An Implementation of

A High Assurance Smart Meter

Using Protected Module Architectures

Jan Tobias Mühlberg

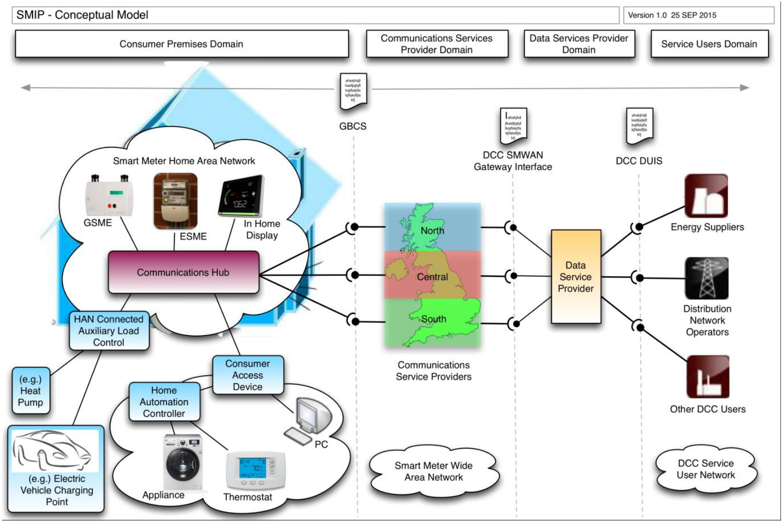
jantobias.muehlberg@cs.kuleuven.be
iMinds-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

WISTP @ Heraklion, September 2016

Joint work with: Sara Cleemput, Mustafa A. Mustafa, Jo Van Bulck, Bart Preneel, Frank Piessens

"The remote cyber attacks directed against Ukraine's electricity infrastructure were bold and successful. The cyber operation was highly synchronised and the adversary was willing to maliciously operate a SCADA system to cause power outages, followed by destructive attacks to disable SCADA and communications to the field." — [LAC16]

Smart Metering Architecture



Component overview of the UK's Smart Metering Implementation Programme (SMIP)

Image: [Dep15]

Smart Metering Architecture

Meter Components [Dep14]

- Clock
- Data Store
- Electricity measuring element
- HAN & WAN Interface
- (Aux.) Load Switch
- Random Number Generator
- User Interface
- Communication via ZigBee: HAN, WAN

Image: `smsmetering.co.uk`

Smart Metering Architecture

Meter Components [Dep14]

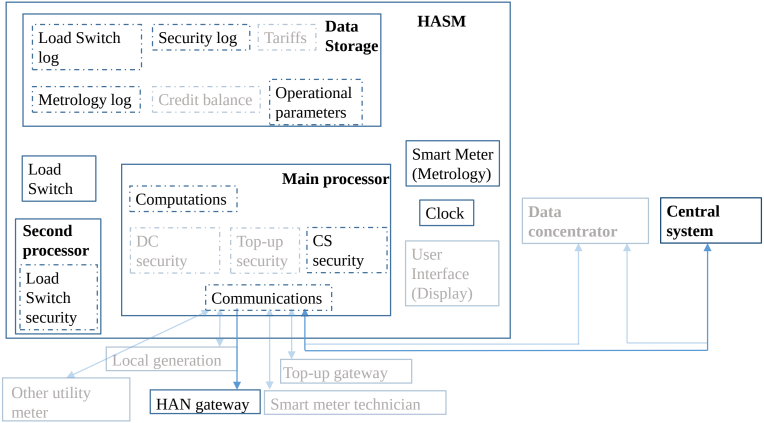
- Clock
- Data Store
- Electricity measuring element
- HAN & WAN Interface
- (Aux.) Load Switch
- Random Number Generator
- User Interface
- Communication via ZigBee: HAN, WAN

Ideal Attacker Model

- Attacker has physical access
- Attacker has no time constraints
- Attacker could be legitimate user

Image: `smsmetering.co.uk`

High Assurance Smart Metering [CMP16]



- HASM design suggests physical component separation to increase security and verifiability

? Attacker model and exact security guarantees unspecified

? Impact on implementation? May depend on platform.

Securing Distributed Embedded Computing

Can we provide strong security guarantees (confidentiality, software integrity, mutual authentication – think of Intel SGX or ARM TrustZone) for distributed embedded applications?

Securing Distributed Embedded Computing

Can we provide strong security guarantees (confidentiality, software integrity, mutual authentication – think of Intel SGX or ARM TrustZone) for distributed embedded applications?

Idea

- A distributed application is deployed as multiple protected modules on distributed computing nodes
- Modules mutually authenticate each other and exchange encrypted messages
- Protected driver modules facilitate I/O

Securing Distributed Embedded Computing

Can we provide strong security guarantees (confidentiality, software integrity, mutual authentication – think of Intel SGX or ARM TrustZone) for distributed embedded applications?

Idea

- A distributed application is deployed as multiple protected modules on distributed computing nodes
- Modules mutually authenticate each other and exchange encrypted messages
- Protected driver modules facilitate I/O
- Scenario: Smart Meter, Load Switch, Central System, Home Area Network

Securing Distributed Embedded Computing

Can we provide strong security guarantees (confidentiality, software integrity, mutual authentication – think of Intel SGX or ARM TrustZone) for distributed embedded applications?

Idea

- A distributed application is deployed as multiple protected modules on distributed computing nodes
- Modules mutually authenticate each other and exchange encrypted messages
- Protected driver modules facilitate I/O
- Scenario: Smart Meter, Load Switch, Central System, Home Area Network

Security Guarantees

- We get a chain of mutual trust among application modules
- Security of each module independent from other software
- Output is guaranteed to be reproducible, based on the applications source code and the input events

Protected Module Architectures

PMAs provide

- Strong isolation of software components in Protected Modules
  - $\rightarrow$ Confidentiality
  - $\rightarrow$ Code Integrity and Control Flow Integrity
- Remote attestation
  - $\rightarrow$ e.g. Load Switch and meter core
- Secure remote communication
  - $\rightarrow$ No spoofing or replay of signals
- Minimal hardware-only TCB

Protected Module Architectures

PMAs provide

- Strong isolation of software components in Protected Modules
  - → Confidentiality
  - → Code Integrity and Control Flow Integrity
- Remote attestation
  - → e.g. Load Switch and meter core
- Secure remote communication
  - → No spoofing or replay of signals
- Minimal hardware-only TCB
- Server/Desktop: Intel SGX [MAB+13], ARM TrustZone [AF04], TrustVisor [MLQ+10], Fides [SP12]
- Embedded: SMART [EFPT12], TrustLite [KSSV14], TyTAN [BEMS+15], Sancus [NAD+13]

Protected Module Architectures

PMAs provide

- Strong isolation of software components in Protected Modules
  - → Confidentiality
  - → Code Integrity and Control Flow Integrity
- Remote attestation
  - → e.g. Load Switch and meter core
- Secure remote communication
  - → No spoofing or replay of signals
- Minimal hardware-only TCB
- Server/Desktop: Intel SGX [MAB+13], ARM TrustZone [AF04], TrustVisor [MLQ+10], Fides [SP12]
- Embedded: SMART [EFPT12], TrustLite [KSSV14], TyTAN [BEMS+15], Sancus [NAD+13]

A Partial Solution to Software Security on Lightweight Embedded Controllers

- There is no free lunch!

Sancus: A PMA for Embedded Devices and IoT

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]
- openMSP430 at `http://opencores.org`

Sancus: A PMA for Embedded Devices and IoT

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]
- openMSP430 at `http://opencores.org`

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O

Sancus: A PMA for Embedded Devices and IoT

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]
- openMSP430 at `http://opencores.org`

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity

Sancus: A PMA for Embedded Devices and IoT

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]
- openMSP430 at `http://opencores.org`

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity
- Even without an attacker: bugs and software ageing

Sancus: A PMA for Embedded Devices and IoT

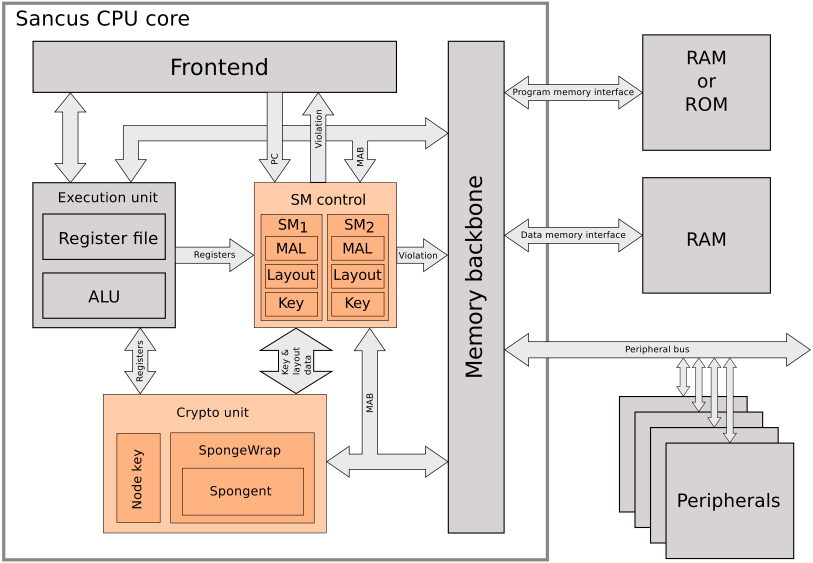
TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]
- openMSP430 at `http://opencores.org`

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity
- Even without an attacker: bugs and software ageing
- We develop an Protected Module Architecture on top of the openMSP430: Sancus

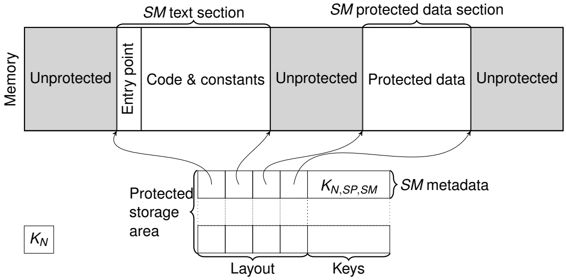
Sancus: A PMA for Embedded Devices and IoT



Sancus: A PMA for Embedded Devices and IoT

Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

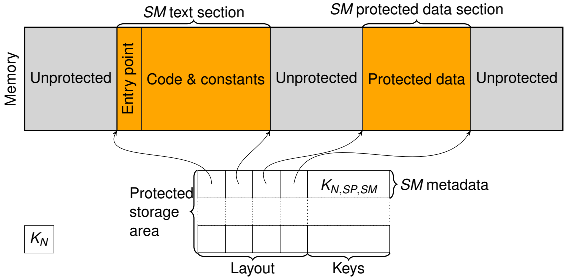


Sancus: A PMA for Embedded Devices and IoT

Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Public and protected sections

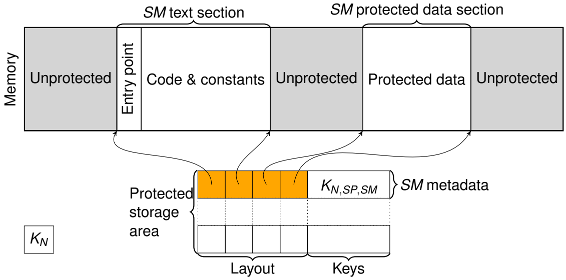


Sancus: A PMA for Embedded Devices and IoT

Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module layout

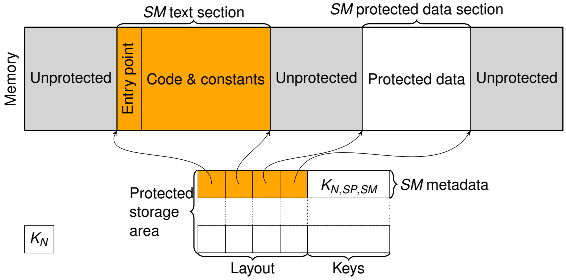


Sancus: A PMA for Embedded Devices and IoT

Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module identity

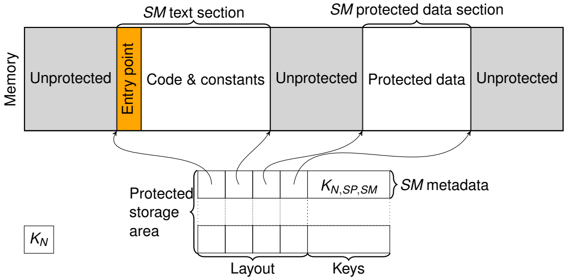


Sancus: A PMA for Embedded Devices and IoT

Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module entry point

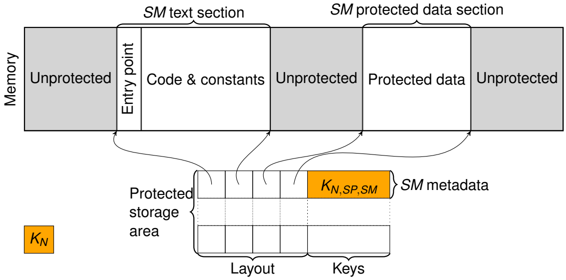


Sancus: A PMA for Embedded Devices and IoT

Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module keys



Sancus: A PMA for Embedded Devices and IoT

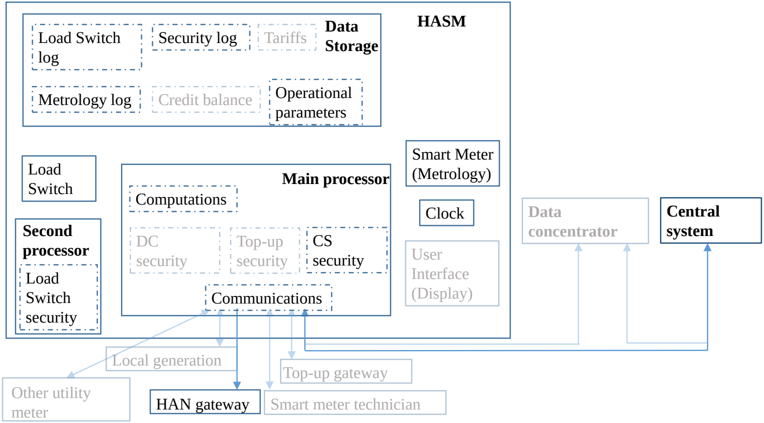
Sancus [NAD+13] enables strong isolation, attestation and secure communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures
- Provides efficient cryptographic primitives and key handling
- Reference implementation based on the openMSP430

How to program it?

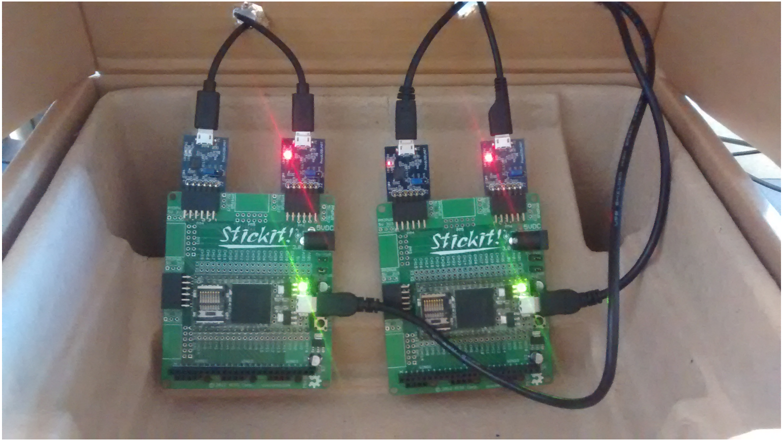
- Isolation vs. shared memory communication: use of PMA must be considered early in development process
- Extensions of C for reactive protected modules, implemented in LLVM
- Deployment, key management and validation is automated
- OS, network stack, module loader, etc.: Contiki OS

High Assurance Smart Metering [CMP16]

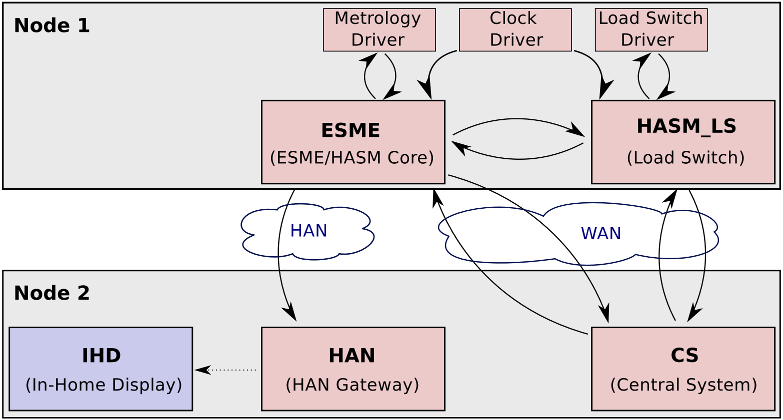


- HASM design suggests physical component separation to increase security and verifiability
- ? Attacker model and exact security guarantees unspecified
- ? Impact on implementation? May depend on platform.

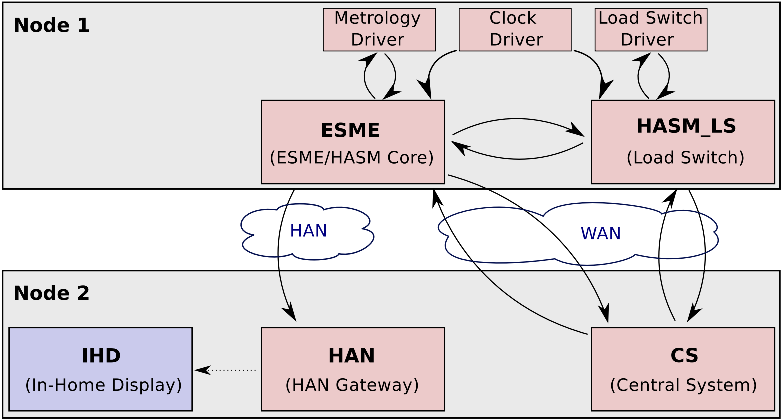
High Assurance Smart Metering with Sancus



High Assurance Smart Metering with Sancus



High Assurance Smart Metering with Sancus



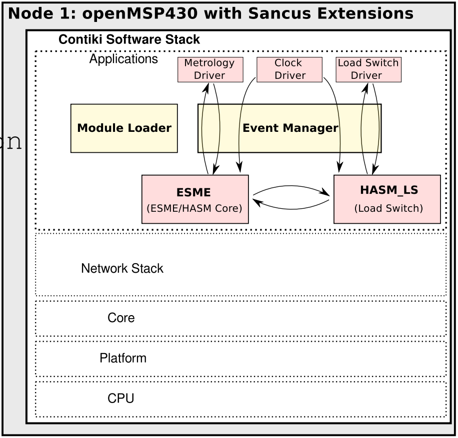
Our Attacker Model

- Attacker has no physical access
- Attacker controls network, Dolev-Yao
- Attacker controls all software except protected application modules: scheduler, network stack, module loader

Authentic Execution & Reactive Programming

Untrusted SW @ Infrastructure

- **Module Loader**: `LoadModule`, `CallEntry`
- **Event Manager**: `AddConnection`, `HandleLocalEvent`, `HandleRemoteEvent`



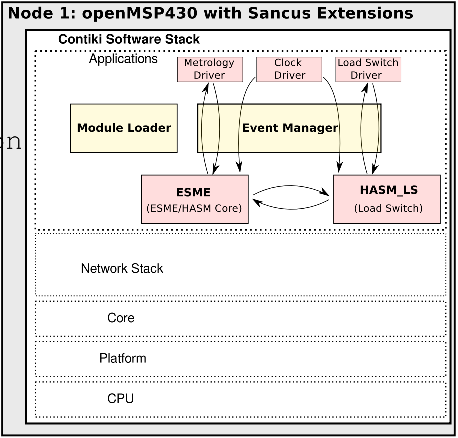
Authentic Execution & Reactive Programming

Untrusted SW @ Infrastructure

- **Module Loader**: `LoadModule`, `CallEntry`
- **Event Manager**: `AddConnection`, `HandleLocalEvent`, `HandleRemoteEvent`

Trusted SW @ Software Provider

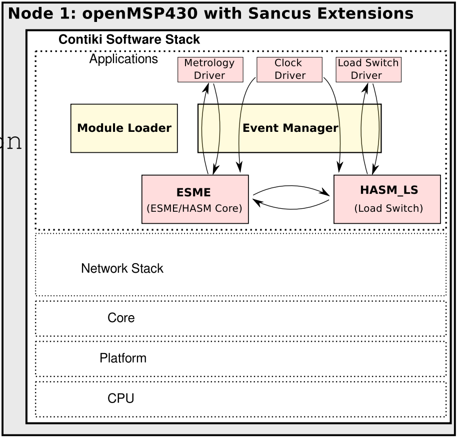
- Compiler, deployment script



Authentic Execution & Reactive Programming

Untrusted SW @ Infrastructure

- **Module Loader**: `LoadModule`, `CallEntry`
- **Event Manager**: `AddConnection`, `HandleLocalEvent`, `HandleRemoteEvent`



Trusted SW @ Software Provider

- Compiler, deployment script

Deployment & Use

- Compile modules, Load modules
- `AddConnection` & distribute keys ($\Longrightarrow$ attestation)
- **Event Manager** now dispatches events to registered modules; modules drop events if authentication or decryption fails

→ Strong integrity but no availability

High Assurance Smart Metering with Sancus

Component	Source LOC	Binary Size (B)	Deployed
Contiki	38386	16316	per node
Event manager	598	1730	per node
Module loader	906	1959	per node
ESME/HASM Core	119	2573	once
Load Switch	79	2377	once
HAN Gateway	30	1599	once
Central System	63	2069	once
Deployment Descriptor	90	n/a	n/a
Run-Time SW TCB	**381**	8618	

Summary & Conclusions

Results

- Critical software components are resilient against attacks
- Security of each module independent from other software
- We have a chain of mutual trust among distributed application modules
- Output is guaranteed to be reproducible, wrt. application's source code and inputs

Summary & Conclusions

Results

- Critical software components are resilient against attacks
- Security of each module independent from other software
- We have a chain of mutual trust among distributed application modules
- Output is guaranteed to be reproducible, wrt. application's source code and inputs
- Run-time TCB is drastically reduced! Certainly verifiable.
- Applicable in other domains: automotive, medical, ...

Summary & Conclusions

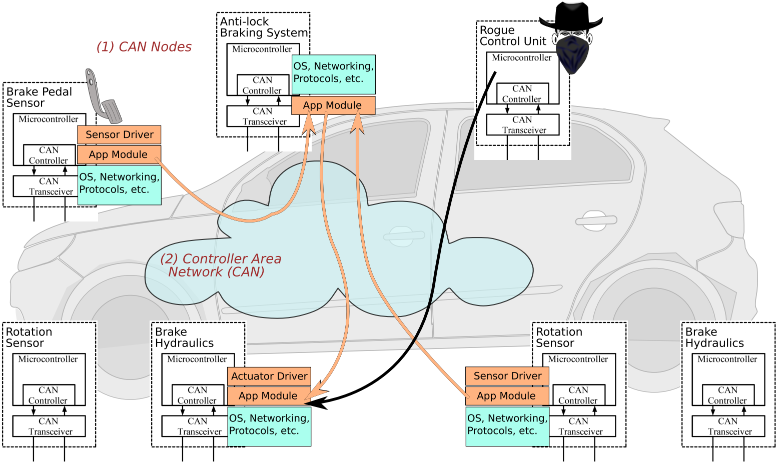
Results

- Critical software components are resilient against attacks
- Security of each module independent from other software
- We have a chain of mutual trust among distributed application modules
- Output is guaranteed to be reproducible, wrt. application's source code and inputs
- Run-time TCB is drastically reduced! Certainly verifiable.
- Applicable in other domains: automotive, medical, ...

Some drawbacks:

- Increased chip size and power consumption
- Isolation vs. shared memory communication
- Availability and real-time in the presence of adversaries
- Re-implementing an existing set of applications as SMs is often not straight-forward and leads to performance issues

Ongoing Research: Secure Automotive Computing



Legitimate nodes of a vehicular communication network run safety-critical applications with Sancus' protection (orange), which mutually authenticate each other and are protected against code-abuse attacks. Rogue nodes cannot interfere with security (but may harm availability), node takeover is very difficult (if not impossible).

Ongoing Research

IoT Trust Assessment: implement light-weight and secure inspection components that integrate seamlessly with existing deployment scenarios [MNP15]

Programming Models and Infrastructure: guarantee authenticity and integrity properties of event-driven distributed applications; integration with server/desktop PMA; secure compilation to PMAs [BNMP15, PAS$^+$15, vGSMP16, PDMS16]

Secure I/O: Trusted Paths between microcontrollers attached to sensors and actuators [NAD$^+$13, MCM$^+$16]

Safe Languages and Formal Verification: Protected Modules must be free of vulnerabilities (e.g. memory safety, information flow) to guarantee safe operation [vGSMP16, PDMS16, AJP15]

Availability and Real-Time Guarantees: to control reactive safety-critical components in, e.g. automotive, avionic and medical domains [VBNMP16]

Thank you! Questions?

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

Further reading: "An Implementation of a High Assurance Smart Meter using Protected Module Architectures", Mühlberg et al., WISTP 2016, pages 53–69.

References I

T. Alves and D. Felton.
TrustZone: Integrated hardware and software security.
ARM white paper, 3(4):18–24, 2004.

P. Agten, B. Jacobs, and F. Piessens.
Sound modular verification of c code executing in an unverified context.
In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, pp. 581–594. ACM, 2015.

F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl.
TyTAN: Tiny trust anchor for tiny devices.
In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pp. 34:1–34:6, New York, 2015. ACM.

J. V. Bulck, J. Noorman, J. T. Mühlberg, and F. Piessens.
Secure resource sharing for embedded protected module architectures.
In *WISTP '15*, vol. 9311 of *LNCS*, pp. 71–87, Heidelberg, 2015. Springer.

S. Cleemput, M. A. Mustafa, and B. Preneel.
High assurance smart metering.
In *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 294–297, 2016.

Department of Energy and Climate Change.
Smart metering implementation programme – smart metering equipment technical specifications; version 1.58, 2014.
https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/381535/SMIP_E2E_SMETS2.pdf.

References II

Department of Energy and Climate Change.
Smart metering implementation programme – end to end technical architecture; version 1.1, 2015.
https://www.smartenergycodecompany.co.uk/docs/default-source/sec-documents/sec-documents/technical-architecture-document.pdf?sfvrsn=5.

K. Eldefrawy, A. Francillon, D. Perito, and G. Tsudik.
SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust.
In *NDSS 2012, 19th Annual Network and Distributed System Security Symposium, San Diego, USA*, 2012.

P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan.
Trustlite: A security architecture for tiny embedded devices.
In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, pp. 10:1–10:14. ACM, 2014.

R. M. Lee, M. J. Assante, and T. Conway.
Analysis of the cyber attack on the Ukrainian power grid – defense use case, 2016.
https://ics.sans.org/blog/2016/03/22/e-isac-and-sans-report-on-the-ukrainian-grid-attack.

F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar.
Innovative instructions and software model for isolated execution.
In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pp. 10:1–10:1. ACM, 2013.

J. T. Mühlberg, S. Cleemput, M. A. Mustafa, J. V. Bulck, B. Preneel, and F. Piessens.
An implementation of a high assurance smart meter using protected module architectures.
In *WISTP '16*, vol. 9895 of *LNCS*, pp. 53–69, Heidelberg, 2016. Springer.

References III

J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig.
Trustvisor: Efficient tcb reduction and attestation.
In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pp. 143–158. IEEE, 2010.

J. T. Mühlberg, J. Noorman, and F. Piessens.
Lightweight and flexible trust assessment modules for the Internet of Things.
In *ESORICS '15*, vol. 9326 of *LNCS*, pp. 503–520, Heidelberg, 2015. Springer.

J. Noorman, P. Agten, W. Daniels, R. Strackx, A. Van Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens.
Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base.
In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pp. 479–494, Berkeley, CA, USA, 2013. USENIX Association.

M. Patrignani, P. Agten, R. Strackx, B. Jacobs, D. Clarke, and F. Piessens.
Secure compilation to protected module architectures.
ACM Trans. Program. Lang. Syst., 37(2):6:1–6:50, 2015.

F. Piessens, D. Devriese, J. T. Mühlberg, and R. Strackx.
Security guarantees for the execution infrastructure of software applications.
In *Cybersecurity Development Conference (SecDev '16)*, New York, 2016. IEEE.
To appear.

M. Seaman.
Powering an MSP430 from a single battery cell.
Technical Report SLAA398, Texas Instruments, 2008.

References IV

R. Strackx and F. Piessens.
Fides: Selectively hardening software application components against kernel-level or process-level malware.
In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pp. 2–13, New York, NY, USA, 2012. ACM.

R. Strackx, F. Piessens, and B. Preneel.
Efficient isolation of trusted subsystems in embedded systems.
In *Security and Privacy in Communication Networks*, vol. 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 344–361. Springer, 2010.

J. Van Bulck, J. Noorman, J. T. Mühlberg, and F. Piessens.
Towards availability and real-time guarantees for protected module architectures.
In *MASS '16, MODULARITY Companion 2016*, pp. 146–151, New York, 2016. ACM.

N. van Ginkel, R. Strackx, J. T. Mühlberg, and F. Piessens.
Towards safe enclaves.
In *4th Workshop on Hot Issues in Security Principles and Trust (HotSpot '16)*, pp. 33–48. IFIP, 2016.