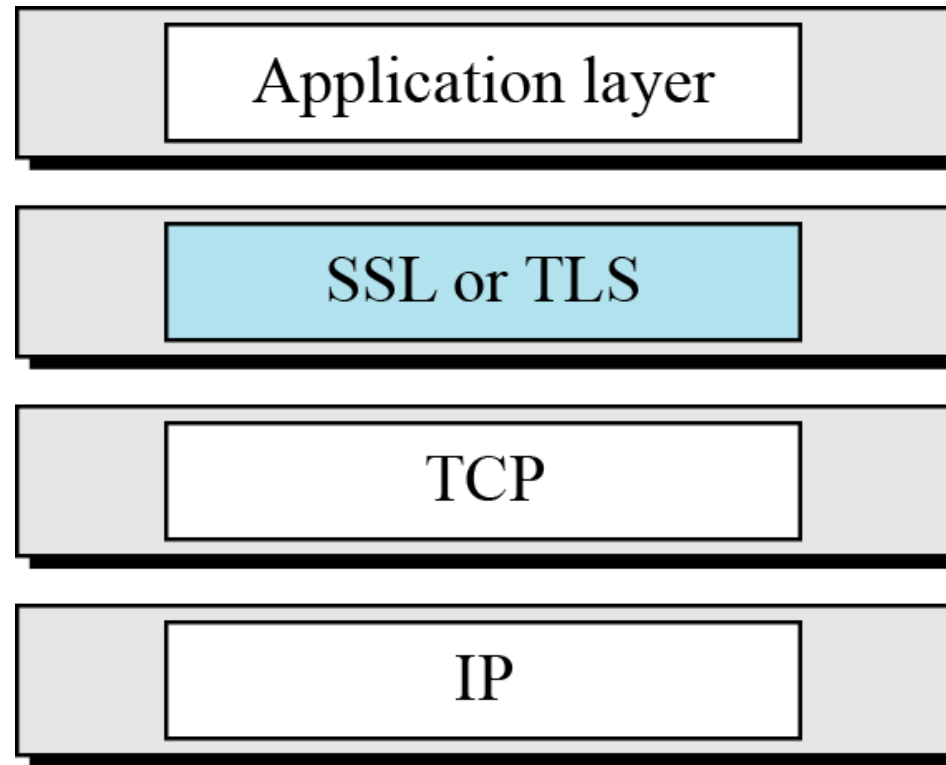


Security at the Transport Layer – SSL and TLS

Location of SSL and TLS in the Internet model



SSL (Secure Socket Layer) and TLS(Transport Layer Security) Protocol

- One of the goal of these protocol is to provide server and client authentication, data confidentiality, and data integrity.
- Application layer protocol such as HTTP that use the services of TCP can encapsulate their data in SSL packets.
- If client and server are capable of running SSL (or TLS) programs then the client can use URL: https:// instead of http://. .. To allow HTTP messages to be encapsulated in SSL(or TLS) packets.

SSL Architecture

- SSL is designed to provide security and compression services to data generated from the application layer.
- Services:

Fragmentation

Compression

Message Integrity

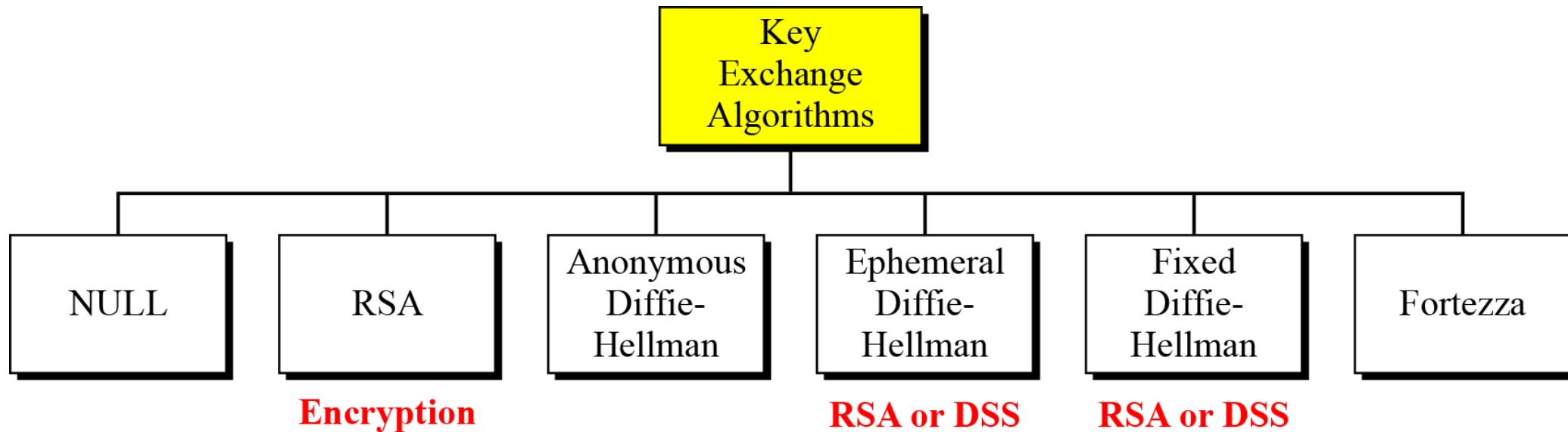
Confidentiality

Framing

Key Exchange Algorithms

- To exchange an authenticated and confidentiality message, the client and server each need six cryptographic secrets
 - Four keys
 - Two initialization vector
- To create these secrets, one pre master secret must be established between the two parties
- SSL defines six key-exchange methods to establish pre-secret

Key Exchange Algorithms

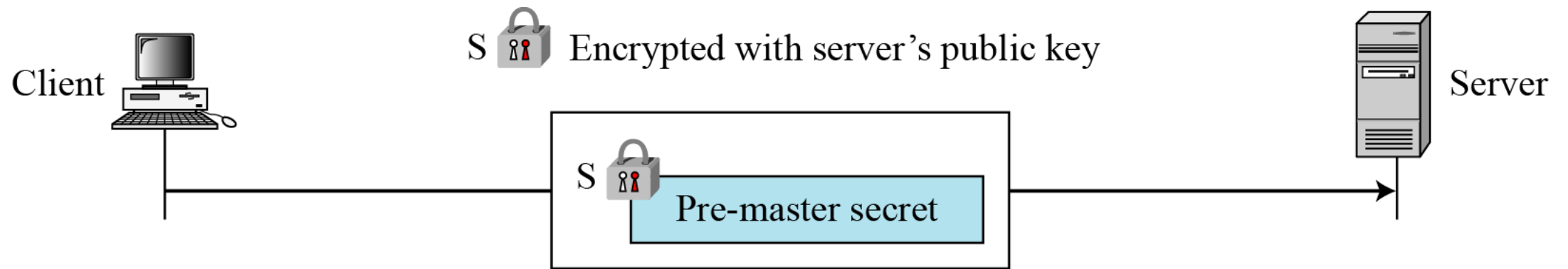


Key Exchange Algorithms

- NULL:
- There is no key exchange in this method. No pre-master secret is established between the client and the server.

Key Exchange Algorithms

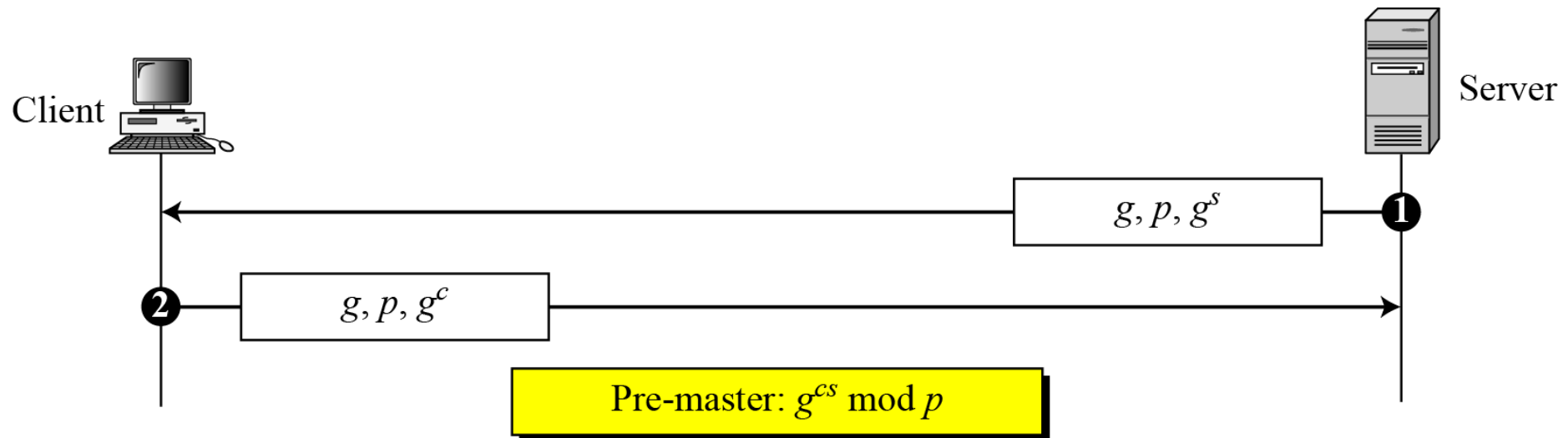
RSA key exchange; server public key



The pre-master secret is a 48 byte random number created by the client

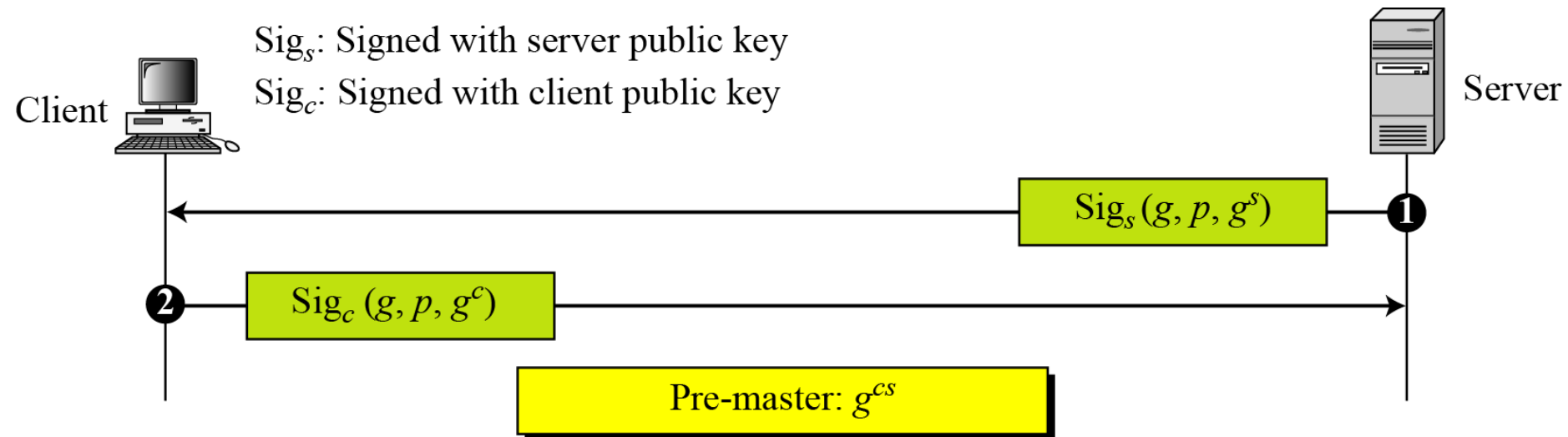
Key Exchange Algorithms

Anonymous Diffie-Hellman key exchange



Key Exchange Algorithms

Ephemeral Diffie-Hellman key exchange



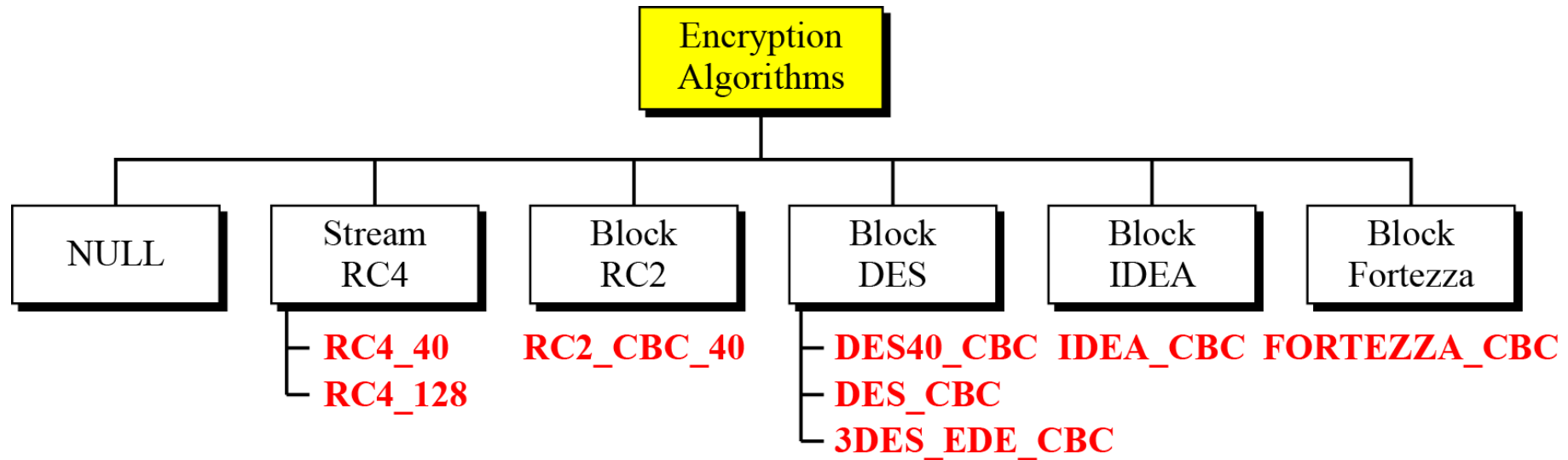
To thwart man-in-the-middle attack, the ephemeral diffie-hellman key exchange is used

Key Exchange Algorithms

- Fixed Diffie Hellman
 - Another solution is the fixed Diffie-Hellman method. All entities in a group can prepare fixed Diffie-Hellman parameters (g and p).
- Fortezza
 - Fortezza is a registered trademark of the U.S. National Security Agency (NSA). It is a family of security protocols developed for the Defense Department.

Encryption Algorithms

Encryption/decryption algorithms



All block protocols use an 8-byte initialization vector

Fortezza uses a 20 byte IV

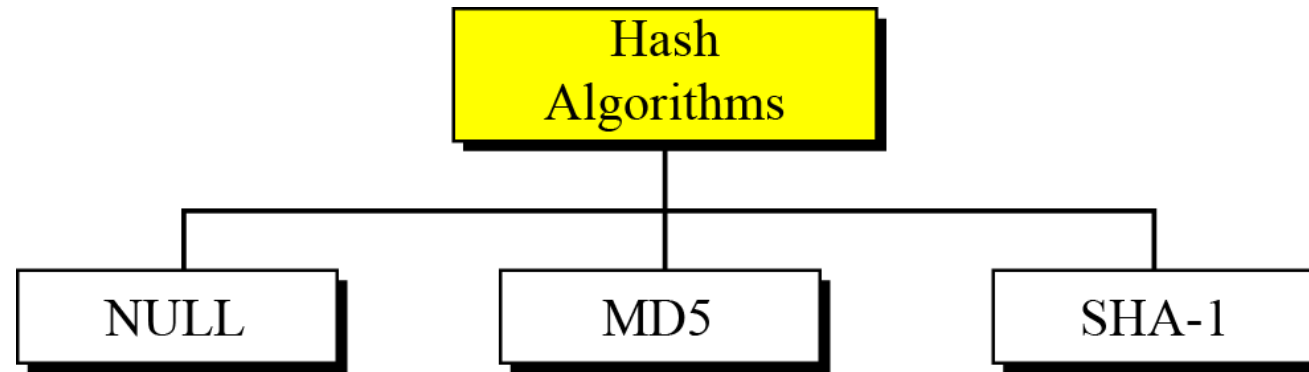
Encryption Algorithms

- NULL
 - The NULL category simply defines the lack of an encryption/decryption algorithm.
- Stream RC
 - Two RC algorithms are defined in stream mode.
 - RC-40 (40 bit key)
 - RC-128 (128 bit key)
- Block RC
 - One RC algorithm is defined in block mode. RC2_CBC_40 (40 bit key)

Encryption Algorithms

- DES
- All DES algorithms are defined in block mode.
- IDEA
- The IDEA algorithm defined in block mode is IDEA_CBC, with a 128-bit key.
- Fortezza
- The one Fortezza algorithm defined in block mode is FORTEZZA_CBC. (96 bit key)

Hash Algorithms



Hash algorithms for message integrity

Hash Algorithms

- NULL
 - The two parties may decline to use an algorithm. In this case, there is no hash function and the message is not authenticated.
- MD5
 - The two parties may choose MD5 as the hash algorithm. In this case, a 128-key MD5 hash algorithm is used.
- SHA1
 - The two parties may choose SHA as the hash algorithm. In this case, a 160-bit SHA-1 hash algorithm is used.

Cipher Suite

- The combination of key exchange, hash, and encryption algorithms defines a cipher suite for each SSL session.

SSL_DHE_RSA_WITH_DES_CBC_SHA

SSL cipher suite list

<i>Cipher suite</i>	<i>Key Exchange</i>	<i>Encryption</i>	<i>Hash</i>
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
SSL_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
SSL_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
SSL_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
SSL_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
SSL_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1
SSL_FORTEZZA_DMS_WITH_NULL_SHA	Fortezza	NULL	SHA-1
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA	Fortezza	Fortezza	SHA-1
SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	Fortezza	RC4	SHA-1

Compression Algorithms

- Compression is optional in SSLv3.
- No specific compression algorithm is defined for SSLv3.
- Therefore, the default compression method is NULL.

Cryptographic Parameter Generation

- To achieve message integrity and confidentiality, the client and server each need three cryptographic secrets
 - Four keys
 - Two initialization vector
- The client needs
 - One key for message authentication (HMAC)
 - One key for encryption
 - One IV for block encryption
- Server needs same

Cryptographic Parameter Generation

- SSL requires that the keys for one direction be different from those for the other direction
- If there is an attack in one direction, the other direction is not affected

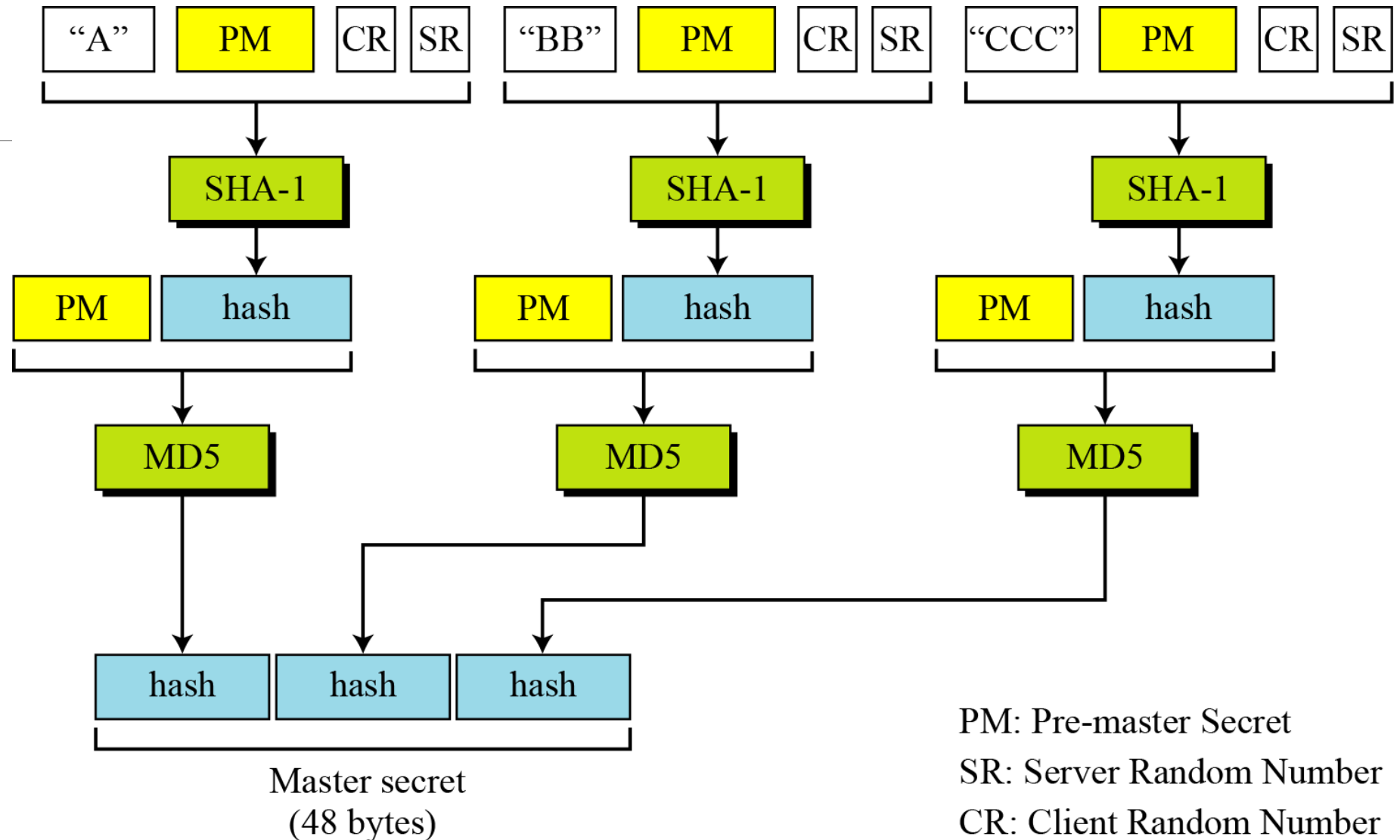
Cryptographic Parameter Generation

- Cryptographic Parameter Generation Procedure:
 1. The client and server exchange two random numbers;
 2. The client and server exchange one pre-master secret using one of the key exchange algorithms
 3. A 48 byte master secret is created from the pre-master secret by applying two hash functions (MD5 and SHA-1)

Cryptographic

Parameter Generation

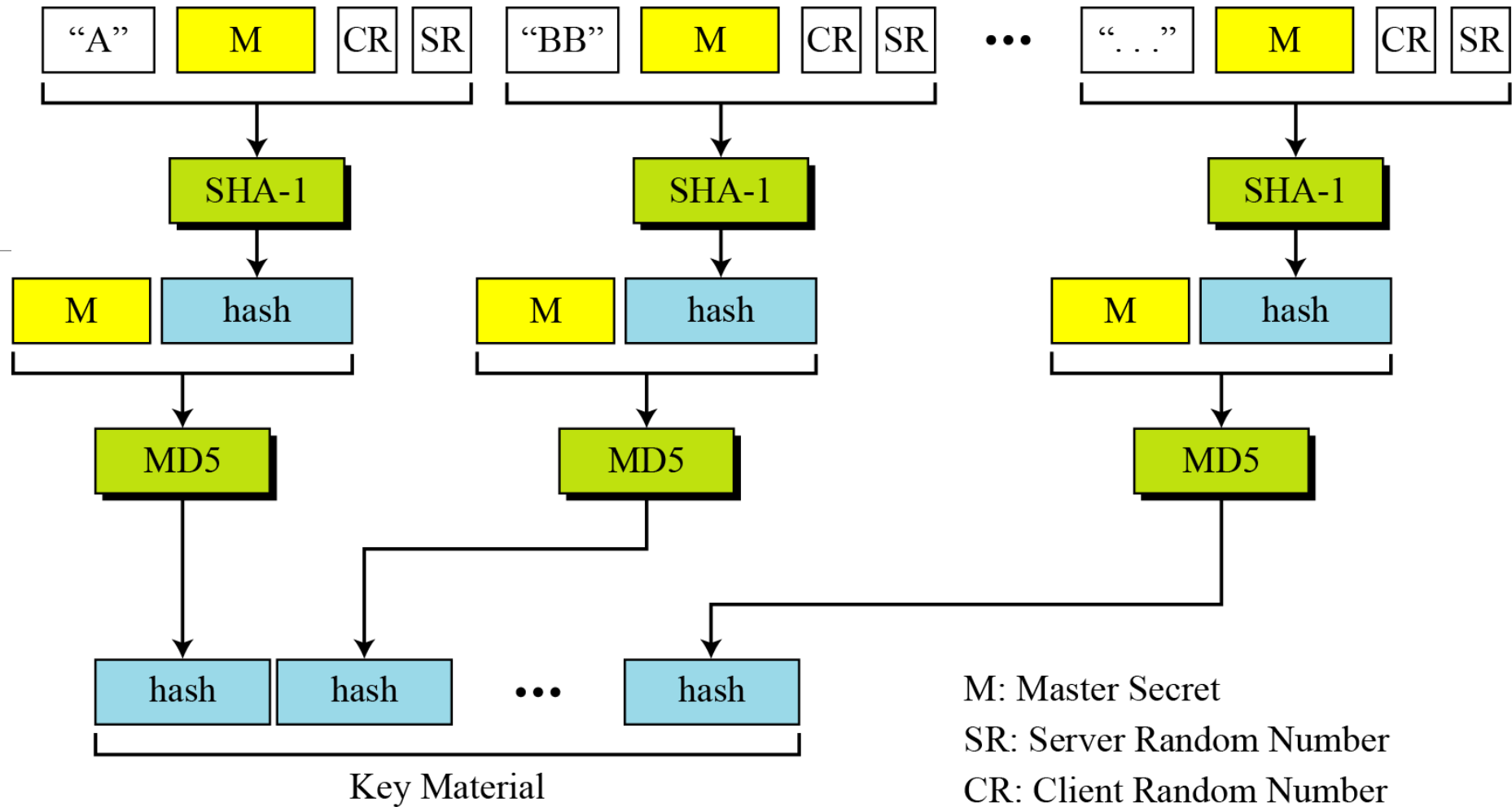
Calculation of master secret from pre-master secret



Cryptographic Parameter Generation

- The master secret is used to create variable-length key material by applying the same set of hash functions and prepending with different constants.
- The module is repeated until key material of adequate size is created
- Length of key material block depends on the cipher suite selected and the size of keys needed for this suite

Cryptographic Parameter Generation

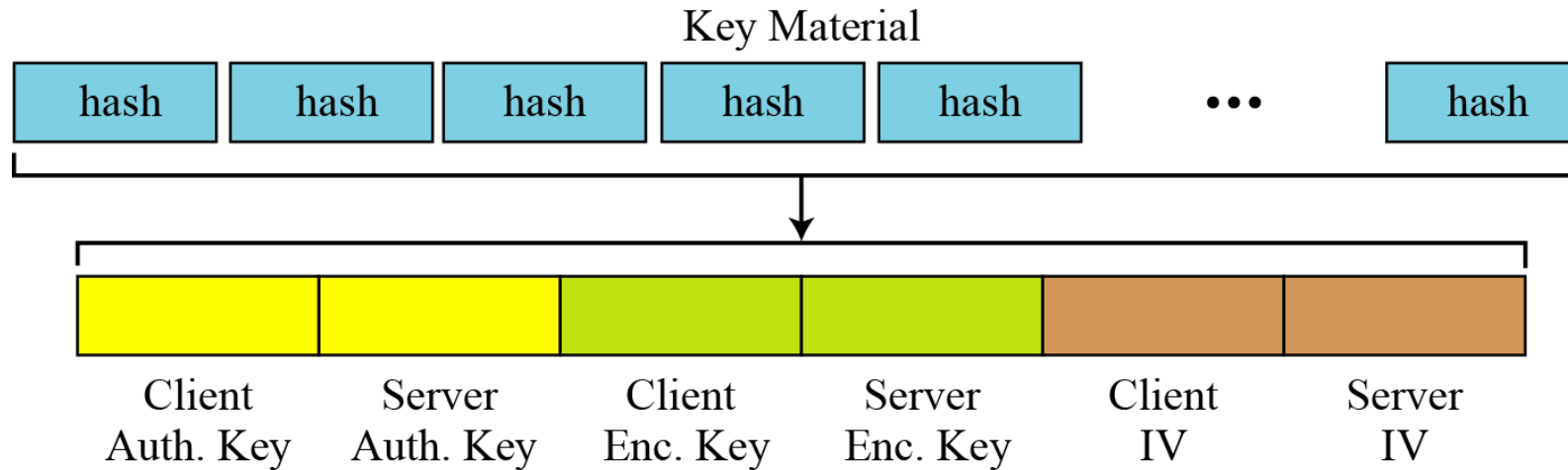


Cryptographic Parameter Generation

Auth. Key: Authentication Key

Enc. Key: Encryption Key

IV: Initialization Vector



Extractions of cryptographic secrets from key material

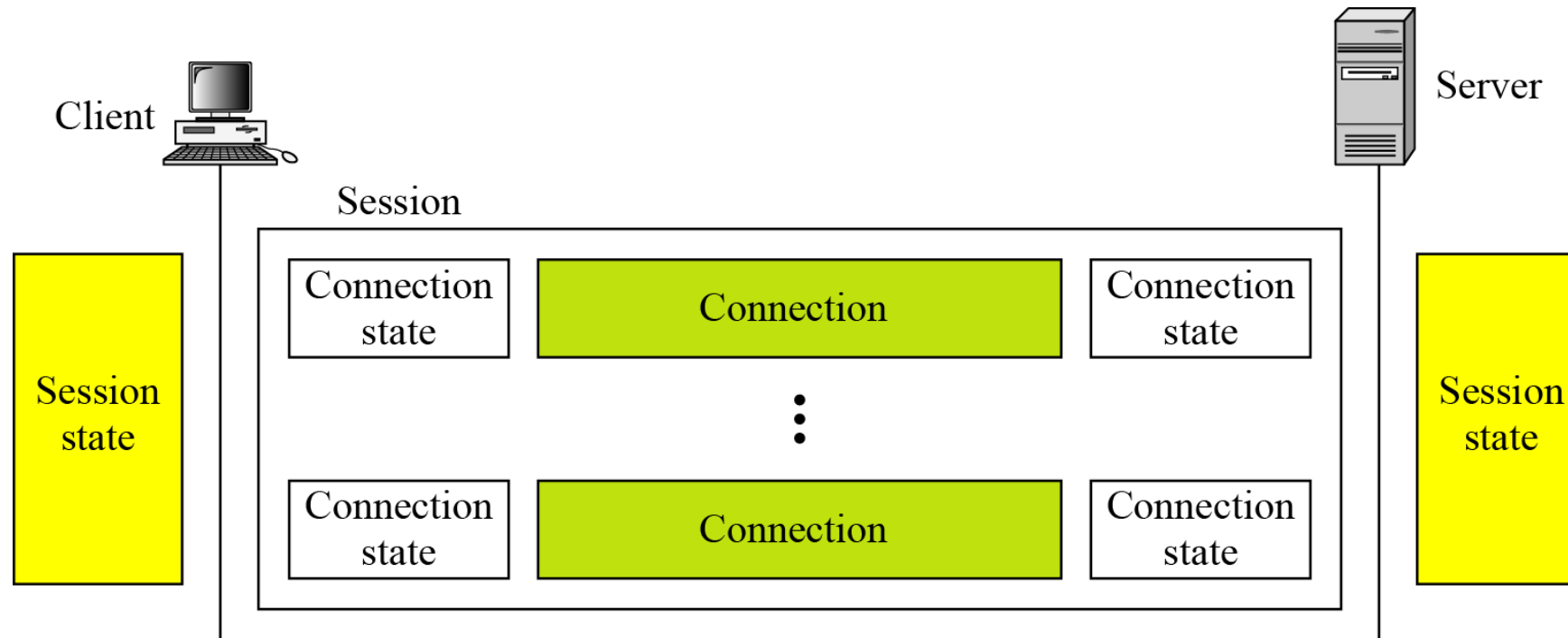
Sessions and Connections

- In a session, one party has the role of a client and the other the role of a server;
- In a connection, both parties have equal roles, they are peers.
- A session is an association between a client and server.
- After a session is established, the two parties have common information,
 - such as the session identifier,
 - the certificate authentication each of them,
 - the compression method,
 - the cipher suite,
 - and master secret that is used to create keys for message authentication encryption

Sessions and Connections

- A session can consist of a many connections,
- A connection between two parties can be terminated and re-established within the same session.
- When a connection is terminated, the two parties can also terminate the session, but it is not mandatory, A session can be suspended and resumed again.

Sessions and Connections



A session and connections

Session State Parameters

<i>Parameter</i>	<i>Description</i>
Session ID	A server-chosen 8-bit number defining a session.
Peer Certificate	A certificate of type X509.v3. This parameter may be empty (null).
Compression Method	The compression method.
Cipher Suite	The agreed-upon cipher suite.
Master Secret	The 48-byte secret.
Is resumable	A yes-no flag that allows new connections in an old session.

Connection State Parameters

<i>Parameter</i>	<i>Description</i>
Server and client random numbers	A sequence of bytes chosen by the server and client for each connection.
Server write MAC secret	The outbound server MAC key for message integrity. The server uses it to sign; the client uses it to verify.
Client write MAC secret	The outbound client MAC key for message integrity. The client uses it to sign; the server uses it to verify.
Server write secret	The outbound server encryption key for message integrity.
Client write secret	The outbound client encryption key for message integrity.
Initialization vectors	The block ciphers in CBC mode use initialization vectors (IVs). One initialization vector is defined for each cipher key during the negotiation, which is used for the first block exchange. The final cipher text from a block is used as the IV for the next block.
Sequence numbers	Each party has a sequence number. The sequence number starts from 0 and increments. It must not exceed $2^{64} - 1$.

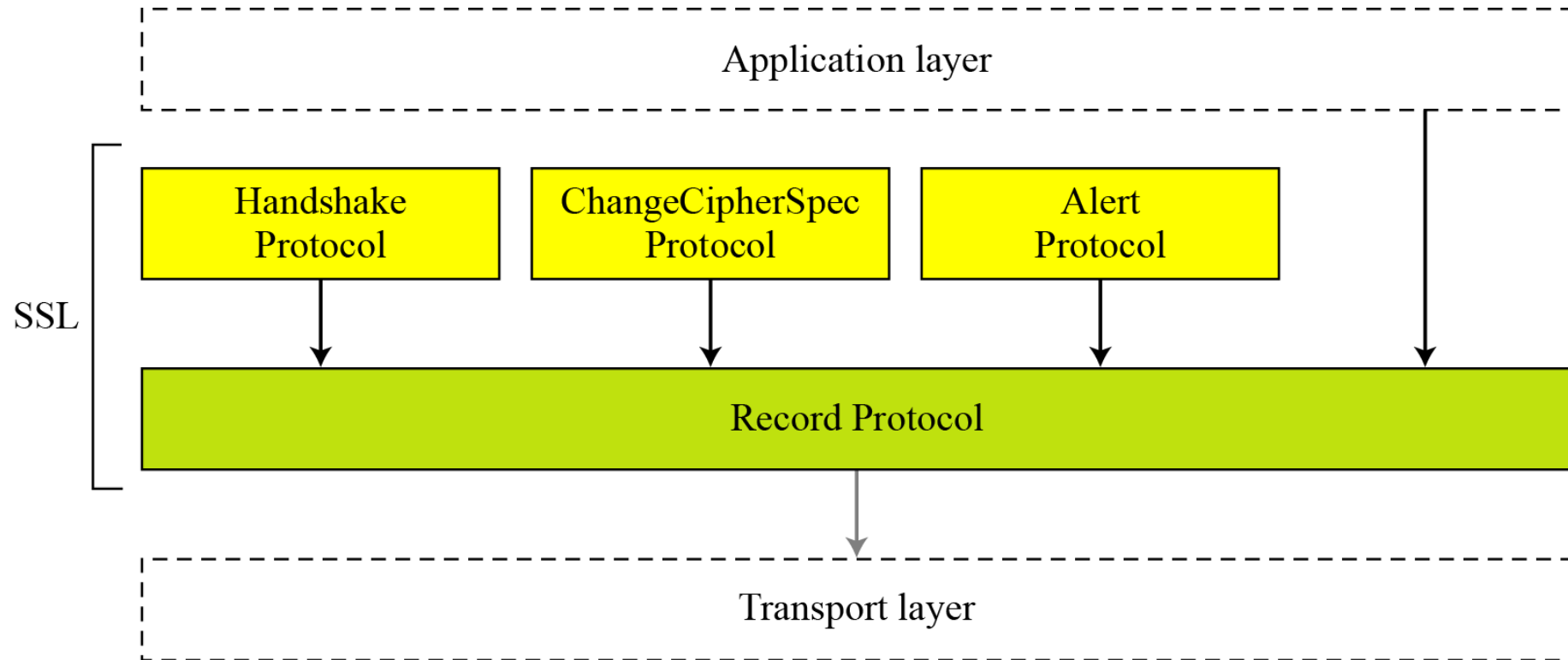
Sessions and Connections

- The client and the server have six different cryptography secrets:
 - three read secrets and three write secrets.
- The read secrets for the client are the same as the write secrets for the server and vice versa.

Four Protocols

- SSL accomplishes secure connection with the help of four protocols.

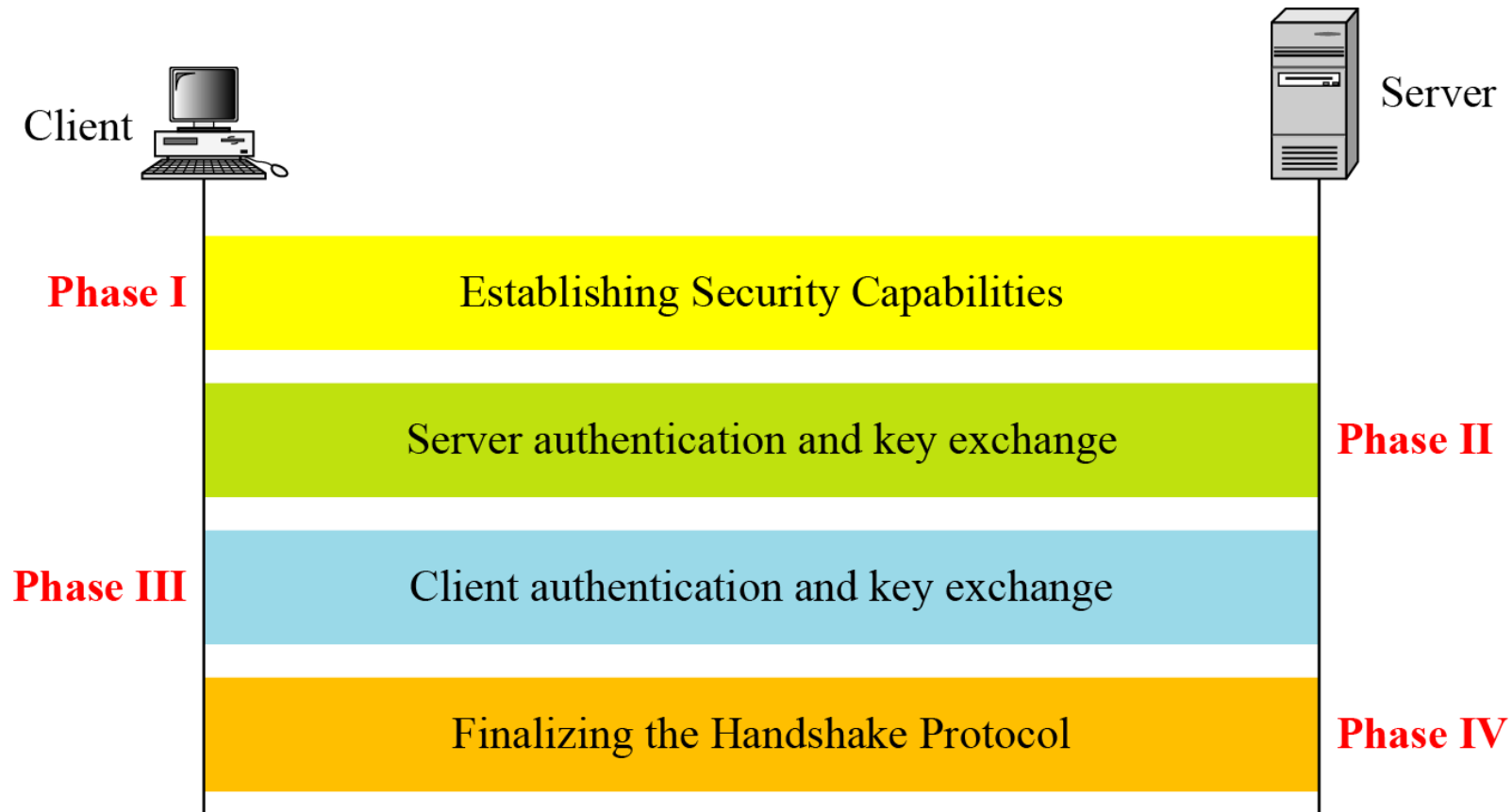
FOUR Protocols



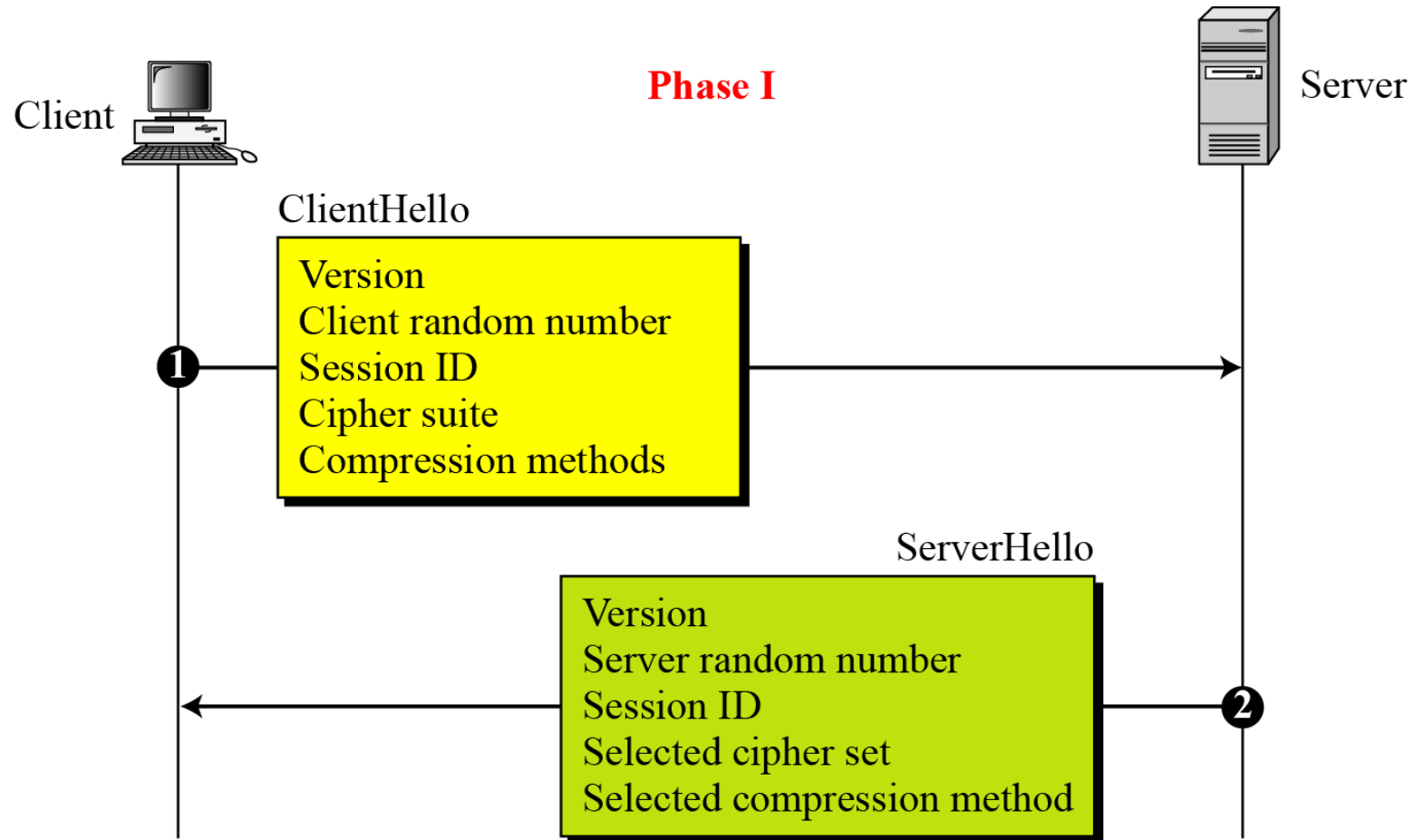
FOUR Protocols

- The record protocol is the carrier.
- It carries message from three other protocols as well as the data coming from the application layer
- Message from the record protocol are payloads to the transport layer, (generally TCP)

Handshake Protocol



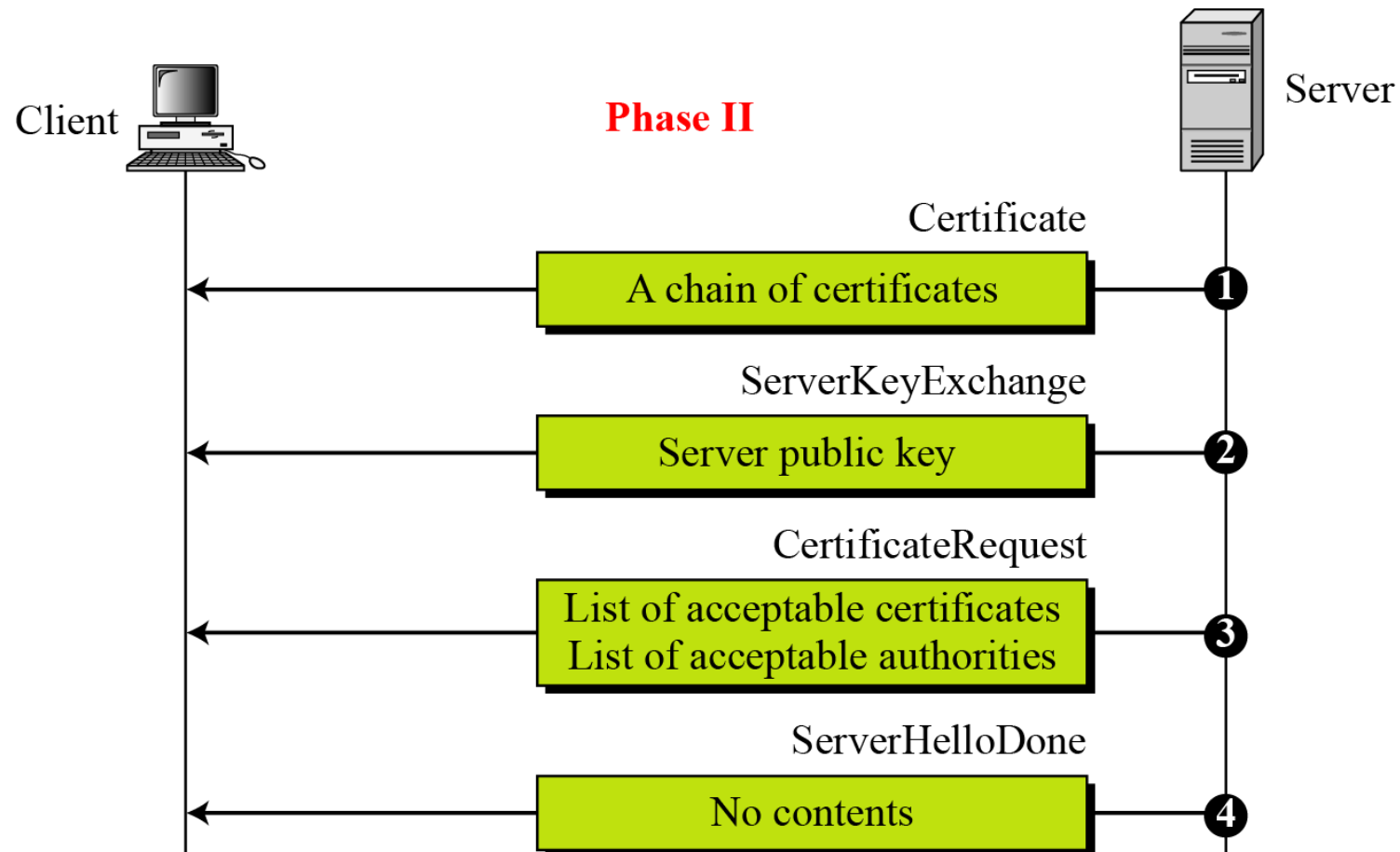
Phase I of Handshake Protocol



Phase I of Handshake Protocol

- **After Phase I, the client and server know the following:**
- The version of SSL
- The algorithms for key exchange, message authentication, and encryption
- The compression method
- The two random numbers for key generation

Phase II of Handshake Protocol



Digital Certificate

- It is an electronic document which uses a digital signature to bind a public key with an identity — information
 - such as the name of a person or an organization, their address, and so forth.
- The certificate can be used to verify that a public key belongs to an individual.

Certificate Authority (CA)

- Issues a digital certificate to users:
- It certifies the public keys of users.
- It must validate applicant's identity before issuing a certificate.
- CA must verify that the applicant has the matching private keys.

Distribution:

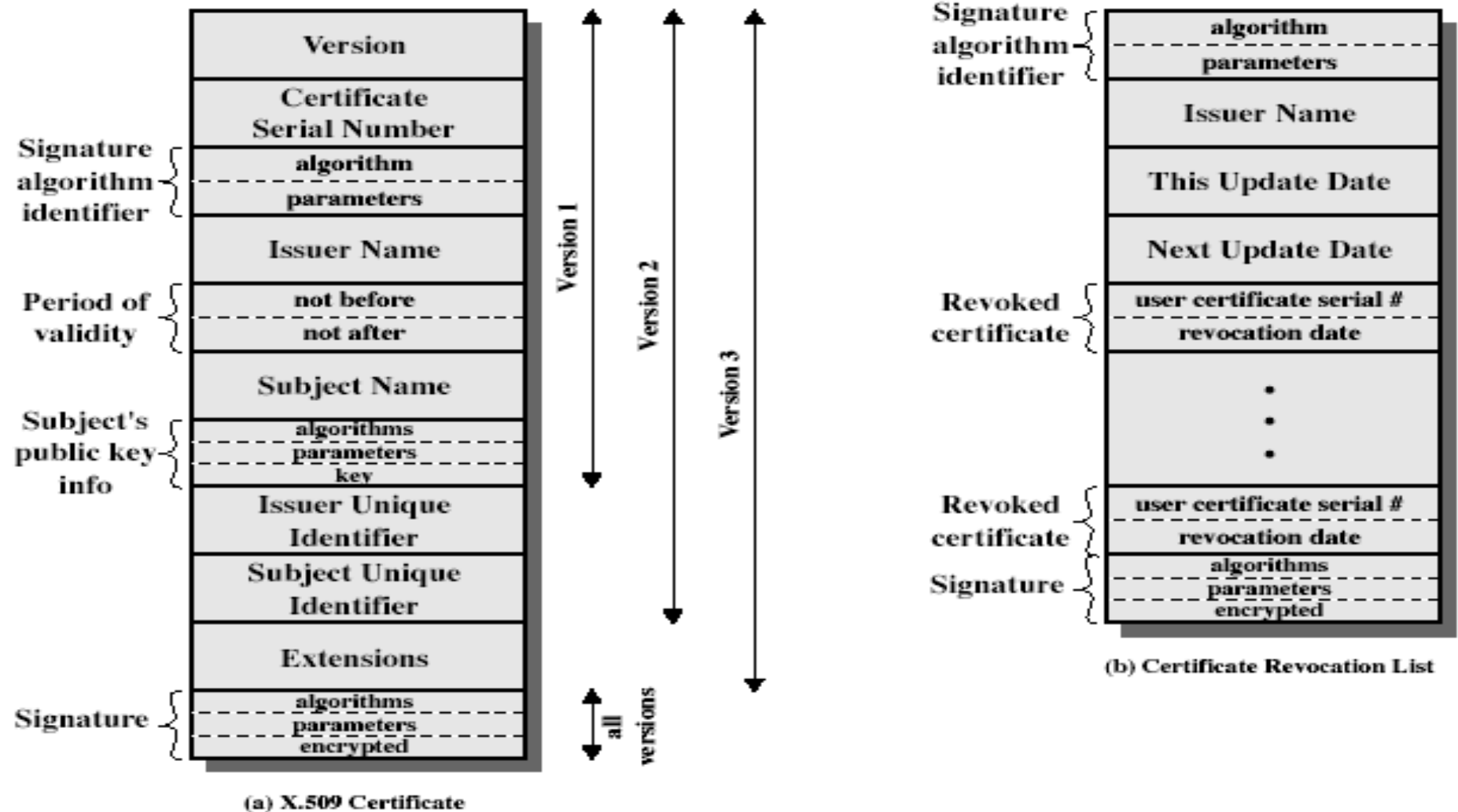
- Digital certificates are not secret.
- On the contrary, they should be widely advertised.

Certificate Authority (CA)

Certificate Revocation:

- A certificate valid during the dates mentioned on the certificate.
- A CA can revoke a certificate prematurely.
//due to various reasons – CA's or applicant's private key compromised.
- Revoked certificates can not be used.
- Revoked certificates are placed on a Certification Revocation List (CRL).
- A certificate user must verify the certificate.

Digital Certificate



X.509 digital certificates format

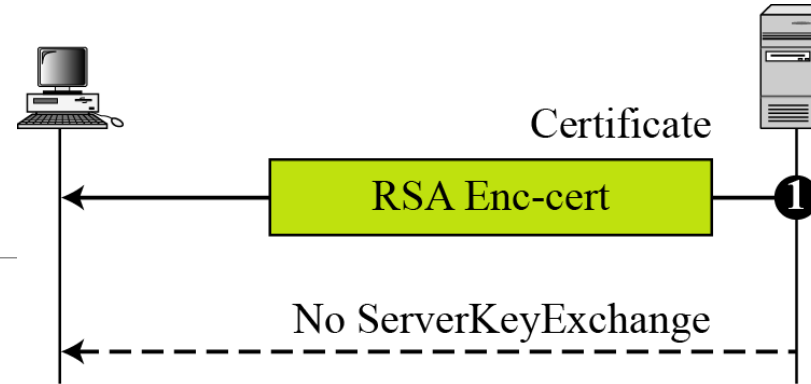
X.509 Certificate

- issued by a Certification Authority (CA), containing:
 - version (1, 2, or 3)
 - serial number (unique within CA) identifying certificate
 - signature algorithm identifier
 - issuer X.500 name (CA)
 - period of validity (from - to dates)
 - subject X.500 name (name of owner)
 - subject public-key info (algorithm, parameters, key)
 - issuer unique identifier (v2+)
 - subject unique identifier (v2+)
 - extension fields (v3)
 - signature (of hash of all fields in certificate)
- notation $CA\langle\langle A \rangle\rangle$ denotes certificate for A signed by CA

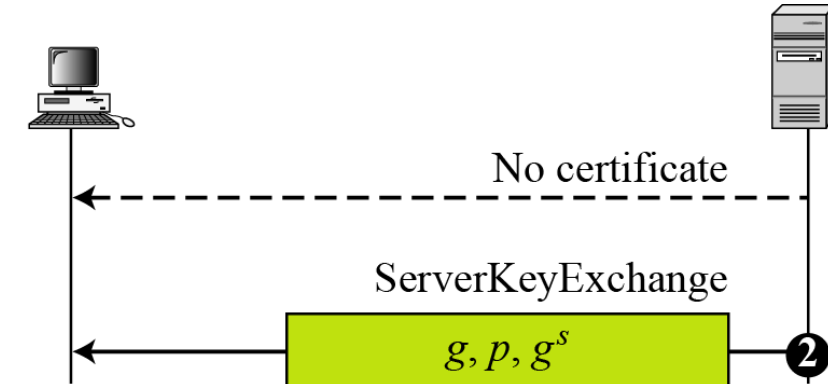
Phase II of Handshake Protocol

- **After Phase II,**
- The server is authenticated to the client.
- The client knows the public key of the server if required.

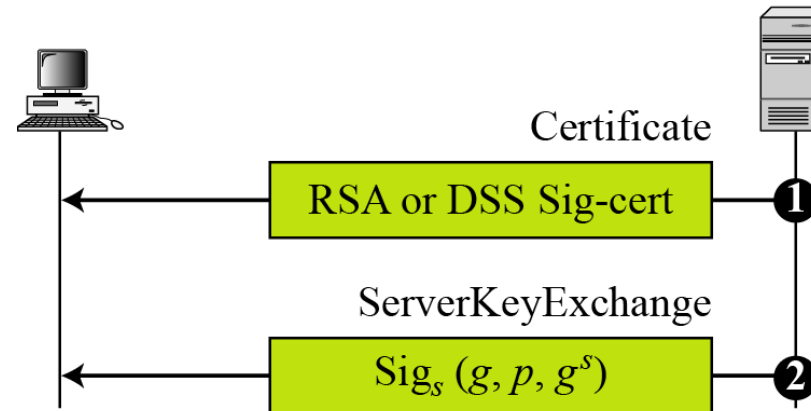
Four cases in Phase II



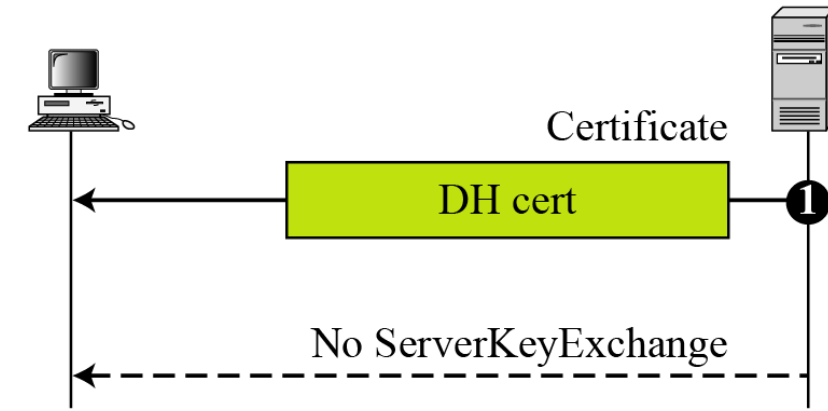
a. RSA



b. Anonymous DH

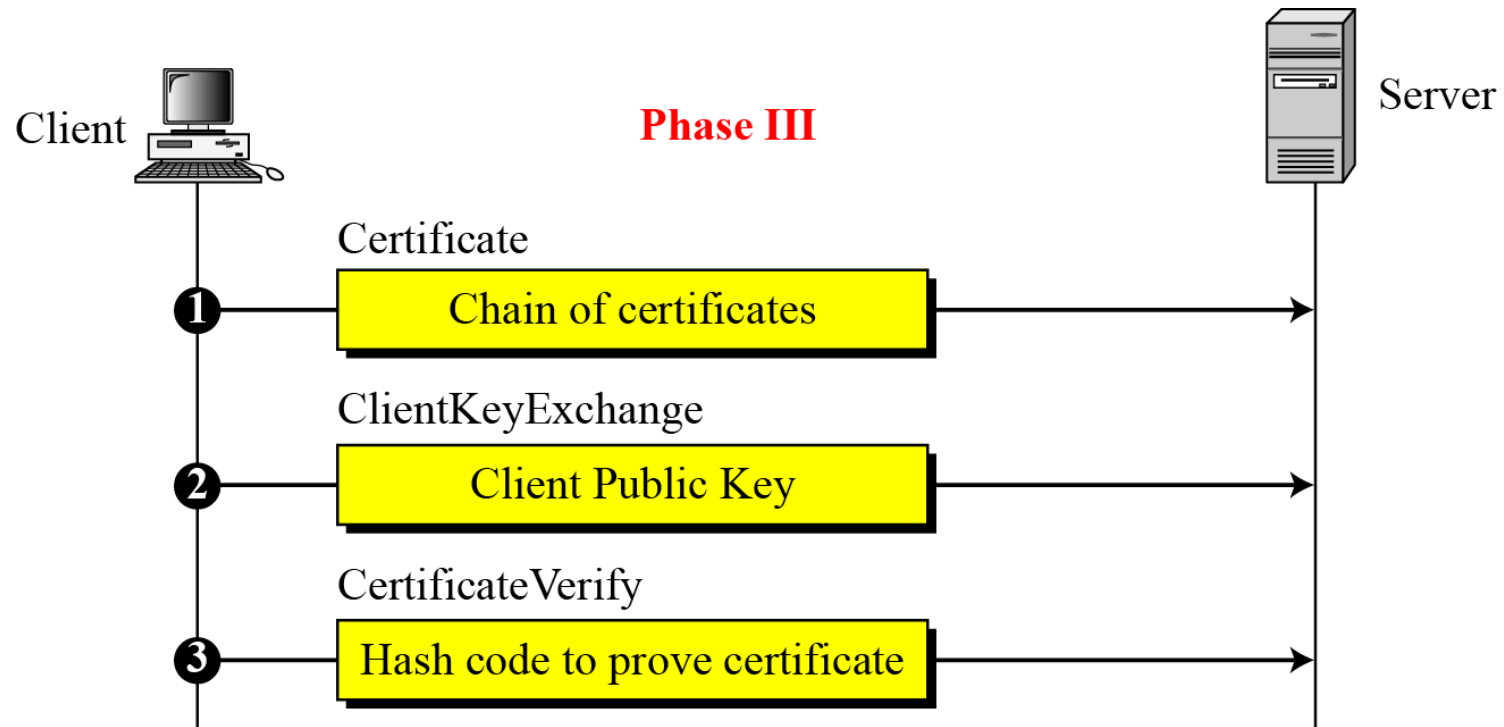


c. Ephemeral DH



d. Fixed DH


Phase III of Handshake Protocol

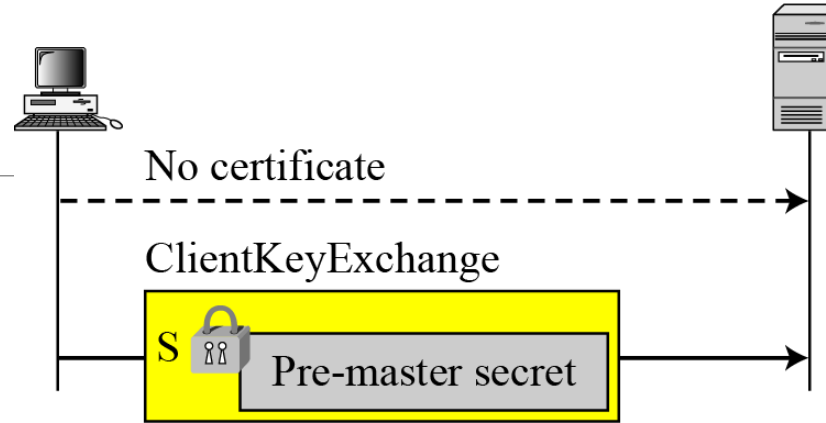


Phase III of Handshake Protocol

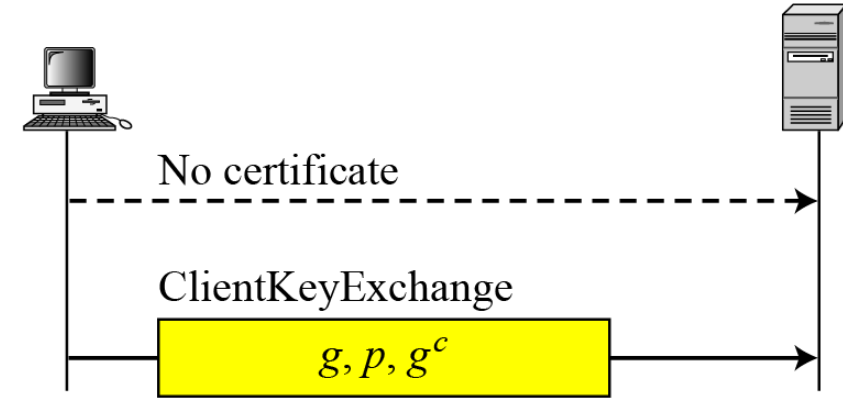
- **After Phase III,**
- The client is authenticated for the server.
- Both the client and the server know the pre-master secret.

Four cases in Phase III

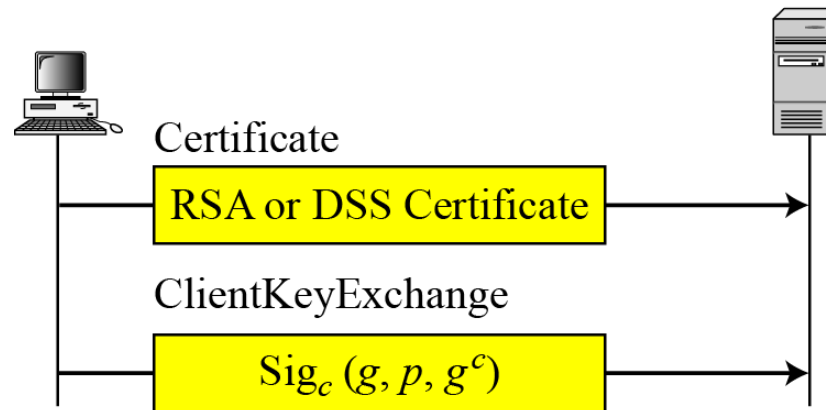
S  encrypted with server's public key
 Sig_c: Signed with client's public key



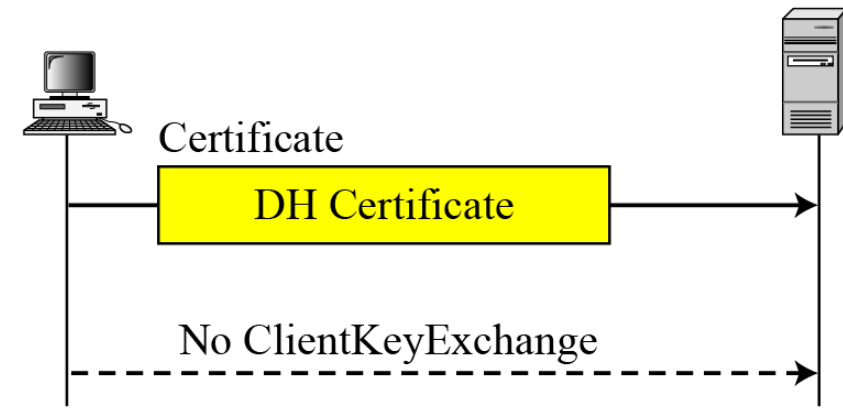
a. RSA



b. Anonymous DH

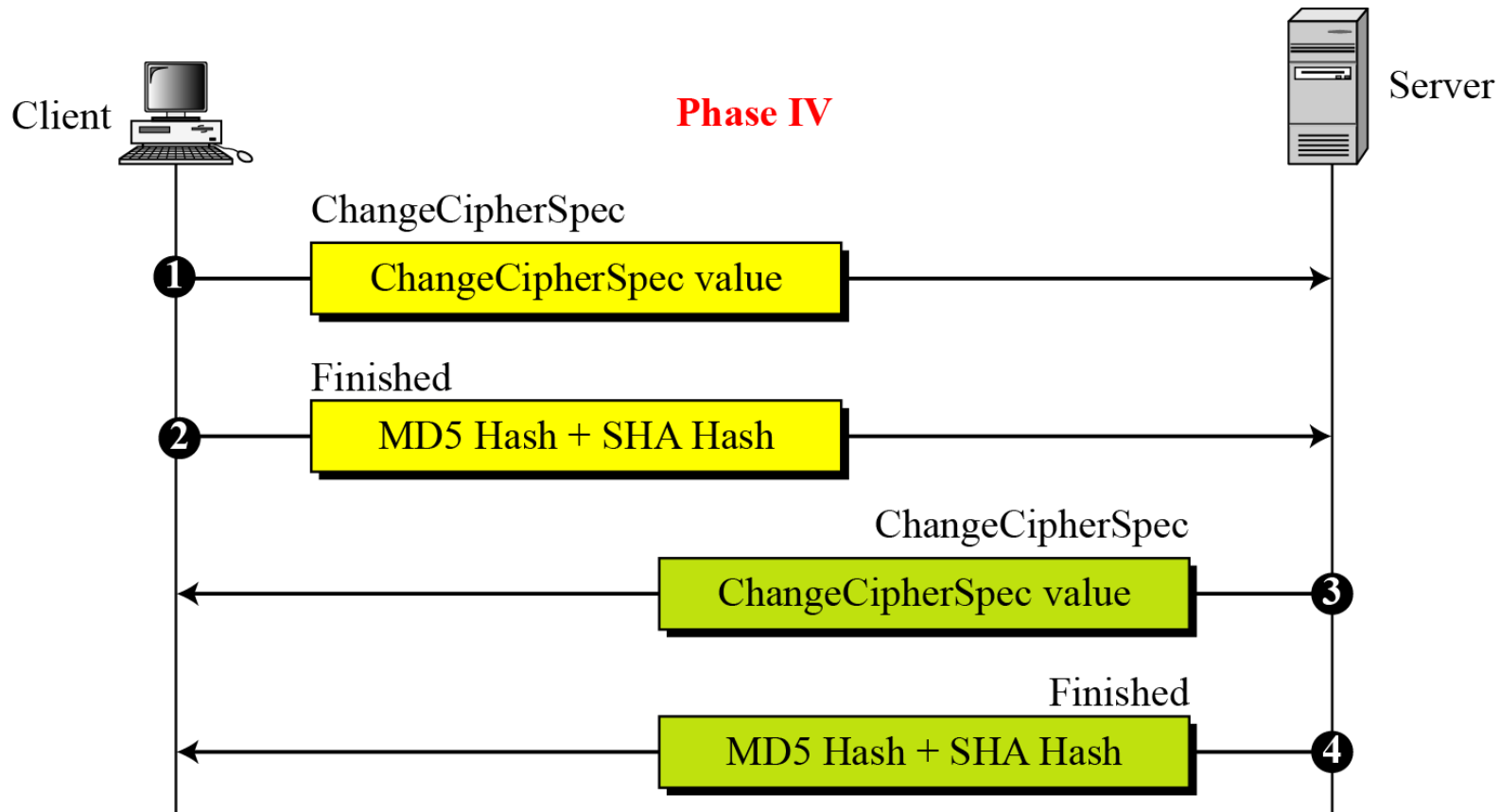


c. Ephemeral DH



d. Fixed DH

Phase IV of Handshake Protocol



Phase IV of Handshake Protocol

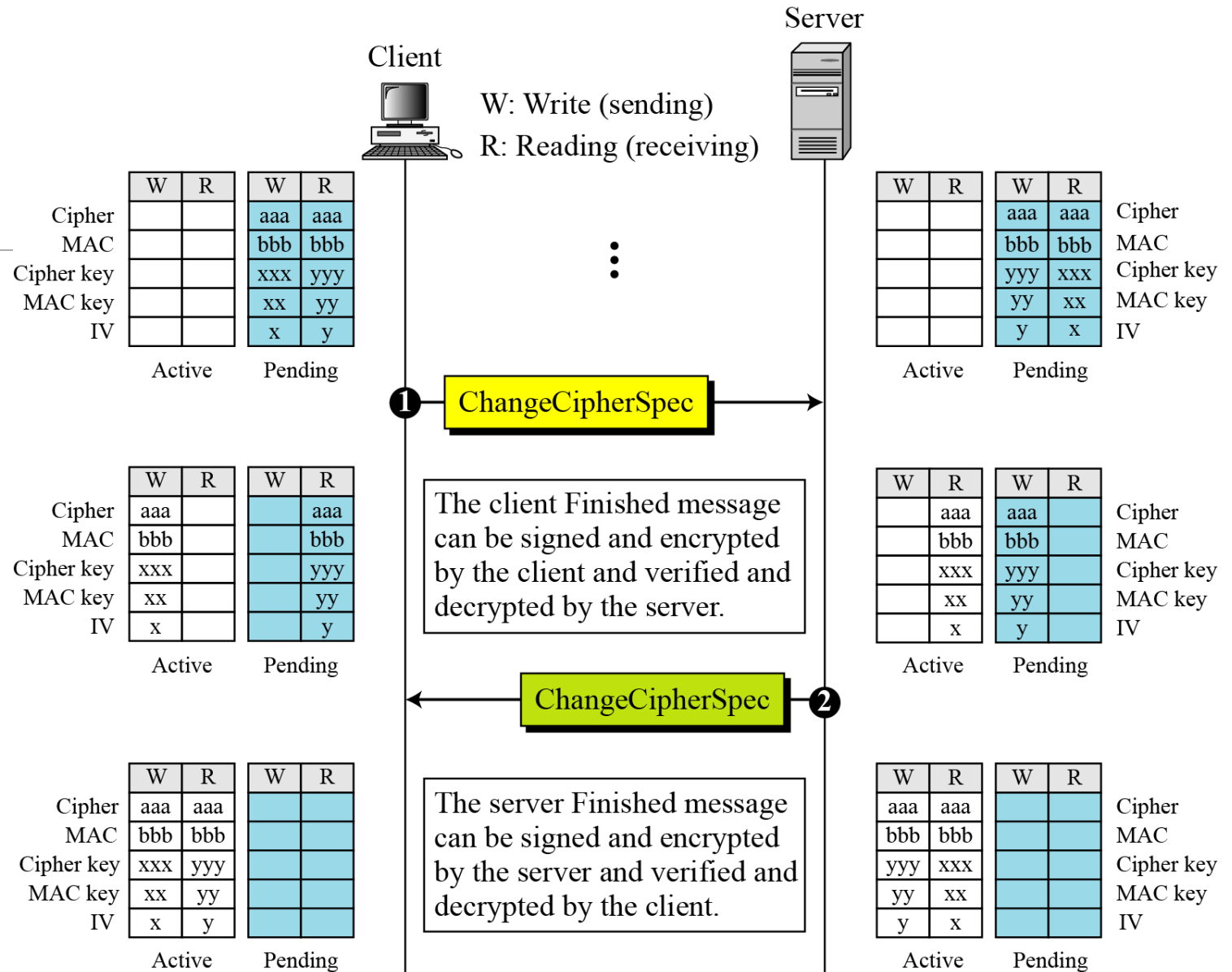
- After Phase IV, the client and server are ready to exchange data.

ChangeCipherSpec Protocol

- The negotiation of the cipher suite and the generation of cryptographic secrets are formed gradually during the handshake protocol
- When can two parties use these parameter secrets?
 - Only after ChangeCipherSpec message

ChangeCipherSpec Protocol

- Movement of parameters from pending state to active state

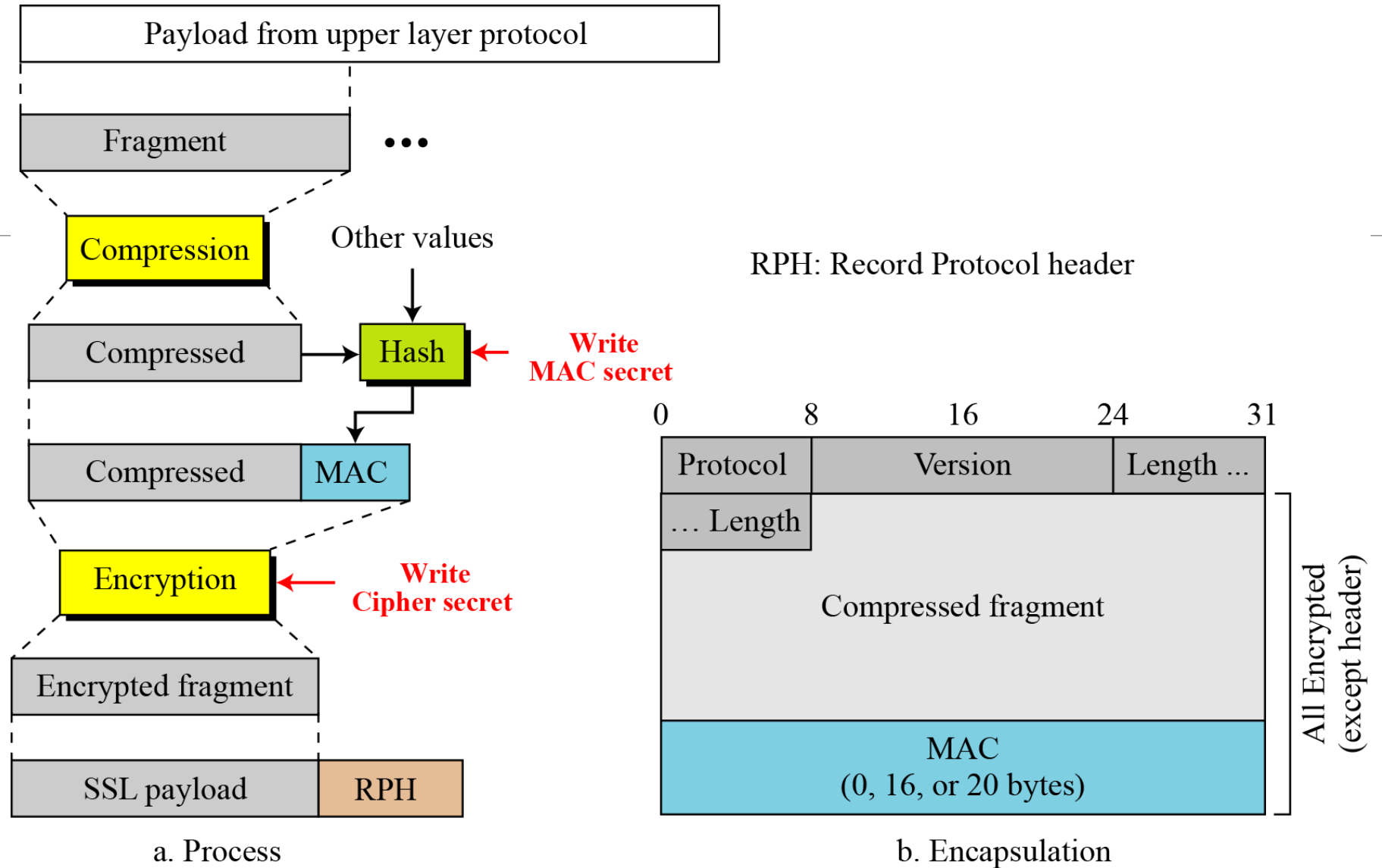


Alert Protocol

<i>Value</i>	<i>Description</i>	<i>Meaning</i>
0	<i>CloseNotify</i>	Sender will not send any more messages.
10	<i>UnexpectedMessage</i>	An inappropriate message received.
20	<i>BadRecordMAC</i>	An incorrect MAC received.
30	<i>DecompressionFailure</i>	Unable to decompress appropriately.
40	<i>HandshakeFailure</i>	Sender unable to finalize the handshake.
41	<i>NoCertificate</i>	Client has no certificate to send.
42	<i>BadCertificate</i>	Received certificate corrupted.
43	<i>UnsupportedCertificate</i>	Type of received certificate is not supported.
44	<i>CertificateRevoked</i>	Signer has revoked the certificate.
45	<i>CertificateExpired</i>	Certificate expired.
46	<i>CertificateUnknown</i>	Certificate unknown.
47	<i>IllegalParameter</i>	An out-of-range or inconsistent field.

SSL uses the Alert Protocol for reporting errors and abnormal conditions

Record Protocol

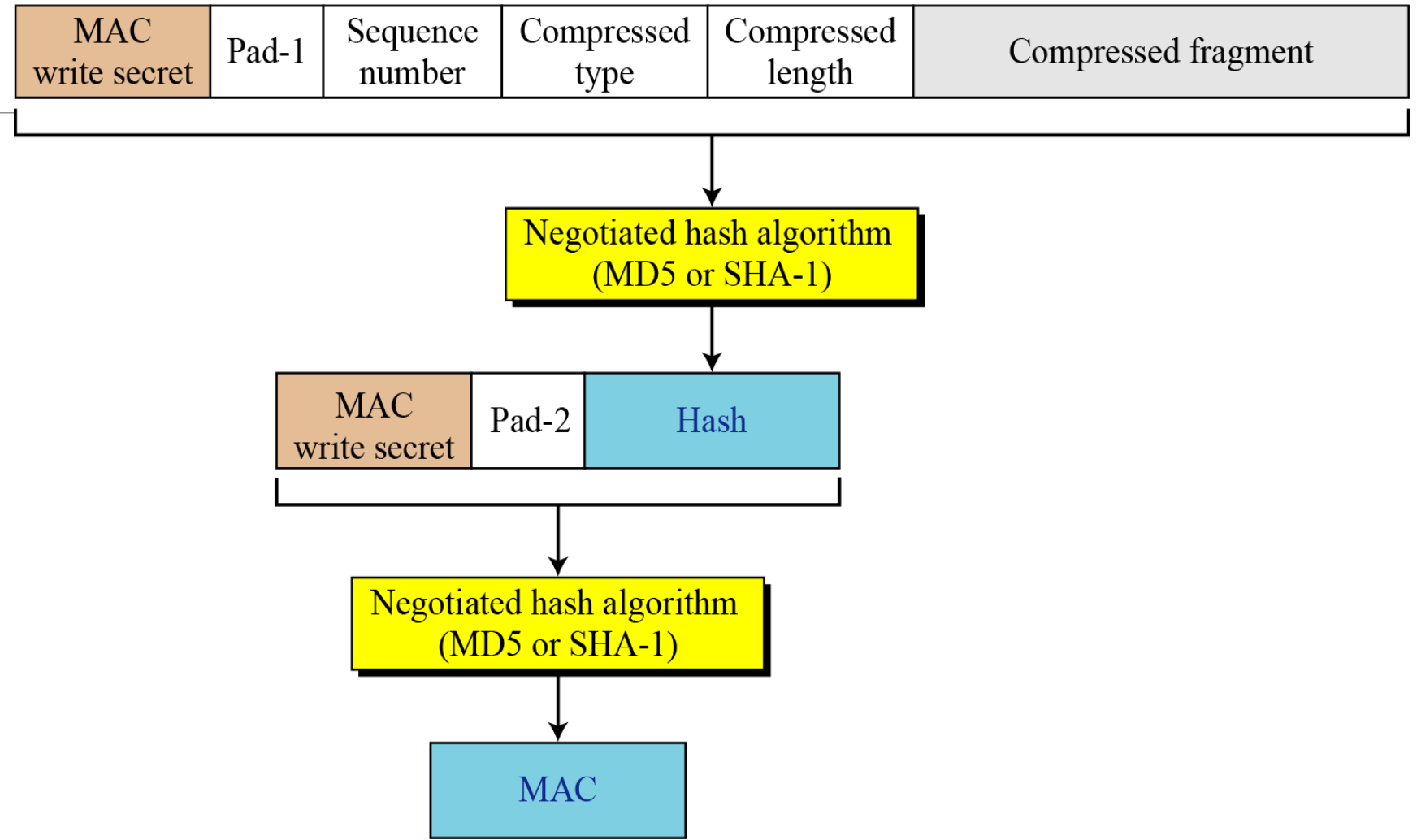


Processing done by the Record Protocol

Calculation of MAC

Pad-1: Byte 0x36 (00110110) repeated 48 times for MD5 and 40 times for SHA-1

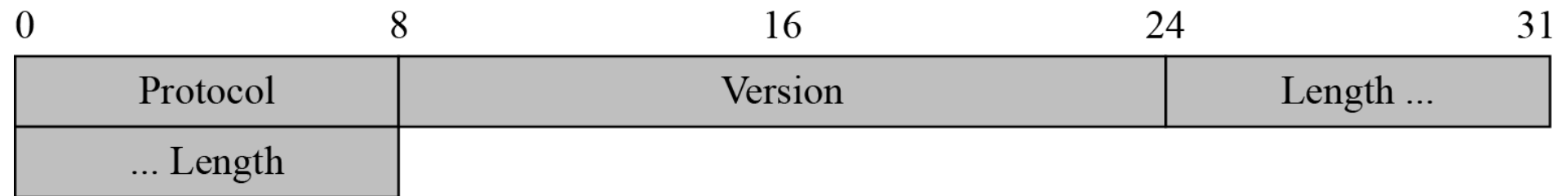
Pad-2: Byte 0x5C (01011100) repeated 48 times for MD5 and 40 times for SHA-1



SSL Message Formats

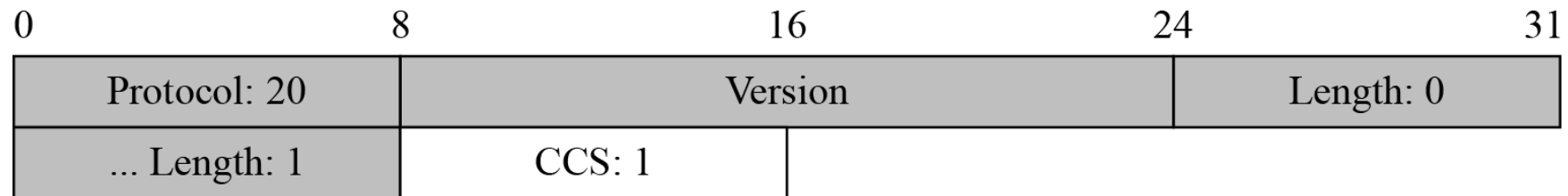
- messages from three protocols and data from the application layer are encapsulated in the Record Protocol messages.

Record Protocol Header



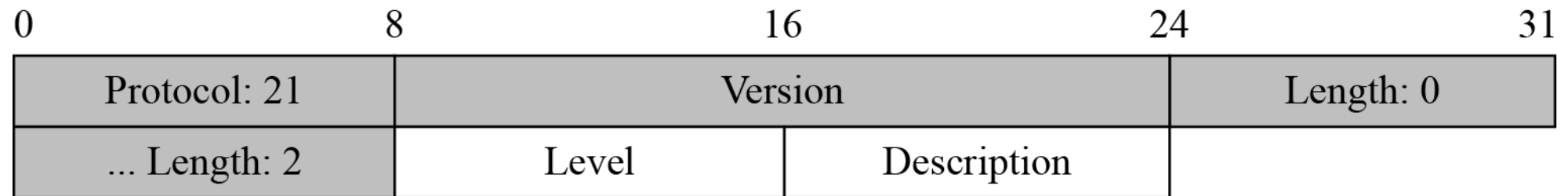
Record Protocol general header

ChangeCipherSpec Protocol

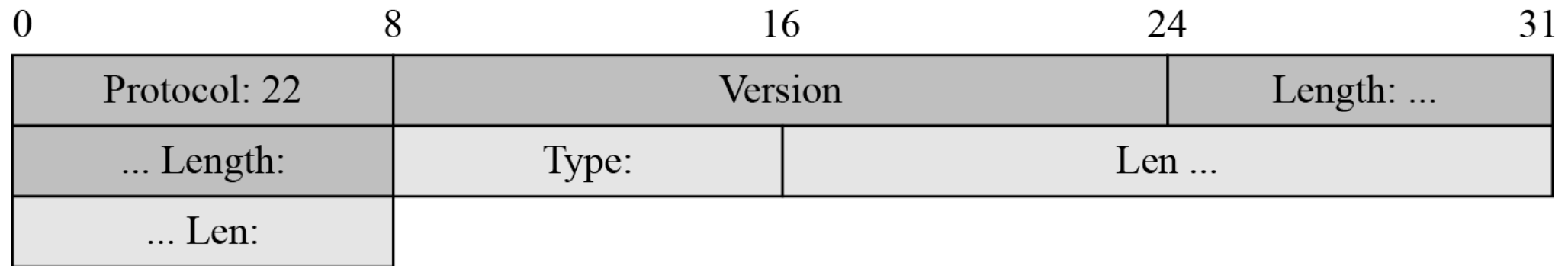


ChangeCipherSpec message

Alert Protocol



Generic header for Handshake Protocol



Types of Handshake messages

<i>Type</i>	<i>Message</i>
0	HelloRequest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
13	CertificateRequest
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Finished

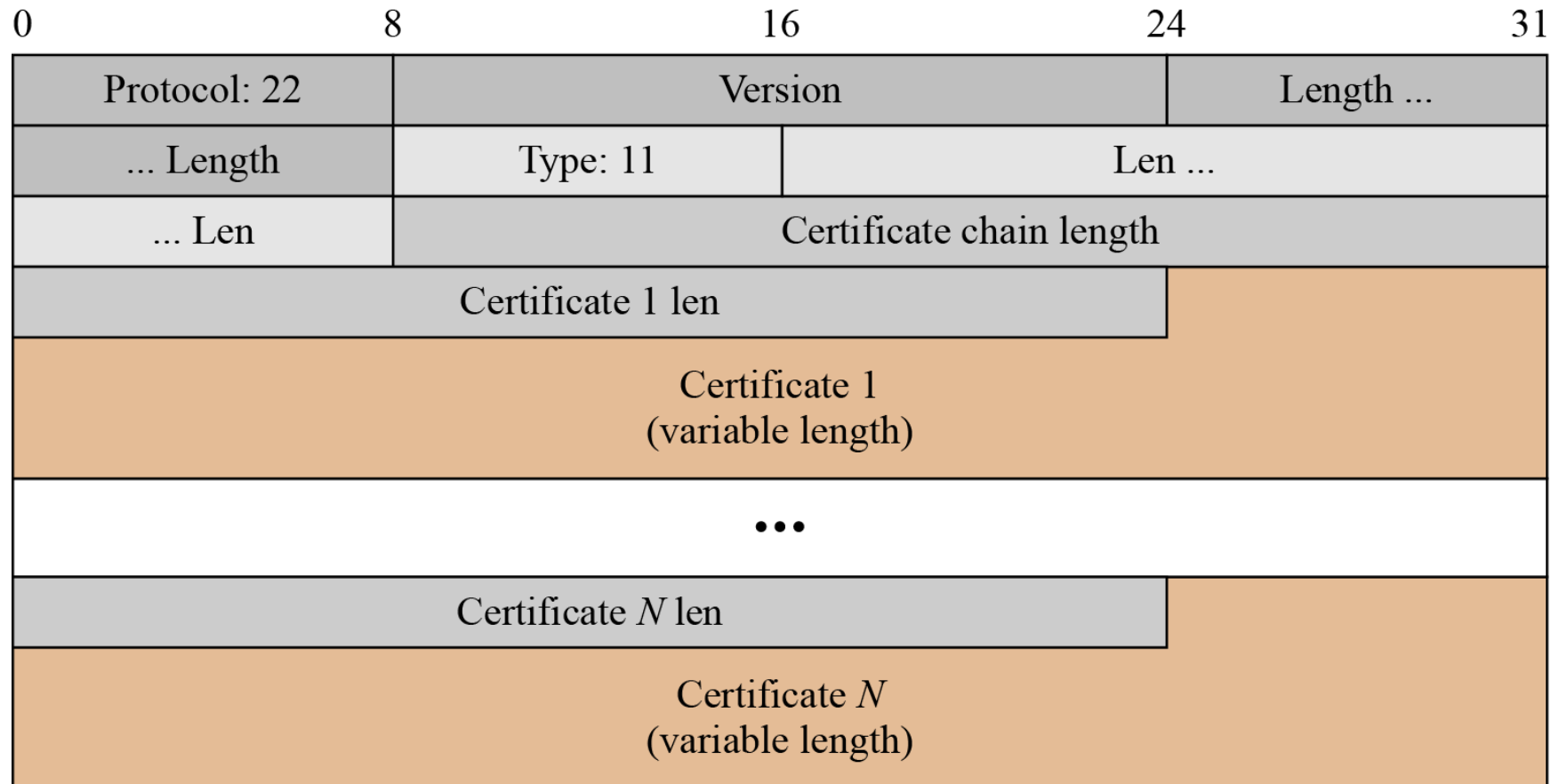
Client Hello Message

0	8	16	24	31
Protocol: 22		Version		Length ...
... Length		Type: 1	Len ...	
... Len		Proposed version		
Client random number (32 bytes)				ID length
Session ID (variable length)				
Cipher suite length		Cipher suites (variable numbers, each of 2 bytes)		
Com. methods length		Compression methods (variable number, each of 1 byte)		

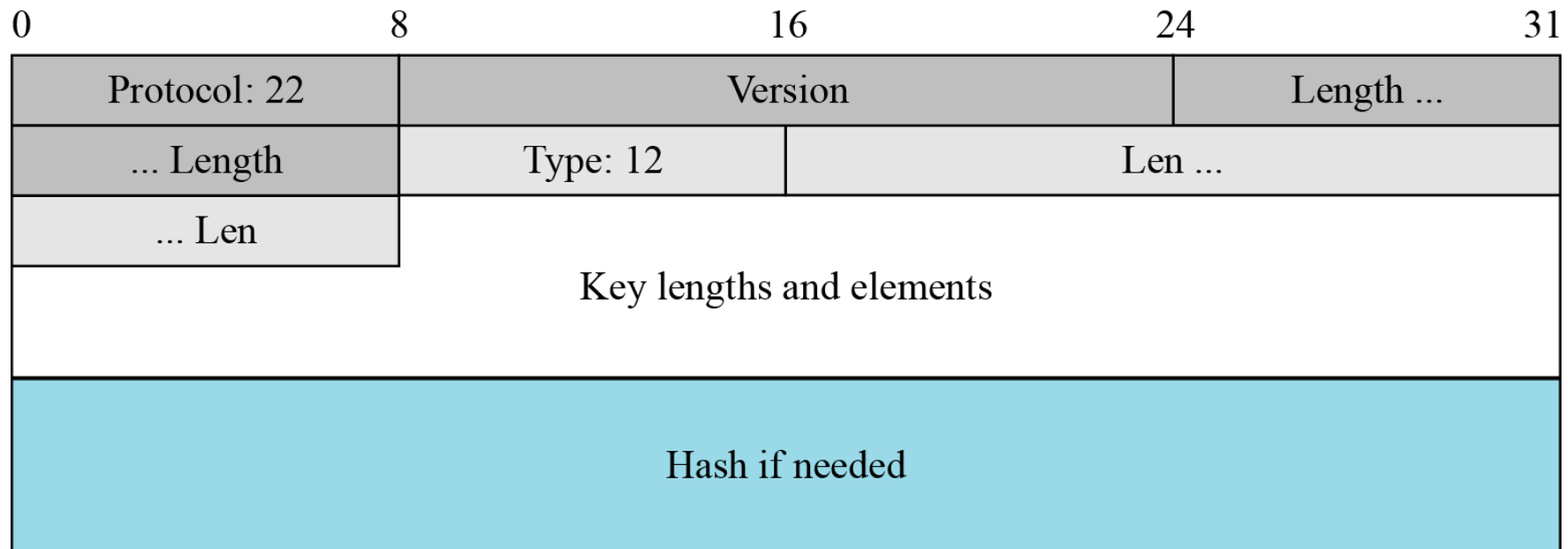
Server Hello Message

0	8	16	24	31
Protocol: 22		Version		Length ...
... Length		Type: 2	Len ...	
... Len		Proposed version		
Server random number (32 bytes)				
Session ID (variable length)				
Selected cipher suite		Selected com.		

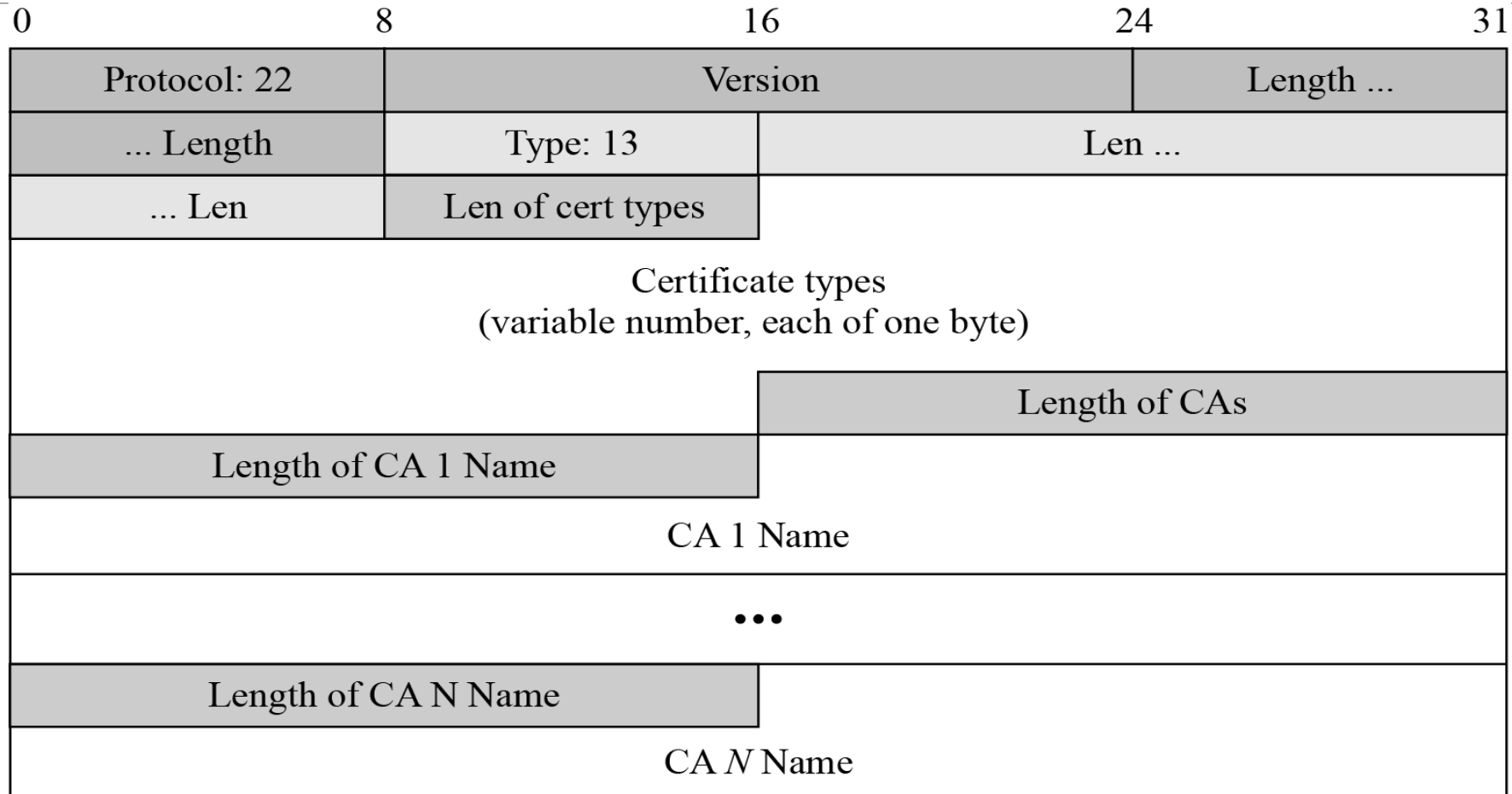
Certificate message



Server Key Exchange Message



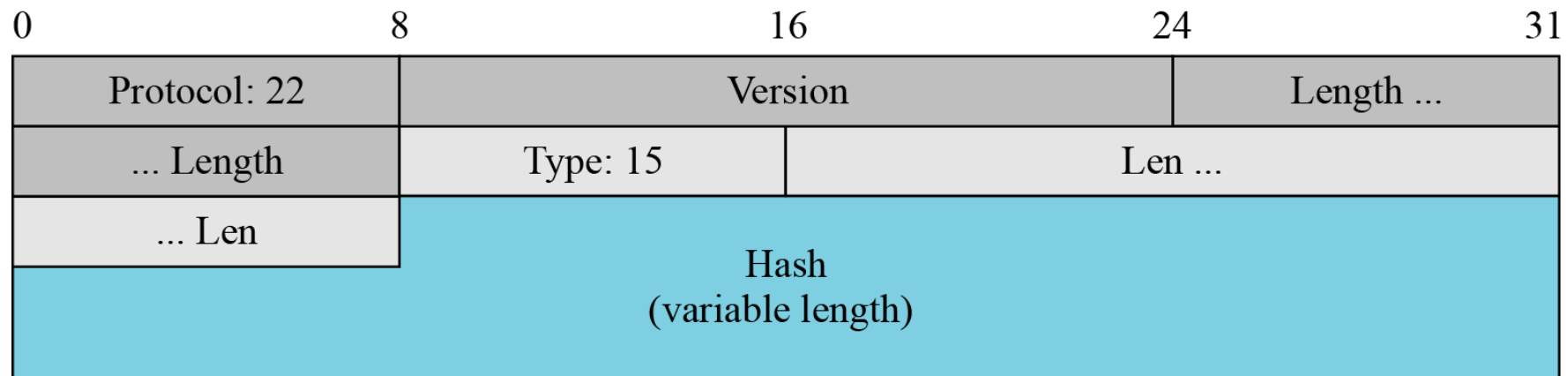
Certificate Request Message



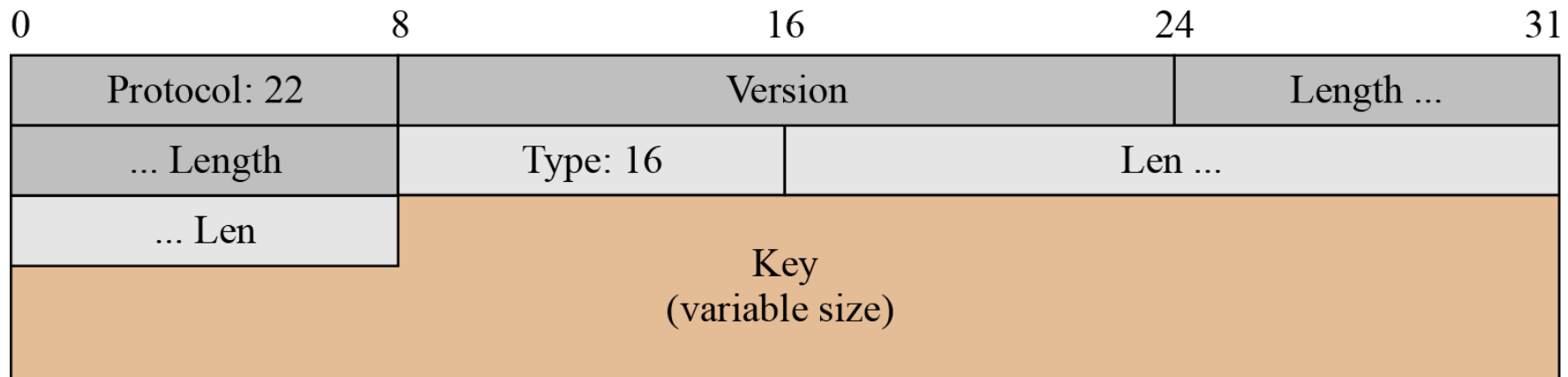
Server Hello Done Message

0	8	16	24	31
Protocol: 22		Version		Length ...
... Length: 4		Type: 14	Len ...	
... Len: 0				

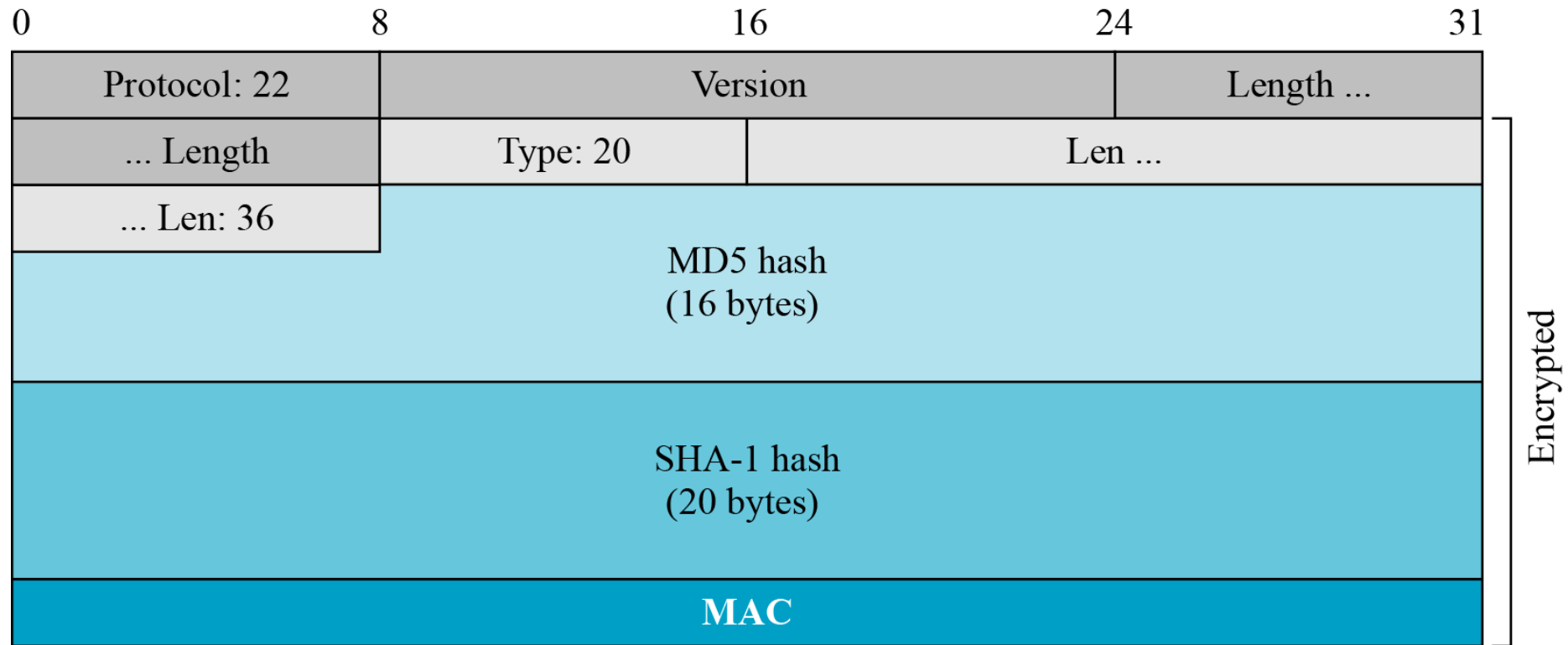
Certificate Verify Message



Client Key Exchange Message



Finished Message



Application Data

