A rollercoaster ride on the formal analysis of attested TLS

Muhammad Usama Sardar[1], Arto Niemi[2], Hannes Tschofenig[3], Thomas Fossati[4]

[1]TU Dresden, Germany

[2]Huawei Technologies, Helsinki, Finland

[3]Siemens, Absam, Austria

[4]Linaro, Lausanne, Switzerland

January 30, 2024

Agenda

Data in transit: Transport protocols

- TLS[1]: widely used protocol

[1] https://datatracker.ietf.org/doc/html/rfc8446

Data in transit: Transport protocols

- TLS[1]: widely used protocol
- Conceptually 2 main protocols:

[1] https://datatracker.ietf.org/doc/html/rfc8446

Data in transit: Transport protocols

- TLS[1]: widely used protocol
- Conceptually 2 main protocols:
  - Handshake

[1] https://datatracker.ietf.org/doc/html/rfc8446

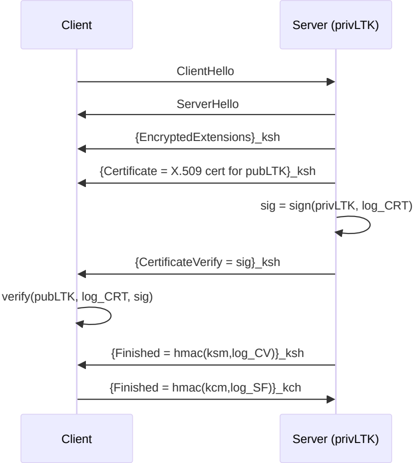
Data in transit: Transport protocols

- TLS[1]: widely used protocol
- Conceptually 2 main protocols:
  - Handshake
  - Record

[1] https://datatracker.ietf.org/doc/html/rfc8446

TLS Handshake Protocol

- Most complex part of TLS
  1. Unauthenticated key exchange (and parameter negotiation)
  2. Authentication (inc. key confirmation)



Problem in TLS

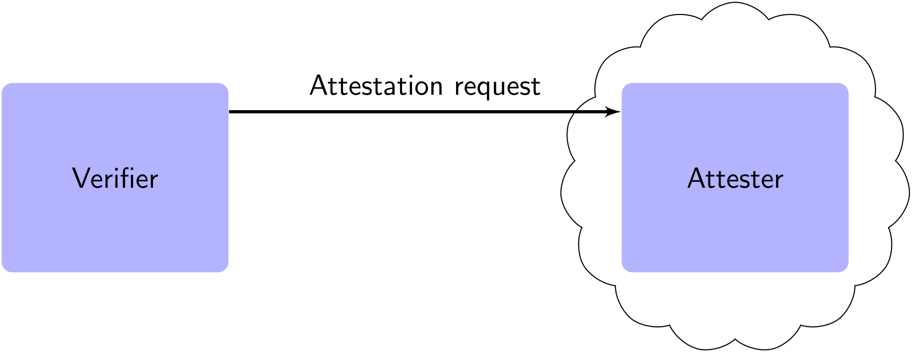
- No validation of security state of endpoint software and platform

Problem in TLS

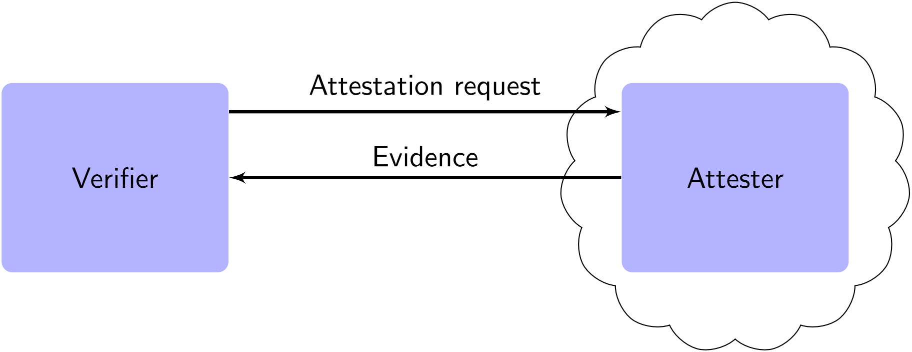
- No validation of security state of endpoint software and platform
- Very complex: exploited at least 15 times

Outline

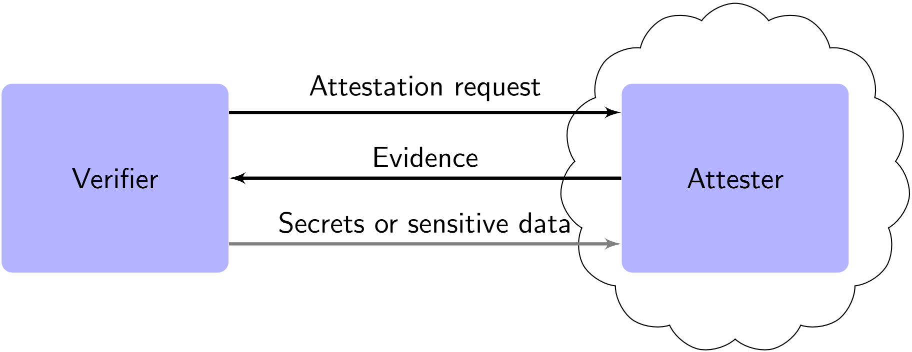
Architecturally-defined Attestation



Architecturally-defined Attestation



Architecturally-defined Attestation



Data in use: Architecturally-defined attestation[2]

- Intel TDX

	Integrity	Freshness	Confidentiality	Authentication
Intel's claimed TCB	×	×	×	×
Our proposed TCB	✓	✓	✓	×

- Arm CCA

Attester	Integrity	Freshness	Confidentiality	Authentication
Platform	✓	×	✓	×
Realm	✓	✓	✓	×

- Problem1: No server authentication
- Problem2: No standard way of implementation

[2] Sardar et al., *Formal Specification and Verification of Architecturally-defined Attestation Mechanisms in Arm CCA and Intel TDX*, 2023.

Outline

Data in transit + Data in use

Transport	TLS/SPDM

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks
- Post-handshake attestation (e.g., SCONE)

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

<table>
<tr><td>Transport</td><td colspan="3">TLS/SPDM</td></tr>
<tr><td rowspan="4">Remote Attestation (arch-def)</td><td colspan="2">Intel</td><td>Arm</td></tr>
<tr><td>SGX</td><td>TDX</td><td>CCA</td></tr>
<tr><td>DCAP

EPID</td><td>DCAP</td><td>PA—RA</td></tr>
</table>

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks
- Post-handshake attestation (e.g., SCONE)
  - Evidence is generated *after* TLS handshake

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks
- Post-handshake attestation (e.g., SCONE)
  - Evidence is generated *after* TLS handshake
  - High latency

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

<table>
<tr><td>Transport</td><td colspan="3">TLS/SPDM</td></tr>
<tr><td rowspan="3">Remote Attestation (arch-def)</td><td colspan="2">Intel</td><td>Arm</td></tr>
<tr><td>SGX</td><td>TDX</td><td>CCA</td></tr>
<tr><td>DCAP

EPID</td><td>DCAP</td><td>PA—RA</td></tr>
</table>

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks
- Post-handshake attestation (e.g., SCONE)
  - Evidence is generated *after* TLS handshake
  - High latency
- Intra-handshake attestation[3]

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks
- Post-handshake attestation (e.g., SCONE)
  - Evidence is generated *after* TLS handshake
  - High latency
- Intra-handshake attestation[3]
  - Evidence is generated *during* TLS handshake

[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Data in transit + Data in use

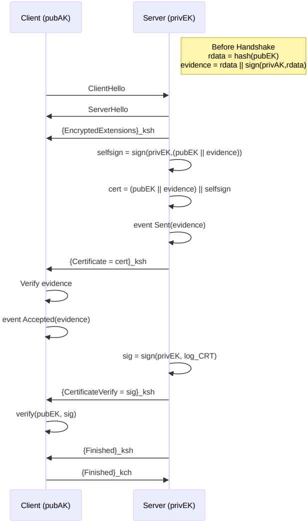
Transport	TLS/SPDM		
Remote Attestation (arch-def)	Intel		Arm
	SGX	TDX	CCA
	DCAP EPID	DCAP	PA—RA

- Idea: compose transport protocol and attestation protocol
- Pre-handshake attestation (e.g., Intel's RA-TLS)
  - Evidence is generated *before* TLS handshake
  - Potentially replay and relay attacks
- Post-handshake attestation (e.g., SCONE)
  - Evidence is generated *after* TLS handshake
  - High latency
- Intra-handshake attestation[3]
  - Evidence is generated *during* TLS handshake
  - Potentially sweet spot

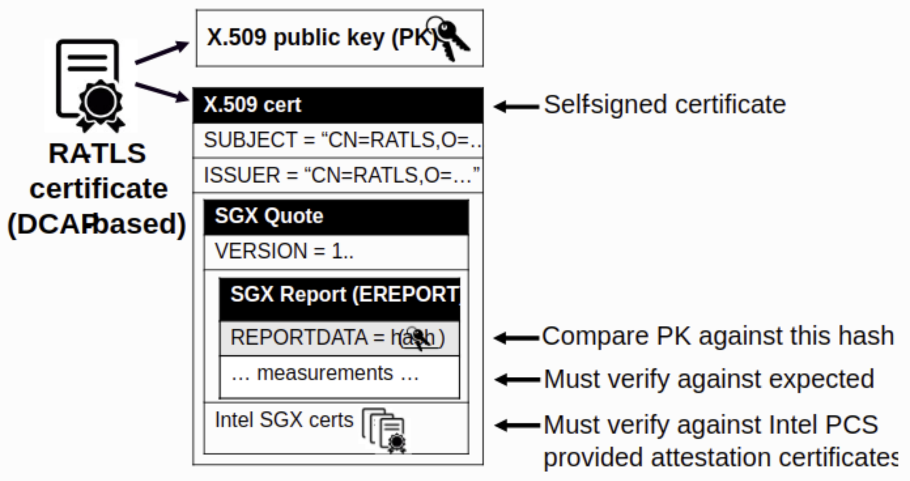
[3] https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

Intel's RA-TLS (simplified) (see Intel-RA-TLSv2.pdf)

- Widely used protocol, e.g., in Gramine, RATS-TLS, Open Enclave Attested TLS, and SGX SDK Attested TLS



Intel's RA-TLS cert[4]



[4] https://gramine.readthedocs.io/en/latest/attestation.html

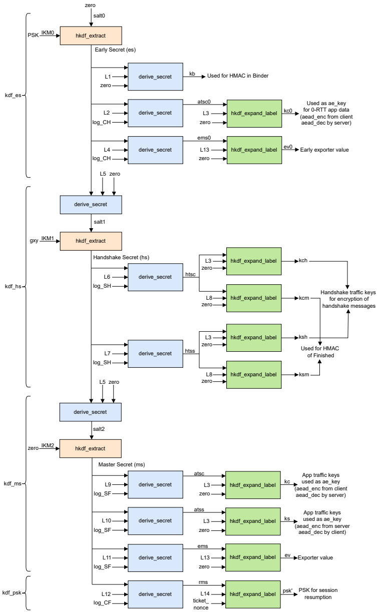
Outline

Key Schedule[5]

```
             0
             |
             v
   PSK ->  HKDF-Extract = Early Secret
             |
             +-----> Derive-Secret(., "ext binder" | "res binder", "")
             |                     = binder_key
             |
             +-----> Derive-Secret(., "c e traffic", ClientHello)
             |                     = client_early_traffic_secret
             |
             +-----> Derive-Secret(., "e exp master", ClientHello)
             |                     = early_exporter_master_secret
             v
       Derive-Secret(., "derived", "")
             |
             v
   (EC)DHE -> HKDF-Extract = Handshake Secret
             |
             +-----> Derive-Secret(., "c hs traffic",
             |                     ClientHello...ServerHello)
             |                     = client_handshake_traffic_secret
             |
             +-----> Derive-Secret(., "s hs traffic",
             |                     ClientHello...ServerHello)
             |                     = server_handshake_traffic_secret
             v
       Derive-Secret(., "derived", "")
             |
             v
   0 -> HKDF-Extract = Master Secret
             |
             +-----> Derive-Secret(., "c ap traffic",
             |                     ClientHello...server Finished)
             |                     = client_application_traffic_secret_0
             |
             +-----> Derive-Secret(., "s ap traffic",
             |                     ClientHello...server Finished)
             |                     = server_application_traffic_secret_0
             |
             +-----> Derive-Secret(., "exp master",
             |                     ClientHello...server Finished)
             |                     = exporter_master_secret
             |
             +-----> Derive-Secret(., "res master",
                                   ClientHello...client Finished)
                                   = resumption_master_secret
```

[5] https://datatracker.ietf.org/doc/html/rfc8446#section-7.1

Key Schedule with 2nd stage (see TLS-KeyDerv2.pdf)



Issue 1[6]

Incorrect implementation of salts for Handshake Secret and Master Secret (draft 20 implementation) #7

Open **muhammad-usama-sardar** opened this issue on Dec 4, 2023 · 0 comments

muhammad-usama-sardar commented on Dec 4, 2023

Salt for Handshake Secret

In ProVerif modeling of draft 20, the salt for Handshake Secret derivation is implemented wrongly:

```
let extra = derive_secret(es,tls13_derived,hash(StrongHash,zero)) in
```

Essentially, instead of implementing `Derive-Secret(es, "derived", "")`, the model implements `Derive-Secret(es, "derived", hash(""))`. Since `Derive-Secret` by definition includes hash over Messages, the above formal model results in an additional iteration of hash.

Hence, in accordance with Sec. 7.1 of draft 20, it should be:

```
let extra = derive_secret(es,tls13_derived,zero) in
```

Salt for Master Secret

Same applies to salt for Master Secret:

```
let extra = derive_secret(hs,tls13_derived,hash(StrongHash,zero)) in
```

which should be

```
let extra = derive_secret(hs,tls13_derived,zero) in
```

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
No branches or pull requ

Notifications
U
You're receiving notificati
thread.

1 participant

[6] https://github.com/Inria-Prosecco/reftls/issues/7

Issue 2[7]

Incorrect derivation of Master Secret (draft 20 implementation) #6

Open **muhammad-usama-sardar** opened this issue on Dec 4, 2023 · 0 comments

muhammad-usama-sardar commented on Dec 4, 2023

In ProVerif modeling of draft 20, the master secret derivation is implemented wrongly:

```
let ms =   hkdf_extract(hs , zero) in
```

Essentially, the model skips the following step shown in the Key Schedule (cf. diagram showing full key derivation schedule on page 88 in Sec. 7.1 of draft 20):
`Derive-Secret(., "derived", "")`

Hence, it should be:

```
let ms =   hkdf_extract(extra , zero) in
```

[7] https://github.com/Inria-Prosecco/reftls/issues/6

TLS WG[8]

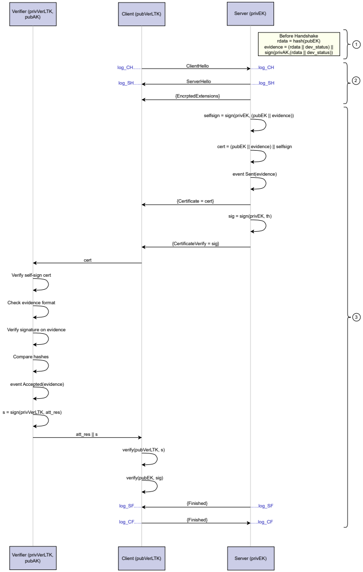
```
Now about the Inria paper that you have mentioned, I am not much
knowledgeable about computational analysis. I understand that it helped
them remove the assumption (that DH group elements do not match the
corresponding labels) in their proof in CryptoVerif but the
corresponding formal analysis in ProVerif in the same paper does not
support this view, i.e., all properties remain the same regardless of
the additional Derive-Secret.

Moreover, the implementation of key hierarchy in draft 20 in ProVerif by
the authors is incorrect [5-6]. For instance, due to a strange reason
and beyond our understanding, the draft 20 implementation does not use
the Derive-Secret for Master Secret [5]. Do you have any
thoughts/opinion on this? The same implementation is being used by other
extensions as a baseline, including Lurk [7].
```

[8] https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPrirj_rkSTYhDo/

Outline

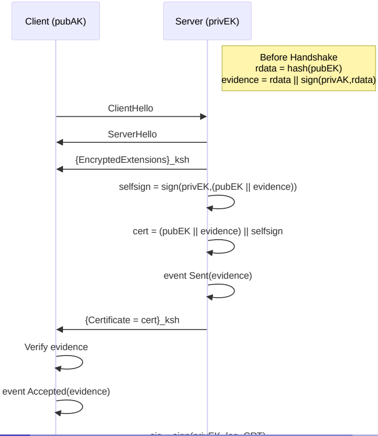
RA-TLS in background check model (Intel-RA-TLSv3.pdf)



Outline

Replay protection of Evidence

$$query\ ev : bitstring;$$
$$inj - event(Accepted(ev)) ==> inj - event(Sent(ev)) \quad (1)$$



Outline

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

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

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
    1. Comments in formal models (best practices)

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
    1. Comments in formal models (best practices)
    2. Validation of formal models
    3. Keep formal verification artifacts up to date (IRTF UFMRG)

---

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools
- Plan

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools
- Plan
  - Formalize the proposed protocol[9]; discuss remaining issues; call for adoption

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools
- Plan
  - Formalize the proposed protocol[9]; discuss remaining issues; call for adoption
  - Compile all issues and discuss at IRTF UFMRG ML/meeting

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools
- Plan
  - Formalize the proposed protocol[9]; discuss remaining issues; call for adoption
  - Compile all issues and discuss at IRTF UFMRG ML/meeting
- Call to action

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools
- Plan
  - Formalize the proposed protocol[9]; discuss remaining issues; call for adoption
  - Compile all issues and discuss at IRTF UFMRG ML/meeting
- Call to action
  - anyone interested?

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Summary

- Intel's RA-TLS is potentially vulnerable to replay attacks
- Need for standardized and formally verified attested TLS
- Lessons learnt
  1. Comments in formal models (best practices)
  2. Validation of formal models
  3. Keep formal verification artifacts up to date (IRTF UFMRG)
  4. Usability of formal tools
- Plan
  - Formalize the proposed protocol[9]; discuss remaining issues; call for adoption
  - Compile all issues and discuss at IRTF UFMRG ML/meeting
- Call to action
  - anyone interested?
  - got someone at your org with expertise?

[9] Tschofenig et al., *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2023.

Key References

Sardar, Muhammad Usama et al. *Formal Specification and Verification of Architecturally-defined Attestation Mechanisms in Arm CCA and Intel TDX*. Nov. 2023. URL: https://www.researchgate.net/publication/375592777_Formal_Specification_and_Verification_of_Architecturally-defined_Attestation_Mechanisms_in_Arm_CCA_and_Intel_TDX.

Tschofenig, Hannes et al. *Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. Internet-Draft draft-fossati-tls-attestation-04. Work in Progress. Internet Engineering Task Force, Oct. 2023. 33 pp. URL: https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/04/.