Automated Analysis of TLS 1.3
0-RTT, Resumption and Delayed Authentication

IEEE S&P, 24/05/2016

Cas Cremers
Marko Horvat
Sam Scott
Thyla van der Merwe

Mozilla
Where our work fits in

Program verification

Provable security

Formal methods

Dowling et al. [draft-05]
Kohlweiss et al. [draft-05]
Krawczyk and Wee [OPTLS]
Dowling et al. [draft-10]
Motivation and approach

TLS 1.3 designed to be more efficient than TLS 1.2:
- 0-RTT handshake mode.
- PSK mode.
- Delayed client authentication.

**Our goal**

Improve the security of TLS 1.3 by analysing the specification using state-of-the-art formal analysis methods.

We focus on interaction attacks:
- Perfect cryptography assumption.
- Dolev-Yao attacker.
Building a model. Client state machine

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication
Unbounded number of concurrent sessions
We encoded our model for use in the Tamarin prover:

- State-of-the-art tool for automated protocol analysis.
  - Loops.
  - Branches.
  - Symbolic Diffie-Hellman.

- However, requires considerable user interaction for very complex models.
We verified the core properties of TLS 1.3 revision 10 as an authenticated key exchange protocol:

- Secrecy of session keys.
  - Forward secrecy included.
- Unilateral and mutual authentication.
- Integrity of handshake messages.
We verified the core properties of TLS 1.3 revision 10 as an authenticated key exchange protocol:

- Secrecy of session keys.
  - Forward secrecy included.
- Unilateral and mutual authentication.
- Integrity of handshake messages.

Is it safe to include delayed client authentication in revision 10?
Attacking client authentication

ClientHello
Receive ServerHello/Finished + Send ClientFinished

Client authentication

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe
Automated Analysis of TLS 1.3
Attacking client authentication

Alice (Client)

Charlie (evil.com)
Attacking client authentication
Attacking client authentication

Alice (Client) → Connect to evil.com → ... → Establish PSK → Charlie (evil.com)
Attacking client authentication

Alice (Client) → Connect to evil.com → Charlie (evil.com) → Establish PSK

PSK₁ → Alice (Client) ← PSK₁
Attacking client authentication

Alice (Client)

Connect to evil.com

... Establish PSK

PSK₁

Charlie (evil.com)

PSK₁

Bob (mybank.com)
Attacking client authentication

Alice (Client) → Connect to evil.com → Charlie (evil.com) → Connect to mybank.com → Bob (mybank.com)

PSK₁ → Establish PSK → PSK₁
Attacking client authentication

Alice (Client) → Connect to evil.com → Charlie (evil.com) → Connect to mybank.com → Bob (mybank.com)

PSK₁ → Establish PSK → PSK₁ → PSK₂ → Establish PSK → PSK₂
Attacking client authentication

Alice (Client) → Connect to evil.com → Establish PSK → PSK₁ ≠ PSK₂ → Connect to mybank.com → Establish PSK → Bob (mybank.com)
Attacking client authentication

Alice (Client) \[
\text{Session resumption}
\]
\[
\text{client_random} = nc
\]
PSK\(_1\)

Charlie (evil.com)

PSK\(_1\) PSK\(_2\)

Bob (mybank.com)

PSK\(_2\)

\[
\text{session_hash} = H(nc)
\]
Attacking client authentication

Alice (Client) \[\text{Session resumption}\] PSK_1

Charlie (evil.com) \[\text{Session resumption}\] PSK_1 PSK_2

Bob (mybank.com)

\[\text{session_hash} = H(nc)\]

\[\text{session_hash} = H(nc)\]
Attacking client authentication

Alice
(Client)

Session resumption
client_random = nc

PSK₁

Charlie
(evil.com)

Session resumption
client_random = nc

PSK₁

server_random = ns

PSK₂

Bob
(mybank.com)

session_hash = H(nc)

session_hash = H(nc ns)
Attacking client authentication

Alice (Client)  
PSK₁  

Session resumption  
client_random = nc  
server_random = ns

Charlie (evil.com)  
PSK₁  
PSK₂

Session resumption  
client_random = nc  
server_random = ns

Bob (mybank.com)  
PSK₂

\[
\text{session_hash} = H(nc \ ns \ ...) \\
\text{session_hash} = H(nc \ ns \ ...)
\]
Attacking client authentication

Alice (Client) $\xrightarrow{\text{ClientFinished}_1} \text{Charlie (evil.com)}$

Keys derived from $\text{PSK}_1$

$\text{PSK}_1 \xrightarrow{} \text{PSK}_1 \ xrightarrow{} \text{PSK}_2$

Bob (mybank.com)

\[
\text{session_hash} = H(nc \ ns \ ...)
\]

\[
\text{session_hash} = H(nc \ ns \ ...)
\]
Attacking client authentication

Alice (Client) \hspace{5mm} \xrightarrow{\text{ClientFinished}_1} \hspace{5mm} Charlie (evil.com) \hspace{5mm} \xrightarrow{\text{ClientFinished}_2} \hspace{5mm} Bob (mybank.com)

Keys derived from \( \text{PSK}_1 \) \hspace{5mm} Keys derived from \( \text{PSK}_2 \)

\[
\text{session_hash} = H(nc \ ns \ \ldots )
\]

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe  
Automated Analysis of TLS 1.3
Attacking client authentication

Alice (Client) \hspace{2cm} Charlie (evil.com) \hspace{2cm} Bob (mybank.com)

Request authentication

\[ \text{session\_hash} = H(nc\ ns\ ...) \]

\[ \text{session\_hash} = H(nc\ ns\ ...) \]
Attacking client authentication

Alice (Client) → Request authentication → Charlie (evil.com) → Request authentication → Bob (mybank.com)

session_hash = H(nc ns ...)

session_hash = H(nc ns ...)
Attacking client authentication

Alice (Client)

Request authentication

Client authentication

sign_{sk_A}(\text{session_hash}, \text{cert}_A, \ldots)

Bob (mybank.com)

Request authentication

session_hash = H(nc \ ns \ldots)

session_hash = H(nc \ ns \ldots)
Attacking client authentication

Alice
(Client)

Request authentication

Client authentication

\[ \text{sign}_{sk_A}(session\_hash, cert_A, \ldots) \]

Bob
(mybank.com)

Request authentication

Client authentication

\[ \text{sign}_{sk_A}(session\_hash, cert_A, \ldots) \]

session_hash = \( H(nc\ ns\ \ldots) \)

session_hash = \( H(nc\ ns\ \ldots) \)
Attacking client authentication

Alice (Client)

Charlie (evil.com)

Bob (mybank.com)

Give Charlie all my money!

Sure thing, Alice.
Disclosure

- We posted analysis results and attack to TLS mailing list end-October 2015.

  “Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern.”
We posted analysis results and attack to TLS mailing list end-October 2015.

“Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern.”

“Thanks for posting this. It’s great to see people doing real formal analysis of the TLS 1.3 draft; this is really helpful in guiding the design.”
We posted analysis results and attack to TLS mailing list end-October 2015.

“Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern.”

“Thanks for posting this. It’s great to see people doing real formal analysis of the TLS 1.3 draft; this is really helpful in guiding the design.”

“This result motivates and confirms the need to modify the handshake hashes to contain the server Finished when we add post-handshake authentication...”
Attack cause and mitigation

- Attack shows initial proposal for delayed client authentication incomplete.
- Highlights strict necessity of binding client signatures to server certificate.
- Working group proposed to include transcript to bind them to sessions.
- This proposal was merged in revision 11, which prevents our attack.
Conclusions

- First comprehensive analysis of the new TLS 1.3 modes and their interaction.
- This story has a happy ending:
  - Revision 10 was successfully verified.
  - Tamarin was used to find an interaction attack on delayed authentication.
  - Proposed fix verified and included in revision 11.
- Future work: Update model and verify revision 13.
- Our work is part of the larger, concerted effort of different approaches to hardening TLS 1.3.

Authors:

Cas Cremers
cas.cremers@cs.ox.ac.uk

Marko Horvat
mhorvat@mpi-sws.org

Sam Scott
sam.scott.2012@live.rhul.ac.uk

Thyla van der Merwe
thyla.vandermerwe.2012@live.rhul.ac.uk