Authentic Execution of Distributed Event-Driven Applications with a Small TCB

Job Noorman, **Jan Tobias Mühlberg** and Frank Piessens
jantobias.muehlberg@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

STM'17, Oslo, September 2017

Authentic Execution of Distributed Event-Driven Applications

Authentic Execution of Distributed Event-Driven Applications

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 Event-Driven Applications

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 Event-Driven Applications

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 infrastructure
- Each module provides input and output channels that transparently connect to other modules' channels
- Physical events enter or leave the application through I/O channels
- Multiple distrusting applications share the infrastructure

Authentic Execution of Distributed Event-Driven Applications

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 infrastructure
- Each module provides input and output channels that transparently connect to other modules' channels
- Physical events enter or leave the application through I/O channels
- Multiple distrusting applications share the infrastructure

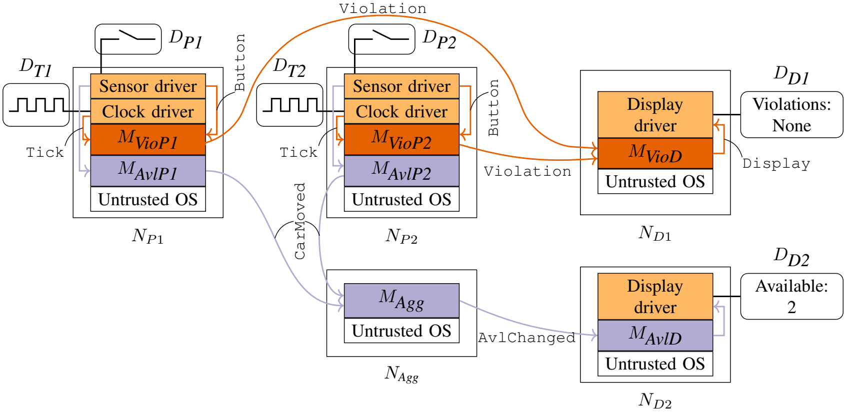
With a small (run-time) Trusted Computing Base

- Code that is not part of an application cannot interfere with that application

Authentic Execution of Distributed Event-Driven Applications

A (complicated & unrealistic) Example Scenario

A car park with 2 parking positions and 2 monitoring applications: Violation monitor A_{Vio} and position availability monitor A_{Avl}



The Shared Infrastructure

Requirements: some form of Trusted Computing

- Authenticated communication
- Software & device attestation
- Secure I/O

The Shared Infrastructure

Requirements: some form of Trusted Computing

- Authenticated communication
- Software & device attestation
- Secure I/O

Implementation options

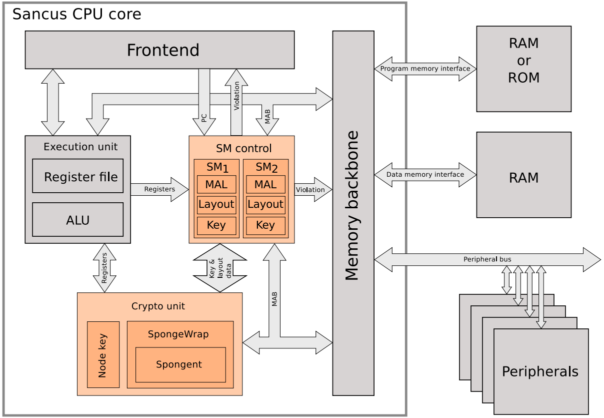
- Intel SGX (no I/O)
- ARM TrustZone (only one trusted world)
- Sancus (lightweight & embedded, no complex computations)
- ...

Embracing heterogeneity

- Application modules can exploit specific features of an architecture, as long as authenticated communication and attestation are available & compatible
- Prototype for Sancus

Sancus: A Security Architecture for IoT [NAD+13, NVBM+17]

- **Extends TI's MSP430 with strong security primitives**
  - Software Component Isolation
  - Cryptography & Attestation
  - Secure I/O through isolation of MMIO ranges
- **Efficient**
  - Authentication in μs
  - 6% increased power consumption
- **Cryptographic key hierarchy for software attestation**

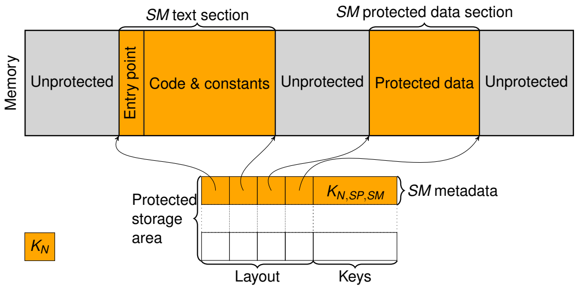


- Isolated components are typically very small ($<$ 1kLOC)
- Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

Sancus: A Security Architecture for IoT [NAD+13, NVBM+17]

- **Extends TI's MSP430 with strong security primitives**
  - Software Component Isolation
  - Cryptography & Attestation
  - Secure I/O through isolation of MMIO ranges
- **Efficient**
  - Authentication in μs
  - 6% increased power consumption
- **Cryptographic key hierarchy for software attestation**

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

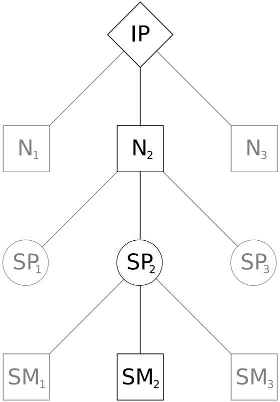


- Isolated components are typically very small ($<$ 1kLOC)
- Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

Attestation and Communication

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

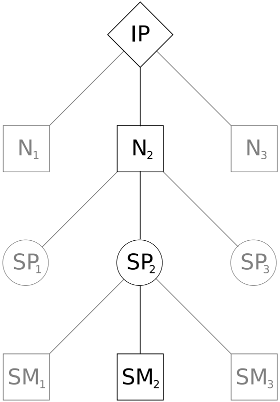
- Only N and SP can calculate $K_{N,SP,SM}$
  N knows K_N and SP knows K_{SP}
- $K_{N,SP,SM}$ on N is calculated after enabling isolation
  No isolation, no key; no integrity, wrong key
- Only SM on N is allowed to use $K_{N,SP,SM}$
  Through special instructions



Attestation and Communication

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

- Only *N* and *SP* can calculate $K_{N,SP,SM}$
  N knows K_N and *SP* knows K_{SP}
- $K_{N,SP,SM}$ on *N* is calculated 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



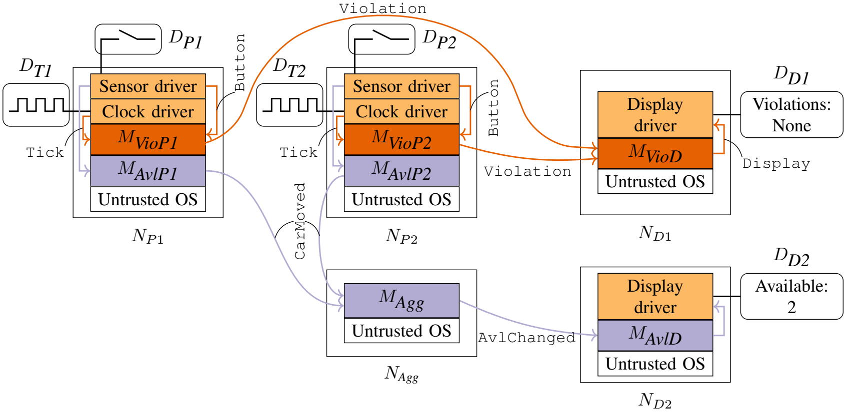
Remote attestation and secure communication by Authenticated Encryption with Associated Data

- 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

Authentic Execution of Distributed Event-Driven Applications

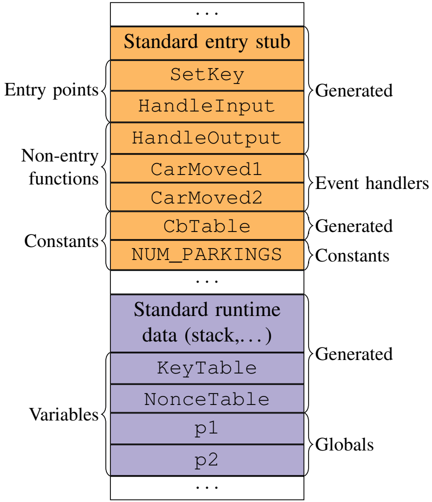
A (complicated & unrealistic) Example Scenario

A car park with 2 parking positions and 2 monitoring applications:
Violation monitor A_{Vio} and position availability monitor A_{Avl}



Authentic Execution of Distributed Event-Driven Applications

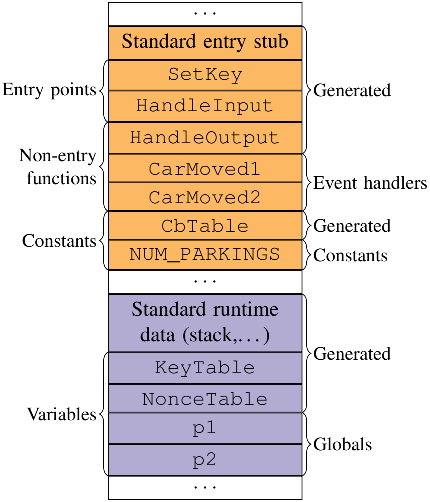
```
module AvlP1
on Button(pressed):
  CarMoved(entered)
module AvlP2
# Similar to AvlP1

module Agg
on CarMoved1(entered):
  p1 = entered
  num_avl = NUM_PARKINGS
  if (p1): num_avl = num_avl - 1
  if (p2): num_avl = num_avl - 1
  AvlChanged(num_avl)
on CarMoved2(entered):
  # Similar to CarMoved1

module AvlD
on AvlChanged(num_avl)
  Display(num_avl)
```



Authentic Execution of Distributed Event-Driven Applications

Developer / Software Provider provides:

- Source code of all application modules
- Deployment descriptor
  - Mapping modules to nodes
  - Configuration of communication channels and I/O channels



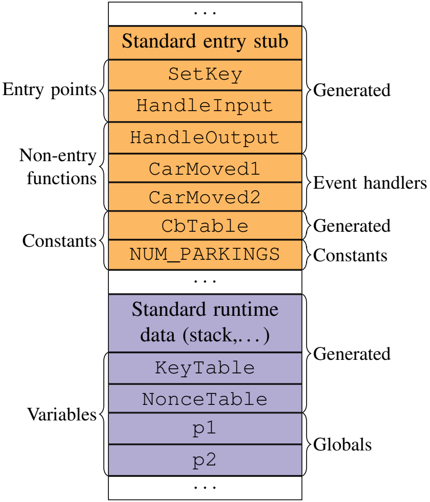
Authentic Execution of Distributed Event-Driven Applications

Developer / Software Provider provides:

- Source code of all application modules
- Deployment descriptor
  - Mapping modules to nodes
  - Configuration of communication channels and I/O channels

Toolchain:

- Compilation and linking
  - Generate code to configure channels, communication keys, and to encrypt and decrypt events
  - Prepare secure linking with I/O modules
- Deployment:
  - Load and activate modules
  - Configure communication channels



Deployment

Deployment of application code

1. Compile all source modules to PMs
2. Load them on the node specified in the deployment descriptor
3. Generate cryptographic keys for each connection
4. Send keys to the sending and receiving modules, encrypted by the appropriate module keys $K_{N,SP,SM}$

Connections to physical I/O channels

5. Generate keys for connections to physical outputs, send them to application module and protected driver module and attest success
6. Generate keys for connections to physical inputs and send them to application module and protected driver module

Infrastructure that implements deployment is trusted.

Protected Driver Modules

Driver modules have to satisfy properties that depend on the desired security guarantee:

- E.g. Integrity:
  - Applications must be able to take exclusive ownership of protected driver modules of output devices
  - But protected driver modules for input devices can broadcast input events to all applications with only integrity protection

  $\rightarrow$ Integrity protected channel from physical inputs to physical outputs
- Confidentiality is dual
- Properties to be checked by the application deployer / *SP*
- Architectural support (i.e. hardware) is crucial

Security Properties

Remember our security objective:

- We can explain every observed output event based on a trace of actual input events and the (source) code of the application
- Security properties that hold for the source code should hold at run-time in the presence of arbitrary attackers within the attacker model

$\rightarrow$ `https://people.cs.kuleuven.be/~jantobias.muehlberg/stm17/`

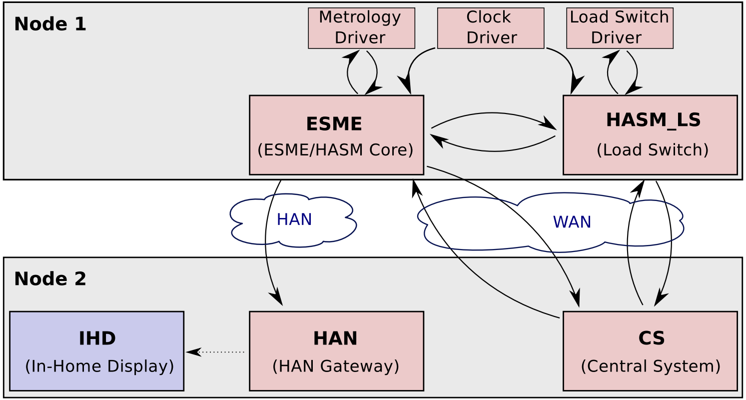
This holds for arbitrary safety properties

- E.g. "No violation is signalled on the display unless a car entered and stayed there for $> n$ clock ticks"

But it does not hold for:

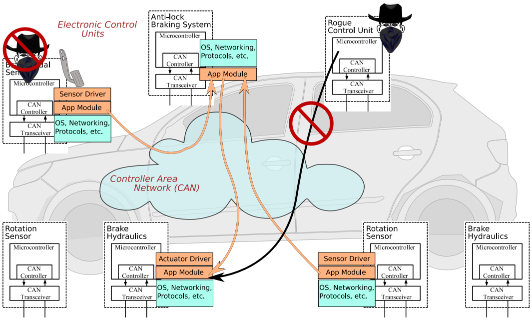
- Arbitrary confidentiality properties; can be fixed at the expense of efficiency
- Availability or real-time properties; work-in-progress, weaker attacker model

Extended Application Scenarios



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

Extended Application Scenarios



“Efficient Component Authentication and Software Isolation for Automotive Control Networks”, Van Bulck et al., ACSAC 2017, in press.

Summary & Conclusions

Authentic Execution of Distributed Event-Driven Applications

- Secure and open software application platforms for distributed IT, in the presence of network / software attackers

Summary & Conclusions

Authentic Execution of Distributed Event-Driven Applications

- Secure and open software application platforms for distributed IT, in the presence of network / software attackers

- Critical software components are resilient against attacks
- Security of each module independent from other software
- Authenticated communication and remote attestation links components of application
- 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

Authentic Execution of Distributed Event-Driven Applications

- Secure and open software application platforms for distributed IT, in the presence of network / software attackers

- Critical software components are resilient against attacks
- Security of each module independent from other software
- Authenticated communication and remote attestation links components of application
- 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!
- Applicable in many domains: automotive, medical, . . .

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; automating secure compilation to PMAs [BNMP15, PAS$^+$15, vGSMP16]

Secure I/O: Trusted Paths between microcontrollers attached to sensors and actuators [NVBM$^+$17, MCM$^+$16]

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

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

Thank you!

Thank you! Questions?

https://people.cs.kuleuven.be/ jantobias.muehlberg/stm17/
https://distrinet.cs.kuleuven.be/software/sancus/

References I

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.

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.

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.

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.

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.

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.

References II

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.