Q1.

TLS 1.3 include 3 main parts:

a. Client Hello
b. Server Hello
c. Check Certification

Step 1: Client Hello

When a client first connects to a server, it is required to send the “ClientHello” as its first TLS message. The client will also send a “ClientHello” when the server has responded to its ClientHello with a HelloRetryRequest.

TLS 1.3 ClientHello messages always contain extensions, it need to "supported_versions" at least, otherwise, they will be interpreted as TLS 1.2 ClientHello messages. TLS 1.3 servers will check the extension part after the Compression Methods. The handshake will stop and send “decode error” if there is no data after the Compress Method or it contain a extension part but not contain data.

The ClientHello provides information including the following:

a. Protocol Versions that the client can support: In TLS 1.3, the client indicates its version preferences in the "supported_versions" extension and the legacy version field must be set to 0x0303, which is the version number for TLS 1.2. TLS 1.3 ClientHellos are identified as having a legacy_version and a supported_versions extension present with 0x0304 as the highest version indicated therein.
b. Client Random Data: 32 bytes generated by a secure random number generator.
c. Legacy Session ID: A client which has a cached session ID set by a pre-TLS 1.3 server should set this field to that value. A client not offering a pre-TLS 1.3 session must generate a new 32-byte value.
d. A list of cipher suites that the client supports: list of the symmetric cipher options supported by the client. The field is consisted of the list of Key exchange, signature algorithm, encryption algorithm and hash algorithm.
e. Compression Methods: For every TLS 1.3 ClientHello, this vector must contain exactly one byte and set to zero. The server will abort the handshake with "illegal_parameter" alert if the ClientHello is received with any other value.
f. Extension information: Clients request extended functionality from servers by sending data in the extensions field. This field is mandatory. It can include key share, signature algorithms, key exchange mode, pre-shared key and another extension information.

- In the key_share field, it saves the public key of the encryption, which is the client's public key parameters.
- In the signature algorithm field, the client provides the signature algorithm and lets the server choose.
- In the key exchange mode field, it saves the exchange mode that client and sever used.
- In the pre-shared key field, it saves the PSK identity and related information.

Step2: Server Hello

The server will send “Server Hello” in response to a ClientHello message to proceed with the handshake if it is able to negotiate an acceptable set of handshake parameters based on the ClientHello.

The ServerHello provides information including the following:

a. Protocol Version: In TLS 1.3, the TLS server indicates its version using the "supported_versions" in extension field. Then the legacy version field is set to the version number for TLS 1.2 = 0x0303.
b. Random Data: 32 bytes generated by a secure random number generator.
c. Session ID echo: The contents of the client's Session ID. A client will abort the handshake with "illegal_parameter" alert if it receives Session ID echo that does not match the ID it included in the clientHello.
d. Cipher Suite: The single cipher suite selected by the server from the list in ClientHello.cipher_suites.
e. Compression Method: A single byte which must have the value 0.
f. Extension: The ServerHello must include extensions. All TLS 1.3 ServerHello messages must contain the "supported_versions" extension. Current ServerHello messages additionally contain either the "pre_shared_key" extension or/and the "key_share" extension. Other extensions are sent separately in the Encrypted Extensions message.
    - The field in server Extension is similar to the client one. Client supply the different method which server can choose, server will choose the method and reply to the client.

Step 3: Check Certification

Now, client and server both have client key share, server key share, client random and server random. They can use ECDHE algorithm to get the Pre-Master (client key share + server key share), then use client random + server random + Pre-Master to get the Master Secret.

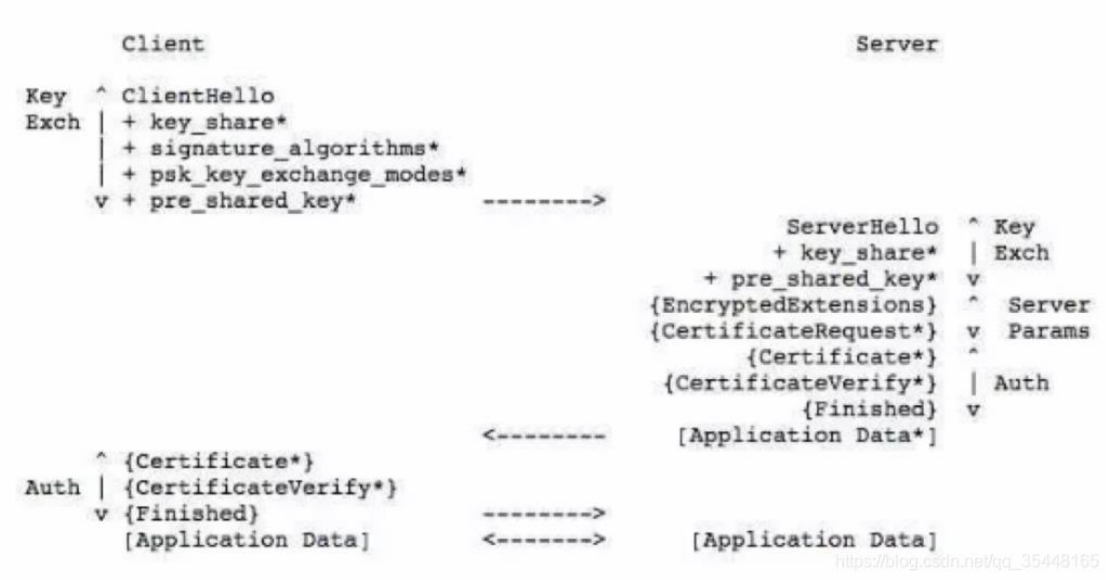
Server will calculate Pre-Master and Master Secret , and send “Change Cipher Spec” to Client and all the following steps are encrypted. Then, Server sends the Encrypted Extensions and it send the server certificate and

CertificateVerify that was the signature of previous handshake message by the private key of the certificate.

Finally, Server send “Finished” to indicate the handshake from server to client is over.

At the client side, Client verify the certificate and signature. If those are verified, client calculate Pre-Master and Master Secret, then sends its Change Cipher Spec and Finished message.

Now, the whole handshake step is over, and the client can now start communicating with the server over HTTPS.

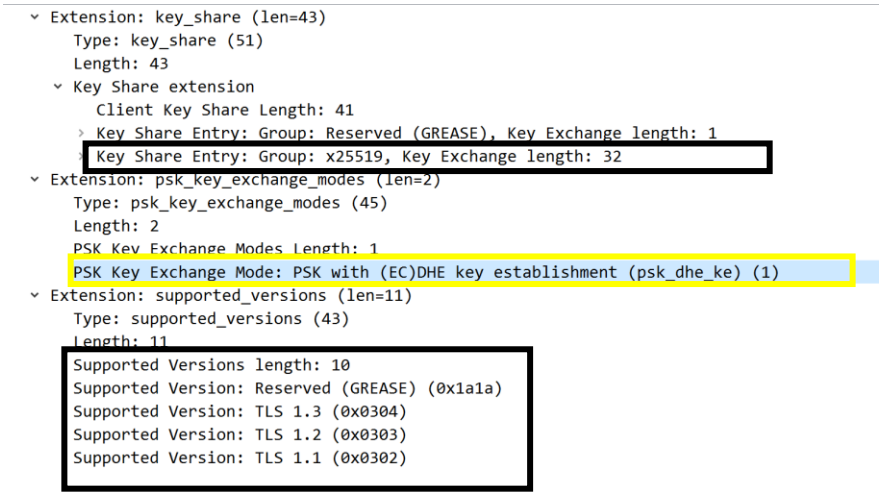


+ Indicates noteworthy extensions sent in the previously noted message.
* Indicates optional or situation-dependent messages/extensions that are not always sent.
{ } Indicates messages protected using keys derived from a [sender]_handshake_traffic_secret.
[ ] Indicates messages protected using keys derived from [sender]_application_traffic_secret_N.

Q2.

```
22609 13.722405    192.168.2.46      157.240.18.15     TLSv1.3   571 Client Hello
22613 13.745244    157.240.18.15     192.168.2.46      TLSv1.3  1446 Server Hello, Change Cipher Spec, E
22614 13.745244    157.240.18.15     192.168.2.46      TLSv1.3  1051 Compressed Certificate, Certificate
22620 13.758182    192.168.2.46      157.240.18.15     TLSv1.3   118 Change Cipher Spec, Finished
```

```
▾ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 508
      Version: TLS 1.2 (0x0303)
      Random: 28fab23e861dd79276c941aacb1865cd14e0faf40180c3846a651f1e9a1a5ec9
      Session ID Length: 32
      Session ID: deca590d9f0d5419b51c5ab357b968b172ebd02931f427733a91f03c2ef2cf60
      Cipher Suites Length: 32
    ▾ Cipher Suites (16 suites)
        Cipher Suite: Reserved (GREASE) (0x5a5a)
        Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
        Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
        Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
      Compression Methods Length: 1
    ▾ Compression Methods (1 method)
        Compression Method: null (0)
      Extensions Length: 403
    › Extension: Reserved (GREASE) (len=0)
    › Extension: server_name (len=28)
    › Extension: extended_master_secret (len=0)
    › Extension: renegotiation_info (len=1)
    › Extension: supported_groups (len=10)
    › Extension: ec_point_formats (len=2)
    ▾ 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)
```

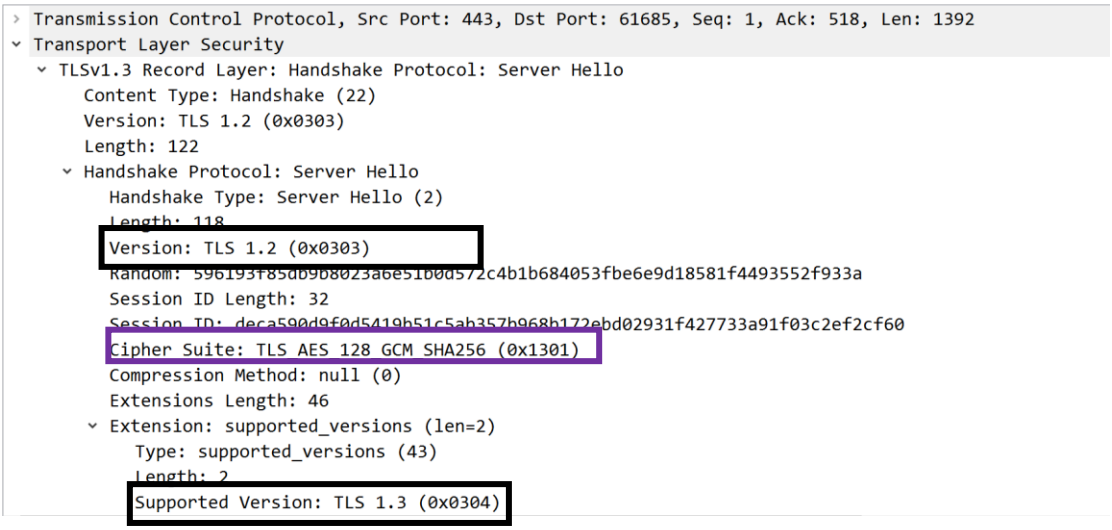


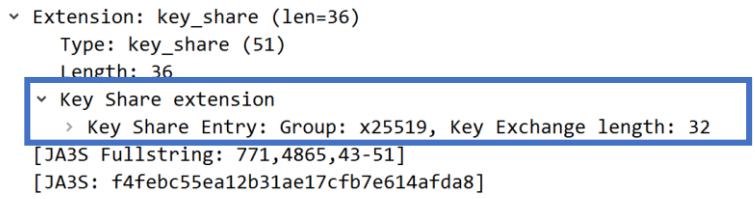
Inherit:

1. Version: the legacy version field of client must be set to 0x0303, which is the version number for TLS 1.2.
2. Session ID: A client not offering a pre-TLS 1.3 session must generate a new 32-byte value
3. Cipher Suite: TLS_AES_128_GCM_SHA256

   Key exchange, signature algorithm, encryption algorithm and hash algorithm (from left to right)
4. Compress Method: For every TLS 1.3 ClientHello, this vector must contain exactly one byte and set to zero.
5. Signature Algorithm: the client provides the signature algorithm and lets the server choose.
6. Key share: the curve is x25519
7. PSK_key_Exchange_ mode is (EC)DHK, so it has key share.

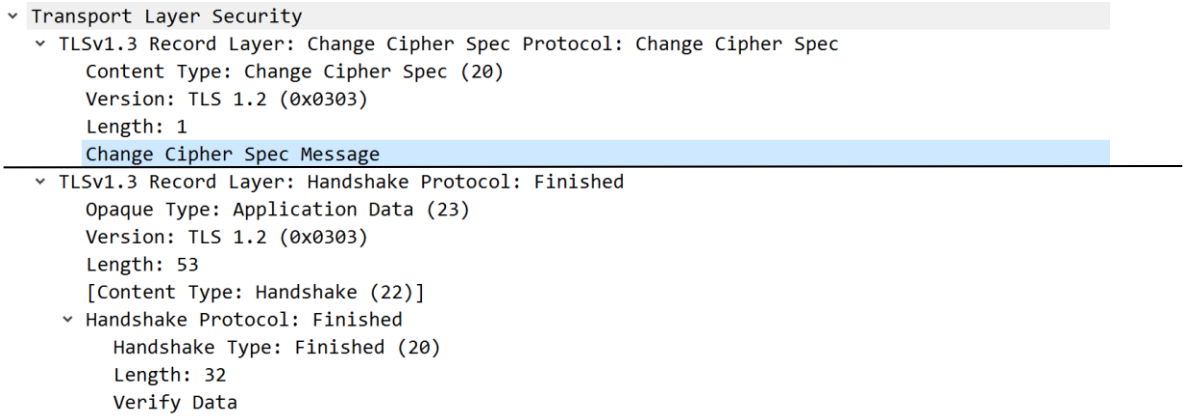
g. Supported Version: supported_versions extension present with 0x0304 as the highest version indicated therein.





1. Legacy version: In TLS 1.3, the TLS server indicates its version using the "supported_versions" in extension field. Then the legacy version field is set to the version number for TLS 1.2 = 0x0303
2. Cipher Suite: Server choose TLS_AES_128_GCM_SHA256 to do the encrypt.
3. Supported Version: All TLS 1.3 ServerHello messages must contain the "supported_versions" extension, and present with 0x0304 as the highest version indicated therein.
4. Key share extension: server key share is x25519

1. Certificate: include the certificate message, length and size.
2. Signature: use signature algorithm to sign the handshake message, the private key is the private key of the certificate



At the client side, Client verify the certificate and signature. If those are verified, client calculate Pre-Master and Master Secret, then sends its Change Cipher Spec and Finished message.