October 3, 2018

TLS 1.3 and Certificate Transparency

Rasmus Dahlberg

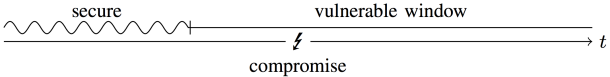
Outline

1. Secure communication and TLS ≤ 1.2
2. What's new in TLS 1.3 (and briefly why)
3. How CT makes TLS more robust

Disclaimer: expect little or no details on cryptographic primitives and attacks on TLS

Properties for secure client-server communication

- Secrecy
- Authentication
- Integrity



(a) Forward Secrecy

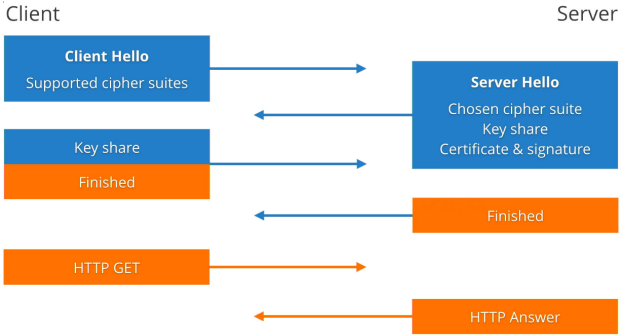
Source: "SoK: Secure Messaging" by Unger *et al.*, 2015.

What applications rely on these properties?

Recall TLS $\leq$ 1.2

Two main components:

- Handshake protocol
- Record protocol



Source: "Deploying TLS 1.3: the great, the good and the bad" by Valsorda and Sullivan, 2016.

Two round-trips before sending any application data!

...and cipher suites!

Key Exchange Authentication Cipher (algorithm strength mode) Mac or PRF

ECDHE-ECDSA-AES128-GCM-SHA256

Source: https://outspokenmedia.com/https/cipher-suites/

Are all cipher suites created equally?

- (EC)DH vs (EC)DHE?
- Static RSA vs (EC)DHE?
- MD5 vs SHA256?
- RC4 vs ChaCha20?
- 3DES vs AES?
- CBC vs GCM?

How hard can it be? (1/2)

Table of Contents

Source: RFC 7457 by Sheffer *et al.*, 2015.

The saga continues: FREAK, SLOTH, DROWN, ROBOT...

How hard can it be? (2/2)

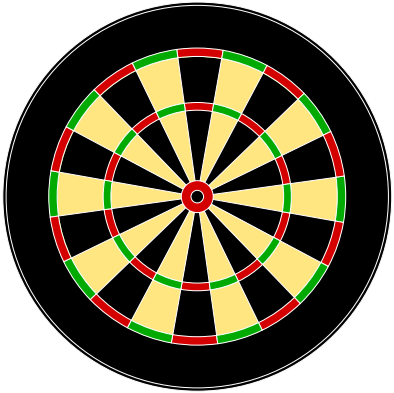
Product	CVE ID	Issue source
OpenSSL	2013-4353, 2015-0206, 2014-[3567, 3512, 3569, 3508, 3470, 0198, 0160]	Memory management
	2015-0205, 2015-0204, 2014-3572, 2014-0224, 2014-3568, 2014-3511	State machine
	2014-8275	Certificate parsing
	2014-2234	Certificate validation
	2014-3509, 2010-5298	Shared mutable state
	2014-0076	Timing side-channel
	2014-3570	Wrong sqrt
GnuTLS	2014-8564, 2014-3465, 2014-3466	Memory management
	2014-1959, 2014-0092, 2009-5138	Certificate validation
NSS	2014-1544	Memory management
	2013-1740	State machine
	2014-1490	Shared mutable state
	2014-1569, 2014-1568	Certificate parsing
	2014-1492	Certificate validation
	2014-1491	DH param validation
SChannel	2014-6321	Memory management
Secure Transport	2014-1266	State machine
JSSE	2014-6593, 2014-0626	State machine
	2014-0625	Memory exhaustion
	2014-0411	Timing side-channel
Applications	2014-2734	Memory management
	2014-3694, 2014-0139, 2014-2522, 2014-8151, 2014-1263	Certificate validation
	2013-7373, 2014-0016, 2014-0017, 2013-7295	RNG seeding
Protocol-level	2014-1771, 2014-1295, 2014-6457	Triple handshake
	2014-3566	POODLE

Table 1: Vulnerabilities in TLS implementations in 2014.

Source: "Not-Quite-So-Broken TLS: Lessons in Re-Engineering a Security Protocol Specification and Implementation" by Kaloper-Meršinjak *et al.*, 2015.

Objectives going into TLS 1.3

- Continuity
- Modern security analysis
- Clean up and simplicity
- Increased privacy & security
- Decreased latency



Simpler, safer and fewer cipher suites

- Five good AEAD choices
- Hash function for HKDF
- Format: TLS_AEAD_HASH

Description	Value
TLS_AES_128_GCM_SHA256	{0x13,0x01}
TLS_AES_256_GCM_SHA384	{0x13,0x02}
TLS_CHACHA20_POLY1305_SHA256	{0x13,0x03}
TLS_AES_128_CCM_SHA256	{0x13,0x04}
TLS_AES_128_CCM_8_SHA256	{0x13,0x05}

Source: RFC 8446 by Rescorla

No legacy ciphers or modes like MD5, RC4, 3DES, and CBC!

What about the key exchange? Selected orthogonally

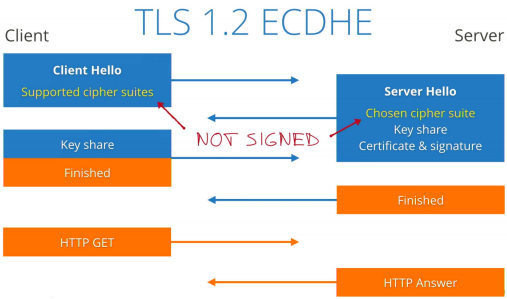
- **DHE**: ffdhe2048, ffdhe3072 (+3 more)
- **ECDHE**: secp256r1, x25519 (+3 more)
- **PSK**—replaces old ressumption mechanisms
- **PSK + (EC)DHE**

No static RSA, DH, or custom (EC)DHE groups!
→ forward secrecy
→ few well-selected defaults

Source: https://ds055uzetaobb.cloudfront.net/image_optimizer/a2a152fce89905aee9c0e051a0be8ea1d9b2c51c.jpg

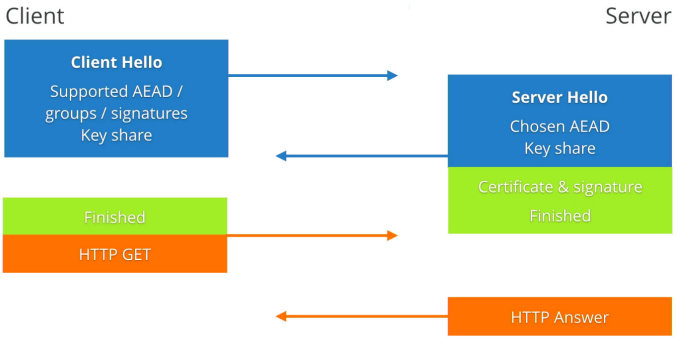
What about authentication? Also selected orthogonally

- Modern signature schemes
  - ECDSA, EdDSA, RSASSA-PSS
- Authenticate **entire** handshake
- Client and server certificates?



Source: "Deploying TLS 1.3: the great, the good and the bad" by Valsorda and Sullivan, 2016.

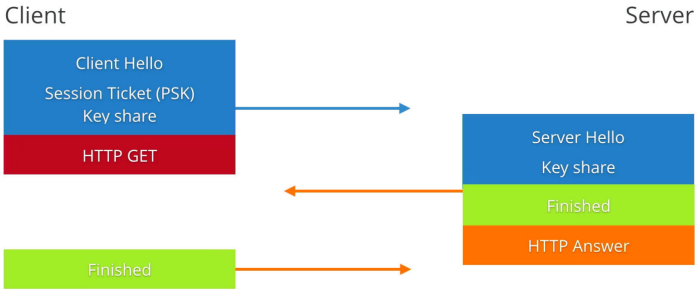
How TLS 1.3 improves latency? 1-RTT



Source: "Deploying TLS 1.3: the great, the good and the bad" by Valsorda and Sullivan, 2016.

- Predict negotiated parameters
- `HelloRetryRequest`

How TLS 1.3 improves latency? 0-RTT



Source: "Deploying TLS 1.3: the great, the good and the bad" by Valsorda and Sullivan, 2016.

The bad news... Forward secrecy and replay protection for 0-RTT?

Nope.

0-RTT data must be idempotent

Complexity vs Functionality ...can be debated

Forward secrecy—now and then

TLS 1.2

- Resumed 0-RTT: **inapplicable**
- Resumed 1-RTT: **never**
- Non-resumed 2-RTT: **sometimes**

TLS 1.3

- Resumed 0-RTT: **never**
- Resumed 1-RTT: **sometimes**[1]
- Non-resumed 1-RTT: **always**

[1] PSK never, PSK+(EC)DHE always

What else changed?

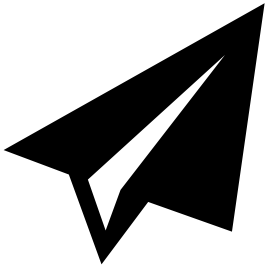
- Increased handshake encryption
  - E.g., server certificate encrypted
- Removed compression
- Downgrade protection
- Fixed renegotiation
- New versioning mechanism
- ...and many minor differences

Source: https://commons.wikimedia.org/wiki/File:Williton_Highbridge_Nursery_topiary_garden.jpg

Take away (1/2)

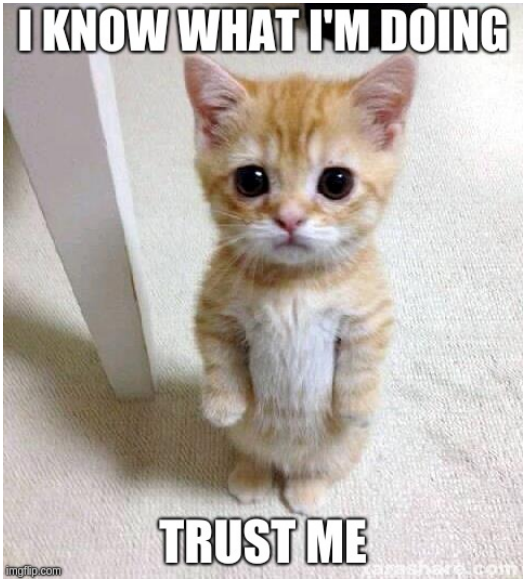
TLS 1.3:

- 'Fewer better choices'
- Mitigates past mistakes
- Use 0-RTT carefully



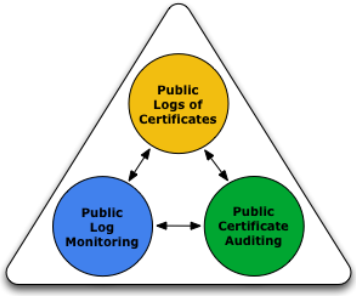
TLS relies on certificates to establish trust

- Signed identity-to-key bindings
- Who signs?
- Certificate Authorities (CAs)
- Problems? Ohh yes..



Certificate Transparency—in short CT[2] (or 'blockchain')

- Tamper-evident and append-only log
- Anyone can monitor the log for mis-issuance
- Anyone can audit that the log stays honest
  - Efficient consistency verification
  - Efficient inclusion verification
- Gossip, s.t., everybody sees the same log



Source: http://www.certificate-transparency.org/what-is-ct

[2] Note that CT is not limited to certificates

Adoption status of CT amongst popular vendors

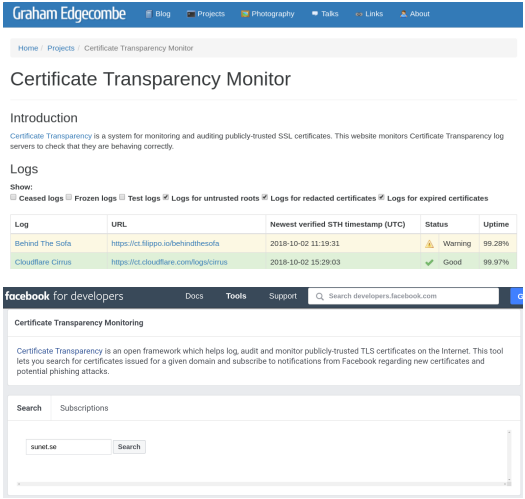
incrementally

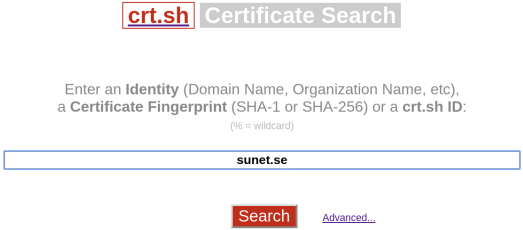
incrementally soon

unclear

- Clients require at least two promises of log inclusion
- Logs are trusted **until gossip-audit model hits deployment**

Get involved—monitor the logs!



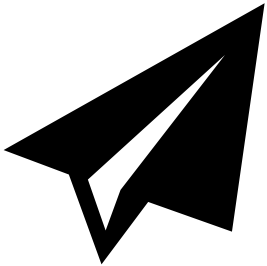


- What should a monitor do?
- Monitoring models
  - Self
  - As-a-service

Take away (2/2)

CT

- Goal? Detect certificate mis-issuance
- How? Require logging of all certificates
- Trust? No, because we can verify
- Who? Many major players involved



Questions?