Formal Analysis of RA-TLS

Muhammad Usama Sardar

TU Dresden, Germany

March 18, 2024
Background: Problem in TLS

- No validation of security state of endpoint software and platform
  - Need a more comprehensive set of security metrics in some use cases, e.g., CC
- Very complex: at least 15 different exploits
  - Is all complexity (e.g., of key schedule) justified?

![Diagram of TLS handshake with various attestations and signing of evidence stages.]
**Approach**

- Spec. docs
- Implementation
- Community input
- Security Goals

**Formal Model**

- ProVerif
- Properties

**Sanity checks**

- Update
- Validation

**Formal Analysis**

- Proof
- Attack
- Could not prove
- No result

**Security Goals**
Validation Framework

Inria artifacts

TLS 1.3 Specs

= = ?

Success Failure

Yes No

Inria artifacts

TLS 1.3 Specs

= = ?

Success

Failure

Yes No
A trace has been found.

Honest Process

Attacker

Beginning of process TestKeySch

\( \sim M = \text{Success} \)

\( \sim M_1 = \text{Success} \)

\( \sim M_2 = \text{Failure} \)

\( \sim M_3 = \text{Success} \)

\( \sim M_4 = \text{Failure} \)

\( \sim M_5 = \text{Failure} \)

\( \sim M_6 = \text{Failure} \)

\( \sim M_7 = \text{Failure} \)

\( \sim M_8 = \text{Failure} \)

\( \sim M_9 = \text{Failure} \)

\( \sim M_{10} = \text{Failure} \)

\( \sim M_{11} = \text{Failure} \)

\( \sim M_{12} = \text{Failure} \)

\( \{240\} \text{event test} \)

\( \sim M_{13} = \text{Failure} \)

The attacker has the message \( \sim M_2 = \text{Failure} \)
Incorrect implementation of salts for Handshake Secret and Master Secret (draft 20 implementation) #7

Muhammad Usama Sardar (TUD)
Incorrect derivation of Master Secret (draft 20 implementation) #6

Muhammad Usama Sardar commented on Dec 4, 2023

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

```proverif
define ms = hkdf_extract(hash, 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):

```proverif
define Derive-Secret(source, derived, tag) in
```

Hence, it should be:

```proverif
define ms = hkdf_extract(extra, zero) in
```

---

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

---

https://mailarchive.ietf.org/arch/msg/tls/ZGmyHwTYh2iPwPirj_rkSTYhDo/