Towards Formal Analysis of Attested TLS

Muhammad Usama Sardar\textsuperscript{1}, Arto Niemi\textsuperscript{2}, Hannes Tschofenig\textsuperscript{3} and Thomas Fossati\textsuperscript{4}

\textsuperscript{1}TU Dresden, Germany
\textsuperscript{2}Huawei Technologies, Helsinki, Finland
\textsuperscript{3}University of Applied Sciences Bonn-Rhein-Sieg and Siemens, Germany
\textsuperscript{4}Linaro, Lausanne, Switzerland

July 24, 2024
Agenda

1. Background
2. Goal
3. Approach, Tool and Challenges
4. Validation of TLS 1.3
5. Formal Analysis of Attested TLS
6. Summary
TLS\(^1\)

- Good for *network* security

\(^1\)https://datatracker.ietf.org/doc/html/rfc8446
TLS

- Good for network security
- Not good for endpoint security

---

TLS\textsuperscript{1}

- Good for *network* security
- Not good for *endpoint* security
  - Keys

\textsuperscript{1}https://datatracker.ietf.org/doc/html/rfc8446
TLS\textsuperscript{1}

- Good for \textit{network} security
- Not good for \textit{endpoint} security
  - Keys
  - Software

\textsuperscript{1}https://datatracker.ietf.org/doc/html/rfc8446
**TLS**

- Good for *network* security
- Not good for *endpoint* security
  - Keys
  - Software
  - Platform

---

TLS

- Good for network security
- Not good for endpoint security
  - Keys
  - Software
  - Platform
- Use case: Confidential Computing

---

Remote Attestation (Key idea)

Attestation

Generation of Evidence

Attester

Ver. RP

Generation of Evidence

Transmission of Evidence

Appraisal of Evidence
Remote Attestation (Key idea)

- Attestation
  - Generation of Evidence
  - Transmission of Evidence

Attester

Ver. RP

Generation of Evidence

Transmission of Evidence

Appraisal of Evidence
Remote Attestation (Key idea)

Attestation

- Generation of Evidence
- Transmission of Evidence
- Appraisal of Evidence

Attester -> Generation of Evidence

Transmission of Evidence

Ver. RP -> Appraisal of Evidence
Design Options for Attested TLS
Design Options for Attested TLS

Attested TLS

- Pre-Handshake Attestation
- Intra-Handshake Attestation
- Post-Handshake Attestation

**TLS Handshake**

- Signing of evidence
- Key Exchange
- Authentication

---

Muhammad Usama Sardar (TUD)  
IETF 120 UFMRG  
July 24, 2024
Pre-HS: Intel’s RA-TLS

- Widely used pre-HS attestation protocol, e.g., in
  - Gramine\(^2\)
  - RATS-TLS\(^3\)
  - Open Enclave Attested TLS\(^4\)
  - SGX SDK Attested TLS\(^5\)

---

\(^2\)https://github.com/gramineproject/gramine/tree/master/CI-Examples/ra-tls-mbedtls
\(^3\)https://github.com/inclavare-containers/rats-tls
\(^4\)https://github.com/openenclave/openenclave/tree/master/samples/attestedtls
\(^5\)https://github.com/intel/linux-sgx/tree/master/SampleCode/SampleAttestedTLS
Outline

1. Background
2. Goal
3. Approach, Tool and Challenges
4. Validation of TLS 1.3
5. Formal Analysis of Attested TLS
   - Flow
   - Threat Model
   - Properties
   - Proposed Mitigations/Fix
6. Summary
Goal

- Formally analyze the security of Intel’s RA-TLS
Outline

1. Background

2. Goal

3. Approach, Tool and Challenges

4. Validation of TLS 1.3

5. Formal Analysis of Attested TLS
   - Flow
   - Threat Model
   - Properties
   - Proposed Mitigations/Fix

6. Summary
Analysis Approach and Tool

• Approach: Symbolic\textsuperscript{7}


\textsuperscript{8}Blanchet, Cheval, and Cortier, “ProVerif with lemmas, induction, fast subsumption, and much more”, 2022.
Analysis Approach and Tool

- Approach: Symbolic\textsuperscript{7}
- Tool used: ProVerif\textsuperscript{8}

\textsuperscript{8}Blanchet, Cheval, and Cortier, “ProVerif with lemmas, induction, fast subsumption, and much more”, 2022.
Approach - Simplified

Diagram:
- Specs
  - Formal Model
  - Security Properties
- Validator
- Formal Analysis
  - Proof
  - Attack
Challenge in Specification of Intel’s RA-TLS

- Incomplete and outdated specs for RA-TLS
  - Specs based on TLS 1.2 (TLS 1.3 is RFC since Aug 2018)
  - Fix: Used implementation and community input for formal model

---

Challenge in Specification of Intel’s RA-TLS

- Incomplete and outdated specs for RA-TLS
  - Specs based on TLS 1.2 (TLS 1.3 is RFC since Aug 2018)
  - Fix: Used implementation and community input for formal model

![Diagram of Remote Attestation Example](image1)

*Figure 1: Remote Attestation Example. The challenger is off-platform with respect to the attester.*

![TLS 1.2 Handshake Messages](image2)

*Figure 2: TLS 1.2 Handshake Messages.*

---

Challenges from Formal Perspective

- Very few comments in Inria's TLS formal model\(^{10}\)

---

\(^{10}\)https://github.com/Inria-Prosecco/reftls/tree/master/pv

\(^{11}\)https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUE9X4JnrX8/
Challenges from Formal Perspective

- Very few comments in Inria’s TLS formal model\(^{10}\)
  - Literally no comments at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!

\(^{10}\)https://github.com/Inria-Prosecco/reftls/tree/master/pv
\(^{11}\)https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUE9X4JnrXx8/
Challenges from Formal Perspective

- Very few comments in Inria’s TLS formal model\textsuperscript{10}
  - Literally no comments at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  - Fix: Added extensive comments for future extensions

\textsuperscript{10}https://github.com/Inria-Prosecco/reftls/tree/master/pv
\textsuperscript{11}https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUe9X4JnrxX8/
Challenges from Formal Perspective

- Very **few comments** in Inria’s TLS formal model\(^{10}\)
  - Literally **no comments** at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  - **Fix**: Added extensive comments for future extensions
  - **Proposal**: ACM-like badges (e.g., reusable)

---

\(^{10}\)https://github.com/Inria-Prosecco/reftls/tree/master/pv

\(^{11}\)https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUe9X4JnrX8/
Challenges from Formal Perspective

- **Very few comments** in Inria’s TLS formal model\(^{10}\)
  - Literally **no comments** at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  - Fix: Added extensive comments for future extensions
  - Proposal: ACM-like badges (e.g., reusable)

- **Incomplete validation** of draft 20 artifacts\(^{11}\)

---

\(^{10}\)[https://github.com/Inria-Prosecco/reftls/tree/master/pv]

\(^{11}\)[https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUe9X4JnrX8/]

Muhammad Usama Sardar (TUD)
Challenges from Formal Perspective

- Very **few comments** in Inria’s TLS formal model\(^\text{10}\)
  - Literally **no comments** at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  - Fix: Added extensive comments for future extensions
  - Proposal: ACM-like badges (e.g., reusable)
- **Incomplete validation** of draft 20 artifacts\(^\text{11}\)
  - Fix: Designed an *automated validation framework* for key schedule

\(^{10}\)https://github.com/Inria-Prosecco/reftls/tree/master/pv

\(^{11}\)https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUE9X4JnrxX8/
Challenges from Formal Perspective

• Very few comments in Inria’s TLS formal model\(^{10}\)
  • Literally no comments at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  • Fix: Added extensive comments for future extensions
  • Proposal: ACM-like badges (e.g., reusable)

• Incomplete validation of draft 20 artifacts\(^{11}\)
  • Fix: Designed an automated validation framework for key schedule

• A simple extension made the artifacts running for 1 month on high-end server (icelake)

---

\(^{10}\)https://github.com/Inria-Prosecco/reftls/tree/master/pv
\(^{11}\)https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUe9X4JnrxX8/
Challenges from Formal Perspective

- Very few comments in Inria’s TLS formal model\(^\text{10}\)
  - Literally no comments at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  - Fix: Added extensive comments for future extensions
  - Proposal: ACM-like badges (e.g., reusable)
- Incomplete validation of draft 20 artifacts\(^\text{11}\)
  - Fix: Designed an automated validation framework for key schedule
- A simple extension made the artifacts running for 1 month on high-end server (icelake)
  - Submitted to ProVerif developers for analysis

---

\(^\text{10}\) [https://github.com/Inria-Prosecco/reftls/tree/master/pv](https://github.com/Inria-Prosecco/reftls/tree/master/pv)

\(^\text{11}\) [https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUe9X4JnrxX8/](https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGUe9X4JnrxX8/)
Challenges from Formal Perspective

- Very **few comments** in Inria’s TLS formal model\(^{10}\)
  - Literally **no comments** at all in main processes (such as Client12, Server12, Client13, Server13, appData, channelBindingQuery and secrecyQuery)!
  - Fix: Added extensive comments for future extensions
  - Proposal: ACM-like badges (e.g., reusable)
- **Incomplete validation** of draft 20 artifacts\(^{11}\)
  - Fix: Designed an automated validation framework for key schedule
- A simple extension made the artifacts running for **1 month** on high-end server (icelake)
  - Submitted to ProVerif developers for analysis
  - Fix: Formal model from **scratch**

\(^{10}\)https://github.com/Inria-Prosecco/reftls/tree/master/pv
\(^{11}\)https://mailarchive.ietf.org/arch/msg/tls/-nFk9Eu7n-YFsFfGue9X4JnrX8/
Approach

- Spec. docs
- Community input
- Security Goals
- Update
- Validation
- Formal Model
- Properties
- Sanity checks
- Formal Verification
- Proof
- Attack
- Could not prove
- No result
Outline

1 Background
2 Goal
3 Approach, Tool and Challenges
4 Validation of TLS 1.3
5 Formal Analysis of Attested TLS
   - Flow
   - Threat Model
   - Properties
   - Proposed Mitigations/Fix
6 Summary
Validation Framework

- **TLS 1.3 Specs**
- **Inria artifacts**

- Success
- Failure

== ?

Yes

Success

No

Failure
A trace has been found.

**Honest Process**

- **Beginning of process** `TestKeySch`
- ~\( M \) = Success
- ~\( M_1 \) = Success
- ~\( M_2 \) = Failure
- ~\( M_3 \) = Success
- ~\( M_4 \) = Failure
- ~\( M_5 \) = Failure
- ~\( M_6 \) = Failure
- ~\( M_7 \) = Failure
- ~\( M_8 \) = Failure
- ~\( M_9 \) = Failure
- ~\( M_{10} \) = Failure
- ~\( M_{11} \) = Failure
- ~\( M_{12} \) = Failure

**Attacker**

- ~\( M_{13} \) = Failure

The attacker has the message ~\( M_2 = \text{Failure} \)
Example issue: Master Secret

**Figure: TLS 1.3 Specs**

**Figure: Inria artifacts**

---

12https://github.com/Inria-Prosecco/reftls/issues/6
Ruling out Abstractions

- Ubuntu 20.04 LTS on an Intel Core i7-11800H processor with 64 GB of RAM

<table>
<thead>
<tr>
<th>Code</th>
<th>ProVerif 2.04</th>
<th>ProVerif 2.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>6 min 06.634 s</td>
<td>6 min 02.256 s</td>
</tr>
<tr>
<td>With issue 1 fixed</td>
<td>5 min 51.682 s</td>
<td>6 min 03.335 s</td>
</tr>
<tr>
<td>With issue 2 fixed</td>
<td>7 min 04.472 s</td>
<td>6 min 14.954 s</td>
</tr>
<tr>
<td>With issue 3 fixed</td>
<td>7 min 11.434 s</td>
<td>6 min 41.872 s</td>
</tr>
<tr>
<td>With all 3 issues fixed</td>
<td>6 min 40.010 s</td>
<td>6 min 31.887 s</td>
</tr>
</tbody>
</table>
A “Tale” of Community input

- Paper authors\textsuperscript{13}


\textsuperscript{14} https://github.com/lurk-t/proverif

\textsuperscript{15} https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\textsuperscript{17} https://wiki.ietf.org/meeting/119/hackathon

\textsuperscript{18} https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet


\textsuperscript{14} https://github.com/lurk-t/proverif
\textsuperscript{15} https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/
\textsuperscript{17} https://wiki.ietf.org/meeting/119/hackathon
\textsuperscript{18} https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet
  - Karthikeyan Bhargavan


\textsuperscript{14}https://github.com/lurk-t/proverif
\textsuperscript{15}https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/
\textsuperscript{17}https://wiki.ietf.org/meeting/119/hackathon
\textsuperscript{18}https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi

\textsuperscript{14}https://github.com/lurk-t/proverif
\textsuperscript{15}https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/
\textsuperscript{17}https://wiki.ietf.org/meeting/119/hackathon
\textsuperscript{18}https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A "Tale" of Community input

- Paper authors\(^{13}\)
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\(^{14}\)

---


\(^{14}\) https://github.com/lurk-t/proverif

\(^{15}\) https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\(^{17}\) https://wiki.ietf.org/meeting/119/hackathon

A “Tale” of Community input

- Paper authors\(^{13}\)
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\(^{14}\)
- IETF TLS WG\(^{15}\)

---


\(^{14}\)https://github.com/lurk-t/proverif

\(^{15}\)https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\(^{17}\)https://wiki.ietf.org/meeting/119/hackathon

A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\textsuperscript{14}
- IETF TLS WG\textsuperscript{15}
- IRTF UFMRG chairs

\textsuperscript{14} https://github.com/lurk-t/proverif
\textsuperscript{15} https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/
\textsuperscript{17} https://wiki.ietf.org/meeting/119/hackathon
\textsuperscript{18} https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A “Tale” of Community input

- Paper authors\(^{13}\)
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\(^{14}\)
- IETF TLS WG\(^{15}\)
- IRTF UFMRG chairs
- CCC attestation SIG\(^{16}\)

---


\(^{14}\) https://github.com/lurk-t/proverif

\(^{15}\) https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\(^{17}\) https://wiki.ietf.org/meeting/119/hackathon

A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\textsuperscript{14}
- IETF TLS WG\textsuperscript{15}
- IRTF UFMRG chairs
- CCC attestation SIG\textsuperscript{16}
- ...


\textsuperscript{14} https://github.com/lurk-t/proverif

\textsuperscript{15} https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\textsuperscript{17} https://wiki.ietf.org/meeting/119/hackathon

\textsuperscript{18} https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\textsuperscript{14}
- IETF TLS WG\textsuperscript{15}
- IRTF UFMRG chairs
- CCC attestation SIG\textsuperscript{16}
- ...
- IETF 119 Hackathon\textsuperscript{17}


\textsuperscript{14}https://github.com/lurk-t/proverif

\textsuperscript{15}https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\textsuperscript{17}https://wiki.ietf.org/meeting/119/hackathon

\textsuperscript{18}https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
A “Tale” of Community input

- Paper authors\(^\text{13}\)
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\(^\text{14}\)
- IETF TLS WG\(^\text{15}\)
- IRTF UFMRG chairs
- CCC attestation SIG\(^\text{16}\)
- ...
- IETF 119 Hackathon\(^\text{17}\)
- IRTF Crypto Forum RG @ IETF 119\(^\text{18}\)


\(^{14}\)https://github.com/lurk-t/proverif

\(^{15}\)https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/


\(^{17}\)https://wiki.ietf.org/meeting/119/hackathon

A “Tale” of Community input

- Paper authors\textsuperscript{13}
  - Bruno Blanchet
  - Karthikeyan Bhargavan
  - Nadim Kobeissi
- LURK authors\textsuperscript{14}
- IETF TLS WG\textsuperscript{15}
- IRTF UFMRG chairs
- CCC attestation SIG\textsuperscript{16}
- ...
- IETF 119 Hackathon\textsuperscript{17}
- IRTF Crypto Forum RG @ IETF 119\textsuperscript{18}
- Tool session @ GT MFS’24


\textsuperscript{14} https://github.com/lurk-t/proverif
\textsuperscript{15} https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/
\textsuperscript{17} https://wiki.ietf.org/meeting/119/hackathon
\textsuperscript{18} https://datatracker.ietf.org/meeting/119/materials/slides-119-cfrg-formal-analysis-of-ra-tls-00
Outline

1. Background
2. Goal
3. Approach, Tool and Challenges
4. Validation of TLS 1.3
5. Formal Analysis of Attested TLS
   - Flow
   - Threat Model
   - Properties
   - Proposed Mitigations/Fix
6. Summary
Agenda

5 Formal Analysis of Attested TLS
   • Flow
   • Threat Model
   • Properties
   • Proposed Mitigations/Fix
Before Handshake
\[ rdata = \text{hash}(\text{pubEK}) \]
\[ \text{evidence} = (rdata \parallel \text{dev_status}) \parallel \text{sign}(\text{privAK}, (rdata \parallel \text{dev_status})) \]

1. 
   ClientHello → log_CH
   ServerHello ← log_SH

2. 
   log_CRT
   selfsign = sign(\text{privEK}, (\text{pubEK} \parallel \text{evidence}))
   cert = (\text{pubEK} \parallel \text{evidence}) \parallel \text{selfsign}

3. 
   event Sent(\text{evidence})
   Certificate = cert → log_CRT
   sig = sign(\text{privEK}, \text{log_CRT})
   event Accepted(\text{evidence})
   CertificateVerify = sig ← log_CV
   verify(\text{pubEK}, \text{sig})
   event PreServerFinished
   log_SF
   event ClientFinished
   Finished → log_CF
   log_CRT

Legend
- TLS
- RA
- Initial knowledge
Agenda

Formal Analysis of Attested TLS
- Flow
- Threat Model
- Properties
- Proposed Mitigations/Fix
Threat Model

- Scope: TLS mode $=$ non-PSK handshake
Threat Model

- Scope: TLS mode = non-PSK handshake
- Weak hash, e.g., SLOTH (represented by WeakHash)
- Weak DH groups, e.g., Logjam (represented by WeakDH)
- Bad elements within strong DH groups (rep. by SentBadElement)
- With and without weak (or compromised) ephemeral key privEK
- Without weak (or compromised) attestation key privAK
Threat Model

- Scope: TLS mode = non-PSK handshake
- Weak hash, e.g., SLOTH (represented by WeakHash)
- Weak DH groups, e.g., Logjam (represented by WeakDH)
Threat Model

- Scope: TLS mode = non-PSK handshake
- Weak hash, e.g., SLOTH (represented by $\text{WeakHash}$)
- Weak DH groups, e.g., Logjam (represented by $\text{WeakDH}$)
- Bad elements within strong DH groups (rep. by $\text{SentBadElement}$)
Threat Model

- Scope: TLS mode = non-PSK handshake
- Weak hash, e.g., SLOTH (represented by WeakHash)
- Weak DH groups, e.g., Logjam (represented by WeakDH)
- Bad elements within strong DH groups (rep. by SentBadElement)
- With and without weak (or compromised) ephemeral key privEK
Threat Model

- Scope: TLS mode = non-PSK handshake
- Weak hash, e.g., SLOTH (represented by WeakHash)
- Weak DH groups, e.g., Logjam (represented by WeakDH)
- Bad elements within strong DH groups (rep. by SentBadElement)
- With and without weak (or compromised) ephemeral key privEK
- Without weak (or compromised) attestation key privAK
Agenda

5 Formal Analysis of Attested TLS
- Flow
- Threat Model
- Properties
- Proposed Mitigations/Fix
∀ ev.

\[ \text{inj} \rightarrow \text{event(Accepted}(\text{ev})) \implies \text{inj} \rightarrow \text{event(Sent}(\text{ev})) \]  (1)
Eliminate Reasons of Failure

\[
\forall \ ev.
\exists \ cr, sr, e.

inj \rightarrow \text{event}(\text{Accepted}(ev)) \implies inj \rightarrow \text{event}(\text{Sent}(ev)) \parallel
\text{event}(\text{ServerChoosesKEX}(cr, sr, DHE_{13}(\text{WeakDH}, e))) \parallel
\text{event}(\text{ServerChoosesHash}(cr, sr, \text{WeakHash})) \parallel
\text{event}(\text{SentBadElement}).
\]
Honest Process
A trace has been found.
Attacker
{1} new privAK_1
{2} new privEK_2
!
Beginning of process ...

Abbreviations

~M_7 = pk(privEK_2)
~M_8 = hash_ideal(p2b(pk(privEK_2)))
~M_9 = dev_status
~M_10 = sign(privAK_1,(hash_ideal(p2b(pk(privEK_2))),dev_status))
~M_11 = sign(privEK_2,(pk(privEK_2),((hash_ideal(p2b(pk(privEK_2))),dev_status),sign(privAK_1,(hash_ideal(p2b(pk(privEK_2))),dev_status)))))
~X_1 = CRT(~M_7,((hash_ideal(p2b(~M_7)),dev_status),~M_10),~M_11)
~X_2 = CRT(~M_7,((hash_ideal(p2b(~M_7)),dev_status),~M_10),~M_11)
Simplified Attack Trace

1. ClientHello
2. ServerHello
3. ClientHello
4. ServerHello
5. ClientHello
6. ServerHello
7. Certificate
8. Certificate
9. Certificate

Event Accepted

Honest Process

Client

Client

Server

Attacker

Muhammad Usama Sardar (TUD)
\[ \forall cr, sr, sid. \]

\[ inj - \text{event}(\text{ClientFinished}(cr, sr, sid)) \implies inj - \text{event}(\text{PreServerFinished}(cr, sr, sid)) \quad (2) \]
Eliminate Reasons of Failure

\[
\forall \ cr, sr, sid.
\exists \ cr', sr', e.
\]

\[
inj - \ event(ClientFinished(cr, sr, sid)) \implies
inj - \ event(PreServerFinished(cr, sr, sid)) \parallel
event(ServerChoosesKEX(cr, sr, DHE_13(WeakDH, e))) \parallel
event(ServerChoosesHash(cr', sr', WeakHash)) \parallel
event(SentBadElement).
\]
Summary so far

<table>
<thead>
<tr>
<th>Property</th>
<th>Without privEK leak</th>
<th>With privEK leak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freshness of evidence</td>
<td>× (1.7 s)</td>
<td>× (6 min 56 s)</td>
</tr>
<tr>
<td>Server authentication</td>
<td>✓ (4.6 s)</td>
<td>× (2 min 08 s)</td>
</tr>
</tbody>
</table>

**Table:** Verification results and times for RA-TLS protocol
Agenda

Formal Analysis of Attested TLS
- Flow
- Threat Model
- Properties
- Proposed Mitigations/Fix
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959

- Open Enclave SDK: 20501231235959
- Intel SGX libraries: 20501231235959

Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences to make TLS stateful.

What if the client would like to reconnect later (with the same TCB, i.e., same evidence)? Perhaps it can use PSK then.

Discussion: other thoughts?
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959

- Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences

- Makes TLS stateful

- What if the client would like to reconnect later (with the same TCB, i.e., same evidence)?

- Perhaps it can use PSK then

Discussion: other thoughts?
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959
  - Intel SGX libraries: 20501231235959
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959
  - Intel SGX libraries: 20501231235959
- Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959
  - Intel SGX libraries: 20501231235959
- Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences
  - Makes TLS stateful
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959
  - Intel SGX libraries: 20501231235959

- Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences
  - Makes TLS stateful
  - What if the client would like to reconnect later (with the same TCB, i.e., same evidence)?

Perhaps it can use PSK then

Discussion: other thoughts?
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959
  - Intel SGX libraries: 20501231235959

- Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences
  - Makes TLS stateful
  - What if the client would like to reconnect later (with the same TCB, i.e., same evidence)?
    - Perhaps it can use PSK then
Mitigations within Pre-HS attestation

- Reduce the validity period of certificates
  - Gramine: 20301231235959
  - Open Enclave SDK: 20501231235959
  - Intel SGX libraries: 20501231235959

- Client maintains a database of hash of all evidences it has seen and each time compares with the seen evidences
  - Makes TLS stateful
  - What if the client would like to reconnect later (with the same TCB, i.e., same evidence)?
    - Perhaps it can use PSK then

- Discussion: other thoughts?
Mitigations within Pre-HS attestation

---

**Before Handshake**

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver. Relying Party</td>
<td>Attester</td>
</tr>
<tr>
<td>(pubAK)</td>
<td>(privEK)</td>
</tr>
</tbody>
</table>

1. **ClientHello** → **log_CH**
2. **ServerHello** → **log_SH**
3. **selfsign = sign(privEK, (pubEK || evidence))**
4. **cert = (pubEK || evidence) || selfsign**
5. **event Sent(evidence)**
6. **Certificate = cert** → **log_CRT**
7. **Check if hash(Evidence) exists**
8. **Verify signature on evidence**
9. **Compare hashes**
10. **Store hash(Evidence)**
11. **log_CH**
12. **log_SH**
13. **log_CRT**
14. **event PreServerFinished**
15. **event ClientFinished**
16. **event Accepted(evidence)**
17. **verify(pubEK, sig)**
18. **Finished**
19. **log_SF**
20. **log_CF**

---

**Ver. Relying Party**

- **Client**
  - (pubAK)
- **Server**
  - (privEK)
Proposed Fix: Intra-HS attestation

Client
Ver. Relying Party
(pubAK)

Server
Attester
(privEK, privAK)

ClientHello

ServerHello

selfsign = sign(privEK, (pubEK || evidence))

cert = (pubEK || evidence) || selfsign

event Sent(evidence)

Certificate = cert

sig = sign(privEK, log_CRT)

CertificateVerify = sig

Verify signature on evidence

Compare hashes

event Accepted(evidence)

verify(pubEK, sig)

event PreServerFinished

ClientFinished

log_CH......

log_SH......

log_CRT......

log_CV......

log_SF......

log_CF......

Ver. Relying Party
Client
(pubAK)

Attester
Server
(privEK, privAK)

rdata = hash(pubEK)
evidence = (cr || rdata || dev_status) ||
sign(privAK,(cr || rdata || dev_status))
selfsign = sign(privEK, (pubEK || evidence))
cert = (pubEK || evidence) || selfsign

Check freshness
Verify signature on evidence
Compare hashes
Proposed Fix: Intra-HS attestation

- Leakage of privEK can be detected (as long as privAK is not leaked)

<table>
<thead>
<tr>
<th>Property</th>
<th>Without privEK leak</th>
<th>With privEK leak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freshness of evidence</td>
<td>✓ (02.5 s)</td>
<td>✓ (02 min 43.7 s)</td>
</tr>
<tr>
<td>Server authentication</td>
<td>✓ (13.9 s)</td>
<td>× (15 min 55.3 s)</td>
</tr>
</tbody>
</table>

Table: Verification results and times for our proposed minimal fix to RA-TLS protocol
Outline

1. Background
2. Goal
3. Approach, Tool and Challenges
4. Validation of TLS 1.3
5. Formal Analysis of Attested TLS
   - Flow
   - Threat Model
   - Properties
   - Proposed Mitigations/Fix
6. Summary
Summary

• Intel’s RA-TLS is potentially vulnerable to evidence replay attacks
Summary

• Intel’s RA-TLS is potentially vulnerable to evidence replay attacks
  • Need for standardized and formally verified attested TLS

Questions for discussion

• Is there any practical attack if the derive secret for 'ms' is skipped?
• Is a fix for RA-TLS within pre-HS attestation possible without storing state (evidence)?
• How to deal with underspecification?
Summary

• Intel’s RA-TLS is potentially vulnerable to evidence replay attacks
  • Need for standardized and formally verified attested TLS
• Questions for discussion
Summary

• Intel’s RA-TLS is potentially vulnerable to evidence replay attacks
  • Need for standardized and formally verified attested TLS
• Questions for discussion
  • Is there any *practical* attack if the derive_secret for ‘ms’ is skipped?
Summary

- Intel’s RA-TLS is potentially vulnerable to evidence replay attacks
  - Need for standardized and formally verified attested TLS
- Questions for discussion
  - Is there any *practical* attack if the derive_secret for ‘ms’ is skipped?
  - Is a fix for RA-TLS within pre-HS attestation possible without storing state (evidence)?
Summary

• Intel’s RA-TLS is potentially vulnerable to evidence replay attacks
  • Need for standardized and formally verified attested TLS
• Questions for discussion
  • Is there any *practical* attack if the derive_secret for ‘ms’ is skipped?
  • Is a fix for RA-TLS within pre-HS attestation possible without storing state (evidence)?
  • How to deal with underspecification?


ACK

- Ionut Mihalcea (Arm)
- Yaron Sheffer (Intuit)
- Thore Sommer (Kiel University)
- Carsten Weinhold (Barkhausen Institut)
- Michael Roitzsch (Barkhausen Institut)
- Yogesh Deshpande (Arm)
- Anonymous HCVS reviewer #3