A Comprehensive Symbolic Analysis of TLS 1.3

David Calavera

CTO, Netlify

- Motivation
- What's new in TLS 1.3
- Modeling the protocol
- Encoding the threat model
- Analysis and Results
- Conclusion

Motivation

How does the internet work?

Welcome to my favourite job interview question!

Until TLS 1.2, all modifications have been retroactive

All Internet traffic is going to be encrypted sooner rather than later

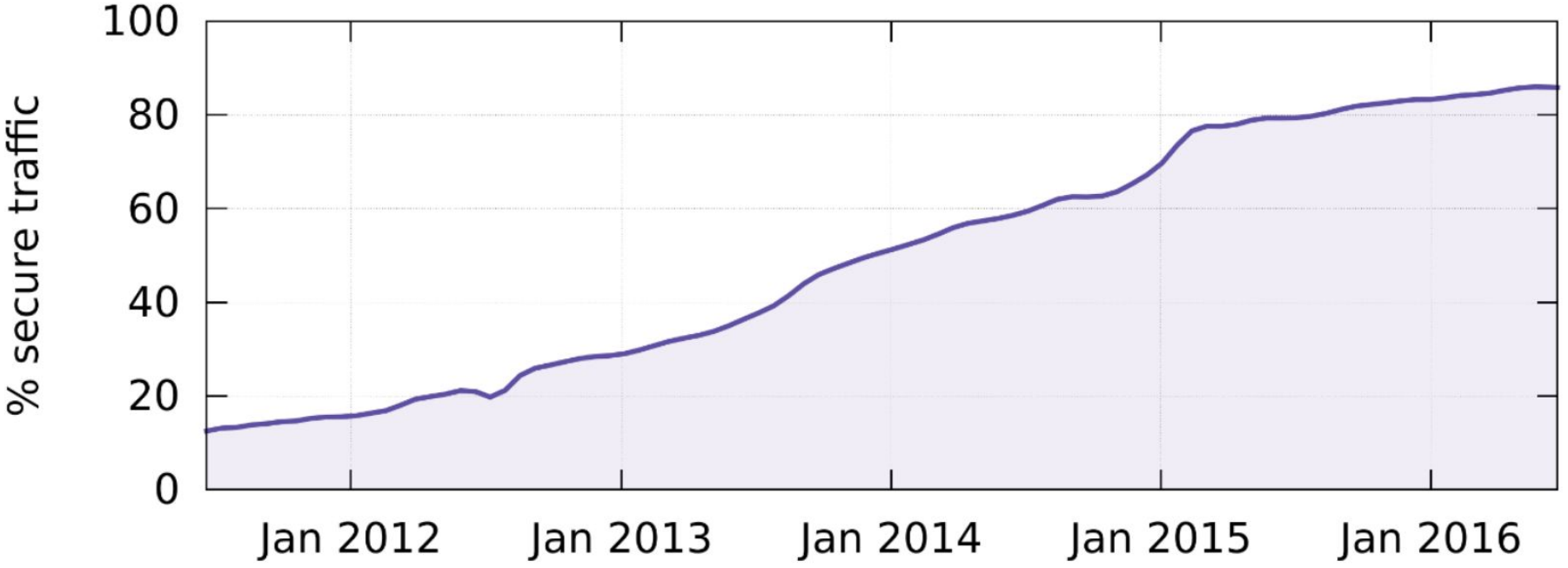
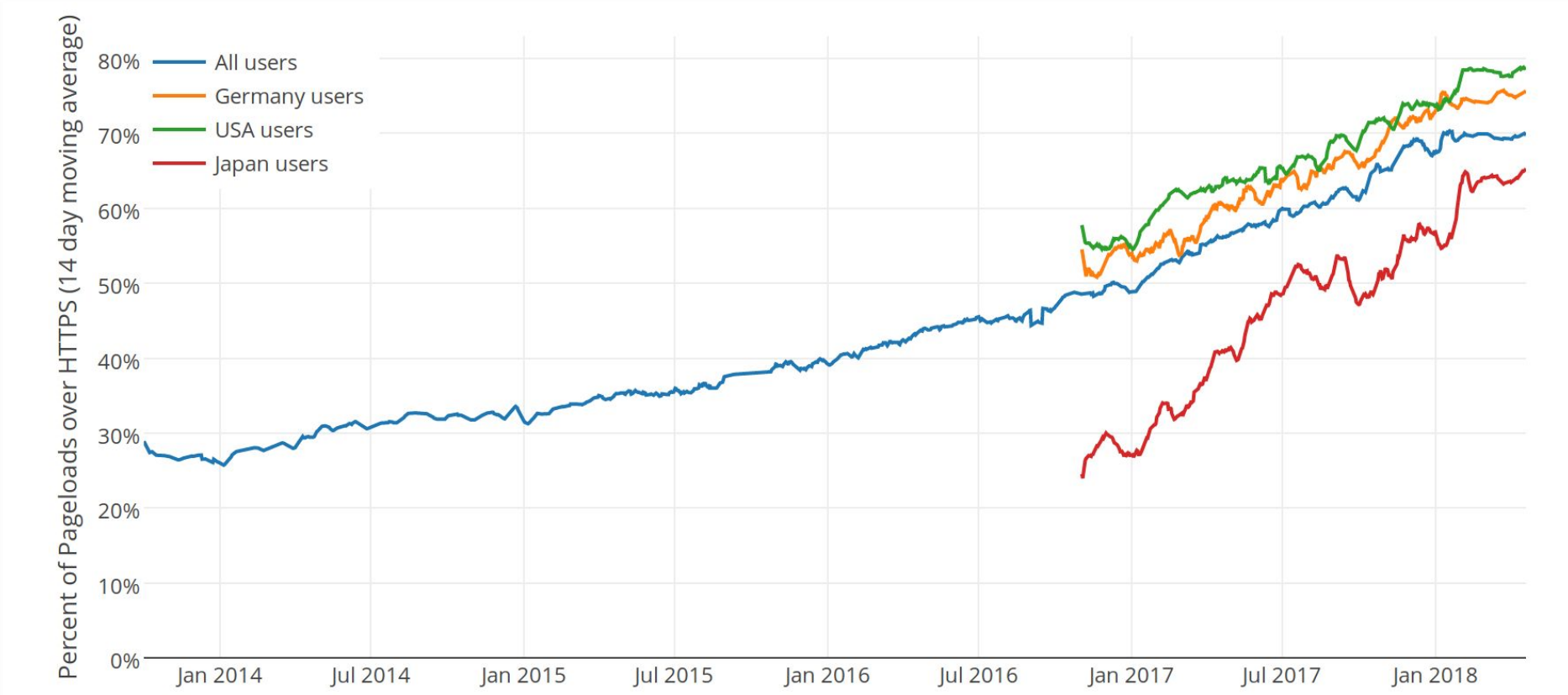


Figure 3: Increase in secure web traffic to Google's front-end servers.

Percentage of Web Pages Loaded by Firefox Using HTTPS

(14-day moving average, source: Firefox Telemetry)



The QUIC Transport Protocol

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46403.pdf

Analysis prior to deployment

What's new in TLS 1.3

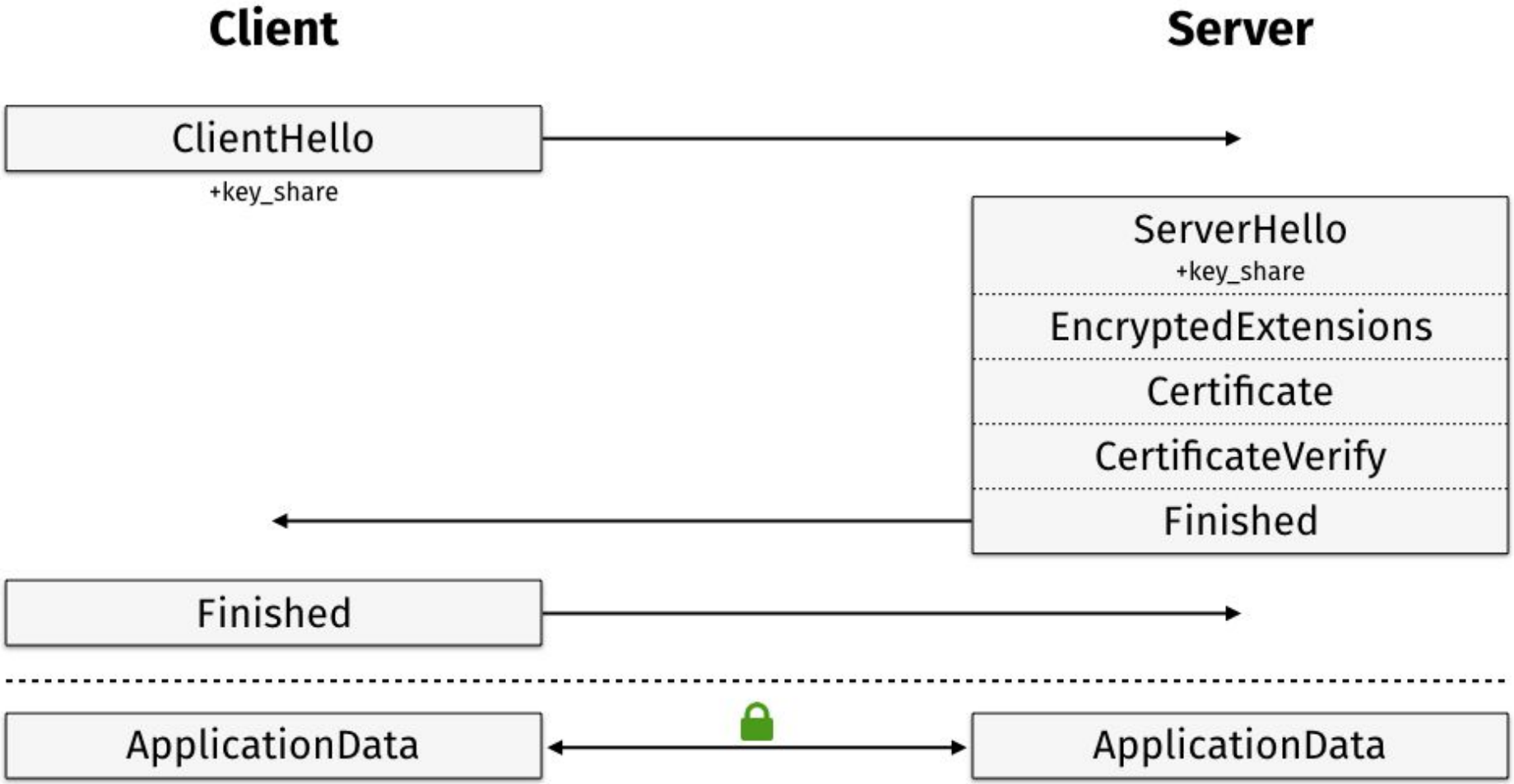
"TLS 1.3 has three key exchange modes, namely, Diffie–Hellman exchange (DHE), pre-shared key (PSK) exchange, and PSK coupled with DHE."

"TLS 1.3 has three post-handshake mechanisms
covering traffic key updates, post-handshake client authentication, and the sending of new session tickets (NSTs) for subsequent re-sumption via a PSK."

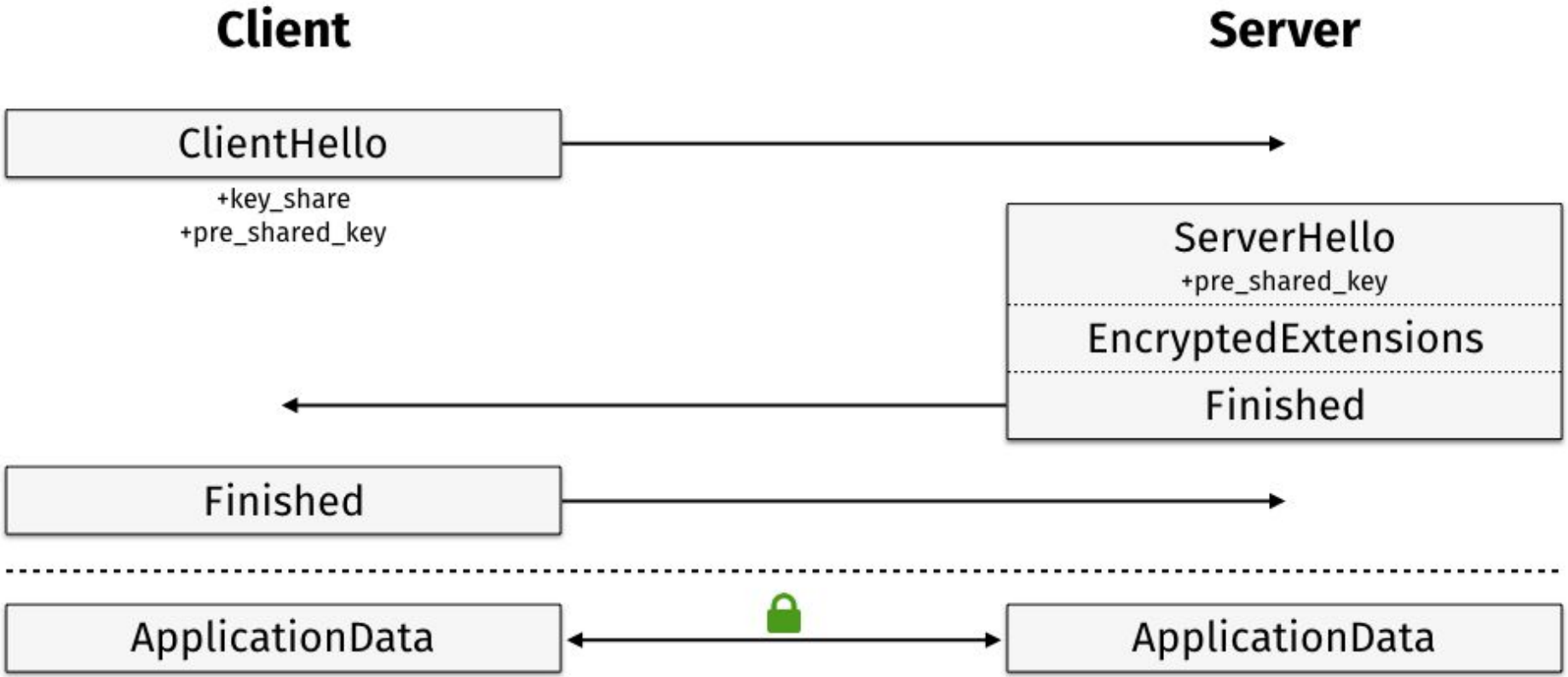
New Mechanisms

Full Handshake

* At worst

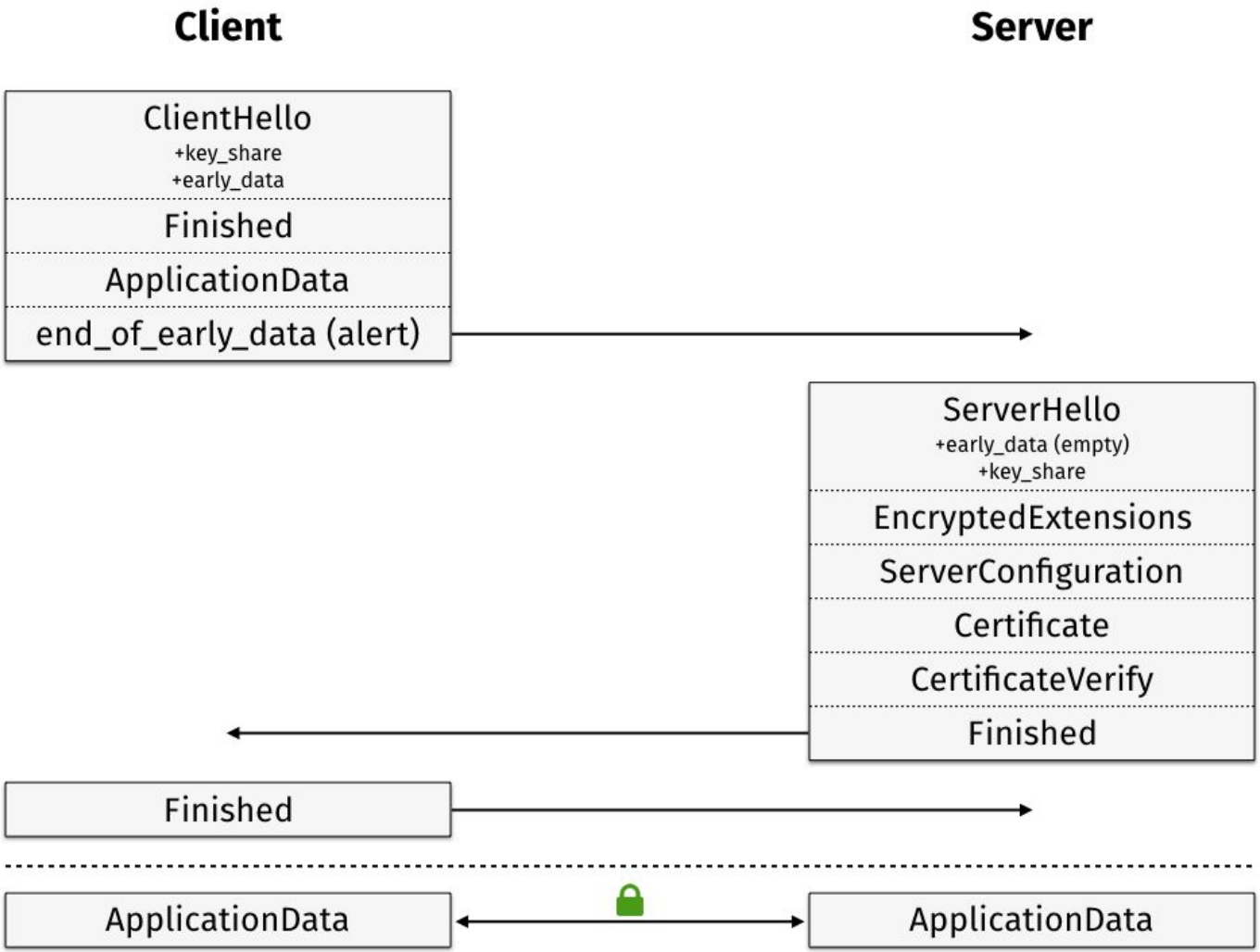

Client
Server
ClientHello
+key_share
ServerHello
+key_share
EncryptedExtensions
Certificate
CertificateVerify
Finished
Finished
ApplicationData
ApplicationData

Session resumption


Client
Server
ClientHello
+key_share
+pre_shared_key
ServerHello
+pre_shared_key
EncryptedExtensions
Finished
Finished
ApplicationData
ApplicationData

0 Round-Trip Time

* At best


Client
Server
ClientHello
+key_share
+early_data
Finished
ApplicationData
end_of_early_data (alert)
ServerHello
+early_data (empty)
+key_share
EncryptedExtensions
ServerConfiguration
Certificate
CertificateVerify
Finished
Finished
ApplicationData
ApplicationData

Stated Goals And Security Properties

- Establishing the same keys
- Secret of the session keys
- Peer authentication
- Uniqueness of session keys

- Downgrade protection*
- Perfect Forward Secrecy
- Key compromise resistance
- Protection of endpoint identities*

Modeling the protocol

The Tamarin Prover is a symbolic modelling analysis tool for security protocols

https://tamarin-prover.github.io

```
rule Register_pk:
    [ Fr(~ltk) ]
  -->
    [ !Ltk($A, ~ltk), !Pk($A, pk(~ltk)) ]
```

```
lemma Client_session_key_secrecy:
  " /* It cannot be that a  */
    not(
      Ex S k #i #j.
        /* client has set up a session key 'k' with a server'S' */
        SessKeyC(S, k) @ #i
        /* and the adversary knows 'k' */
      & K(k) @ #j
        /* without having performed a long-term key reveal on 'S'. */
      & not(Ex #r. LtkReveal(S) @ r)
    )
  "
```

Closely modeling the specification

```
rule send:
[ SendStream(~tid, $actor, $peer, auth_status, app_key_out), Fr(~data)]
--
[ Send(~tid), SendData(~tid, $actor, $peer, auth_status, ~data)]
->
[ SendStream(~tid, $actor, $peer, auth_status, app_key_out),
  Out(senc{data_record(~data)}app_key_out)]
```

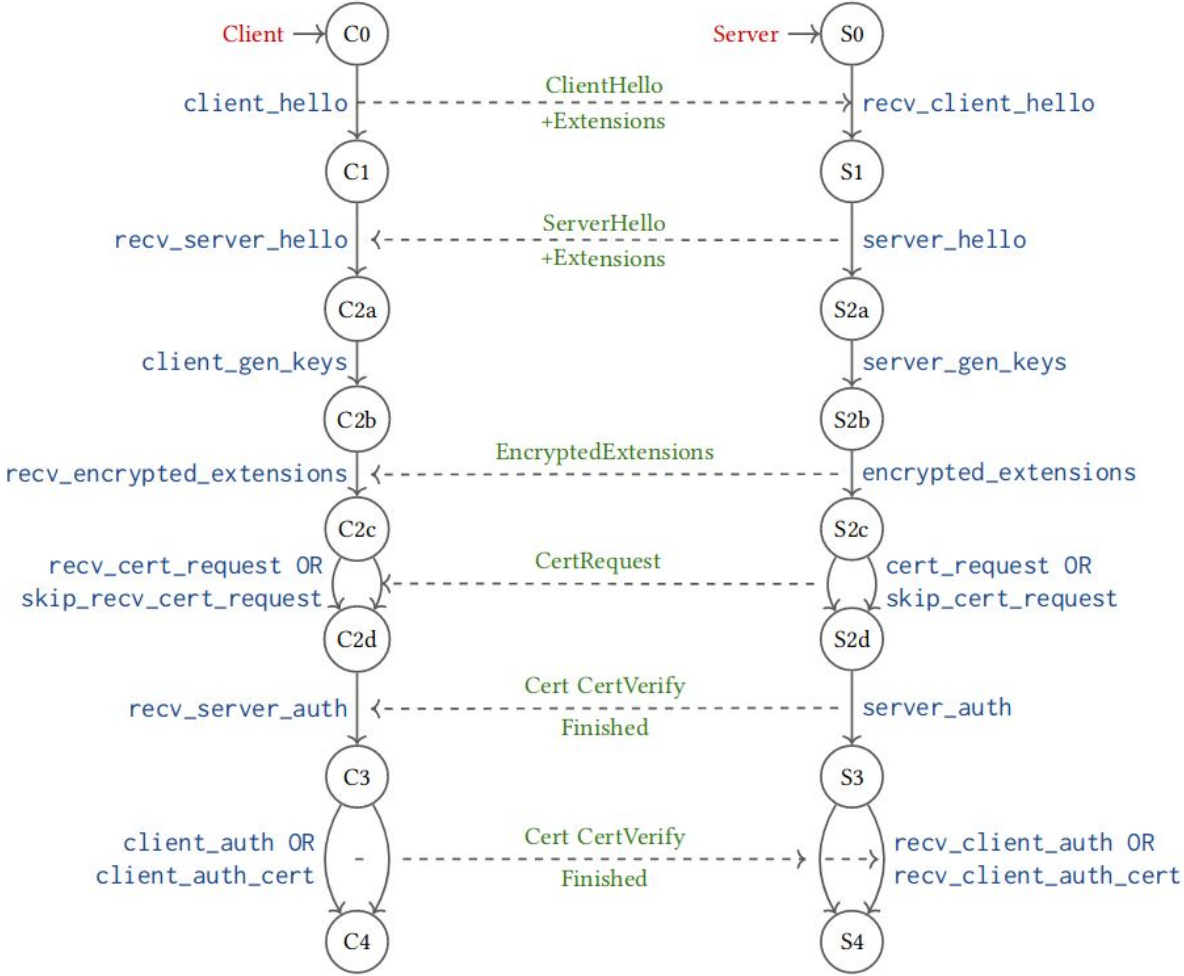


Figure 4: Partial state diagram for full TLS 1.3 handshake. Tamarin rules are indicated in blue. The messages exchanged between entities are given in green. Our full model contains many more transitions. We omit these here for the sake of simplicity.

Encoding the threat model

The attacker has complete control of the network

The Dolev-Yao model

http://www.cs.huji.ac.il/~dolev/pubs/dolev-yao-ieee-01056650.pdf

Proving the TLS 1.3 security properties

```
lemma secret_session_keys:
  "All tid actor peer write_key read_key peer_auth_status #i.
      SessionKey(tid, actor, peer, <peer_auth_status, 'auth'>, <write_key, read_key>)@i &
      not (Ex #r. RevLtk(peer)@r & #r < #i) &
      not (Ex tid3 x #r. RevDHExp(tid3, peer, x)@r & #r < #i) &
      not (Ex tid4 y #r. RevDHExp(tid4, actor, y)@r & #r < #i) &
      not (Ex resumption_master_secret #r. RevealPSK(actor, resumption_master_secret)@r) &
      not (Ex resumption_master_secret #r. RevealPSK(peer, resumption_master_secret)@r)
    ==> not Ex #j. K(read_key)@j"
```

Figure 5: secret_session_keys (Section 4.2.2)

```
lemma entity_authentication [use_induction, reuse]:
  "All tid actor peer nonces client_auth_status #i.
      CommitNonces(tid, actor, 'client', nonces)@i &
      CommitIdentity(tid, actor, 'client', peer, <client_auth_status, 'auth'>)@i &
      not (Ex #r. RevLtk(peer)@r & #r < #i) &
      not (Ex tid3 x #r. RevDHExp(tid3, peer, x)@r & #r < #i) &
      not (Ex tid4 y #r. RevDHExp(tid4, actor, y)@r & #r < #i) &
      not (Ex resumption_master_secret #r. RevealPSK(actor, resumption_master_secret)@r & #r < #i) &
      not (Ex resumption_master_secret #r. RevealPSK(peer, resumption_master_secret)@r & #r < #i)
    ==> (Ex tid2 #j. RunningNonces(tid2, peer, 'server', nonces)@j & #j < #i)"
```

Figure 6: entity_authentication (Section 4.2.3)

Analysis and results

Possitive results

"In general we find that TLS 1.3 meets the properties outlined in the specification that our modelling process was able to capture."

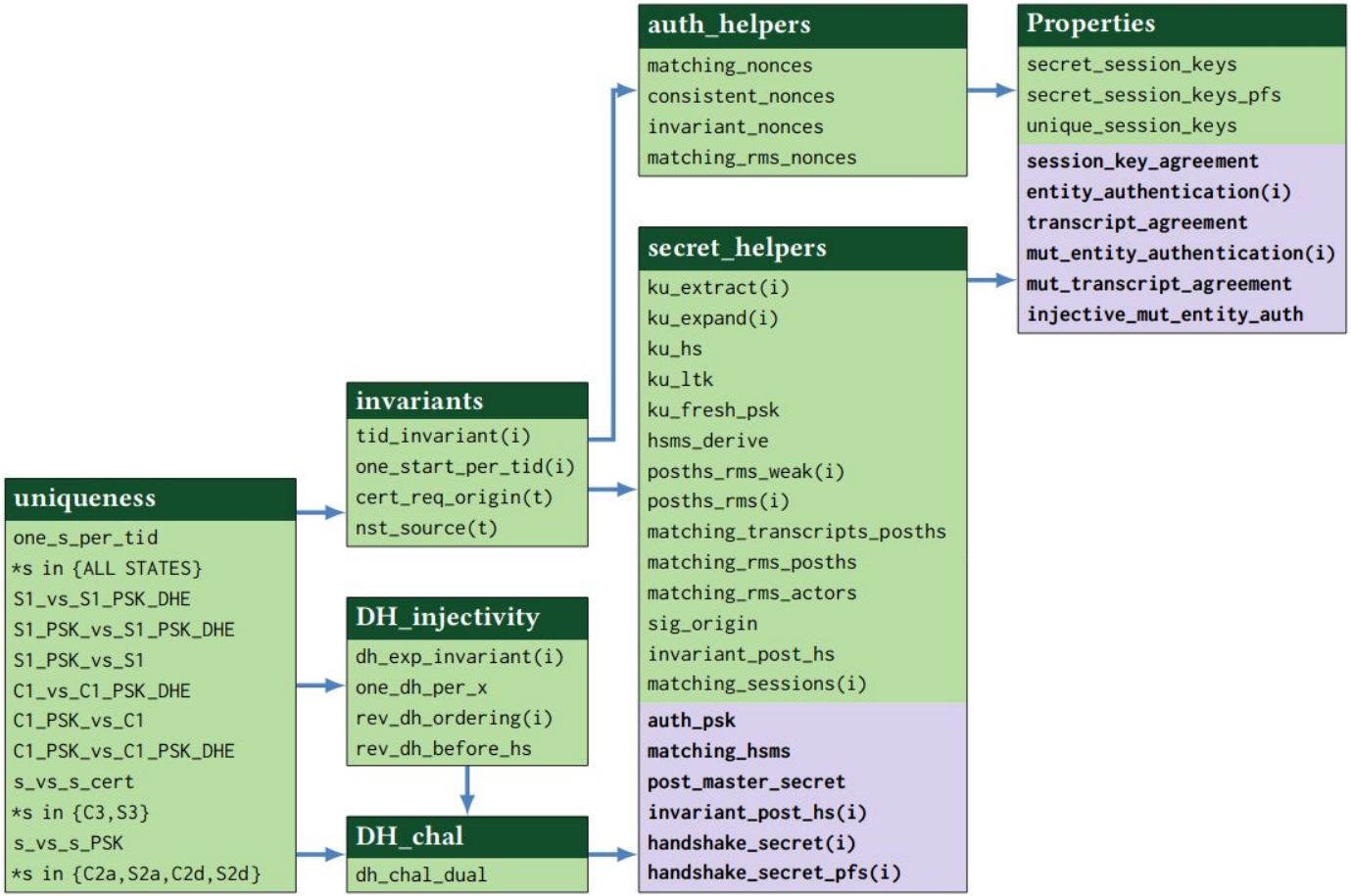


Figure 7: Lemma Map. Bold lemma names with a purple background indicate where manual interaction via the Tamarin visual interface was required. The remaining lemmas were automatically proven by Tamarin, without manual interaction. An arrow from one category to another implies that the proof of the latter depends on the former. The Properties box contains the main TLS 1.3 properties.

Negative results

"During our analysis of the post-handshake client authentication, it became apparent that the client does not receive any explicit confirmation that the server has successfully received the client’s response."

Conclusion

During the course of our analysis we also developed a line-by-line modelling aide that accurately captured which parts of the specification we were able to mode

https://samscott89.github.io/TLS13_Tamarin/

"Does NOT satisfy the traditional notion of forward secrecy"

More privacy, Less handshake

https://timtaubert.de/blog/2015/11/more-privacy-less-latency-improved-handshakes-in-tls-13/

Thank you for listening!

David Calavera

@calavera