Formal Specification and Verification of Attestation in Confidential Computing

Muhammad Usama Sardar

TU Dresden

November 8, 2023

Funded by CPEC
Agenda

1 Motivation

2 Approach

3 Results

4 Summary
We all know RATS\textsuperscript{1}

\textsuperscript{1}Birkholz et al., \textit{Remote ATtestation procedureS (RATS) Architecture}, 2023.
But is RATS sufficient for CC (e.g., SGX)?

Signing Your Own Quotes

We understand that remote attestation can be very tricky to pass. However, since we already done all the hard work of getting genuine attestation keys, we decided to help you out by developing a Twitter bot that passes SGX attestation for you. Our bot provides Attestation as a Service (AaaS), which allows you to get your own quotes signed with the keys we extracted using SG Axe. This way you can pass attestation without even owning an SGX machine. If you want to make use of our service, you can send a tweet to our bot @SGAxe_AaaS. If you’ll tweet it, we’ll sign it!

@SGAxe_AaaS

2 www.sgaxe.com
More recently TDX³

Intel Let Google Cloud Hack Its New Secure Chips and Found 10 Bugs

To protect its Confidential Computing cloud infrastructure and gain critical insights, Google leans on its relationships with chipmakers.

Architecturally-defined Attestation in CC

Verifier

Attestation challenge

Attester

---

Architecturally-defined Attestation in CC

Verifier

Attestation challenge

Evidence

Attester

---

Architecturally-defined Attestation in CC

Verifier

Attestation challenge
Evidence
Secrets or sensitive data

Attester

---

Architecturally-defined Attestation

- Holistic coverage of phases
Architecturally-defined Attestation

- Holistic coverage of phases
  - Provisioning
Architecturally-defined Attestation

- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
Architecturally-defined Attestation

- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
    - Verifier Provisioning
Architecturally-defined Attestation

- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
    - Verifier Provisioning
  - Initialization
Architecturally-defined Attestation

- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
    - Verifier Provisioning
  - Initialization
  - Attestation Protocol
Contributions

- Most detailed formal model of Intel TDX attestation, including
Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain

First formal analysis of Arm CCA attestation
Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
  - Initialization phase
Contributions

• Most detailed formal model of Intel TDX attestation, including
  • Certificate chain
  • Verifier steps
  • Initialization phase
  • Variable measurements
Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
  - Initialization phase
  - Variable measurements
- Formal proof of insecurity of Intel’s claimed TCB
Contributions

• Most detailed formal model of Intel TDX attestation, including
  • Certificate chain
  • Verifier steps
  • Initialization phase
  • Variable measurements
• Formal proof of insecurity of Intel’s claimed TCB
• First formal analysis of Arm CCA attestation
Intel TDX Module challenge (nonce)
Request TD report
Assemble tdi from TDCS and compute \( \text{tdih} = \text{hash(tdi)} \)

Request SEAMREPORT

Quote = QuoteHeader || QuoteBody || AKsig || AKcert || PCKcert || ICAcert || rootcert

Verification result (true/false)

Quote

17

AKsig = sign (AK, QuoteHeader || QuoteBody)

Check hashes tcbh = hash(tcbi) & tdih = hash(tdi)?

REPORTMACSTRUCT (msa = mBody || mac)

Verify report mac = hmac(MK, mBody)?)

Verify Sign chain, freshness and measurements event QuoteVerified(tcbiClaims, rdata)

PCE \( \leftrightarrow \) PCK

Guest TD (User TD) \( \leftrightarrow \) TDK

Host VMM

Intel TDX Module

Verifier \( \leftrightarrow \) pubIRK

CPU Hardware (SoC) \( \leftrightarrow \) MK

Secure

Secure

Secure

Secure

Local Attestation

Local Attestation

Local Attestation

Attestation Protocol

 Initialization

Attestation in CC

November 8, 2023 9 / 19
Challenge: Complicated designs with vague and outdated specs and very little support

In the "Get TDX TCB Info" flow (https://api.portal.trustedservices.intel.com/documentation#pcs-tcb-info-tdx-v4), step 4 states:

"For the selected TCB level verify that SVN at index 1 in tdxtcbcomponents array matches the value of SVN at index 1 in TEE TCB SVNs array (from TD Report in Quote). In case of a mismatch the selected TCB level should be rejected as TCB Info that was used for the comparison is not supported for this platform configuration."

My question is:

What is so special about index 1 that it requires an equality check? What does index 1 represent? Typically all SVNs have a non-equality check (>=) as in step 3 (a,b,c).
Properties

- Sanity checks
Properties

- Sanity checks
- Integrity of Evidence
Properties

- Sanity checks
- Integrity of Evidence
- Freshness of Evidence
Properties

- Sanity checks
- Integrity of Evidence
- Freshness of Evidence
- Confidentiality/Secrecy of attestation-related keys
Properties

- Sanity checks
- Integrity of Evidence
- Freshness of Evidence
- Confidentiality/Secrecy of attestation-related keys
- Attester Authentication
Outline

1 Motivation

2 Approach

3 Results

4 Summary
TCB Claimed by Intel\textsuperscript{6}

\textbf{TRUSTED BY TD}:
- INTEL\textsuperscript{®} TDX MODULE
- INTEL AUTHENTICATED CODE MODULES (ACM)
- TD QUOTING ENCLAVE
- INTEL CPU HARDWARE

\textbf{NOT TRUSTED BY TD}:
- PLATFORM ADMIN
- DISCRETE AND INTEGRATED DEVICES
- ALL OTHER SOFTWARE
- OTHER PLATFORM Firmware
- HOST-OS/VMM
- BIOS/SMM

\textit{Legend}:
- Entity on Intel key server
- X.509 certs
- custom format cert-like structure
- CRLs

\textit{Figure 5.1. Trust Boundaries for TDX}

---

\textsuperscript{6}Intel, \textit{Intel \textsuperscript{®} Trust Domain Extensions}, 2021.
# Verification Summary in ProVerif

<table>
<thead>
<tr>
<th></th>
<th>Integrity</th>
<th>Freshness</th>
<th>Confidentiality</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel’s claimed</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Our proposed</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
Take-home

• Identified grey areas in RATS (e.g., Endorsements)
  • can be more precise! Thanks to Dave for pursuing this!
• Need for systematic design of attestation protocols
Take-home

• Identified grey areas in RATS (e.g., Endorsements)
  • can be more precise! Thanks to Dave for pursuing this!
• Need for systematic design of attestation protocols
• Open questions
Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?

Trusted until formally verified!

Muhammad Usama Sardar
Take-home

• Identified grey areas in RATS (e.g., Endorsements)
  • can be more precise! Thanks to Dave for pursuing this!
• Need for systematic design of attestation protocols
• Open questions
  • How to verify the Verifier?
  • How to verify the runtime configurations?
Take-home

• Identified grey areas in RATS (e.g., Endorsements)
  • can be more precise! Thanks to Dave for pursuing this!
• Need for systematic design of attestation protocols
• Open questions
  • How to verify the Verifier?
  • How to verify the runtime configurations?
  • Is RATS architecture better than split architecture?
Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?
  - How to verify the runtime configurations?
  - Is RATS architecture better than split architecture?
Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?
  - How to verify the runtime configurations?
  - Is RATS architecture better than split architecture?

Trusted until formally verified!
Next steps

- TEEP WG
Next steps

• TEEP WG
  • Found a problem\(^8\) in FV of TEEP during hackathon

---

\(^8\) [https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1](https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1)
Next steps

- TEEP WG
  - Found a problem\(^8\) in FV of TEEP during hackathon
  - Integrate RA with TEEP

\(^8\)https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1
Next steps

- TEEP WG
  - Found a problem in FV of TEEP during hackathon
  - Integrate RA with TEEP
- UFMRG: Sample problem

---

8 https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1
Next steps

- TEEP WG
  - Found a problem\(^8\) in FV of TEEP during hackathon
  - Integrate RA with TEEP

- UFMRG: Sample problem

- TLS WG

\(^8\)https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1
Next steps

• TEEP WG
  • Found a problem\(^8\) in FV of TEEP during hackathon
  • Integrate RA with TEEP

• UFMRG: Sample problem

• TLS WG
  • Attested TLS
    https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

\(^8\)https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1


Call to Action

• Bring your expertise: https://github.com/CCC-Attestation/formal-spec-TEE

• Additional information: link here

• Paper on formal verification coming soon