DATE 14
Design, Automation & Test in Europe
24-28 March, 2014 - Dresden, Germany
The European Event for Electronic System Design & Test

A Minimalist Approach to Remote Attestation

Aurélien Francillon
Eurecom
Quan Nguyen, Kasper B. Rasmussen, Gene Tsudik
UC Irvine

EURECOM
Sophia Antipolis

THE UNIVERSITY OF CALIFORNIA
LET THERE BE LIGHT
IRVINE

Overview

- **Motivation**
- **Definition of Remote Attestation**
- **From Definition to Properties**
- **From Properties to Features**
- **Conclusion**

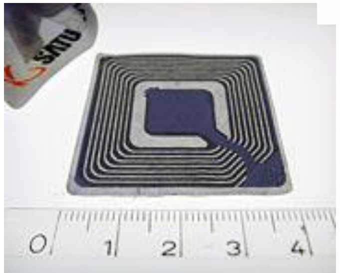
Embedded Systems

SmartCards

Connected devices

Sensors

RFID

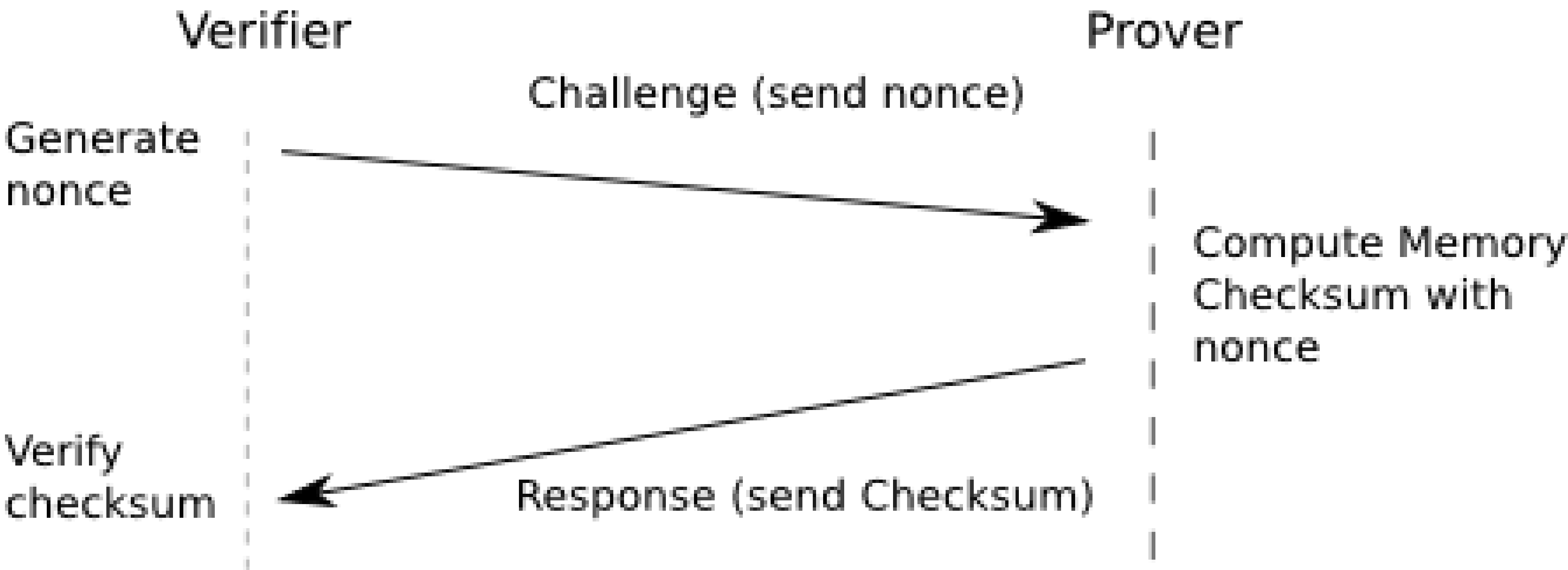


Industrial systems

Remote Attestation

Remote attestation:

- **The act of remotely verifying the state of a device**



Requires guarantees that Prover is not lying

Remote Attestation Examples

Remote attestation can rely on:

- **Static root of trust (TPM, Secure boot, ...)**
  - **Only attests initial state of software**
- **Dynamic root of trust (TXT, ARM TrustZone, SMART, ...)**
- **Software-based attestation**
- **Hybrids of the above (Sancus,..)**

Remote attestation is a popular field

- **Many publications and deployed systems**
- **Some for tiny devices**

Remote Attestation Problem

Lack of agreement about <u>what is remote attestation</u> and its required properties

We define remote attestation and its minimum requirements.

We then apply this to the case of:

- Low-end microcontrollers: HW can be modified
- Software attacks
- Basic hardware interaction (not really hardware attacks)

The Definition

An attestation protocol P = (Setup, Attest, Verify):

- **k = Setup(1^{κ})**

  a setup procedure to generate a shared key

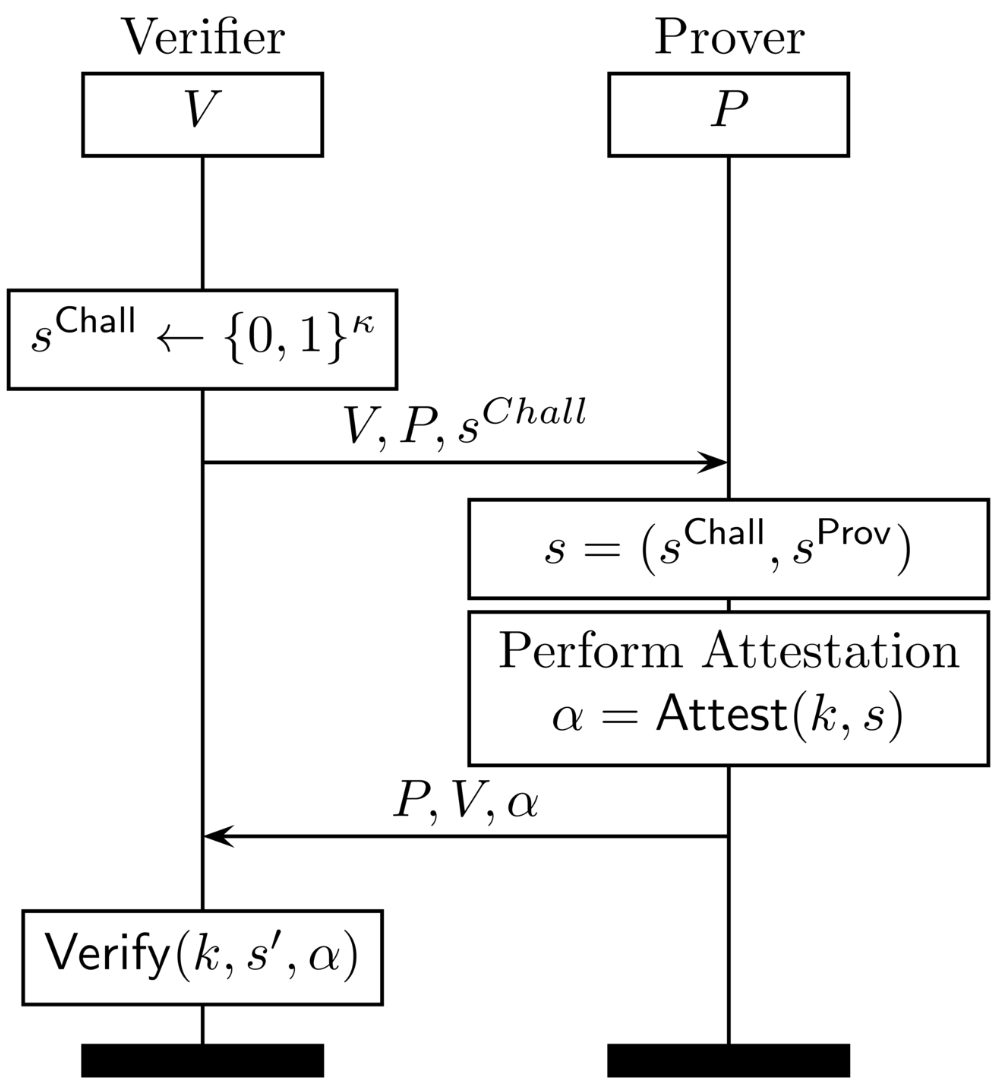
- **α = Attest(k, s)**

  Key, Device state => Attestation token

- **verdict = Verify(k, s', α)**

  Key, Expected state, Token => Yes/No

Remote Attestation



The Att-Forgery Game

We define $\text{Att-Forgery}_{\text{Chal,Prov}}(\kappa)$ **game, as:**

- **Prover has q attempts to generate states that differ from its real state and submit them to Attest() oracle**
- **Eventually returns an α to the verifier**

Game outputs 1 iff Verify(k, s, α) = 1

The protocol is Att-Forgery-secure if:

- **Probabilistic polynomial time prover Prov**
- **Large enough** ***K***

$$Pr[\text{Att-Forgery}_{\text{Chal,Prov}}(\kappa) = 1] \leq negl(\kappa)$$

Requirements and Attacks

From the definition we see that

- **Only attest can compute α**
- **α = Attest(k, s) captures the device state**

This leads to 2 attack types

- **Adversary knows k, simulates attest, computes α**
- **Returned α does not correspond to prover’s actual state**

Properties

- **Exclusive Access**
  - **Only *Attest(k,s)*, can access k**
- **No Leaks**
  - **Only α should depend on k**
  - **No side channels or information leakages**
- **Immutability**
- **Un-interruptibility**
- **Controlled Invocation**

From Properties to Features

- High-level properties → Features
- Features are implementation choices and constraints. We chose them so as to:
  - Have minimal impact on the system
  - Be necessary and sufficient to guaranty security properties
- However we claim minimality of properties, which are design independent, not Features

Features

- Key: Hardware protection from software access
- No Leaks
  - Memory erasure, side-channel resistance?
- Immutability
  - Attest code resides in ROM
- Uninterruptibility
  - Attest is atomic, IRQ disabled…
- Controlled Invocation
  - Execution only from valid entry points, hardware support

Conclusion

In-depth systematic treatment of remote attestation, from which we derived:

- **definitions and global security goals**
- **derived properties**
- **which are mapped into required features**

Helps identify limitations and shortcomings of current designs:

- **Many attacks discovered by checking manually**
- **See long version of the paper**

Future work

- **perform formal verification / proofs of real systems**

Conclusion

Questions ?

Extra slides

Minimality of properties

- **Exclusive Access**
  - **If adversary learns key,**
- **No Leaks**
  - **Information about k**
- **Immutability**
  - **Changing the code could be fatal**
- **Uninterruptibility**
  - **Moving malicious code during attestation**
- **Controlled Invocation**
  - **Invoking attest by skipping parts of it**