Securely Deploying TLS 1.3

September 2017

Agenda

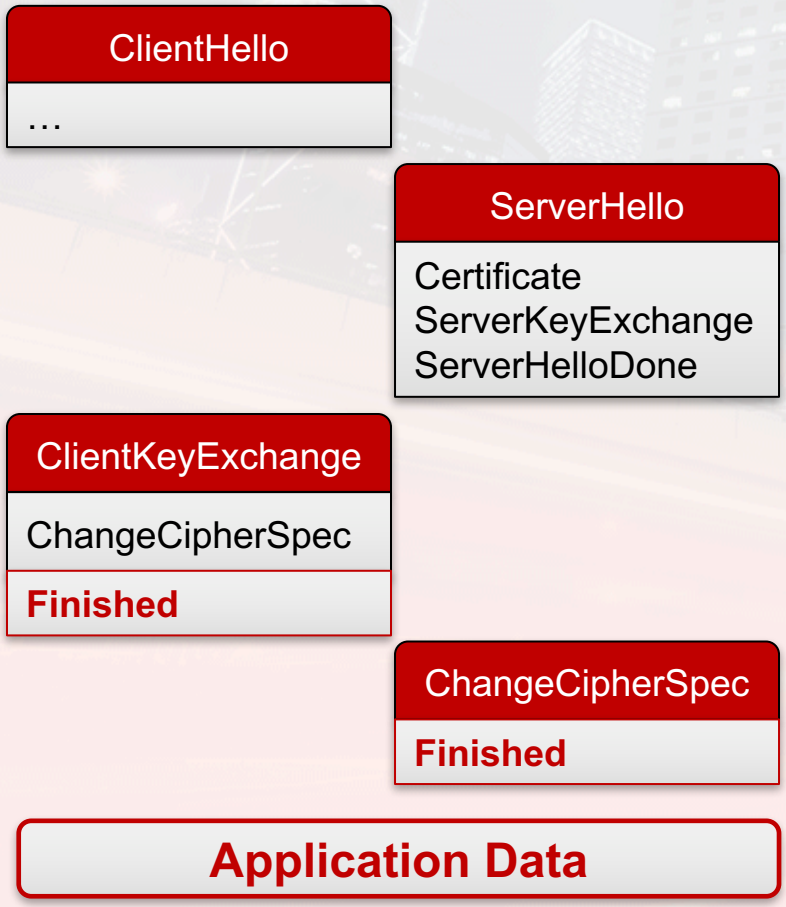
- Why TLS 1.3?
- Zero Round Trip Time (0-RTT) requests
- Forward secrecy
- Resumption key management

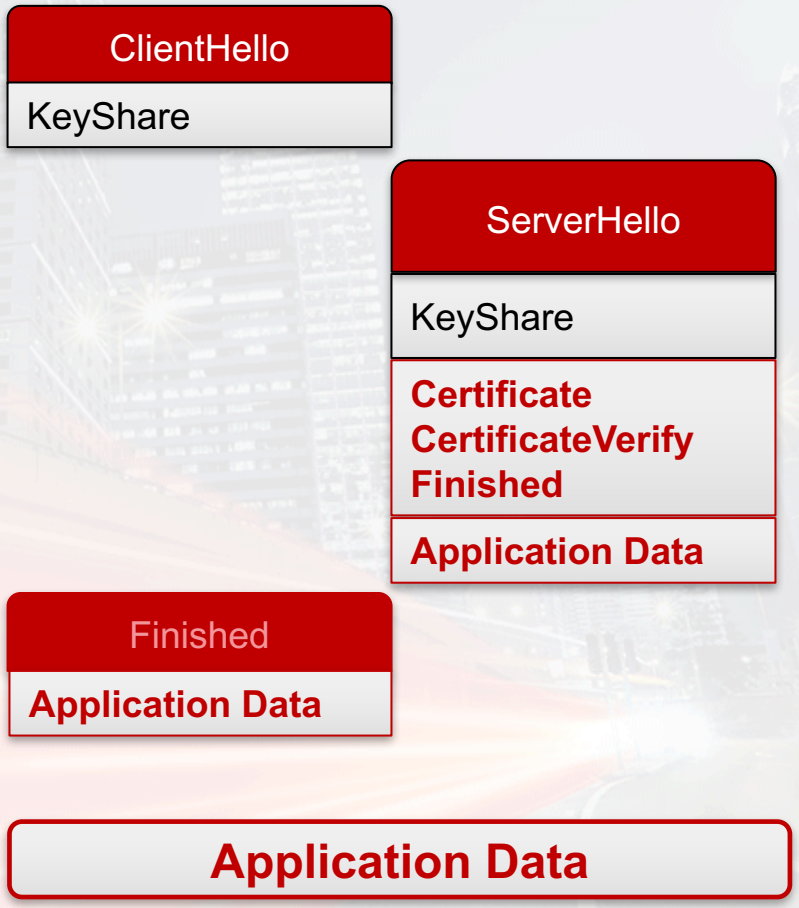
Why TLS 1.3?

Speed

- TLS impacts *latency*, not thoroughput
- Protocol setup requires one round trip
- Resume can be *zero* round trips
- Send application data ASAP

TLS 1.2 vs 1.3


ClientHello
…
ServerHello
Certificate
ServerKeyExchange
ServerHelloDone
ClientKeyExchange
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
Application Data


ClientHello
KeyShare
ServerHello
KeyShare
Certificate
CertificateVerify
Finished
Application Data
Finished
Application Data
Application Data

Your POODLE will not DROWN in CRIME

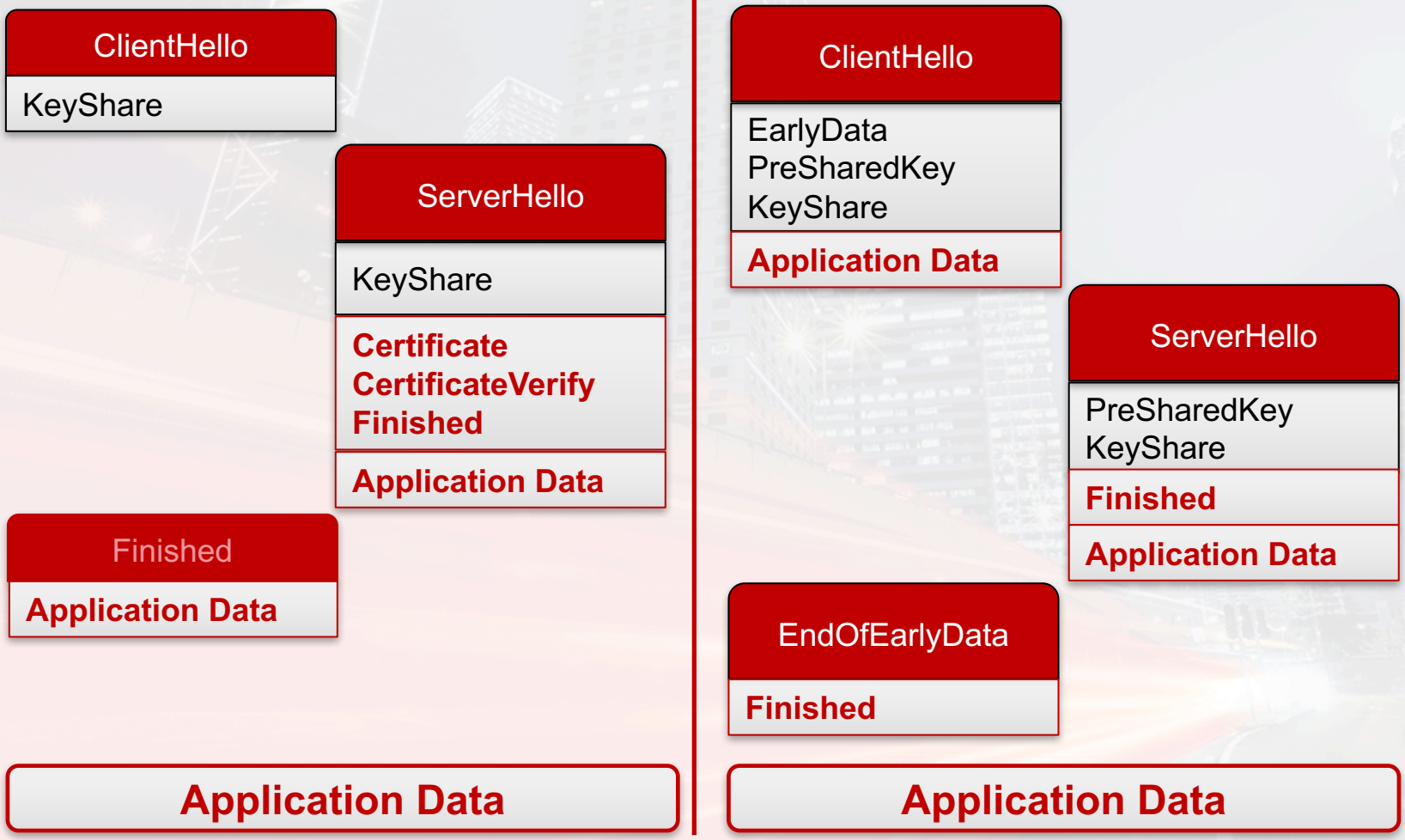
- All symmetric ciphers are AEAD
  - AES-GCM, AES-CCM, ChaCha20-Poly1305
- All key exchanges are ephemeral
  - FFDH over standard groups and ECDH
- All signatures are modern
  - RSA-PSS, ECDSA, EdDSA
- Troublesome features discarded
  - Compression, Export Ciphers, Explicit IV

Why TLS 1.3?

- Lower latency == happier users
- Conservative design == less churn
- Heavily reviewed and *deployed today*


nccgroup
Zero Round Trip Time

Standard Setup vs. 0-RTT



Security implications

- 0-RTT requests can be replayed
- Let's replay "Transfer 5 dollars to Scott"
- Another corner case – early server data
- *We have a layering violation!*

Reetbleed!

How on Earth did this happen?

- Unintended replays are a problem *now*
- Important transactions are *idempotent*
- Spec suggests users opt-in to 0-RTT
- Early draft adopters are working on patterns for application-level checks

Everything is ok

Zero Round Trip Time

Do…

- Design for idempotence
- Check for your stack's flag if you can't

Do Not...

- Turn on 0-RTT blindly for all requests
- Make a logo


nccgroup
Monitoring Traffic Securely

Agreeing on a common key

1. Client generates key and encrypts to server's public key

2. Client and Server use Diffie-Hellman with ephemeral parameters

RSA Key Exchange

- Option 1 is secure so long as the server's private key is never disclosed
- If that key is leaked or broken, all historic traffic can be decrypted

Diffie-Hellman Key Exchange

- Option 2 is secure as long as the server is not using a compromised key
- Attacker needs server private key AND intercept the DH exchange to compromise the session key

You get forward secrecy!

- All key exchanges in TLS 1.3 provide forward secrecy
- Great for practical security
- Great for hedge against unknown cryptographic breaks

...but

Monitoring solutions impacted

- If you rely on decrypting historic ciphertext, this means you
- There's a reason - we broke attackers that want to do the same thing
- IF you are affected, hit the whiteboard

Monitoring Traffic Securely

Do:

- Deploy TLS 1.3
- Monitor managed environments

Don’t:

- Hobble TLS 1.3
- Prefer down-level for ease of monitoring

Resumption Key Management

Session Resumption

- Remember 0-RTT?
- That pre-shared key needs to be shared
- In practice, client informs server of key

Session Resumption

1. Keep a list of all historic keys and give the client an identifier

2. Keep one key, use it to encrypt PSK

Session Resumption

- The spec leaves it to the implementer
- Option 2 is a safe bet
- Key management is your problem

Key Management Hiccups

- Unsynchronized keys across servers
  - 0-RTT Fails
- Failing to rotate aggressively
  - Great single point of failure
- Failing to negotiate ephemeral key
  - Limited benefits of forward secrecy

Resumption Key Management

Do:

- Rotate keys on an aggressive schedule
- Distribute keys to server farm securely
- Negotiate ephemeral keys after PSK

Don’t:

- Think it is secure out of the box


nccgroup
Thank You!

Thank You

- Crypto Services at NCC Group

- Joe Salowey of Tableau
- Nick Sullivan of Cloudflare

- The IETF Working Group

More Information

TLS 1.3 Specification

https://github.com/tlswg/tls13-spec

Bulletproof TLS Newsletter

https://www.feistyduck.com/bulletproof-tls-newsletter/

Cloudflare Blog

https://blog.cloudflare.com/

Questions? Comments?
scott.stender@nccgroup.trust
@ScottStender