Demystifying Attestation in Intel Trust Domain Extensions (TDX) via Formal Verification

Muhammad Usama Sardar and Christof Fetzer
Ack: Anna Galanou, Amna Shahab, Bruno Blanchet
Funding: CPEC, CeTI

Chair of Systems Engineering
Institute of Systems Architecture
Technische Universität Dresden

Dresden, Germany

April 23, 2022
Promise of talk

• Need of logic in an emerging and important domain
Promise of talk

- Need of **logic** in an emerging and important domain
- CCC: more **marketing** than scientific\(^1,^2\) (highlights only)

\(^1\) Confidential Computing Consortium, *Whitepaper feedback from Muhammad Usama Sardar, Issue #77, 2020*

Promise of talk

- Need of **logic** in an emerging and important domain
- CCC: more **marketing** than scientific\(^1,^2\) (highlights only)
  - Attestation: one of the most **critical and essential** parts of TEE

---

\(^1\) Confidential Computing Consortium, *Whitepaper feedback from Muhammad Usama Sardar, Issue #77, 2020*

\(^2\) Sardar and Fetzer, *Confidential Computing and Related Technologies: A Review, 2021*
Promise of talk

• Need of logic in an emerging and important domain
• CCC: more marketing than scientific\textsuperscript{1,2} (highlights only)
  • Attestation: one of the most critical and essential parts of TEE
• Complexity is the worst enemy of security (B. Schneier)

\textsuperscript{1}Confidential Computing Consortium, \textit{Whitepaper feedback from Muhammad Usama Sardar, Issue #77, 2020}
\textsuperscript{2}Sardar and Fetzer, \textit{Confidential Computing and Related Technologies : A Review, 2021}
Promise of talk

• Need of **logic** in an emerging and important domain
• CCC: more **marketing** than scientific\(^1,2\) (highlights only)
  • Attestation: one of the most **critical and essential** parts of TEE
• Complexity is the **worst enemy** of security (B. Schneier)
  • Complexity is the **best friend** of Intel!

---

\(^1\) Confidential Computing Consortium, *Whitepaper feedback from Muhammad Usama Sardar, Issue #77, 2020*

\(^2\) Sardar and Fetzer, *Confidential Computing and Related Technologies: A Review, 2021*
Outline

1. Introduction

2. Formal Security Analysis Approach

3. TDX
   - Formal Specification
   - Discrepancies Identified
   - Automated Verification

4. Summary
Developer
CC in Public Cloud Scenario

Developer → Application

Application contains TEE
TEEs Granularity (Public cloud commercial solutions)

- Process-based
  - Smaller TCB
TEEs Granularity (Public cloud commercial solutions)

- Process-based
- VM-based

- Ease of use
TEEs Granularity (Public cloud commercial solutions)

- TEE
  - Process-based
    - SGX
  - VM-based
TEEs Granularity (Public cloud commercial solutions)

- TEE
  - Process-based
    - SGX
  - VM-based
    - TDX
TEEs Granularity (Public cloud commercial solutions)

- TEE
  - Process-based
    - SGX
    - RA: EPID-based
  - VM-based
    - TDX
**TEEs Granularity (Public cloud commercial solutions)**

- **Process-based**
  - **SGX**
    - RA: EPIID-based
- **VM-based**
  - **TDX**
    - RA: DCAP

Smaller TCB
Ease of use
Different report generation mechanism
Runtime TD measurements
TEEs Granularity (Public cloud commercial solutions)

- TEE
  - Process-based
    - SGX
      - RA: EPID-based
  - VM-based
    - TDX
      - RA: DCAP

- Different report generation mechanism
- Runtime TD measurements
Attestation

- **Trust** to developer: right app in right platform
Attestation

- **Trust** to developer: right app in right platform
Attestation

- **Trust** to developer: right app in right platform

Verify

- Identity of application
- Validity of platform
Attestation

- **Trust** to developer: right app in right platform

- Verify
  - Identity of application
  - Validity of platform

- Secure channel creation
Attestation

• **Trust** to developer: right app in right platform

  ![Diagram](image)

  Verify

  - Identity of application
  - Validity of platform

• Secure channel creation

• Importance → **Provisioning of secrets and config.**
Workflow of the Analysis Approach

System configuration
Workflow of the Analysis Approach

- System configuration
- Operational policies
Workflow of the Analysis Approach

System configuration → Symbolic model → Operational policies
Workflow of the Analysis Approach

- System configuration
- Operational policies
- Security goals

Symbolic model
Workflow of the Analysis Approach

- System configuration
- Operational policies
- Security goals

Symbolic model
Security properties
Workflow of the Analysis Approach

System configuration → Symbolic model → Automatic translation and resolution → Security properties

Operational policies

Security goals
Workflow of the Analysis Approach

- System configuration
- Operational policies
- Security goals

- Symbolic model
- Security properties

- Automatic translation and resolution

- Fact derivable
  - Attack reconstruction
Workflow of the Analysis Approach

- System configuration
- Operational policies
- Security goals

Symbolic model

Security properties

Automatic translation and resolution

Fact derivable

Attack reconstruction
Workflow of the Analysis Approach

System configuration

Operational policies

Security goals

Symbolic model

Security properties

Automatic translation and resolution

Fact derivable

Attack reconstruction

Yes

False

No

Don't know
Workflow of the Analysis Approach

- System configuration
- Operational policies
- Security goals

Symbolic model → Security properties

Automatic translation and resolution

Fact derivable
- True
- False
- Don’t know

Not derivable

Attack reconstruction
- Yes
- No
Inference System and Horn Clauses (Simplified)

- Composition rules

- Decomposition rules

- Projection

- Decomposition

- Verification
Inference System and Horn Clauses (Simplified)

- Composition rules
  - pair \( \frac{x}{\langle x, y \rangle} \) \( att(x) \land att(y) \rightarrow att(\langle x, y \rangle) \)
Inference System and Horn Clauses (Simplified)

- Composition rules
  - pair $\frac{x \quad y}{\langle x, y \rangle}$ \( \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  - hash $\frac{m}{h(m)}$ \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
Inference System and Horn Clauses (Simplified)

- **Composition rules**
  - pair $x \quad y \quad \langle x, y \rangle \quad \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle)$
  - hash $m \quad h(m) \quad \text{att}(m) \rightarrow \text{att}(h(m))$
  - hmac $m_k \quad m \quad \text{hmac}(m_k, m) \quad \text{att}(m_k) \land \text{att}(m) \rightarrow \text{att}(\text{hmac}(m_k, m))$
• Composition rules
  • pair \( \frac{x}{\langle x, y \rangle} \) \( \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  • hash \( \frac{m}{h(m)} \) \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
  • hmac \( \frac{mk}{hmac(mk,m)} \) \( \text{att}(mk) \land \text{att}(m) \rightarrow \text{att}(hmac(mk,m)) \)
  • senc \( \frac{sek}{senc(sek,m)} \) \( \text{att}(sek) \land \text{att}(m) \rightarrow \text{att}(senc(sek,m)) \)
Inference System and Horn Clauses (Simplified)

- **Composition rules**
  - **pair** \( \frac{x}{\langle x,y \rangle} \quad att(x) \land att(y) \rightarrow att(\langle x,y \rangle) \)
  - **hash** \( \frac{m}{h(m)} \quad att(m) \rightarrow att(h(m)) \)
  - **hmac** \( \frac{mk}{hmac(mk,m)} \quad att(mk) \land att(m) \rightarrow att(hmac(mk,m)) \)
  - **senc** \( \frac{sek}{senc(sek,m)} \quad att(sek) \land att(m) \rightarrow att(senc(sek,m)) \)
  - **aenc** \( \frac{aek}{aenc(aek,m)} \quad att(aek) \land att(m) \rightarrow att(aenc(aek,m)) \)

- **Decomposition rules**
  - **projection** \( \langle x,y \rangle \quad x, y \quad att(\langle x,y \rangle) \rightarrow att(x), att(\langle x,y \rangle) \rightarrow att(y) \)
  - **sdec** \( \frac{sek}{senc(sek,m)} \quad att(sect) \land att(m) \rightarrow att(senc(sect,m)) \)
  - **adec** \( \frac{pk(adk)}{aenc(pk(adk),m)} \quad att(adk) \land att(m) \rightarrow att(aenc(pk(adk),m)) \)
  - **verifysign** \( \frac{vpk}{signAppDet(sk,m)} \quad att(pvk) \land att(m) \rightarrow att(signAppDet(sk,m)) \)
Inference System and Horn Clauses (Simplified)

- Composition rules
  - pair \( \frac{x}{\langle x, y \rangle} \) \( \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  - hash \( \frac{m}{h(m)} \) \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
  - hmac \( \frac{mk}{\text{hmac}(mk,m)} \) \( \text{att}(mk) \land \text{att}(m) \rightarrow \text{att}(\text{hmac}(mk,m)) \)
  - senc \( \frac{sek}{\text{senc}(sek,m)} \) \( \text{att}(sek) \land \text{att}(m) \rightarrow \text{att}(\text{senc}(sek,m)) \)
  - aenc \( \frac{aeq}{\text{aenc}(aeq,m)} \) \( \text{att}(aeq) \land \text{att}(m) \rightarrow \text{att}(\text{aenc}(aeq,m)) \)
  - sign \( \frac{sk}{\text{signAppDet}(sk,m)} \) \( \text{att}(sk) \land \text{att}(m) \rightarrow \text{att}(\text{signAppDet}(sk,m)) \)
Inference System and Horn Clauses (Simplified)

• Composition rules
  • pair \( \frac{x}{\langle x, y \rangle} \) \( \text{att}(x) \wedge \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  • hash \( \frac{m}{h(m)} \) \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
  • hmac \( \frac{mk}{hmac(mk, m)} \) \( \text{att}(mk) \wedge \text{att}(m) \rightarrow \text{att}(hmac(mk, m)) \)
  • senc \( \frac{sek}{senc(sek, m)} \) \( \text{att}(sek) \wedge \text{att}(m) \rightarrow \text{att}(senc(sek, m)) \)
  • aenc \( \frac{aek}{aenc(aek, m)} \) \( \text{att}(aek) \wedge \text{att}(m) \rightarrow \text{att}(aenc(aek, m)) \)
  • sign \( \frac{sk}{signAppDet(sk, m)} \) \( \text{att}(sk) \wedge \text{att}(m) \rightarrow \text{att}(signAppDet(sk, m)) \)

• Decomposition rules
Inference System and Horn Clauses (Simplified)

• **Composition rules**
  - pair \( \frac{x}{\langle x, y \rangle} \) \( \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  - hash \( \frac{m}{h(m)} \) \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
  - hmac \( \frac{mk}{hmac(mk, m)} \) \( \text{att}(mk) \land \text{att}(m) \rightarrow \text{att}(hmac(mk, m)) \)
  - senc \( \frac{sek}{senc(sek, m)} \) \( \text{att}(sek) \land \text{att}(m) \rightarrow \text{att}(senc(sek, m)) \)
  - aenc \( \frac{aek}{aenc(aek, m)} \) \( \text{att}(aek) \land \text{att}(m) \rightarrow \text{att}(aenc(aek, m)) \)
  - sign \( \frac{sk}{signAppDet(sk, m)} \) \( \text{att}(sk) \land \text{att}(m) \rightarrow \text{att}(signAppDet(sk, m)) \)

• **Decomposition rules**
  - projection \( \frac{\langle x, y \rangle}{x} \), \( \frac{\langle x, y \rangle}{y} \) \( \text{att}(\langle x, y \rangle) \rightarrow \text{att}(x), \text{att}(\langle x, y \rangle) \rightarrow \text{att}(y) \)
Inference System and Horn Clauses (Simplified)

- **Composition rules**
  - pair \( \frac{x}{\langle x, y \rangle} \) \( \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  - hash \( \frac{m}{h(m)} \) \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
  - hmac \( \frac{mk}{hmac(mk, m)} \) \( \text{att}(mk) \land \text{att}(m) \rightarrow \text{att}(hmac(mk, m)) \)
  - senc \( \frac{sek}{senc(sek, m)} \) \( \text{att}(sek) \land \text{att}(m) \rightarrow \text{att}(senc(sek, m)) \)
  - aenc \( \frac{aek}{aenc(aek, m)} \) \( \text{att}(aek) \land \text{att}(m) \rightarrow \text{att}(aenc(aek, m)) \)
  - sign \( \frac{sk}{signAppDet(sk, m)} \) \( \text{att}(sk) \land \text{att}(m) \rightarrow \text{att}(signAppDet(sk, m)) \)

- **Decomposition rules**
  - projection \( \frac{\langle x, y \rangle}{x}, \frac{\langle x, y \rangle}{y} \) \( \text{att}(\langle x, y \rangle) \rightarrow \text{att}(x), \text{att}(\langle x, y \rangle) \rightarrow \text{att}(y) \)
  - sdec \( \frac{sek}{\text{senc}(sek, m)} \) \( \text{att}(sek) \land \text{att}(\text{senc}(sek, m)) \rightarrow \text{att}(m) \)
Inference System and Horn Clauses (Simplified)

- **Composition rules**
  - **pair**: \( \frac{x}{\langle x, y \rangle} att(x) \land att(y) \rightarrow att(\langle x, y \rangle) \)
  - **hash**: \( \frac{m}{h(m)} att(m) \rightarrow att(h(m)) \)
  - **hmac**: \( \frac{mk}{hmac(mk, m)} att(mk) \land att(m) \rightarrow att(hmac(mk, m)) \)
  - **senc**: \( \frac{sek}{senc(sek, m)} att( sek ) \land att(m) \rightarrow att(senc( sek, m )) \)
  - **aenc**: \( \frac{aeck}{aenc(aek, m)} att(aek) \land att(m) \rightarrow att(aenc(aek, m)) \)
  - **sign**: \( \frac{sk}{signAppDet(sk, m)} att(sk) \land att(m) \rightarrow att(signAppDet(sk, m)) \)

- **Decomposition rules**
  - **projection**: \( \frac{\langle x, y \rangle}{x} \), \( \frac{\langle x, y \rangle}{y} \) \( att(\langle x, y \rangle) \rightarrow att(x), att(\langle x, y \rangle) \rightarrow att(y) \)
  - **sdec**: \( \frac{sek}{senc( sek, m )} att( sek ) \land att(senc( sek, m )) \rightarrow att(m) \)
  - **adec**: \( \frac{adk}{aenc( pk( adk ), m )} att(adk) \land att(aenc( pk( adk ), m )) \rightarrow att(m) \)
Inference System and Horn Clauses (Simplified)

- **Composition rules**
  - pair \( \frac{x \land y}{\langle x, y \rangle} \) \( \text{att}(x) \land \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \)
  - hash \( \frac{m}{h(m)} \) \( \text{att}(m) \rightarrow \text{att}(h(m)) \)
  - hmac \( \frac{mk \land m}{hmac(mk, m)} \) \( \text{att}(mk) \land \text{att}(m) \rightarrow \text{att}(hmac(mk, m)) \)
  - senc \( \frac{sek \land m}{senc(sek, m)} \) \( \text{att}(sek) \land \text{att}(m) \rightarrow \text{att}(senc(sek, m)) \)
  - aenc \( \frac{aek \land m}{aenc(aek, m)} \) \( \text{att}(aek) \land \text{att}(m) \rightarrow \text{att}(aenc(aek, m)) \)
  - sign \( \frac{sk \land m}{signAppDet(sk, m)} \) \( \text{att}(sk) \land \text{att}(m) \rightarrow \text{att}(signAppDet(sk, m)) \)

- **Decomposition rules**
  - projection \( \frac{\langle x, y \rangle}{x} , \frac{\langle x, y \rangle}{y} \) \( \text{att}(\langle x, y \rangle) \rightarrow \text{att}(x) , \text{att}(\langle x, y \rangle) \rightarrow \text{att}(y) \)
  - sdec \( \frac{sek \land senc(sek, m)}{m} \) \( \text{att}(sek) \land \text{att}(senc(sek, m)) \rightarrow \text{att}(m) \)
  - adec \( \frac{adk \land aenc(pk(adk), m)}{m} \) \( \text{att}(adk) \land \text{att}(aenc(pk(adk), m)) \rightarrow \text{att}(m) \)
  - verifysign \( \frac{vpk(sk) \land m}{true} \) \( \text{signAppDet}(sk, m) \)
Contributions

• Precise specification in ProVerif$^3$

$^3$Blanchet et al., “Modeling and verifying security protocols with the applied pi calculus and ProVerif”, 2016
Contributions

- Precise specification in ProVerif
- Identification of discrepancies

\(^3\)Blanchet et al., “Modeling and verifying security protocols with the applied pi calculus and ProVerif”, 2016
Contributions

- Precise specification in ProVerif$^3$
- Identification of discrepancies
- Automated verification of properties in ProVerif

$^3$Blanchet et al., “Modeling and verifying security protocols with the applied pi calculus and ProVerif”, 2016
TDX Attestation Flow for Quote Generation

1. Attestation request
2. Request TD report
3. Assemble tdi from TDCS and compute its hash tdih
4. Request SEAMREPORT
5. Create SEAMREPORT
6. Create TDREPORT
7. Create TDREPORT
8. Create TDREPORT
9. Request Quote
10. Request Quote
11. Check hashes
12. Call EVERIFYREPORT2
13. Verify report
14. Verification result
15. Sign to form Quote
16. Quote
17. Quote
18. Quote

---

Sardar, Musaev, and Fetzer, “Demystifying Attestation in Intel Trust Domain Extensions via Formal Verification”, 2021
Discrepancies Identified

- SEAMINFO vs. TEE_TCB_INFO (e.g., p.2-8)\(^5\)

\(^5\)Intel, *Intel® Trust Domain CPU Architectural Extensions*, 2020

Discrepancies Identified

- Ambiguous/undefined names
- Missing fields

- MOWNERCONFIG missing in TDINFO (Fig. 10.1, p.85)\(^6\)

---

\(^5\) Intel, *Intel® Trust Domain CPU Architectural Extensions*, 2020

Discrepancies Identified

- Ambiguous/undefined names
- Missing fields
- Inconsistent information

---

5 Intel, *Intel® Trust Domain CPU Architectural Extensions*, 2020
Inconsistent Information: Example 1

Figure 10.1: TDX Measurement Reporting

Inconsistent Information: Example 1

```c
tmp_seamreport.REPORTMACSTRUCT.TEE_TCB_INFO_HASH = SHA384(tmp_seamreport.TEE_TCB_INFO);
```

---

**Table 2-3. TEE_TCB_INFO Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Offset (Bytes)</th>
<th>Size (Bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALID</td>
<td>0</td>
<td>8</td>
<td>Indicates TEE_TCB_INFO fields which are valid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 in the i-th significant bit reflects that the 8 bytes starting at</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>offset (8 * i) are valid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 in the i-th significant bit reflects that either 8 bytes starting at</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>offset (8 * i) is not populated or reserved, and is set to zero.</td>
</tr>
<tr>
<td>TEE_TCB_SVN</td>
<td>8</td>
<td>16</td>
<td>TEE_TCB_SVN array.</td>
</tr>
<tr>
<td>MRSIGNERSEAM</td>
<td>72</td>
<td>48</td>
<td>Measurement of TDX module signer if valid.</td>
</tr>
<tr>
<td>ATTRIBUTES</td>
<td>120</td>
<td>8</td>
<td>Additional configuration ATTRIBUTES if valid.</td>
</tr>
<tr>
<td>RESERVED</td>
<td>128</td>
<td>111</td>
<td>Must be zero.</td>
</tr>
</tbody>
</table>

---

\(^8\)Intel, *Intel® Trust Domain CPU Architectural Extensions*, 2020
Automated Verification

- Validation: reachability of all parts of code
- Confidentiality: reachability property
- Authentication properties, e.g.,
  \[ x \equiv \langle rtyp, res1, csvn, tcbh, tdih, rdata, res2 \rangle \]

\[ \forall x. \]
\[ \exists mac, tcbi. \]
\[ \text{event}(\text{QuoteVerified}(x)) \Rightarrow \text{event}(\text{CPUsentSMR}(x, mac, tcbi)) \]
Outline

1 Introduction

2 Formal Security Analysis Approach

3 TDX
   - Formal Specification
   - Discrepancies Identified
   - Automated Verification

4 Summary
Take-home

- TDX specifications are inconsistent and poorly documented
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws

- Reported to Intel and being updated by Intel

- Works in progress

- Model: PCE and cert chain, verifier end

- Properties:
  - Mutual authentication
  - Freshness

- Reveal the hidden assumptions of Intel (e.g., on verifier side)

- Shameless plug: we are hiring PhDs, post-docs
  - [muhammad.usama.sardar, christof.fetzer]@tu-dresden.de

Muhammad Usama Sardar
Formal Analysis of Intel TDX
April 23, 2022
18 / 19
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws
- Reported to Intel and being updated by Intel
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws
- Reported to Intel and being updated by Intel
- Works in progress
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws
- Reported to Intel and being updated by Intel
- Works in progress
  - Model: PCE and cert chain, verifier end
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws
- Reported to Intel and being updated by Intel
- Works in progress
  - Model: PCE and cert chain, verifier end
  - Properties:
Take-home

• TDX specifications are inconsistent and poorly documented
  • may lead to design and implementation flaws
• Reported to Intel and being updated by Intel
• Works in progress
  • Model: PCE and cert chain, verifier end
  • Properties:
    • Mutual authentication

Shameless plug: we are hiring PhDs, post-docs

[muhammad.usama.sardar,christof.fetzer]@tu-dresden.de
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws
- Reported to Intel and being updated by Intel
- Works in progress
  - Model: PCE and cert chain, verifier end
  - Properties:
    - Mutual authentication
    - Freshness

Shameless plug: we are hiring PhDs, post-docs
([muhammad.usama.sardar,christof.fetzer]@tu-dresden.de)
Take-home

- TDX specifications are inconsistent and poorly documented
  - may lead to design and implementation flaws
- Reported to Intel and being updated by Intel
- Works in progress
  - Model: PCE and cert chain, verifier end
  - Properties:
    - Mutual authentication
    - Freshness
  - Reveal the hidden assumptions of Intel (e.g., on verifier side)

Shameless plug: we are hiring PhDs, post-docs
[muhammad.usama.sardar,christof.fetzer]@tu-dresden.de
Take-home

• TDX specifications are inconsistent and poorly documented
  • may lead to design and implementation flaws
• Reported to Intel and being updated by Intel
• Works in progress
  • Model: PCE and cert chain, verifier end
  • Properties:
    • Mutual authentication
    • Freshness
  • Reveal the hidden assumptions of Intel (e.g., on verifier side)
• Shameless plug: we are hiring PhDs, post-docs
  ([muhammad_usama.sardar,christof.fetzer]@tu-dresden.de)


