Adaptive Isolation for Security

Patrick Schaumont
Virginia Tech

Dagstuhl Seminar 16441
1 November 2016

Objective

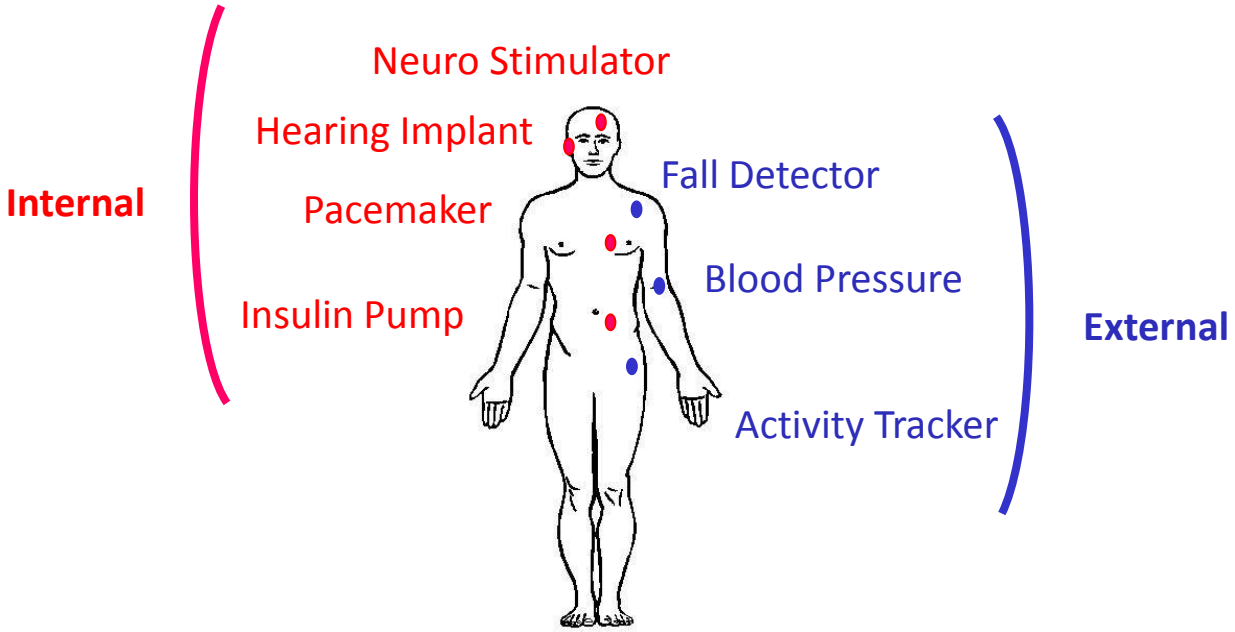
1. **Contemporary Secure Computing**
   An Example: Trusted Medical Applications

2. **Building Blocks of Secure Computing**
   - **Attacker Models**
   - **Trust**

3. **Isolation for Security in Practice**
   - **Lightweight Isolation using SANCUS**
   - **Server-class Isolation using SGX**

4. **Open Issues**

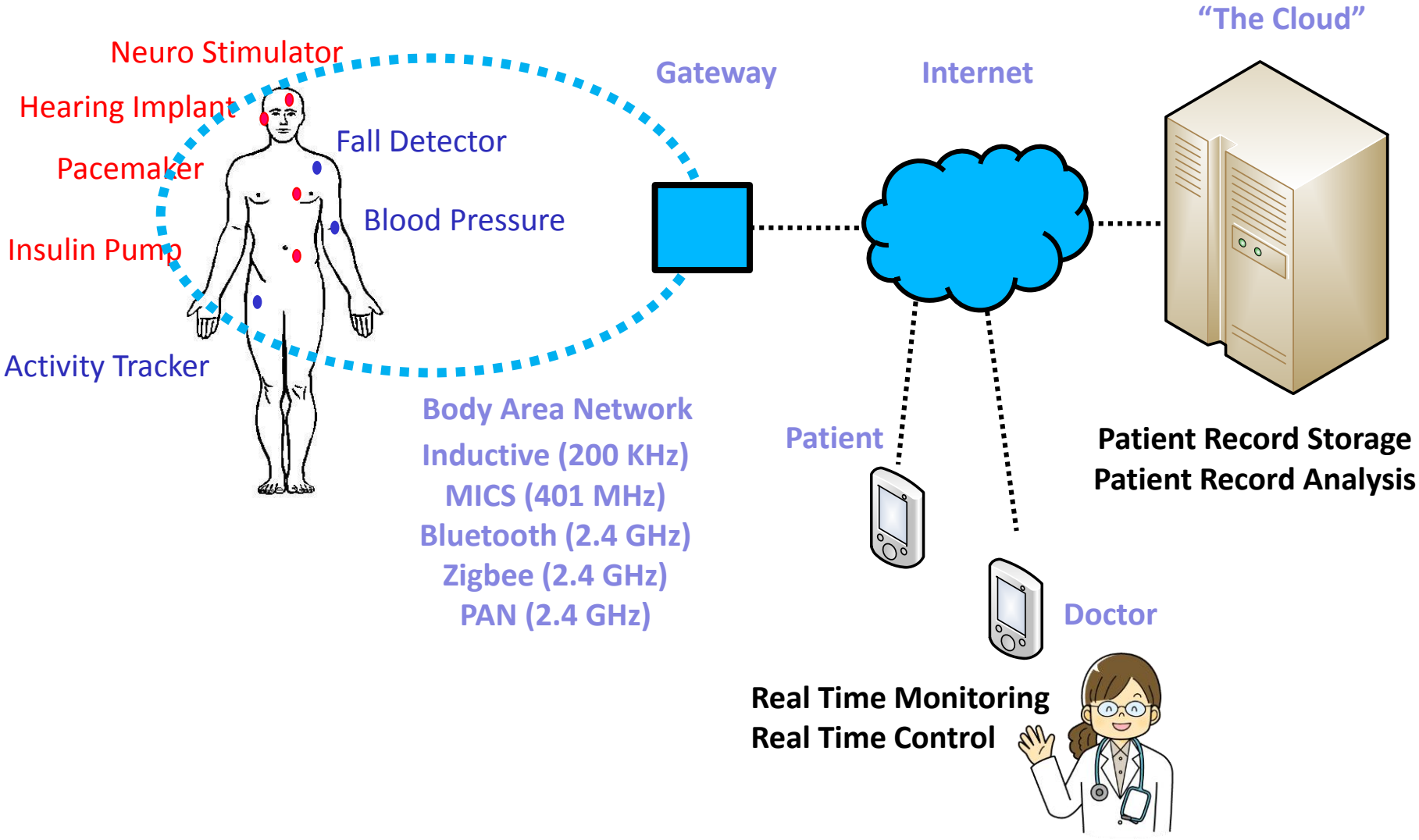
Implantable/Wearable Medical Devices



	Sensing	Actuation	Control
Insulin Pump	Glucose Level	Insulin	Open-loop (programmer)
Defibrillator	Heart Rate	Shock	Closed-loop

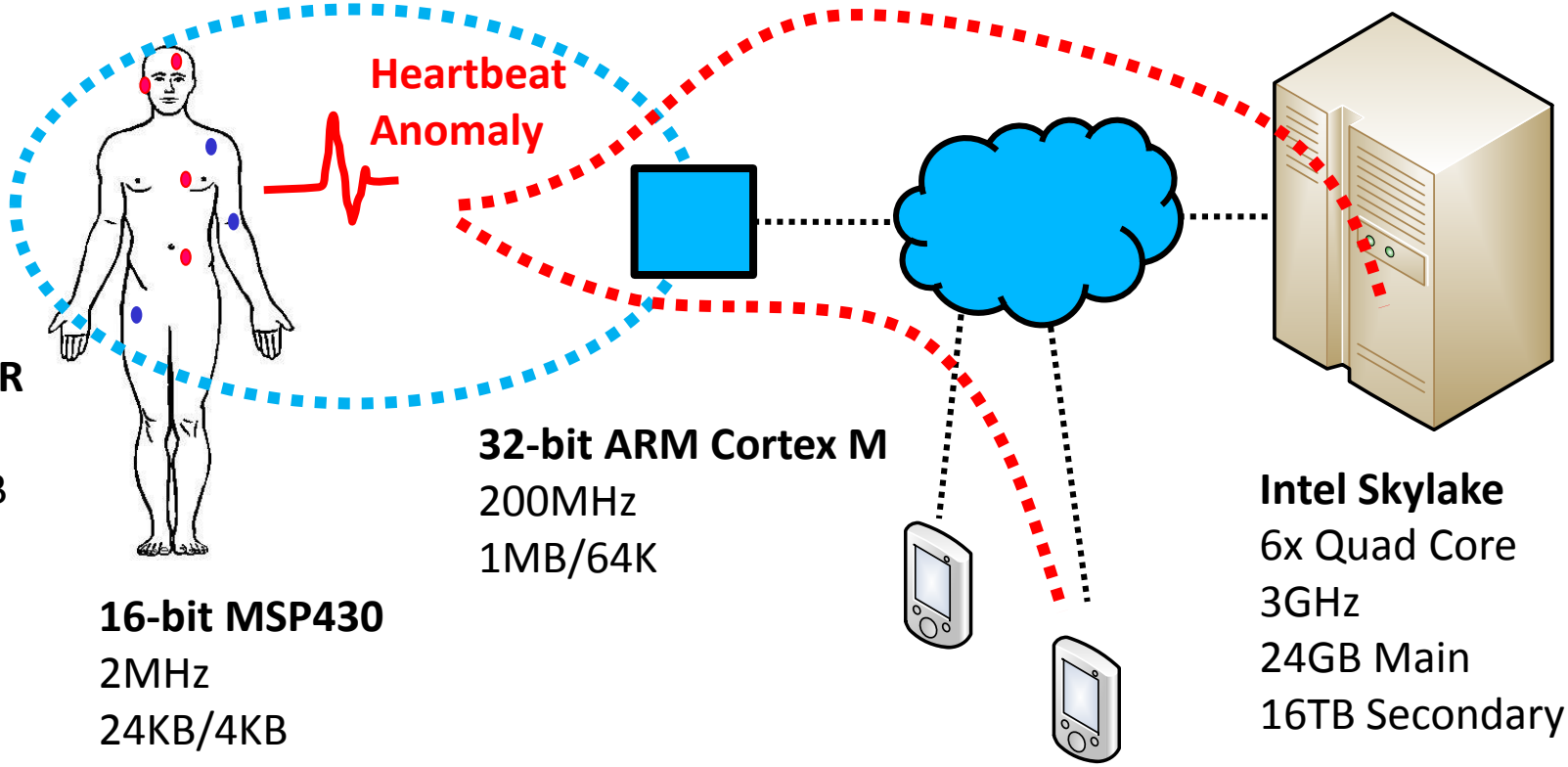
Ref. [2]

Implantable/Wearable Medical Devices



Computers Everywhere!

Data bits have a uniform privacy/security concern



8-bit AVR
100KHz
8KB/2KB

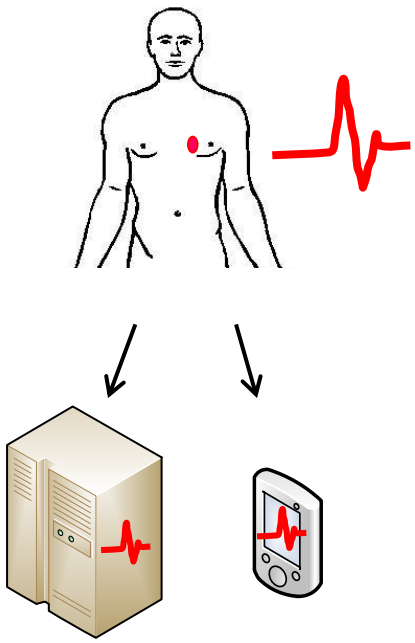
16-bit MSP430
2MHz
24KB/4KB

32-bit ARM Cortex M
200MHz
1MB/64K

64-bit ARM Cortex A53
Quad Core
800MHz
2GB

Intel Skylake
6x Quad Core
3GHz
24GB Main
16TB Secondary

Medical Data and IMD Concerns



Security

- Data confidentiality storage + transmission
- Data access authorization
- Data origin authentication
- Data integrity
- Data & device availability

Safety

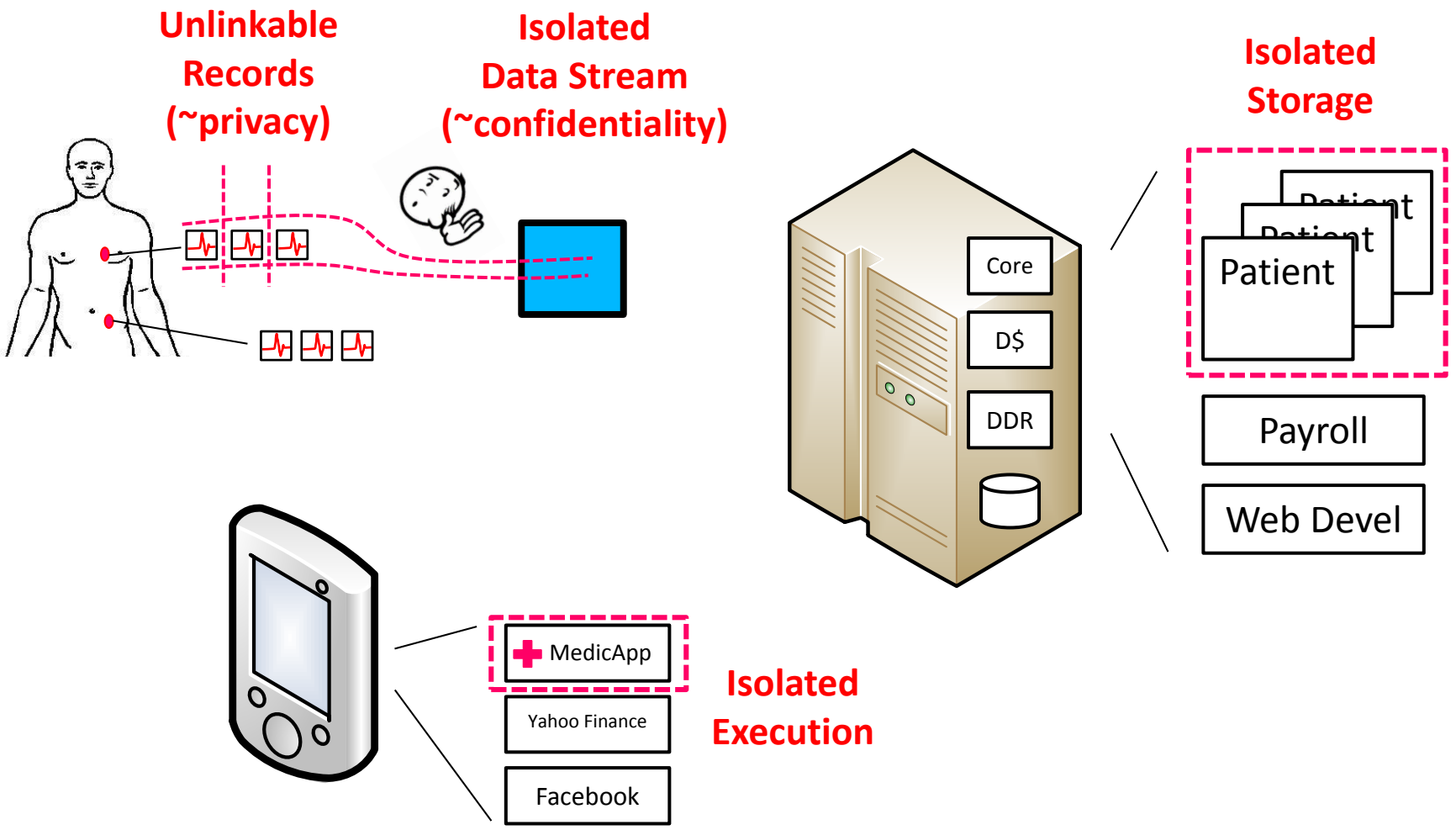
- Device access
- Device update

Privacy

- Device existence, type, ID
- Link patient identity, device data
- Device tracking, fingerprinting

Ref. [1]

Isolation


Unlinkable Records (~privacy)
Isolated Data Stream (~confidentiality)
Isolated Storage
Core
D$
DDR
Patient
Payroll
Web Devel
MedicApp
Yahoo Finance
Facebook
Isolated Execution

Two (or more...) worlds of secure computing

Implanted	Wearable	Gateway	Mobile	Cloud
8-bit AVR 100KHz 8KB/2KB	**16-bit MSP430** 2MHz 24KB/4KB	**32-bit ARM Cortex M** 200MHz 1MB/64K	**64-bit ARM Cortex A53** Quad Core 800MHz 2GB	**Intel Skylake** 6x Quad Core 3GHz 24GB Main 16TB Secondary

Microcontrollers

Simple Architecture

Computation (Crypto) is slow

Statically-stored Secrets

Architecture Isolation is add-on

Servers

Extremely Complex

Computation (Crypto) is fast

Ephemeral Secrets

Architecture Isolation is built-in

Objective

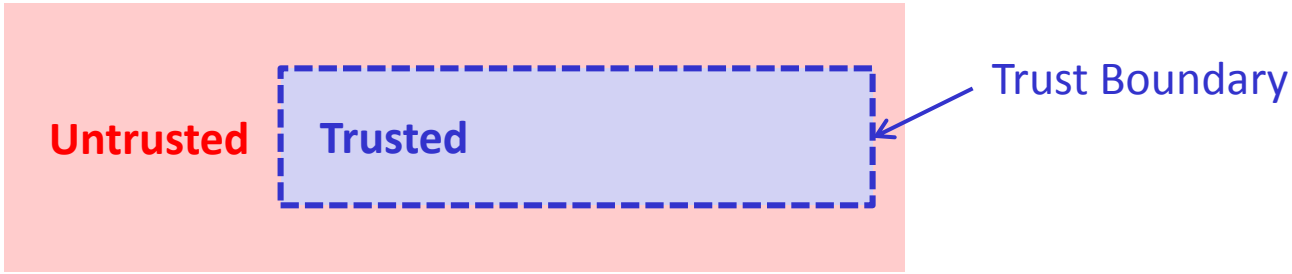
1. **Contemporary Secure Computing**
   An Example: Trusted Medical Applications

2. **Building Blocks of Secure Computing**
   - **Attacker Models**
   - **Trust**

3. **Isolation for Security in Practice**
   - **Lightweight Isolation using SANCUS**
   - **Server-class Isolation using SGX**

4. **Open Issues**

Trust

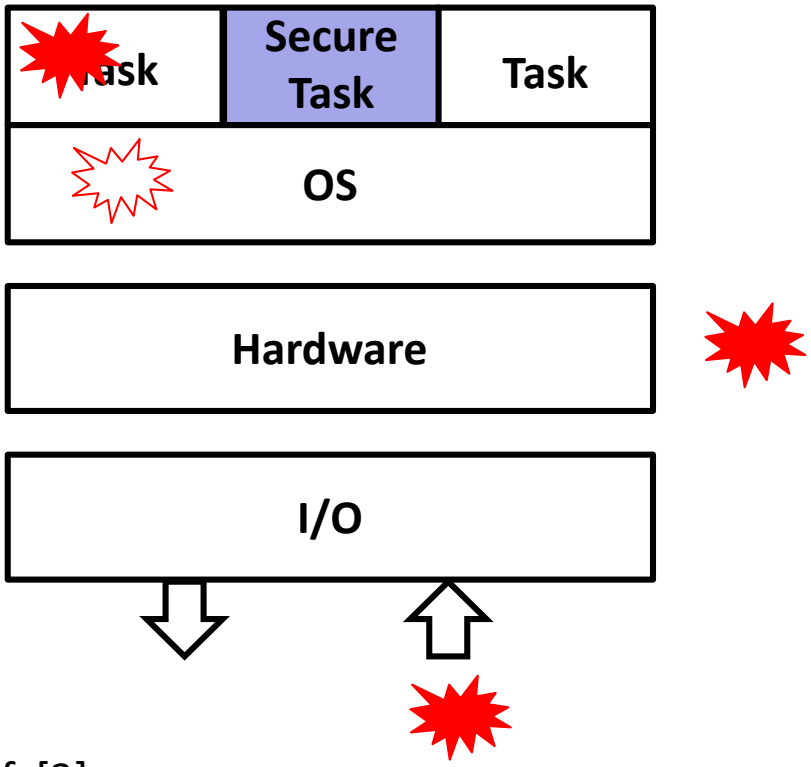


Trusted = to behave as expected

Untrusted = we don’t know what will happen

Attacker Models

An Attacker Model describes how the Adversary may breach trust boundary



Machine Code Attacker Model

- Interact, directly or indirectly, with memory image of secure task

Hardware Attacker Model

- Observe or influence task implementation effects

I/O Attacker Model

- Manipulate or Control all I/O to secure task

Ref. [3]

Countermeasures anticipate Attack Models

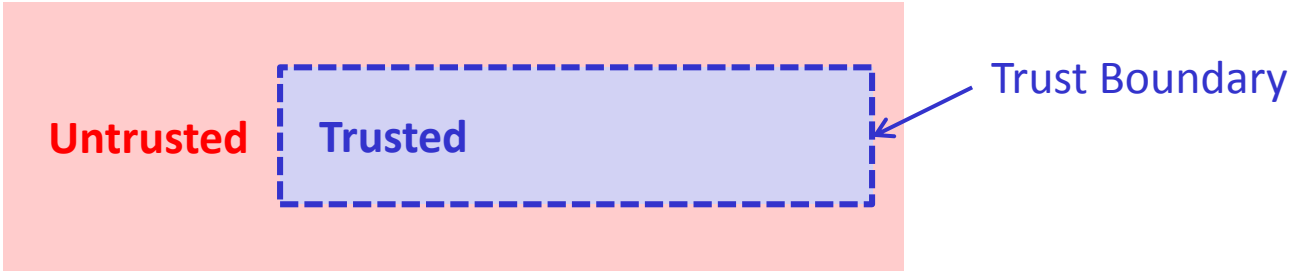
Attack Model	Countermeasure
Machine Code	**Task Isolation** • Virtual Machines • Sandboxing • Protected Module Architectures
Hardware	• Masking/TI • Fault Tolerance • Secure Scan/Debug
I/O	**Memory Safety** • Stack Canaries • Data Execution Prevention • Address Space Layout Randomization

Countermeasures always come with overhead on performance and/or implementation cost.

Security is *never* free.

Trust and Isolation

Assuming an Attacker Model implies choosing what you trust and what you do not trust



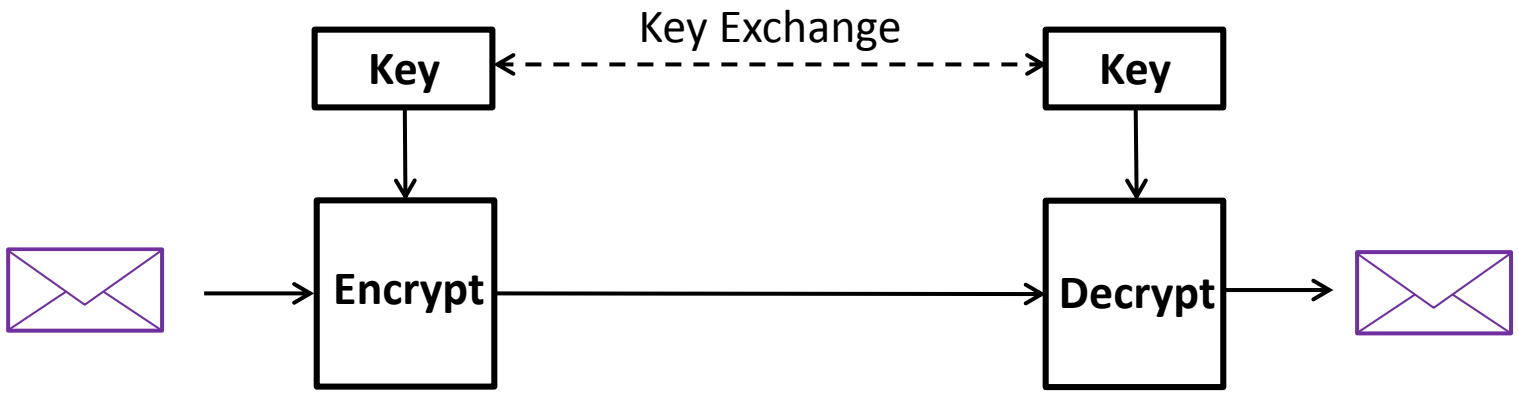
Isolation is one (but not the only) way to achieve trust

Abstraction	Achieving Trust	Example
Information and Data	Information Security	Encryption & Decryption Signing & Verification
Programs	Trusted Computing Base	Isolated Execution
Physical Implementation	Physical Security	Side-channel Countermeas.

InfoSec = Isolation + Interaction

How does isolation help in achieving security?

- **Isolation is a central concept to achieve confidentiality guarantees in a secure implementation**
- **But *completely* isolated architectures have no useful security policy (Alice is lonely without Bob)**



Isolation for Encryption
Isolation for Key Storage
***Communication* for Key Exchange**

Protocol Level (multi-architecture)

Objective

1. **Contemporary Secure Computing**
   An Example: Trusted Medical Applications

2. **Building Blocks of Secure Computing**
   - **Attacker Models**
   - **Trust**

3. **Isolation for Security in Practice**
   - **Lightweight Isolation using SANCUS**
   - **Server-class Isolation using SGX**

4. **Open Issues**

Two (or more...) worlds of isolation

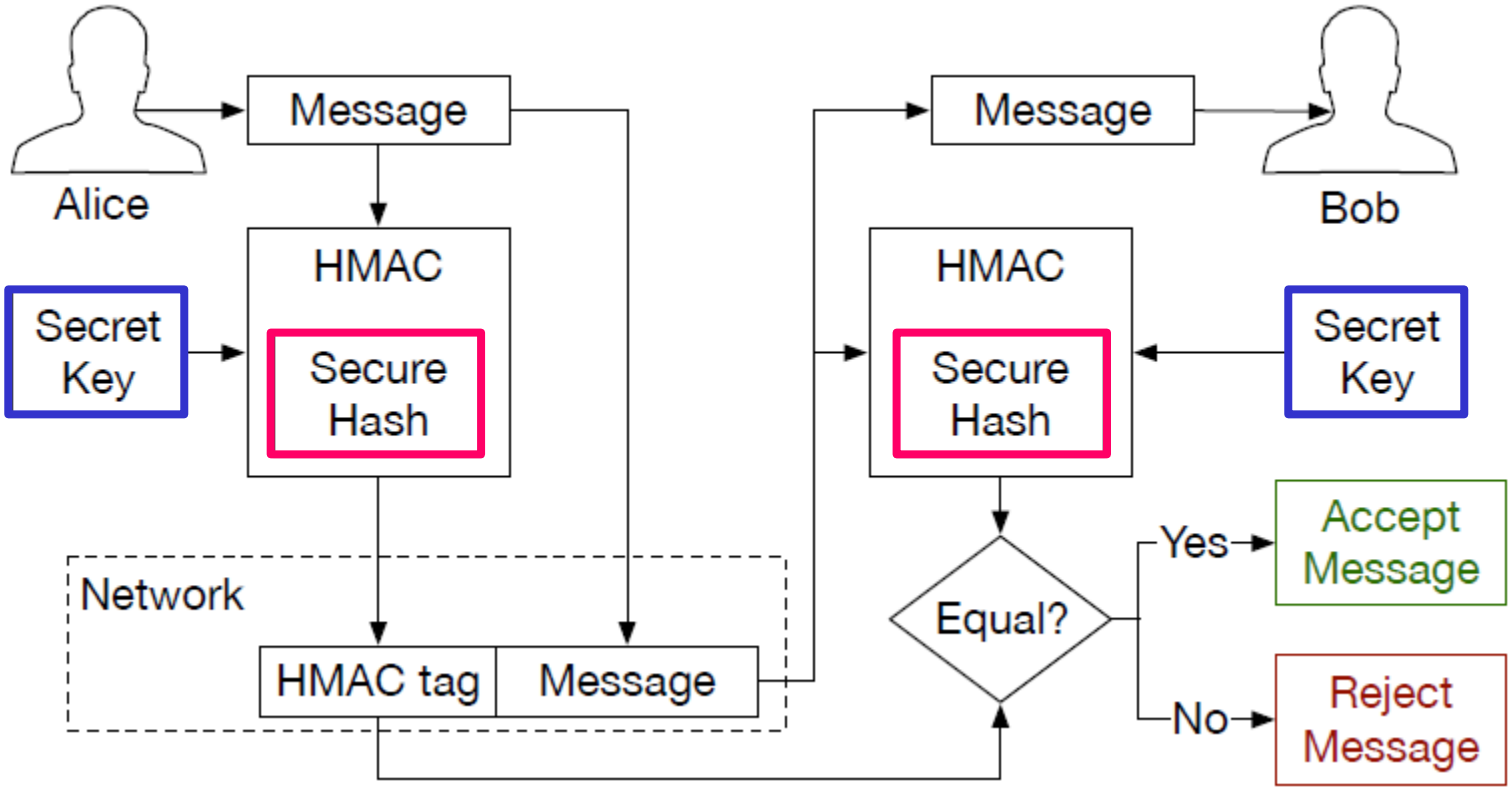
Implanted	Wearable	Gateway	Mobile	Cloud
8-bit AVR 100KHz 8KB/2KB	**16-bit MSP430** 2MHz 24KB/4KB	**32-bit ARM Cortex M** 200MHz 1MB/64K	**64-bit ARM Cortex A53** Quad Core 800MHz 2GB	**Intel Skylake** 6x Quad Core 3GHz 24GB Main 16TB Secondary

Microcontrollers
Driving Example: SANCUS

Servers
Driving Example: SGX

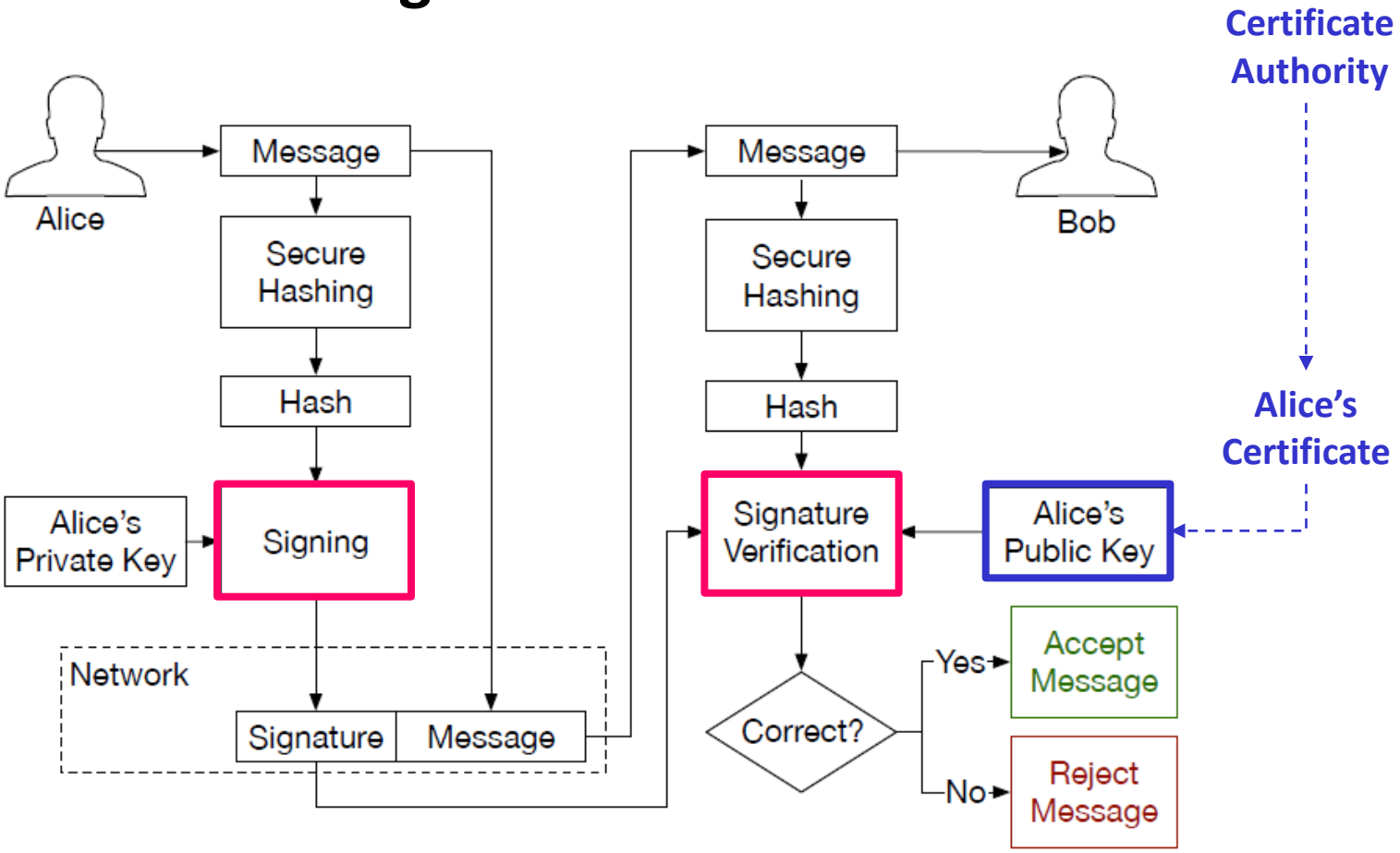
Measuring Integrity -> Remote or Local Attestation

- **Symmetric Setting**



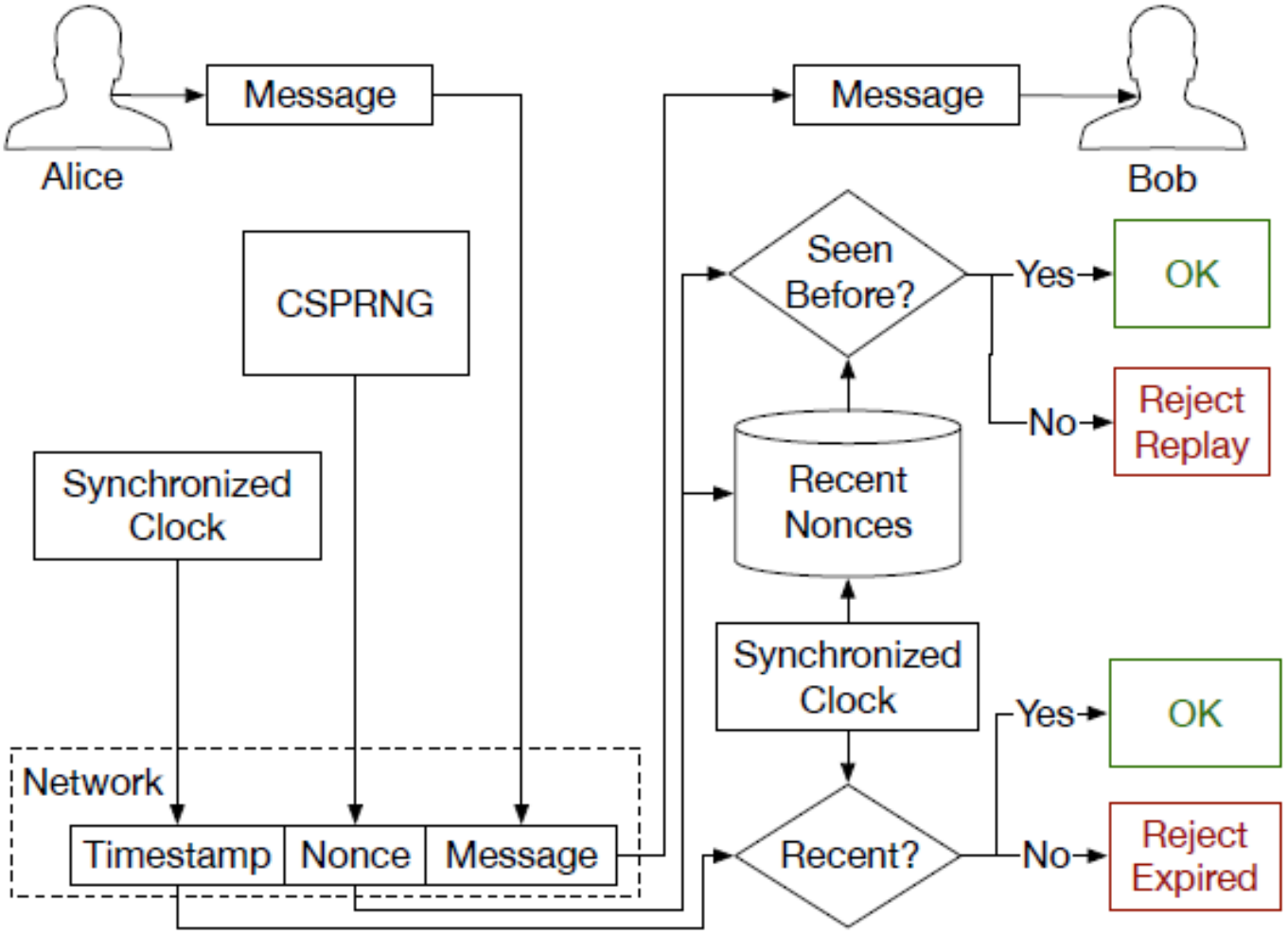
Ref. [5]

- **Asymmetric Setting**



Ref. [5]

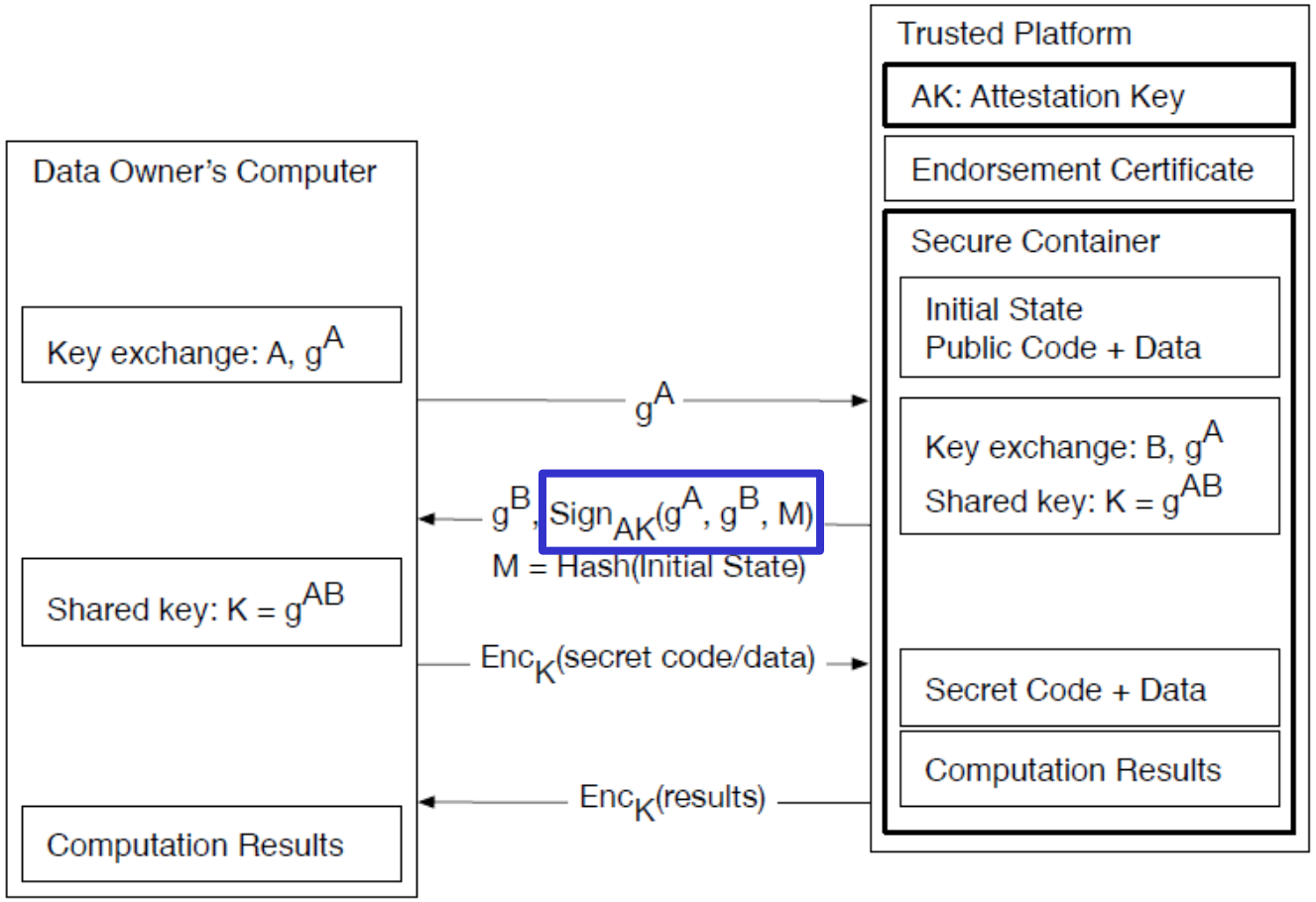
Freshness



Ref. [5]

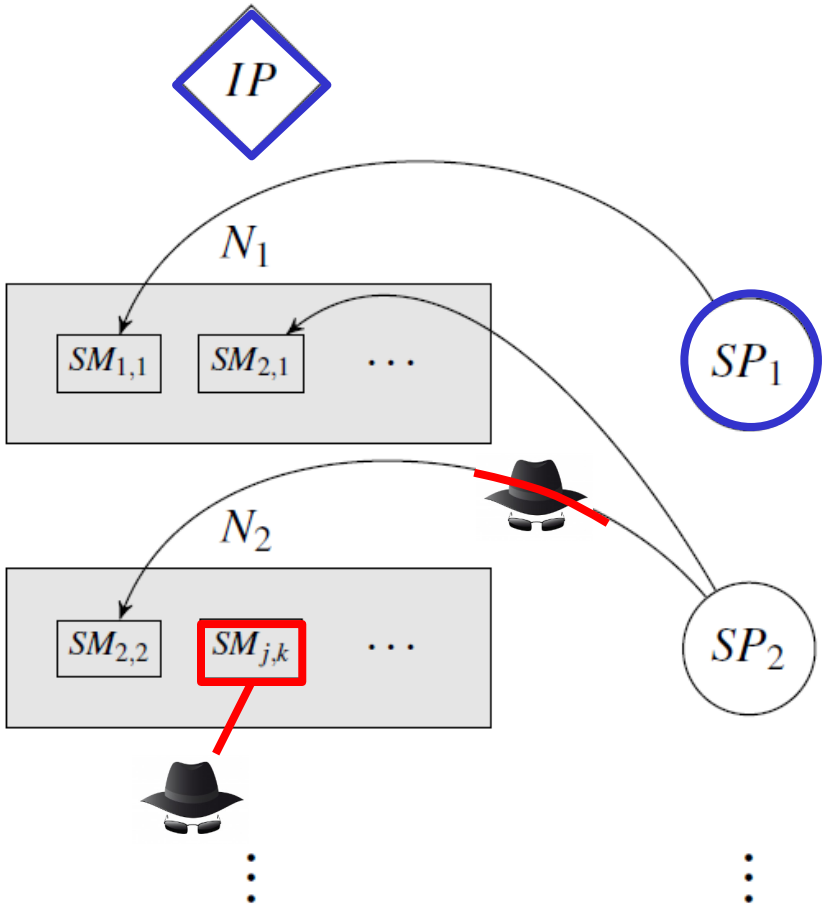
Attestation

Data Owner's Computer gets assurance that it is talking to a Secure Container with specific Code, Data



Ref. [5]

SANCUS: Secure System Model

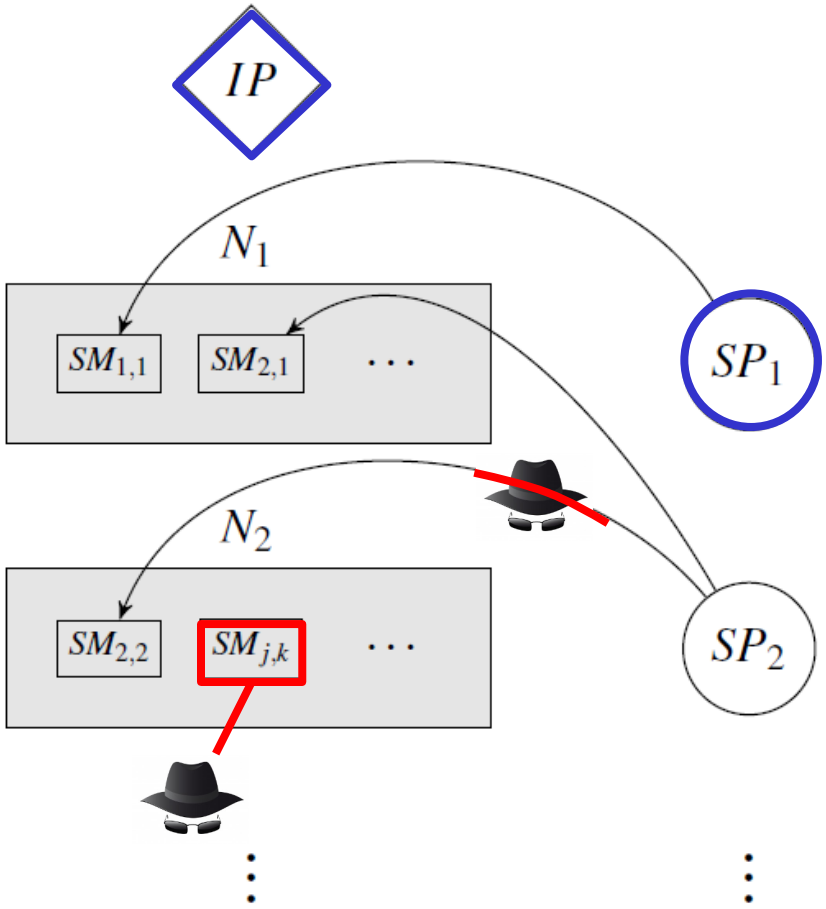


- Infrastructure Provider IP manages Micro-Controller Node N
- Software Provider SP deploy Software SM

- Adversary can control all software
- Adversary can control all communications
- *Hardware is Trusted*

Ref. [4]

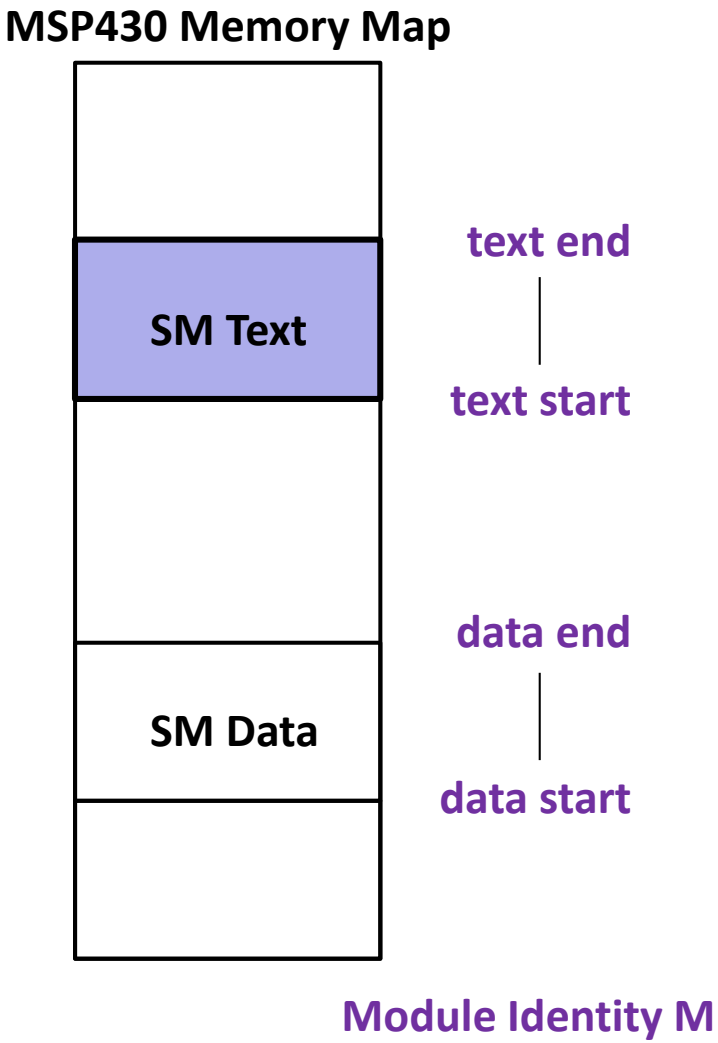
SANCUS: Security Properties



1. (HW Enforced) Isolation of SM + designated entry points
2. Remote Attestation for SM to SP
3. Secure Communication Auth, Integrity, Freshness between SM to SP
4. SM on same node can securely communicate

Ref. [4]

SANCUS: Isolation

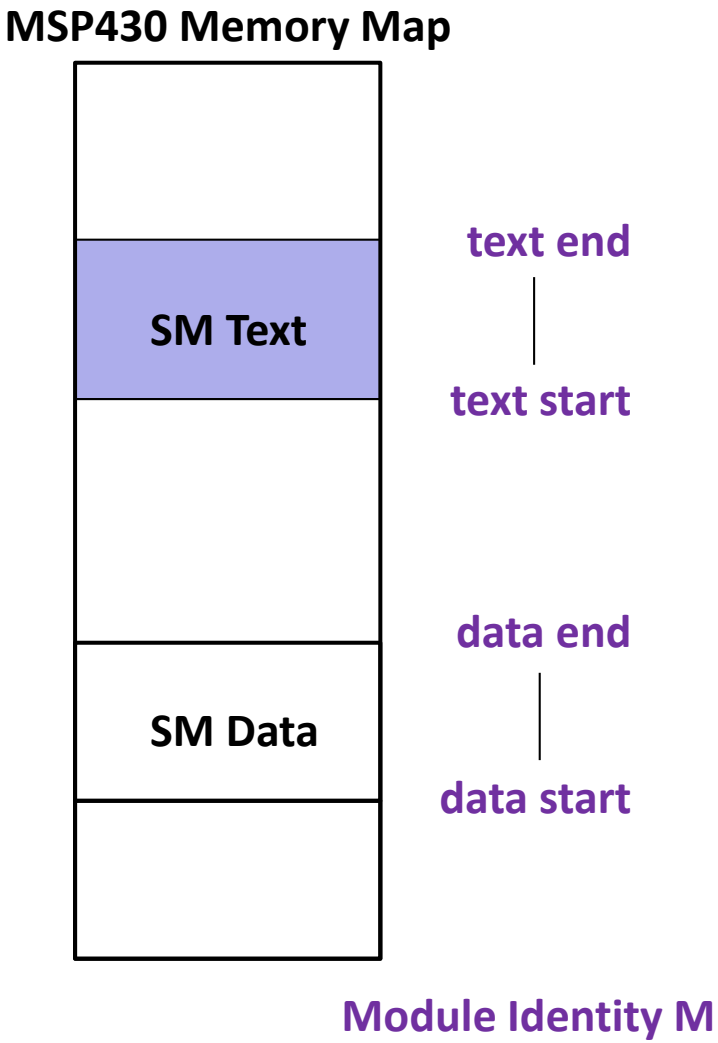


- **A secure module SM**
  - **code section with entry points**
  - **data section**
- **Hardware-enforced memory access control**
  1. **protected code access has protected data**
  2. **protected code has controlled entry point**
- **Dedicated Instructions**

```
protect SP, layout

unprotect
```

SANCUS: Privileged Communications



Node Key K_N

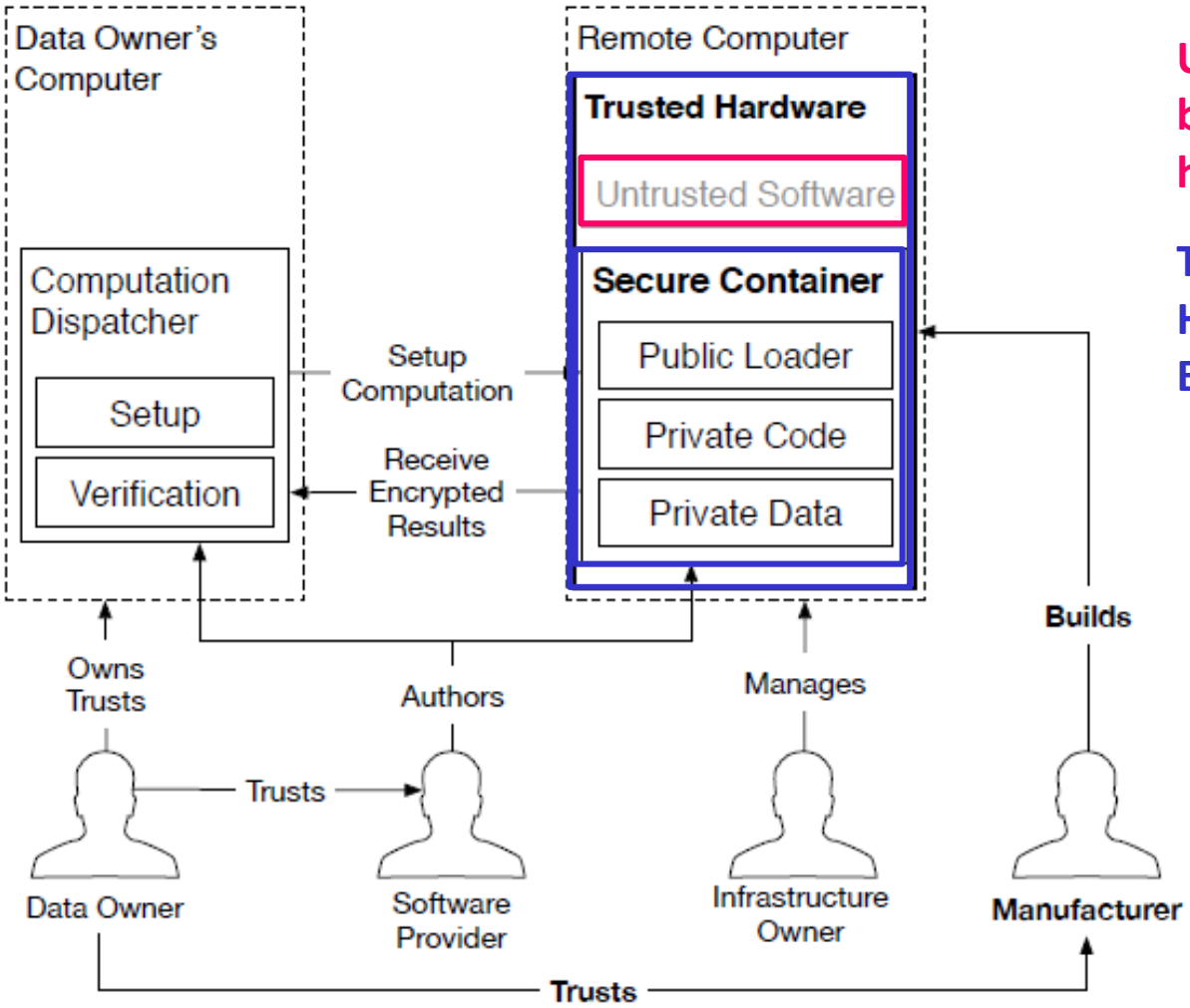
Hardware Root of Trust

Provider Key $K_{N,SP} = kdf(K_N, SP)$

Module Key $K_{N,SP,M} = kdf(K_N, SP, M)$

- **Dedicated Instruction**
  `MAC-seal start, length, result`
- **Remote Attestation**
  - SP sends nonce
  - SM replies MAC using $K_{N,SP,M}$
- **Integrity**
  - SM self-MAC using $K_{N,SP,M}$
  - SM MAC over result using $K_{N,SP,M}$

SGX: Secure System Model

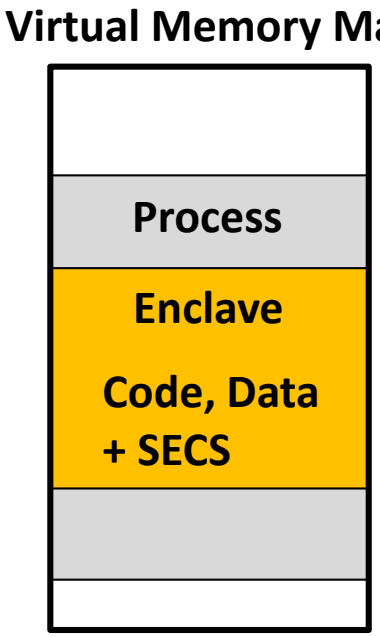


Untrusted:
bios, drivers, kernel, hypervisor

Trusted:
Hardware (Intel CPU)
Enclave (SW App)

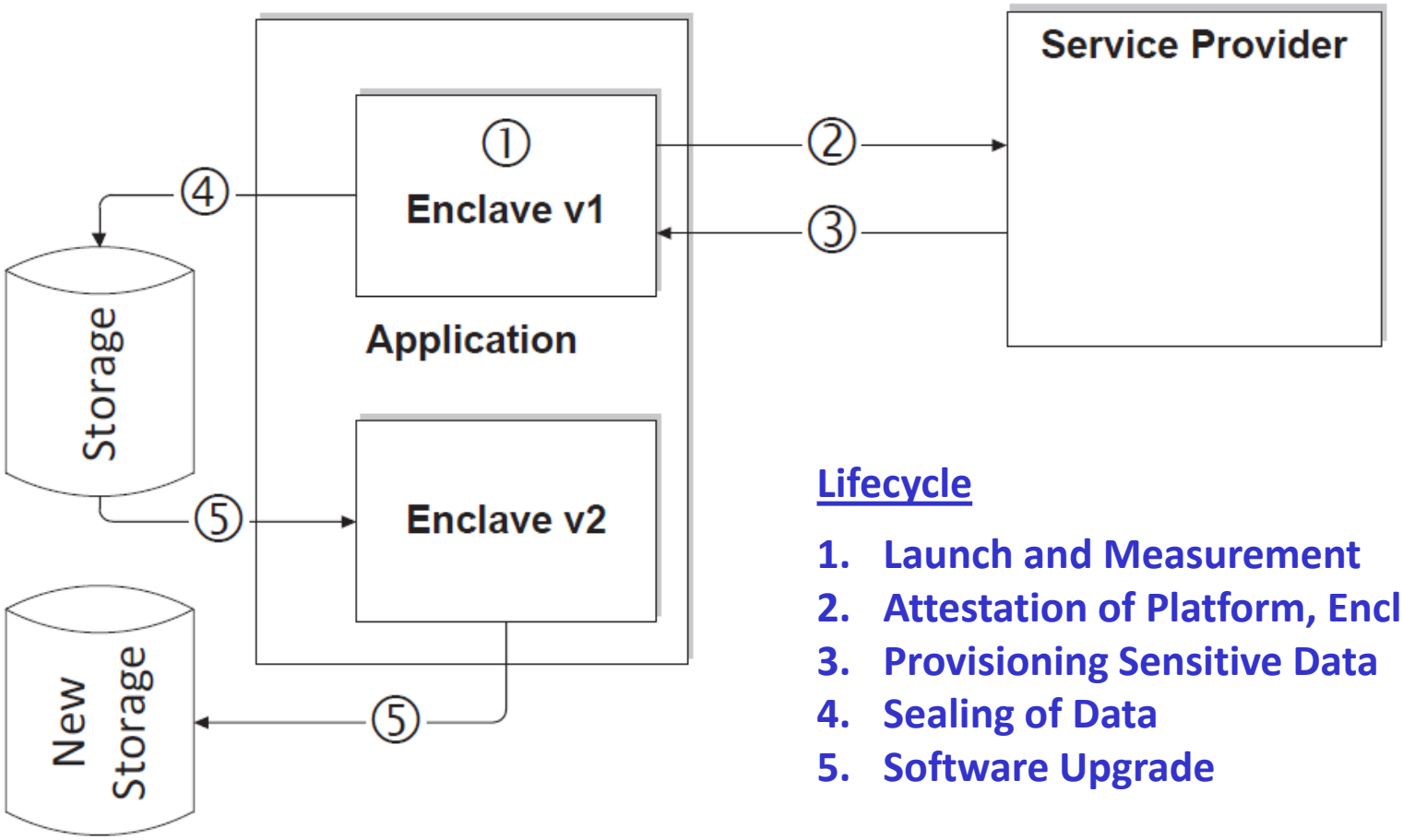
Ref. [5]

SGX Enclave



- Integrity, Confidentiality on Code and Data
- Controlled Entry Points
- Handling of Faults, Interrupts, Syscalls
- Support Multiple Processors, threads
- Access control on Physical Memory pages allocated to Enclaves
- Encryption of Swapped Pages

SGX Enclave Application Scenario



Lifecycle

1. **Launch and Measurement**
2. **Attestation of Platform, Enclave**
3. **Provisioning Sensitive Data**
4. **Sealing of Data**
5. **Software Upgrade**

Ref. [6]

Objective

1. **Contemporary Secure Computing**
   An Example: Trusted Medical Applications

2. **Building Blocks of Secure Computing**
   - **Attacker Models**
   - **Trust**

3. **Isolation for Security in Practice**
   - **Lightweight Isolation using SANCUS**
   - **Server-class Isolation using SGX**

4. **Open Issues**

Open Challenges

- **While performance can quantified (MB/s, MIPS, ...), security is *hardly* quantified**
  - ***Security Level, FIPS-140 Level***
  - **What is the meaning of *resource overhead* for a secure architecture?**
- **What are good metrics for secure computing?**
  - **Formal proofs and properties?**
  - **Performance of primitive secure operations?**
- **What are the orthogonal properties of secure computing?**
  - **If isolation is property #1, what are the others?**
  - **Can we classify secure computer architectures?**

References

1. Michael Rushanan, Aviel D. Rubin, Denis Foo Kune, Colleen M. Swanson: *SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks*. IEEE Symposium on Security and Privacy 2014: 524-539.
2. Wayne Burleson, Shane S. Clark, Benjamin Ransford, Kevin Fu: *Design challenges for secure implantable medical devices*. DAC 2012: 12-17.
3. Frank Piessens, Ingrid Verbauwhede: *Software security: Vulnerabilities and countermeasures for two attacker models*. DATE 2016: 990-999.
4. Job Noorman, Pieter Agten, Wilfried Daniels, Raoul Strackx, Anthony Van Herrewege, Christophe Huygens, Bart Preneel, Ingrid Verbauwhede, Frank Piessens: Sancus: Low-cost Trustworthy Extensible Networked Devices with a Zero-software Trusted Computing Base. USENIX Security Symposium 2013: 479-494.
5. Victor Costan, Srinivas Devadas: Intel SGX Explained. IACR Cryptology ePrint Archive 2016: 86 (2016).
6. Ittai Anati, Shay Gueron, Simon Johnson, Vincent Scarlata: *Innovative Technology for CPU Based Attestation and Sealing.* Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP 2013.