A Minimalist Approach to Remote Attestation

Aurélien Francillon
Eurecom

Quan Nguyen, Kasper B. Rasmussen, Gene Tsudik
UC Irvine
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 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
Lack of agreement about what is remote attestation 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)
An attestation protocol $P = (\text{Setup}, \text{Attest}, \text{Verify})$:

- $k = \text{Setup}(1^\kappa)$
  
a setup procedure to generate a shared key
- $\alpha = \text{Attest}(k, s)$
  
  Key, Device state => Attestation token
- $\text{verdict} = \text{Verify}(k, s', \alpha)$
  
  Key, Expected state, Token => Yes/No
Remote Attestation

Verifier

$V$

$S^{\text{Chall}} \leftarrow \{0, 1\}^\kappa$

Prover

$P$

$s = (S^{\text{Chall}}, S^{\text{Provin}})$

Perform Attestation

\[ \alpha = \text{Attest}(k, s) \]

Verify$(k, s', \alpha)$
We define $\text{Att-Forgery}_{\text{Chal}, \text{Prov}}(\kappa)$ game, as:

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

Game outputs 1 iff $\text{Verify}(k, s, \alpha) = 1$

The protocol is Att-Forgery-secure if:

- Probabilistic polynomial time prover Prov
- Large enough $\kappa$

$$\Pr[\text{Att-Forgery}_{\text{Chal}, \text{Prov}}(\kappa) = 1] \leq \text{negl}(\kappa)$$
From the definition we see that

- Only attest can compute $\alpha$
- $\alpha = \text{Attest}(k, s)$ captures the device state

This leads to 2 attack types

- Adversary knows $k$, simulates attest, computes $\alpha$
- Returned $\alpha$ does not correspond to prover’s actual state
Properties

• Exclusive Access
  – Only $Attest(k,s)$, can access $k$

• No Leaks
  – Only $\alpha$ should depend on $k$
  – No side channels or information leakages

• Immutability

• Un-interruptibility

• Controlled Invocation
From Properties to Features

• High-level properties $\rightarrow$ 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
Questions?
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