Trusted Execution – and how far you can trust it

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

CIF Seminar, CiTiP, KU Leuven, 2020-02-07

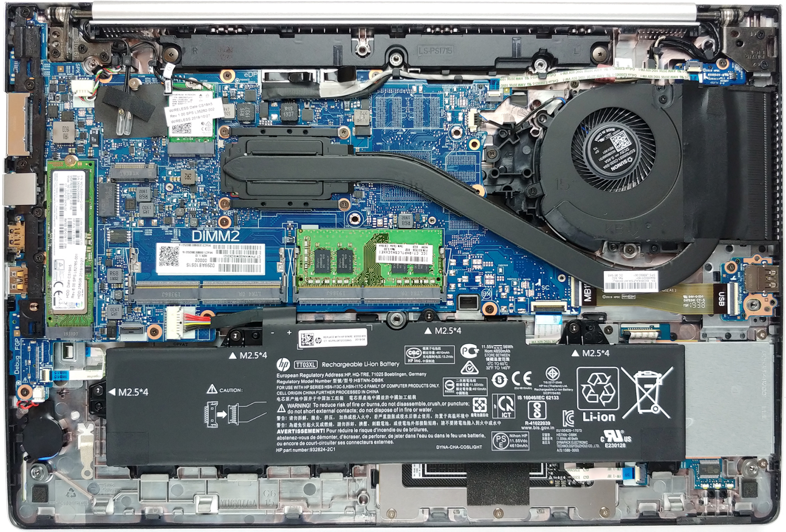
Trusted Computing. . .

- Strong integrity protection and isolation for software components
- Software attestation: cryptographically bind a software to the executing hardware
- Sealed storage: bind data to attested software

. . . and how far you can trust it

- Under which assumptions and attacker models?
- What about privacy?
- What are interesting use cases?

Computers and how they get hacked

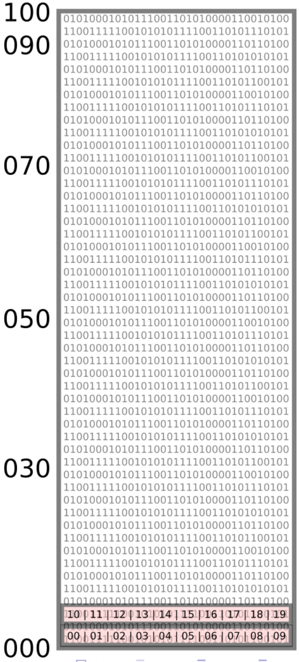


Computers and how they get hacked

Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

Computers and how they get hacked

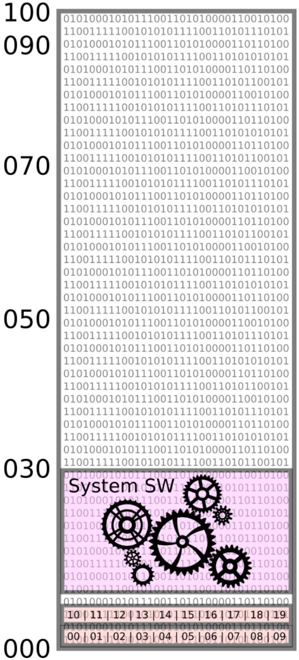
Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.



Computers and how they get hacked

Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

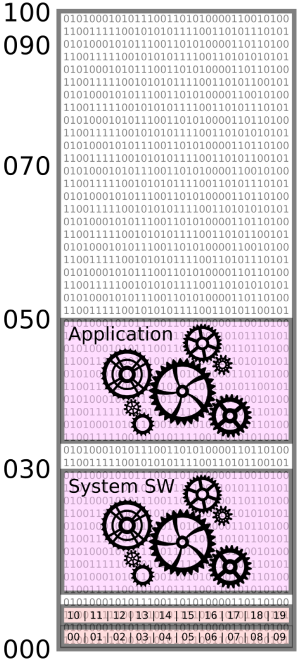
Software can instruct the **processor** to load bytes into memory.



Computers and how they get hacked

Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

Software can instruct the **processor** to load bytes into memory.

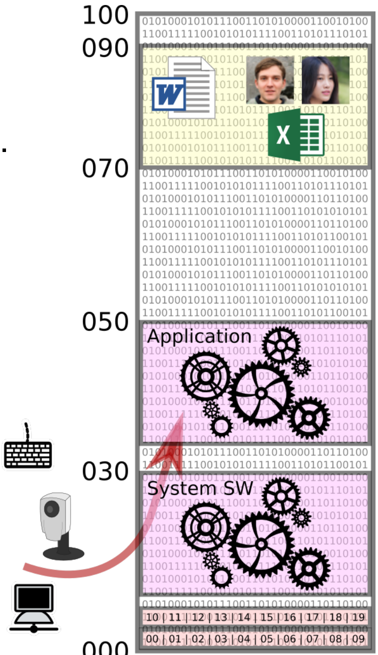


Computers and how they get hacked

Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

Software can instruct the **processor** to load bytes into memory.

Memory content can be **interpreted as code or data.**



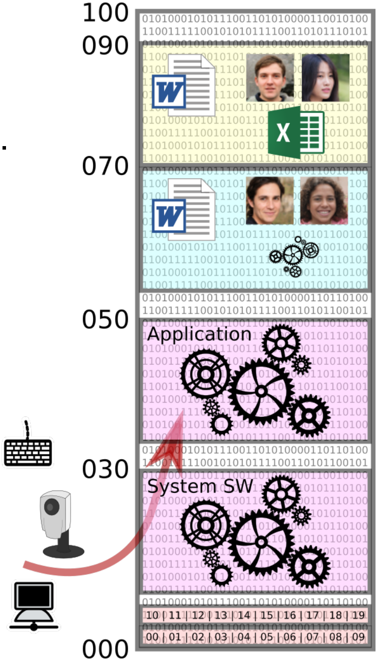
Computers and how they get hacked

Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

Software can instruct the **processor** to load bytes into memory.

Memory content can be **interpreted as code or data.**

Software is modular! Applications rely on a wealth of system software and shared code libraries to implement functionality.



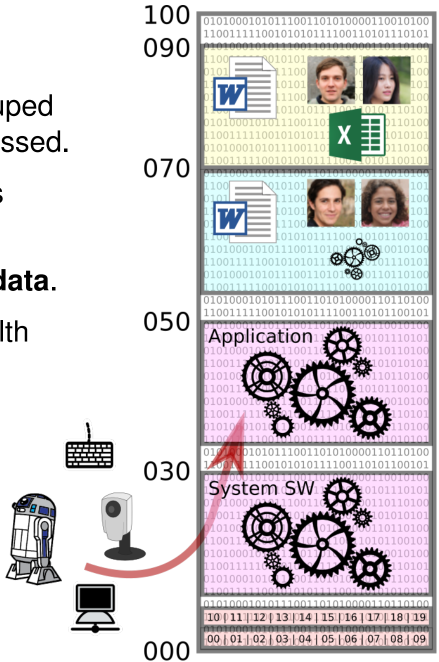
Computers and how they get hacked

Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

Software can instruct the **processor** to load bytes into memory.

Memory content can be **interpreted as code or data.**

Software is modular! Applications rely on a wealth of system software and shared code libraries to implement functionality.



Computers and how they get hacked

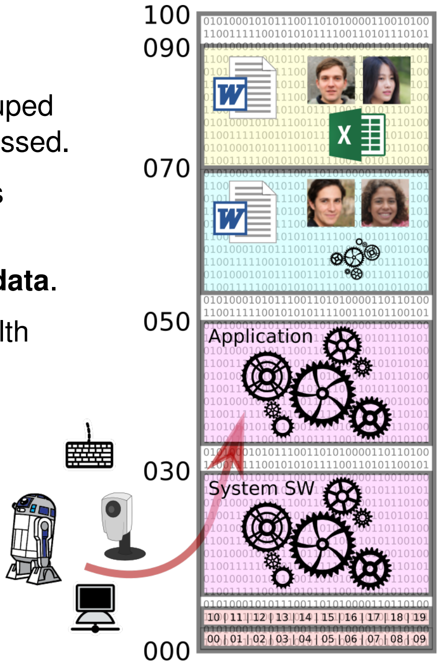
Memory contains bits. Lots of them. Bits are grouped in bytes or words, which can be individually addressed.

Software can instruct the **processor** to load bytes into memory.

Memory content can be **interpreted as code or data.**

Software is modular! Applications rely on a wealth of system software and shared code libraries to implement functionality.

Security is based on assumptions. What is trusted? What do we expect an attacker to do? What vulnerabilities are likely to exist in our code? Is there a moment when the system is supposedly secure?



Computers and how they get hacked

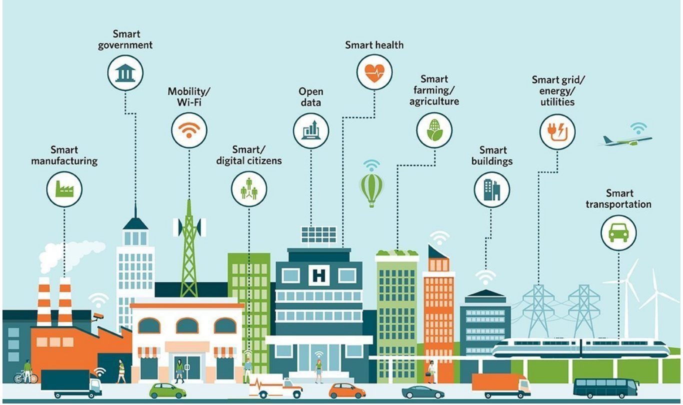
```c
/* stack1.c; https://github.com/gerasdf/InsecureProgramming */

#include <stdio.h>

int main() {
        int cookie;
        char buf[80];

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);

        if (cookie == 0x41424344) {
                printf("you win!\n");
        }
}
```

Computers and how they get hacked

```c
/* stack1.c; https://github.com/gerasdf/InsecureProgramming */

#include <stdio.h>

int main() {
        int cookie;
        char buf[80];

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);

        if (cookie == 0x41424344) {
                printf("you win!\n");
        }
}
```

Task: Compile and exploit to get “you win!”.

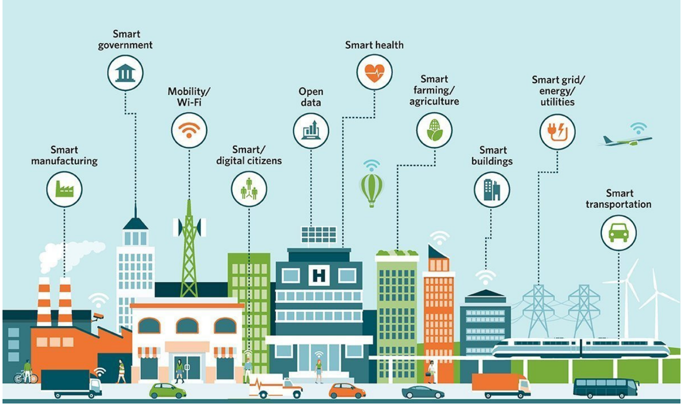
Security in Smart Environments



Vulnerabilities can hide anywhere: There are 150M lines of code in a modern car. Compartmentalisation can help with managing complexity and bug containment.

Image source: `https://internetofthingsagenda.techtarget.com/definition/smart-city`

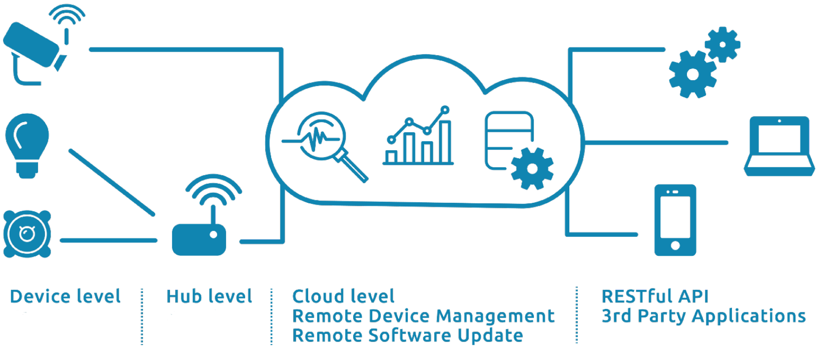
Security in Smart Environments



Infrastructure needs to be developed with safety, security and privacy in mind! What is critical infrastructure? What is critical code? Where is personal data being processed? What's the impact of failure?

Image source: `https://internetofthingsagenda.techtarget.com/definition/smart-city`

Security in Smart Environments



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 in Smart Environments

Facebook Is Breached by Hackers, Putting 50 Million Users' Data at Risk

One of the challenges for Facebook's chief executive Mark Zuckerberg is convincing users that the company handles their data responsibly.

Source: `https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html`

Security in Smart Environments

"The risks are about to get worse, because computers are being embedded into physical devices and will affect lives, not just our data."

— Bruce Schneier, [Sch18]

Security in Smart Environments

WIRED

Technology | Science | Culture | Gear | Business | Politics

Sex

The looming deluge of connected dildos is a security nightmare

Just because the teledildonics patent has expired, sex tech companies shouldn't rush to bring connectivity to their products

Source: `https://www.wired.co.uk/article/teledildonics-hacking-sex-toys` (2017)

Security in Smart Environments

WIRED Technology | Science | Culture | Gear | Business |

Meet us at WIRED Smarter this October BOOK TICKET

Smart dildos and vibrators keep getting hacked – but Tor could be the answer to safer connected sex

Connected sex toys are gathering huge amounts of data about our most intimate moments. Problem is, they're always getting hacked. Welcome to the emerging field of Onion Dildonics

Source: `https://www.wired.co.uk/article/sex-toy-bluetooth-hacks-security-fix` (2018)

Security in Smart Environments

WIRED — Inside the Cunning, Unprecedented Hack of Ukraine's Power G

KIM ZETTER SECURITY 03.03.16 07:00 AM

SHARE

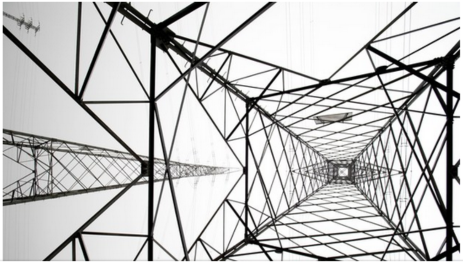
SHARE 5941

TWEET

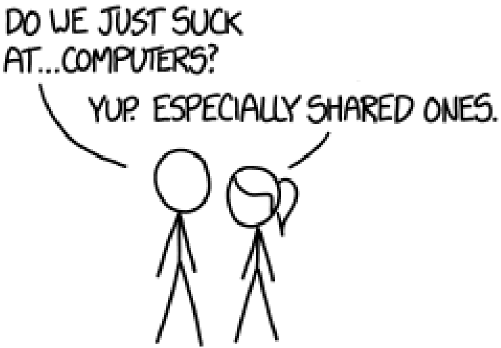
COMMENT

EMAIL

INSIDE THE CUNNING, UNPRECEDENTED HACK OF UKRAINE'S POWER GRID

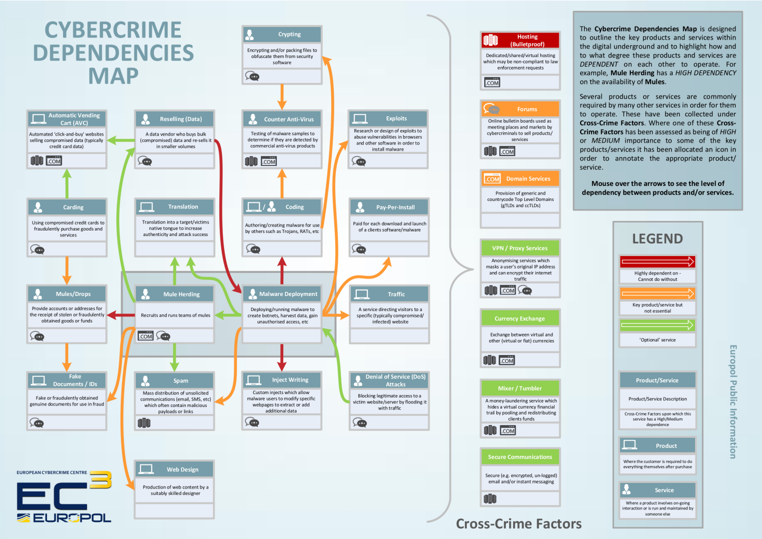


Source: `https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/`



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

Security in Smart Environments



Source: `https://www.europol.europa.eu/publications-documents/cybercrime-dependencies-map`

What can we trust?

- **Reasoning about security is about setting boundaries**
  - Which parts are considered trusted, and which parts are not? And why?
  - How far do you want to go in defending your application?
  - What kind of security is economically viable?

- **Building secure systems requires rigorous security arguments**
  - Having a good idea about what you are building.
  - Determining which attackers are considered to be in scope.
  - Analysing potential vulnerabilities, and introducing appropriate countermeasures.

- **A security argument is a rigorous argument that under a given system and adversary model, a countermeasure effectively counters a threat, or a security mechanism achieves a security goal.**

What can we trust?

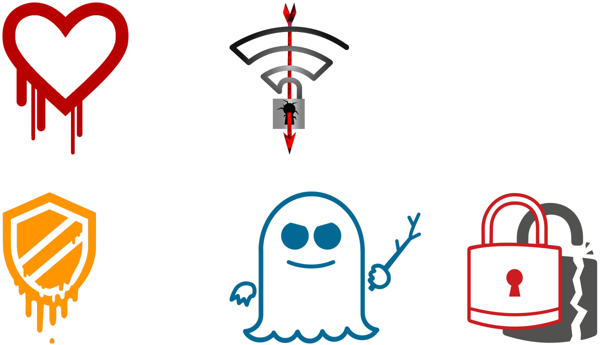
Software?

Hardware?

Supply Chains?

People?

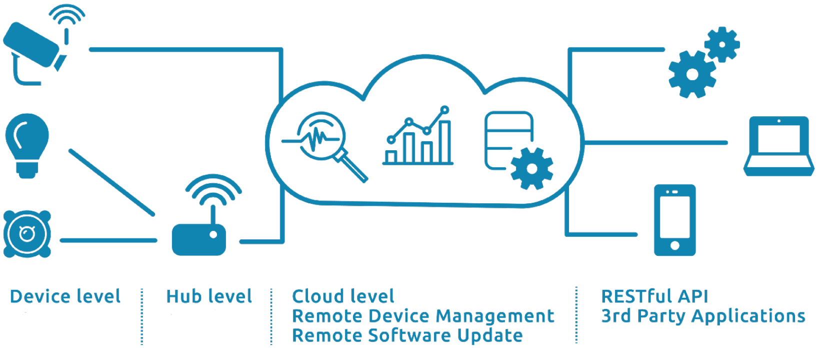
. . .



What can we trust?

Source: https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america...

Gathering Platform Requirements – A Thought Experiment



Sensors come from different vendors. Why would you trust them?
The cloud is “other people’s computers”. Why trust them?
Terminals may be used and managed by health care professionals. . .
There are huge software and hardware stacks with multiple vendors everywhere.

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

Gathering Platform Requirements – A Thought Experiment

Key elements of secure system design?

Gathering Platform Requirements – A Thought Experiment

Key elements of secure system design?

- Shift liability to 3rd party, get a cyber insurance!

Gathering Platform Requirements – A Thought Experiment

Key elements of secure system design?

- Shift liability to 3rd party, get a cyber insurance!
- Thread modelling, risk assessment, etc.
- Anonymisation of data, if possible
- Zero Trust, micro-segmentation and granular perimeters

Gathering Platform Requirements – A Thought Experiment

Key elements of secure system design?

- Shift liability to 3rd party, get a cyber insurance!
- Thread modelling, risk assessment, etc.
- Anonymisation of data, if possible
- Zero Trust, micro-segmentation and granular perimeters

How can the execution environment (= hardware) help you?

Gathering Platform Requirements – A Thought Experiment

Key elements of secure system design?

- Shift liability to 3rd party, get a cyber insurance!
- Thread modelling, risk assessment, etc.
- Anonymisation of data, if possible
- Zero Trust, micro-segmentation and granular perimeters

How can the execution environment (= hardware) help you?

- Encryption
- Isolation, Security Rings

Gathering Platform Requirements – A Thought Experiment

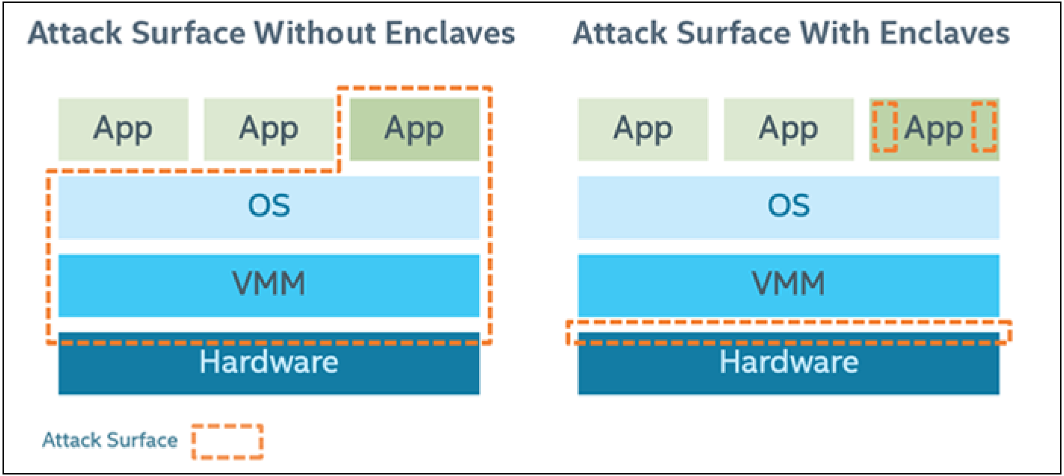
Key elements of secure system design?

- Shift liability to 3rd party, get a cyber insurance!
- Thread modelling, risk assessment, etc.
- Anonymisation of data, if possible
- Zero Trust, micro-segmentation and granular perimeters

How can the execution environment (= hardware) help you?

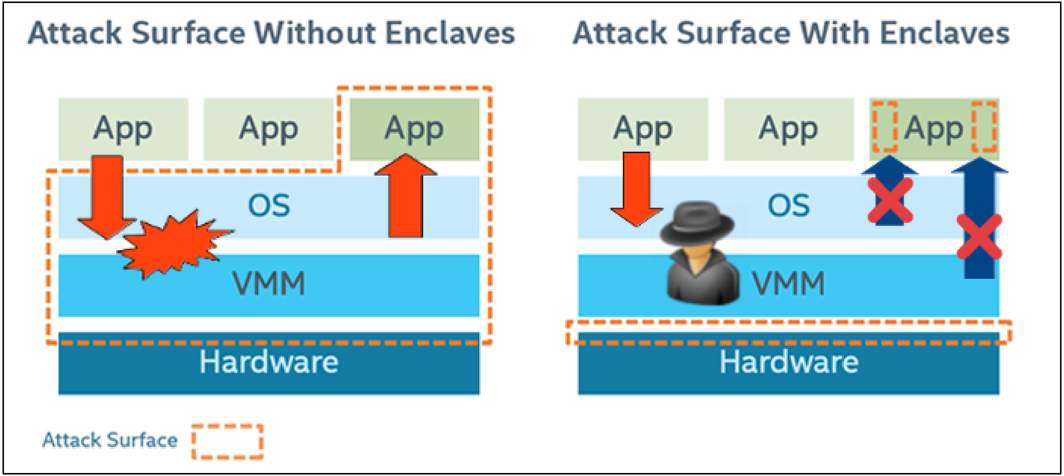
- Encryption
- Isolation, Security Rings
- Minimise Trusted Computing Base:
  remove hypervisors, OSs, libraries from TCB
  only trust hardware and your own code

Motivation: Application Attack Surface



https://software.intel.com/en-us/articles/intel-software-guard-extensions-tutorial-part-1-foundation

Motivation: Application Attack Surface



https://software.intel.com/en-us/articles/intel-software-guard-extensions-tutorial-part-1-foundation

Layered architecture $\leftrightarrow$ **hardware-only TCB**

Gathering Platform Requirements – A Real System

"We don't want the Signal service to have visibility into the social graph of Signal users. Signal is always aspiring to be as 'zero knowledge' as possible, and having a durable record of every user's friends and contacts on our servers would obviously not be privacy-preserving."

Signal

Source: `https://signal.org/blog/private-contact-discovery/`

Gathering Platform Requirements – A Real System

"We don't want the Signal service to have visibility into the social graph of Signal users. Signal is always aspiring to be as 'zero knowledge' as possible, and having a durable record of every user's friends and contacts on our servers would obviously not be privacy-preserving."

1. Run a contact discovery service in a secure SGX enclave.
2. Clients that wish to perform contact discovery negotiate a secure connection over the network all the way through the remote OS to the enclave.
3. Clients perform remote attestation to ensure that the code which is running in the *enclave is the same as the expected published open source code*.
4. Clients transmit [...] their address book to the enclave.
5. The enclave looks up a client's contacts in the set of all registered users and encrypts the results back to the client.

Source: `https://signal.org/blog/private-contact-discovery/`

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

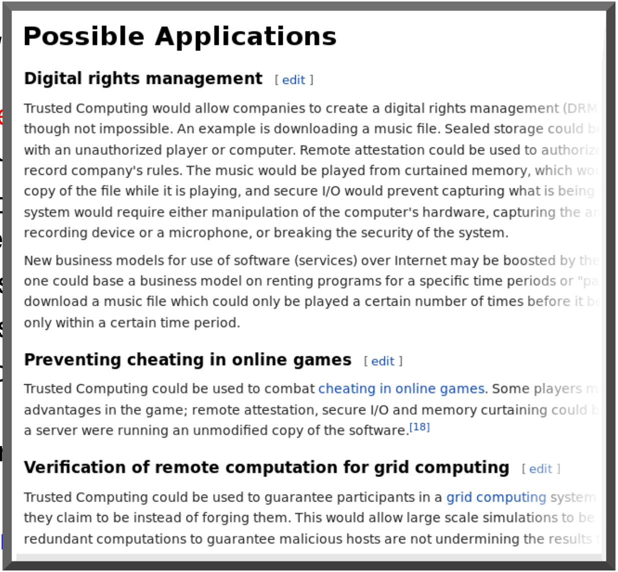
Source: https://en.wikipedia.org/wiki/Trusted_Computing

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 be
only within a certain time period.

Preventing cheating in online games [edit]

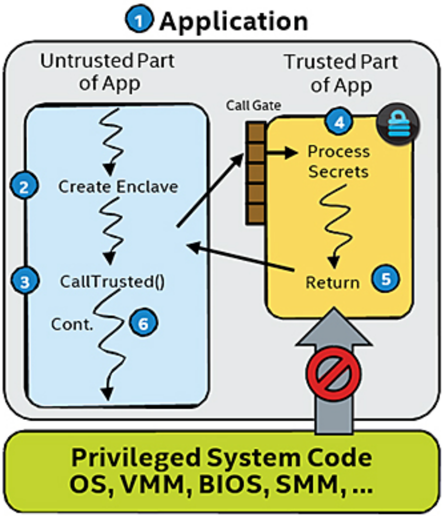
Trusted Computing could be used to combat cheating in online games. Some players m
advantages in the game; remote attestation, secure I/O and memory curtaining could b
a server were running an unmodified copy of the software.[18]

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`

Intel SGX Helicopter View



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

- Protected enclave in application's **virtual address space**
- Enclave can be entered through restrictive **call gate** only
- Provides **attestation** interface
- **Memory encryption** defends against untrusted system software and cold boot attacks

Comparing Hardware-Based Trusted Computing Architectures

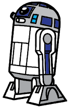
	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance	Memory Protection	Lightweight	Coprocessor	HW-Only TCB	Preemption	Dynamic Layout	Upgradeable TCB	Backwards Compatibility	Open-Source	Academic	Target ISA
AEGIS	●	●	●	●	●	○	●	○	○	●	●	●	○	●	○	●	–
TPM	○	●	●	○	●	–	◐	○	●	●	–	–	○	●	○	○	–
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

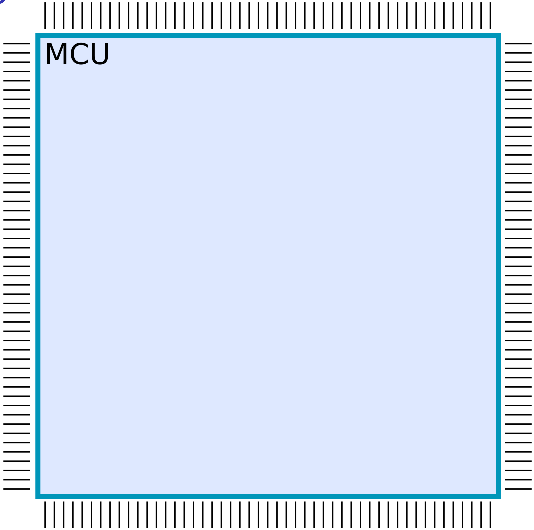
● = Yes; ◐ = Partial; ○ = No; – = Not Applicable

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

Isolation and Attestation on Light-Weight MCUs

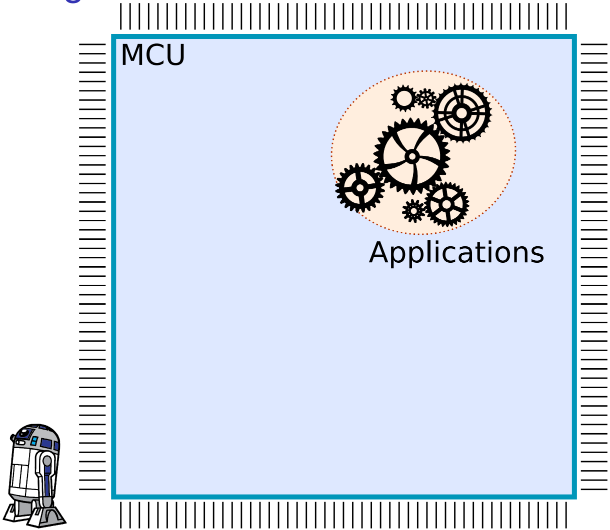
Many microcontrollers feature little security functionality





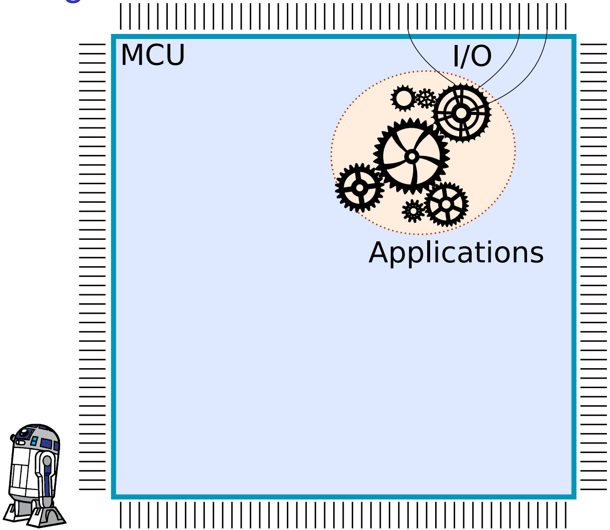
Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality



Isolation and Attestation on Light-Weight MCUs

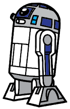
Many microcontrollers feature little security functionality

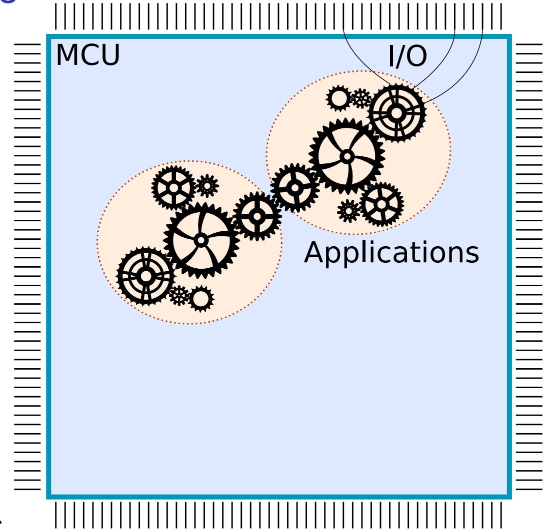


Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

- Applications share address space

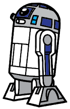


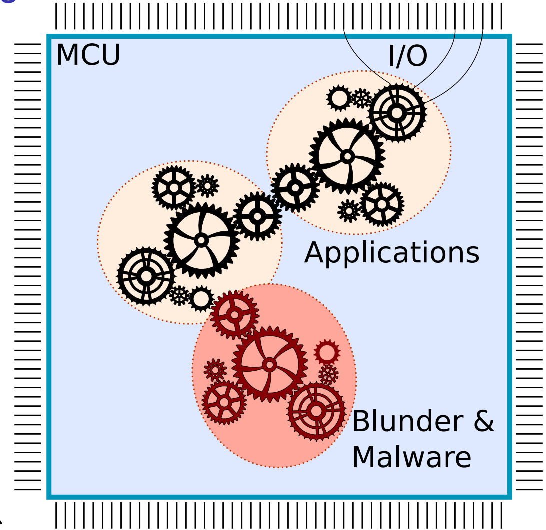


Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

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

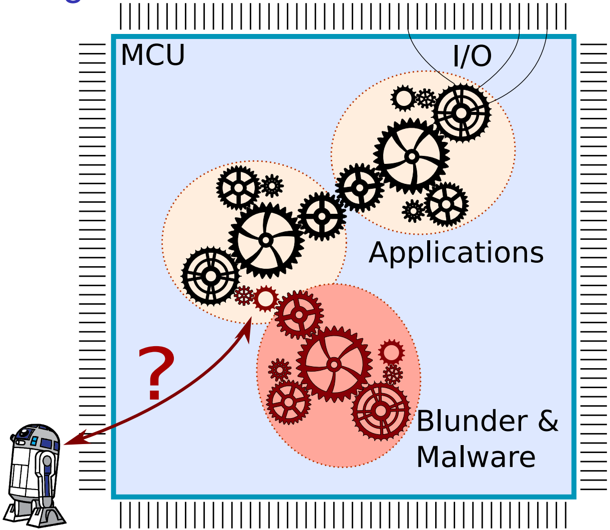




Isolation and Attestation on Light-Weight MCUs

Many microcontrollers feature little security functionality

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



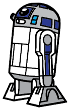
Isolation and Attestation on Light-Weight MCUs

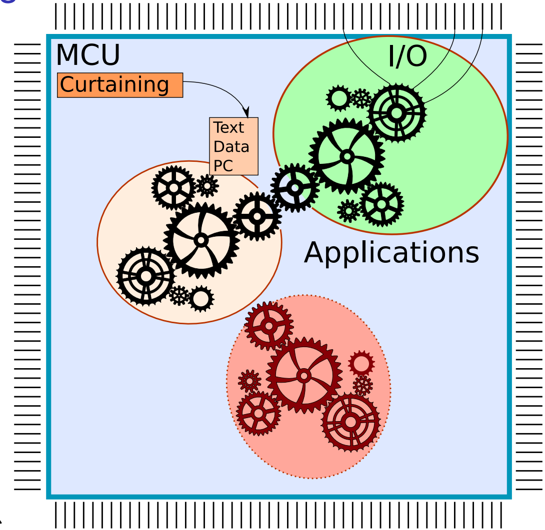
Many microcontrollers feature little security functionality

- 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





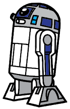
Isolation and Attestation on Light-Weight MCUs

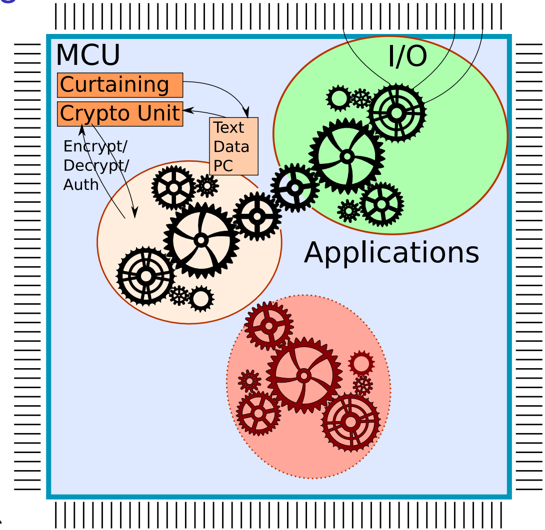
Many microcontrollers feature little security functionality

- 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





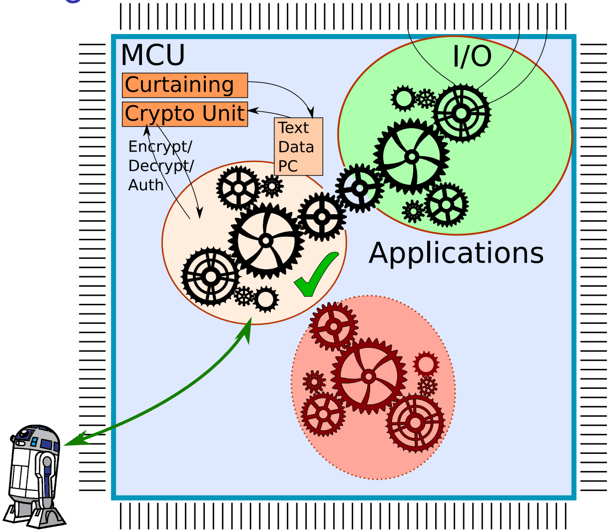
Isolation and Attestation on Light-Weight MCUs

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



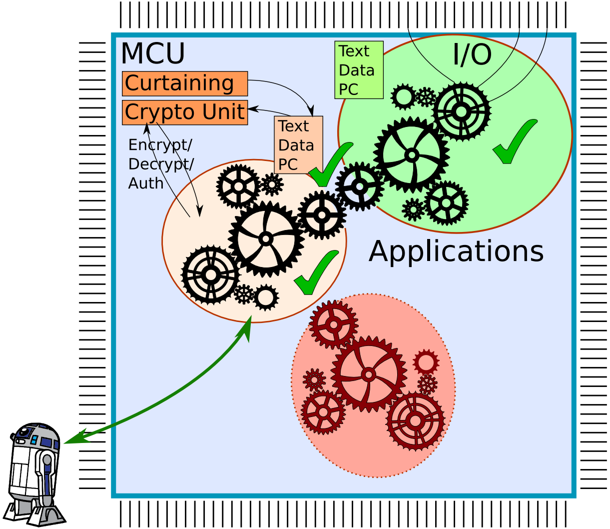
Isolation and Attestation on Light-Weight MCUs

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



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

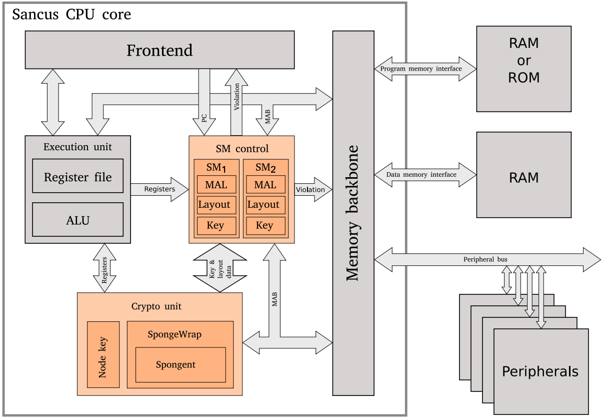
Extends openMSP430 with strong security primitives

- 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

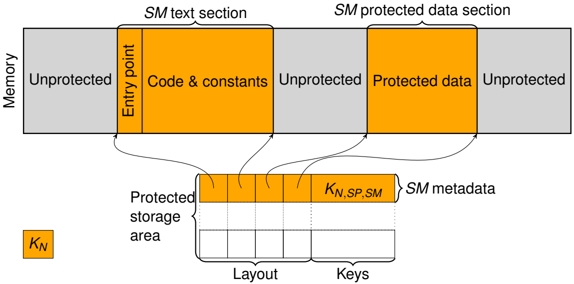
- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges

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



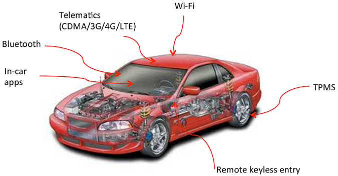
Isolated components are typically very small (< 1kLOC)

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

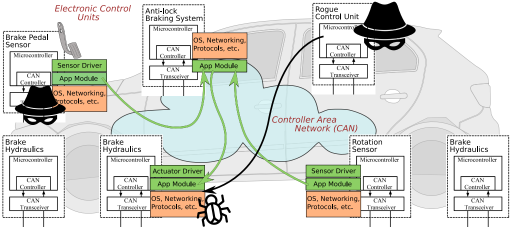
Secure Automotive Computing with Sancus [VBMP17]

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]

Authentic Execution of Distributed Event-Driven Applications

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

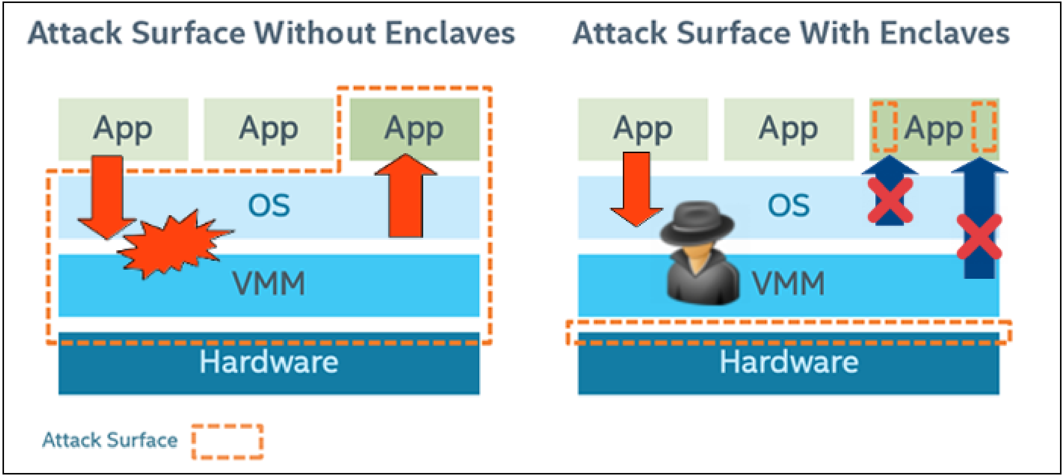
When **not** to trust your TEE…

Trusted Execution does not help you against bugs in your own (trusted) code.

Trusted Execution does not help you if you don't know what to protect.

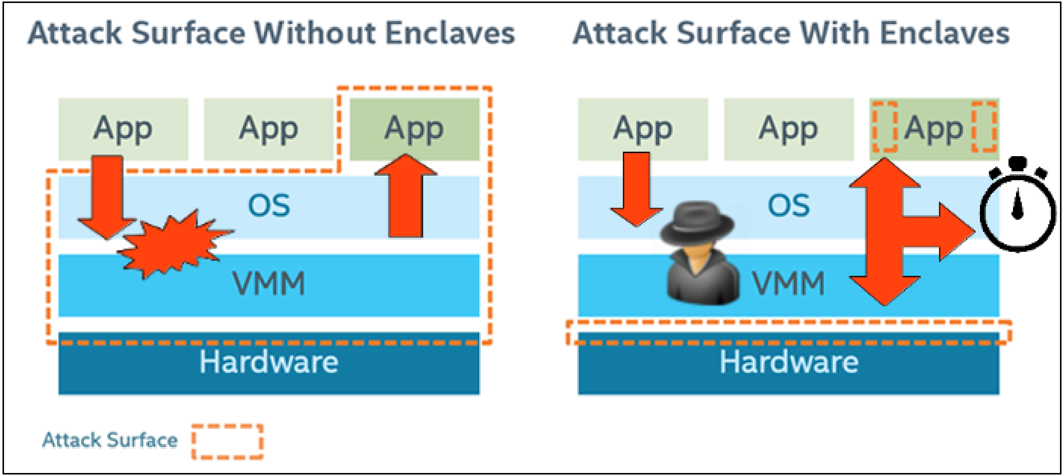
(Trusted) Execution can be observed through indirect channels and may leak secrets through these channels.

Motivation: Application Attack Surface



Layered architecture ↔ **hardware-only TCB**

Motivation: Application Attack Surface

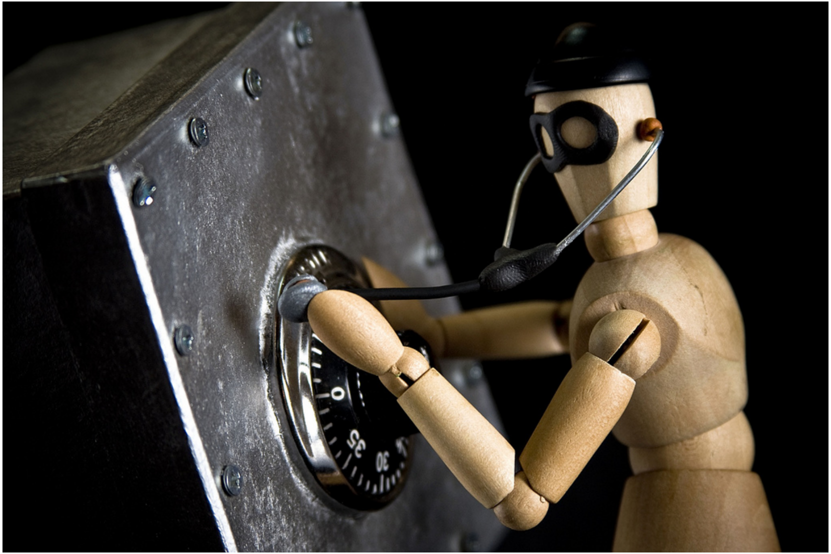


Untrusted OS → new class of powerful **side-channels**

Side-Channel Attack Principle

Source: `https://commons.wikimedia.org/wiki/File:WinonaSavingsBankVault.JPG`

Side-Channel Attack Principle



Summary

[Background]

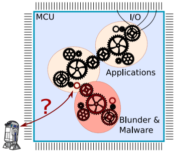
1. Software vulnerabilities are hard to eliminate and can be exploited by attackers
2. Even correct code needs protection against layer-below attacks!

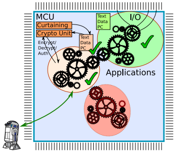
Trusted Execution Technology

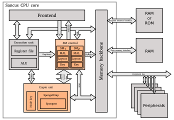
1. Strong application isolation and attestation: hardware-level security and taming complexity
2. No protection against buggy software!
3. Potential for invasive use

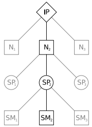
Sancus

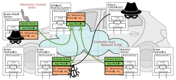
1. The Open-Source Trusted Computing Architecture
2. Built upon openMSP430 16-bit MCU, applications in IoT and embedded control systems
3. Research prototype under active development!

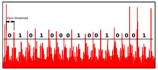












Trusted Execution for Everyone

Fortanix solves cloud security and privacy using runtime encryption technology build upon Intel SGX. `https://fortanix.com/`

SCONE enables secure execution of containers and programs using Intel SGX. `https://sconecontainers.github.io/`

Graphene-SGX: A practical library OS for unmodified applications on SGX. `https://github.com/oscarlab/graphene`

Open Enclave is an SDK for building enclave applications in C and C++. `https://github.com/Microsoft/openenclave`

Our Tutorial: Building distributed enclave applications with Sancus and SGX `https://github.com/sancus-pma/tutorial-dsn18`

The Impact of ICT…

…and why the right choice of sustainable solutions really matters.

Image sources: Electronic waste recycling in Ghana, https://en.wikipedia.org/; Martin Falbisoner, Garzweiler surface mine, https://en.wikipedia.org/; Sebastian Meyer, “Blood, Sweat and Batteries”, https://www.sebmeyer.com/

Food For Thoughts

Trusted Computing in **public communications infrastructure**

Trusted Computing to **protect critical industrial infrastructure**

Secret computations on secret data, executing on a public clouds

Hiding malware, computing cryptographic signatures on other people's computers

Thank you!

“The risks are about to get worse, because computers are being embedded into physical devices and will affect lives, not just our data.”

— Bruce Schneier, [Sch18]

Thank you! Questions?

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

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.

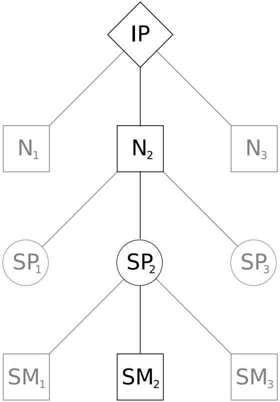
B. Schneier.
Internet hacking is about to get much worse.
The New York Times, 10 2018.

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.

Attestation and Communication with Sancus

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

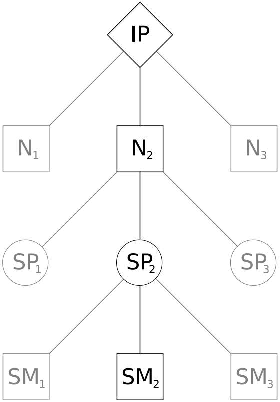
- Only N and SP can compute $K_{N,SP,SM}$
  N knows K_N and SP knows K_{SP}
- $K_{N,SP,SM}$ on N is computed after enabling isolation
  No isolation, no key; no integrity, wrong key
- Only SM on N is allowed to use $K_{N,SP,SM}$
  Through special instructions



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