SSL/TLS

SSL/TLS: Background

- SSL was originated by Netscape in 1994.
- TLSv1.0 ~ SSLv3.1, and very close to SSLv3.
- TLSv1.2 is specified by RFC 5246.
- TLSv1.3 is specified by RFC 8446.

SSL/TLS: Applications

- Web security (or HTTPS)
- VPN (e.g., CISCO AnyConnect)
- Internet of Things
- E-Commerce
- 5G

……

khoa@capa.uow.edu.au:22 - Bitvise xterm

```
khoa@capa:~$ openssl s_client -connect uow.edu.au:443
CONNECTED(00000005)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
verify return:1
depth=0 C = AU, ST = New South Wales, L = Wollongong, O = University of Wollongong, CN = uow.edu.au
verify return:1
---
Certificate chain
 0 s:C = AU, ST = New South Wales, L = Wollongong, O = University of Wollongong, CN = uow.edu.au
   i:C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
 1 s:C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIGxzCCBa+gAwIBAgIQBG8N+ZLOgXHbrACNfgVP3DANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMSkwJwYDVQQDEyBE
aWdpQ2VydCBUTFMgUlNBIFNIQTI1NiAyMDIwIENBMTAeFw0yMjAyMjQwMDAwMDBa
Fw0yMzAyMjQyMzU5NTlaMHQxCzAJBgNVBAYTAkFVMRgwFgYDVQQIEw9OZXcgU291
dGggV2FsZXMxEzARBgNVBAcTCldvbGxvbmdvbmcxITAfBgNVBAoTGFVuaXZlcnNp
dHkgb2YgV29sbG9uZ29uZzETMBEGA1UEAxMKdW93LmVkdS5hdTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAKtbU5FLDAyiaQOvLRPLB/0cUpwTrkjOdVTt
qAX5vWbt1wYuYeFXVX64bXAajH2TqNbDfYxkoZ51ZCddc7j3Zlgc/fKrnyFnVPfz
fciovvEJ9Jfv42UtzfWN65cfi8YOPk5awIUPEiTh3UP+y8YSuGkcwxivAz6t5868
fwX1K18Dh5TomdVPbLASQdsNl/aNVCpNcc5XyRo8k2jLu1DHNWxM3ajYf7Md9eUX
Y3m0utAVyf8+5Qw8e9mn6EidHab21NoMAPiM+xy32GtUN0uxvnNk/s5MGnRwsCuS
ubqR4RaBgu7ATQjB7vDu2tl1n82r3hqtaXwdHiHHwW5bXK8XmyMCAwEAAaOCA3gw
ggN0MB8GA1UdIwQYMBaAFLdrouqoqoSMeeq02g+YssWVdrn0MB0GA1UdDgQWBBTU
dEkdcSTqaCpG5pfDTW6fjQsbmDAjBgNVHREEHDAaggp1b3cuZWR1LmF1ggwqLnVv
dy51ZHUuYXUwDgYDVR0PAQH/BAQDAgWgMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggr
BgEFBQcDAjCBjwYDVR0fBIGHMIGEMECgPqA8hjpodHRwOi8vY3JsMy5kaWdpY2Vy
dC5jb20vRGlnaUNlcnRTFNSU0FTSEEyNTYyMDIwQ0ExLTQuY3JsMECgPqA8hjpo
dHRwOi8vY3JsNC5kaWdpY2VydC5jb20vRGlnaUNlcnRUFNSU0FTSEEyNTYyMDIw
Q0ExLTQuY3JsMD4GA1UdIAQ3MDUwMwYGZ4EMAQICMCkwJwYIKwYBBQUHAgEWG2h0
dHA6Ly93d3cuZGlnaWNlcnQuY29tL0NQUzB/BggrBgEFBQcBAQRzMHEwJAYIKwYB
BQUHMAGGGGh0dHA6Ly9vY3NwLmRpZ2ljZXJ0LmNvbTBJBggrBgEFBQcwAoY9aHR0
cDovL2NhY2VydHMuZGlnaWNlcnQuY29tL0RpZ2lDZXJ0VExTUlNBU0hBMjU2MjAy
MENBMS0xLmNydDAJBgNVHRMEAjAAMIIBfgYKKwYBBAHWeQIEAgSCAW4EggFqAWgA
dwCt9776fP8QyIudPZwePhhqtGcpXc+xDCTKhYY069yCigAAAX8uDbEFAAAEAwBI
MEYCIQC+euD4ByGUTRp8IcnVCtXV6o9AHYQODFjbFIKz8yjIfAIhANFZaiiYe7Sw
d+qxeLpVt5Prge0p244GA0iRb8Rn401vAHYANc8ZG7+xbFe/D61MbULLu7YnICZR
6j/hKu+oA8M71kwAAAF/Lg2w+QAABAMARzBFAiEAyjtZlUKEIdcLk2i4HE+DPCz0
Hx38+k2xoYT+5DEBfbgCIEZ4egYH8c108tcqVc2EVTRoRFlFgxA7OTsTWpy9PU7e
AHUAs3N3B+GEUPhjhtYFqdwRCUp5LbFnDAuH3PADDnk2pZoAAAF/Lg2xKwAABAMA
RjBEAiAmmaSvqeQ8YD5CGHXq6fZvH9jJ2aSQynRJS9gdAcHfmQIgWSMDiaK1slJh
W5nKNvsz0fmtj7BTflx0Uan898dBusEwDQYJKoZIhvcNAQELBQADggEBAB49jg7v
L8KG3NSZn8RcmckCpealEzzeReAqwCjMuxaqYh0pkx9bMz/ezC53F14VJWh11/yJ
Me8whC1lpeauQ0HWd9cWD1sBcthSAXCji3De0vJaVjIC1UJEx9STzUXCNamD+L+A
pJgbTTQHfcokSweUt0IdKfrCgbUXEd8TZqnUUVD8XJ2iBWVLWnPK/5u6Ee1xT9/V
j/8Iq/QRzxIGVjdRxD/91b+xD6c9/2KPH0hEP7kKBKokSOzki29ji543WvMBU1eb
m714NUWjPp8Hm38geEtgNh76tzWPKvCgi2KdFDaf78ZiZnxgC/k/Rd/spUaV9hIv
naOK2usKQk9pJL0=
-----END CERTIFICATE-----
```

khoa@capa.uow.edu.au:22 - Bitvise xterm

```
dHRwOi8vY3JsNC5kaWdpY2VydC5jb20vRGlnaUN1cnRUTFNSU0FTSEEyNTYyMDIw
Q0ExLTQuY3JsMD4GA1UdIAQ3MDUwMwYGZ4EMAQICMCkwJwYIKwYBBQUHAgEWG2h0
dHA6Ly93d3cuZGlnaWN1cnQuY29tL0NQUzB/BggrBgEFBQcBAQRzMHEwJAYIKwYB
BQUHMAGGGGh0dHA6Ly9vY3NwLmRpZ2ljZXJ0LmNvbTBJBggrBgEFBQcwAoY9aHR0
cDovL2NhY2VydHMuZGlnaWN1cnQuY29tL0RpZ21DZXJ0VExTU1NBU0hBMjU2MjAy
MENBMS0xLmNydDAJBgNVHRMEAjAAMIIBfgYKKwYBBAHWeQIEAgSCAW4EggFqAWgA
dwCt9776fP8QyIudPZwePhhqtGcpXc+xDCTKhYY069yCigAAAX8uDbEFAAAEAwBI
MEYCIQC+euD4ByGUTRp8IcnVCtXV6o9AHYQODFjbFIKz8yjIfAIhANFZaiiYe7Sw
d+qxeLpVt5Prge0p244GA0iRb8Rn40lvAHYANc8ZG7+xbFe/D61MbULLu7YnICZR
6j/hKu+oA8M71kwAAAF/Lg2w+QAABAMARzBFAiEAyjtZlUKEIdcLk2i4HE+DPCz0
Hx38+k2xoYT+5DEBfbgCIEZ4egYH8cl08tcqVc2EVTRoRFlFgxA7OTsTWpy9PU7e
AHUAs3N3B+GEUPhjhtYFqdwRCUp5LbFnDAuH3PADDnk2pZoAAAF/Lg2xKwAABAMA
RjBEAiAmmaSvqeQ8YD5CGHXq6fZvH9jJ2aSQynRJS9gdAcHfmQIgWSMDiaK1sLjH
W5nKNvsz0fmtj7BTflx0Uan898dBusEwDQYJKoZIhvcNAQELBQADggEBAB49jg7v
L8KG3NSZn8RcmckCpealEzzeReAqwCjMuxaqYh0pkx9bMz/ezC53Fl4VJWh11/yJ
Me8whC11peauQ0HWd9cWD1sBcthSAXCji3De0vJaVjIC1UJEx9STzUXCNamD+L+A
pJgbTTQHfcokSweUt0IdKfrCgbUXEd8TZqnUUVD8XJ2iBWVLWnPK/5u6Ee1xT9/V
j/8Iq/QRzxIGVjdRxD/91b+xD6c9/2KPH0hEP7kKBKokSOzki29ji543WvMBU1eb
m714NUWjPp8Hm38geEtgNh76tzWPKvCgi2KdFDaf78ZiZnxgC/k/Rd/spUaV9hIv
naOK2usKQk9pJL0=
-----END CERTIFICATE-----
subject=C = AU, ST = New South Wales, L = Wollongong, O = University of Wollongong, CN = uow.edu.au

issuer=C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1

---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 3517 bytes and written 438 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: DEA4C11FF2A8F5C347ED86BCD8711CC86422C00713AB6EFB2478A0D4AA016CDB
    Session-ID-ctx:
    Master-Key: 7CD9069C444B38A65128761CF8F4E2C08DCAEC55CFBE8DD0550BE94EAF8BDC131007E142B900A56AB1ED17FBABEBC8AC
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1662514978
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: yes
---
```

SSL/TLS Architecture

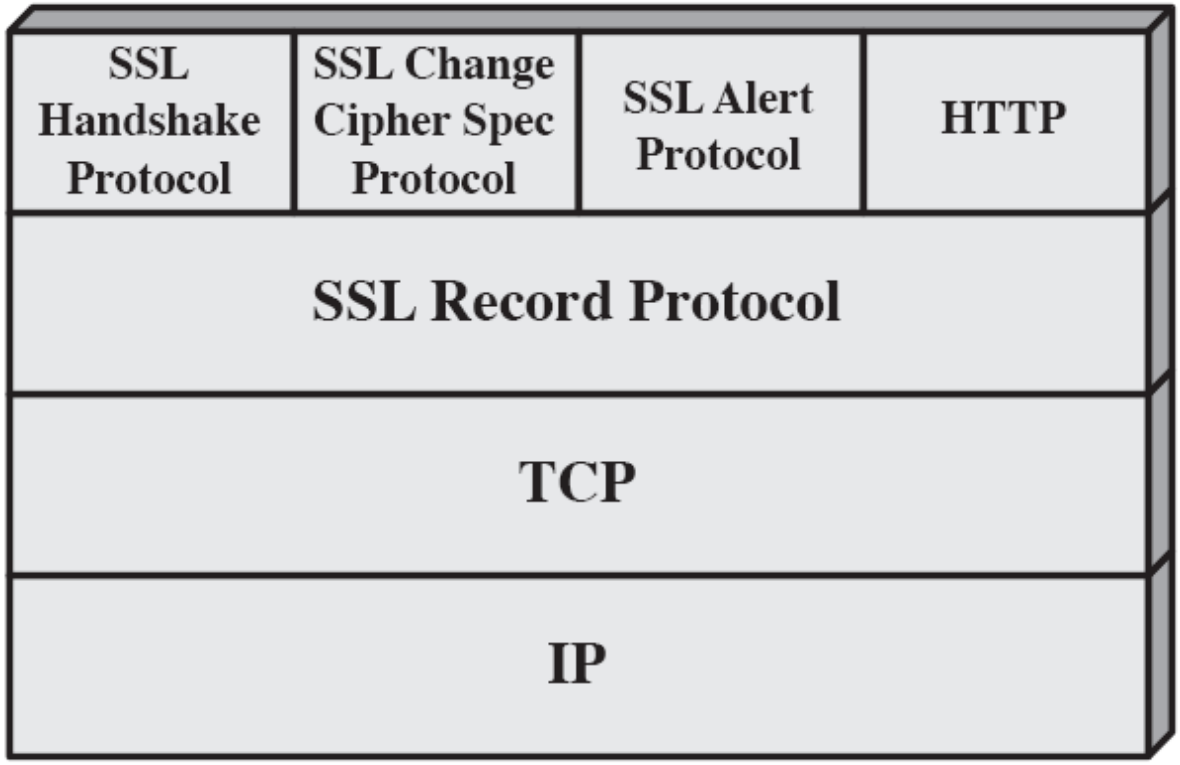


Figure 17.2 SSL Protocol Stack

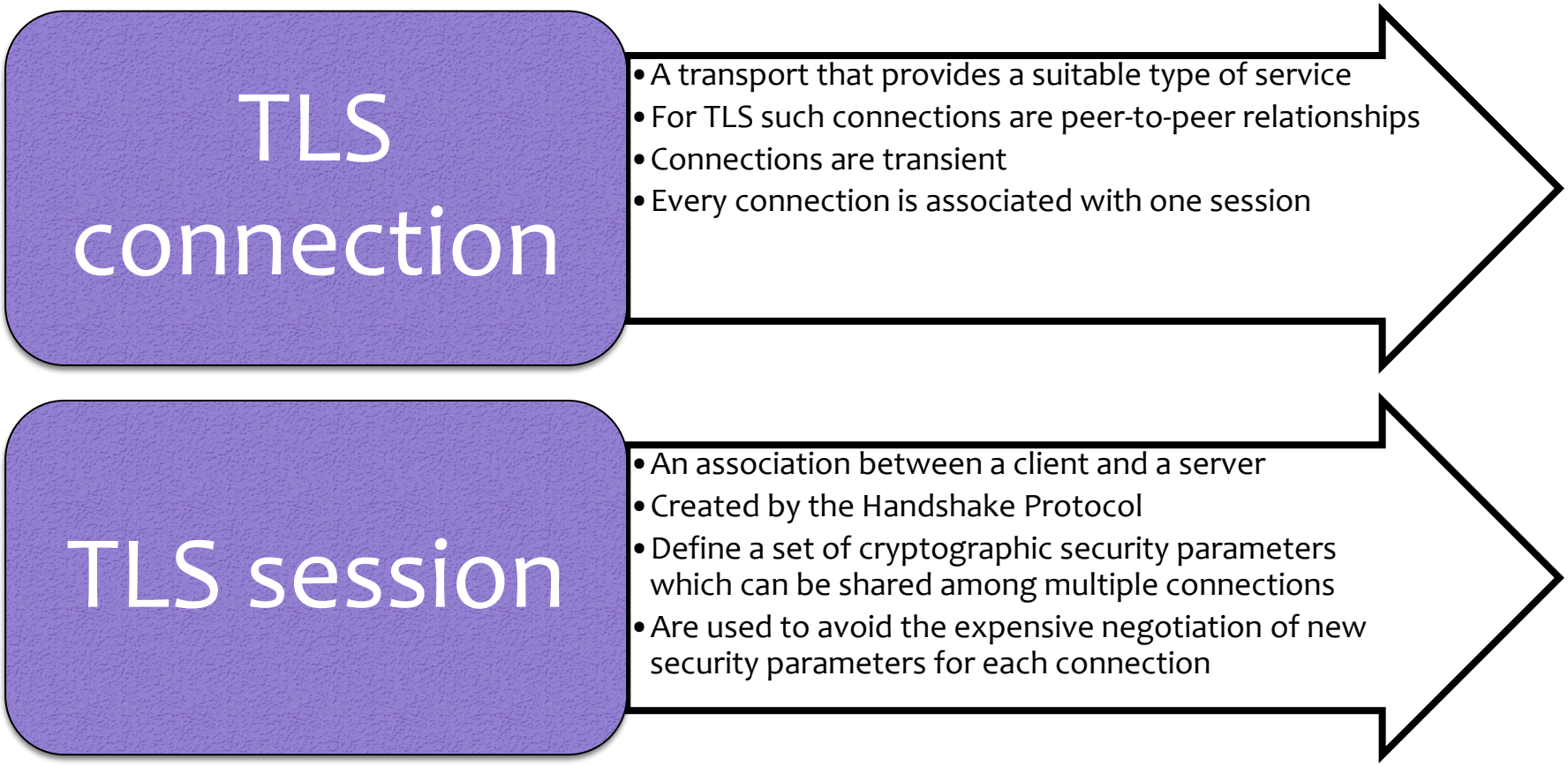
SSL/TLS Architecture

SSL/TLS provides a reliable **end-to-end** security service by using TCP. It has two layers of protocols:

- **Record Protocol:** Provides basic security services to various higher-layer protocols (e.g. HTTP), and supports TLS management protocols.
- **Three management protocols:**

  - **The Change Cipher Spec Protocol**: Updates the cipher suite.
  - **The Alert Protocol:** Conveys TLS-related alerts to the peer entity.
  - **The Handshake Protocol:** Allow the server and client to authenticate each other and to negotiate encryption and MAC algorithms, together with secret keys used in an TLS record.

SSL/TLS Architecture



SSL/TLS Architecture

A **session state** is defined by:

- **Session Identifier.**
- **Peer Certificate**: An X509.v3 certificate of the peer.
- **Compression Method**.
- **Cipher Spec**: Specifies the encryption and hash algorithms.
- **Master Secret**: 48-byte secret shared between the client and server.
- **Is Resumable**: A flag showing if the session can be used to initiate new connections.

SSL/TLS Architecture

A **connection state** is defined by:

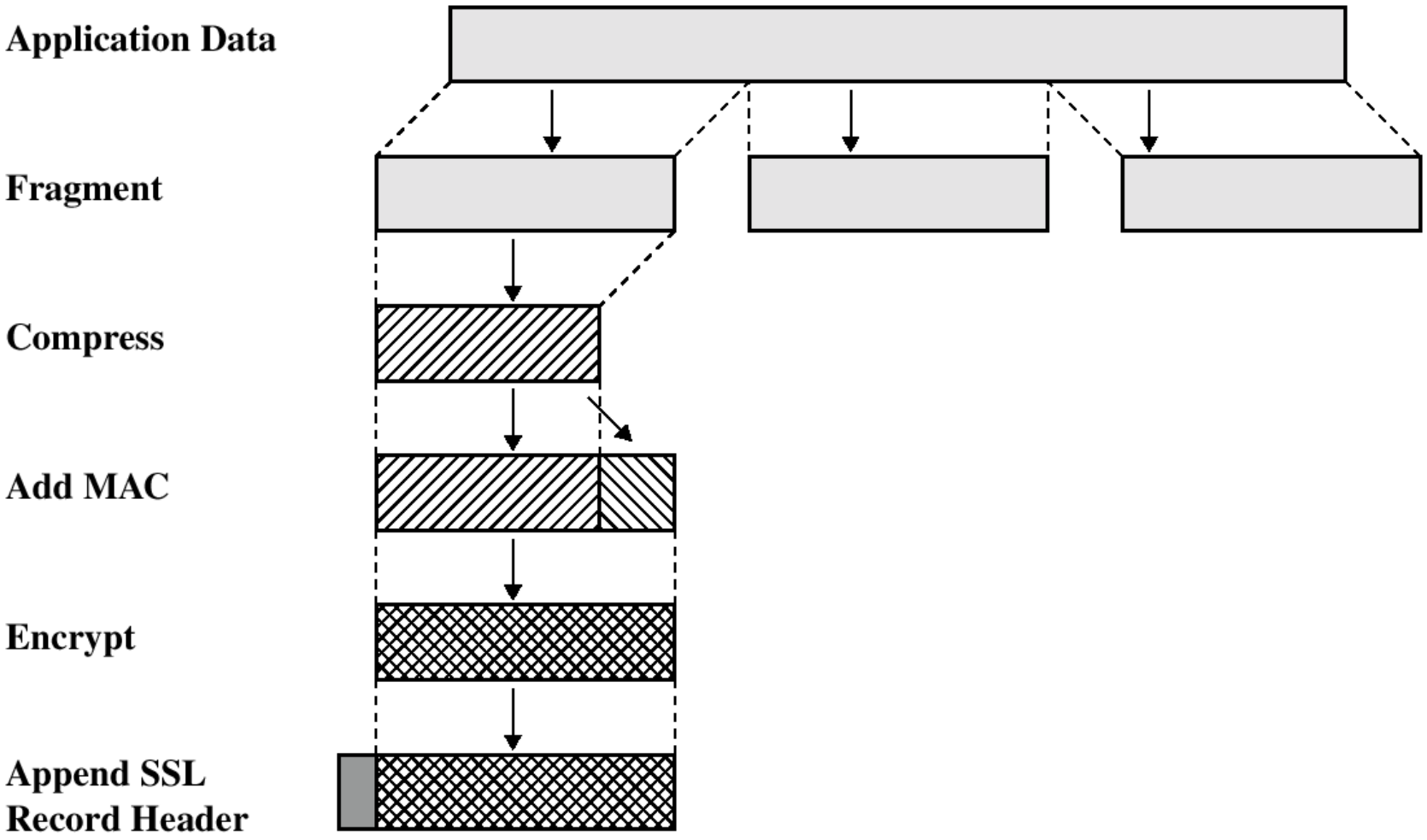
- Server and Client Random.
- Server Write MAC Secret: Server's key for computing MAC.
- Client Write MAC Secret.
- Server Write Key: Server's key for encrypting data.
- Client Write Key.
- Initialization Vectors: Indicates the IV, if CBC mode is used.
- Sequence Numbers: A value from 0 to 2^{64}-1. Used in the MAC computation.

TLS Record Protocols

TLS Record Protocol provides two security services for application data:

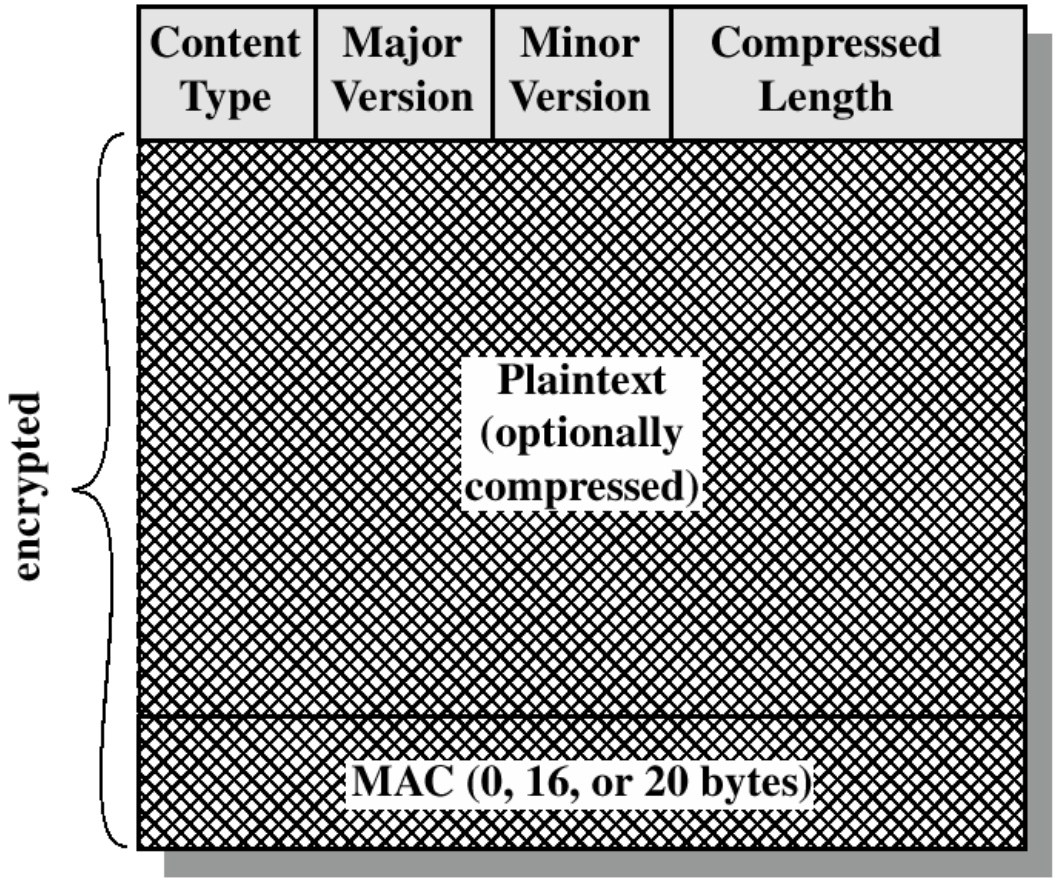
- **Confidentiality:**
  - TLS payloads are encrypted by using symmetric encryption with a shared secret key
  - AES, 3DES, RC4-128, ...
- **Message Integrity:**
  - A MAC is generated for TLS payloads with another shared secret key
  - HMAC

TLS Record Protocol



TLS Record Protocol

Fig. 5.4 TLS Record Format



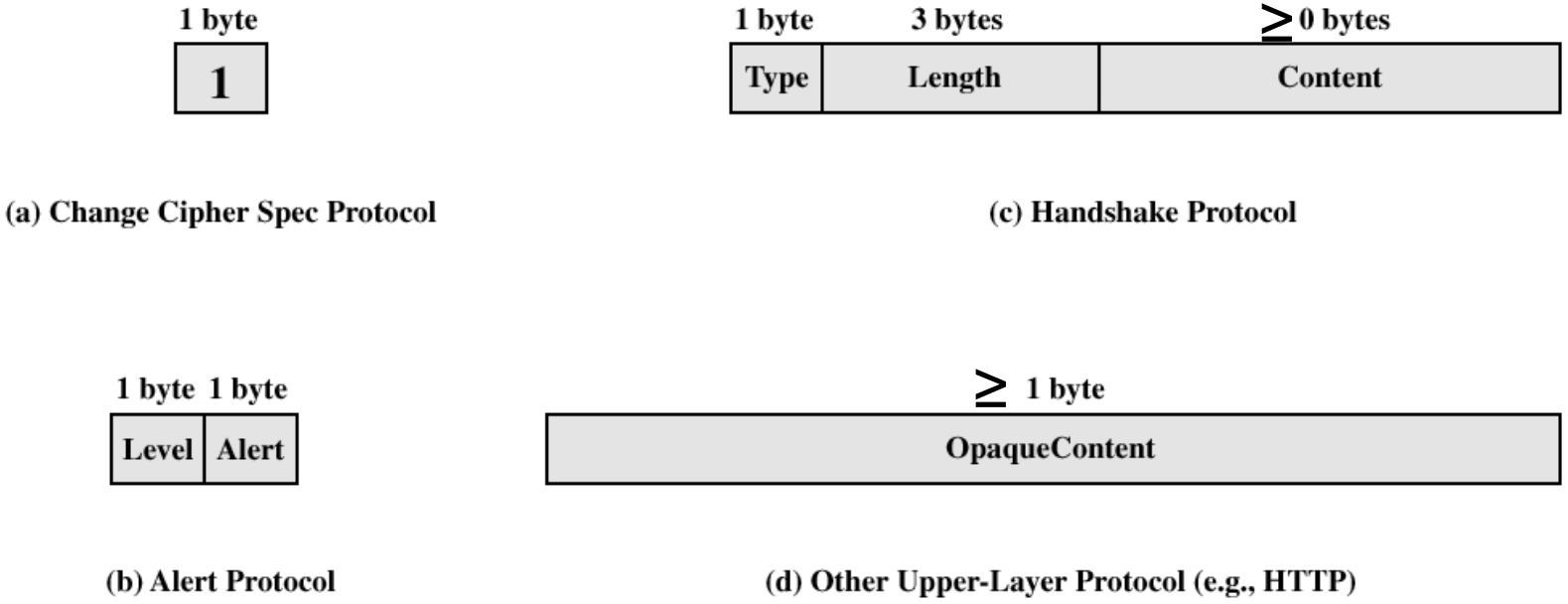
TLS Record Protocol

The Content Type:

- alert
- Handshake
- change_cipher_spec
- application data

TLS Record Protocol

TLS Record Protocol Payload



TLS Record Protocol

- **Change Cipher Spec Protocol:**
  - Consists of a single message with the value 1.
  - Used to copy the pending state into the current state so that the cipher suite is updated.
- **Alert Protocol:**
  - Used to convey TLS-related alerts to the peer entity.
  - A number of Warning or Fatal alerts have been defined.

TLS Record Protocol

- **each alert message consists of 2 fields (bytes)**
- **first field (byte): “warning” or “fatal”**
- **second field (byte):**
  - fatal
    - unexpected_message
    - bad_record_MAC
    - decompression_failure
    - handshake_failure
    - illegal_parameter
    - …
  - warning
    - bad_certificate
    - unsupported_certificate
    - certificate_revoked
    - certificate_expired
    - certificate_unknown
    - …

Handshake Protocol

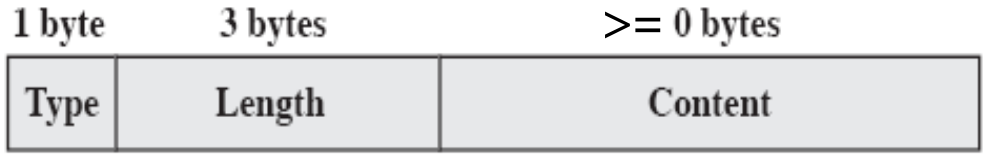
As the most complex part of TLS, the Handshake protocol allows the server and the client to finish the following tasks:

- **Agree on a version of TLS to be used;**
- **Authenticate each other (optional);**
  - Use digital certificates to learn the other's identity and public keys; issue and verify signatures.
- **Negotiate a set of cryptographic algorithms and shared keys to be used in the TLS record protocol.**

Handshake Protocol

The Handshake protocol consists of a number of messages exchanged between client and server. Each message has three fields:

- **Type (1 byte):** Indicates one of 10 messages.
- **Length (3 byte):** The length of message in bytes.
- **Content (>= 0 bytes):** The parameters associated with this message.



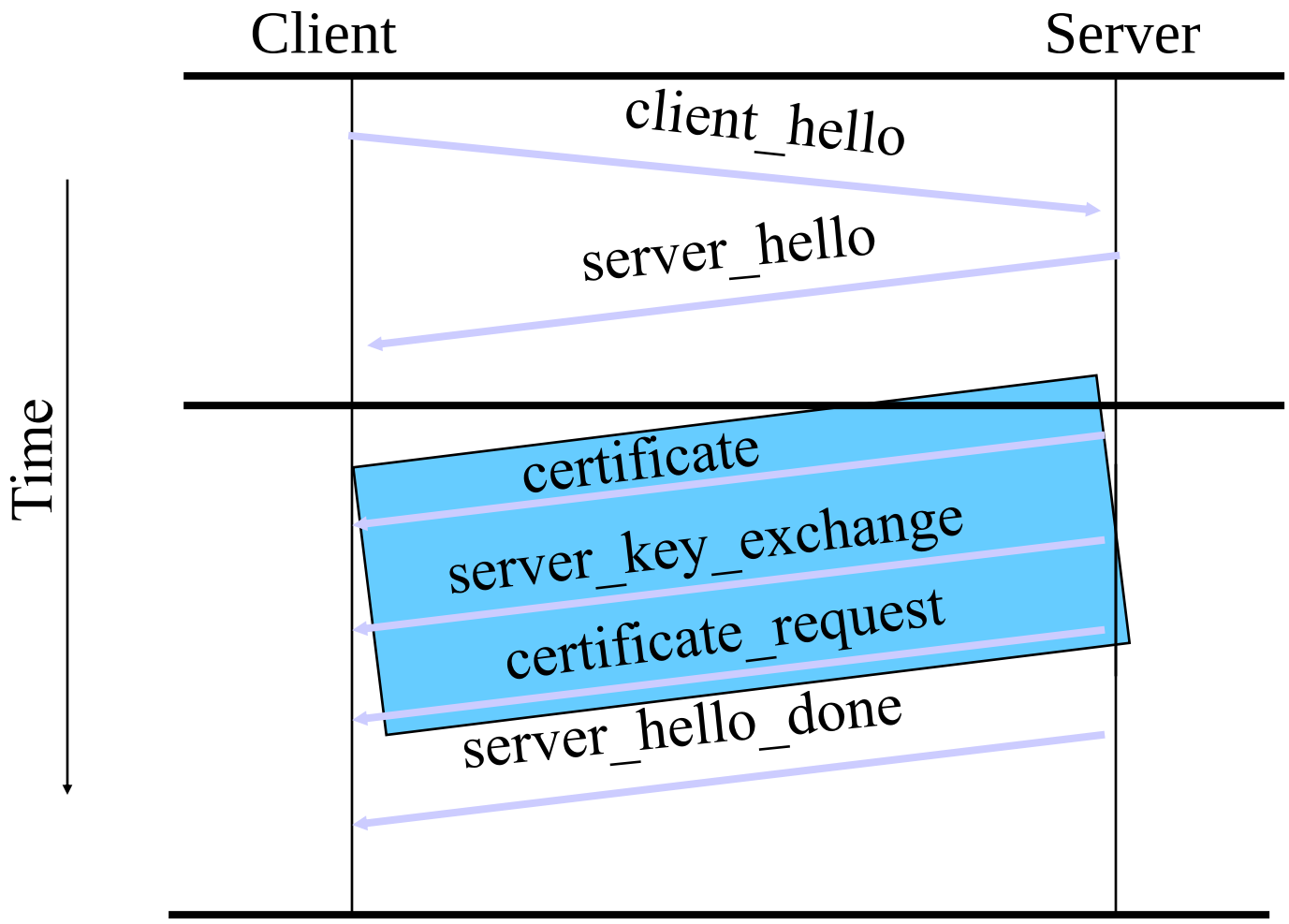
(c) Handshake Protocol

Handshake Protocol

To establish a logical connection between client and server, the Handshake protocol will run in four phases:

- **Phase 1: Establish Security Capabilities:**
  - Negotiate protocol version and algorithms
- **Phase 2: Server Authentication and Key Exchange:**
  - Server sends key exchange messages
- **Phase 3: Client Authentication and Key Exchange**
  - Client sends key exchange messages
- **Phase 4: Finish**
  - Switch to new algorithms and keys.

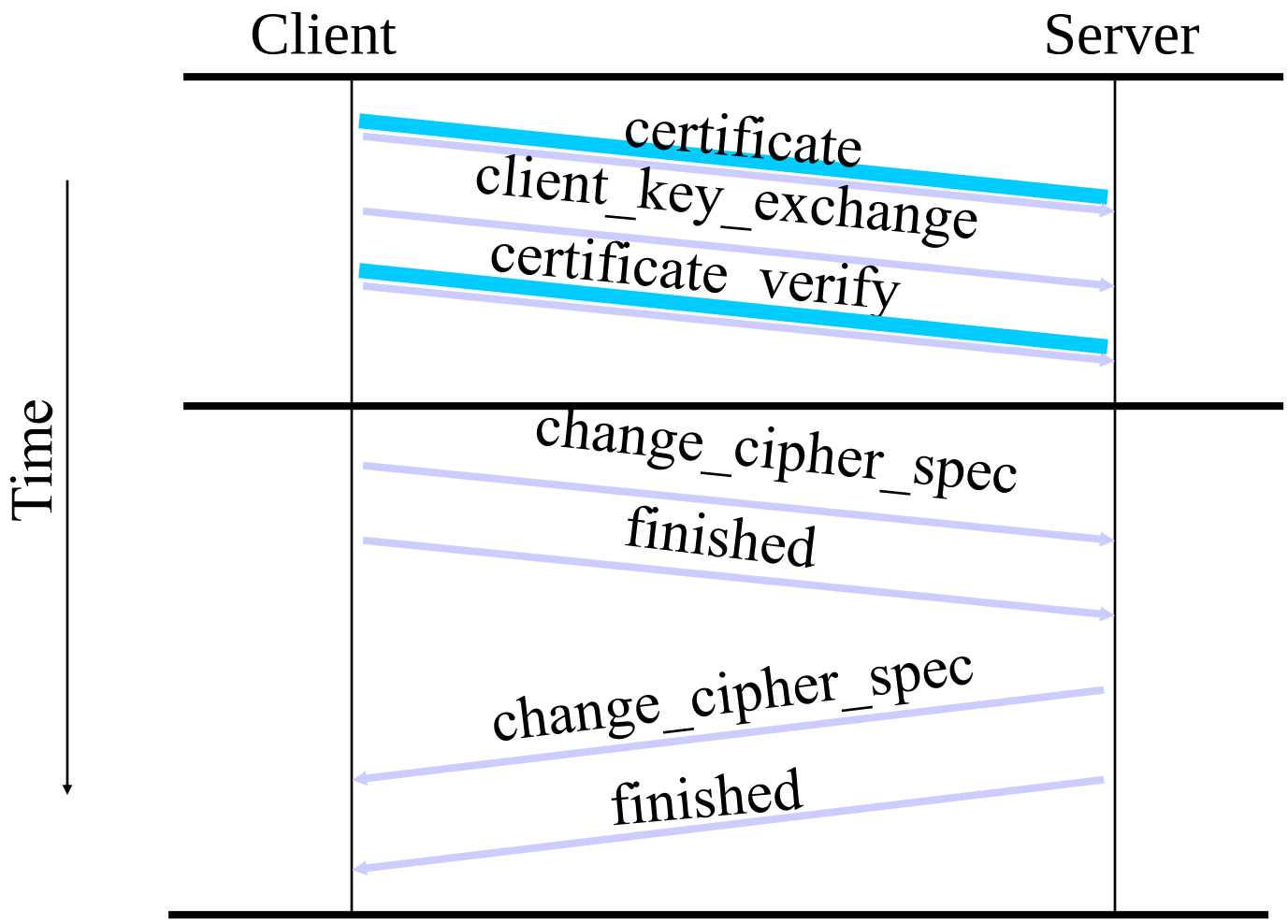
Handshake Protocol



Establish security capabilities, including protocol version, session ID, cipher suite, compression method and initial random numbers.

Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase

Handshake Protocol



Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Change cipher suite and finish handshake protocol.

NOTE:- Shaded transfers are optional or situation-dependent messages that are not always sent.

Handshake Protocol

- Key exchange methods supported by TLS1.2:
  - **RSA**: Client encrypt a secret with server's RSA public key.
  - **Anonymous DH**: No authentication (without certificates).
  - **Fixed DH**: Server has an authorized DH key, while client may or may not have an authorized DH key.
  - **Ephemeral DH**: Use one-time DH keys, which are signed by the senders using RSA or DSS.

Handshake Protocol

- ClientHello
  - Contain protocol version, client nonce as well as the client's list of preferred ciphersuites
- ServerHello
  - Contain chosen protocol version, server nonce as well as the chosen ciphersuite

Handshake Protocol

- Server Certificate
  - Required when server authentication is needed
  - X.509 certificate for one of the following type (depending on the key exchange method)
    - RSA Encryption Key
    - Fixed Diffie-Hellman Public Key
    - Digitial Signature Public Key

Handshake Protocol

- Server Key Exchange
  - Required when Ephemeral Diffie-Hellman is used
    - The server's signature over the client random, server random, and Diffie–Hellman parameters is included
  - Not required when RSA key transport is used as the key exchange method

Handshake Protocol

- Client Key Exchange
  - Always included regardless of the key exchange method
    - RSA key transport: encryption of a random number under server's RSA public key
    - Diffie-Hellman key agreement: client's Diffie-Hellman key

Handshake Protocol

- Certificate Verify
  - Required when client has a certificate for RSA/DSS signature
  - Client signs all the handshake messages it has sent and received previously

Handshake Protocol

- Change Cipher Spec
  - A single-byte message to indicate that all future messages it sends will be encrypted and authenticated.
- Finished
  - A MAC calculated using the master secret key (derived from the key exchange) and the communication transcripts in the Handshake protocol

Handshake Protocol

Session resumption

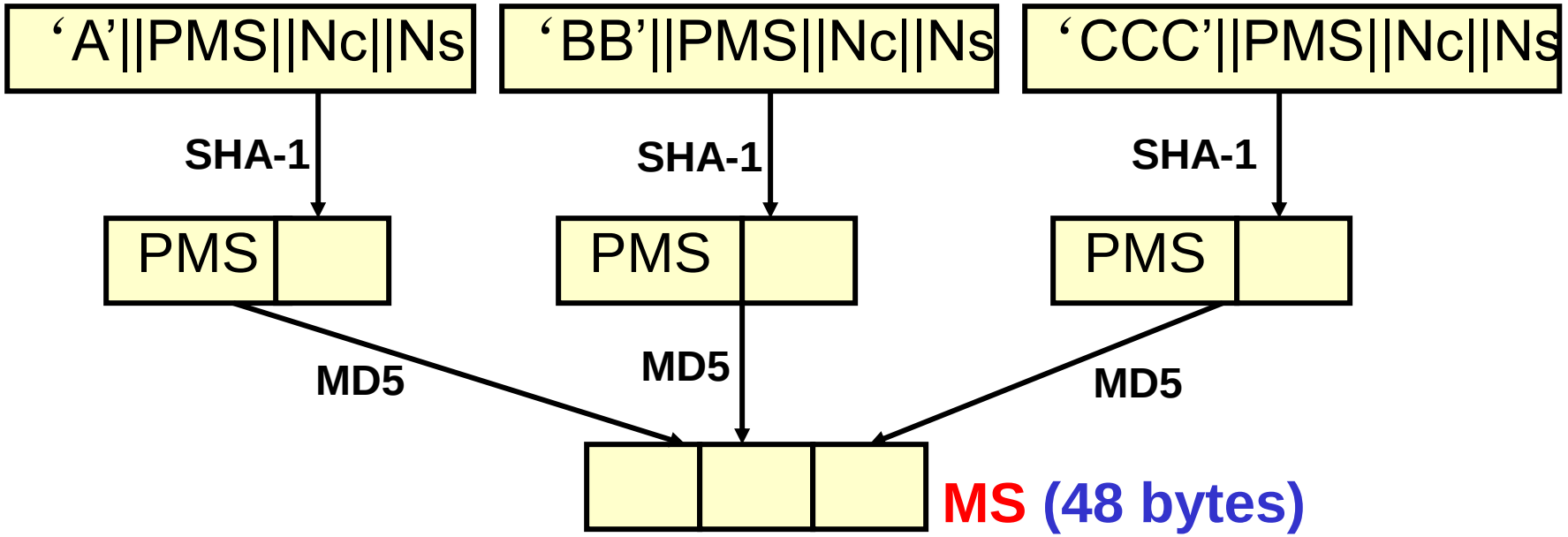
It allows client and server to use an abbreviated handshake to resume a previously established session.

- More efficient than a full handshake
- Only to re-establish the encryption and MAC keys with **new** server nonce and client nonce

Handshake Protocol

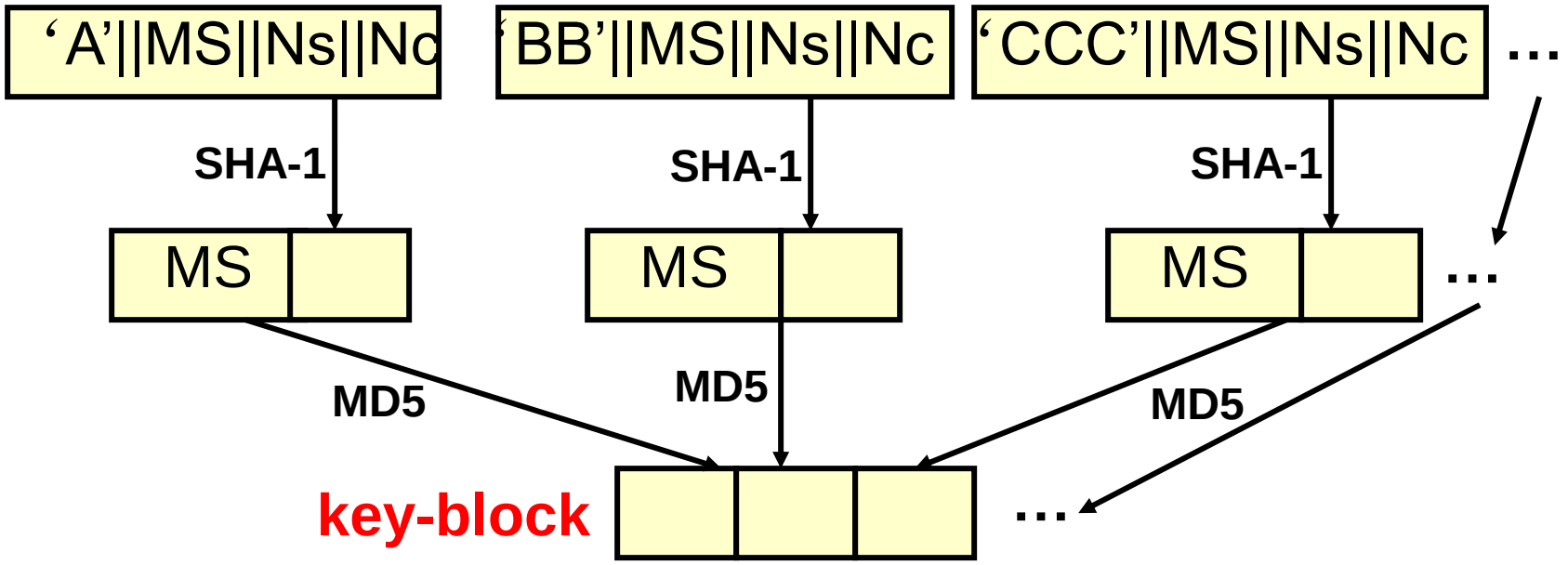
- **The key derived by an TLS key exchange method is called pre-master secret (48 bytes), which can be**
  - **a secret** encrypted with server's RSA public key or
  - **a value** derived by the DH key exchange technique.
- **pre-master secret →master secret → shared keys**

SSL Key Derivation



PMS: Pre-Master Secret
MS: Master Secret
Nc: ClientHello.random
Ns: SeverHello.random

SSL Key Derivation



Key-block consists of (in the following order):

client write MAC secret, sever write MAC secret, client write key, server write key, client write IV, and server write IV.

TLS Key Derivation

Cryptographic computations by using PRF:

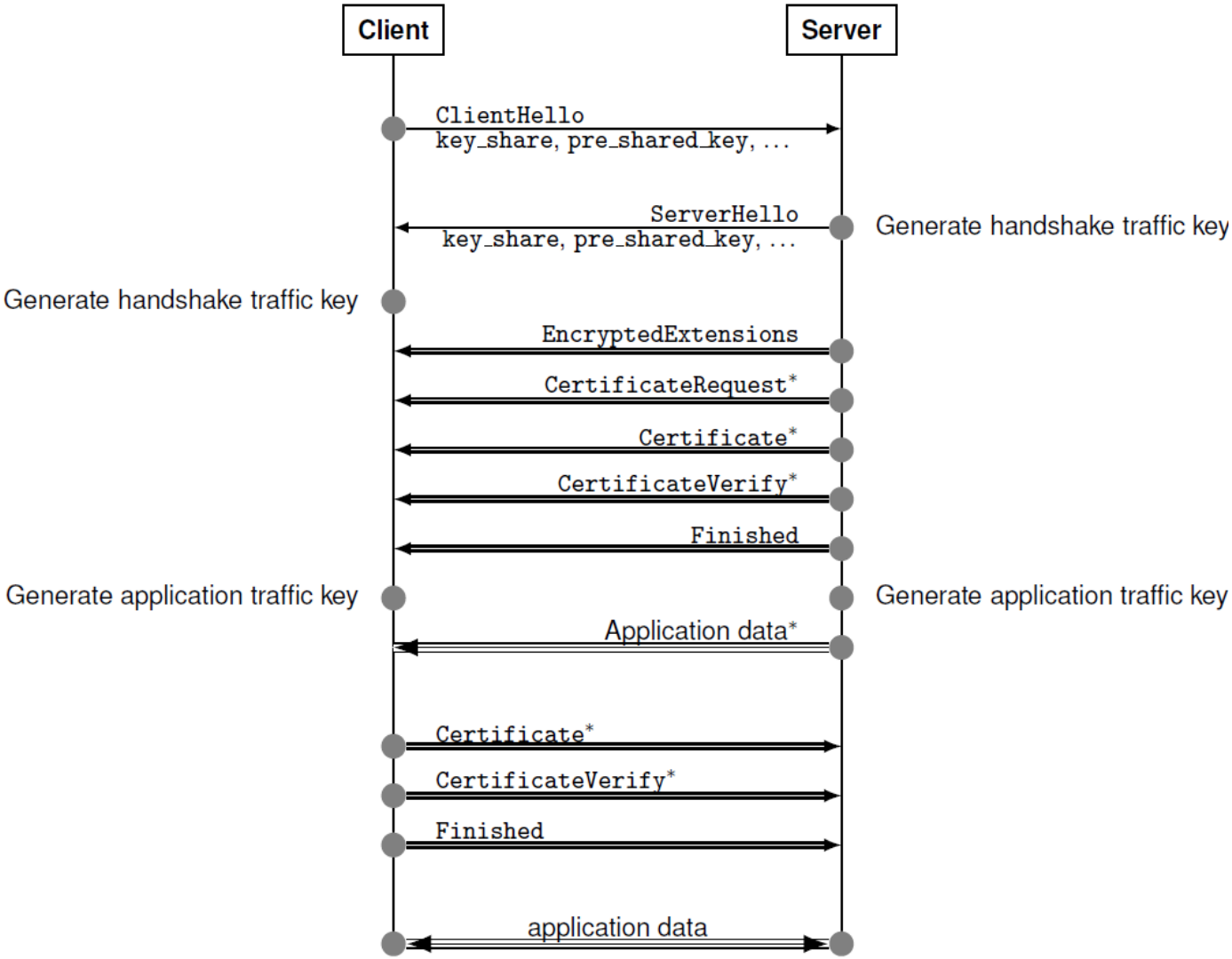
- Handshake protocol → **pre_master_secret** → **master_secret** → **key-block.**
- **master_secret=PRF(pre_master_secret, 'master secret', client_random||server_random)**
- **key-block=PRF(master_secret, 'key expansion', server_random||client_random)**

TLS 1.3

- Static Diffie–Hellman and RSA key transport methods are removed
- All asymmetric key exchange algorithms in TLS 1.3 are based on ephemeral Diffie–Hellman to provide forward secrecy

TLS 1.3

- Unauthenticated ephemeral key exchange happens in the first two flows of the protocol
- A temporary ‘handshake traffic key’ is established to encrypt the remainder of handshake



* denotes messages that may not be present in all ciphersuites.
Single lines denote plaintext flows; double lines denote encrypted flows using the handshake traffic key; triple lines denote encrypted flows using the application traffic key.

Boyd et al., *Protocols for Authentication and Key Establishment.* pp. 287

HTTPS

HTTPS (HTTP over SSL/TLS)

- use https:// rather than http://

  and port 443 rather than 80
- encrypts

  URL, document contents, form data, cookies, HTTP headers