KU LEUVEN

Introduction to Trusted Computing

Pieter Maene, Johannes Götzfried, Ruan de Clercq,
Tilo Müller, Felix Freiling, and Ingrid Verbauwhede

[1]KU Leuven/COSIC, Belgium
[2]FAU Erlangen-Nürnberg, Germany

January 31, 2017

Trusted Computing

Trusted Computing

"An entity can be trusted if it always behaves in the expected manner for the intended purpose."—Trusted Computing Group 2004

Hardware-Based Architectures

- Limitations of software-based solutions
- Protect against system-level attacker
- Hardware considered immutable

Architecture	Security Properties							Architectural Features							Other		
	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance[1]	Memory Protection[2]	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	●	●	○	●	○	●	○	●	○	●	○	○	○	●	●	●	MSP430
Soteria	●	●	○	●	●	●	○	●	○	●	○	○	○	●	●	●	MSP430
SecureBlue++	●	○	●	●	●	○	●	○	○	●	●	●	○	●	○	○	POWER
SGX	●	●	●	●	●	○	●	○	○	○	●	●	●	●	○	○	x86_64
Iso-X	●	●	○	●	○	○	●	○	○	○	●	●	●	●	○	●	OpenRISC
TrustLite	●	●	○	○	○	●	○	●	○	○	●	●	●	●	○	●	Siskiyou Peak
TyTAN	●	●	●	●	○	●	○	●	○	○	●	●	●	●	○	●	Siskiyou Peak
Sanctum	●	●	●	●	●	●	○	○	○	○	●	●	●	●	●	●	RISC-V

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

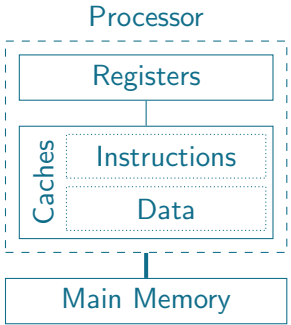
[1]Resistance against software side-channel attacks targeting memory access patterns only.

[2]Protection from physical attacks, both passive (e.g., probing) and active (e.g., fault injection).

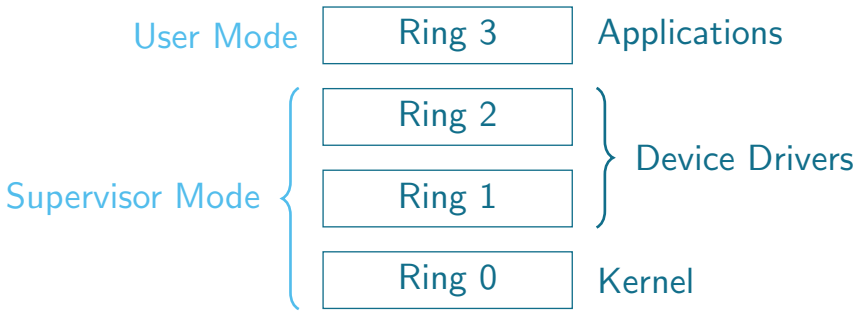
Outline

1. Introduction
2. Background
3. Attacker Model
4. Properties
5. Architectures
6. Comparison
7. Conclusion

Memory Hierarchy

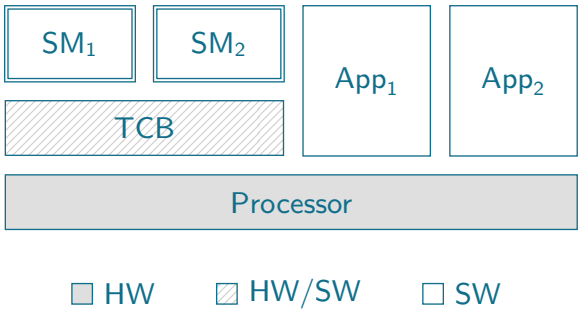


Protection Rings



Protected Module Architectures (PMAs)

- Protect smaller, verifiable code base
- Trusted Computing Base (TCB)

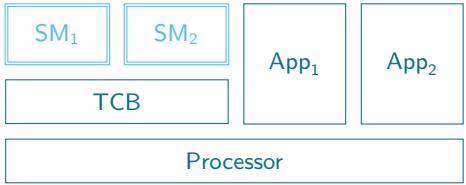


Attacker Model

1. Controls all software outside the TCB
2. Access to communication channel
3. Dolev-Yao
4. No Denial-of-Service protection
5. Physical attacks out of scope
   - Some allow off-chip memory attacks
   - Hardware side-channels not considered
6. Software side-channels generally excluded

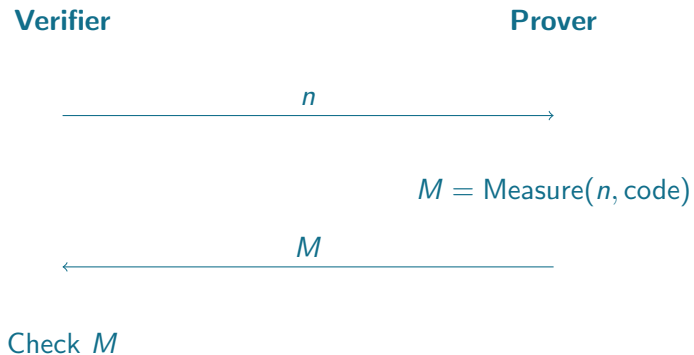
Isolation

- Access control mechanism
- Entry point

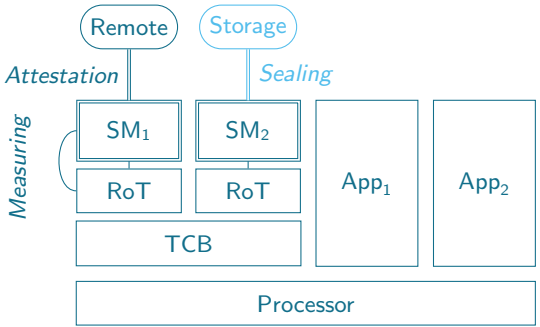


Attestation

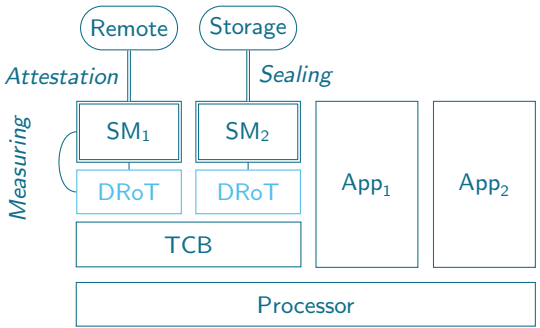
- Measurements anchored in Root of Trust (RoT)



Sealing


Remote
Storage
Attestation
Sealing
Measuring
SM1
SM2
RoT
RoT
App1
App2
TCB
Processor

Dynamic Roots of Trust (DRoTs)



Code Confidentiality

Side-Channel Resistance

- Software side-channels
- Untrusted software only learns I/O behaviour

Memory Protection

- Integrity and authenticity of main memory
- Active and passive attacks

Architectural Features

Lightweight

- Architectures without MMU
- Limited number of applications

Preemption

- Suspension of running tasks at any time
- Mainly impacts context switching

Upgradeable TCB

- Hardware-only TCB is not upgradeable
- Some designs include trusted software
- Design flexibility and later upgrades

Architectures

SMART ([El Defrawy et al., 2012])
Lightweight remote attestation mechanism

Sancus ([Noorman et al., 2013])
Protected module architecture for embedded systems

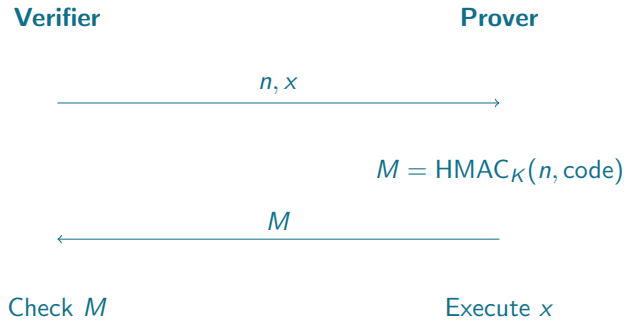
TrustZone (ARM, 2009)
Isolation mechanism in ARM's processors

SMART

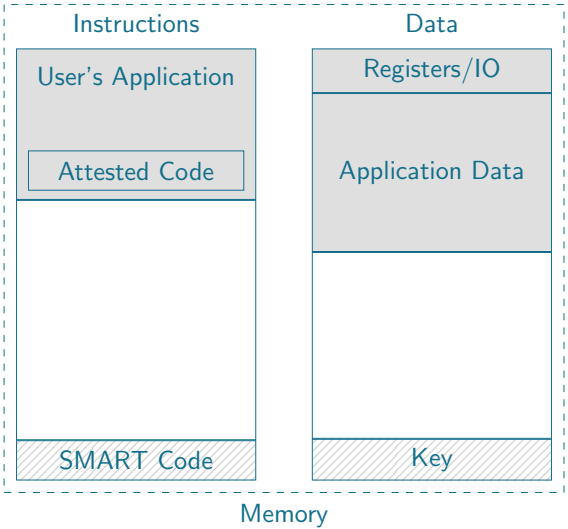
- Lightweight remote attestation mechanism
- Minimal (proven by [Francillon et al., 2014])

Architecture	Security Properties							Architectural Features							Other		
	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance[1]	Memory Protection[2]	Lightweight	Coprocessor	HW-Only TCB	Preemption	Dynamic Layout	Upgradeable TCB	Backwards Compatibility	Open-Source	Academic	Target ISA
SMART	○	●	○	●	○	–	○	●	○	○	–	–	○	●	○	●	AVR/MSP430

SMART


Verifier
Prover
n, x
$M = \mathrm{HMAC}_K(n, \mathrm{code})$
M
Check M
Execute x

SMART



HMAC

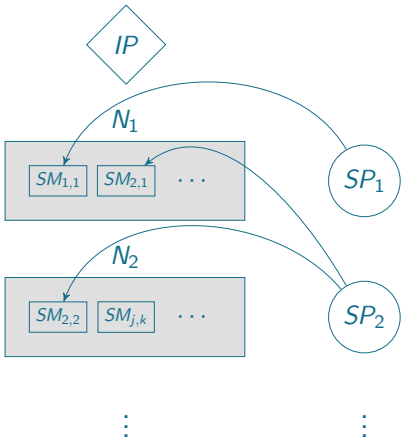
Reset
Memory Erasure

Sancus

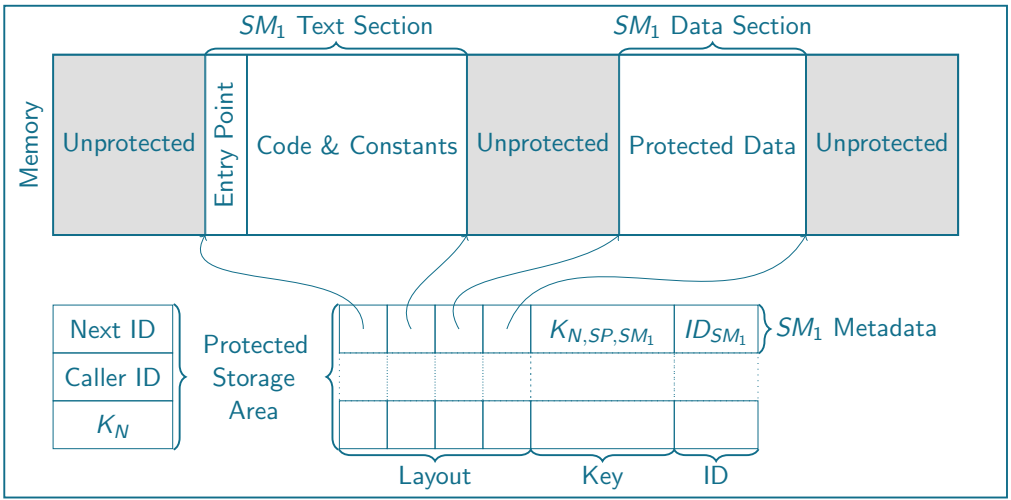
- Hardware-only protected module architecture for embedded devices
- Program counter-based access control
- Extended with code confidentiality ([Götzfried et al., 2015])

Architecture	Security Properties							Architectural Features							Other		
	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance[1]	Memory Protection[2]	Lightweight	Coprocessor	HW-Only TCB	Preemption	Dynamic Layout	Upgradeable TCB	Backwards Compatibility	Open-Source	Academic	Target ISA
Sancus	●	●	○	●	○	●	○	●	○	●	○	○	○	●	●	●	MSP430
Soteria	●	●	○	●	●	●	○	●	○	●	○	○	○	●	●	●	MSP430

Sancus



Sancus

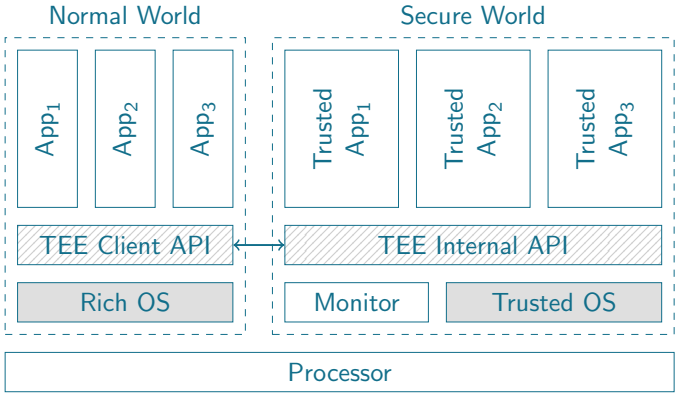

SM_1 Text Section
SM_1 Data Section
Memory
Unprotected
Entry Point
Code & Constants
Unprotected
Protected Data
Unprotected
Next ID
Caller ID
K_N
Protected Storage Area
K_{N,SP,SM_1}
ID_{SM_1}
SM_1 Metadata
Layout
Key
ID

TrustZone

- Global Platform's *Trusted Execution Environment (TEE)*
- Normal World (REE) and Secure World (TEE)

Architecture	Security Properties							Architectural Features							Other		
	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance[1]	Memory Protection[2]	Lightweight	Coprocessor	HW-Only TCB	Preemption	Dynamic Layout	Upgradeable TCB	Backwards Compatibility	Open-Source	Academic	Target ISA
TrustZone	●	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	ARM

TrustZone


Normal World
Secure World
App1
App2
App3
Trusted App1
Trusted App2
Trusted App3
TEE Client API
TEE Internal API
Rich OS
Monitor
Trusted OS
Processor

Comparison

Isolation

- Provided by all except TPM and SMART
- Lightweight: program counter-based memory access control
- Complex architectures extend MMU, coarser granularity

Attestation

- Wide variety of approaches
- Simple symmetric protocols in hardware
- Trusted software for advanced algorithms

Comparison

TCBs

- Hardware-only TCB cannot be upgradeable
- Stronger guarantees, as no part is vulnerable to software attackers
- Carefully designed software components increase flexibility

Trust Boundaries

- Typically extend to the CPU package
- Protection against physical bus and memory attacks

Attacker Model

- Very similar for all isolation architectures
- Internal vulnerabilities remain exploitable

Comparison

Code Injection Attacks

- Protected against by isolation mechanism
- Attestation enables detection of changes

Code Reuse Attacks

- Prevented by enforcing the entry point

Software Side-Channel Attacks

- No general protection mechanism
- Sanctum addresses cache timing attacks

Comparison

Backwards Compatibility

- Mechanisms integrated by programmers or enabled transparently
- Legacy applications typically remain vulnerable

Inter-Process Communication

- Register-based for smaller messages
- Larger messages sent through shared memory

Architecture	Security Properties							Architectural Features						
	Isolation	Attestation	Sealing	Dynamic RoT	Code Confidentiality	Side-Channel Resistance	Memory Protection	Lightweight	Coprocessor	HW-Only TCB	Preemption	Dynamic Layout	Upgradeable TCB	Backwards Compatibility
AEGIS	●	●	●	●	●	○	●	○	○	●	●	●	○	●
TPM	○	●	●	○	●	–	◐	○	●	●	–	–	○	●
TXT	●	●	●	●	●	●	◐	○	●	●	○	●	○	●
TrustZone	●	○	○	●	○	○	○	○	○	○	●	●	○	●
Bastion	●	○	●	●	●	○	●	○	○	○	●	●	●	●
SMART	○	●	○	●	○	–	○	●	○	○	–	–	○	●
Sancus	●	●	○	●	○	●	○	●	○	●	○	○	○	●
Soteria	●	●	○	●	●	●	○	●	○	●	○	○	○	●
SecureBlue++	●	○	●	●	●	○	●	○	○	●	●	●	○	●
SGX	●	●	●	●	●	○	●	○	○	○	●	●	●	●
Iso-X	●	●	○	●	○	○	●	○	○	○	●	●	●	●
TrustLite	●	●	○	○	○	●	○	●	○	○	●	●	●	●
TyTAN	●	●	●	●	○	●	○	●	○	○	●	●	●	●
Sanctum	●	●	●	●	●	●	○	○	○	○	●	●	●	●

Architecture	Other		
	Open-Source	Academic	Target ISA
AEGIS	○	●	–
TPM	○	○	–
TXT	○	○	x86_64
TrustZone	○	○	ARM
Bastion	○	●	UltraSPARC
SMART	○	●	AVR/MSP430
Sancus	●	●	MSP430
Soteria	●	●	MSP430
SecureBlue++	○	○	POWER
SGX	○	○	x86_64
Iso-X	○	●	OpenRISC
TrustLite	○	●	Siskiyou Peak
TyTAN	○	●	Siskiyou Peak
Sanctum	●	●	RISC-V

Conclusion

- Protect applications and users from malicious software
- All architectures offer strong guarantees
- Very few support all possible trusted computing mechanisms
- Many researchers do not open-source their designs

Conclusion

- Protect applications and users from malicious software
- All architectures offer strong guarantees
- Very few support all possible trusted computing mechanisms
- Many researchers do not open-source their designs

Questions?

pieter.maene@esat.kuleuven.be