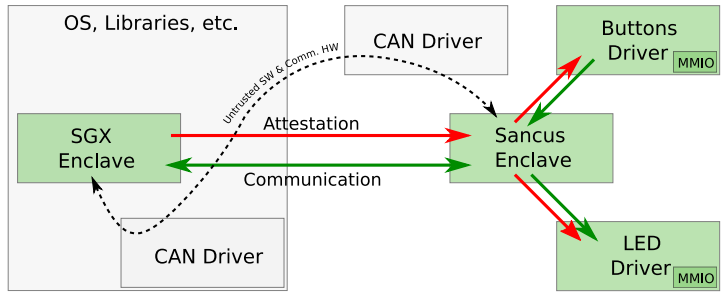
Remote Attestation



Attest a Sancus Enclave From an SGX Enclave

- SGX enclave requires proof that
  - an unmodified Sancus enclave
  - executes under protection on a certain MCU

1. Trusted Computing hardware measures enclave and derives module key K_{SM}
2. Attesting enclave (verifier) sends a challenge
3. Attested enclave (prover) encrypts/signs challenge w/ K_{SM} and returns result
4. Attesting enclave verifies signature



Remote Attestation

https://github.com/sancus-pma/tutorial.../app/sancus/001-attestation

Sancus: Receive Challenge, Send Signature

- `cd tutorial-dsn18/app/sancus/001-attestation`
- `foo.c` – implement an entry-point function `attest_foo()` that *tags* a challenge under K_{SM}
- `FPGA_DEV=/dev/ttyUSBX make load` – compile and run
- `sancus-crypto` – compute K_{SM} on the host
- `app/sgx/001-attestation-unprotected` – add key and try attestation from a “normal” process

Remote Attestation

https://github.com/sancus-pma/tutorial.../app/sgx/001-attestation

SGX: attesting enclave

- Based on `sgx/001-attestation-unprotected`, develop an SGX enclave that attests the Sancus enclave
- Which parts of the application logic must be executed in an enclave to secure K_{SM}?
- `cd tutorial-dsn18/app/sgx/002-auth-exec`
- `Enclave/encl.c` – implement challenge generation and verification
- `make run`

Authentic Execution

https://github.com/sancus-pma/tutorial.../app/sgx/002-auth-exec
https://github.com/sancus-pma/tutorial.../app/sancus/002-auth-exec

Sancus & SGX: Local Attestation, Secure I/O, Authentic Execution

- End-to-end scenario with
  - Local attestation of driver modules in Sancus
  - Secure inter-enclave calls to interact with drivers
  - Remote attestation of Sancus enclaves and I/O events

Thank you!

Questions?

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

Building Distributed Enclave Applications with Sancus and SGX

Jan Tobias Mühlberg and **Jo Van Bulck**
<jantobias.muehlberg|jo.vanbulck>@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

DSN 2018 @ Luxembourg, 25th of June 2018

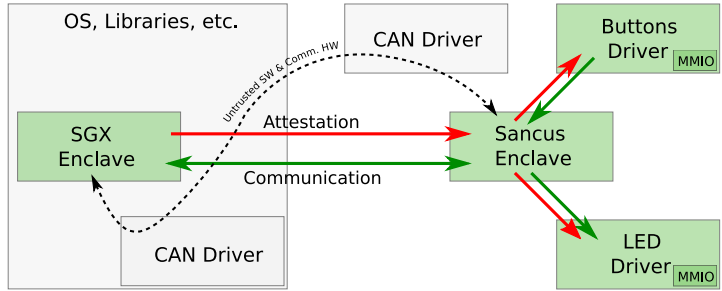
Lecture 2: Pitfalls, Advanced Topics, Wrap-Up

DistriNet

Tutorial Overview – Learning Outcomes

Programming Enclaves

- Remote attestation
- ECALLs and OCALLs
- Untrusted pointers
- Secure random numbers
- Local attestation
- Secure I/O



Tricky bits

- Sanitising untrusted pointers
- Information leakage and side channels
- Freshness and non-repudiation: nonces and session keys
- Attesting SGX enclaves – what is the root of trust?

Concepts

- Authentic Execution: end-to-end security for distributed applications on heterogeneous Protected Module Architecture

Thank you!

Questions?

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

References I

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.

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.

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