TurboTLS

How to run TLS twice as fast with no changes to the protocol!

Carlos Aguilar-Melchor[1], Thomas Bailleux[1], Jason Goertzen[1,2], Adrien Guinet[1], David Joseph[1], Douglas Stebila[2]

2023-05-31

SandboxAQ[1], University of Waterloo[2]

Introduction

- What is TLS?
  - Overview
  - TLS over TCP

Introduction

- What is TLS?
  - Overview
  - TLS over TCP
- TurboTLS: shortening a round trip
  - Overview
  - Implementation details
  - Results

What is TLS?

Transport Layer Security

TLS - Overview

- Widely used cryptographic protocol

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers
- Confidentiality, integrity and authenticity

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers
- Confidentiality, integrity and authenticity
- Regardless of the transport mode

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers
- Confidentiality, integrity and authenticity
- Regardless of the transport mode
- Extendable

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers
- Confidentiality, integrity and authenticity
- Regardless of the transport mode
- Extendable

Versions

- `SSL 1.0, 2.0, 3.0`: deprecated

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers
- Confidentiality, integrity and authenticity
- Regardless of the transport mode
- Extendable

Versions

- `SSL 1.0, 2.0, 3.0`: deprecated
- `TLS 1.0, 1.1`: also deprecated

TLS - Overview

- Widely used cryptographic protocol
- To establish secure channels between two peers
- Confidentiality, integrity and authenticity
- Regardless of the transport mode
- Extendable

Versions

- `SSL 1.0`, `2.0`, `3.0`: deprecated
- `TLS 1.0`, `1.1`: also deprecated
- `TLS 1.2` `TLS 1.3`

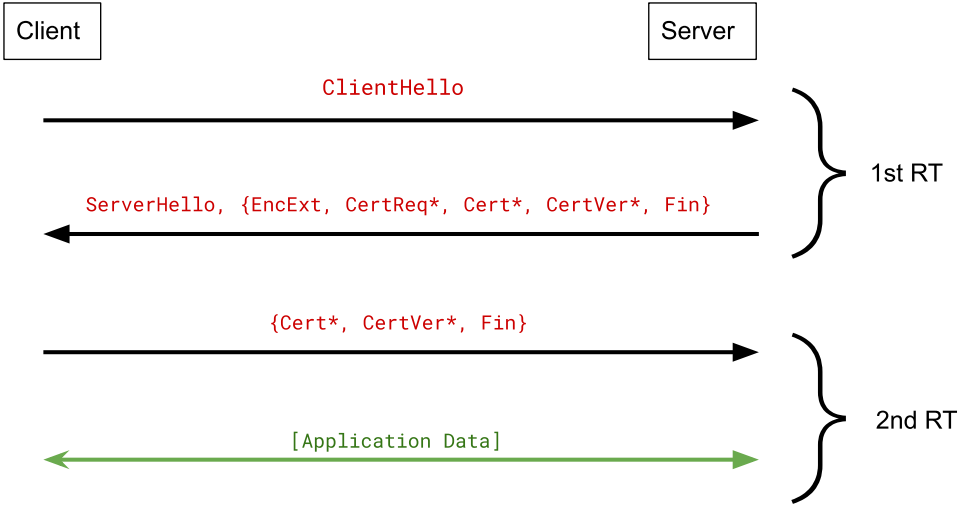


Figure 1: TLS 1.3 connection establishment.

TLS over TCP

HTTPS, SMTPS, ...

- TLS over TCP

- TLS messages transmitted via TCP

- TLS over TCP

- TLS messages transmitted via TCP
- Widely used: HTTPS, SMTPS, IRC,...

- TLS over TCP

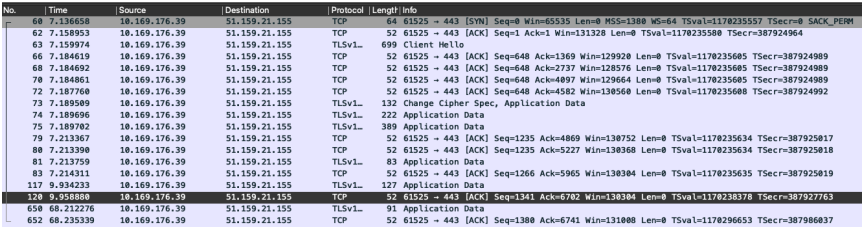
- TLS messages transmitted via TCP
- Widely used: HTTPS, SMTPS, IRC,...
- *Virtual* transport protocol

- TLS over TCP

- TLS messages transmitted via TCP
- Widely used: HTTPS, SMTPS, IRC,...
- *Virtual* transport protocol
- To encapsulate anything

- TLS over TCP

- TLS messages transmitted via TCP
- Widely used: HTTPS, SMTPS, IRC,...
- *Virtual* transport protocol
- To encapsulate anything



No.	Time	Source	Destination	Protocol	Length	Info
60	7.136658	10.169.176.39	51.159.21.155	TCP	64	61525 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1380 WS=64 TSval=1170235557 TSecr=0 SACK_PERM
62	7.158953	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=1170235580 TSecr=387924964
63	7.159974	10.169.176.39	51.159.21.155	TLSv1…	699	Client Hello
66	7.184619	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=648 Ack=1369 Win=129920 Len=0 TSval=1170235605 TSecr=387924989
68	7.184692	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=648 Ack=2737 Win=128576 Len=0 TSval=1170235605 TSecr=387924989
70	7.184861	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=648 Ack=4097 Win=129664 Len=0 TSval=1170235605 TSecr=387924989
72	7.187760	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=648 Ack=4582 Win=130560 Len=0 TSval=1170235608 TSecr=387924992
73	7.189509	10.169.176.39	51.159.21.155	TLSv1…	132	Change Cipher Spec, Application Data
74	7.189696	10.169.176.39	51.159.21.155	TLSv1…	222	Application Data
75	7.189702	10.169.176.39	51.159.21.155	TLSv1…	389	Application Data
79	7.213367	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=1235 Ack=4869 Win=130752 Len=0 TSval=1170235634 TSecr=387925017
80	7.213390	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=1235 Ack=5227 Win=130368 Len=0 TSval=1170235634 TSecr=387925018
81	7.213759	10.169.176.39	51.159.21.155	TLSv1…	83	Application Data
83	7.214311	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=1266 Ack=5965 Win=130304 Len=0 TSval=1170235635 TSecr=387925019
117	9.934233	10.169.176.39	51.159.21.155	TLSv1…	127	Application Data
120	9.958880	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=1341 Ack=6702 Win=130304 Len=0 TSval=1170238378 TSecr=387927763
650	60.212276	10.169.176.39	51.159.21.155	TLSv1…	91	Application Data
652	60.235339	10.169.176.39	51.159.21.155	TCP	52	61525 → 443 [ACK] Seq=1380 Ack=6741 Win=131008 Len=0 TSval=1170296653 TSecr=387986037

Figure 2: TLS 1.3 over TCP, Wireshark

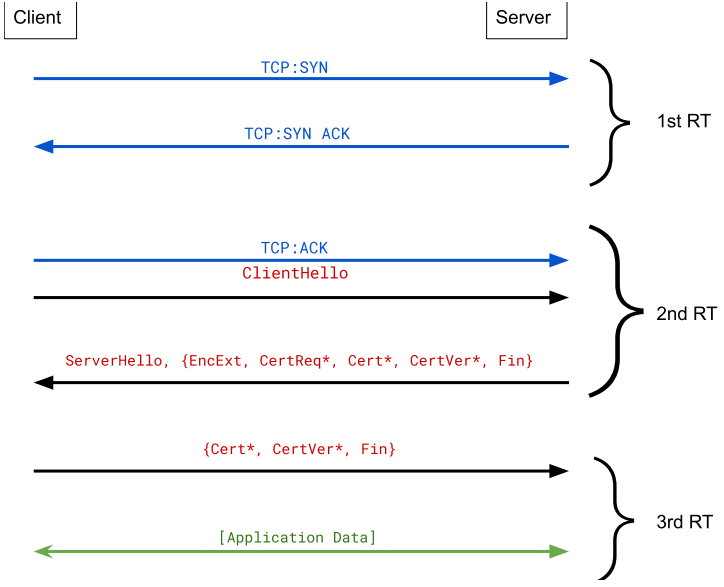


Figure 3: TLS 1.3 connection establishment over TCP.

Latency-critical contexts

TLS depth, post-quantum cryptography

Websites & TLS depth - What it is

Imagine https://domain.com that loads:

Websites & TLS depth - What it is

Imagine https://domain.com that loads:

- some javascript from https://mycdn.com

Websites & TLS depth - What it is

Imagine https://domain.com that loads:

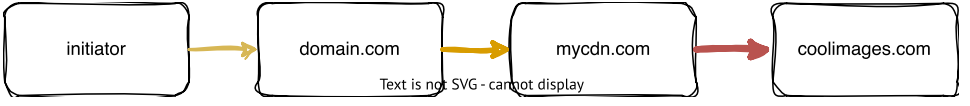
- some javascript from https://mycdn.com
- that javascript then fetches some images for the website from https://coolimages.com
- (please don't do this at home)

Websites & TLS depth - What it is

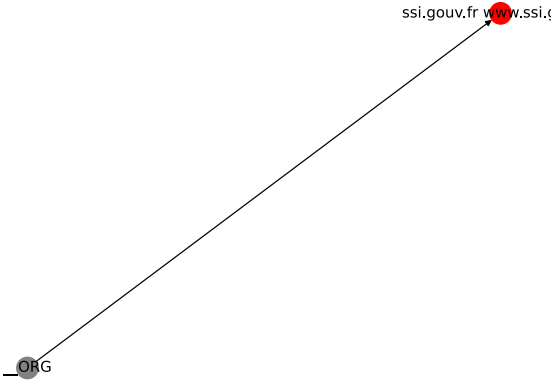
Imagine https://domain.com that loads:

- some javascript from https://mycdn.com
- that javascript then fetches some images for the website from https://coolimages.com
- (please don't do this at home)

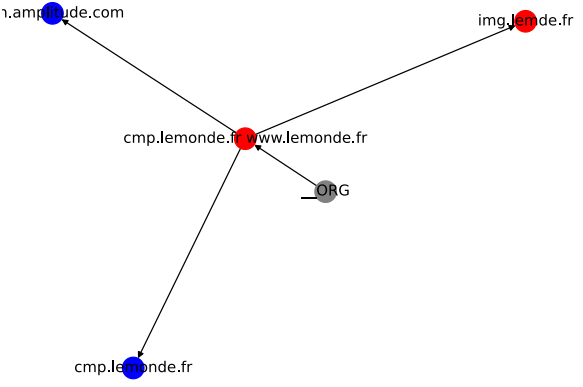
Assuming these three domains resolve to three different IPs, there is a TLS depth of 3 to get that image:



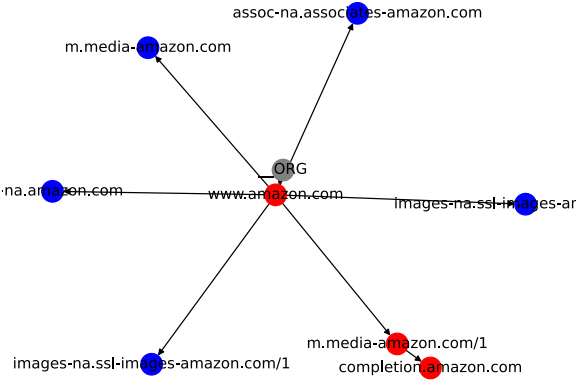
Websites & TLS depth - Example: ssi.gouv.fr



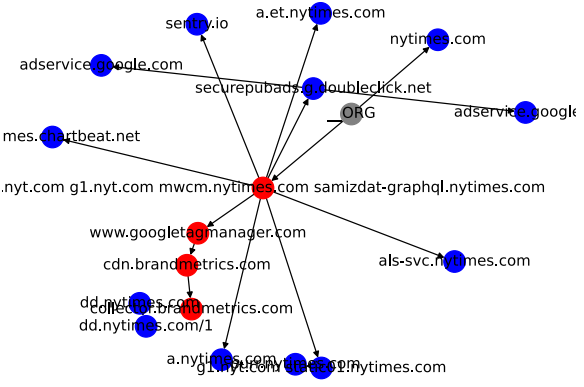
Websites & TLS depth - Example: lemonde.fr



Websites & TLS depth - Example: amazon.com



Websites & TLS depth - Example: nytimes.com



Post-quantum cryptography - TLS

- Thought to be secure against attacks by a quantum computer

Post-quantum cryptography - TLS

- Thought to be secure against attacks by a quantum computer
- Trade off computation time / size of materials

Post-quantum cryptography - TLS

- Thought to be secure against attacks by a quantum computer
- Trade off computation time / size of materials
- Sign/Verify primitives usually slower

Post-quantum cryptography - TLS

- Thought to be secure against attacks by a quantum computer
- Trade off computation time / size of materials
- Sign/Verify primitives usually slower
- Impact on TLS handshake latencies
- Impact on the whole connection establishment

Post-quantum vs Classical cryptography - Sizes

Signature algorithms

	Secret key	Public key	Signature
P-256	96 B	64 B	64 B
sphincs+s256-sha256-simple	128 B	64 B	29,792 B
Falcon512	1,281 B	897 B	660 B
Falcon1024	2,305 B	1,793 B	1,275 B
Dilithium2	2,544 B	1,312 B	2,420 B
Dilithium3	4,016 B	1,952 B	3,293 B
Dilithium5	4,880 B	2,592 B	4,595 B

source: SUPERCOP

PQ vs Classical - Computation time

amd64, AMD Ryzen 5 Pro 5650G, 3,900MHz

	`genkeypair`	`sign`	`verify`
ed25519	11µs	11µs	41µs
P-256	22µs	31µs	68µs
Dilithium5	51µs	108µs	53µs
Falcon512	5,170µs	176µs	22µs
Falcon1024	15,017µs	351µs	43µs
sphincs+-s256-shake256-simple	64,642µs	794,568µs	1,371µs

source: SUPERCOP

How to reduce the latency of the TLS handshake?

TurboTLS

TCP+UDP

TurboTLS - Overview

- *TurboTLS*: TLS over TCP+UDP

TurboTLS - Overview

- *TurboTLS*: TLS over TCP+UDP
- Handshake plane over UDP, record plane over TCP

TurboTLS - Overview

- *TurboTLS*: TLS over TCP+UDP
- Handshake plane over UDP, record plane over TCP
- TLS handshake \\ TCP handshake

TurboTLS - Overview

- *TurboTLS*: TLS over TCP+UDP
- Handshake plane over UDP, record plane over TCP
- TLS handshake \\ TCP handshake
- Fallback to TCP

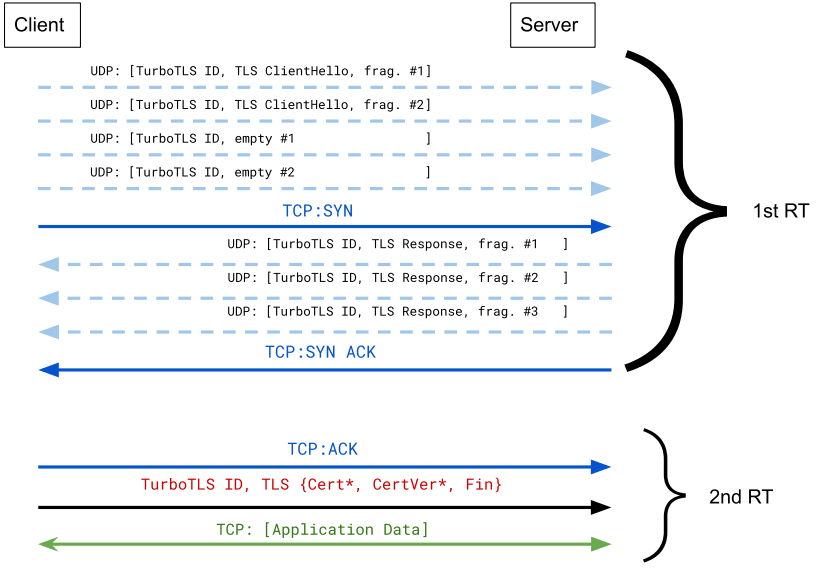


Figure 9: TurboTLS connection establishment.

Implementation

TurboTLS - TurboTLS frame

- Session ID, `SID`, random and unique

TurboTLS - TurboTLS frame

- Session ID, `SID`, random and unique
- Fragment index (`[0 - 255]`)

TurboTLS - TurboTLS frame

- Session ID, `SID`, random and unique
- Fragment index (`[0 - 255]`)
- TLS payload

TurboTLS - TurboTLS frame

- Session ID, `SID`, random and unique
- Fragment index (`[0 - 255]`)
- TLS payload

TurboTLS frame.

TurboTLS - Client-based request fragmentation

1. *client*: `ClientHello` sent

TurboTLS - Client-based request fragmentation

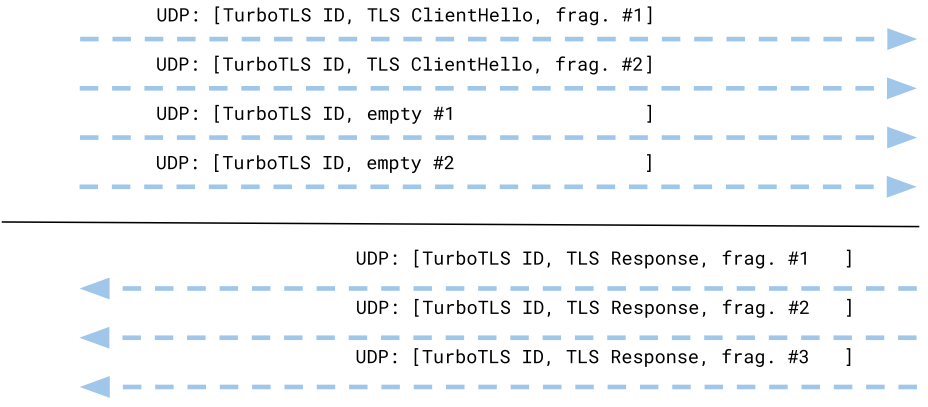
1. *client*: `ClientHello` sent
2. *client*: `n` empty datagrams sent

TurboTLS - Client-based request fragmentation

1. *client*: `ClientHello` sent
2. *client*: `n` empty datagrams sent
3. *server*: `n'` datagrams sent back, `n'<= n`

TurboTLS - Client-based request fragmentation

1. *client*: `ClientHello` sent
2. *client*: `n` empty datagrams sent
3. *server*: `n'` datagrams sent back, `n'<= n`



Client-based request fragmentation in TurboTLS

Security

DDoS, reliability and advertising

TurboTLS - Security and reliability

DDoS and Amplification attack

TurboTLS - Security and reliability

DDoS and Amplification attack

- Attacker sends a large number of partial TLS records over UDP
- $\longrightarrow$ memory exhaustion on server side

TurboTLS - Security and reliability

DDoS and Amplification attack

- Attacker sends a large number of partial TLS records over UDP
- $\longrightarrow$ memory exhaustion on server side
- **Mitigation:** ring buffer

TurboTLS - Security and reliability

DDoS and Amplification attack

- Attacker sends a large number of partial TLS records over UDP
- $\longrightarrow$ memory exhaustion on server side
- **Mitigation:** ring buffer
- **Mitigation:** client-based request fragmentation

TurboTLS - Security and reliability

DDoS and Amplification attack

- Attacker sends a large number of partial TLS records over UDP
- $\longrightarrow$ memory exhaustion on server side
- **Mitigation:** ring buffer
- **Mitigation:** client-based request fragmentation

Reliability

- UDP is not reliable

TurboTLS - Security and reliability

DDoS and Amplification attack

- Attacker sends a large number of partial TLS records over UDP
- → memory exhaustion on server side
- **Mitigation:** ring buffer
- **Mitigation:** client-based request fragmentation

Reliability

- UDP is not reliable
- **Mitigation:** TCP fallback
- → switch the TLS handshake from UDP to TCP

TurboTLS - Security and reliability

DDoS and Amplification attack

- Attacker sends a large number of partial TLS records over UDP
- → memory exhaustion on server side
- **Mitigation:** ring buffer
- **Mitigation:** client-based request fragmentation

Reliability

- UDP is not reliable
- **Mitigation:** TCP fallback
- → switch the TLS handshake from UDP to TCP

Advertising

- Additional flag in existing DNS records (HTTPS record), ⟶ no extra latency

Experimental results

Latencies and throughput

TurboTLS - experimental results

Paris - Paris

ping: 261 µs

Machine: *Intel Xeon E5-2696V4, 16GB, 4 pcores*

	Sig/KEX	Throughput (h/sec)	Lat. Med (µ s)	Lat. P99 (µ s)
TLS 1.3	`ECDSAp256/ECDH nistp256`	2,901	1,299	1,935
TurboTLS 1.3	**`ECDSAp256/ECDH nistp256`**	**3,461**	**1,057**	**1,634**
TLS 1.3	`Dilithium2/Kyber512`	2,478	1,546	2,497
TurboTLS 1.3	**`Dilithium2/Kyber512`**	**4,400**	**821**	**1,290**

TurboTLS - experimental results

Paris - Oregon

ping: 133,021 µs

Machine: *Intel Xeon E5-2696V4, 16GB, 4 pcores*

	Sig/KEX	Throughput (h/sec)	Lat. Med (µ s)	Lat. P99 (µ s)
TLS 1.3	`ECDSAp256/ECDH nistp256`	16	266,984	268,540
TurboTLS 1.3	**`ECDSAp256/ECDH nistp256`**	**32**	**133,981**	**135,583**
TLS 1.3	`Dilithium2/Kyber512`	16	269,477	272,534
TurboTLS 1.3	**`Dilithium2/Kyber512`**	**32**	**133,764**	**135,835**

TurboTLS - experimental results

Paris - Sydney

ping: 269,478 µs

Machine: *Intel Xeon E5-2696V4, 16GB, 4 pcores*

	Sig/KEX	Throughput (h/sec)	Lat. Med (µ s)	Lat. P99 (µ s)
TLS 1.3	`ECDSAp256/ECDH nistp256`	8	540,036	540,464
TurboTLS 1.3	**`ECDSAp256/ECDH nistp256`**	**16**	**270,276**	**270,792**
TLS 1.3	`Dilithium2/Kyber512`	8	542,831	543,488
TurboTLS 1.3	**`Dilithium2/Kyber512`**	**16**	**270,154**	**271,569**

Thank you!

Questions?