Computer Networks Advanced Course

Secure Sockets Part 5 – TLS 1.3

Barak Gonen

Part 5 – TLS 1.3 Intro

- Presentation focus – how TLS 1.3 performs faster handshake?

- TLS 1.3 Motivation
- Changes from v1.2
  - Cipher suites
  - Less RTT
- Handshake
- QUIC (future lesson)

Issues not covered

- Key schedule
- 0 RTT handshake
- Session resumption (Pre-Shared Key)
- Mutual authentication
- TLS 1.3 extensions

TLS 1.3 Motivation

- Prefer simplicity over backwards compatibility
- Examples:
  - No more support for 1970's ciphers
  - Save memory – great for IOT devices
  - Reduced RTT – better user experience

TLS 1.3 Main Changes

- Cipher suites
- Handshake
- Session re-negotiation & resumption

TLS 1.3 Cipher Suites

- Recall 1.2: TLS_x_y_with_z_w
  - X – Key Exchange
  - Y – Authentication
  - Z – Encryption
  - W – Hashing

Key Exchange

ECDHE
DHE
ECDH
DH
RSA
PSK

Authentication

ECDSA
RSA
DSS
PSK

Encryption

CHACHA20
AES256GCM
AES128GCM
AES256CBC
AES128CBC
3DES CBC
RC4 128
DES CBC

Hashing

POLY1305
SHA384
SHA256
SHA
MD5

TLS 1.3 Cipher Suites

- Recall 1.2: TLS_x_y_with_z_w
  - X – Key Exchange
  - Y – Authentication
  - Z – Encryption
  - W – Hashing

Key Exchange

ECDHE
DHE
~~ECDH~~
~~DH~~
~~RSA~~
~~PSK~~

Authentication

ECDSA
RSA
~~DSS~~
~~PSK~~

Encryption

CHACHA20
AES256GCM
AES128GCM
~~AES256CBC~~
~~AES128CBC~~
~~3DES CBC~~
~~RC4 128~~
~~DES CBC~~

Hashing

POLY1305
SHA384
SHA256
~~SHA~~
~~MD5~~

Key Exchange

- “E” stands for “Ephemeral”– temporal
- There is no private key which can be broken
  - Forward Secrecy – if sessions keys are broken, only this session is compromised
  - For that reason, RSA is no longer an option

Key Exchange

ECDHE
DHE
~~ECDH~~
~~DH~~
~~RSA~~
~~PSK~~

TLS 1.3 Cipher Suites

Key Exchange | Authentication

Chosen Independently

Encryption | Hashing

TLS 1.3 Cipher Suite
TLS_Encryption_Hashing

- TLS_AES_128_GCM_SHA256 — MUST implement
- TLS_AES_256_GCM_SHA384 — SHOULD implement
- TLS_CHACHA20_POLY1305_SHA256 — SHOULD implement
- TLS_AES_128_CCM_8_SHA256
- TLS_AES_128_CCM_SHA256

- Only 5 – low memory required – great for IOT devices
- All suites are AEAD – next ☺

TLS 1.3 Cipher Suites

- Study by Wireshark
- Sniff Client Hello v1.3
- Check the extension “supported versions”, verify 1.3
- Check the cipher suites:
  - Which are 1.2? 1.3?
  - Client Hello – all possible options
  - Server Hello – selected suite

No.	Time	Source	Destination	Protocol	Length	Host	info
372	2.984736	192.168.1.221	45.60.207.1	TCP	66		58111 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
376	2.989943	45.60.207.1	192.168.1.221	TCP	66		443 → 58111 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1452 SACK_PERM WS=128
378	2.990051	192.168.1.221	45.60.207.1	TCP	54		58111 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0
379	2.990453	192.168.1.221	45.60.207.1	TCP	1506		58111 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=1452 [TCP segment of a reassembl
380	2.990453	192.168.1.221	45.60.207.1	TLSv1.3	398		Client Hello (SNI=www.bankhapoalim.co.il)
381	2.994846	45.60.207.1	192.168.1.221	TCP	60		443 → 58111 [ACK] Seq=1 Ack=1453 Win=64128 Len=0
382	2.994846	45.60.207.1	192.168.1.221	TCP	60		443 → 58111 [ACK] Seq=1 Ack=1797 Win=63872 Len=0
383	2.994894	45.60.207.1	192.168.1.221	TLSv1.3	2958		Server Hello, Change Cipher Spec, Encrypted Extensions
384	2.994894	45.60.207.1	192.168.1.221	TCP	1246		443 → 58111 [PSH, ACK] Seq=2905 Ack=1797 Win=64128 Len=1192 [TCP segment of a

Client Hello Extensions

- The cipher suite does not include:
  - Key exchange algorithm
  - Authentication algorithm
- The information is in the extensions

```
> Extension: supported_groups (len=12)
> Extension: psk_key_exchange_modes (len=2)
> Extension: session_ticket (len=0)
> Extension: extended_master_secret (len=0)
> Extension: encrypted_client_hello (len=282)
> Extension: key_share (len=1263) X25519Kyber768Draft00, x25519
> Extension: server_name (len=30) name=bcodes.bankhapoalim.co.il
> Extension: ec_point_formats (len=2)
> Extension: renegotiation_info (len=1)
> Extension: status_request (len=5)
> Extension: signed_certificate_timestamp (len=0)
> Extension: supported_versions (len=7) TLS 1.3, TLS 1.2
> Extension: signature_algorithms (len=18)
```

Key Exchange

- Now that we support only DH, the client may try to guess the server's key exchange method, save time ☺
  - "Hi server, here are the DH curves I support"

```
Extension: supported_groups (len=12)
    Type: supported_groups (10)
    Length: 12
    Supported Groups List Length: 10
    Supported Groups (5 groups)
        Supported Group: Reserved (GREASE) (0x2a2a)
        Supported Group: X25519Kyber768Draft00 (0x6399)
        Supported Group: x25519 (0x001d)
        Supported Group: secp256r1 (0x0017)
        Supported Group: secp384r1 (0x0018)
```

Key Exchange

ECDHE
DHE
~~ECDH~~
~~DH~~
~~RSA~~
~~PSK~~

Key Exchange

- Now that we support only DH, the client may try to guess the server's key exchange method, save time ☺
  - "Hi server, here are the DH curves I support"
  - …"I guess you will pick one of these"

Key Exchange

ECDHE
DHE
~~ECDH~~
~~DH~~
~~RSA~~
~~PSK~~

```
Extension: key_share (len=1263) X25519Kyber768Draft00, x25519
```

Key Exchange

- Now that we support only DH, the client may try to guess the server's key exchange method, save time ☺
  - "Hi server, here are the DH curves I support"
  - …"I guess you will pick one of these"
  - … "and here is my part of DH key for the curve/s I guess you pick"

Key Exchange

ECDHE
DHE
~~ECDH~~
~~DH~~
~~RSA~~
~~PSK~~

```
Key Share Entry: Group: X25519Kyber768Draft00, Key Exchange length: 1216
    Group: X25519Kyber768Draft00 (25497)
    Key Exchange Length: 1216
    Key Exchange [truncated]: 09805ad00c1b75b13b26b075bb6b24a5b36aa06e25dffb78800f5
Key Share Entry: Group: x25519, Key Exchange length: 32
    Group: x25519 (29)
    Key Exchange Length: 32
    Key Exchange: 111b69793d65ca2fd364ad9de3fb75d38dc35740534b733f3ad8499a11639b44
```

Key Exchange

- What if the client is wrong?
  - Servers sends HelloRetryRequest (SHA256 hash) as “Random”
  - tls.handshake.random == CF:21:AD:74:E5:9A:61:11:BE:1D:8C:02:1E:65:B8:91:C2:A2:11:16:7A:BB:8C:5E:07:9E:09:E2:C8:A8:33:9C
- Find it on Wireshark!

tls.handshake.random == CF:21:AD:74:E5:9A:61:11:BE:1D:8C:02:1E:65:B8:91:C2:A2:11:16:7A:BB:8C:5E:07:9E:09:E2:C8:A8:33:9C

No.	Time	Source	Destination	Protocol	Length	Host	info
35589	193.867709	192.229.133.221	192.168.1.221	TLSv1.3	153		Hello Retry Request, Change Cipher Spec
35631	193.967723	192.229.133.221	192.168.1.221	TLSv1.3	153		Hello Retry Request, Change Cipher Spec
36454	195.042964	192.229.133.221	192.168.1.221	TLSv1.3	153		Hello Retry Request, Change Cipher Spec
37871	197.846875	2620:1ec:bdf::43	2a0d:6fc2:131c:500:a163:b653:34de:…	TLSv1.3	173		Hello Retry Request, Change Cipher Spec

Authentication

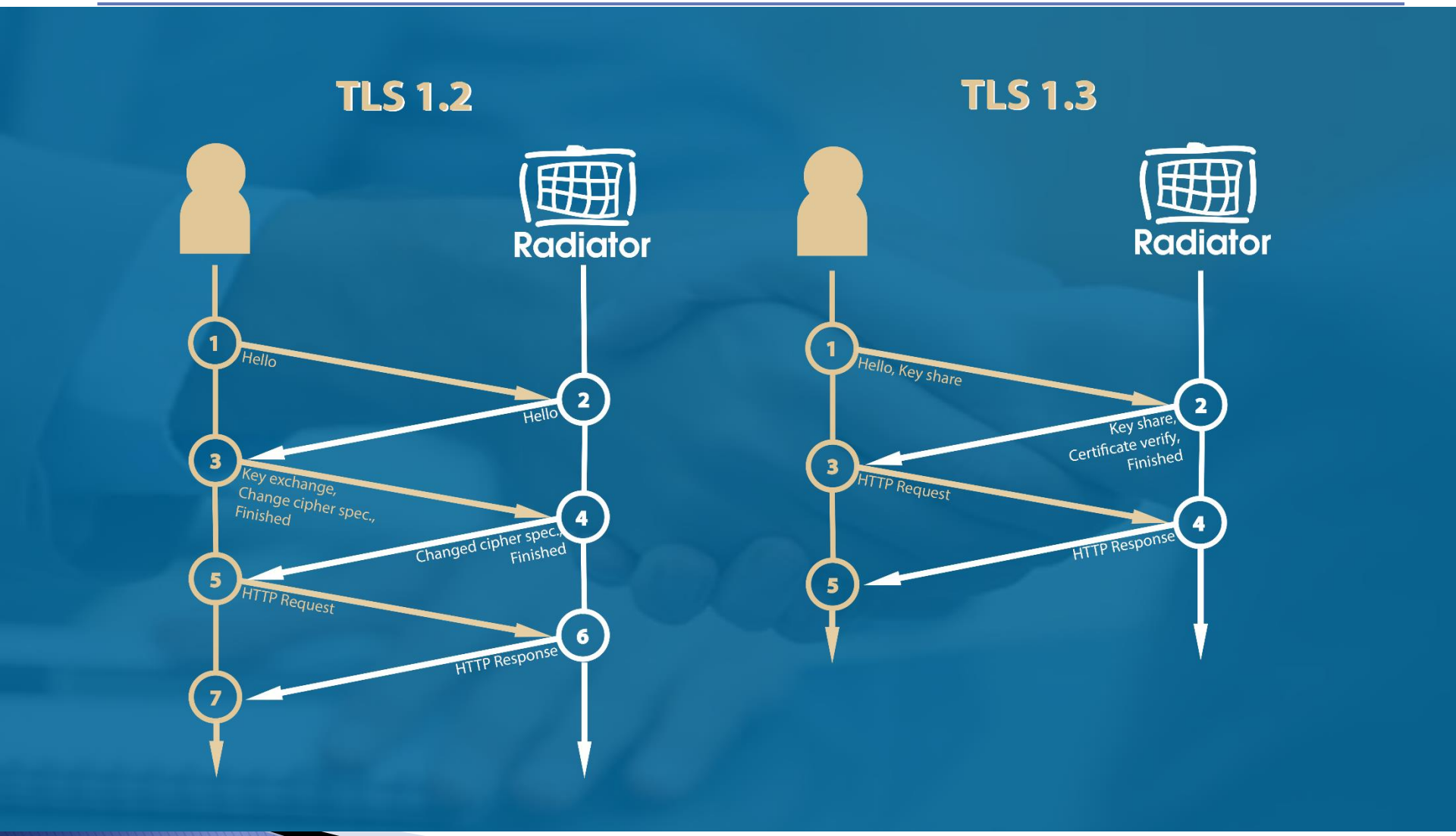
- The signature will be used for the certificate
  - Not the Hash used for the MAC
- Wireshark:
  - Client Hello – Extensions
  - Signature + Hash
  - Look for response in Server Certificate

```
Extension: signature_algorithms (len=18)
    Type: signature_algorithms (13)
    Length: 18
    Signature Hash Algorithms Length: 16
    Signature Hash Algorithms (8 algorithms)
        Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
        Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
        Signature Algorithm: rsa_pkcs1_sha256 (0x0401)
        Signature Algorithm: ecdsa_secp384r1_sha384 (0x0503)
        Signature Algorithm: rsa_pss_rsae_sha384 (0x0805)
        Signature Algorithm: rsa_pkcs1_sha384 (0x0501)
        Signature Algorithm: rsa_pss_rsae_sha512 (0x0806)
        Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
```

TLS 1.2 Handshake- DH version

TLS 1.3 Handshake

TLS Handshake – Save RTT



Source: https://blog.radiatorsoftware.com/2022/07/radiator-supports-eap-tls-13.html

Handshake

- By the end of TLS 1.3 both sides have:
  - Version – TLS 1.3
  - Cipher Suite
  - Signature Algorithm
  - DH Group
  - Client Key Share
  - Server Key Share
- From here on, all session keys can be generated

Generating Session Keys

TLS 1.3 Key Schedule

v1.0

0000 | 0000 | Pre-Shared-Key | 0000 (or) → Early Secret

PRK	Label	Handshake Transcript		Keying Material	
Early Secret	<res|ext> binder		=	Binder Key	
Early Secret	c e traffic	Client Hello	=	Client Early Traffic Secret	Client Early Traffic Key, Client Early Traffic IV
Early Secret	e exp master	Client Hello	=	Early Exporter Master Secret	
Early Secret	derived		=	Early Derivative Key	

Early Derivative Key | (EC)DHE Shared Secret → Handshake Secret

PRK	Label	Handshake Transcript		Keying Material	
Handshake Secret	c hs traffic	Client Hello ... Server Hello	=	Client Handshake Traffic Secret	Client Handshake Key, Client Handshake IV
Handshake Secret	s hs traffic	Client Hello ... Server Hello	=	Server Handshake Traffic Secret	Server Handshake Key, Server Handshake IV
Handshake Secret	derived		=	Handshake Derivative Key	

Handshake Derivative Key | 0000 → Master Secret

PRK	Label	Handshake Transcript		Keying Material	
Master Secret	c ap traffic	Client Hello ... Server Finished	=	Client Application Traffic Secret	Client Application Key, Client Application IV
Master Secret	s ap traffic	Client Hello ... Server Finished	=	Server Application Traffic Secret	Server Application Key, Server Application IV
Master Secret	exp master	Client Hello ... Server Finished	=	Exporter Master Secret	
Master Secret	res master	Client Hello ... Client Finished	=	Resumption Master Secret	