Towards Systematic Design of Collective Remote Attestation Protocols

Ivan De Oliveira Nunes¹, Ghada Dessouky², Ahmad Ibrahim², Norrathep (Oak) Rattanavipanon¹, Ahmad-Reza Sadeghi², and Gene Tsudik¹

¹UC Irvine, ²TU Darmstadt
Internet-of-Things (IoT) proliferation
Groups/swarms of IoT Devices

Drones

Smart Factories

Robot Swarms

Transportation
Groups/swarms of IoT Devices

Hacked drones an increasing concern as government seeks to ease restrictions

IoT insecurity is opening the door for deadly-accurate AI-powered swarmbot attacks

Cyber-attacks on smart factories are on the rise

Auto Industry could lose $24 billion to Cyber Attacks
Remote Attestation (RA)

**Single-Device Remote Attestation**

- Two-party challenge-response protocol between:
  - **Verifier** is trusted
  - **Prover** is untrusted

- Verification of *internal state* of a prover by a verifier
Remote Attestation (RA)

Single-Device Remote Attestation

• Two-party challenge-response protocol between:
  • Verifier is trusted
  • Prover is untrusted

• Verification of internal state of a prover by a verifier

(2) Usually realized as a Message Authentication Code (MAC) or PK signature over Prover’s memory
Collective Remote Attestation (cRA)

• Efficiently scale *single-device* RA to *many devices*...

• While ensuring *security* of all devices

• Verifier measures *internal state of multiple remote Provers*
cRA protocols

- **SEDA [CCS15]**
  - Static topology
  - Remote and network attacks

- **SANA [CCS16]**
  - SEDA follow-on, faster aggregation scheme

- **LISAs [AsiaCCS17]**
  - Minimal implementation, bandwidth and computation complexity.

- **DARPA [WiSec16] and SCAPI [WiSec17]**
  - Resilience to physical attacks

- **SALAD [AsiaCCS18]**
  - Highly dynamic device topologies
What is missing?

All of cRA schemes designed in an ad-hoc fashion

• Lack of formal definitions for assumptions

• No concrete security guarantees

• Hard to determine use-case applicable for a given protocol
Overview

• Formal definition of generic cRA
• Formal model for cRA in a specific use-case
• Protocol design
• Proof
• Implementation and evaluation
• Extension to other use-cases
Overview

• Formal definition of generic cRA
• Formal model for cRA in a specific use-case
• Protocol design
• Proof
• Implementation and evaluation
• Extension to other use-cases

Our contribution:
Methodology for systematically designing cRA protocols
Design Space for cRA

• Network configuration and topology
• Device homogeneity
• Adversary model
• Device hardware architecture
• Network redundancy
• Quality of attestation (QoA): binary, number or list
• Efficiency, e.g., CPU, memory or energy
• And much more…
Design Space for cRA

- Network configuration and topology
- Device homogeneity
- Adversary model
- Device hardware architecture
- Network redundancy
- Quality of attestation: binary, number or list
- Efficiency, e.g., CPU, memory or energy
- And much more...

**Single formal model for cRA cannot account for all possibilities**
Example: Timely Collective Attestation (TCA)

TCA Assumptions

- Low-end devices
- HW (device) homogeneity
- Static network topology
- Reliable network

TCA Goals

- Synchronous attestation: attest all devices at (roughly) the same time
- Binary QoA: 0 or 1 as end-result
- Efficiency: (1) runtime: log in # devices, (2) bandwidth: linear
- Mitigation of remote and network adversaries
Modeling TCA

• Convert informal assumptions and goals to formal definitions

• Formalize assumptions require machine and network model
  • TCA Machine Model captures “low-end devices”
  • TCA Network Model captures “HW homogeneity, static topology and reliable network”
TCA Machine Model

• Random Access Machines (RAM): [CPU, M]

• Memory (M)
  • DMEM: data memory
  • PMEM: program memory
  • ROM: read-only memory
  • ProMEM: protected memory based on access control policies

• 3 types of CPU instructions
  • Read: modify registers based on M’s values
  • Write: modify M based on registers’ values
  • Execute: modify registers based on registers’ values
TCA Network Model

• **Device homogeneity**
  - $\forall m_i \in S: \{m_i \equiv \text{RAM}[\text{CPU}, M]\}$

• **Device degree**
  - $\forall m_i \in S: degree(m_i) > 0$

• **Transmission rate**
  - $\forall m_i \in S: \{\exists h \in [1, N - 1]: \{sPath(m_i, Vrf) = h\}\}$
  - $\forall m_i, m_j \in S: \{sPath(m_i, m_j) = 1 \rightarrow rate(m_i, m_j) = \mu\}$

• **Transmission delay**
  - $delay(m_i, m_j, d) = sPath(m_i, m_j) \times \frac{d}{\mu}$
Formalizing TCA goals

**TCA-Efficiency**
- Network util: linear in # devices
- Runtime: log in # devices

**TCA-Soundness**
- If protocol executed successfully...
- ... then verification should succeed

---

**Definition 2 (TCA-Efficiency).** A scheme is **TCA-efficient** iff:

1. \( \forall m_i \in S : \text{degree}(m_i) = O(1) \)
2. \( U_{CA} = O[N \times l] \)
3. \( T_{CA} = O[\log(N) \times (l/\mu + T_{agg}) + T_{att}] \)

where \( T_{att} \) is the execution time of **attest** on any \( m_i \in S \), \( T_{agg} \) is the elapsed time for \( m_i \) to aggregate \( h_i \) tokens according to **report** specification, \( N \) is the total number of members in \( S \), and \( l \) is the security parameter.

**Definition 3 (TCA-Soundness).** Let \( H_S \) be a cRA report received by Vrf in response to the execution of **request**. A cRA scheme is **TCA-Sound** iff:

\[
Pr[\text{verify}(H_S, VS) = 0 | \neg \text{Adv}] < \text{negl}(l)
\]

where \( \text{negl} \) is a negligible function, and \( l \) is the security parameter.
TCA-Security

Informal Goals

• Binary Quality of Attestation
• Synchronous attestation
• Remote and network adversaries

Formal Definitions

• Security game modelling adversarial capabilities

Definition 4 (TCA-Security).
TCA-Security-Game.
Assumptions:
- A2: \( N \) – number of devices in \( S \), and \( N = \text{poly}(l) \).
- A3: chal – a point in time selected by the challenger.
- A4: VS is public.
- A5: setup, request, attest, report, and verify are public.

Challenger plays the following game with Adv:
1) **Challenge:** Challenger presents chal to Adv.
2) **Queries:** Adv can modify \( M(S) \) at will, at any time, and make poly(l) oracle calls to attest\(^{m_i} \) for all \( m_i \in S \).
3) **Response:** Adv responds with \( H_{Adv} \) and wins iff verify\((H_{Adv}, VS) = 1 \) and \( \text{PMEM}(S, t = \text{chal}) \neq VS \).

Security Definition.
A cRA protocol is TCA-Secure if there is no ppt Adv capable of winning TCA-Security-Game with probability \( \Pr[\text{Adv, TCA-Security-Game}] > \text{negl}(l) \).
Synchronous Attestation Protocol (SAP)

Simple and Provable TCA Scheme
SAP: setup

Verifier

D1
D2
D3
D4
D5
D6
D7
1. Setup network topology
2. Distribute keys
3. Determine valid state
SAP: request

Specifies time to attest

\[ T_{\text{chal}} + \log_2(N+1) \times |\text{message}|/\mu \]

where \( N \) is a number of devices and \( \mu \) is transmission rate
SAP: request

Specifies time to attest

\[ T_{\text{chal}} + \log_2(N+1) \times |\text{message}| / \mu \]

where \( N \) is a number of devices and \( \mu \) is transmission rate
SAP: request

Verifier

Specifies time to attest

\[ T_{\text{chal}} + \log_2(N+1) \times |\text{\messageBox}|igg{/}\mu \]

where N is a number of devices and \( \mu \) is transmission rate
SAP: attest
SAP: attest

= HMAC(, SW || time)
SAP: report

1. Forward token
SAP: report

1. Forward token
2. Aggregate tokens using XOR
SAP: report

1. Forward token
2. Aggregate tokens using XOR
1. Forward token
2. Aggregate tokens using XOR
SAP: report
SAP: verify

1. Verify by re-computing all tokens
2. Output 0 or 1
Proof strategy

- Proof of TCA-Efficiency and TCA-Soundness
  - Mathematical proof

- Proof of TCA-Security
  - Extend single-device attestation proof from VRASED [SEC19]
  - Secure hardware: ROM, access control and clock

- See paper for more info!
Evaluation

- Implement **SAP** by extending **TrustLite** [EuroSys14]

- **Overhead**
  - SW: 200 bytes
  - HW: 2.5%
Conclusion

- **Need systematic design for collective remote attestation**
  - Determine appropriate use-case
  - However, too many design parameters...
  - ...no single formal model for all use-cases

- **An example targeting a use-case for timely collective attestation (TCA)**
  - Formal definition of assumptions and goals
  - Propose a protocol, called **SAP**
  - Proof that **SAP** satisfies all goals under assumptions

- **Extensions to other use-cases**
  - Follow the same methodology
  - Modify TCA-Efficiency, TCA-Soundness and TCA-Security
  - Design and proof protocol
Thank you!

Questions?