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

TRON, 21 February 2016
What we did (nutshell)

- We built a symbolic model of the TLS 1.3 specification - draft 10
- We wanted to verify the **main** properties of TLS 1.3 as an authenticated key exchange protocol
  - secrecy of session keys
  - unilateral (mutual) authentication
- We extended our draft 10 model to include the delayed authentication mechanism
- We found a potential attack - disclosed this to the IETF TLS WG
- We have updated our model to draft 11 of the specification and are ‘half-way’ through proving
Where our work fits in...

Program verification

Ongoing work – later talks

Provable security

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

Formal methods

We are here!

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe

Automated Analysis of TLS 1.3
TLS 1.3

(a) Initial (EC)DHE handshake

(b) 0-RTT handshake

(c) PSK-resumption handshake (+ PSK-DHE)
Look at the full interaction of all these components!
A symbolic model allows us to examine the logical soundness of the protocol.

We systematically specify and hope to exclude a class of attacks - ‘logical’ attacks (think Triple Handshake for TLS 1.2).

We assume black-box cryptography.

We analyse the specification only.
Using Tamarin

Tool details available at:
http://www.infsec.ethz.ch/research/software/tamarin.html
We built our model for use in the Tamarin prover

- Automated tool for protocol analysis
- Good symbolic Diffie-Hellman support
- Considers an unbounded number of parties/handshakes
Using Tamarin

We built our model for use in the Tamarin prover

- Automated tool for protocol analysis
- Good symbolic Diffie-Hellman support
- Considers an unbounded number of parties/handshakes

How does it work?

- For simple models/properties, can prove automatically
- Complex models require more user interaction
- A proof shows that a property holds in all possible combinations of client, server, and adversary behaviours
Using Tamarin

We built our model for use in the Tamarin prover

- Automated tool for protocol analysis
- Good symbolic Diffie-Hellman support
- Considers an unbounded number of parties/handshakes

How does it work?

- For simple models/properties, can prove automatically
- Complex models require more user interaction
- A proof shows that a property holds in all possible combinations of client, server, and adversary behaviours

Tamarin is good because

- We can precisely model the TLS state machines
- We can accurately capture security properties
Step 1: Building a model
Step 1: Building a model

- Encode honest party and adversary actions as Tamarin ‘rules’
- For honest clients and servers rules correspond to flights of messages
- Rules transition the protocol from one state to the next
Step 1: Building a model

**Diagram Description:**
- **Start:** $c_0$
- **Step 1:** $c_1 - psk$
- **Step 2a:** $c_2$
- **Step 3:** $c_3$

- **Client Hello:** $C_{1 - psk}$
- **Receive ServerHello/Finished + Send ClientFinished:** $C_{2 - psk, DHE}$
- **Client authentication:** $C_{2 - PSK, KC}$
- **C_1 retry:** $C_{1 -PSK}$
- **C_1 KC:** $C_{1 -KC}$
- **C_1 retry:** $C_{1 -KC, Auth}$
- **C_1 PSK:** $C_{1 - PSK}$
- **C_2 KC:** $C_{2 -KC}$
- **C_2 PSK:** $C_{2 -PSK}$
- **C_2 NoAuth:** $C_{2 -NoAuth}$
- **C_2 Auth:** $C_{2 -Auth}$
- **C_3 NST:** $C_{3 -NST}$
- **C_send:** $C_{send}$
- **C_recv:** $C_{recv}$
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

C_1
C_1.PSK
C_1.dhe
C_1.psk
C_2
C_2.PSK
C_2.PSK.DHE
C_2.KC
C_3
C_3.NST
C_1.KC.Auth
C_1.retry
C_2.Auth
C_2.NoAuth
C_3
C_send
C_recv

Client authentication

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe
Automated Analysis of TLS 1.3
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication

C.2_NoAuth

C.2_Auth

C.3

C.send

C.recv

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe
Automated Analysis of TLS 1.3
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication
Step 1: Building a model

```
ClientHello
Receive ServerHello/Finished + Send ClientFinished
```

Client authentication
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication

C_1 PSK

C_1 PSK DHE

C_2 PSK

C_2 PSK DHE

C_1KC

C_2 KC

C_2 _NoAuth

C_3 NST

C_2 _Auth

C_2 _NoAuth

C_3 _NST

C_1 retry

C_1 retry

C_2 _ Auth

C_2 _NoAuth

C_3 _NST

C send

C recv
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe
Automated Analysis of TLS 1.3
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe
Automated Analysis of TLS 1.3
Step 1: Building a model

ClientHello

Receive ServerHello/Finished + Send ClientFinished

Client authentication
Step 1: Building a model

rule C_1:
  let
    // Default C1 values
    tid = ~nc

    // Client Hello
    C = $C
    nc = ~nc
    pc = $pc
    S = $S

    // Client Key Share
    ga = 'g' ^ ~a

    messages = <nc, pc, ga>
  in
    [ Fr(nc) , Fr(~a) ]
  --> [ C1(tid) , Start(tid, C, 'client') , Running(C, S, 'client', nc) , DH(C, ~a) ]
    [ St_C_1_init(tid, C, nc, pc, S, ~a, messages, 'no_auth') , Out(<C, nc, pc, ga>) ]
Step 1: Building a model
TLS 1.3 goals include:

- unilateral authentication of the server (mandatory)
- mutual authentication (optional)
- confidentiality and perfect forward secrecy of session keys

We have Dolev-Yao style adversary that has the ability to compromise long-term keys
### Step 2: Encoding security properties

<table>
<thead>
<tr>
<th>Security property</th>
<th>Covered by analysis</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unilateral authentication (server)</td>
<td>Y</td>
<td>D.1.1</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Y</td>
<td>D.1.1</td>
</tr>
<tr>
<td>Confidentiality of ephemeral secret</td>
<td>Y</td>
<td>D.1.1</td>
</tr>
<tr>
<td>Confidentiality of static secret</td>
<td>Y</td>
<td>D.1.1</td>
</tr>
<tr>
<td>Perfect forward secrecy</td>
<td>Y</td>
<td>D.1.1.1</td>
</tr>
<tr>
<td>Integrity of handshake messages</td>
<td>Y</td>
<td>D.1.3</td>
</tr>
<tr>
<td>Protection of application data</td>
<td>N</td>
<td>D.2</td>
</tr>
<tr>
<td>Denial of service</td>
<td>N</td>
<td>D.3</td>
</tr>
<tr>
<td>Version rollback</td>
<td>N</td>
<td>D.1.2</td>
</tr>
</tbody>
</table>

Table: TLS 1.3 rev 10 properties
Step 2: Encoding security properties

secret_session_keys:
(1) "All actor peer role k #i.
(2) SessionKey(actor, peer, role, <k, 'authenticated'>)@i
(3) & not ((Ex #r. RevLtk(peer)@r & #r < #i)
    |(Ex #r. RevLtk(actor)@r & #r < #i))
(4) ==> not Ex #j. KU(k)@j"

This says...

- for all possible values of variables on the first line (1),
- if key k is accepted at time point i (2), and
- the adversary has not revealed the long term keys of the actor or the peer before the key is accepted (3),
- then the adversary cannot derive the key (4).
secret_session_keys:
(1) "All actor peer role k #i.
(2) SessionKey(actor, peer, role, <k, 'authenticated'>)@i
(3) & not ((Ex #r. RevLtk(peer)@r & #r < #i)
| (Ex #r. RevLtk(actor)@r & #r < #i))
(4) ==> not Ex #j. KU(k)@j"

This says...

- for all possible values of variables on the first line (1),
- if key k is accepted at time point i (2), and
- the adversary has not revealed the long term keys of the actor or the peer before the key is accepted (3),
- then the adversary cannot derive the key (4).

Want to show that this holds for all combinations of client, server and adversary behaviours - ALL traces
Step 3: Proving security properties

...and so on

What can the adversary do?

What can the adversary do?

eventually will boil down to needing to break DH

C2_No_Auth

S2

C2_Auth

S2_Auth

SessionKey(....)
Step 3: Proving security properties

- Not a straightforward application of Tamarin
  - several man-months of work
  - specification a moving target
  - updating takes time, can be error-prone

- Need an intimate knowledge of the protocol - high degree of interaction with the tool in some cases

- Not a push-button analysis

- We have 45 auxiliary lemmas
Step 3: Proving security properties

- secret_session_keys
- es_basic
- psk_dhe_es_basic
- kc_origin
- psk_helper
- psk_helper_client
- psk_helper_server
- key_deriv_rs
- key_deriv_hkeys
- key_deriv_hkeyc
- psk_basic
- + many more

HINTS:
- tid_invariant
- one_start_tid
- + many more

HELPERS (PROOF SHORTCUTS):
- key_deriv_rs
- key_deriv_hkeys
- key_deriv_hkeyc
- psk_basic
- + many more

SECRET SESSION KEYS

PSK HELPERS

ES_BASIC

K C ORIGIN

SS_BASIC

TID INVARIANT

ONE_START_TID

+ MANY MORE
We verified the main properties of TLS 1.3 revision 10 as an authenticated key exchange protocol:

- Secrecy of session keys
  - holds for both client and server
  - forward secrecy
- Mutual authentication
Attacking client authentication (revision 10+)

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) Connect to evil.com ... Establish PSK

PSK₁

Charlie (evil.com) Connect to mybank.com ...

PSK₁ PSK₂ Establish PSK

Bob (mybank.com)
Attacking client authentication

Alice (Client) \quad \text{Connect to evil.com} \quad \text{…} \quad \text{Connect to mybank.com} \quad \text{…} \quad \text{Bob (mybank.com)}

PSK$_1$ \quad \text{Establish PSK} \quad \text{PSK$_1$} \neq \text{PSK$_2$} \quad \text{Establish PSK} \quad \text{PSK$_2$}

psk$_{id}$ = psk$_{id}$
Attacking client authentication

Alice (Client) \rightarrow \text{Session resumption} \rightarrow \text{Charlie (evil.com)} \rightarrow \text{Bob (mybank.com)}

PSK_1 \rightarrow \text{client_random} = nc \rightarrow PSK_1, PSK_2

\text{session_hash} = H(nc)

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

Alice (Client) \[\text{Session resumption} \Rightarrow \text{Charlie (evil.com)}\]

\[\text{client_random} = nc\]

\[\text{PSK}_1 \Rightarrow \text{PSK}_1 \quad \text{PSK}_2\]

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

Bob (mybank.com) \[\text{Session resumption} \Rightarrow \text{Session resumption}\]

\[\text{client_random} = nc\]

\[\text{PSK}_2\]
Attacking client authentication

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

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

Bob (mybank.com)

\[ \text{client_random} = nc \]

\[ \text{client_random} = nc \]

\[ \text{server_random} = ns \]

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

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

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

$\text{client\_random} = nc$

server\_random = ns

$\text{PSK}_1$

Charlie (evil.com)

PSK$_1$ PSK$_2$

$\text{Session resumption}$

$\text{client\_random} = nc$

server\_random = ns

$\text{PSK}_1$

Bob (mybank.com)

$\text{PSK}_2$

$session\_hash = H(nc\ ns\ \ldots)$

$session\_hash = H(nc\ ns\ \ldots)$
Attacking client authentication

Alice (Client)

Bob (mybank.com)

ClientFinished$_1$

Keys derived from PSK$_1$

PSK$_1$

PSK$_2$

Charlie (evil.com)

session$_\text{hash} = H(nc\ ns\ \ldots)$

PSK$_1$

session$_\text{hash} = H(nc\ ns\ \ldots)$

PSK$_2$
Attacking client authentication

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

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

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

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

Alice (Client) \quad \text{Request authentication} \quad \text{Charlie (evil.com)} \quad \text{Request authentication} \quad \text{Bob (mybank.com)}

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

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

Alice (Client)

Request authentication

Client authentication

sign_{sk_A}(session_hash, cert_A, ...)

Bob (mybank.com)

Request authentication

session_hash = H(nc ns ...)

session_hash = H(nc ns ...)

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe

Automated Analysis of TLS 1.3
Attacking client authentication

Alice
(Client)

Request authentication

Client authentication

sign_{sk_A}(session_hash, cert_A, ...)

session_hash = H(nc ns ...)

Charlie
(evil.com)

Request authentication

Client authentication

sign_{sk_A}(session_hash, cert_A, ...)

session_hash = H(nc ns ...)

Bob
(mybank.com)
Attacking client authentication

Alice (Client)

Charlie (evil.com)

Give Charlie all my money!

Sure thing, Alice.

Bob (mybank.com)
Prime example of an attack that can arise because of the interaction of modes

No binding between the client signature and session for which it is intended

Complicated to find

Example of the positive interaction of analysis and design

Communicated this to the IETF TLS Working Group...
Dear all,

We [1] are in the process of performing an automated symbolic analysis of the TLS 1.3 specification draft (revision 10) using the Tamarin prover [2], which is a tool for automated security protocol analysis.

While revision 10 does not yet appear to permit certificate-based client authentication in PSK (and in particular resumption using PSK), we modelled what we believe is the intended functionality. By enabling client authentication either in the initial handshake, or with a post-handshake signature over the handshake hash, our Tamarin analysis finds an attack. The result is a complete breakage of client authentication, as the attacker can impersonate a client when communicating with a server:

Suppose a client Alice performs an initial handshake with Charlie. Charlie, masquerading as Alice, subsequently performs a handshake with Bob. Following a PSK resumption, Bob requests authentication from Charlie (impersonating Alice). Charlie then requests authentication from Alice, and the returned signature will also be a valid signature for the session with Bob.
“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.”
“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.”
“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 as is done in PR#316...”
“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 as is done in PR#316…”

Although already included in PR316, the attack highlights strict necessity of including the Finished message in the session hash and draft 11 does this - creates a binding between signature and the session
Where we stand

- We show that draft 10 meets the goals of authenticated key exchange, and the changes in draft 11 appear to address the attack.
- First comprehensive analysis of the interaction of the new TLS 1.3 modes:
  - we confirmed the base design is solid
  - prevented a potential weakness
- Draft 11...
Where we stand

Cas Cremers, Marko Horvat, Sam Scott, Thyla van der Merwe

Automated Analysis of TLS 1.3
Where we stand

Half-way through...
Making changes

- The model can be updated to accommodate changes
- Additions can complicate the analysis (adding loops)
- Reductions/simplifications are easier to handle
- Changes mean more man-hours
Making changes

- The model can be updated to accommodate changes
- Additions can complicate the analysis (adding loops)
- Reductions/simplifications are easier to handle
- Changes mean more man-hours

http://tls13tamarin.github.io/TLS13Tamarin/

Authors:

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

Marko Horvat
marko.horvat@cs.ox.ac.uk

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

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