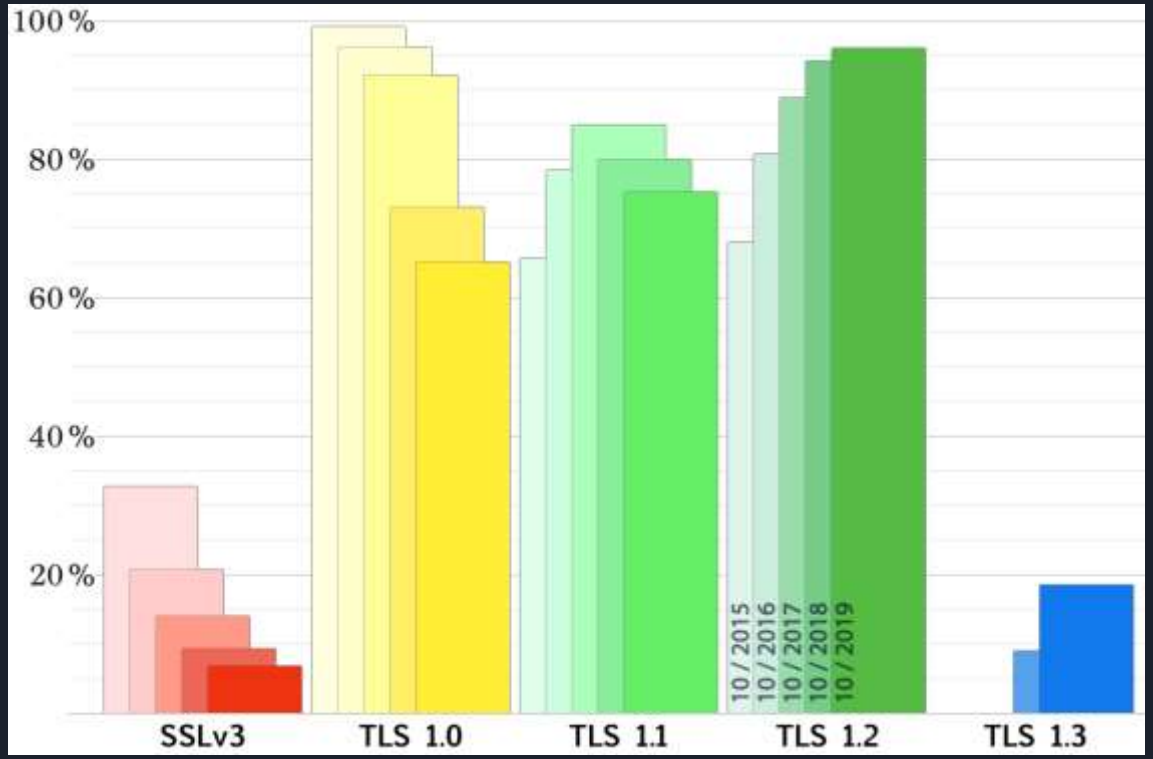
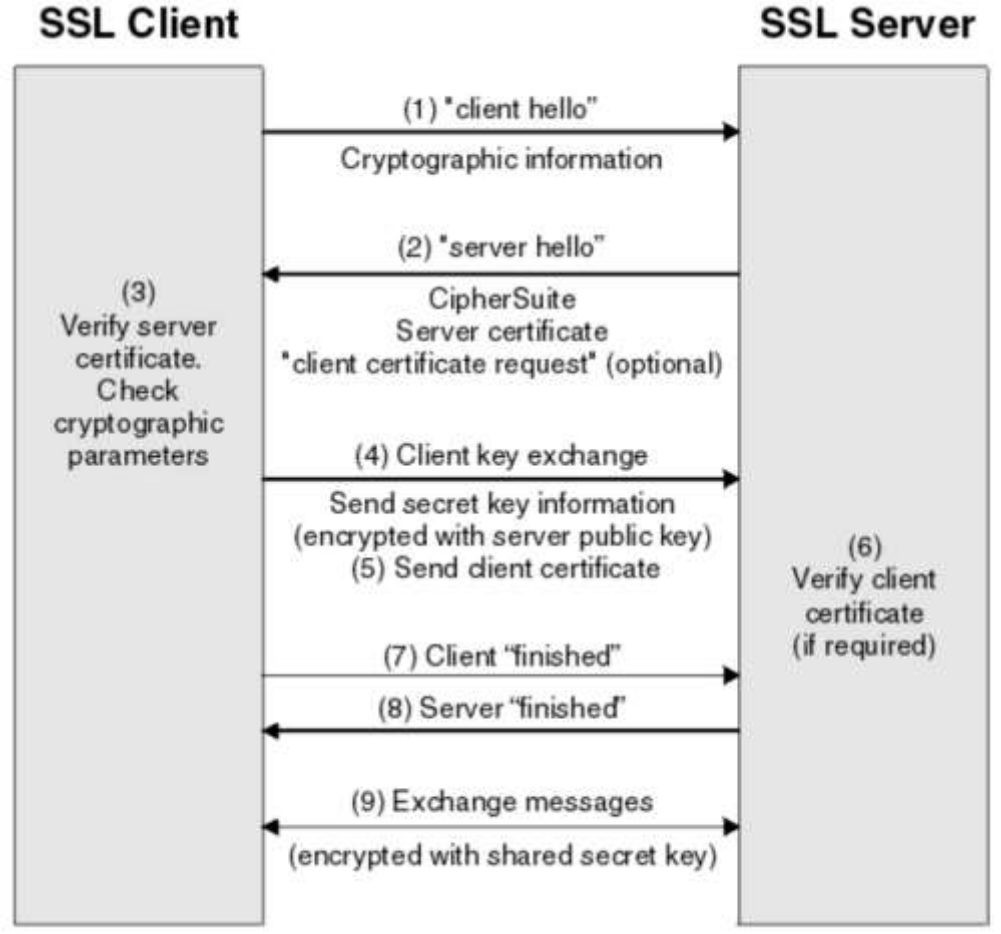
Analysis and Experimentation of TLS 1.3

OUZINEB Sohaïb, ASSOMANY Marvin, PROIETTI Harith

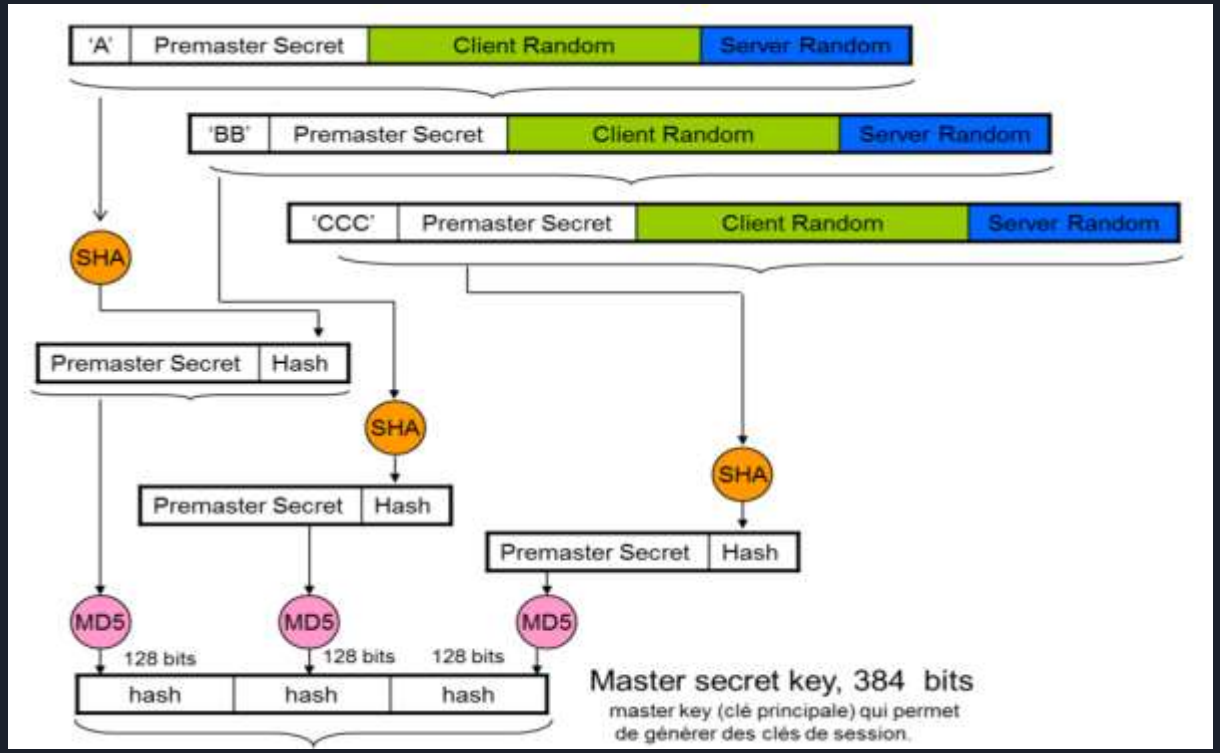
Introduction



TLS 1.2 ...


SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server
certificate.
Check
cryptographic
parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client
certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Construction of the master secret

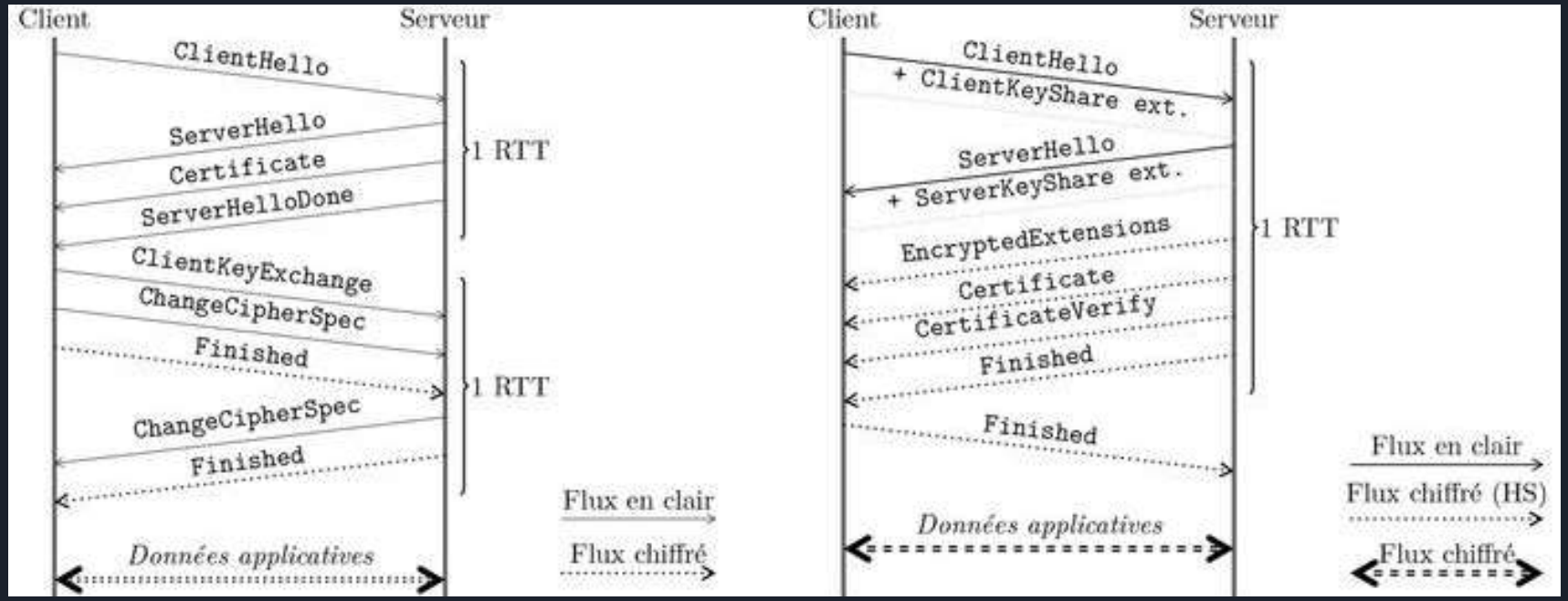


… to TLS 1.3

- Ssuppression of all cryptographic suites deemed too weak
- Reduced handshake time
- 0-RTT
- Compression removed

```
– TLS_AES_128_GCM_SHA256
– TLS_AES_256_GCM_SHA384
– TLS_CHACHA20_POLY1305_SHA256
– TLS_AES_128_CCM_SHA256
– TLS_AES_128_CCM_8_SHA256
```

TLS1.2 vs TLS 1.3



Capturing a session establishment - Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.1.117	172.16.1.130	TCP	74	34152 → 4433 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=269767201 TSecr=0 WS=128
2	0.009143	172.16.1.130	172.16.1.117	TCP	74	4433 → 34152 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=269762153 TSecr=26976
3	0.010254	172.16.1.117	172.16.1.130	TCP	66	34152 → 4433 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=269767212 TSecr=269762153
4	0.010885	172.16.1.117	172.16.1.130	TLSv1.3	301	Client Hello
5	0.012012	172.16.1.130	172.16.1.117	TCP	66	4433 → 34152 [ACK] Seq=1 Ack=236 Win=30080 Len=0 TSval=269762156 TSecr=269767212
6	0.014010	172.16.1.130	172.16.1.117	TLSv1.3	1139	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application
7	0.015756	172.16.1.117	172.16.1.130	TCP	66	34152 → 4433 [ACK] Seq=236 Ack=1074 Win=31360 Len=0 TSval=269767217 TSecr=269762158
8	0.017415	172.16.1.117	172.16.1.130	TLSv1.3	146	Change Cipher Spec, Application Data
9	0.017879	172.16.1.117	172.16.1.130	TLSv1.3	124	Application Data, Application Data

```
▾ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 230
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 226
      Version: TLS 1.2 (0x0303)
      Random: 8147c166d51bfa4bb5e02ae1a787131d11aac6cefc7fab94...
      Session ID Length: 32
      Session ID: 6f9d07f1953e99d8f36d97ee190b061bf4840bb68fccdee2...
      Cipher Suites Length: 8
    ▸ Cipher Suites (4 suites)
```

```
0000  b8 27 eb 45 99 91 00 0c  29 dd 61 7a 08 00 45 00   .'.E....).az..E.
0010  01 1f 85 0b 40 00 40 06  59 b6 ac 10 01 75 ac 10   ....@.@.Y....u..
0020  01 82 85 68 11 51 7f 2a  d7 29 ec 60 a9 af 80 18   ...h.Q.*.).`....
0030  00 e5 2a 04 00 00 01 01  08 0a 10 14 52 2c 10 14   ..*.........R,..
0040  3e 69 16 03 01 00 e6 01  00 00 e2 03 03 81 47 c1   >i..........G.
0050  66 d5 1b fa 4b b5 e0 2a  e1 a7 87 13 1d 11 aa c6   f...K..*........
0060  ce fc 7f ab 94 c8 62 ad  c8 ab 0c dd cb 20 6f 9d   ......b...... o.
0070  07 f1 95 3e 99 d8 f3 6d  97 ee 19 0b 06 1b f4 84   ...>...m........
0080  0b b6 8f cc de e2 d0 2d  6b 0c 1f 52 53 13 00 08   .......-k..RS...
0090  13 02 13 03 13 01 00 ff  01 00 00 91 00 00 00 0c   ................
00a0  00 0a 00 00 07 64 6f 67  66 69 73 68 00 0b 00 04   .....dogfish....
00b0  03 00 01 02 00 0a 00 0c  00 0a 00 1d 00 17 00 1e   ................
```

Hello Customer

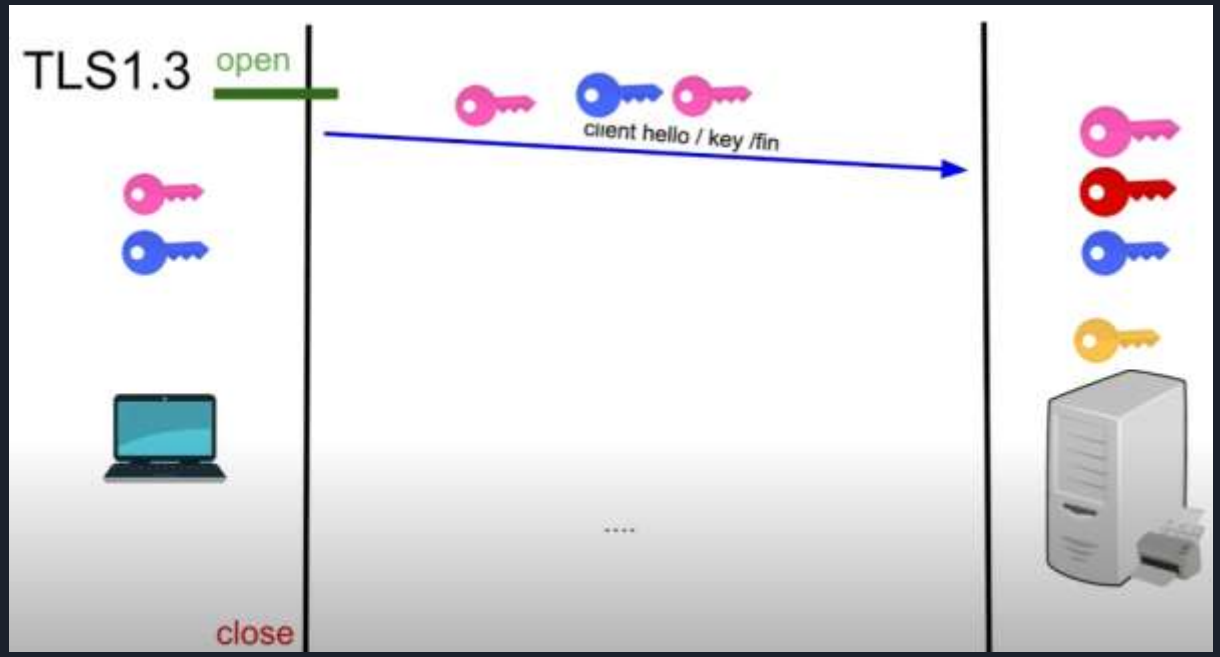
- Random client
- Cipher Suites

```
Cipher Suites (4 suites)
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
```

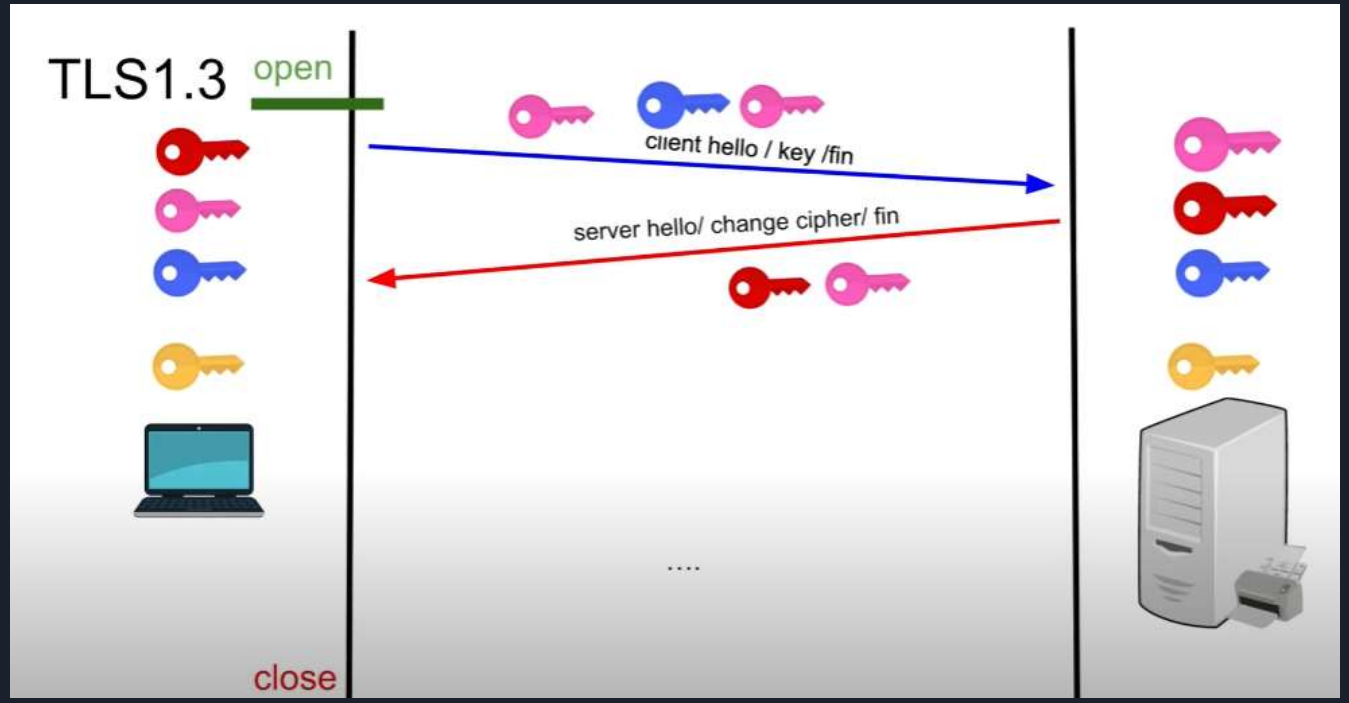
- Key_share *
- Signature_algorithms *
- pre_shared_key *
- psk_key_exchanges_modes *
- Extensions

```
Extensions Length: 145
Extension: server_name (len=12)
Extension: ec_point_formats (len=4)
Extension: supported_groups (len=12)
Extension: SessionTicket TLS (len=0)
Extension: encrypt_then_mac (len=0)
Extension: extended_master_secret (len=0)
Extension: signature_algorithms (len=30)
Extension: supported_versions (len=7)
Extension: psk_key_exchange_modes (len=2)
Extension: key_share (len=38)
```

Key Share



Key Share


TLS1.3
open
client hello / key /fin
server hello/ change cipher/ fin
....
close

Server Hello

No.	Time	Source	Destination	Protocol	Length	Info
2	0.009143	172.16.1.130	172.16.1.117	TCP	74	4433 → 34152 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=269762153 TSecr=26976
3	0.010254	172.16.1.117	172.16.1.130	TCP	66	34152 → 4433 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=269767212 TSecr=269762153
4	0.010885	172.16.1.117	172.16.1.130	TLSv1.3	301	Client Hello
5	0.012012	172.16.1.130	172.16.1.117	TCP	66	4433 → 34152 [ACK] Seq=1 Ack=236 Win=30080 Len=0 TSval=269762156 TSecr=269767212
6	0.014010	172.16.1.130	172.16.1.117	TLSv1.3	1139	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application
7	0.015756	172.16.1.117	172.16.1.130	TCP	66	34152 → 4433 [ACK] Seq=236 Ack=1074 Win=31360 Len=0 TSval=269767217 TSecr=269762158
8	0.017415	172.16.1.117	172.16.1.130	TLSv1.3	146	Change Cipher Spec, Application Data
9	0.017879	172.16.1.117	172.16.1.130	TLSv1.3	124	Application Data, Application Data
10	0.024132	172.16.1.130	172.16.1.117	TLSv1.3	321	Application Data

```
Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 118
    Version: TLS 1.2 (0x0303)
    Random: 3964dbec5022bfbd0783a15f8fd02518c8cf05be901c389b...
    Session ID Length: 32
    Session ID: 6f9d07f1953e99d8f36d97ee190b061bf4840bb68fccdee2...
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Compression Method: null (0)
    Extensions Length: 46
  ▸ Extension: supported_versions (len=2)
  ▸ Extension: key_share (len=36)
TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
```

```
0000  b8 27 eb 45 99 91 00 0c  29 ee 6c 65 08 00 45 00   .'.E....).le..E.
0010  04 65 97 c5 40 00 40 06  43 b6 ac 10 01 82 ac 10   .e..@.@.C.......
0020  01 75 11 51 85 68 ec 60  a9 af 7f 2a d8 14 80 18   .u.Q.h.`...*....
0030  00 eb 48 48 00 00 01 01  08 0a 10 14 3e 6e 10 14   ..HH........>n..
0040  52 2c 16 03 03 00 7a 02  00 00 76 03 03 39 64 db   R,....z...v..9d.
0050  ec 50 22 bf bd 07 83 a1  5f 8f d0 25 18 c8 cf 05   .P"....._..%....
0060  be 90 1c 38 9b 8a 28 46  39 e3 7c db 66 20 6f 9d   ...8..(F9.|.f o.
0070  07 f1 95 3e 99 d8 f3 6d  97 ee 19 0b 06 1b f4 84   ...>...m........
0080  0b b6 8f cc de e2 d0 2d  6b 0c 1f 52 53 13 13 02   .......-k..RS...
0090  00 00 2e 00 2b 00 02 7f  1c 00 33 00 24 00 1d 00   ....+.....3.$...
00a0  20 ee d2 11 97 9a c7 94  1f dd de 11 1c d4 f5 b9    ...............
00b0  81 ec d0 69 bb f9 e3 ef  f2 b4 2d 01 cb 6b 9e 57   ...i......-..k.W
```

Server Hello

- Key share *
- pre_shared_key *
- {Encrypted_Extensions}
- {Certificate Request *}
- {Certificate *}
- {CertificateVerify *}
- {Finished}
- [Application Data *]

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  | Exch
                                            + pre_shared_key*  v
                                        {EncryptedExtensions}  ^  Server
                                        {CertificateRequest*}  v  Params
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                                 <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]

              +  Indicates noteworthy extensions sent in the
                 previously noted message.

              *  Indicates optional or situation-dependent
                 messages/extensions that are not always sent.

              {} Indicates messages protected using keys
                 derived from a [sender]_handshake_traffic_secret.

              [] Indicates messages protected using keys
                 derived from [sender]_application_traffic_secret_N.

               Figure 1: Message Flow for Full TLS Handshake
```

Certificate Verify - Server Finished

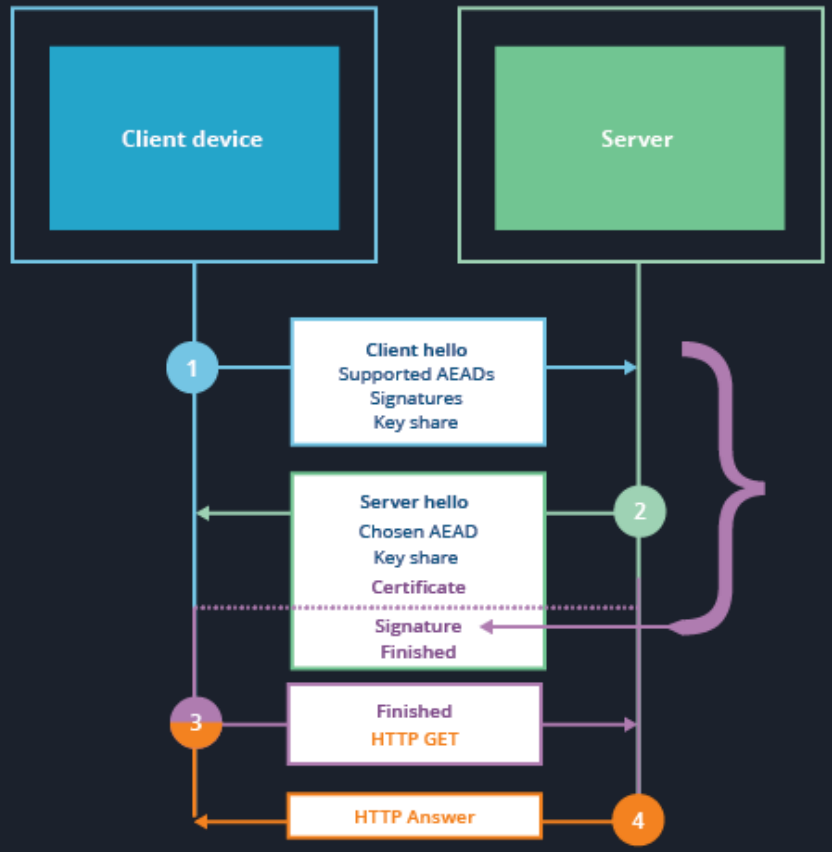
GOAL: To ensure the integrity of the handshake and the possession of certain keys

CertificateVerify:

Signature on the whole handshake using the private key corresponding to the public key of the certificate.

Finished:

MAC on the handshake using a context dependent base_key, allows authentication in PSK mode



Last step

Optional client authentication unless required by the server

Client Finished

Application Data

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  | Exch
                                            + pre_shared_key*  v
                                        {EncryptedExtensions}  ^  Server
                                        {CertificateRequest*}  v  Params
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                                 <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <-------->  [Application Data]

              +  Indicates noteworthy extensions sent in the
                 previously noted message.

              *  Indicates optional or situation-dependent
                 messages/extensions that are not always sent.

              {} Indicates messages protected using keys
                 derived from a [sender]_handshake_traffic_secret.

              [] Indicates messages protected using keys
                 derived from [sender]_application_traffic_secret_N.

               Figure 1: Message Flow for Full TLS Handshake
```

Session resumption

Using the PSK

struct {opaque identity <1..2 ^ 16-1>; // label on a PSK
uint32 obfuscated_ticket_age; // age of key obfuscated}
PskIdentity;

opaque PskBinderEntry <32..255>; // HMAC PSK-handshake

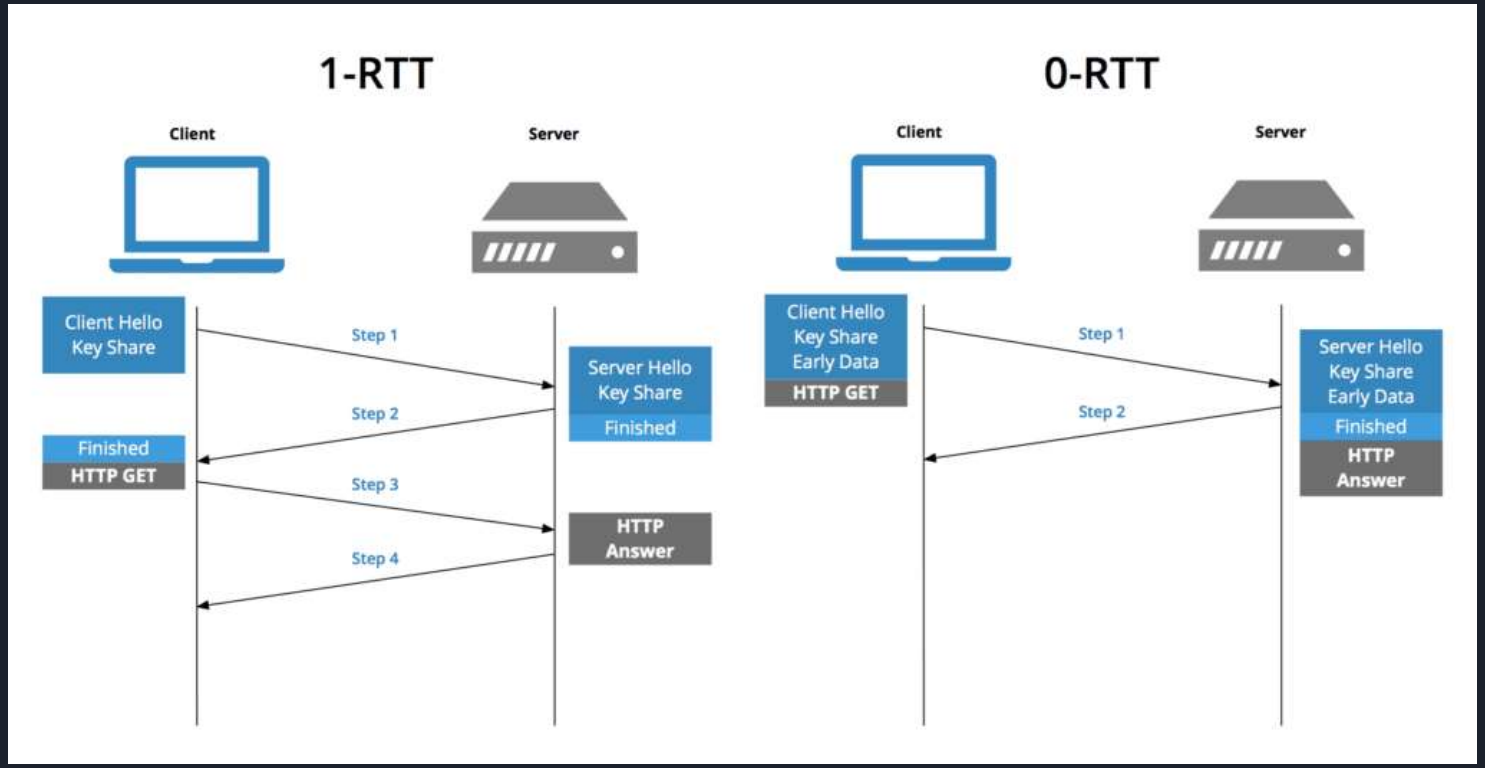
Potentially make a new key share.

The server proves the integrity of the handshake as well as the possession of the PSK through the Finished

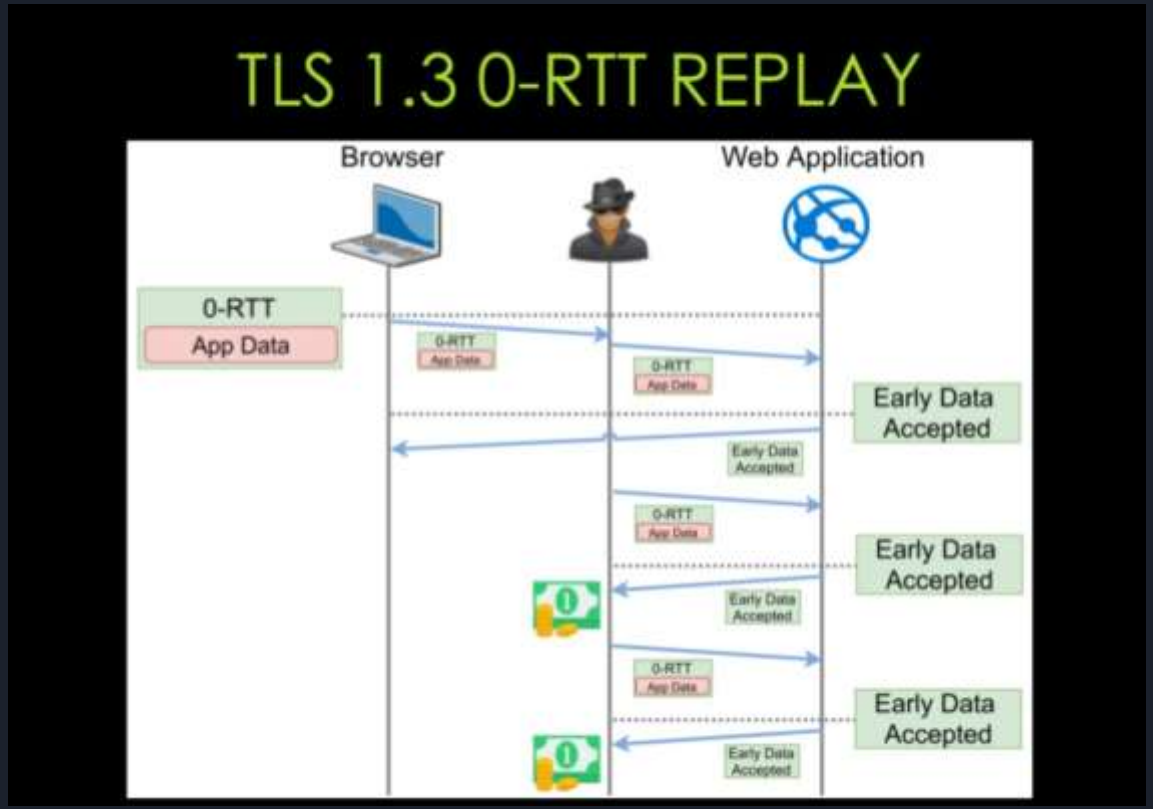
```
       Client                                                  Server

Initial Handshake:
       ClientHello
       + key_share              -------->
                                                       ServerHello
                                                       + key_share
                                             {EncryptedExtensions}
                                             {CertificateRequest*}
                                                    {Certificate*}
                                              {CertificateVerify*}
                                                        {Finished}
                                <--------     [Application Data*]
       {Certificate*}
       {CertificateVerify*}
       {Finished}               -------->
                                <--------     [NewSessionTicket]
       [Application Data]       <------->     [Application Data]


Subsequent Handshake:
       ClientHello
       + key_share*
       + pre_shared_key         -------->
                                                       ServerHello
                                                  + pre_shared_key
                                                      + key_share*
                                             {EncryptedExtensions}
                                                        {Finished}
                                <--------     [Application Data*]
       {Finished}               -------->
       [Application Data]       <------->     [Application Data]
```

Figure 3: Message Flow for Resumption and PSK

0-RTT


1-RTT
Client
Server
Client Hello
Key Share
Step 1
Server Hello
Key Share
Finished
Step 2
Finished
HTTP GET
Step 3
HTTP
Answer
Step 4
0-RTT
Client
Server
Client Hello
Key Share
Early Data
HTTP GET
Step 1
Server Hello
Key Share
Early Data
Finished
HTTP
Answer
Step 2

Attacks on 0-RTT



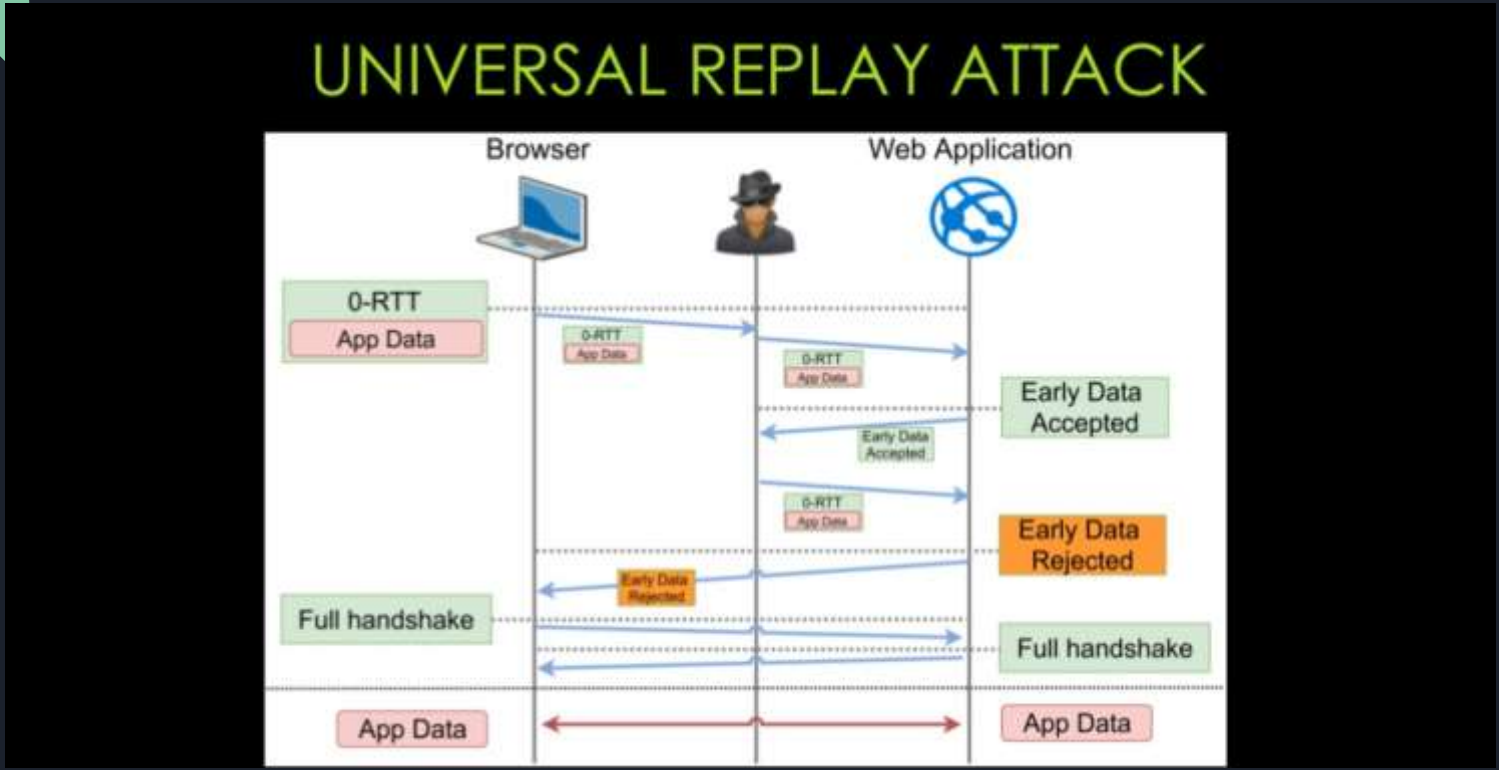
Anti-replay mechanisms: Anti-replay and anti-replay-policy extensions

- Single use Tickets: the server deletes the ticket as soon as it has been used

- Hello Recording Client: PSK binder

- Time stamp: Customer signs the sending time with the PSK

Not enough…


UNIVERSAL REPLAY ATTACK
Browser
Web Application
0-RTT
App Data
0-RTT
App Data
0-RTT
App Data
Early Data
Accepted
Early Data
Accepted
0-RTT
App Data
Early Data
Rejected
Early Data
Rejected
Full handshake
Full handshake
App Data
App Data

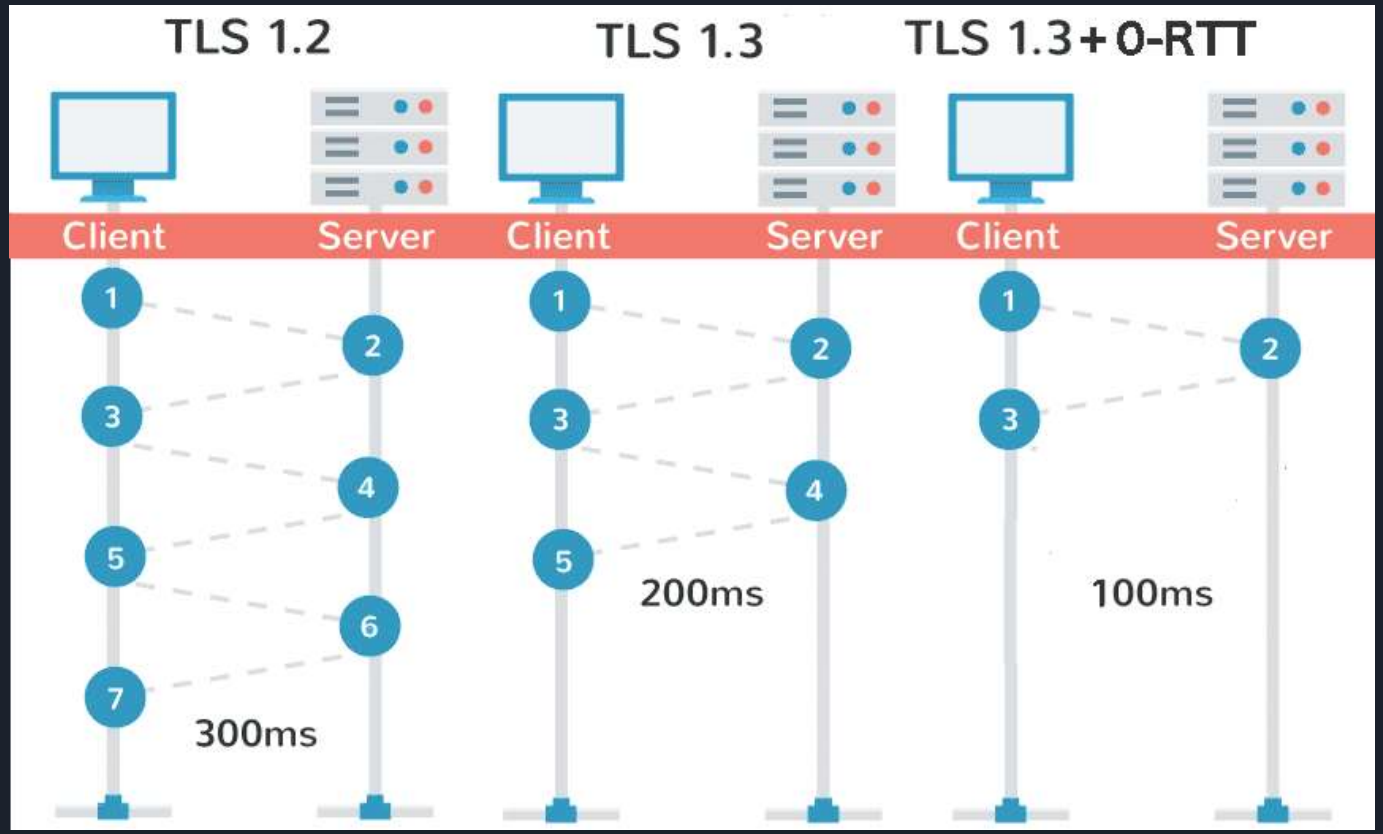
Possible solution: require_rejected_reason extension

- Ask the server to request the cause of the rejection
- As soon as the server detects a replay, it indicates in the Early_data_refused the unique identifier allowing the Client to recognize the 0-RTT which has been replayed
- When the customer receives

Early_data_refused (cause: replay)> Replayed id PSK_binder: XXXXXX

He understands that his 0-RTT has been sent and processed, so he goes on to the next application content, the server can then forget that there has been replay

Benchmarking



root@kali: ~

root@kali: ~

File Actions Edit View Help

root@kali: ~

VBoxClient: Faile

```
root@kali:~# openssl req -newkey rsa:2048 -x509 -out cert.pem -keyout key.p
em
Generating a RSA private key
.....................................................................
.....................................................+++++
........+++++
writing new private key to 'key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:YV
Locality Name (eg, city) []:VE
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TE
Organizational Unit Name (eg, section) []:SR2I
Common Name (e.g. server FQDN or YOUR name) []:Local_Server
Email Address []:email@adress.com
root@kali:~#
```

```
.....+++++
.....+++++
writing new private key to 'key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:YV
Locality Name (eg, city) []:VE
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TE
Organizational Unit Name (eg, section) []:SR2I
Common Name (e.g. server FQDN or YOUR name) []:Local_Server
Email Address []:email@adress.com
root@kali:~# openssl s_server -accept 4433 -cert cert.pem -key key.pem -www
Enter pass phrase for key.pem:
Using default temp DH parameters
ACCEPT
```

Prise ... root@... root - ... 11:40 AM 34%

root@kali: ~

File Actions Edit View Help

root@kali: ~

```
total=0
for i in $(seq 10); do
 milliseconds=$( (time nc 192.168.0.30 4433 ) 2>&1 | sed -n 's/^real,*0m0.0*\([^0]*\)s$/\1/p')
 (( total += milliseconds ))
done
echo $((total / 10))
~
```

Afin de produire une mesure moins bruitée, on peut exécuter le script **bash** suivant pour faire plusieurs connexions et moyenner le temps obtenu :

```
root@kali:~# ./rtt.sh
7
root@kali:~# ./tls12_connect_time.sh
60
root@kali:~# ./tls13_connect_time.sh
46
root@kali:~#
```

Modeling of several clients

```
root@kali: ~/nghttp2
File  Actions  Edit  View  Help
root@kali: ~/nghttp2
root@kali:~/nghttp2# nghttpd -v 8080 ~/key.pem ~/cert.pem
Enter PEM pass phrase:
IPv4: listen 0.0.0.0:8080
IPv6: listen :::8080
```

root@kali: ~/Downloads/nghttp2/src

File Actions Edit View Help

root@kali: ~/...s/nghttp2/src

```
root@kali:~/Downloads/nghttp2/src# h2load -n1000 -c150 -m50 https://192.168.43.90:8080
starting benchmark ...
spawning thread #0: 150 total client(s). 1000 total requests
TLS Protocol: TLSv1.3
Cipher: TLS_AES_256_GCM_SHA384
Server Temp Key: ECDH P-256 256 bits
Application protocol: h2
progress: 10% done
progress: 20% done
progress: 30% done
progress: 40% done
progress: 50% done
progress: 60% done
progress: 70% done
progress: 80% done
progress: 90% done
progress: 100% done

finished in 1.57s, 637.59 req/s, 118.86KB/s
requests: 1000 total, 1000 started, 1000 done, 0 succeeded, 1000 failed, 0 errored, 0 timeout
status codes: 0 2xx, 0 3xx, 1000 4xx, 0 5xx
traffic: 186.43KB (190900) total, 17.87KB (18300) headers (space savings 85.70%), 147.46KB (151
000) data
                     min         max         mean         sd        +/- sd
time for request:   265.49ms      1.13s     643.27ms    195.11ms    69.40%
time for connect:   292.43ms    774.43ms    499.25ms    125.55ms    62.67%
time to 1st byte:   560.68ms      1.56s       1.14s    285.78ms    54.00%
req/s           :       3.92     12.48        6.28       1.96     67.33%
root@kali:~/Downloads/nghttp2/src#
```

Conclusion

Thank you for your attention !