Remote Attestation of low-end Embedded, IoT and "Smart" devices

GENE TSUDIK
Computer Science Department
UCI

gene.tsudik@uci.edu
LAB: http://sprout.ics.uci.edu

Joint work with colleagues from:
UCI, HRL, Eurecom, TU Darmstadt, Aalto U, Intel Labs

1

Outline

- Introduction/Motivation
- Remote Attestation (simple setting)
- Attacks on Prover
- Attesting Many Provers
- Coping with Physical Attacks
- The End

2

Current Research Topics

- Privacy in Social Networks
  - Stylometric Linkability and Attribution
  - Off-Line Private Social Interactions
- Genomic Privacy and Security
- Security of Embedded Devices & Systems
- Private Database Querying
- Usable Security
- Biometrics
- S&P in Future Internet Architectures

For more info see: **sprout.ics.uci.edu**

What's an embedded device?

- Buzzwords: Embedded Systems/Devices, **IoT**, CPS, etc.
- Anything that's not a general-purpose computer

4

Widening Range of Specialized Embedded Devices

Already here or coming soon…

- **Smart watches, e.g., Samsung, Apple**
- **Smart eye-wear, e.g., Google Glass**
- **Smart toys**
- **Smart pills**
- **Smart footwear**
- **Smart clothes**

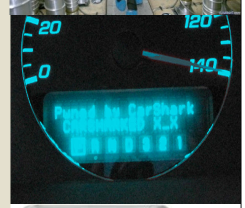
6

Why?

- **Default PINs or passwords**
- **Wide-open communication**
- **Buggy software**
- **No (or inadequate) hardware protection**
- **Limited “real estate”, limited budgets**

- **HW/FW/SW trojans (aka malware)**
- **Attacks aim to:**
  - **Snoop, exfiltrate**
  - **Cause physical damage**

Notable Attacks

- Stuxnet [1] (also DUQU)
  - Infected controlling windows machines
  - Changed parameters of the PLC (*programmable logic controller)* used in centrifuges of Iranian nuclear reactors
- Attacks against automotive controllers [2]
  - Internal controller-area network (CAN)
  - Exploitation of one subsystem (e.g., bluetooth) allows access to critical subsystems (e.g., braking)
- Medical devices
  - Insulin pump hack [3]
  - Implantable cardiac defibrillator [4]

[1] W32.Stuxnet Dossier, Symantec 2011
[2] Comprehensive Experimental Analyses of Automotive Attack Surfaces, USENIX 2011
[3] Hacking Medical Devices for Fun and Insulin: Breaking the Human SCADA System, Blackhat 2011
[4] Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses, S&P 2008

Adversarial & Attack Flavors

- **Remote**
  - Goal: infect device with malware
  - Malware propagates from the outside, perhaps slowly (e.g., jumps air-gaps)
- **Local**
  - Goal: impersonate and/or clone device, collect information
  - Eavesdrops on -- and/or controls -- communication to/from device
- **Physical Non-intrusive**
  - Goal: Learn device secrets, impersonate and/or clone
  - Located near device
  - Side-channel attacks
- **Physical Intrusive**
  - Goal: clone and/or manually infect device
  - Captures device and physically extract secrets
  - Stealthy or not?
- **Some hybrids of the above…**

9

What can we do?

- Prevention or detection?
- Protect devices individually or in bulk?

10

Outline

- **Introduction/Motivation**
- **Remote Attestation (simple setting)**
- **Attacks on Prover**
- **Attesting Many Provers**
- **Coping with Physical Attacks**
- **The End**

11

Detection necessitates Remote Attestation

What is Remote Attestation?

- 2-party security protocol between trusted Verifier and untrusted Prover
- A service that allows the former to verify **internal state** of the latter

Where:

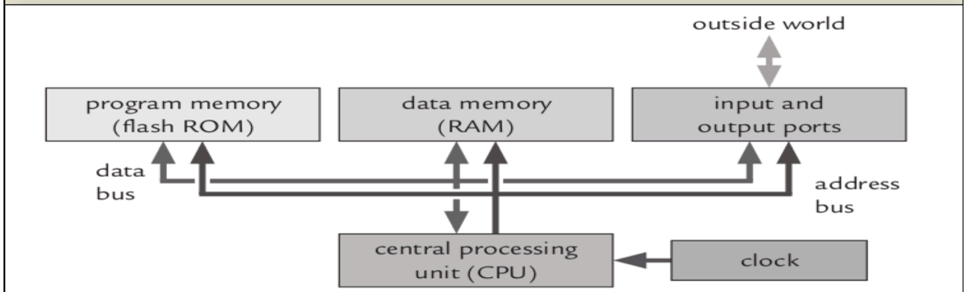
- Prover – **untrusted** (possibly compromised/infected) device
- Verifier – **trusted** reader/controller/base-station (not always present)
- **Internal state** of Prover includes:
  - Code, Registers, Data Memory (RAM), I/O, etc.

Adversary:

- Can compromise Prover at will (remote)
- Can control communication channels (local)
- **Physical attacks** usually considered **out of scope**
  - Will re-visit this…

12

Low-End Embedded Devices are
Amoebas of the Computing World

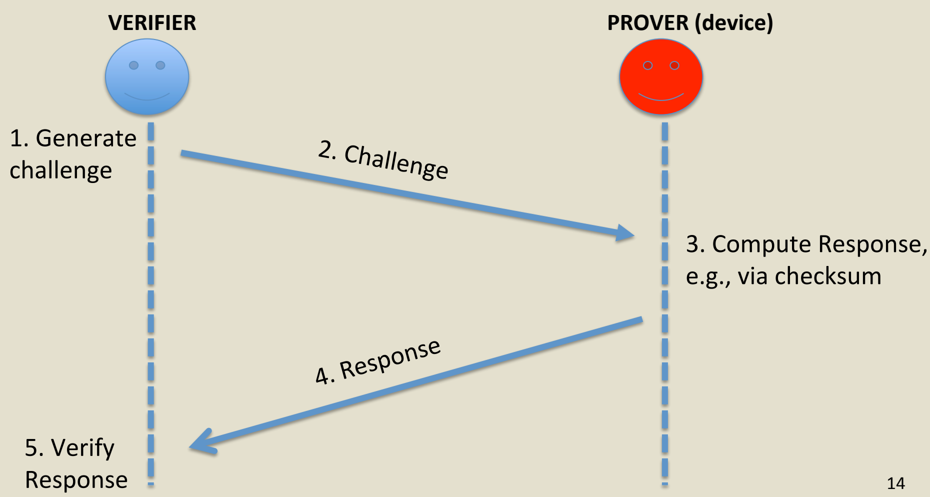


- Memory: program and data
- CPU, Integrated clock
- As well as:
  - Communication interfaces (USB, CAN, Serial, Ethernet, etc.)
  - Analog to digital converters
- Examples: TI MSP430, Atmel AVR, Raspberry Pi

13

Remote Attestation

- If Prover is infected, resident malware lies about software state
- Need to have guarantees that Prover is “telling the truth”



14

Remote Attestation

Prior work:

- Very popular topic
- Can bootstrap other services
  - e.g., code update, secure erasure
- Many publications and deployed systems

- Secure Hardware-based
  - Uses OTS TPM components

- Software-based (aka time-based)
  - Uses custom checksums

- Hybrid (sw/hw co-design)

15

Software Attestation

- Prover has no architectural support for security
  - Commodity/legacy device
  - Peripheral, e.g., adapter, camera, keyboard, mouse
- Verifier sends customized (random-seeded) checksum routine which covers memory in a unique (unpredictable) pattern
- Prover runs checksum over memory, returns result
- Verifier uses precise timing to determine presence/absence of malware
- Main idea: malware has nowhere to hide, no place to go…
  - Even if it does manage to hide itself physically, delay will be noticed

For this to work, need 3 assumptions:

1. Verifier←→Prover round-trip time must be either **negligible** or **constant**
   - Meaning: one-hop communication
2. Checksum code must be minimal in both time and space
   - How to prove that?
3. Prover must not have outside help
   - No extraneous communication during attestation (aka "adversarial silence")

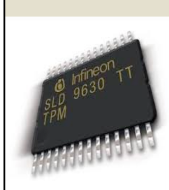
16

SW Attestation

- Some prominent SW attestation techniques have been attacked
  - None provide concrete proofs or guarantees
- Still, the only choice for legacy devices, e.g., peripherals

HW-based Attestation

- Prover has architectural support for attestation, usually using a TPM
- TPM is essentially a tamper-evident or tamper-resistant “alien”
- A heavy-weight approach, not suitable for low-end devices
  - Due to: cost, size, energy, etc.
- Overkill: not clear what features are really needed for attestation



Hybrid Attestation

Main Idea: systematically derive/identify exact features/components necessary for remote attestation under a given adversarial model

19

SMART: Secure & Minimal Architecture for Remote Trust (NDSS 2012)

Motivation:

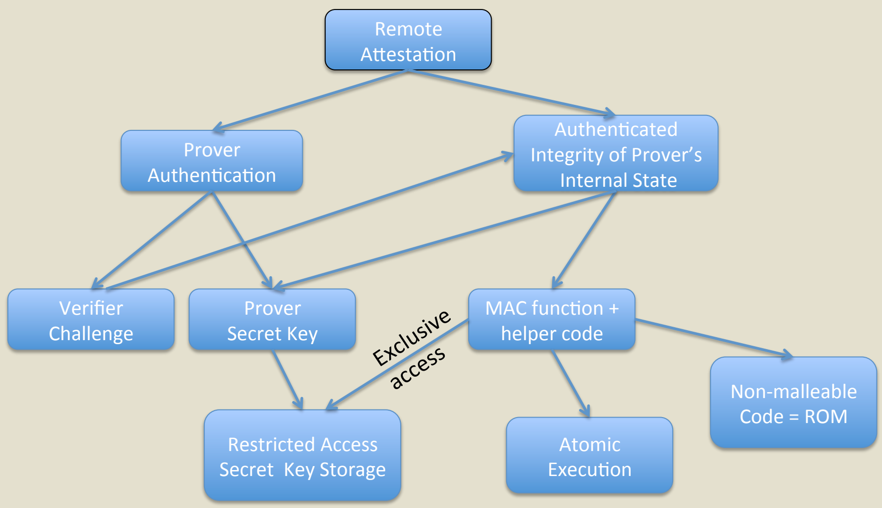
- **Secure Hardware** techniques too costly for low-end devices
- **Software** attestation not applicable for **remote** settings
- What is the minimal set of architectural (sw & hw) features needed to achieve provably secure remote attestation?

Desired properties:

- Minimal modifications to current platforms
  - Lowest # of additional gates
- Security under a strong attacker model
- Applicability to low-end MCU platforms
- No physical attacks (for now)

20

Deriving Features needed for Remote Attestation



Building Blocks

1. **Secure Key Storage** (as little as 180 bits)
   - Mandatory for remote Prover
   - Enables Prover authentication
2. **Trusted ROM code** memory region
   - Read-only means integrity: computes response
   - Accesses/uses key (exclusively)
3. **MCU access control**
   - Grants access to key **only** from within ROM
4. **MCU-enforced atomicity of ROM code execution**
   - Atomically disable/enable interrupts on entry/exit
   - No invocation other than from the start
   - No termination other than from the end

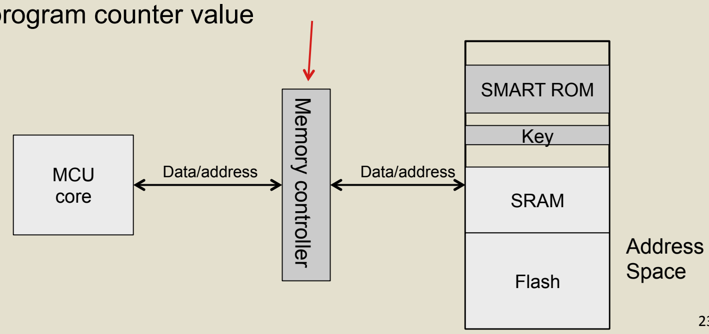
22

Key Storage & Memory Access Control

- Key facilitates Prover authentication
- Can't be stored in regular memory
  - Or malware would steal it
- Need to restrict access

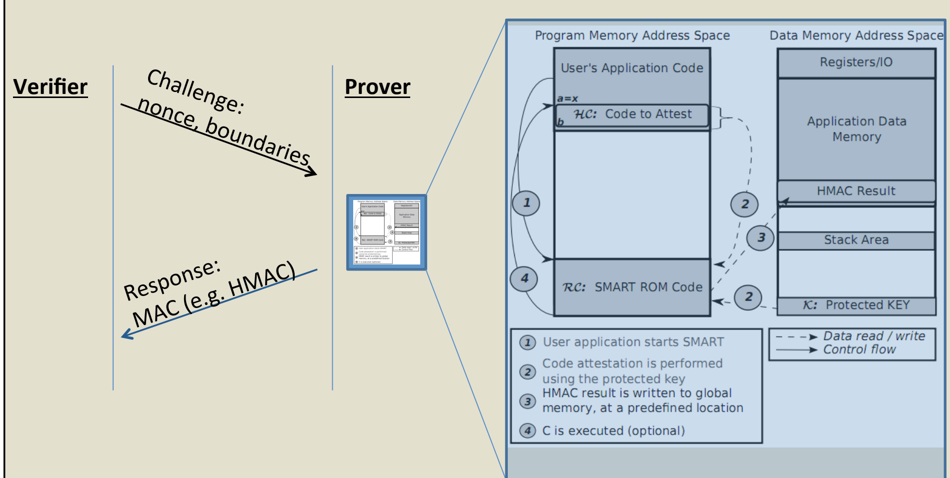
Our approach

- Restrict key access to trusted ROM
- Control program counter value



23

The complete protocol



24

Issues…

If Prover is infected, ROM code and malware share the same MCU resources

- Malware can set up execution environment to compromise ROM code and extract key
- Malware can schedule interrupts to occur asynchronously while key (or some function thereof) is in main memory
- Malware can use code gadgets in ROM to access key
  - Return-Oriented Programming (ROP)
- ROM code might leave traces of key in memory after its execution

25

Countermeasures

- Atomic ROM code execution: enforced in hardware
  - Enter at first instruction
  - Exit at last instruction
  - If **IP** points to ROM, previous instruction must be in ROM*
- ROM code instrumented to check for memory safety
  - Used DEPUTEE package
  - Upon detecting error, reboot and clean up memory
- Interrupts disabled immediately upon ROM entry
  - Before key usage (enabled upon exit)
  - DINT instruction must itself be <u>atomic</u>
- Erase key-related material before end of execution

26

Costs of ROM and Access Control

Prototyped on commodity low-end MCU platforms

Component		Original Size in kGE	Changed Size in kGE	Ratio
AVR MCU		103	113	10%
Core		11.3	11.6	2.6%
Sram	4 kB	26,6	26.6	0%
Flash	32 kB	65	65	0%
ROM	6 kB	-	10.3	-
MSP430 MCU		128	141	10%
Core		7.6	8.3	9.2%
Sram	10 kB	55.4	55.4	0%
Flash	32 kB	65	65	0%
ROM	4 kB	-	12.7	-

27

Outline

- **Introduction/Motivation**
- **Remote Attestation (simple setting)**
- **Attacks on Prover**
- **Attesting Many Provers**
- **Coping with Physical Attacks**
- **The End**

28

SMART follow-ons

- TrustLite (EuroSys'12): multiple application processes
- TyTan (DAC'15): real-time applications, secure interrupts

29

Verifier Impersonation + DoS Attacks

- ❑ So far, assumed that Verifier is honest while Prover is possibly compromised
  - ❑ What if Prover is honest but Verifier is not?
  - ❑ What if Adversary's goal is DoS of Prover?
  - ❑ Attestation can be quite resource-consuming
  - ❑ Attestation takes Prover away from its real and maybe critical job (e.g., fire, CO2 or vibration detection)
  - ❑ What if attestation is combined with erasure and/or sw update?
  - ❑ Verifier Impersonation and DoS are easier than compromising Prover
- ❑ Prior remote attestation techniques don't address this
- ❑ How to do this with minimal overhead & minimal additional features?

30

Challenges

- Software attestation can't do this at all
  - No secure place to store any secret or public key
- With secure hardware, this is easy
  - A ``Cadillac'' solution, unsuitable for low-end MCUs

31

Verifier Authentication

- Prover's attestation request must be authenticated
  - Authentication is, in itself, a form of DoS
- Authentication requires:
  - Key(s)
  - Freshness & Timeliness

32

In more detail:

- Key(s)
  - Shared secret key (in access-restricted ROM)
  - Verifier public key (in ROM)
  - Access-restricted PUF

- Freshness
  - Challenges
  - Sequence numbers
  - Timestamps

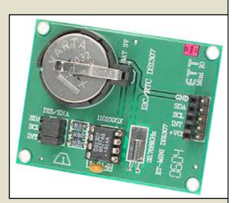
- Well-known pros and cons

33

Bottom-line:

NEED:

- Reliable Read-Only Clock
  - Time-check incoming requests
  - Timeliness and freshness
  - Battery, clock crystal, etc.



OR

- Secure Writeable Memory
  - Check for monotonically increasing timestamps
  - Freshness only

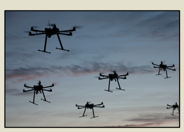
34

Outline

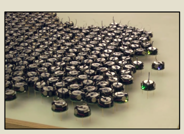
- Introduction/Motivation
- Remote Attestation (simple setting)
- Attacks on Prover
- Attesting Many Provers (swarms/networks)
- Coping with Physical Attacks
- The End

35

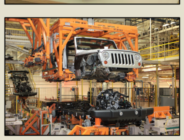
Attesting Groups of Embedded Devices



Drones
Video surveillance, environment monitoring



Robot swarms
Prospecting, rescue, etc.



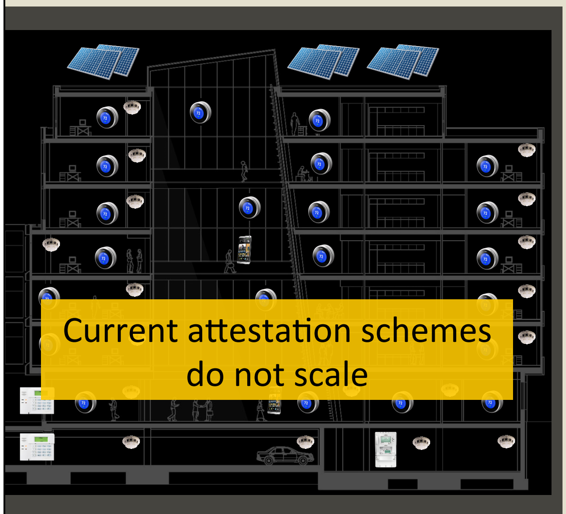
Smart factories and buildings (home/office)
Collaborating CPS

Transportation
Automotive, marine, avionic systems

All are subject to attacks..

36

EXAMPLE: Smart Building



Multiple CPS:

HVAC
Heating, temperature sensors, …

Fire alarm
Smoke detectors, sprinklers, …

Access control
Card readers, burglar alarm, …

Energy management
Smart meters, solar panels, …

Decentralized control
Smartphones, tablets, …

Many (possibly wirelessly) connected CPS sharing resources
e.g., CPS monitoring windows used by air conditioning and alarm system

37

SEDA: **S**calable **E**mbedded **D**evice **A**ttestation (ACM CCS'15)

- **Cumulative**

More efficient than attesting each single device

- **Scalable**

Supports integrity verification of large device groups

- **Decentralized**

Distributes (not evenly) load and energy consumption over all devices

- **Flexible**

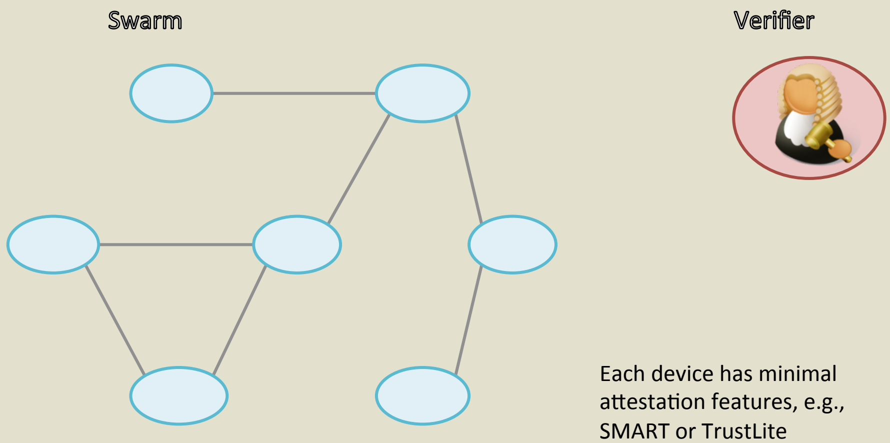
Independent of integrity measurement mechanism used by devices

- **Applicable to low-end MCU-s**

Implementation based on SMART and TrustLite security architectures

38

System Model

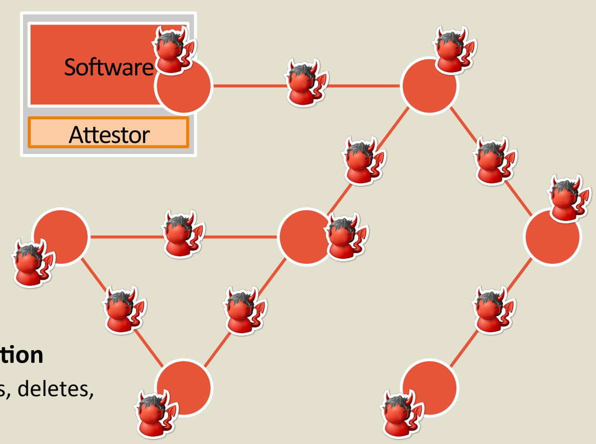


Each device has minimal attestation features, e.g., SMART or TrustLite

Devices in swarm may have different hardware and software configurations

Each device can communicate only to its neighbors

Adversary Model and Assumptions



Controls Communication

- Eavesdrops, modifies, deletes, inserts messages

Remotely Compromises Multiple Devices

- Any set of (even all) devices

But, no physical attacks

Scalable Embedded Device Attestation (SEDA)

Device Initialization
- Prepares devices to be deployed
- Executed by swarm operator $\mathcal{O}$ in a trusted environment

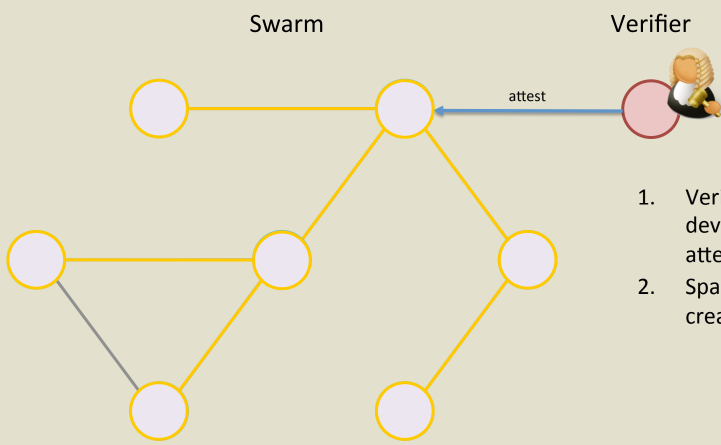
Device Join (join)
- Run when new device is added to a swarm
- Uses public key crypto (to avoid need for pre-established shared keys)

Swarm Attestation (attest)
- Between verifier and one device
- Uses public key crypto (to avoid need for pre-established shared keys)

Device Attestation (attdev)
- Between devices
- Uses only symmetric crypto for high performance

41

Attestation (1)

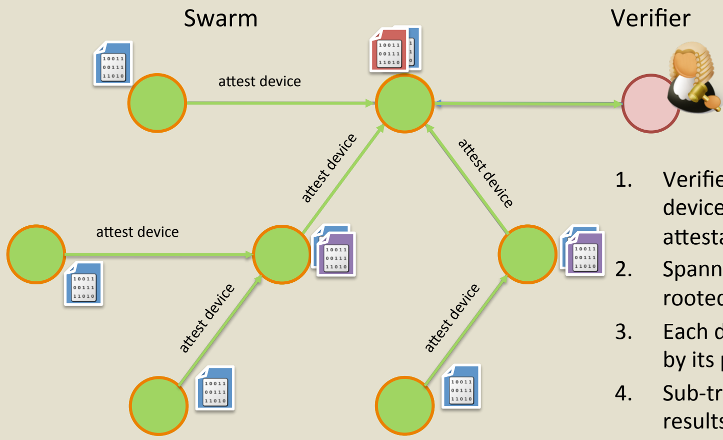


1. Verifier selects random device (D_1) initializes attestation
2. Spanning tree is created, rooted at D_1

Spanning Tree
Authentic channel

42

Attestation (2)



1. Verifier selects random device (D_1) initializes attestation
2. Spanning tree is created rooted at D_1
3. Each device gets attested by its parent (leaves first)
4. Sub-tree roots accumulate results and report to their parent
5. D_1 reports overall result to verifier

Cumulative attestation
Spanning Tree
Authentic channel
Device attestation report
Accumulated attestation report

43

Outline

- **Introduction/Motivation**
- **Remote Attestation (simple setting)**
- **Attacks on Prover**
- **Attesting Many Provers**
- **Coping with Physical Attacks**
- **The End**

44

DARPA: **D**evice **A**ttestation **R**esilient to **P**hysical **A**ttacks (current work)

- Physical Attacks are difficult to mitigate in a single-Prover setting
- More realistic in group context
  - many distributed Provers
  - some devices might be subject to capture and physical attack
- How to mitigate device capture?
- Observation: capture ➔ absence

45

DARPA: **D**evice **A**ttestation **R**esilient to **P**hysical **A**ttacks

- Main idea: each interval of duration **T**, devices periodically collectively emit and collect each other's authentic "heartbeats"
- Heartbeats are periodically collected by verifier
- Missing heartbeat ➔ absent device ➔ possible capture
- Minimal Requirements:
  - SMART architecture
  - Reliable Read-Only Clock
- Can tolerate VERY powerful adversary:
  - REMOTE (can infect all devices with malware)

  \+
  - PHYSICAL (can capture/attack all devices but one)

46

Current Topics/Directions

- Single Prover/Verifier Setting
  - Verifier Authentication, DoS Mitigation
  - Formal proofs and analyses
  - Customization: code update, secure erasure, secure boot
  - Experiments and implementation
- Groups/Swarms of devices (multiple Provers)
  - Efficient collective attestation techniques
  - Heterogeneous devices and variable attestation support
  - Physical Attack (Capture) mitigation

47

Some references

F. Brasser, et al.
Remote Attestaion: the Prover's Perspective
IEEE/ACM DAC 2016.

A. Ibrahim, et al.
DARPA: Device Attestation Resilient to Physical Attacks,
ACM WISEC 2016.

T. Abera, et al.,
C-FLAT: Control-FLow ATtestation for Embedded Systems Software,
ACM CCS 2016.

N. Asokan, et al.,
SEDA: Scalable Embedded Device Attestation,
ACM CCS 2015.

K. El Defrawy, et al.,
Remote Attestation of Heterogeneous Cyber-Physical Systems: The Automotive Use Case,
ESCAR 2015.

A. Francillon, et al.,
A Minimalist Approach to Remote Attestation,
ACM/IEEE DATE 2014.

P. Koeberl, et al.
TrustLite: a security architecture for tiny embedded devices
EUROSYS 2014.

K. Eldefrawy, et al.,
SMART: Secure and Minimal Architecture for Establishing Dynamic Root of Trust,
NDSS 2012.

D. Perito and G. Tsudik,
Secure Code Update for Embedded Devices via Proofs of Secure Erasure,
ESORICS 2010.

48


Thanks!