Aalto University

Internet of Things (IoT) Security: Threats, Challenges & Recommendations

MSS – Lecture 6 (part 1)

Samuel Marchal

What is the Internet of Things ?

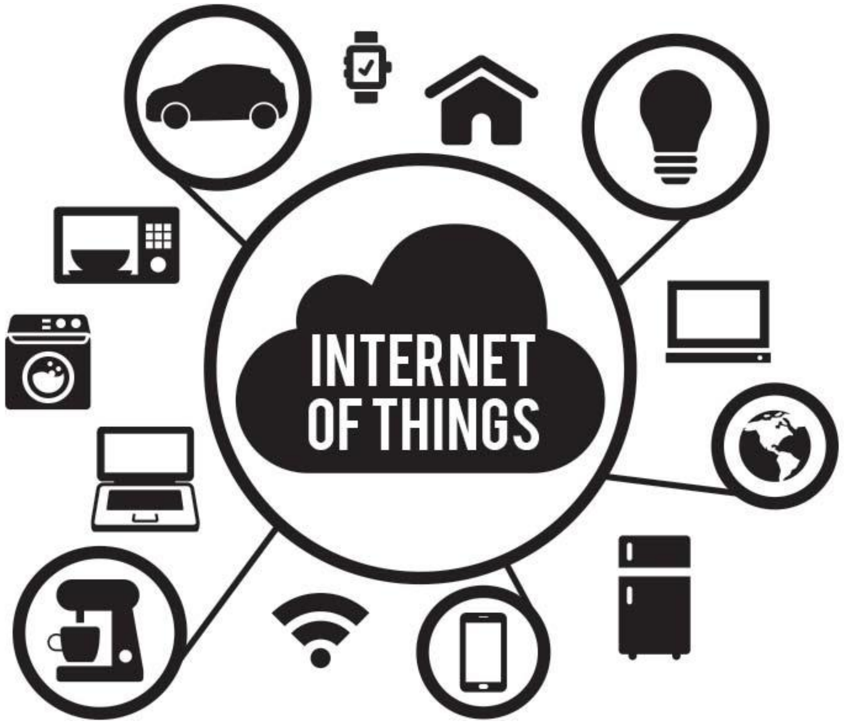
Latest evolution of the Internet:

- machine to machine / device to device communications

Contents (WWW)

Services (WEB 2.0)

People (Social Media)



Definition

Wikipedia:

- “Network of physical objects, devices, vehicles, buildings and other items embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data.”

Gartner:

- “Network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment.”

IoT-based DDoS attack are the most powerful

State-of-the-Art DDoS attack: 20 Gbps

January 2016 (new World Hackers)

- IoT-based DDoS against *bbc.co.uk*
- 600 Gbps (2x previous record)

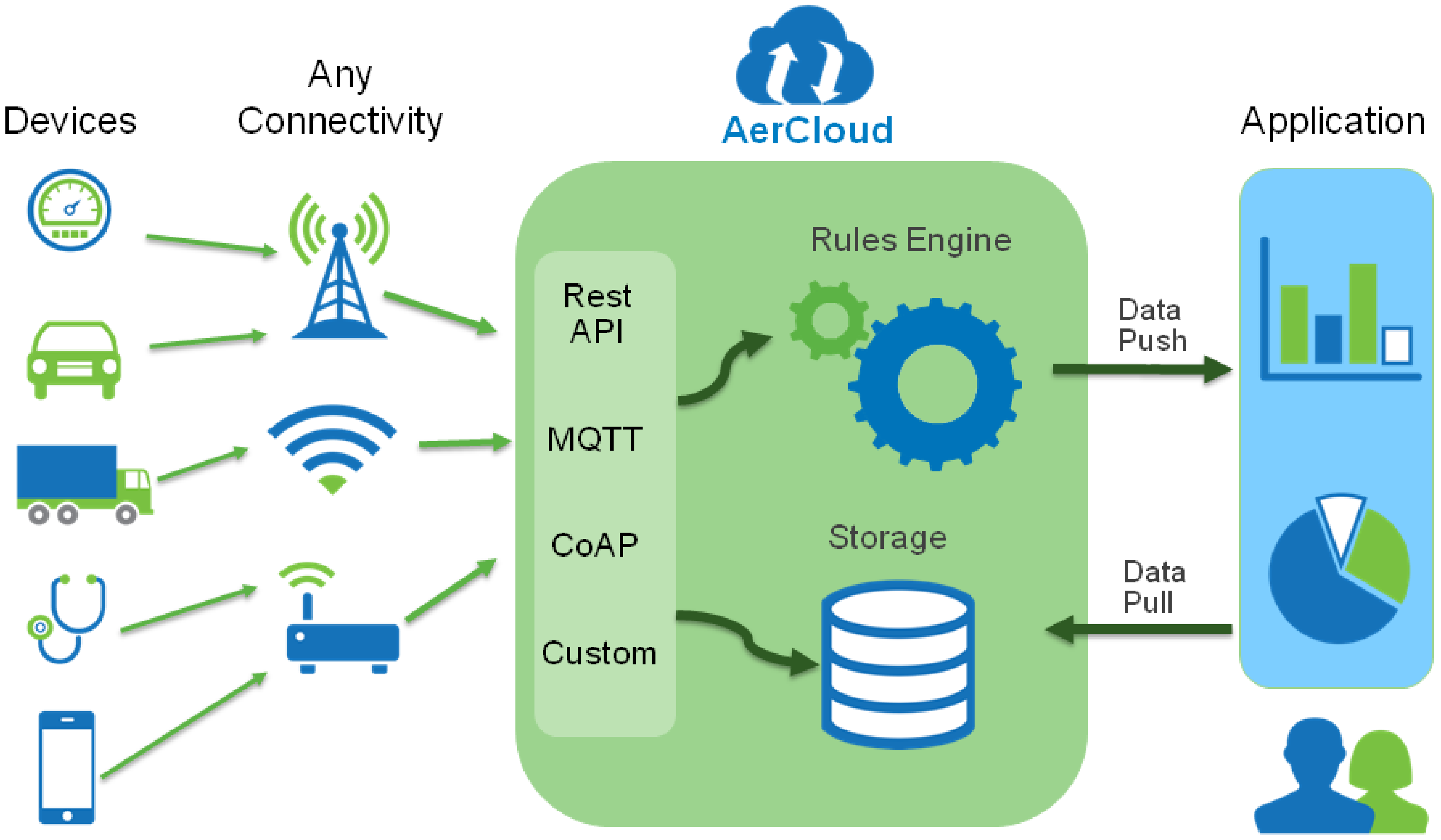
September 2016 (Mirai botnet)

- DDoS against OVH web host
- 1.5 Tbps
- 150,000 IoT devices (IP cameras / DVRs) used

October 2016 (Mirai botnet)

- DDoS against Dyn: dynamic DNS service
- 10s of millions of IoT devices (IP address)
- Disrupts service of Twitter, Spotify, Reddit, SoundCloud, PayPal, etc.

IoT ecosystem



Threats & Challenges

IoT specific threats

- Handling of privacy sensitive data
  - Devices (e.g. smart home) are closely tied to users and their environment
  - Capture / analyze / export user related data
- Usage for critical applications
  - Industry: power (e.g. nuclear) plant, water treatment, etc.
  - Autonomous / remotely controllable systems: cars, drones, etc.

Increased impact of attack

- Easy access (physical)
  - Unattended devices (e.g. outdoor)
- Manufactured by non-security (even IT) experts
  - Low security measures implemented
  - E.g. 400,000 D-Link devices (39 models) vulnerable to single flaw[1]

Decreased effort to attack

[1] http://blog.senr.io/blog/400000-publicly-available-iot-devices-vulnerable-to-single-flaw

IoT heterogeneity

- Entity heterogeneity (3 tiers)[1]:
  - High-end devices (laptop, smartphone, tablets)
  - Low-end devices (sensors, actuators)
  - Passive entities (barcode, QR-code, RFID)

- Communication heterogeneity:
  - Wired communications (ethernet)
  - Long range wireless communications: WiFi / 3G / 4G
  - Short range wireless communications: Bluetooth (LE) / Zigbee / 6LoWPAN

→ **No single security solution for the "IoT"**

[1] Corvington and Carskadden "Threat Implications of the Internet of Things", 2013

Node Constraints [1]

- Maximum code complexity (ROM/Flash)
- Size of state buffers (RAM)
- Amount of computation feasible in a period of time (processing capabilities)
- Available power
- User interface and accessibility in deployment (set keys, update software)

[1] RFC7228 “Terminology for Constrained-Node Networks” (https://tools.ietf.org/html/rfc7228)

Network Constraints [1]

- Low achievable throughput
- High packet loss
- Asymmetric link characteristics
- Penalties for using large packets (e.g. high packet loss due to link layer fragmentation)
- Reachability over time (wake-up and sleeping time of devices)
- Lack of advance services (e.g. IP multicast)

[1] RFC7228 “Terminology for Constrained-Node Networks” (https://tools.ietf.org/html/rfc7228)

Designing secure IoT systems

Securing the IoT

- System (access control, authentication, etc.)
- Application
- Mobile
- Cloud
- Network (communications)

Recommended approaches

1. Threat analysis (e.g. RFC 3552)
2. Follow security recommendation (e.g. NIST, IETF, etc.)
3. Learn from attacks
4. Follow design patterns

1. Threat Analysis

- Define assumptions
  - E.g. “*the attacker has complete control of the communication channel*”
- Explore scenarios
  - Active vs. passive attacker
  - On-path vs. off-path
- Risk analysis ➡ Security requirements
- Fulfill requirements:
  - Authentication
  - Authorization
  - Traffic Security (confidentiality, data-origin, integrity, availability)
  - Non-repudiation (optional)

Threat Analysis: Limitations

Gives theoretic security requirements to meet

But: leaves room for interpretation in implementation

- Which layer to apply security protection to ?
- Which existing security frameworks to use ?

Complex to perform

- Difficult to be comprehensive
  - E.g. consideration of vulnerable devices used as attack vector

2. Security recommendations (e.g. NIST, IETF)

- Key management requirements [1]
- Key length recommendations [2]
- Randomness requirements [3]
- Avoid possibility of pervasive monitoring
- Protocol or domain specific recommendation (crypto algorithm, WLAN security, use of TLS / DTLS, etc.)

[1] RFC 4107 “Guidelines for Cryptographic Key Management” (https://tools.ietf.org/html/rfc4107)
[2] RFC 4492 “Elliptic Curve Cryptography Cipher Suites for TLS” (https://tools.ietf.org/html/rfc4492)
[3] RFC 4086 “Randomness Requirements for Security” (https://tools.ietf.org/html/rfc4086)

3. Learn form Attacks

Selected attacks to illustrate common problems:

- Inadequate software update mechanism
- Missing Key Management
- Insecure configuration files and default passwords
- Missing communication security
- Physical attacks

Inadequate software update mechanism

Example: Huawei Home gateway

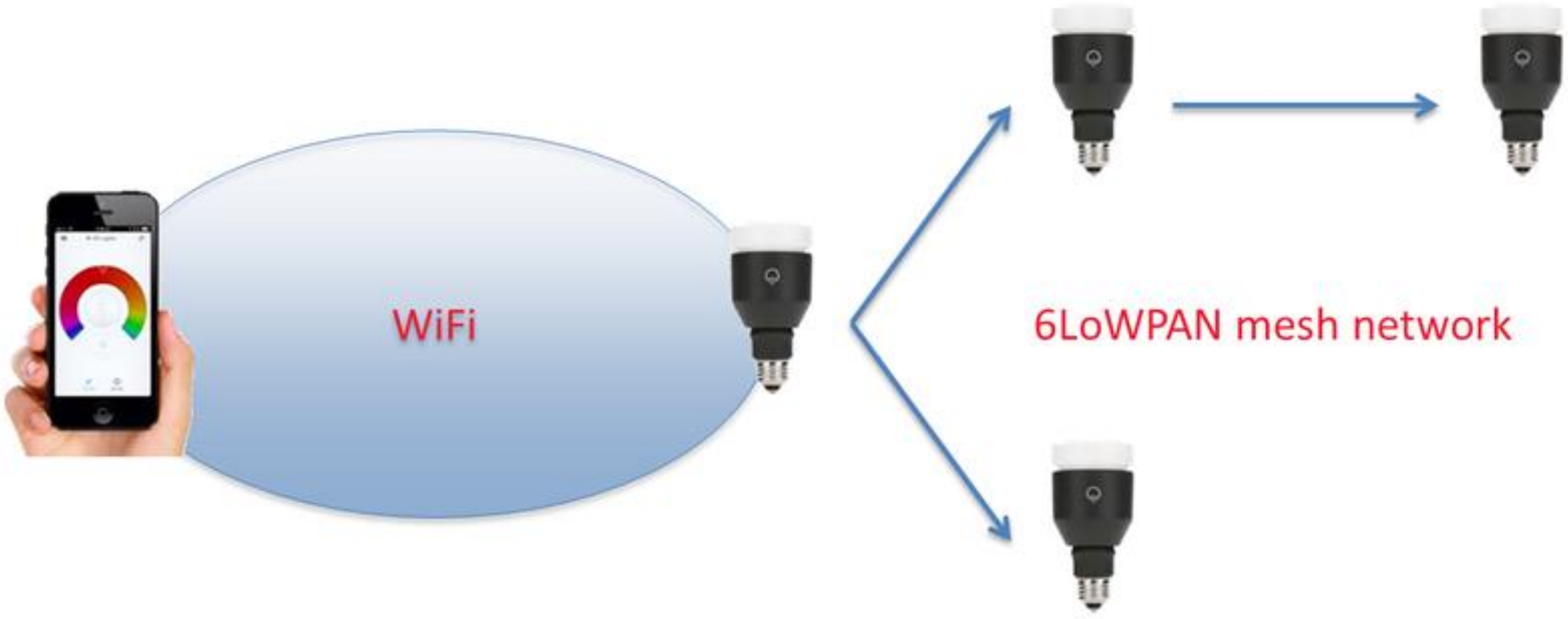
- Embedded web server (released 2002) with buffer overflow vulnerability
- Fix released in 2005 by web server company
- Vulnerability still exploited [1] (2015)

[1] http://www.computerworld.com/article/2860843/vulnerability-in-embedded-web-server-exposes-millions-of-routers-to-hacking.html

Missing Key Management Problem

Example: LIFX [1] - Internet connected light bulb

- AES key shared among all devices to simplify key management



[1] http://www.contextis.com/resources/blog/hacking-internet-connected-light-bulbs/

Insecure Configuration Files and Default Passwords

Example: Foscam, Linksys, Panasonic surveillance / baby monitoring cameras

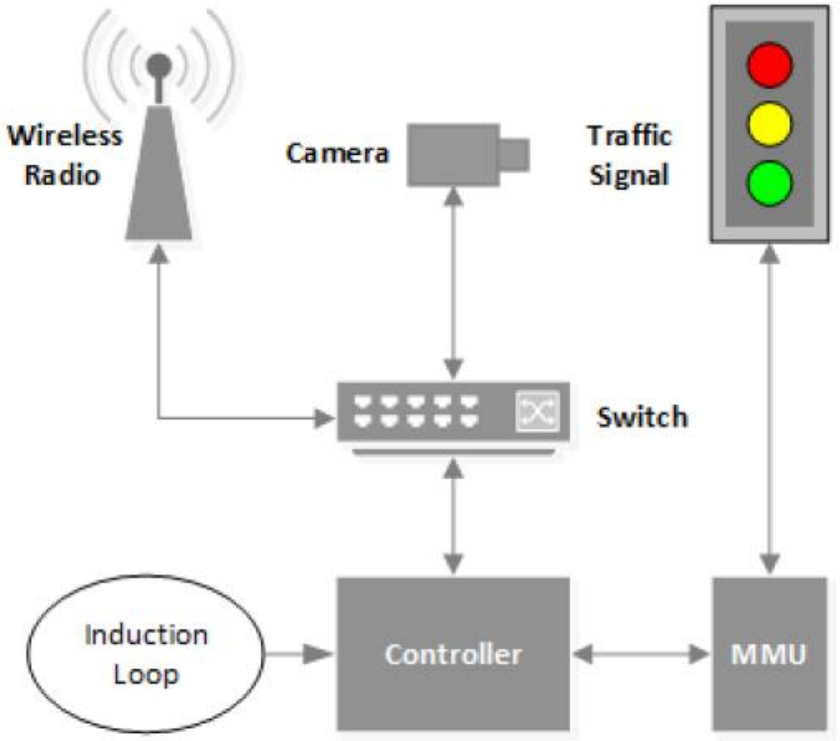
- Default passwords or insecure default settings
- Similar problems on LED bulbs, etc.

[1] http://www.networkworld.com/article/2844283/microsoft-subnet/peeping-into-73-000-unsecured-security-cameras-thanks-to-default-passwords.html

Missing Communication Security

Example: Traffic infrastructure [1]

- Unencrypted wireless communications
- Default Username and Passwords (published online by manufacturer)
- Controller settings can be configured remotely
- FTP connection to write configuration files
- Physical attacks

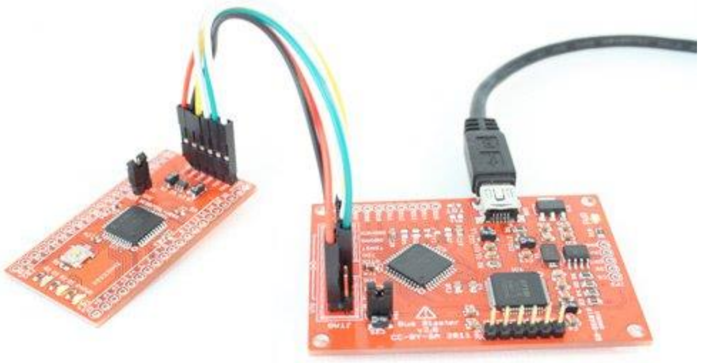


[1] Ghena,et al. Green “Lights Forever: Analyzing the Security of Traffic Infrastructure”

Physical Attacks

Example: extract keys, configuration data, firmware images

- Use of debug / test interfaces
- Sniffing on inter-bus communication interfaces be comprehensive
  - Serial Peripheral Interface (SPI)
  - Inter-Integrated Circuit (I^2C)
- Key extraction within a trusted execution environment
  - Power analysis
  - Fault injection (glitching) attacks



Intermediate Summary (methods 1/2/3)

- 90% of the threats are common among all protocols
- Most (exploited) security vulnerabilities are basic
- Many exploits of IoT systems (particularly in the consumer space) were hoaxs.
  But this is changing: Mirai botnet, Reaper botnet

4. Communications Design Patterns

- Device-to-Device
- Device via Smart Phone to Cloud
- Device via Gateway to Cloud
- P2P Communication in Local Network
- Device-to-Cloud

Device-to-Device Communication

Characteristics:

- Direct communications between devices
- Link layer communication protocol (often no IP)

Security:

- Direct association between devices: pairing
- Channel security provided at the link layer

Standardization:

- RFID, 6LowPAN, ZigBee
- Bluetooth Low Energy (LE)[1]

[1] https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy

Device via Smart Phone to Cloud

Characteristics:

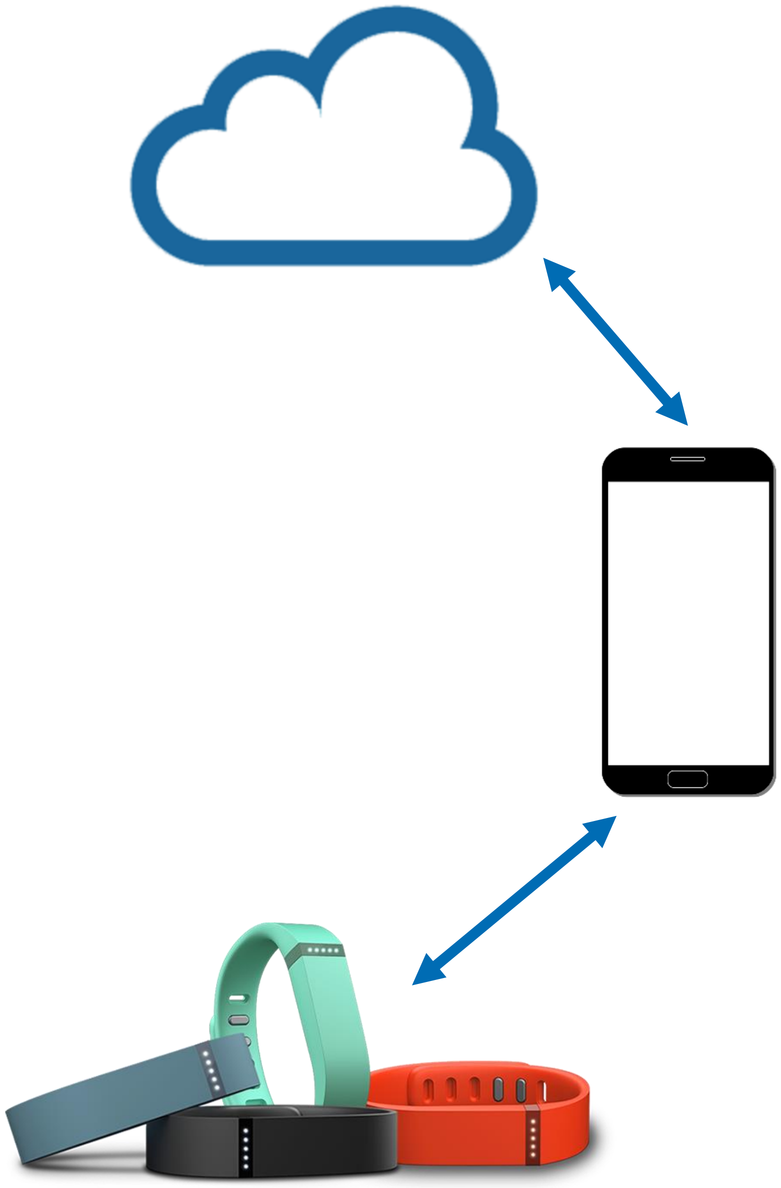
- Extension of *device-to-device* communication
- Device interacts with smart phone and cloud service

Security:

- D2D security
- Smart phone app / Web security
- Better provide end-to-end security (usually not the case)

Standardization:

- Bluetooth LE, NFC



Device via Gateway to Cloud

Characteristics:

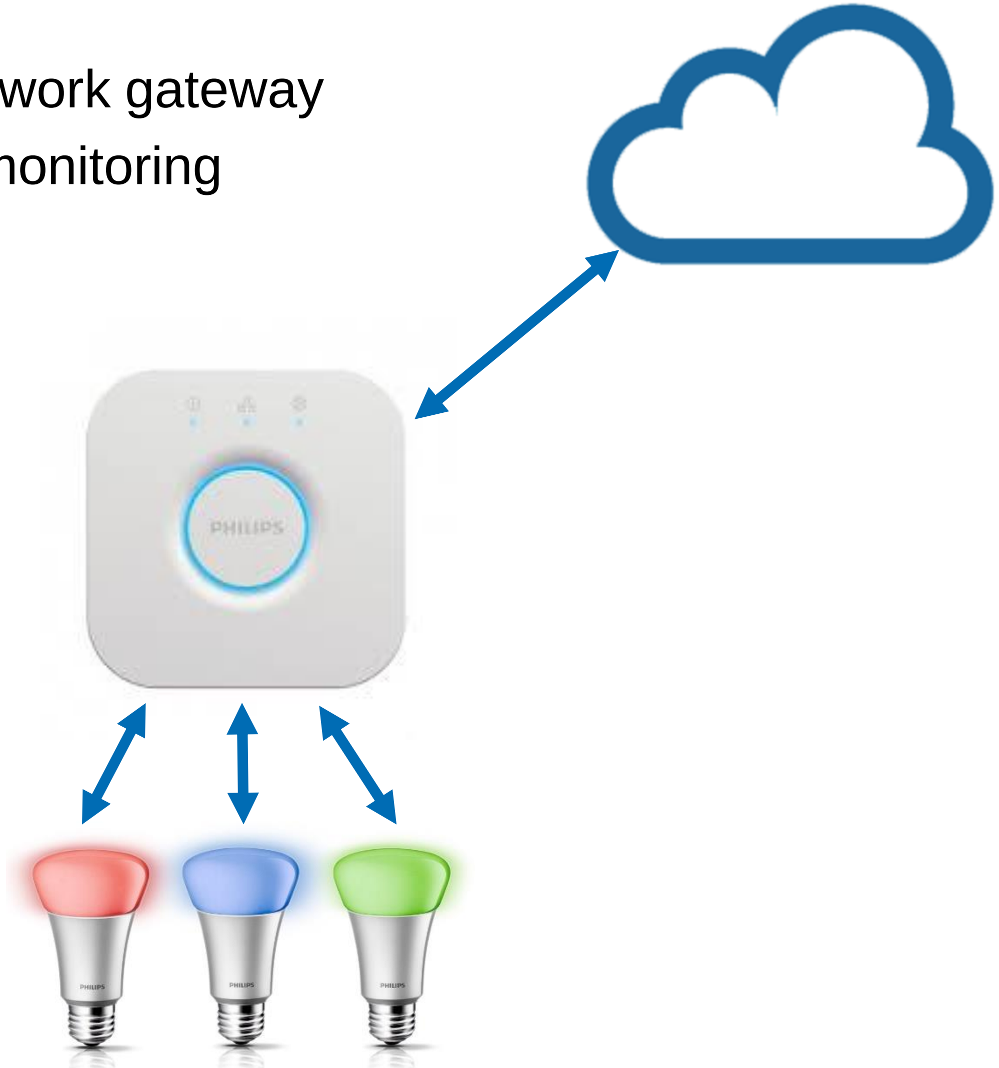
- Devices communicate with cloud services via a network gateway
- Apps/websites allow user-friendly, remote access/monitoring

Security:

- D2D security / No end-to-end security
- Authentication / pairing / key management
- Example: EAP, PANA, AAA, etc.

Standardization:

- IEEE 802.15.4, WiFi, Bluetooth LE



P2P Communication in Local Network

Characteristics:

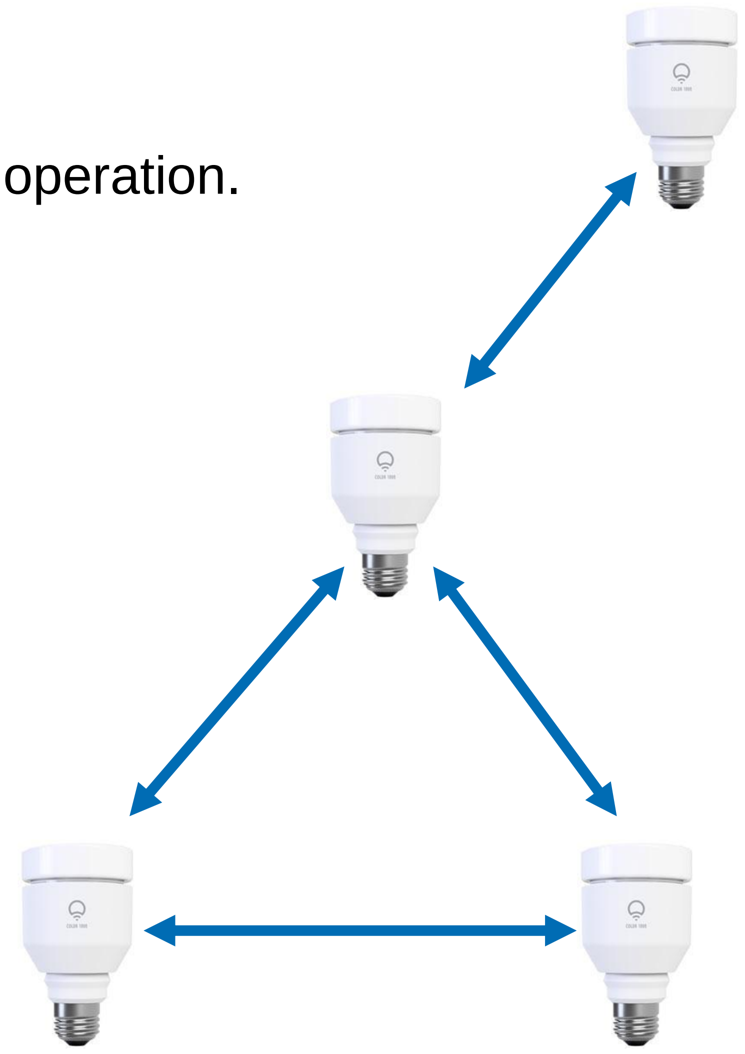
- Variant of “device via gateway to cloud” with local-only operation.
- Discovery of nodes to communicate with

Security:

- Authentication of nodes
- Trust management system
- Low need for protecting communications?

Standardization:

- Universal Plug and Play (UPnP) + UPnP-UP
- DNS Service Discovery
- Bonjour (Apple)



Device-to-Cloud

Characteristics:

- Direct communication with cloud service
- Pre-configured for specific cloud service only
- Always-on reachability required
- Radio technology and IP-connectivity required

Security:

- Network access + cloud authentication
- End-to-end security

Standardization:

- WiFi

Good Practices (recommendation)

- Always encrypt ➡ avoid pervasive monitoring
- Follow encryption key length recommendation (112-bit symmetric key equivalent)
- Support automatic key management
- Automatic software update mechanism
- Communication channel security (e.g. DTLS/TLS)
- Authentication and authorization solution
- Reduce physical attack surface:
  - Crypto implementations that consider side channel attacks
  - Disabled debug facilities before launching product
  - Hardware-based crypto support
  - Memory protection unit (MPU) integration

Research in IoT security:
1. IoT Sentinel

Problem: Vulnerable & unpatched IoT devices

Many deployed IoT devices have security vulnerabilities

- E.g. Mirai botnet (100,000s of compromised IoT devices)

Patching of devices is challenging

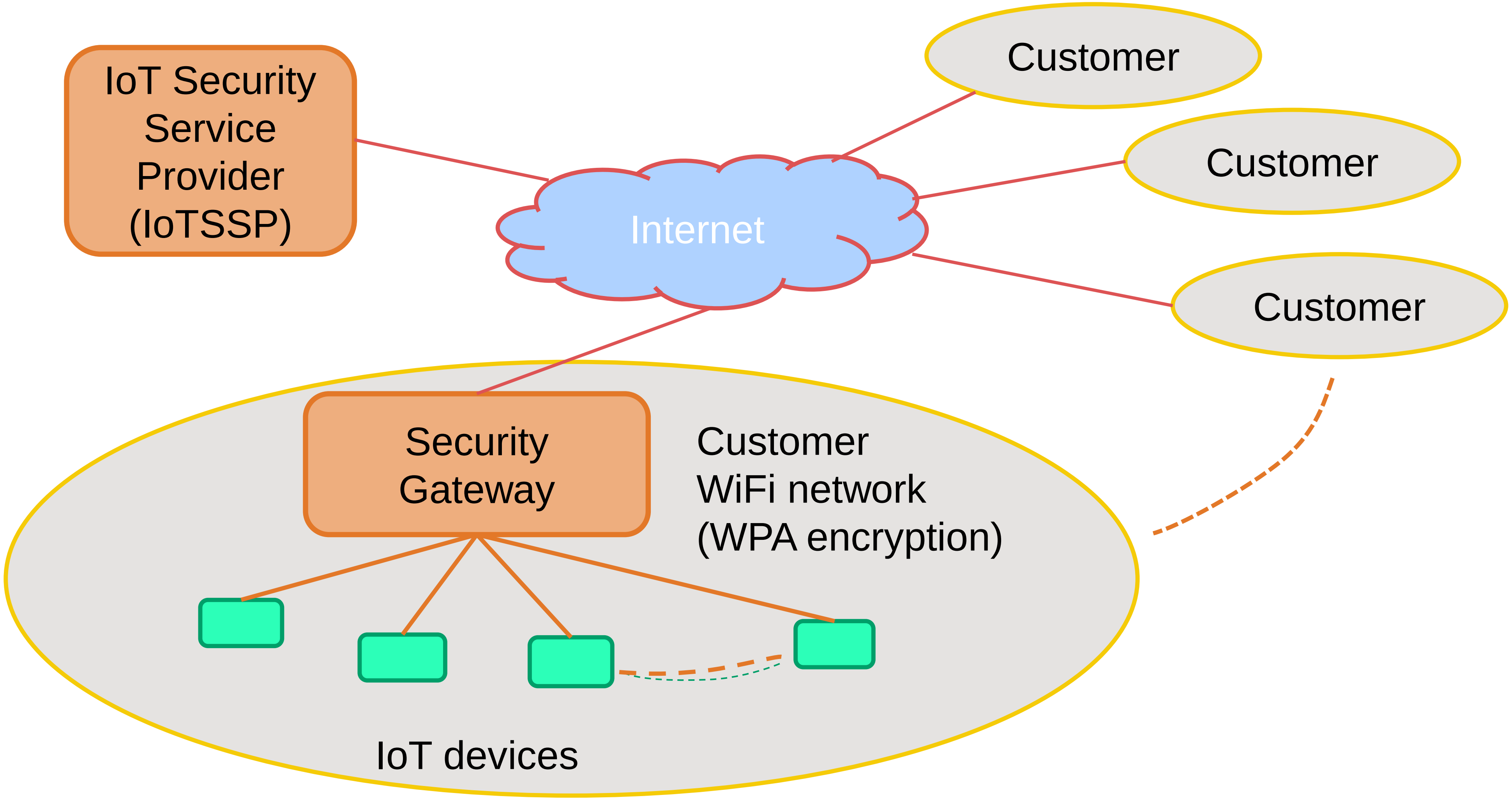
- Patches often not available
- Missing facilities for automatic patching

Need for a *brownfield* security solution accommodating the fact that some devices *are and will be vulnerable*

IoT Sentinel principles

1. **Identify the type (model / SW version) of new devices introduced to the network**
2. **Assign enforcement rules for device based on its type (isolate vulnerable types)**
3. **Constrain communications of vulnerable devices**
   - Avoid vulnerable devices to be compromised
   - Contain infection

System model



System model

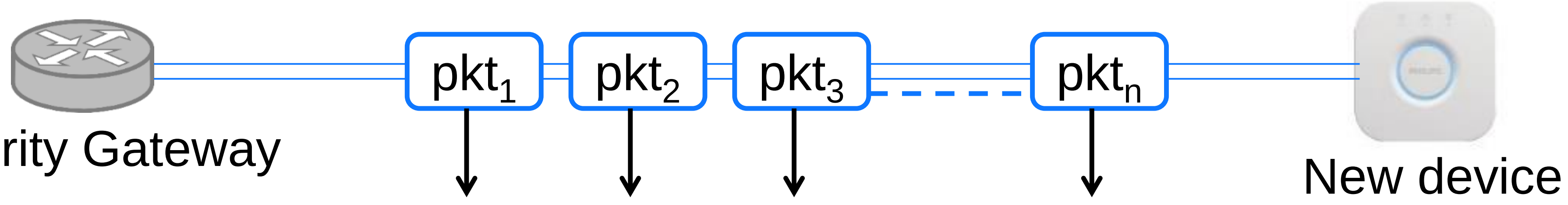
Security Gateway:

- Monitors communications and profiles devices: **extract a fingerprint**
- Sends fingerprints to IoTSSP
- **Applies enforcement rules**: device isolation through traffic filtering

IoT Security Service Provider (IoTSSP):

- **Identifies device-type** (using received fingerprints)
- Infers isolation level and enforcement rules for device-type (related to exposed vulnerability from e.g. CVE)
- Returns enforcement rules to be applied by Security Gateway

Device-type fingerprint



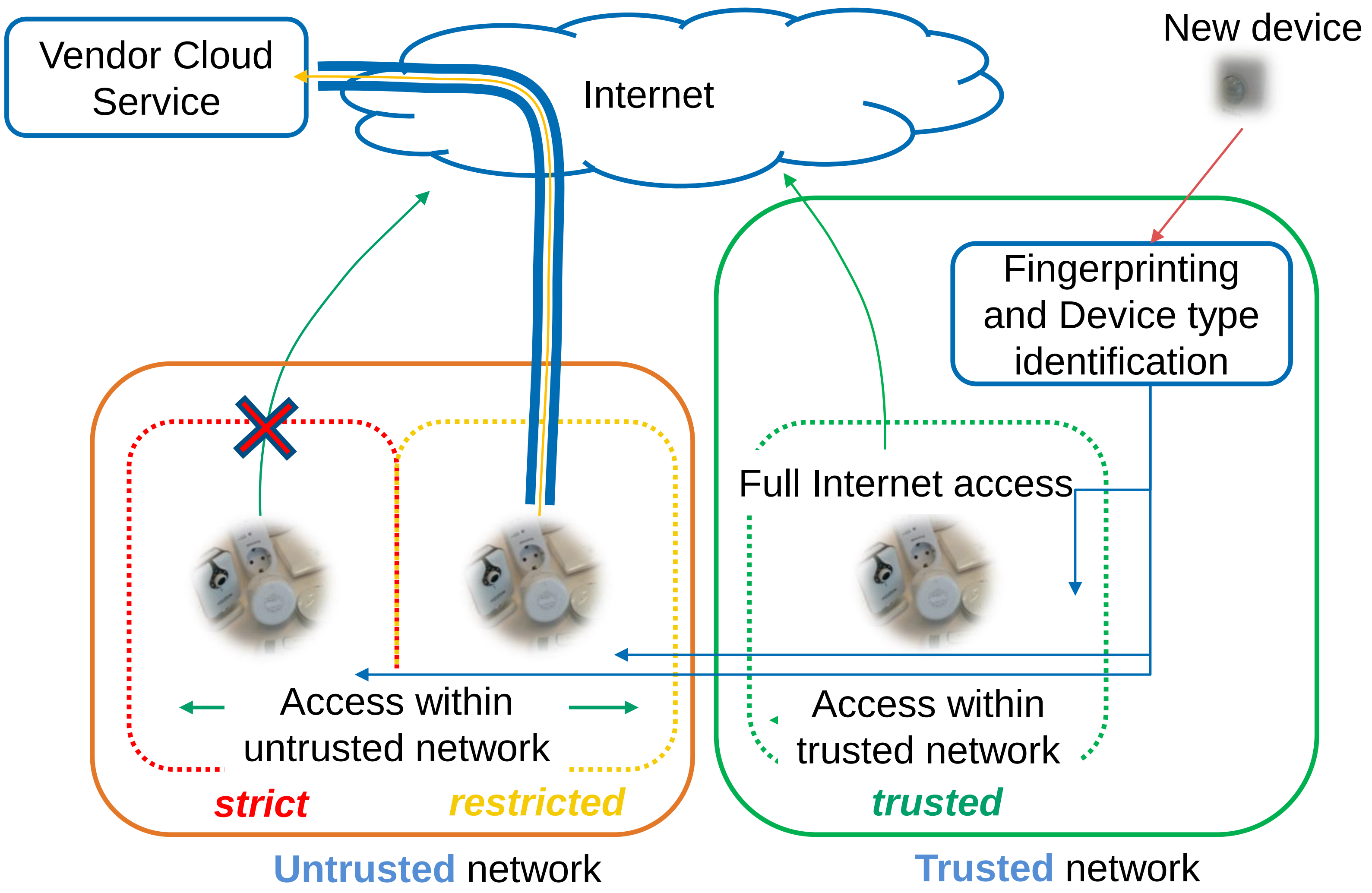
$$\mathbf{F} = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} & \cdots & f_{1,n} \\ f_{2,1} & f_{2,2} & f_{2,2} & \cdots & f_{2,n} \\ f_{3,1} & f_{3,2} & f_{3,3} & \cdots & f_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{23,1} & f_{23,2} & f_{23,3} & \cdots & f_{23,n} \end{pmatrix}$$

Features:

- 2 link layer protocol
- 4 network layer protocol
- 2 transport layer protocol
- 8 application layer protocol

- 2 IP options
- 2 packet content
- 1 IP address counter
- 2 port class

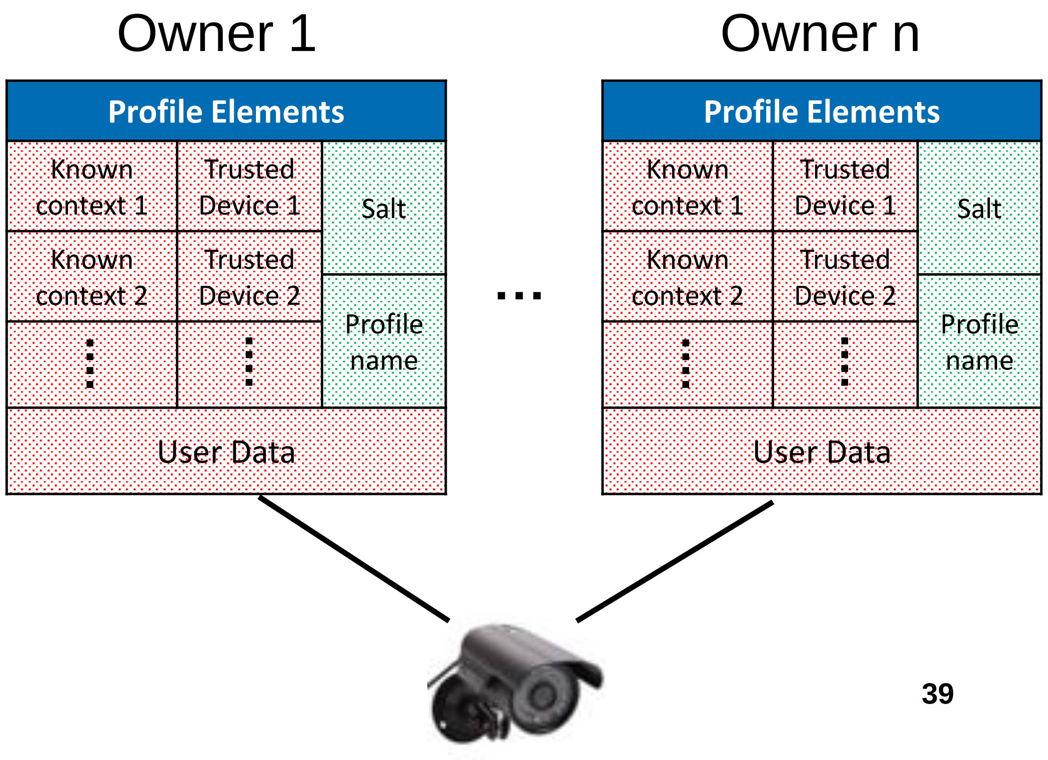
Device isolation



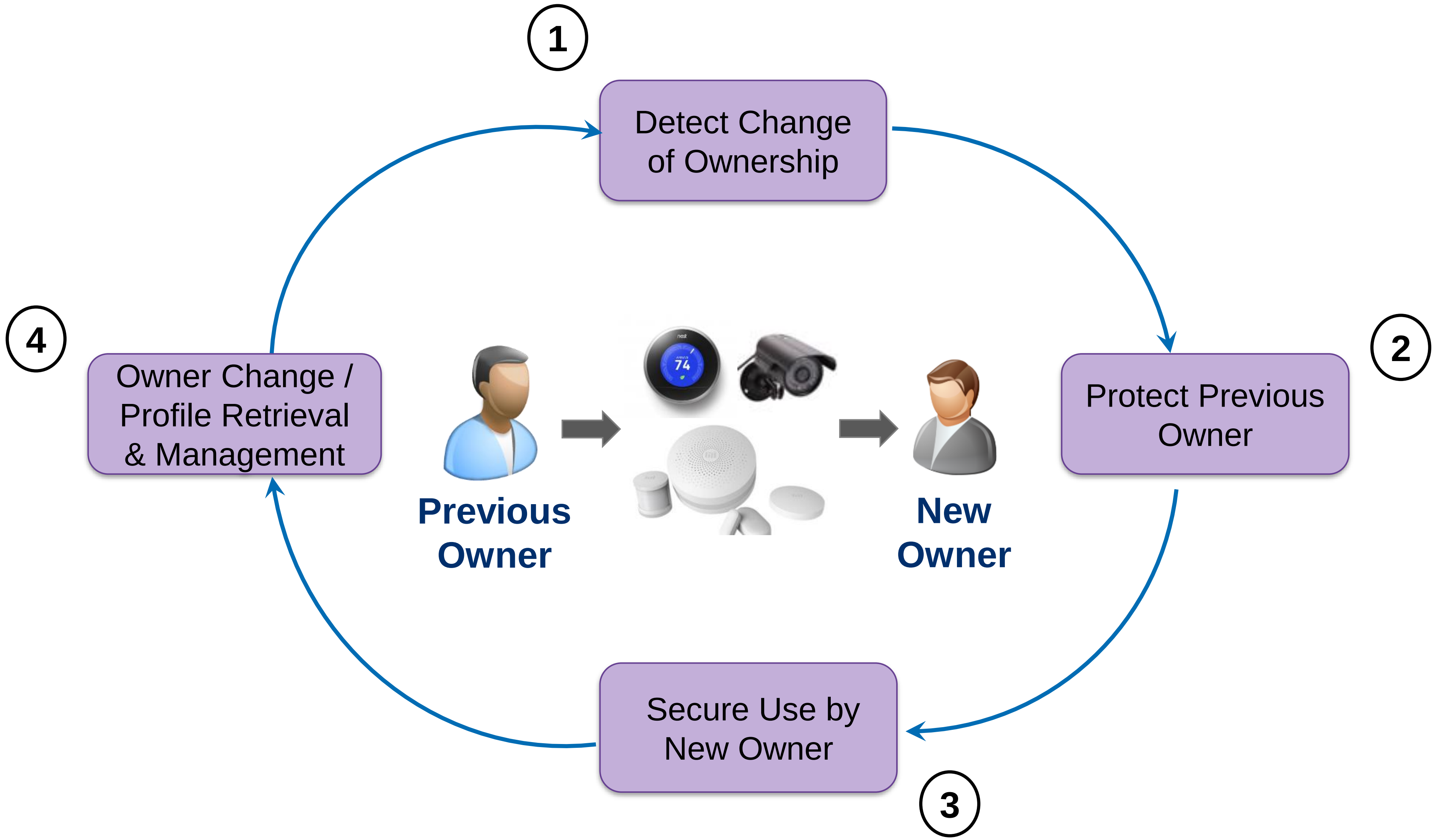
Research in IoT security:
2. ChownIoT

Problem: Ownership change of IoT devices

- Ownership of Smart Home (SH) device can change during lifetime
  - Lend
  - Resell / stealing
  - Change of tenant (rental places)
- May introduce unauthorized access to privacy sensitive data
  - Historical / personal data not wiped (threat to previous owner)
  - Authentication credentials for cloud / network access saved (threat to previous owner)
  - Old authentication credentials still valid (threat to new owner)
  - Etc.



chownIoT overview



Aalto University

Internet of Things (IoT) Security: Threats, Challenges & Recommendations

MSS – Lecture 6 (part 1)
Samuel Marchal

Attestation in the Internet of Things (IoT)

MSS – Lecture 6 (part 2)
Andrew Paverd

You will learn about:

- Why do we need (remote) attestation in IoT?
- What technologies can we use for this?

Motivating example

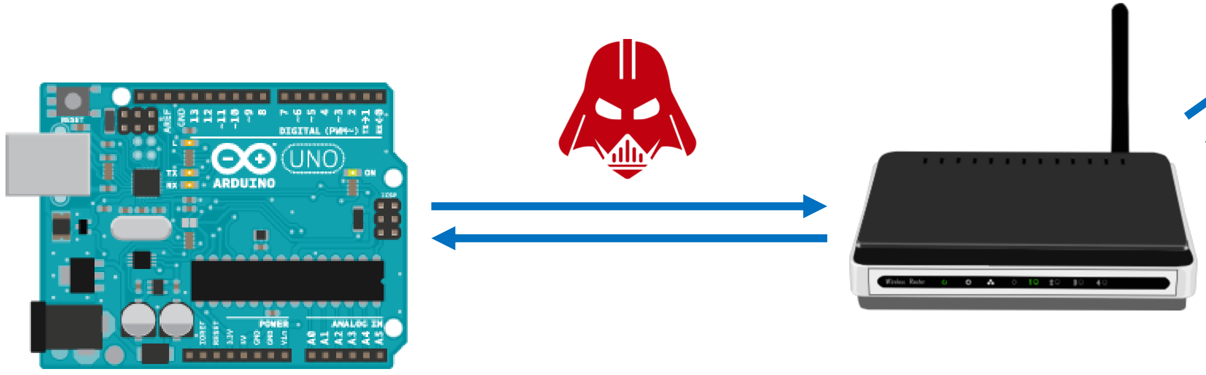
You receive the following message:

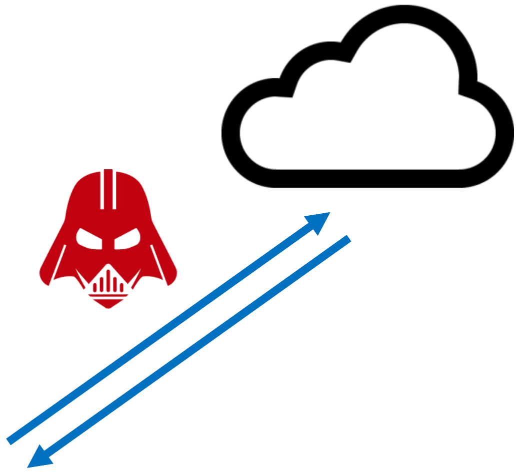

```
{
        "name": temperature,
        "value": 23.5,
        "timestamp": 1430905326.2
}
```


What does it mean?

Motivating example

Where is the adversary?





Network adversary: read, modify, falsify communication

Motivating example

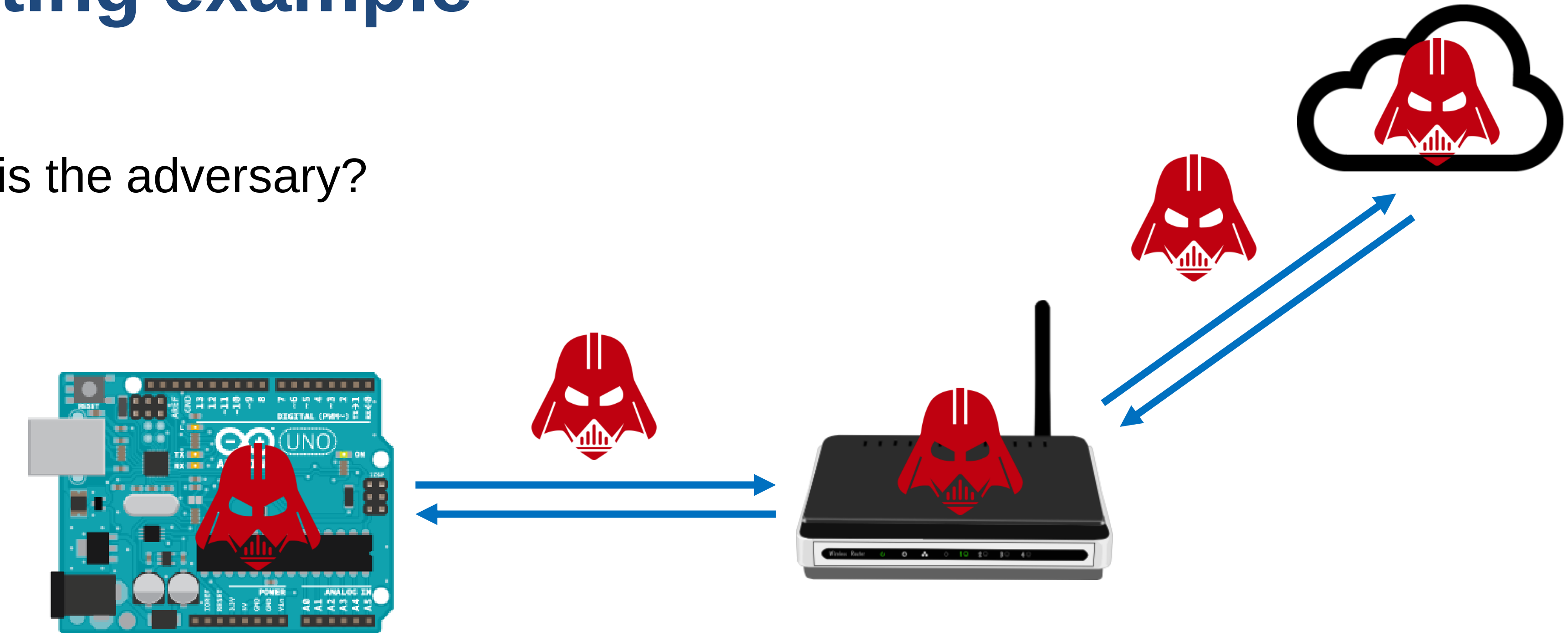
You receive the following message over an authenticated, integrity-protected communication channel:

```
{
        "name": temperature,
        "value": 23.5,
        "timestamp": 1430905326.2
}
```

What does it mean?

Motivating example

Where is the adversary?



Network adversary: read, modify, falsify communication

Malware: extract secrets, change state, modify behaviour

IoT malware: Stuxnet

WIRED An Unprecedented Look at Stuxnet, the World's Firs... SUBSCRIBE

BUSINESS | CULTURE | DESIGN | GEAR | SCIENCE | SECURITY | TRANSPORTATION

KIM ZETTER SECURITY 11.03.14 6:30 AM

AN UNPRECEDENTED LOOK AT STUXNET, THE WORLD'S FIRST DIGITAL WEAPON

The Real Story of Stuxnet

How Kaspersky Lab tracked down the malware that stymied Iran's nuclear-fuel enrichment program

By David Kushner
Posted 26 Feb 2013 | 14:00 GMT

IoT malware: Stuxnet

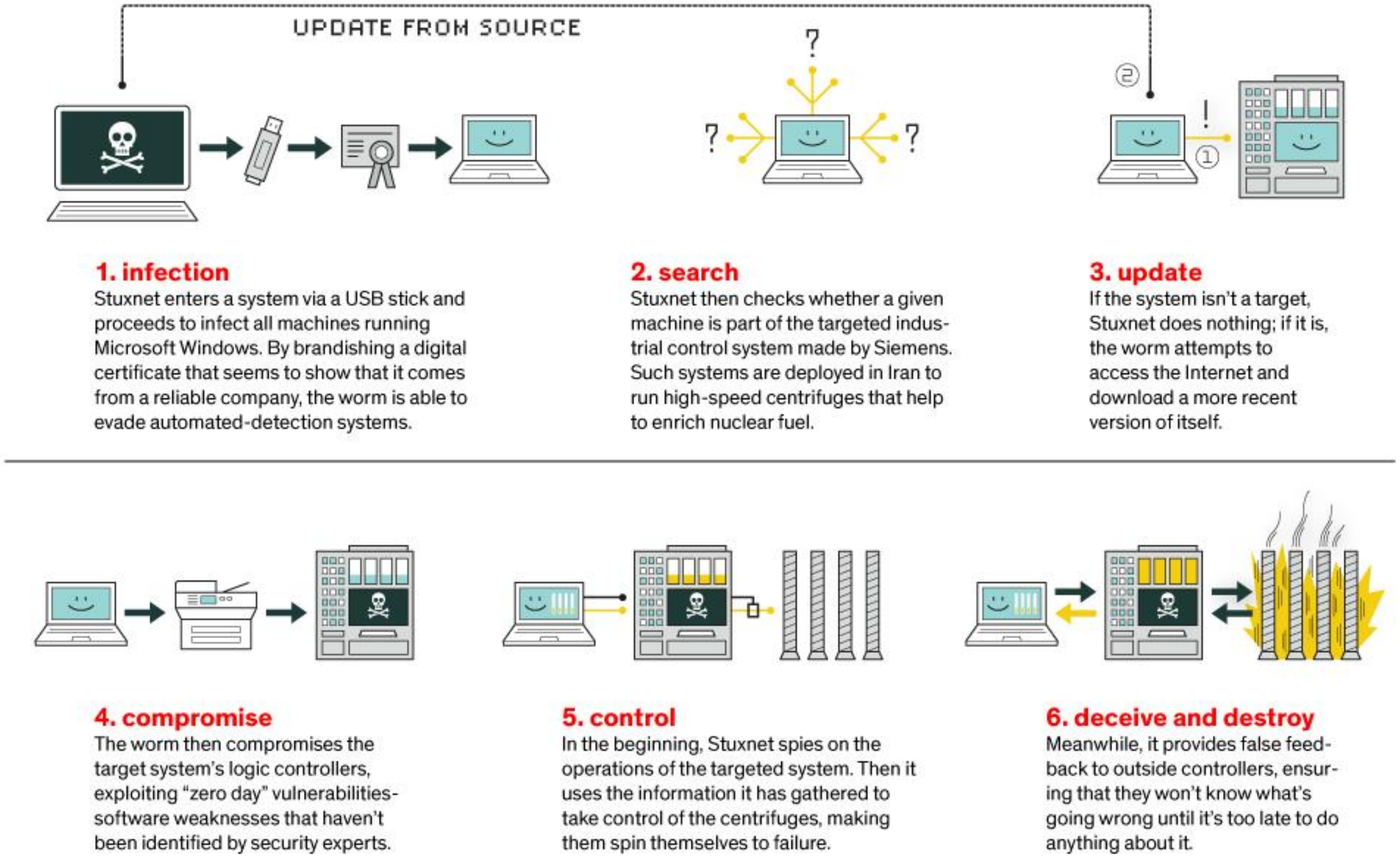


Illustration: L-Dopa
http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet

IoT malware

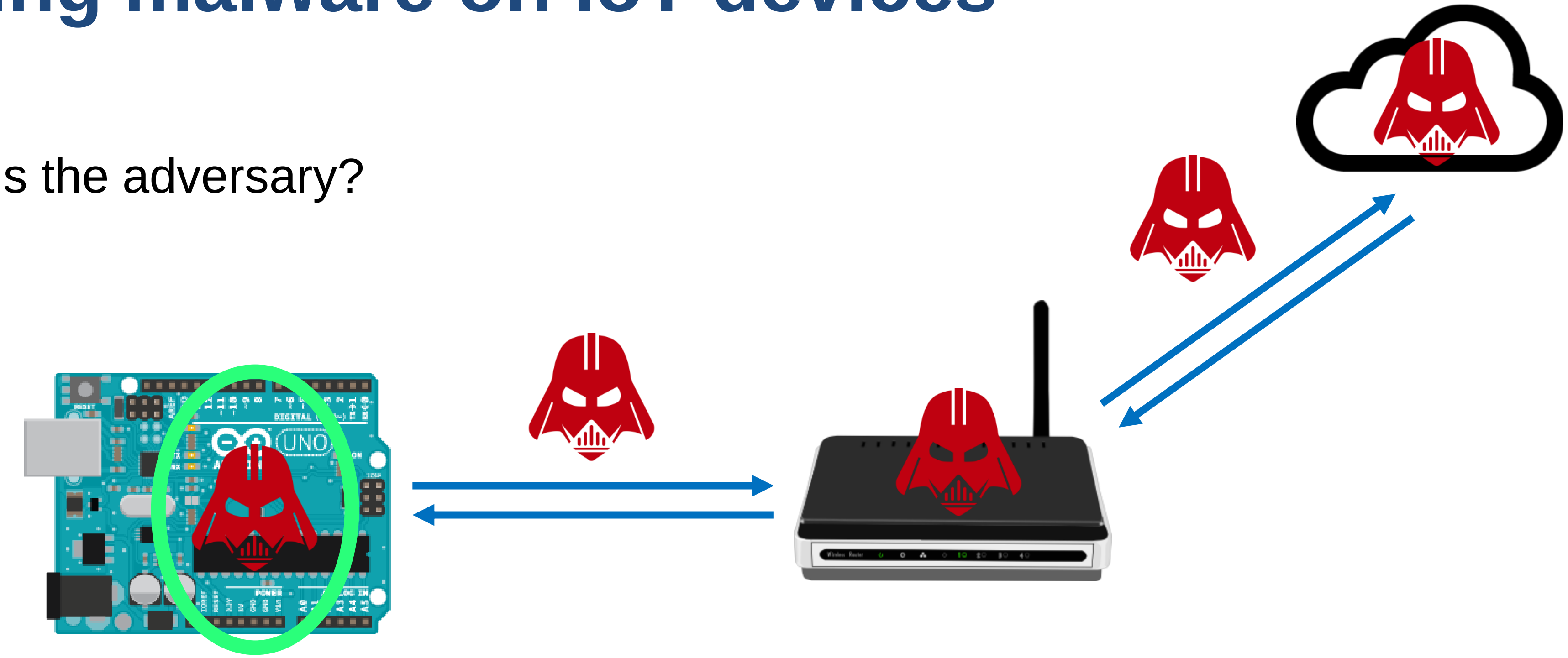
Malware is not a new threat, but IoT…

- Broadens the attack surface
  - cost-constrained and/or resource-constrained devices
  - many more interconnected devices

- Amplifies the impact
  - access to detailed personal information
  - control of physical environment

Detecting malware on IoT devices

Where is the adversary?

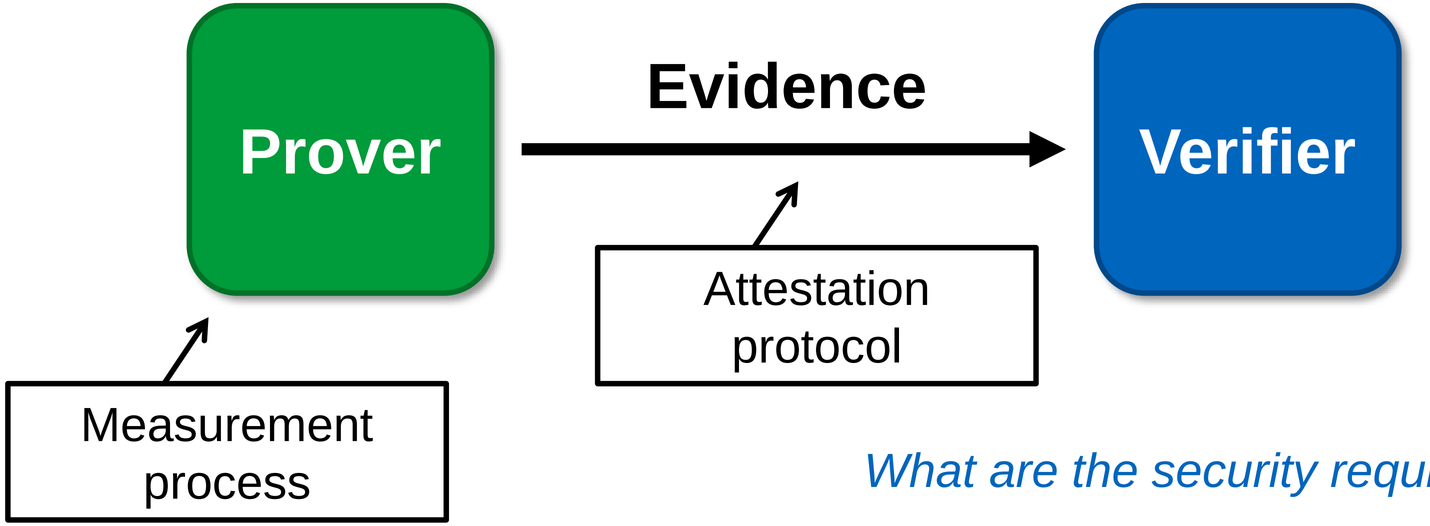


Network adversary: read, modify, falsify communication

Malware: extract secrets, change state, modify behaviour

Attestation

- An interaction between two parties through which the *verifier* ascertains the current state and/or behaviour of the *prover*.



What are the security requirements?

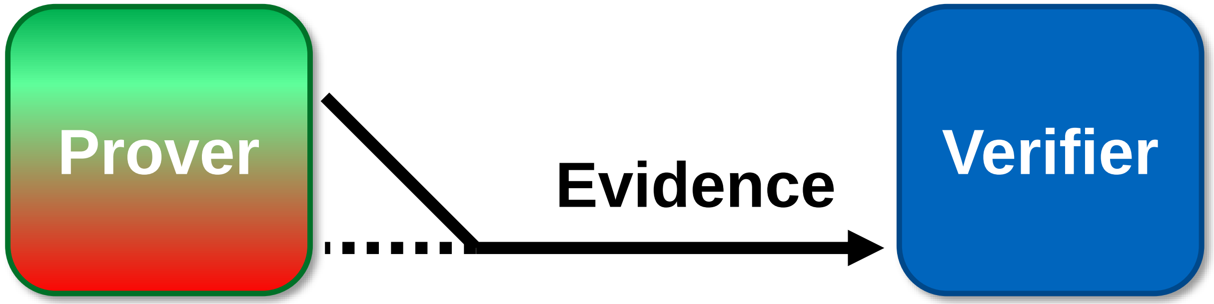
Attestation Requirements

1. Authenticity
   – representation of the *real* state of the system

Attestation Requirements

1. Authenticity
   – representation of the *real* state of the system

2. “Timeliness”
   – representation of the *current* state



TPM Attestation (Lecture 4)

- TPM Platform Configuration Registers (PCRs)
  - store cryptographic hash
  - cannot be over-written by software, only *extended*

PCR_1 = 0x00

tpm_extend(PCR_1 $hash_1$)

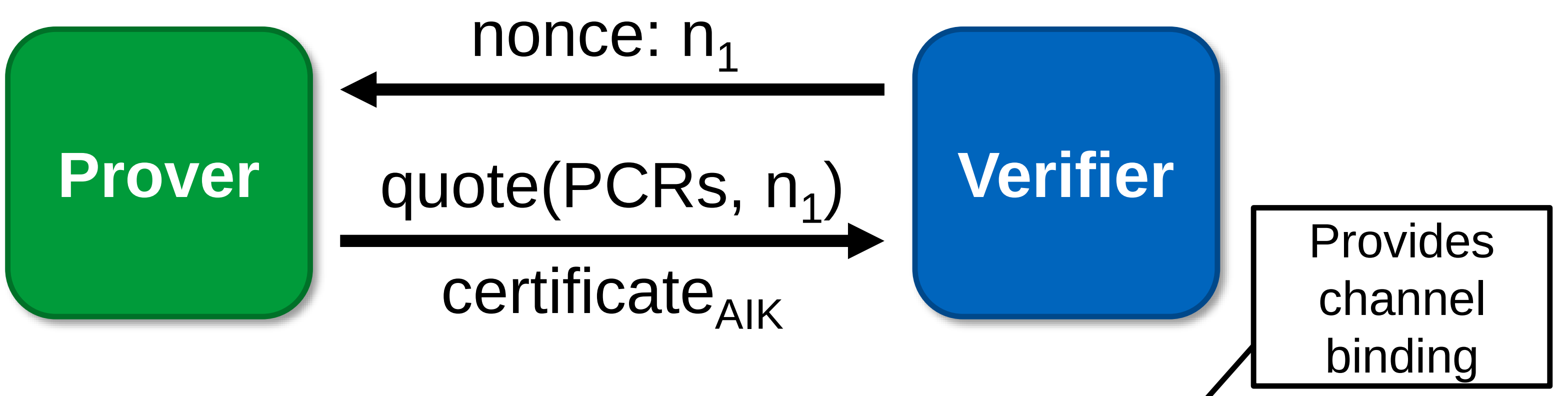
PCR_1 = hash(0x00 || $hash_1$)

tpm_extend(PCR_1 $hash_2$)

PCR_1 = hash(hash(0x00 || $hash_1$) || $hash_2$)

TPM Attestation (Lecture 4)

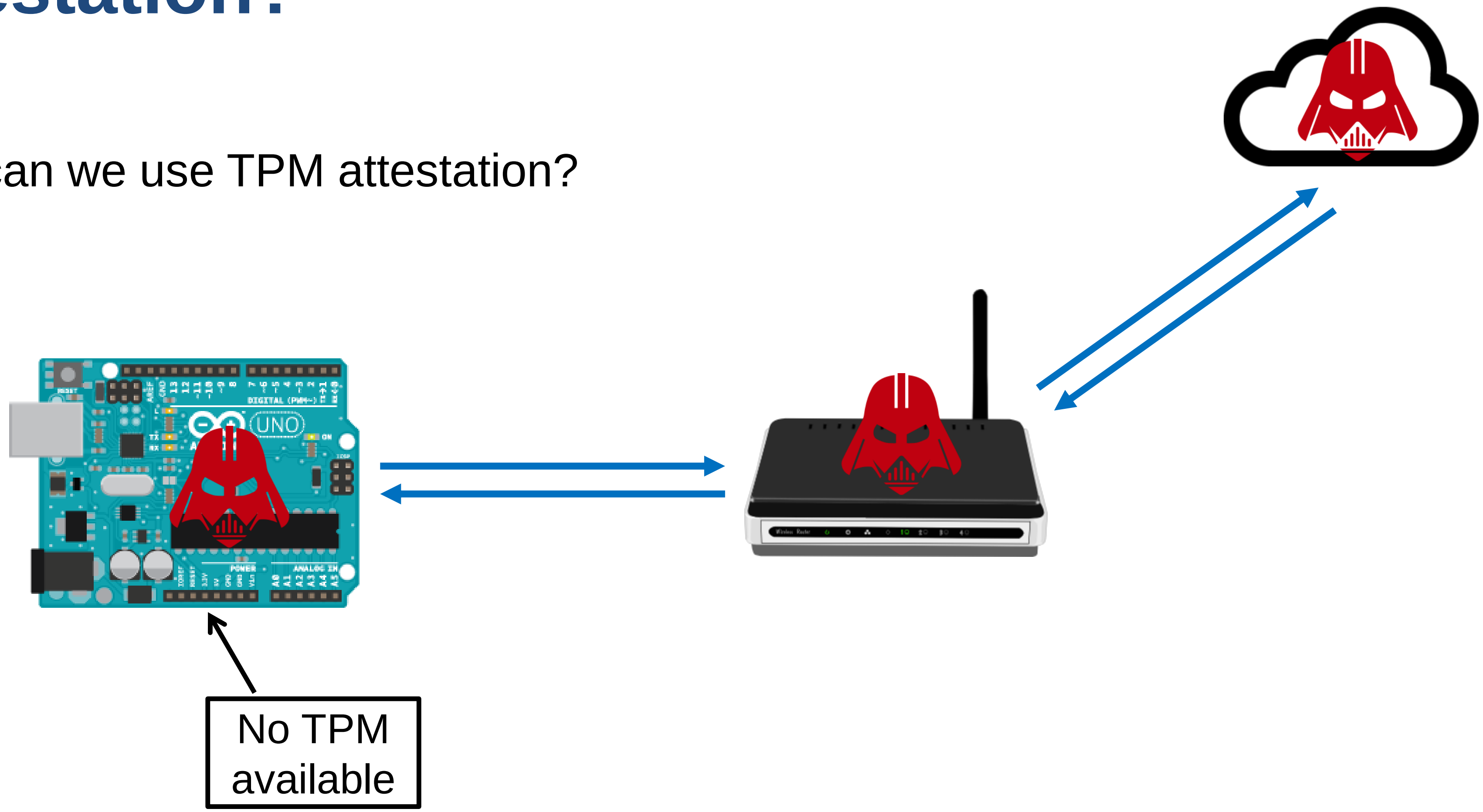
- TPM Quote
  - PCR values signed by TPM-bound Attestation Identity Key (AIK)
  - includes nonce to ensure timeliness



$$\text{quote(PCRs, } n_1\text{)} = \text{signature}_{AIK}\text{(PCRs, } n_1 \text{ || channel info)}$$

IoT attestation?

Where can we use TPM attestation?



TPM Attestation “Costs”

- Additional hardware
  - takes up space
  - uses power
  - increases hardware cost (TPM + integration)

- Additional software
  - requires driver and software library

TPM Attestation Limitations

- Covers only the initial loading of software
- Deals with only one prover and one verifier
- *“Decision of trustworthiness”* does not scale
  - measurements change with every software update

Attestation of Things (AoT)

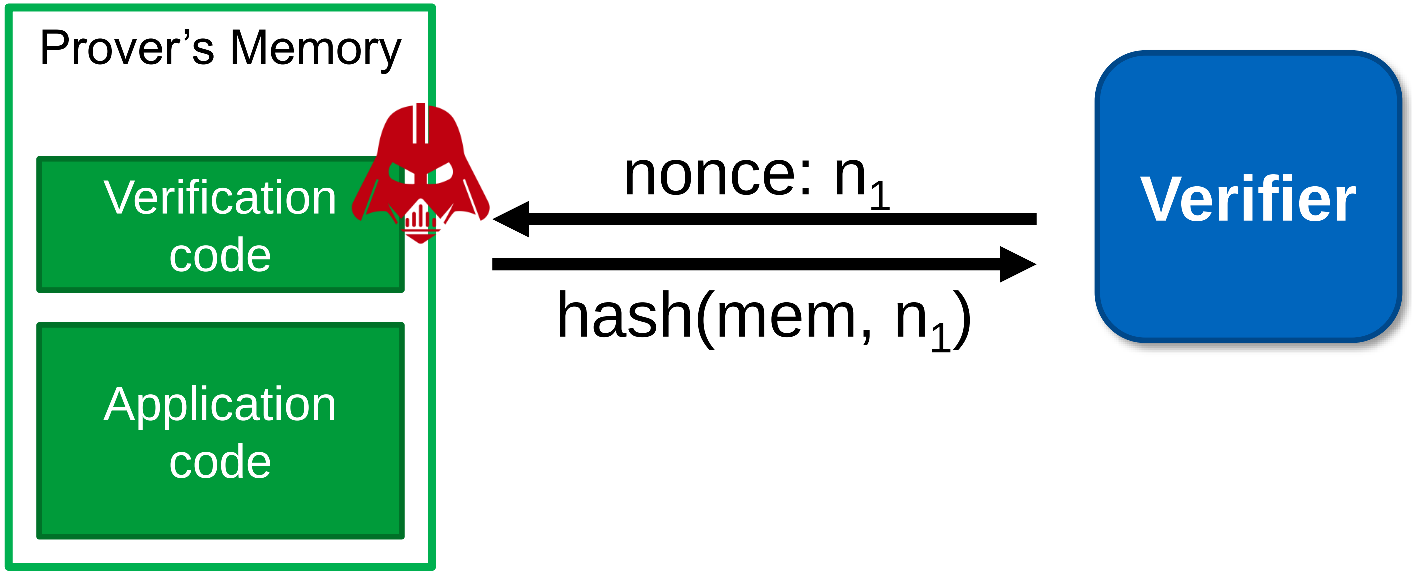
- Software-based attestation
- Hybrid attestation
- Scalability of attestation
- Run-time attestation

Attestation of Things (AoT)

- Software-based attestation
- Hybrid attestation
- Scalability of attestation
- Run-time attestation

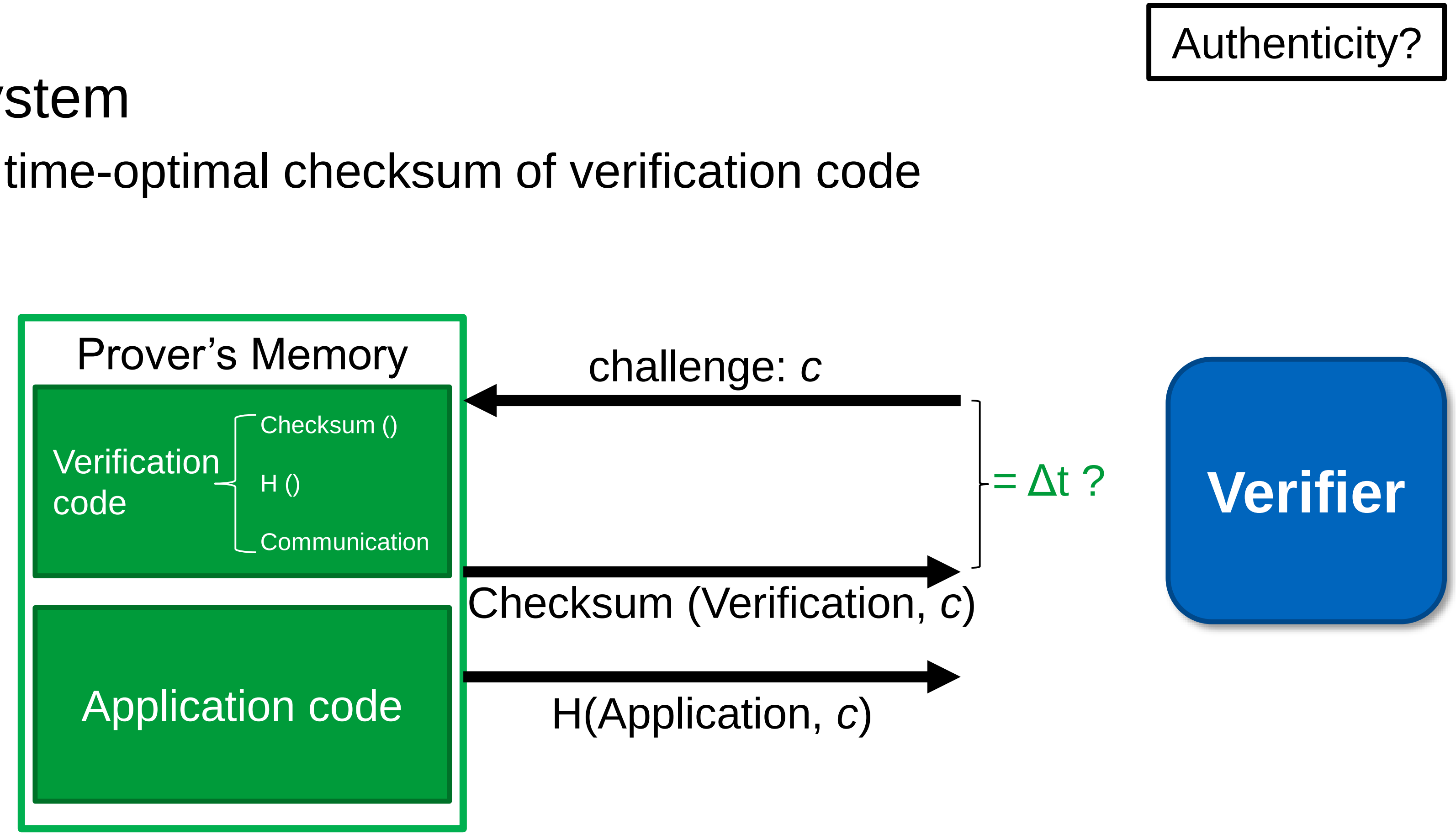
Software-based Attestation

- Assumes no hardware features to support attestation
  - No secrets on prover (e.g. no AIK)
  - Cannot guarantee specific code being run



Software-based Attestation

- Pioneer system
  - compute time-optimal checksum of verification code



A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. **Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms**. SOSP '05

Software-based Attestation: summary

- Limitations of timing side channels
  - verifier must know exact hardware configuration
  - difficult to prove time-optimality
  - limited to “one-hop” networks
  - requires authenticated channel (e.g. physical connection)

Attestation of Things (AoT)

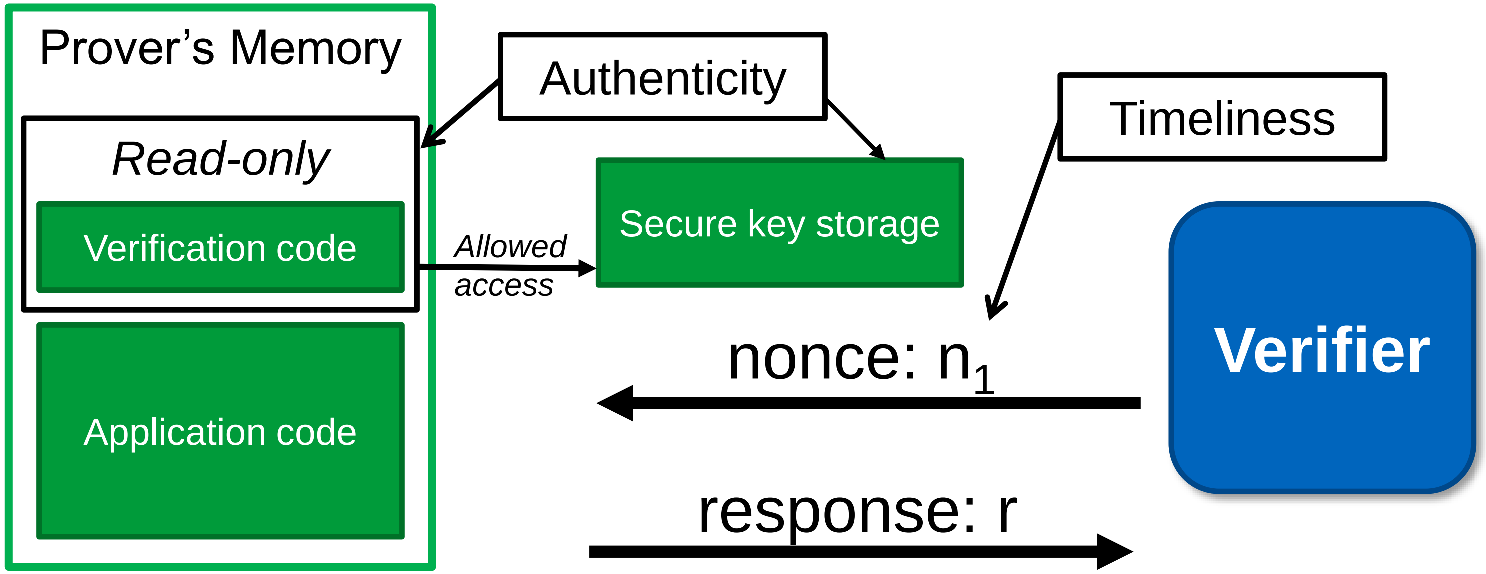
- Software-based attestation
- Hybrid attestation
- Scalability of attestation
- Run-time attestation

Hybrid Attestation

- Minimal trust anchors
  - small changes to hardware
  - “hardware/software co-design”

Hybrid Attestation: SMART

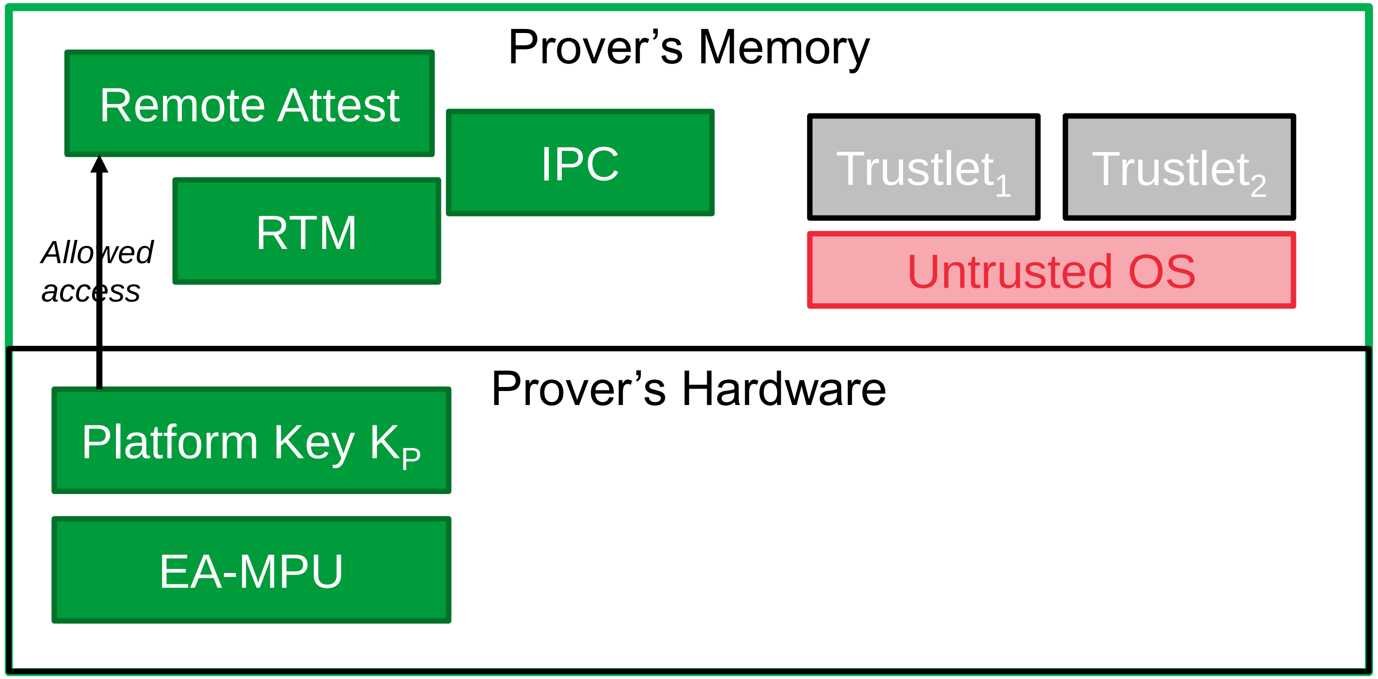
Read-only Verification code, secure key storage and atomicity of execution of Verification code



K El Defrawy, A. Francillon, D. Perito, and G. Tsudik. **SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust**. NDSS '12

Hybrid Attestation: TrustLite & TyTAN

- Execution-Aware Memory Protection Unit (EA-MPU)
  - access control based on memory request target and origin



- P. Koeberl, et al. **TrustLite: A Security Architecture for Tiny Embedded Devices**. EuroSys '14
- F. Brasser, et al, **TyTAN: Tiny Trust Anchor for Tiny Devices,** DAC '15

Hybrid Attestation: summary

- Advantages of hybrid approaches
  - can be used across a network / over an untrusted channel
  - Verifier need not know prover's exact hardware configuration

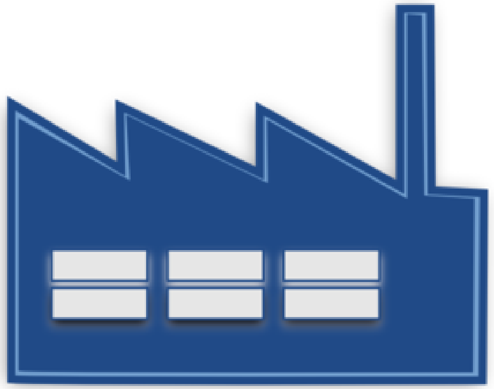
- Drawbacks
  - Needs additional hardware support
  - But minimal MCU trust anchors soon available commercially
    - TrustZone-M (ARM v8), …

Attestation of Things (AoT)

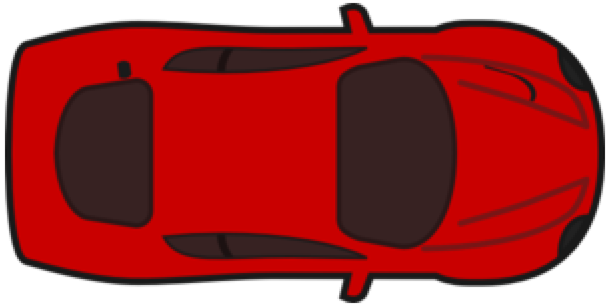
- Software-based attestation
- Hybrid attestation
- Scalability of attestation
- Run-time attestation

Scalability of Attestation

- Attestation protocols usually assume a single prover
  - but IoT scenarios may involve groups of (many) provers



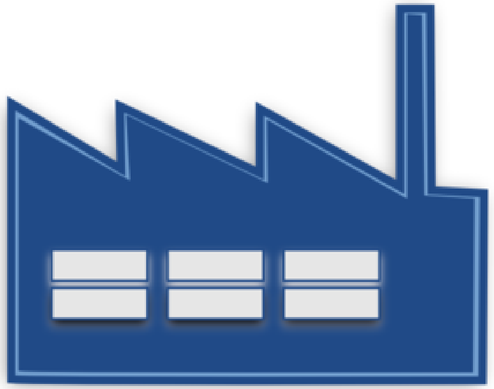
Smart factories



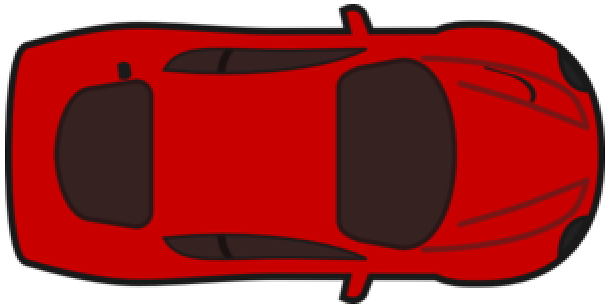
Smart vehicles

Scalability of Attestation

- Device *swarms*
  - dynamic topology: nodes move within swarm
  - dynamic membership: nodes join and leave the swarm



Smart factories



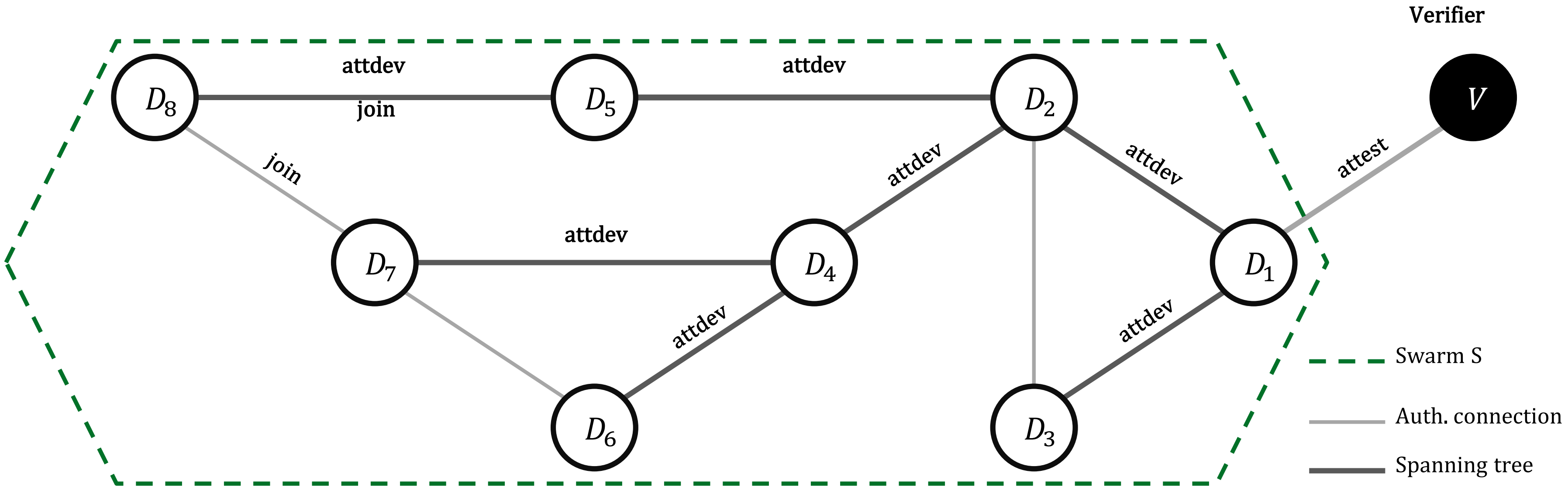
Smart vehicles

Scalability of Attestation

- SEDA: Scalable Embedded Device Attestation
  - more efficient than attesting each node individually
  - can use any type of measurement process
  - assumes homogeneous provers

N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

Scalability of Attestation



[N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS ‘15]

Scalability of Attestation: summary

How to extend SEDA to

- support highly dynamic swarms?
- be resilient to physical compromise of some devices?

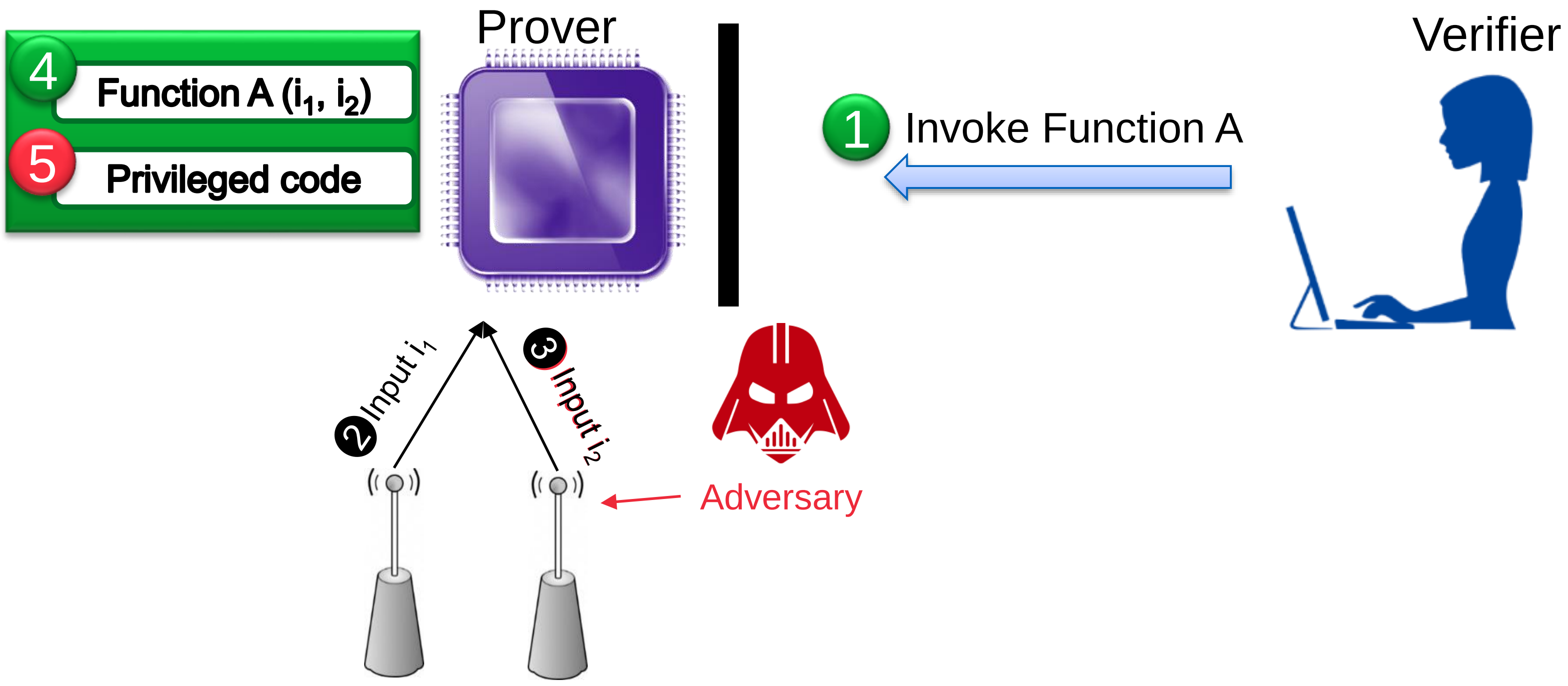
Attestation of Things (AoT)

- Software-based attestation
- Hybrid attestation
- Scalability of attestation
- Run-time attestation

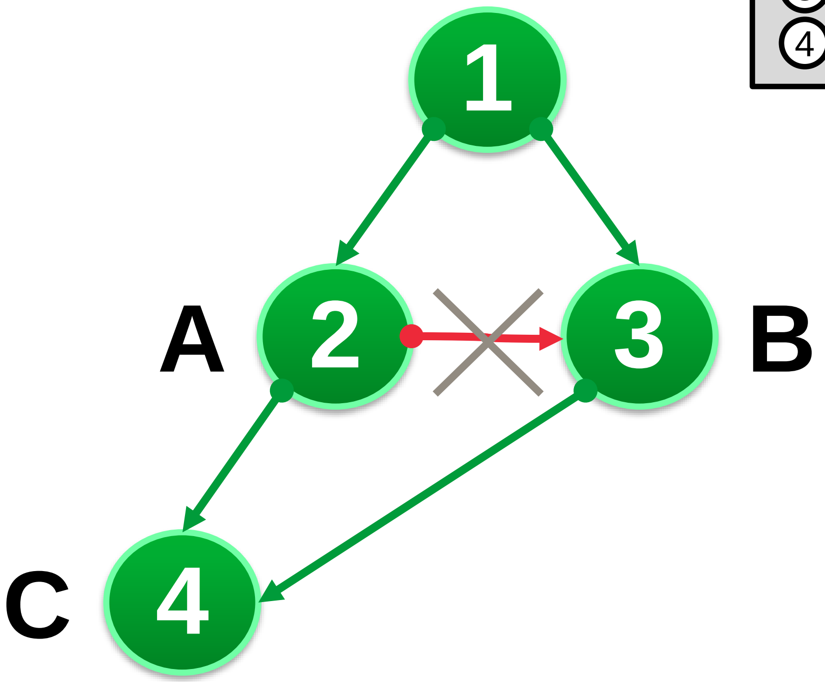
Why run-time attestation?

- Traditional attestation measures binaries at load time
- Cannot capture run-time attacks
  - return-oriented programming
  - control data attacks

Run-time attacks



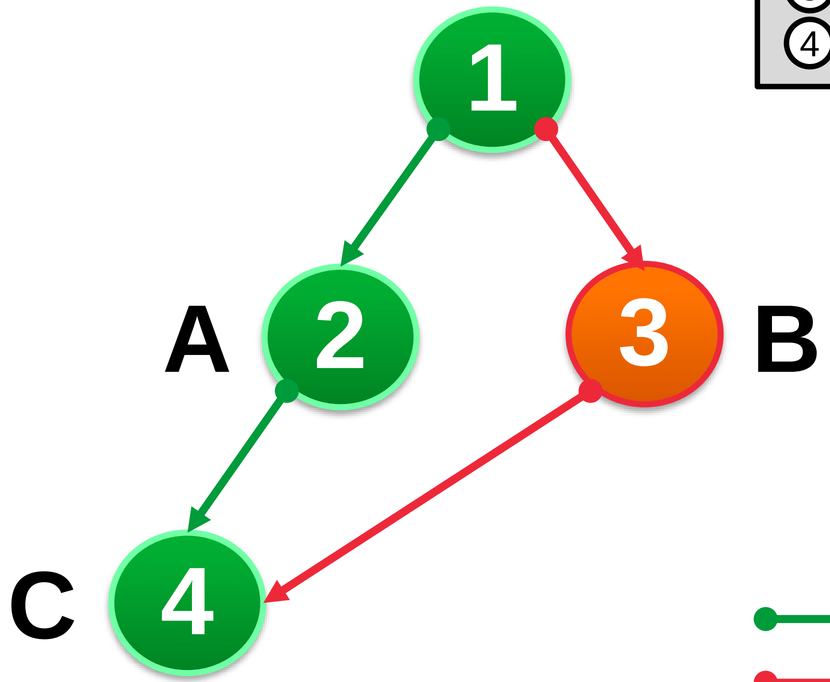
Control flow integrity (CFI)

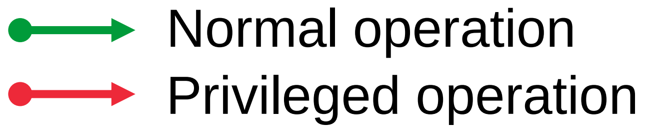



```
① if (condition)
②  then: block A
③  else: block B
④ block C
```

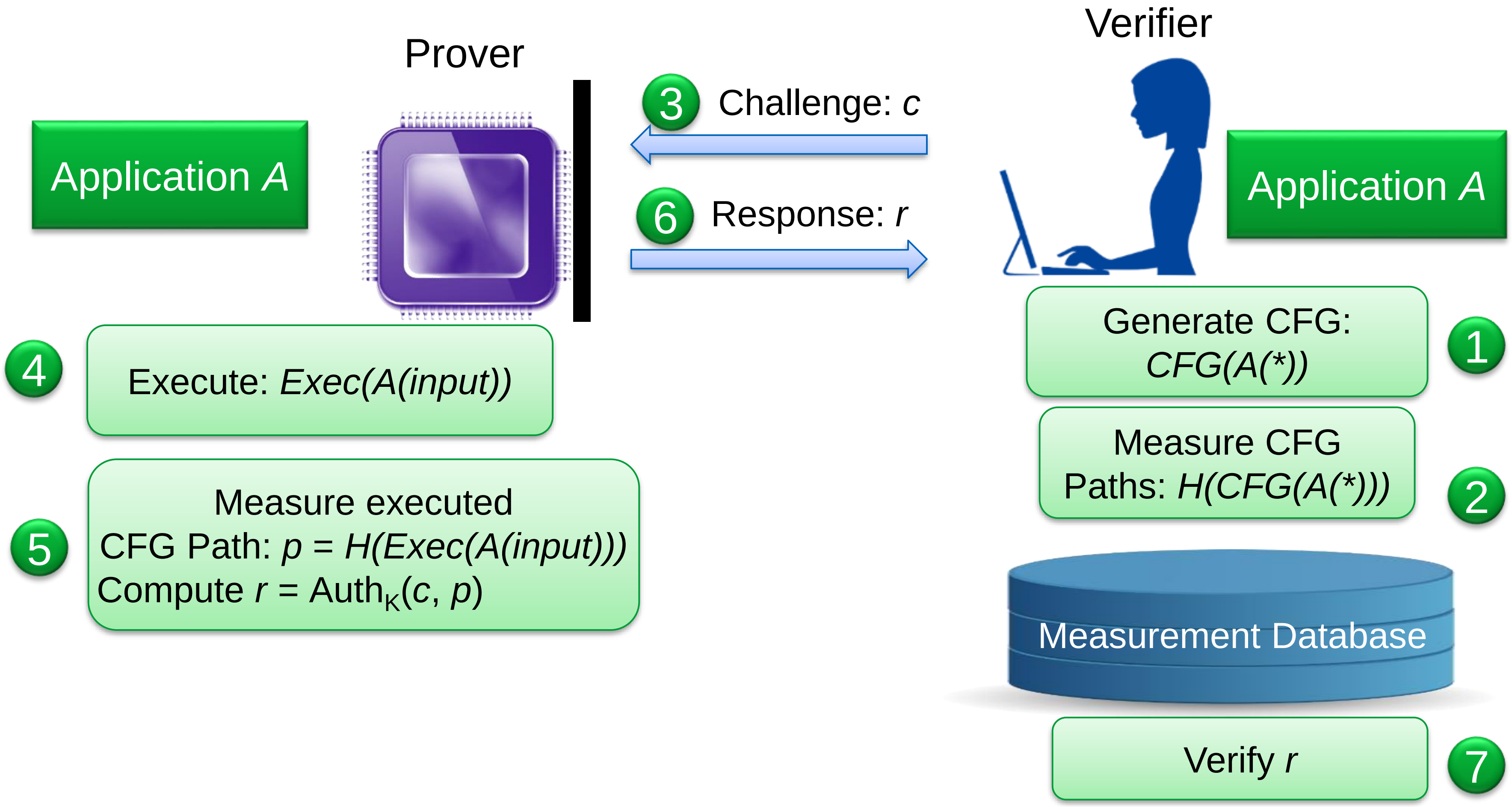
Run-time attacks without violating CFI

```
① if (condition)
②   then: block A
③   else: block B
④ block C
```





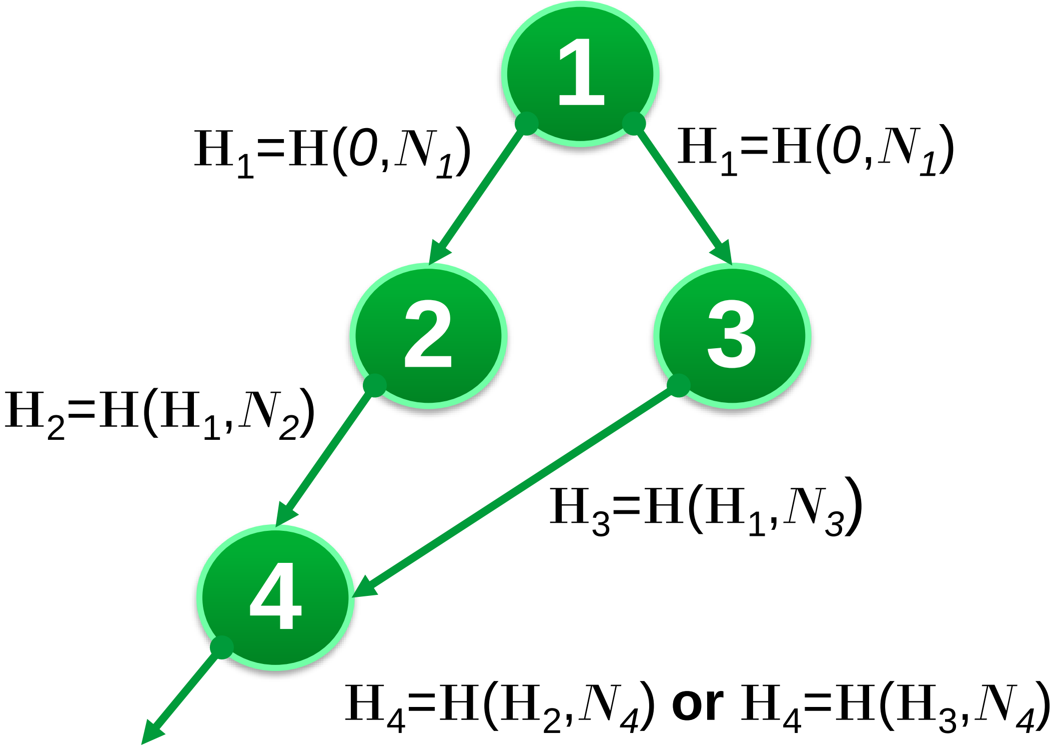
Control-Flow Attestation (C-FLAT)



T. Abera, N. Asokan, L. Davi, J-E. Ekberg, A. Paverd, T. Nyman, A-R. Sadeghi, G. Tsudik, **C-FLAT: Control Flow Attestation for Embedded Systems Software**, CCS '16. http://arxiv.org/abs/1605.07763,

C-FLAT: High-Level Idea

- Cumulative Hash Value: $H_j = H(H_i, N)$,
  where H_i previous hash result and N is the current node



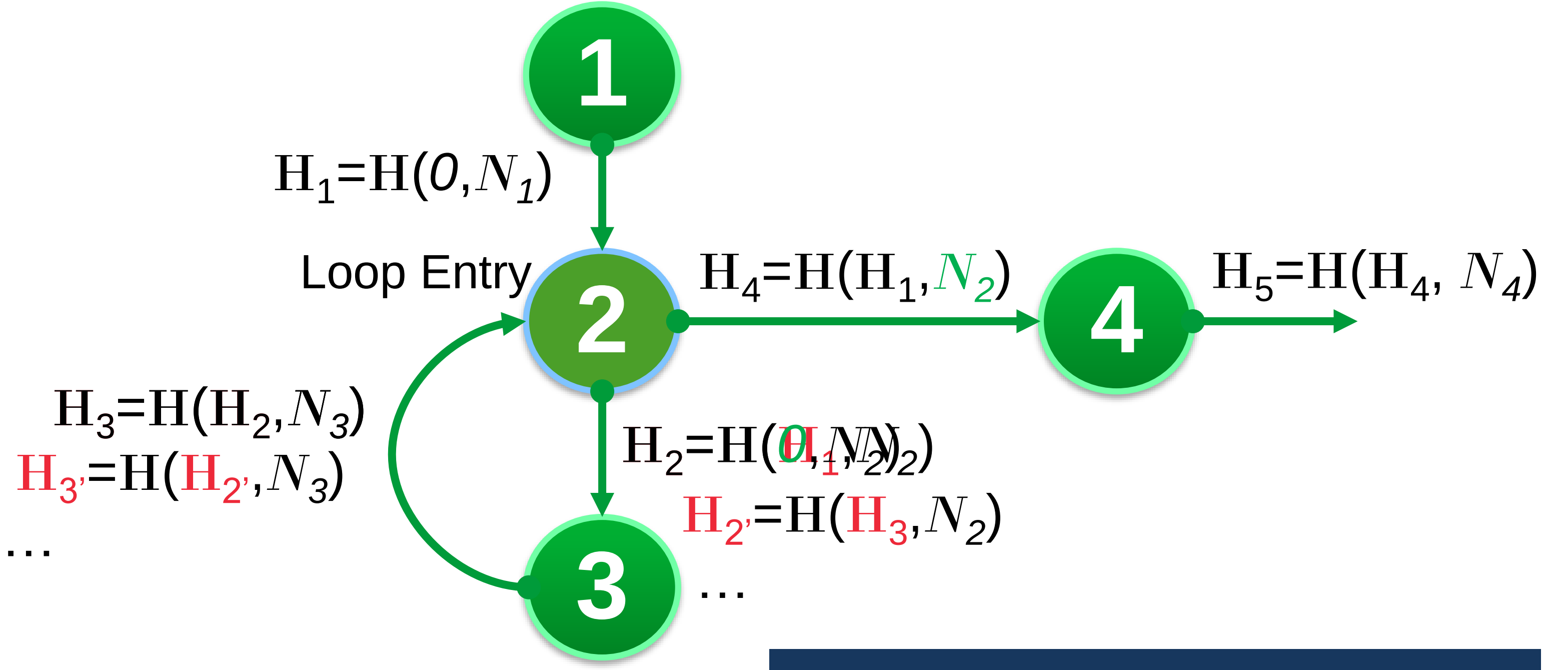
$p = H_4$

T. Abera, N. Asokan, L. Davi, J-E. Ekberg, A. Paverd, T. Nyman, A-R. Sadeghi, G. Tsudik, **C-FLAT: Control Flow Attestation for Embedded Systems Software**, CCS '16. http://arxiv.org/abs/1605.07763,

Handling Loops

H_x different for each loop iteration

- Different loop paths/iterations → many valid hash values
  - Our approach: treat loops as separate sub-graphs



$p = H_5, <H1, \{H_3, \#H_3\}>$

T. Abera, N. Asokan, L. Davi, J-E. Ekberg, A. Paverd, T. Nyman, A-R. Sadeghi, G. Tsudik, **C-FLAT: Control Flow Attestation for Embedded Systems Software**, CCS ‘16. http://arxiv.org/abs/1605.07763,

Proof-of-Concept Implementation

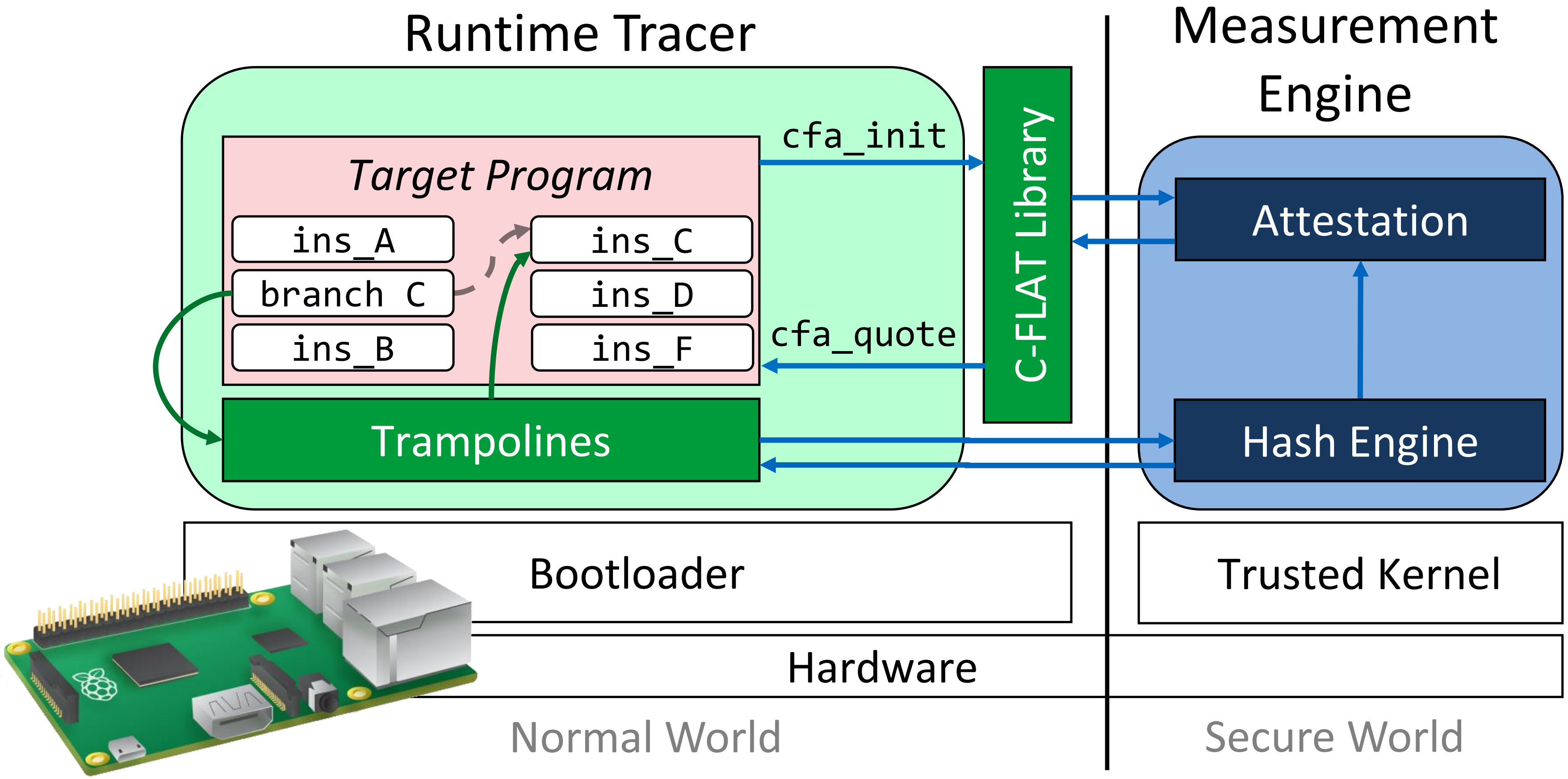
- Bare-metal prototype on Raspberry Pi 2
  - Single-purpose program instrumented using binary-rewriting
  - Runtime Monitor written in ARM assembler
  - Measurement Engine isolated in TrustZone-A Secure World

cfa_quote: 7c 16 d6 51 20 a2 a0 c7 90 f5 ef 04 0c 2e ba bc
loop[000]: 78 22 5b 62 92 41 ca 02 7b ff 29 57 c6 6f 9b a2
path[000]: 2f a5 8c dc 1b 35 41 29 ab dd 35 5c f2 69 08 37 (1)
loop[001]: d6 90 9e a0 8c ae 90 84 9e 66 09 f8 a6 7b 52 04
path[000]: 92 fb d1 e8 90 cb 02 e5 6c f2 65 8c 86 72 0e d3 (2)
….
loop[006]: 05 e3 92 40 95 ef 7b 46 13 7d 6e 8b 05 be bf 41
path[000]: 67 c6 5e d4 18 13 02 bc 4a 5d 60 a0 16 85 f4 ed (9)
path[001]: 78 19 af 09 0f d5 64 f4 39 b4 7a 0d 97 57 77 8c (2)

Source: https://github.com/control-flow-attestation/c-flat/blob/master/samples/syringe/syringe-auth.txt

T. Abera, N. Asokan, L. Davi, J-E. Ekberg, A. Paverd, T. Nyman, A-R. Sadeghi, G. Tsudik, **C-FLAT: Control Flow Attestation for Embedded Systems Software**, CCS ‘16. http://arxiv.org/abs/1605.07763,

Proof-of-Concept Implementation



T. Abera, N. Asokan, L. Davi, J-E. Ekberg, A. Paverd, T. Nyman, A-R. Sadeghi, G. Tsudik, **C-FLAT: Control Flow Attestation for Embedded Systems Software**, CCS '16. http://arxiv.org/abs/1605.07763,

Run-time attestation: summary

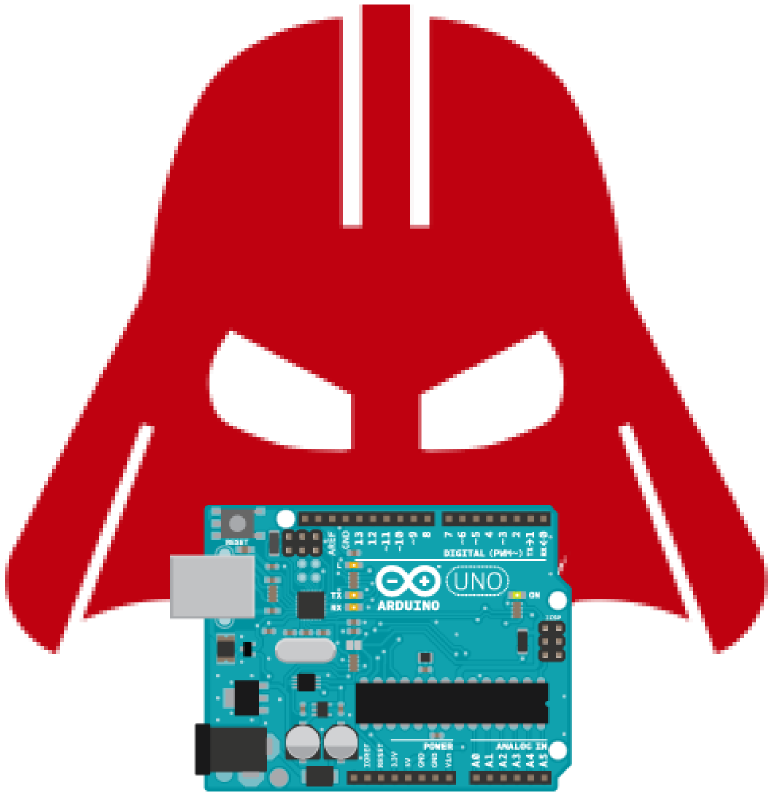
- How can we scale control flow attestation?
  - Better ways to aggregate measurements?
  - Faster/simpler purpose-built hash functions?
  - Purpose-built hardware support?

- What next?
  - Attestation of properties rather than measurements?
    - from attestation to checking compliance with a (dynamic) policy?

Ghada Dessouky, Shaza Zeitouni, Thomas Nyman, Andrew Paverd,
Lucas Davi, Patrick Koeberl, N. Asokan, Ahmad-Reza Sadeghi,
LO-FAT: Low-Overhead Control Flow Attestation in Hardware,
DAC ‘17.

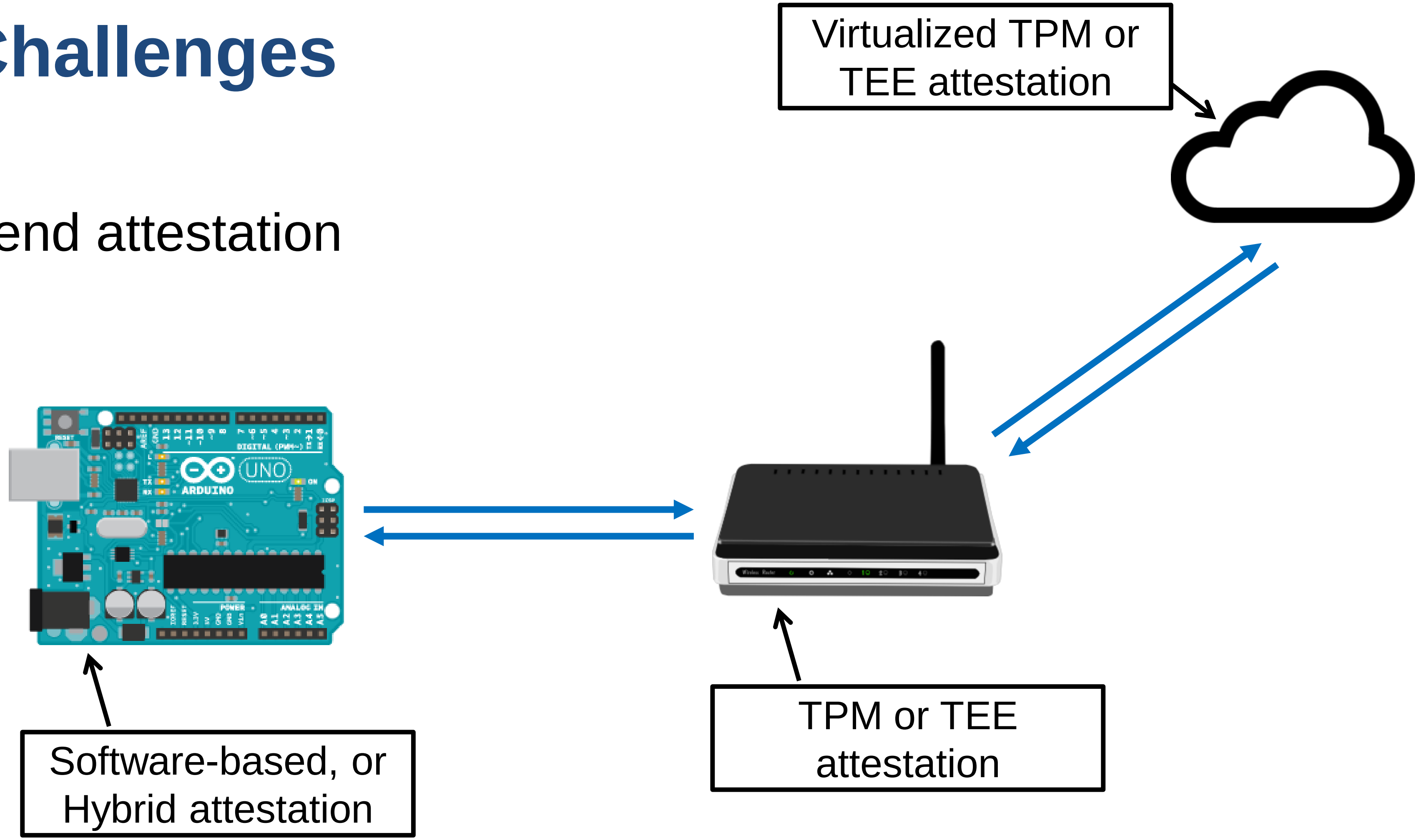
Open Challenges

- Physical adversary



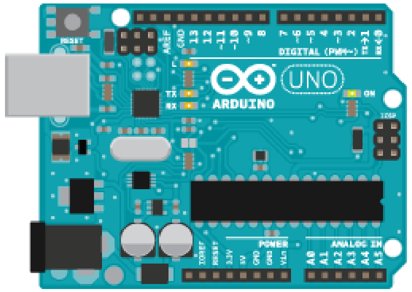
Open Challenges

- End-to-end attestation



Open Challenges

- Device heterogeneity



Did you learn about:

- Why we need attestation in IoT

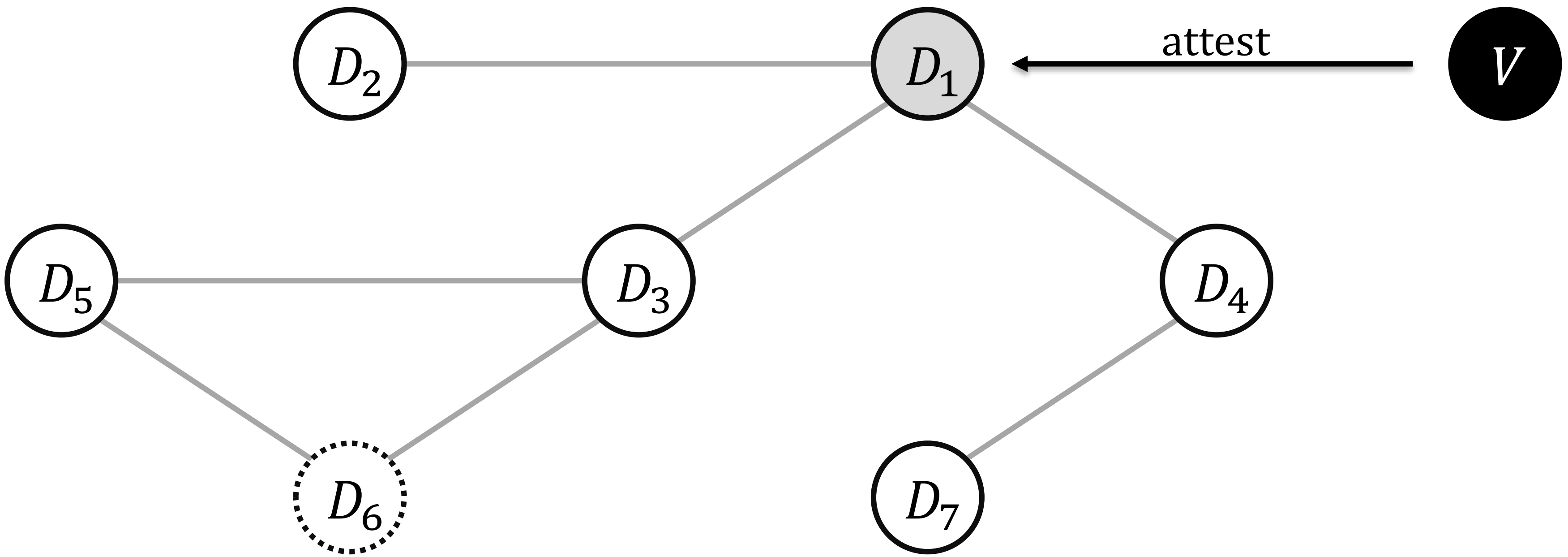
- Some recent research towards achieving this

Aalto University

Attestation in the Internet of Things (IoT)

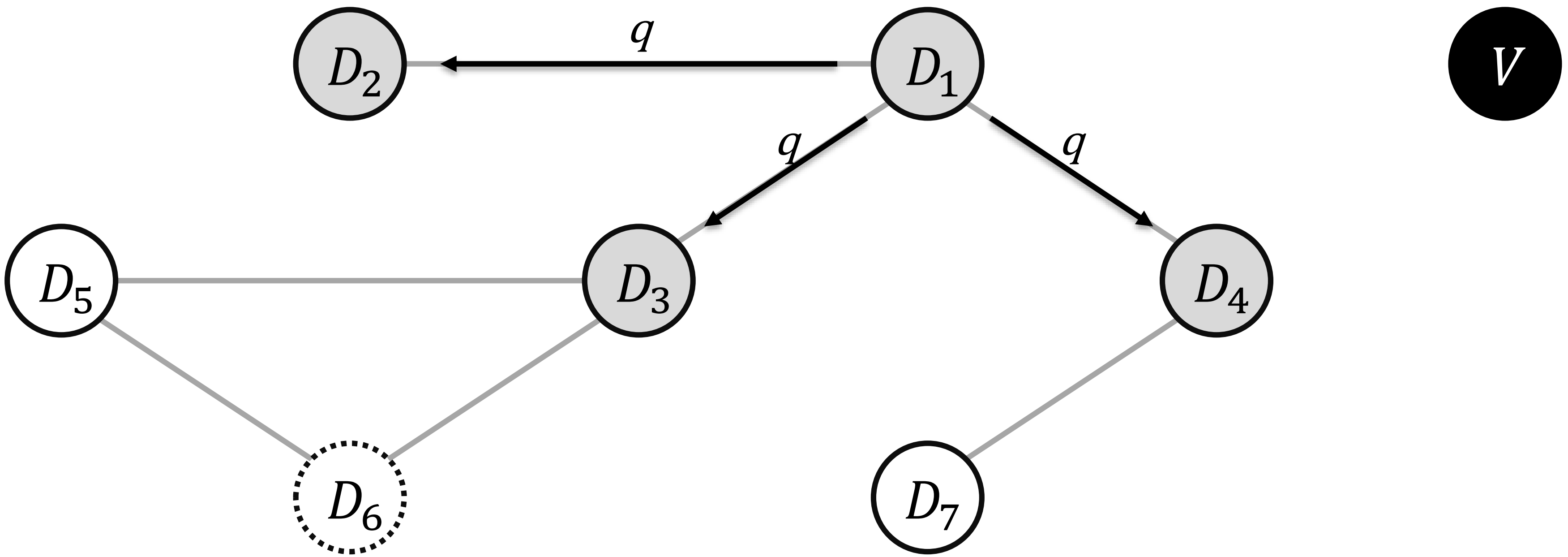
MSS – Lecture 6 (part 2)
Andrew Paverd

SEDA (in detail)



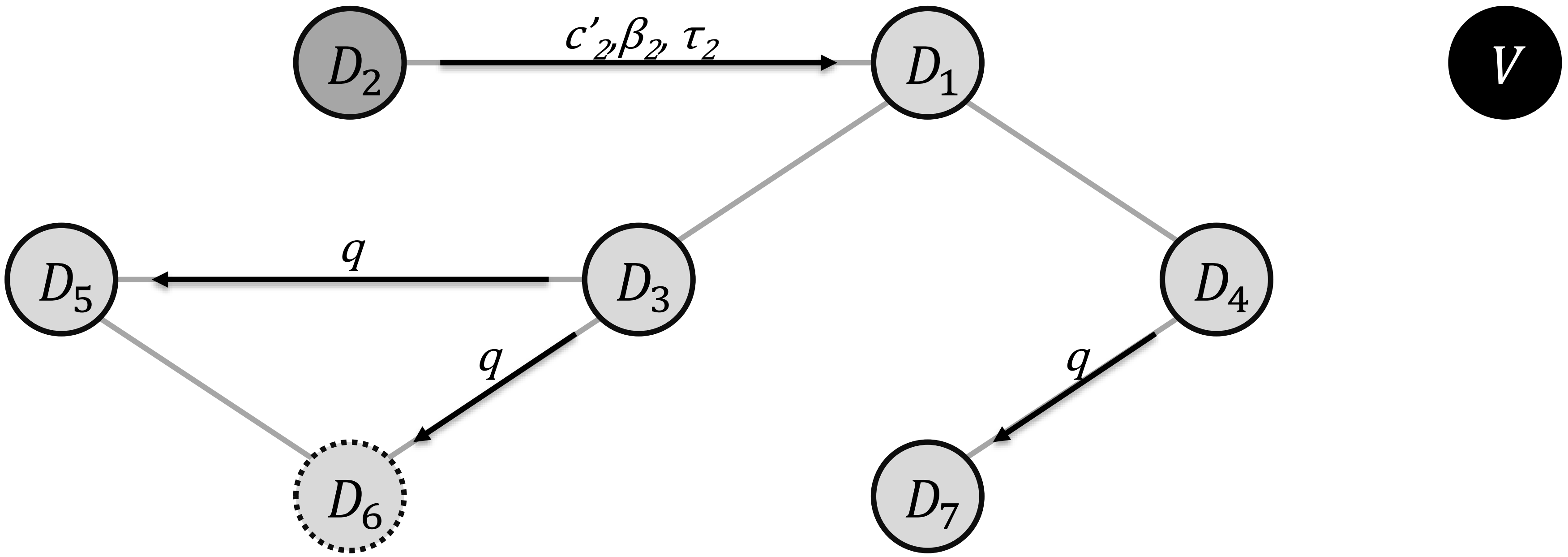
N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

SEDA (in detail)



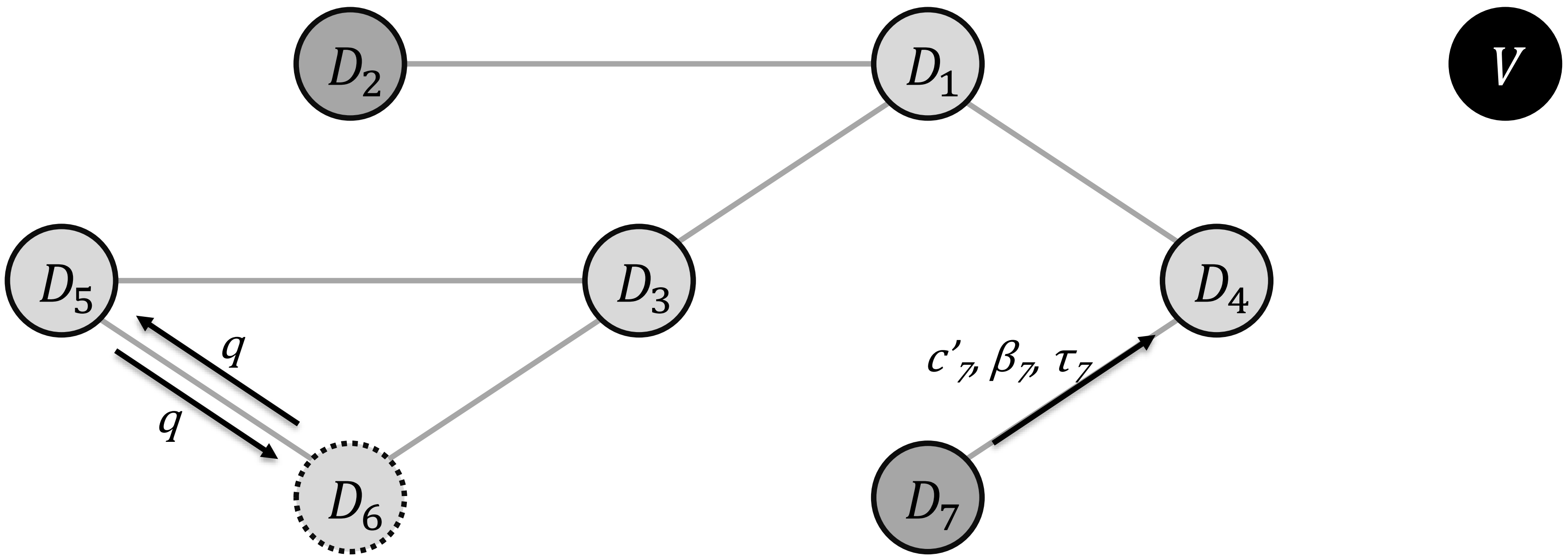
N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

SEDA (in detail)



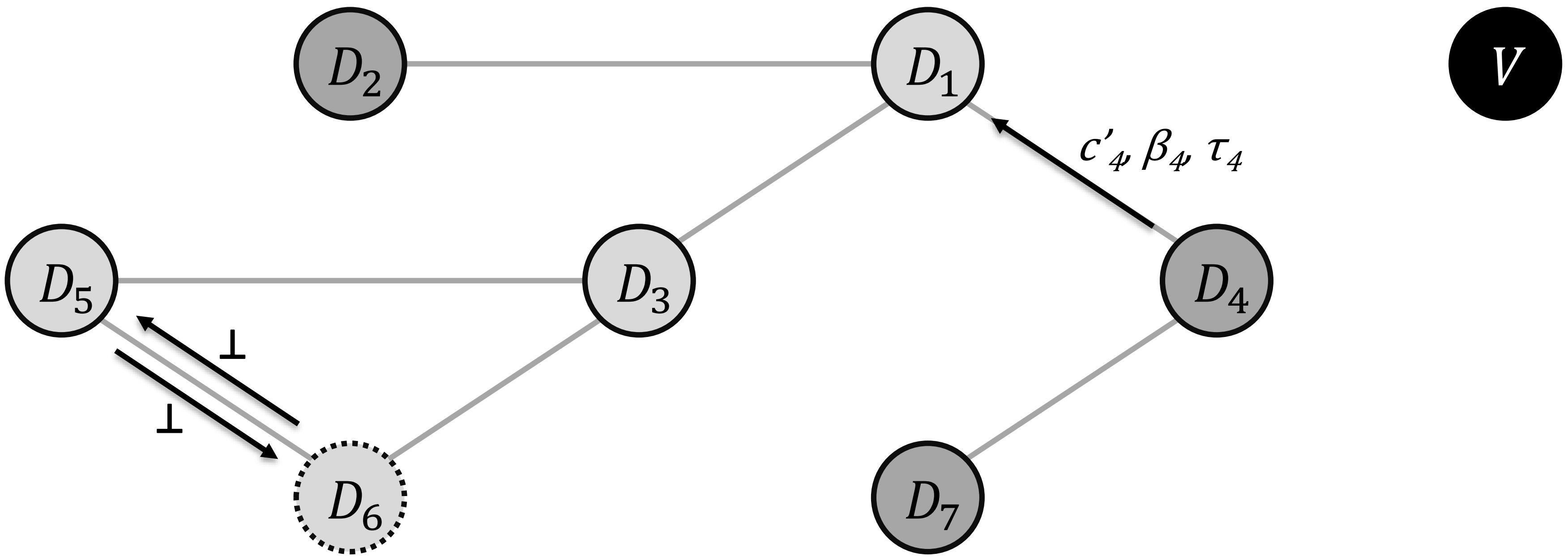
N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

SEDA (in detail)



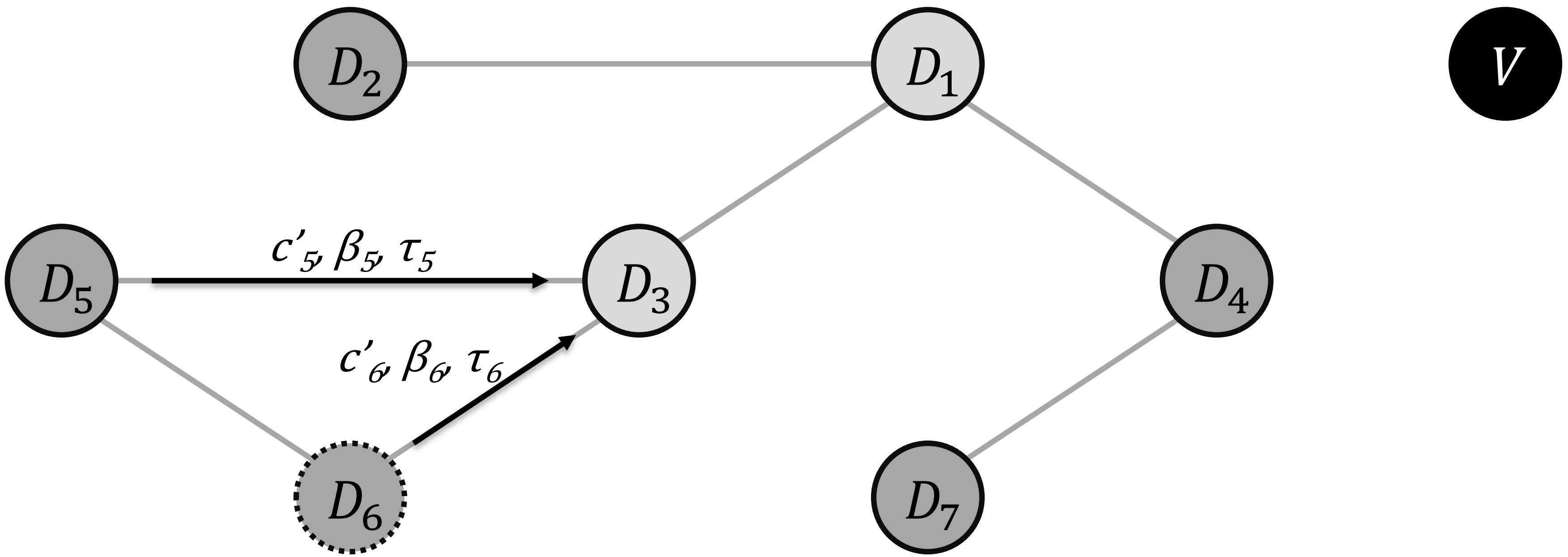
N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS ‘15

SEDA (in detail)



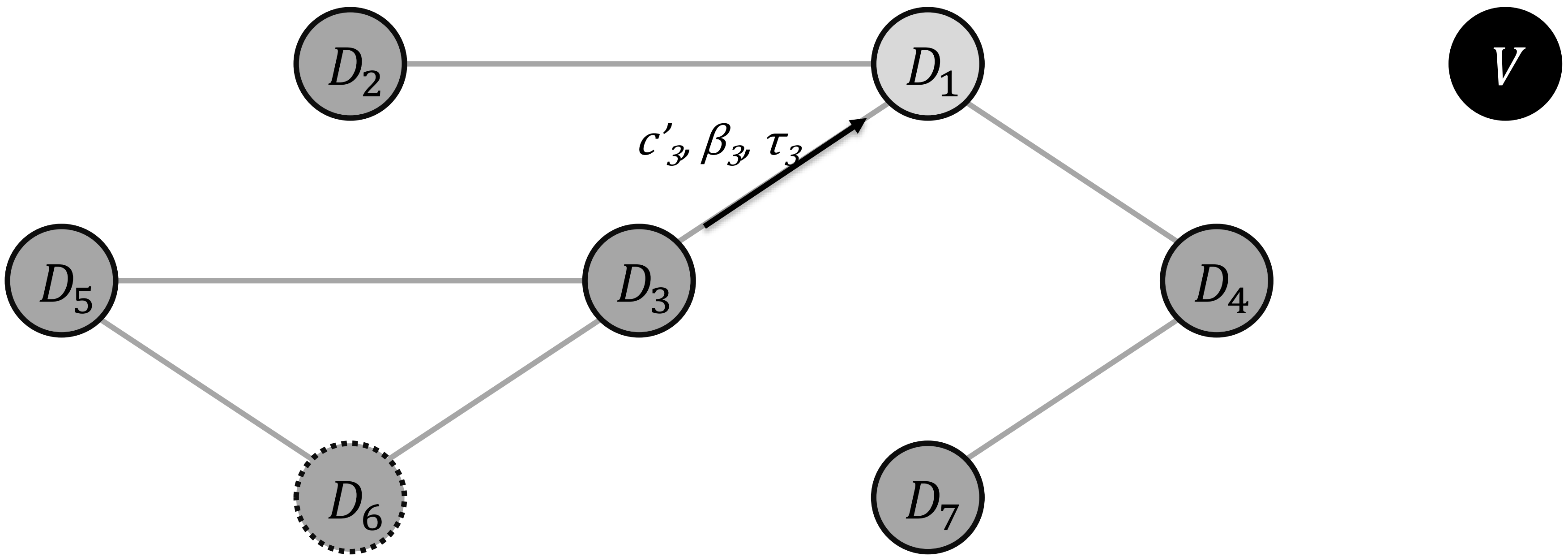
N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS ‘15

SEDA (in detail)



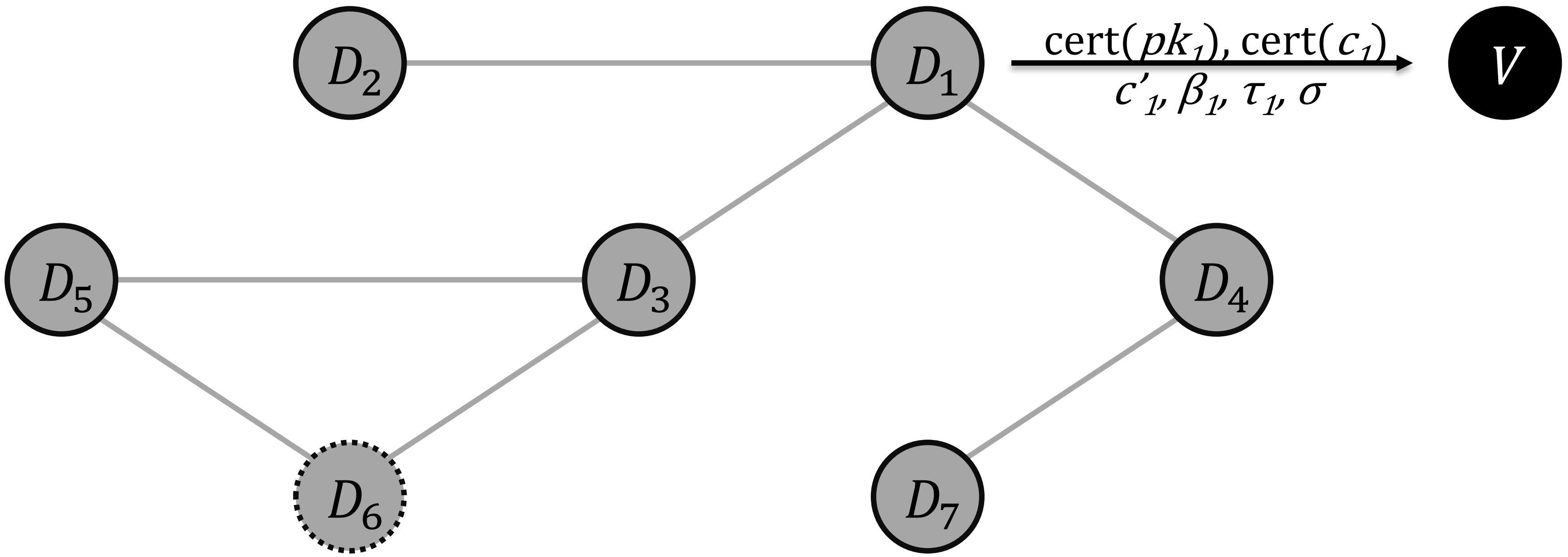
N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

SEDA (in detail)



N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

SEDA (in detail)



N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. **SEDA: Scalable Embedded Device Attestation**. CCS '15

Abuse of Attestation

- Attestation protocols usually consider a benign verifier
- But adversary could impersonate verifier to abuse attestation
- E.g., computing a MAC over all memory in a typical microcontroller could take ~ 750 ms
  - could be used for denial of service (DoS) attack

F. Brasser, A.-R Sadeghi, K. B. Rasmussen, and G. Tsudik. **Remote Attestation for Low-End Embedded Devices: the Prover's Perspective**. DAC '16

Abuse of Attestation

- Verifier must be authenticated to prover
  - but asymmetric crypto is computationally expensive

- Prover must detect replays of previous requests
  - can use nonces
  - can use counters
  - can use timestamps

Abuse of Attestation

- Nonces
  - require integrity-protected storage for previous nonces
- Counters
  - require minimal integrity-protected storage for counter
- Timestamps
  - require trusted synchronized clock at prover side

Use Execution-Aware Memory Access Control (EA-MAC) to protect counters and timers.

F. Brasser, A.-R Sadeghi, K. B. Rasmussen, and G. Tsudik. **Remote Attestation for Low-End Embedded Devices: the Prover's Perspective**. DAC '16