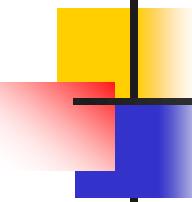




## Chapter 17

### *Security at the Transport Layer: SSL and TLS*



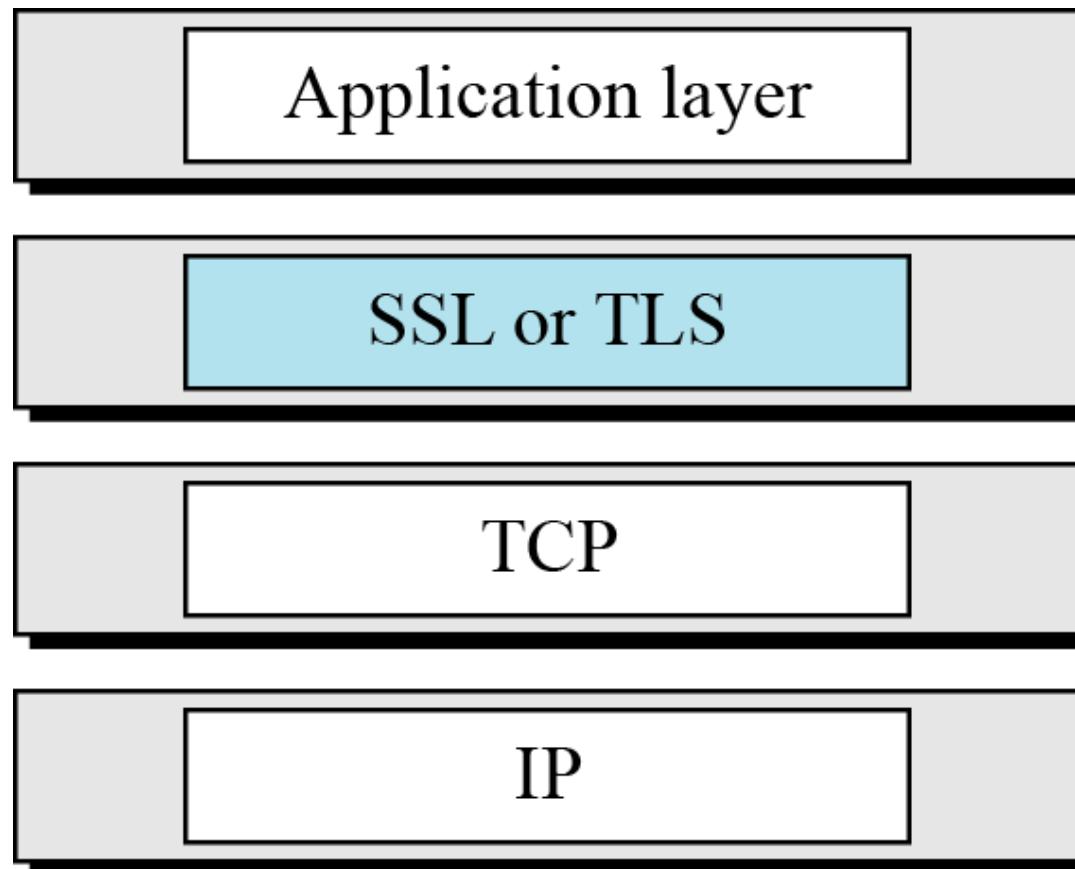
# **Chapter 17**

## **Objectives**

- To discuss the need for security services at the transport layer of the Internet model
- To discuss the general architecture of SSL
- To discuss the general architecture of TLS
- To compare and contrast SSL and TLS

# 17 Continued

**Figure 17.1** *Location of SSL and TLS in the Internet model*



# 17-1 SSL ARCHITECTURE

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

## Topics discussed in this section:

17.1.1 Services

17.1.2 Key Exchange Algorithms

17.1.3 Encryption/Decryption Alogrithms

17.1.4 Hash Algorithms

17.1.5 Cipher Suite

17.1.6 Compression Algorithms

17.1.7 Cryptography Parameter Generation

17.1.8 Session and Connections

## *17.1.1 Services*

**Fragmentation**

**Compression**

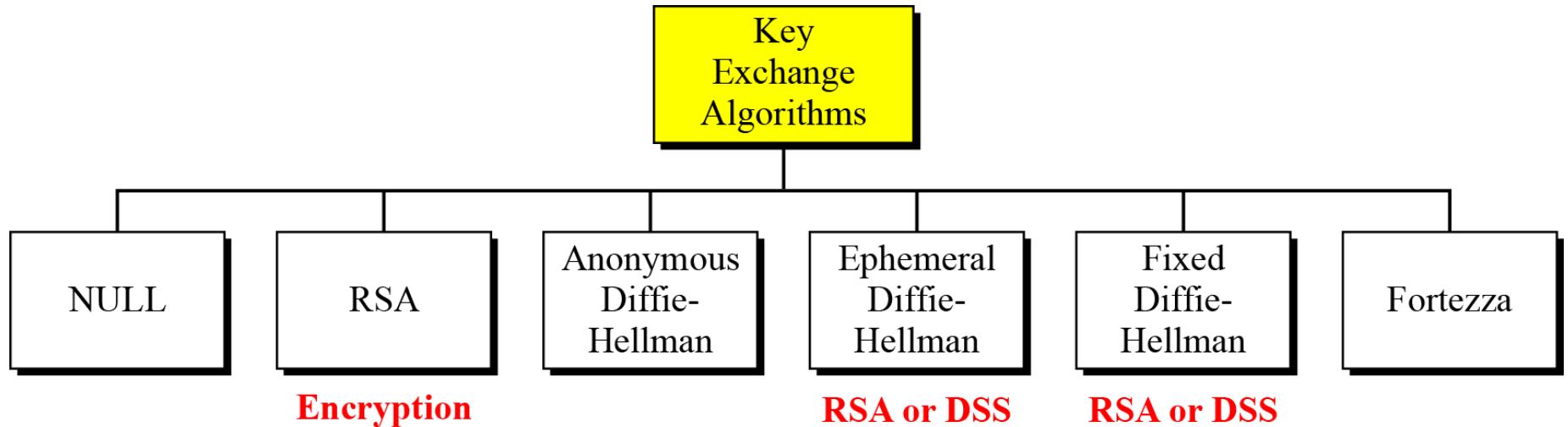
**Message Integrity**

**Confidentiality**

**Framing**

## 17.1.2 Key Exchange Algorithms

Figure 17.2 Key-exchange methods



## **17.1.2 Continued**

**Null**

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

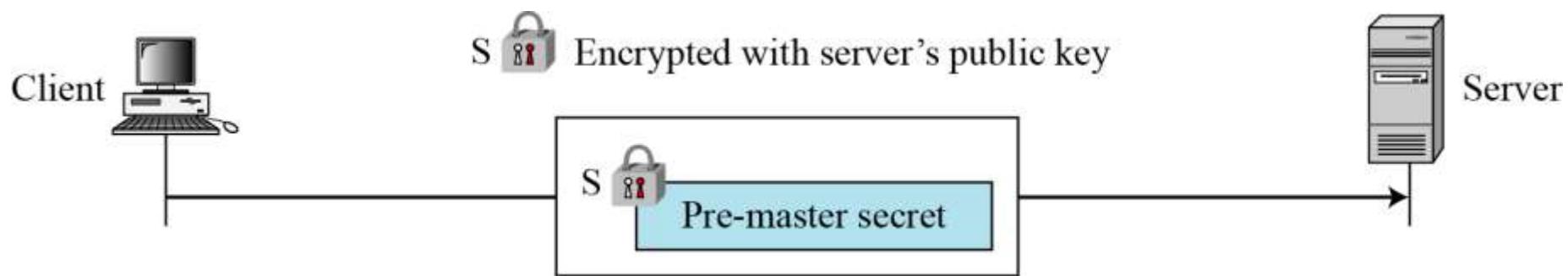
**Note**

**Both client and server need to know the value of the pre-master secret.**

## 17.1.2 *Continued*

### RSA

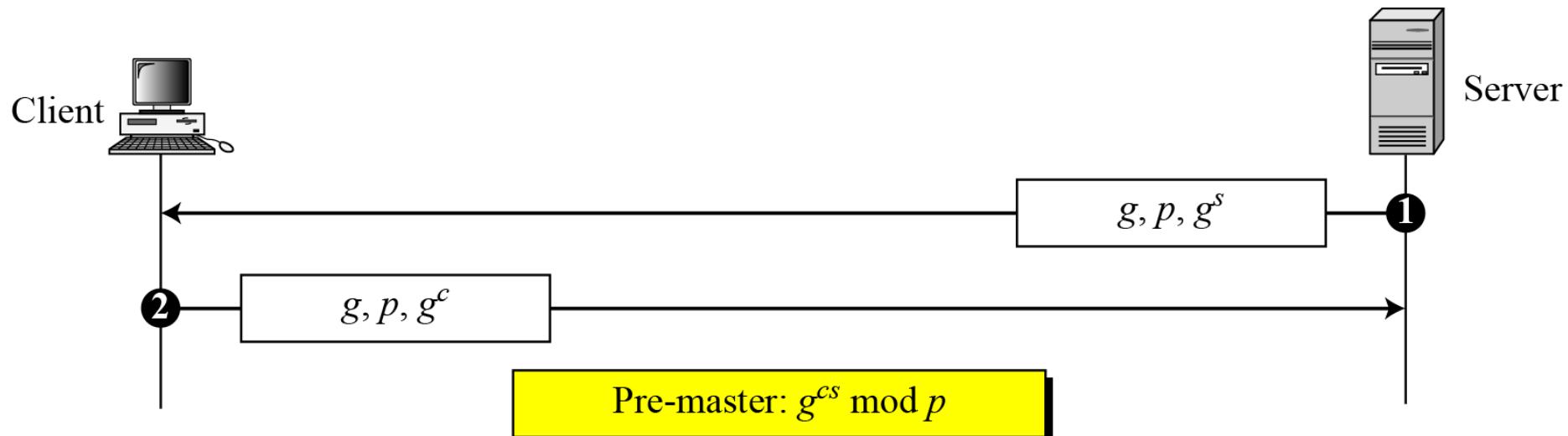
**Figure 17.3 RSA key exchange; server public key**



## 17.1.2 Continued

### Anonymous Diffie-Hellman

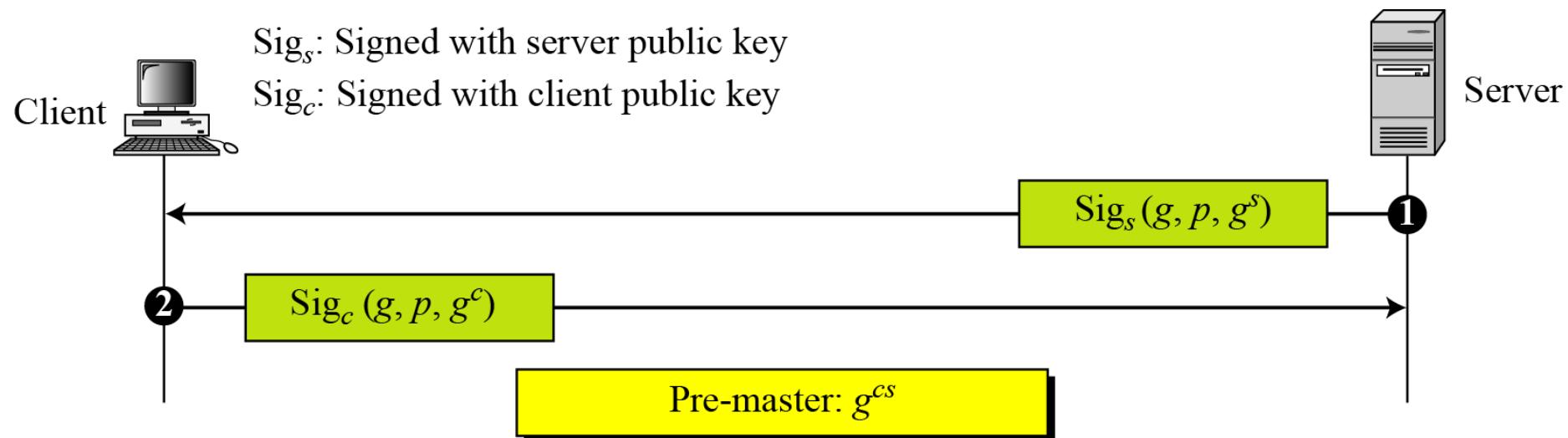
**Figure 17.4** *Anonymous Diffie-Hellman key exchange*



## 17.1.2 Continued

### Ephemeral Diffie-Hellman key exchange

**Figure 17.5 Ephemeral Diffie-Hellman key exchange**



## **17.1.2 Continued**

### **Fixed Diffie-Hellman**

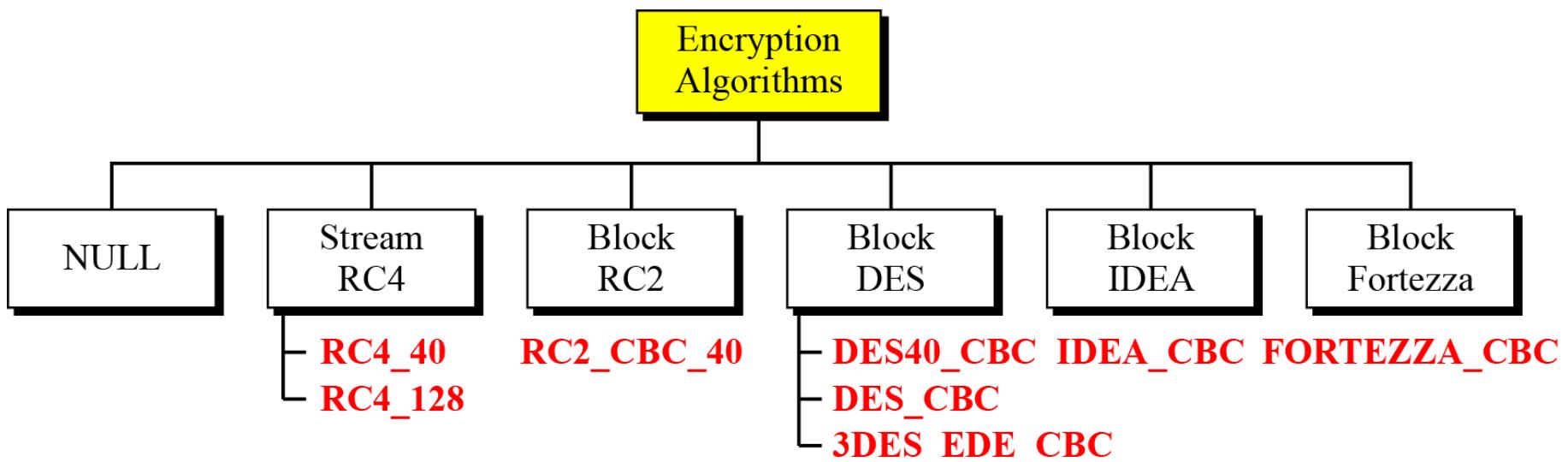
*Another solution is the fixed Diffie-Hellman method. All entities in a group can prepare fixed Diffie-Hellman parameters ( $g$  and  $p$ ).*

### **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.*

## 17.1.3 Encryption/Decryption Algorithms

Figure 17.6 Encryption/decryption algorithms



## **17.1.3 Continued**

### **NULL**

*The NULL category simply defines the lack of an encryption/decryption algorithm.*

### **Stream RC**

*Two RC algorithms are defined in stream mode.*

### **Block RC**

*One RC algorithm is defined in block mode.*

### **DES**

*All DES algorithms are defined in block mode.*

## **17.1.3 Continued**

### **IDEA**

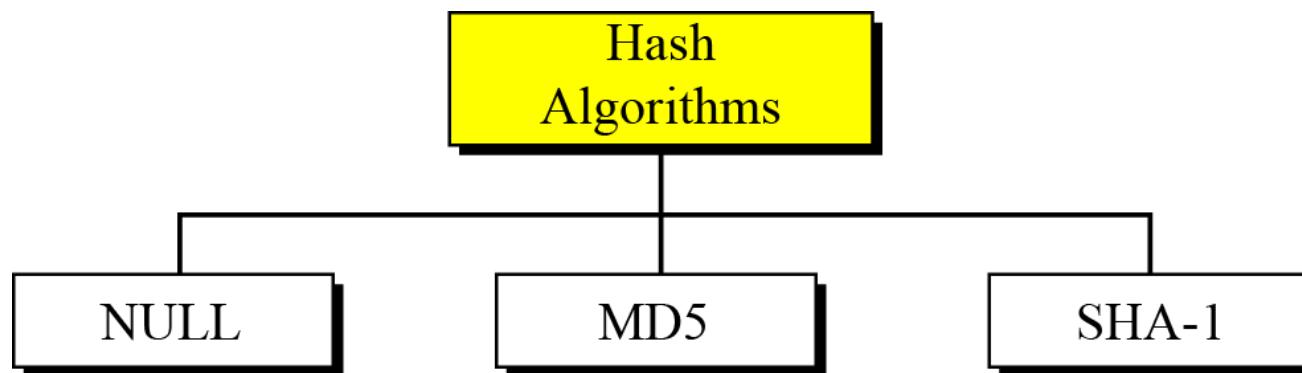
*The IDEA algorithm defined in block mode is IDEA\_CBC, with a 128-bit key.*

### **Fortezza**

*The one Fortezza algorithm defined in block mode is FORTEZZA\_CBC.*

## 17.1.4 Hash Algorithm

**Figure 17.7 Hash algorithms for message integrity**



## **17.1.4 Continued**

### **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.*

### **SHA-1**

*The two parties may choose SHA as the hash algorithm. In this case, a 160-bit SHA-1 hash algorithm is used.*

## 17.1.5 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

## 17.1.5 Continued

**Table 17.1 SSL cipher suite list**

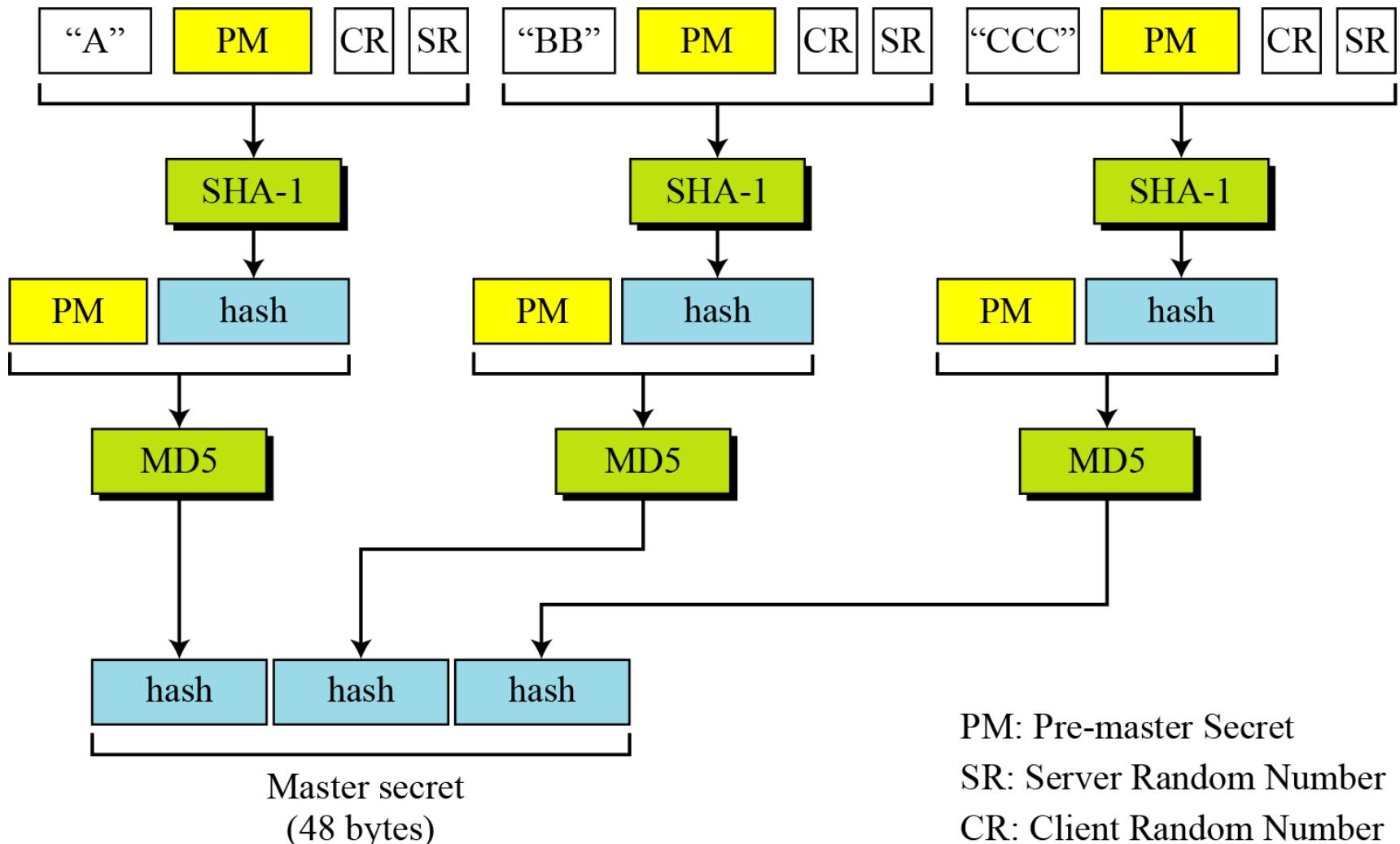
<i>Cipher suite</i>	<i>Key Exchange</i>	<i>Encryption</i>	<i>Hash</i>
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
SSL_RSA_WITH 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

## **17.1.6 Compression Algorithms**

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

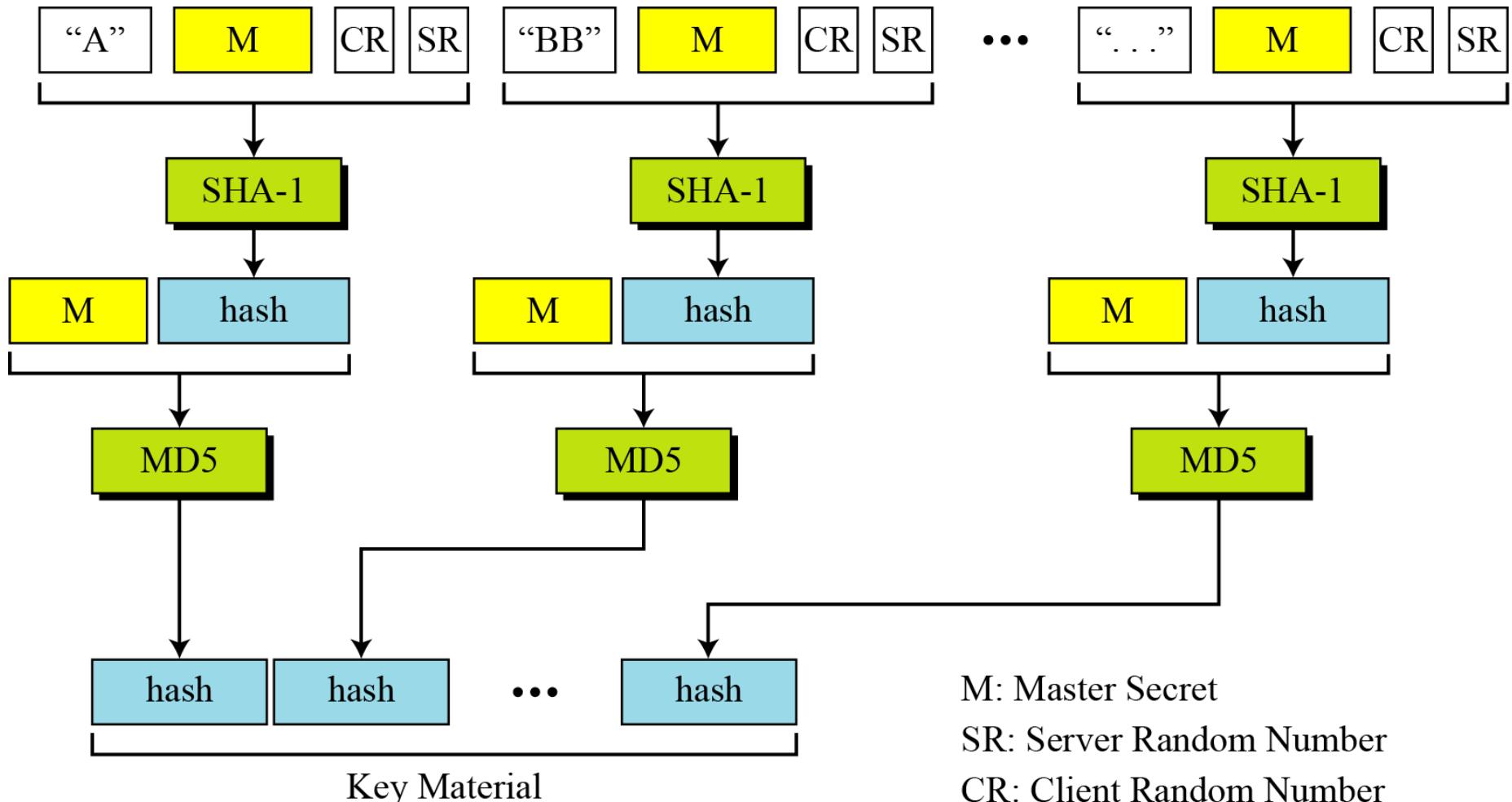
## 17.1.7 Cryptographic Parameter Generation

Figure 17.8 Calculation of master secret from pre-master secret



## 17.1.7 Continued

Figure 17.9 Calculation of key material from master secret



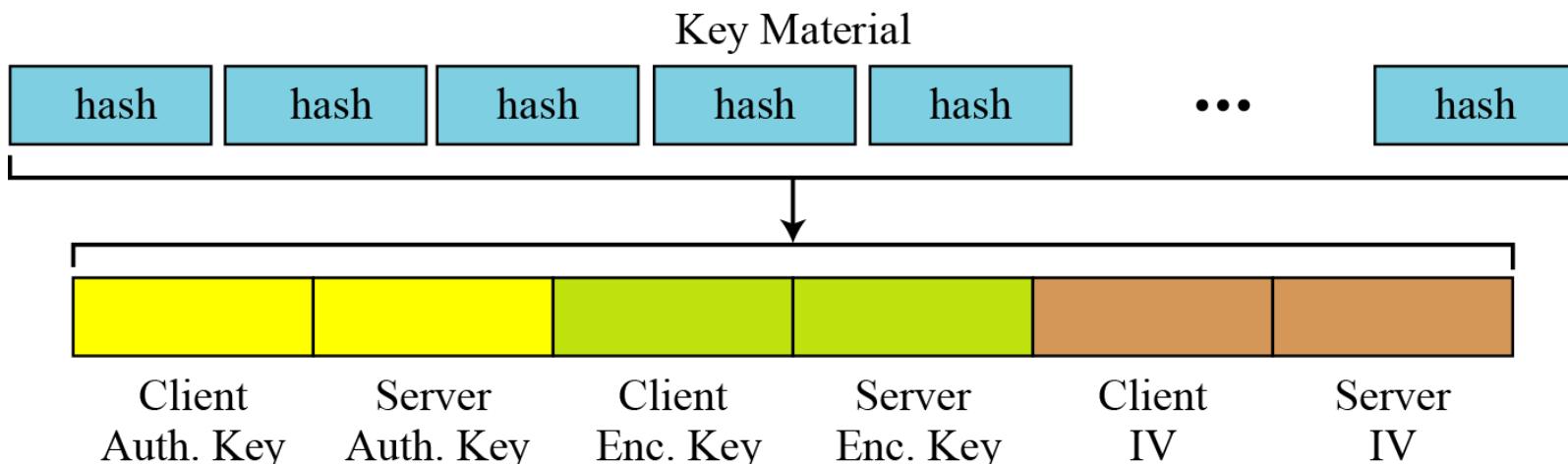
## 17.1.7 Continued

**Figure 17.10** *Extractions of cryptographic secrets from key material*

Auth. Key: Authentication Key

Enc. Key: Encryption Key

IV: Initialization Vector



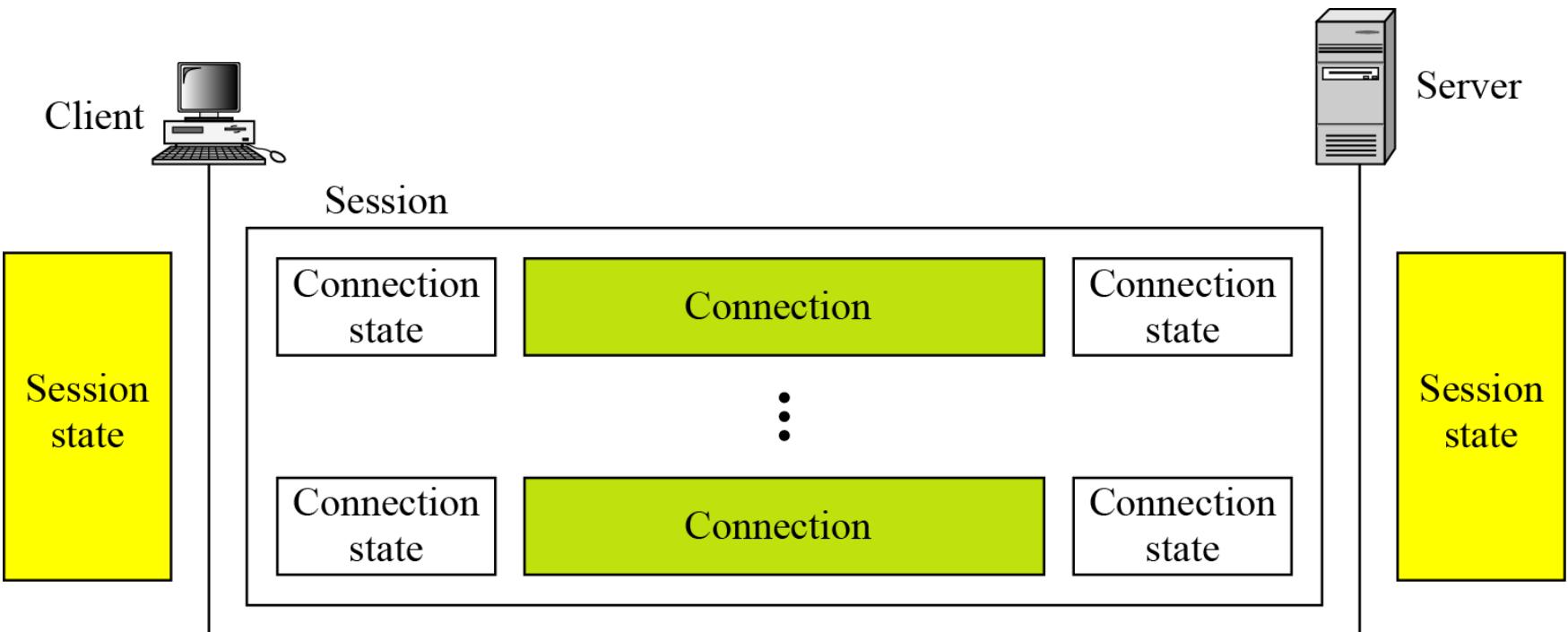
## *17.1.8 Sessions and Connections*

### **Note**

**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.**

## 17.1.8 Continued

**Figure 17.11 A session and connections**



## 17.1.8 Continued

### Session State

**Table 17.2 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 by 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.

## 17.1.8 Continued

### Connection State

**Table 17.3 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 $2^{64} - 1$ .

## **17.1.8 *Continued***

### **Note**

**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.**

## 17-2 Four Protocols

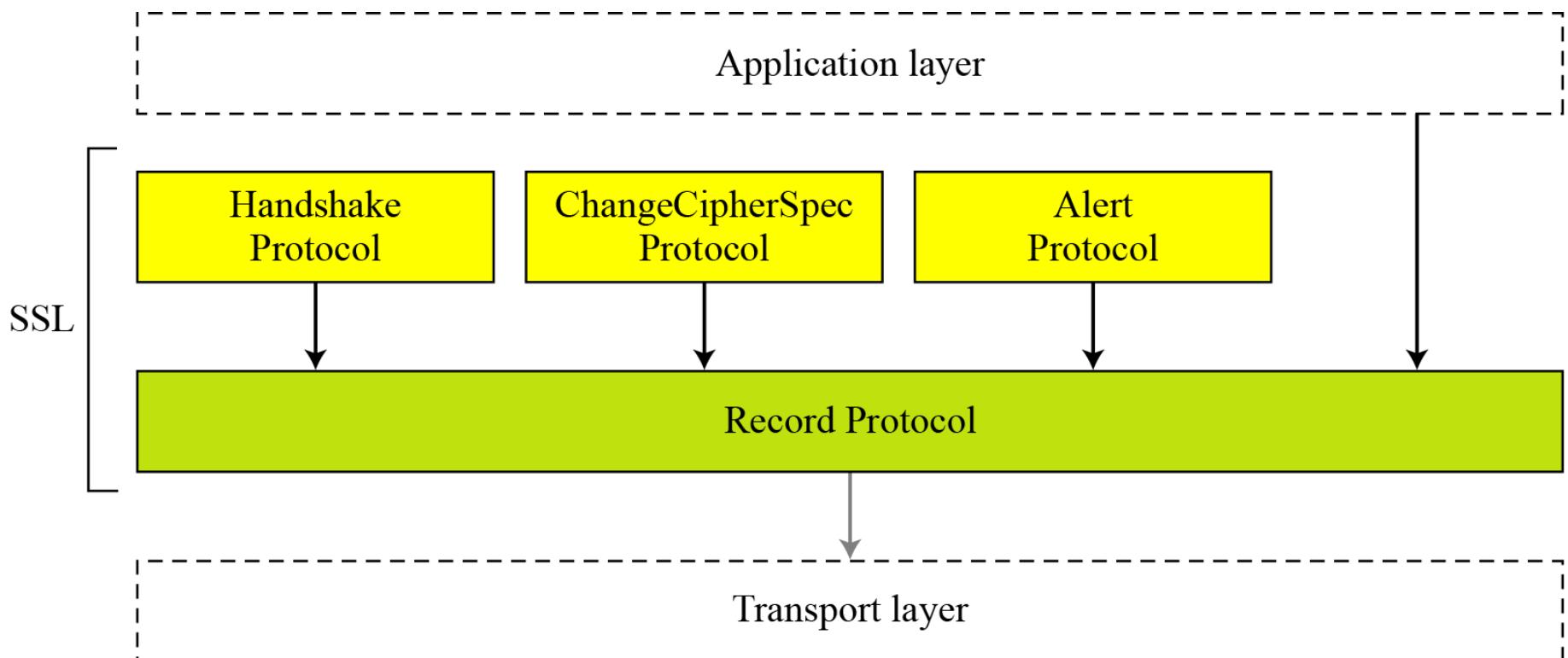
*We have discussed the idea of SSL without showing how SSL accomplishes its tasks. SSL defines four protocols in two layers, as shown in Figure 17.12.*

### Topics discussed in this section:

- 17.2.1 Handshake Protocol**
- 17.2.2 ChangeCipher Spec Protocol**
- 17.2.3 Alert Protocol**
- 17.2.4 Record Protocol**

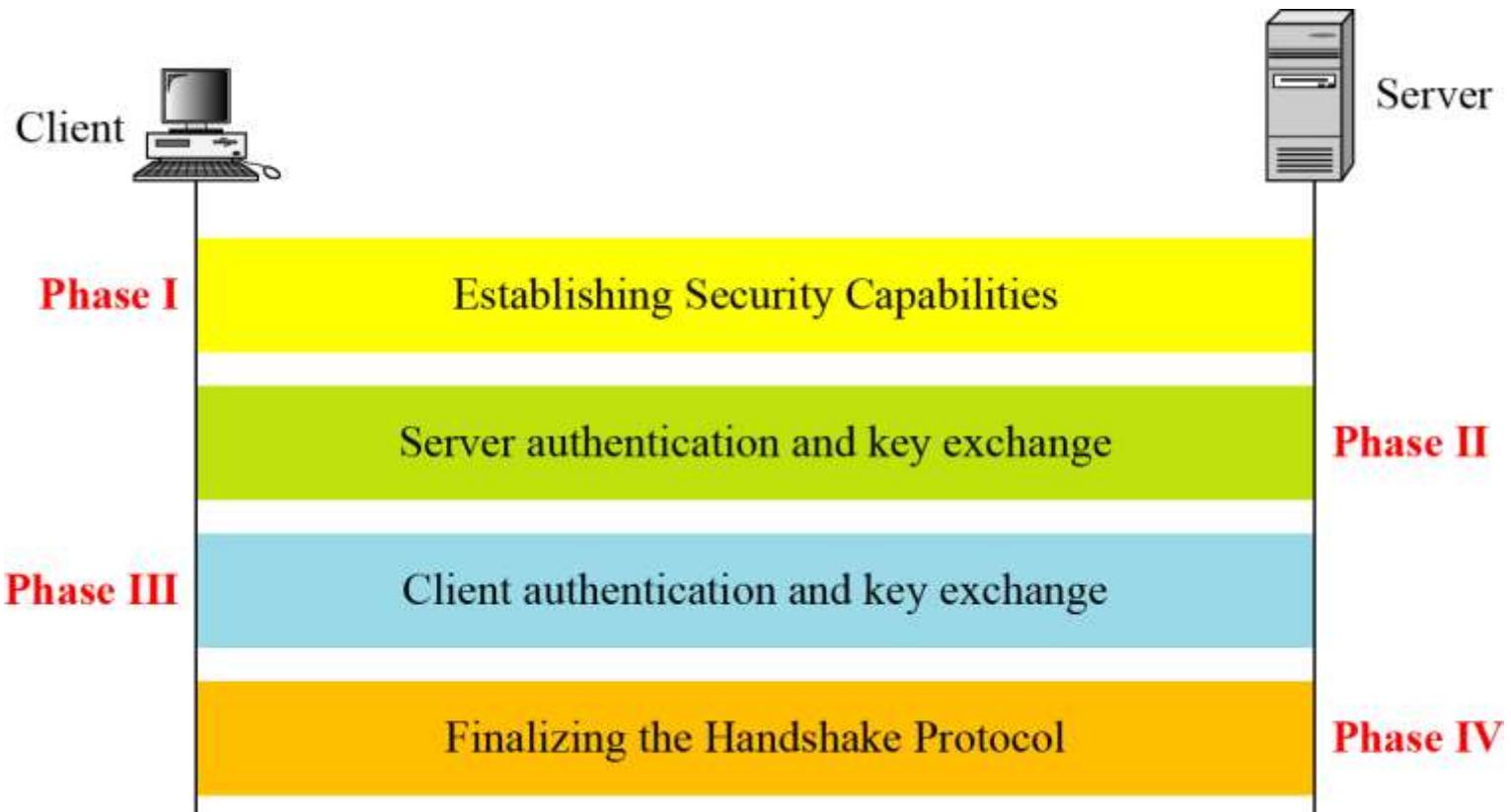
## 17.2. Continued

Figure 17.12 Four SSL protocols



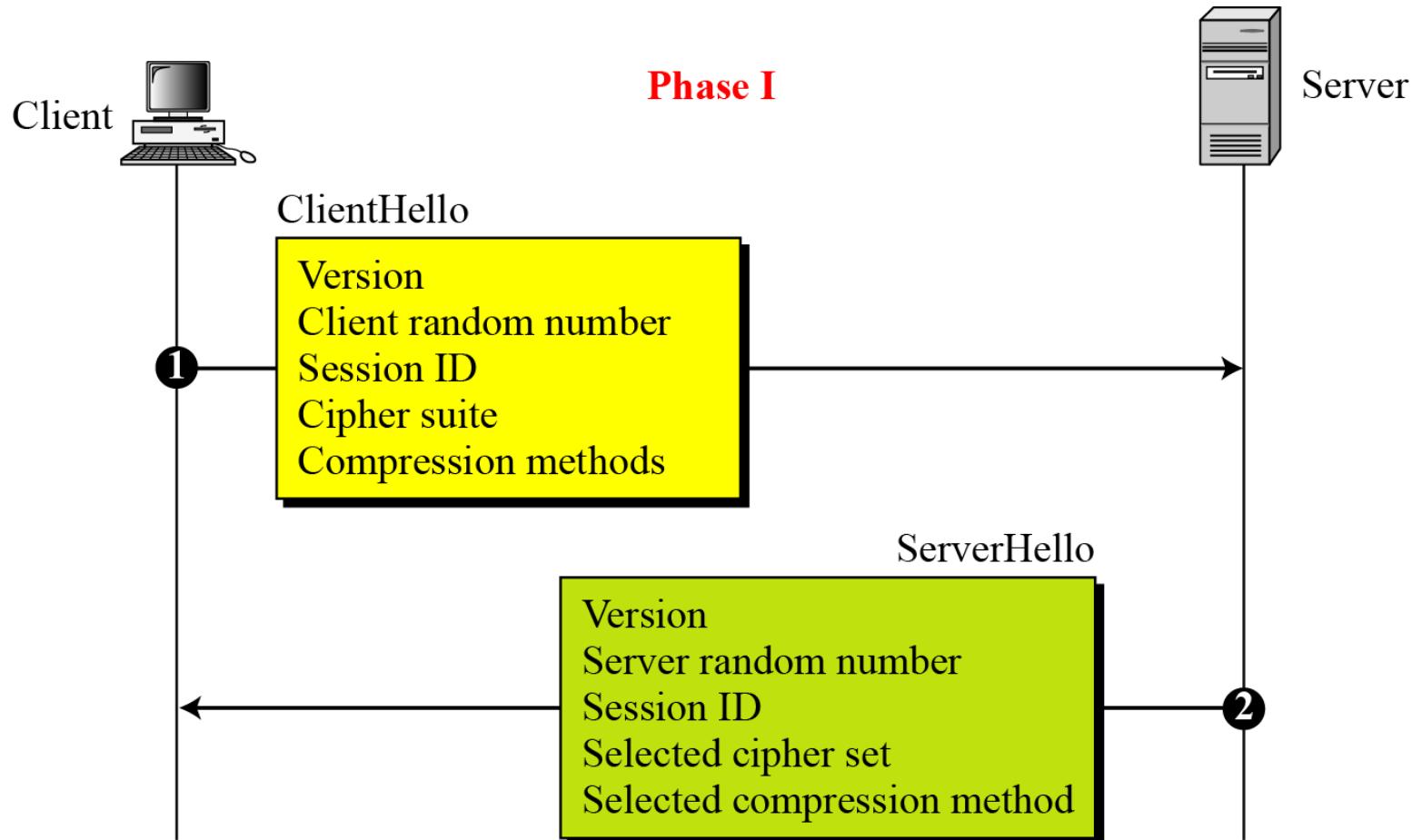
## 17.2.1 Handshake Protocol

Figure 17.13 Handshake Protocol



## 17.2.1 Continued

**Figure 17.14 Phase I of Handshake Protocol**



## **17.2.1 *Continued***

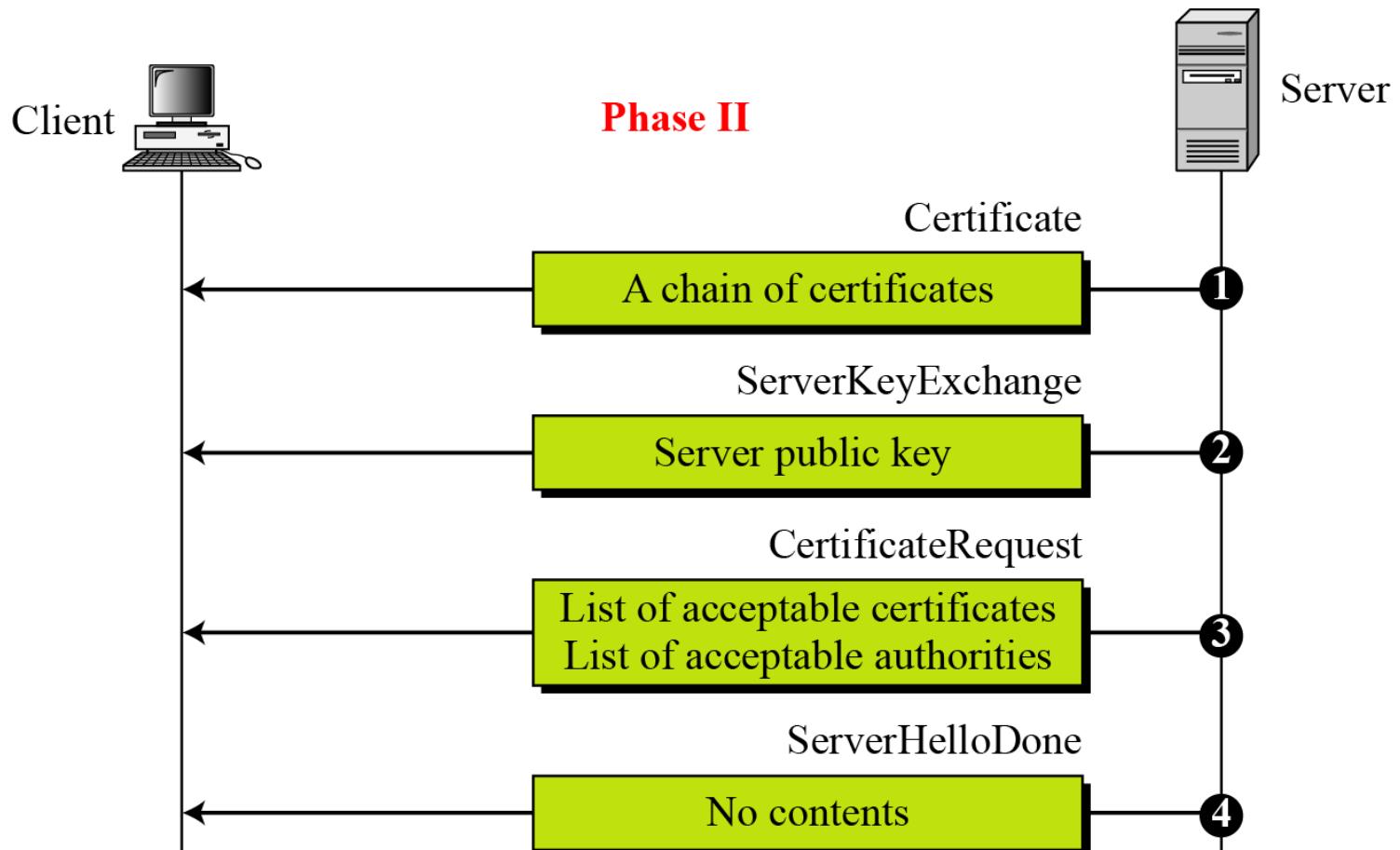
### **Note**

**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**

## 17.2.1 Continued

Figure 17.15 Phase II of Handshake Protocol



## **17.2.1 *Continued***

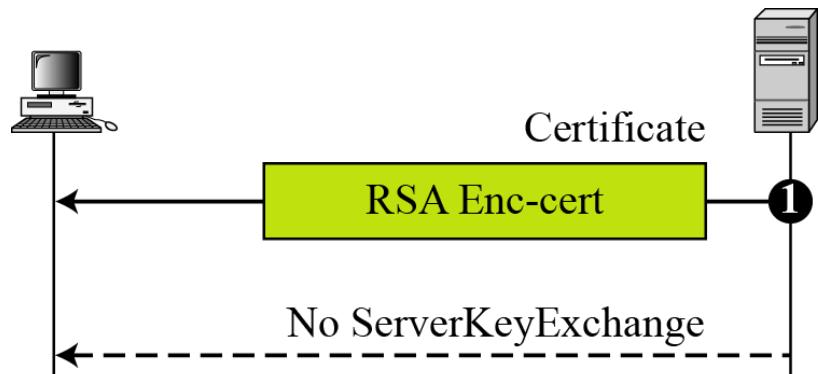
### **Note**

**After Phase II,**

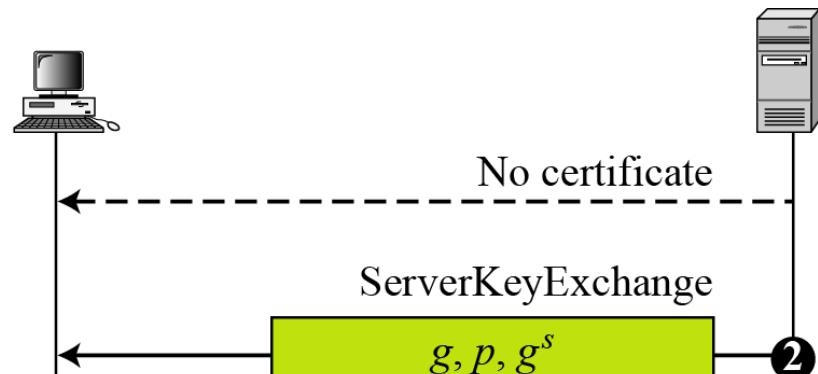
- The server is authenticated to the client.**
- The client knows the public key of the server if required.**

## 17.2.1 Continued

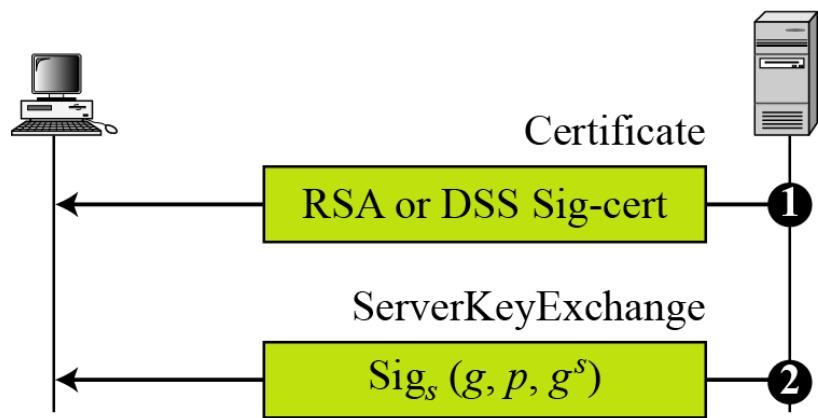
Figure 17.16 Four cases in Phase II



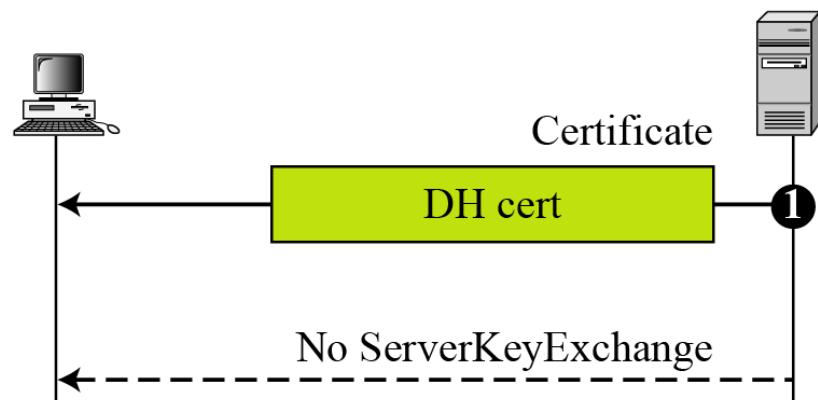
a. RSA



b. Anonymous DH



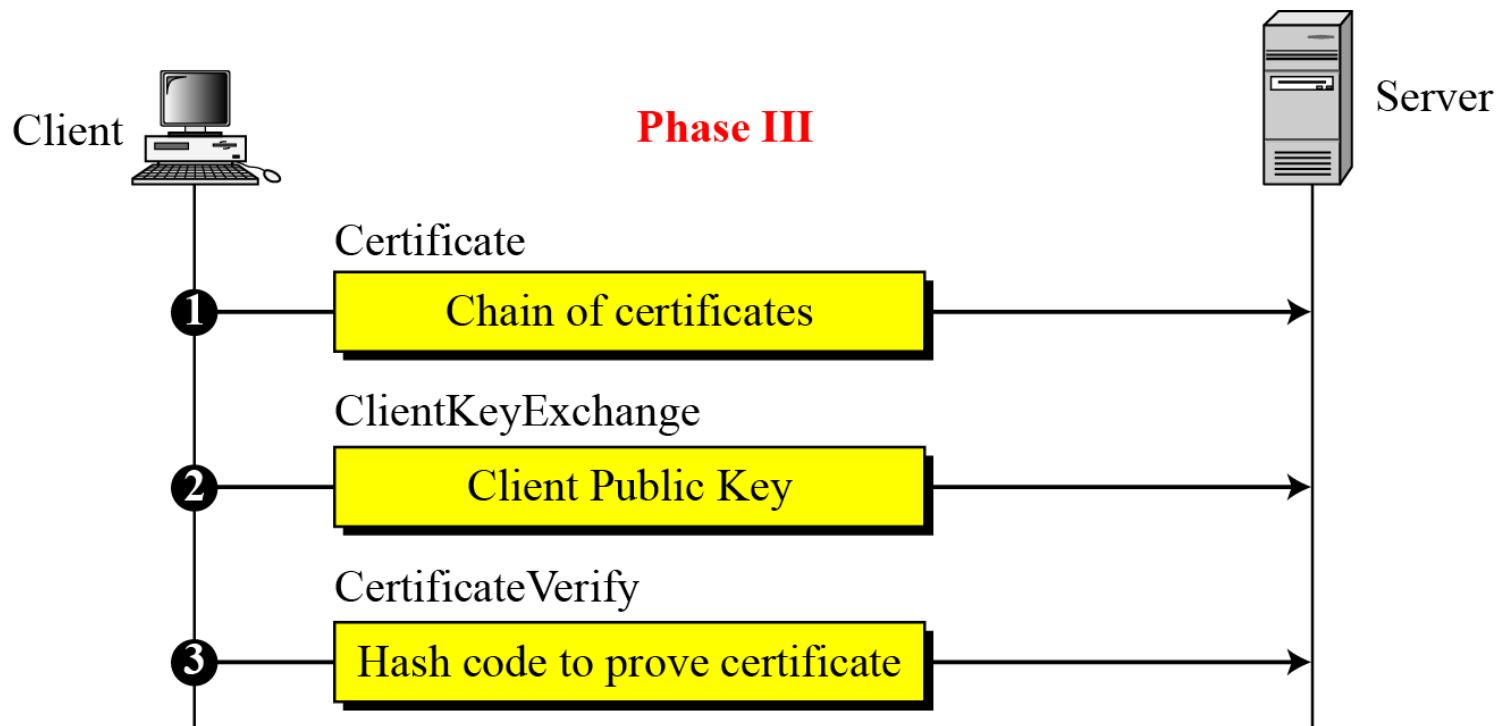
c. Ephemeral DH

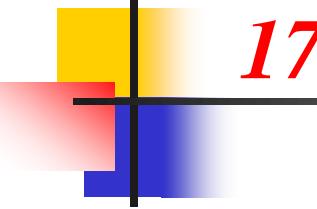


d. Fixed DH

## 17.2.1 Continued

**Figure 17.17 Phase III of Handshake Protocol**





## **17.2.1 Continued**

**Note**

**After Phase III,**

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

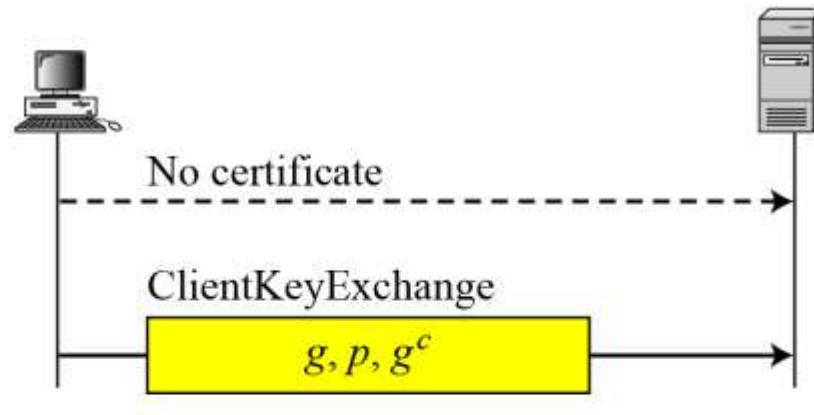
## 17.2.1 Continued

Figure 17.18 Four cases in Phase III

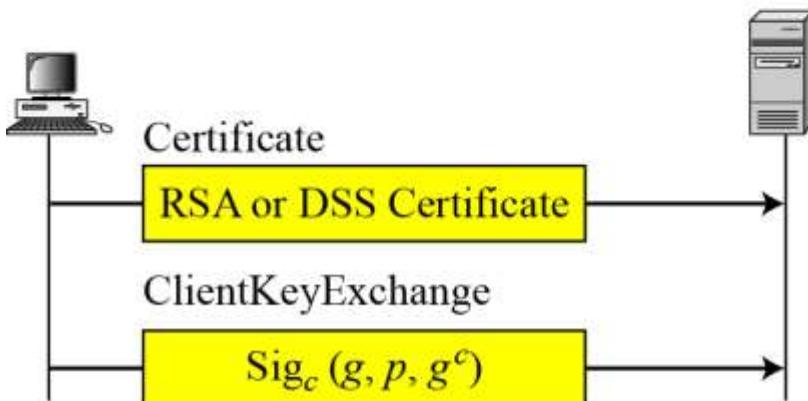
S encrypted with server's public key  
Sig<sub>c</sub>: Signed with client's public key



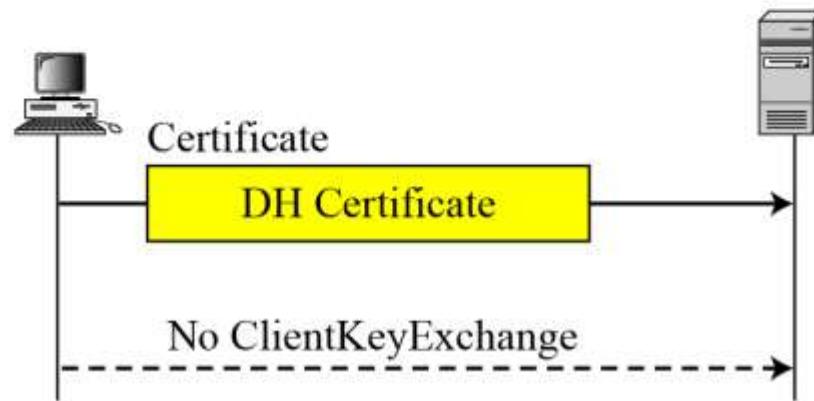
a. RSA



b. Anonymous DH



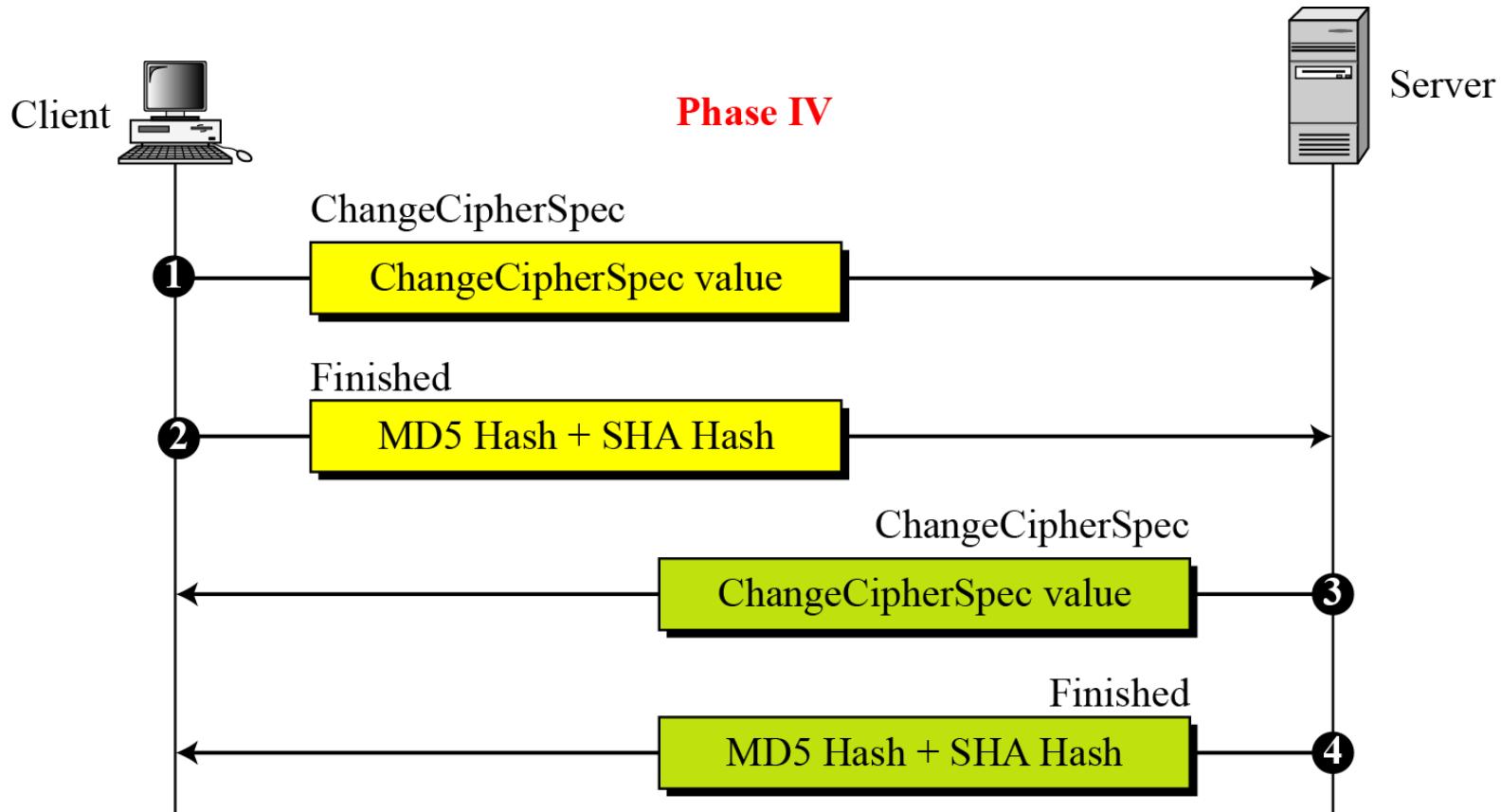
c. Ephemeral DH

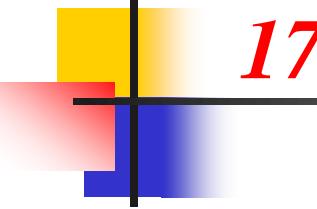


d. Fixed DH

## 17.2.1 Continued

Figure 17.19 Phase IV of Handshake Protocol





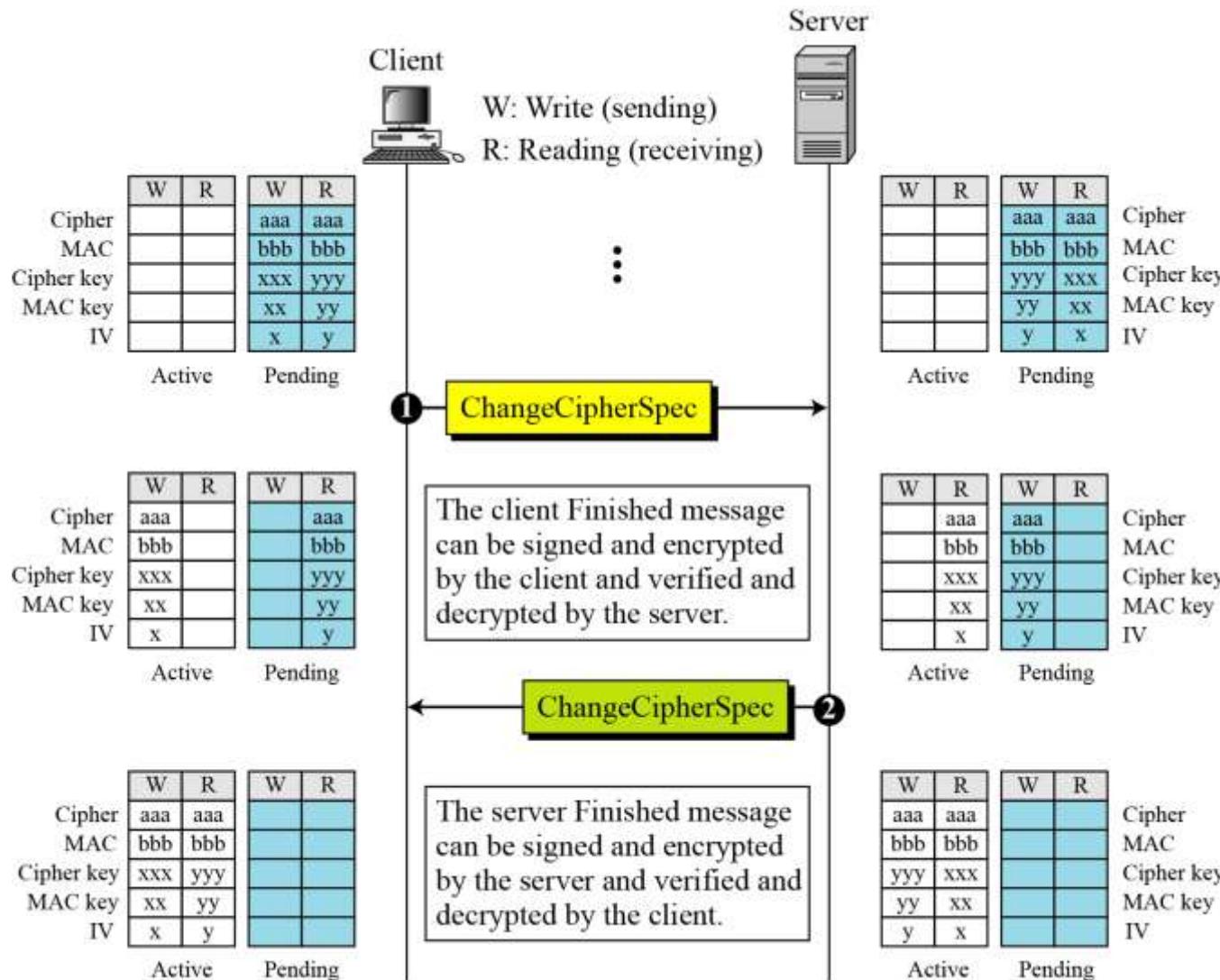
## ***17.2.1 Continued***

***Note***

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

## 17.2.2 ChangeCipherSpec Protocol

Figure 17.20 Movement of parameters from pending state to active state



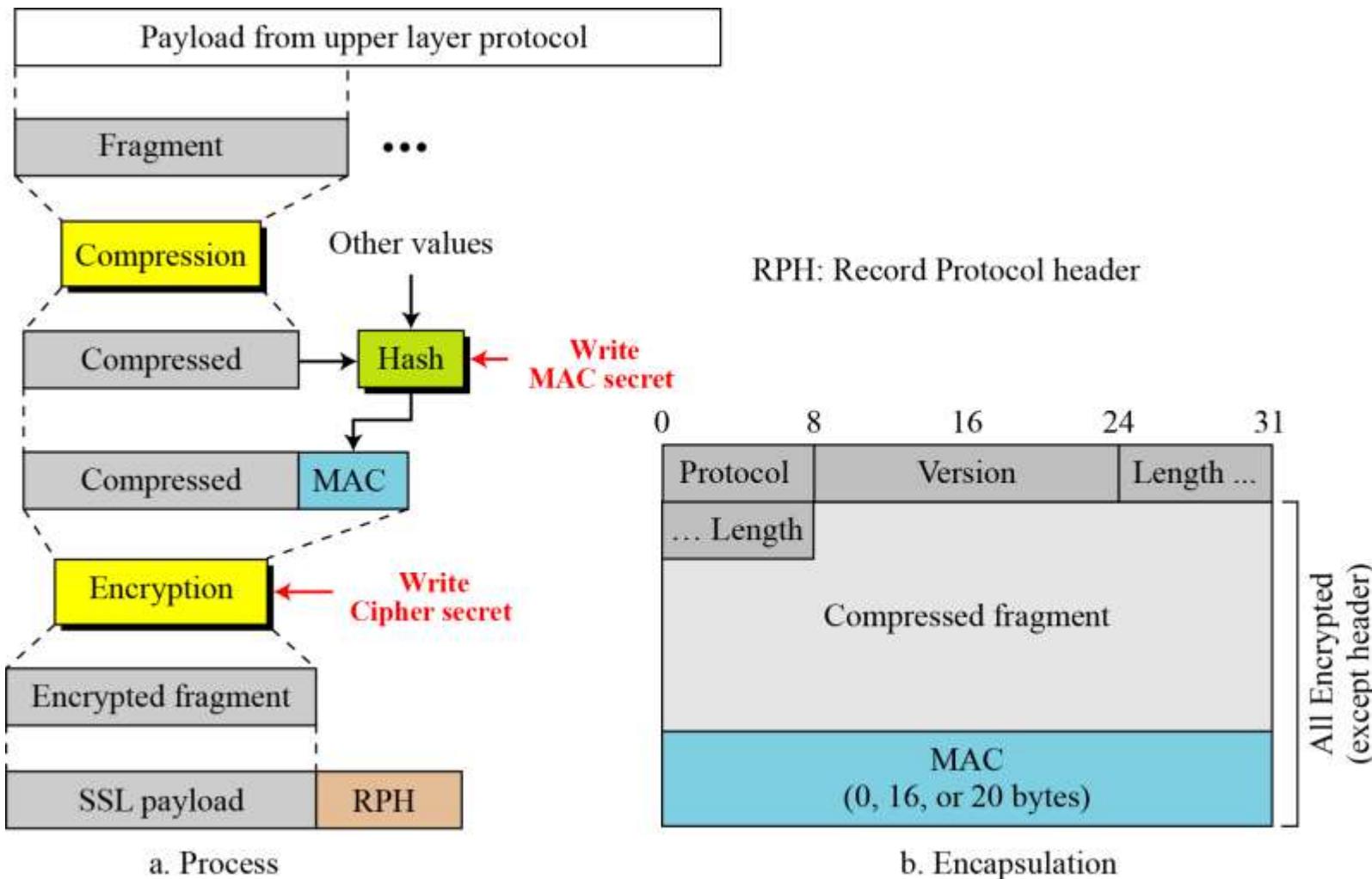
## 17.2.3 Alert Protocol

**Table 17.4 Alerts defined for SSL**

<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.

## 17.2.4 Record Protocol

Figure 17.21 Processing done by the Record Protocol



a. Process

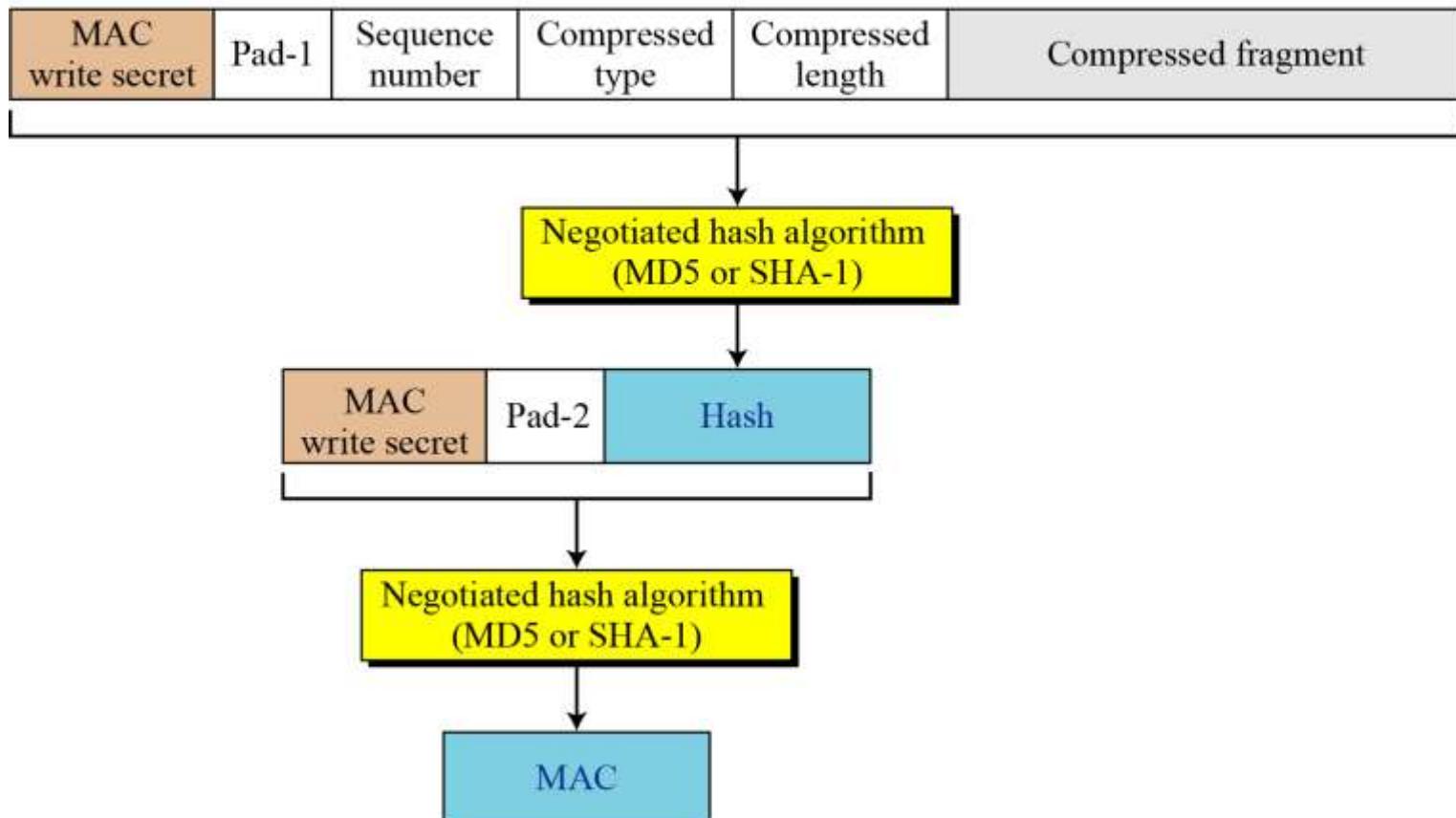
b. Encapsulation

## 17.2.4 Continued

**Figure 17.22 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



## 17-3 SSL MESSAGE FORMATS

*As we have discussed, messages from three protocols and data from the application layer are encapsulated in the Record Protocol messages.*

**Topics discussed in this section:**

**17.3.1 ChangeCipherSpec Protocol**

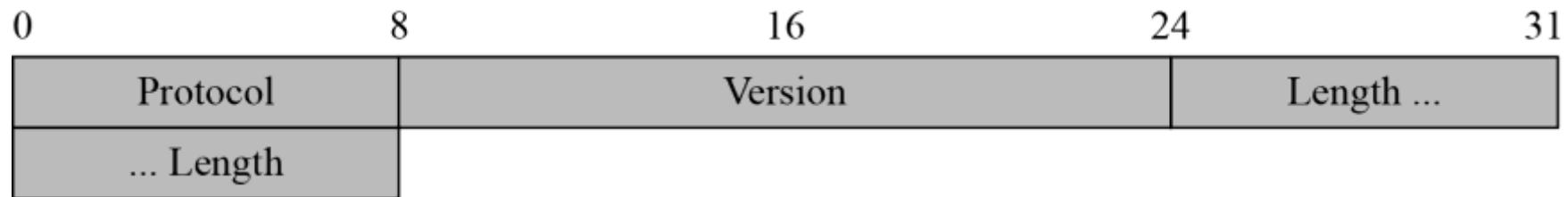
**17.3.2 Alert Protocol**

**17.3.3 Handshake Protocol**

**17.3.4 Application Data**

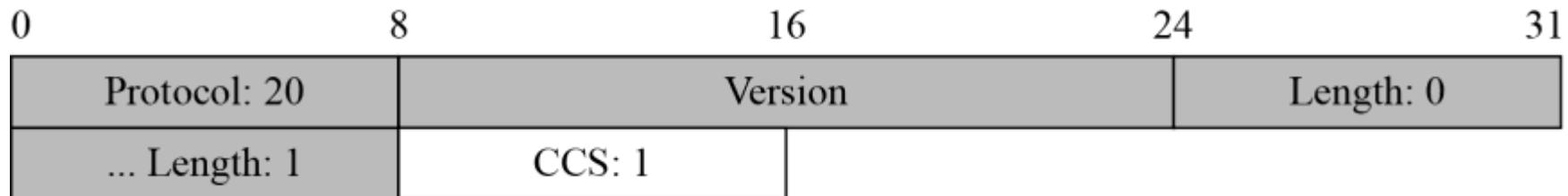
## 17.3 Continued

**Figure 17.23 Record Protocol general header**



## 17.3.1 ChangeCipherSpec Protocol

Figure 17.24 ChangeCipherSpec message



## 17.3.2 Alert Protocol

**Figure 17.25** Alert message

0	8	16	24	31
Protocol: 21		Version		Length: 0
... Length: 2	Level		Description	

### 17.3.3 Handshake Protocol

**Figure 17.26 Generic header for Handshake Protocol**

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

### 17.3.3 Continued

**Table 17.5** *Types of Handshake messages*

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

### 17.3.3 Continued

Figure 17.27 *Virtual tributary types*

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

### 17.3.3 Continued

Figure 17.28 ClientHello 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)			

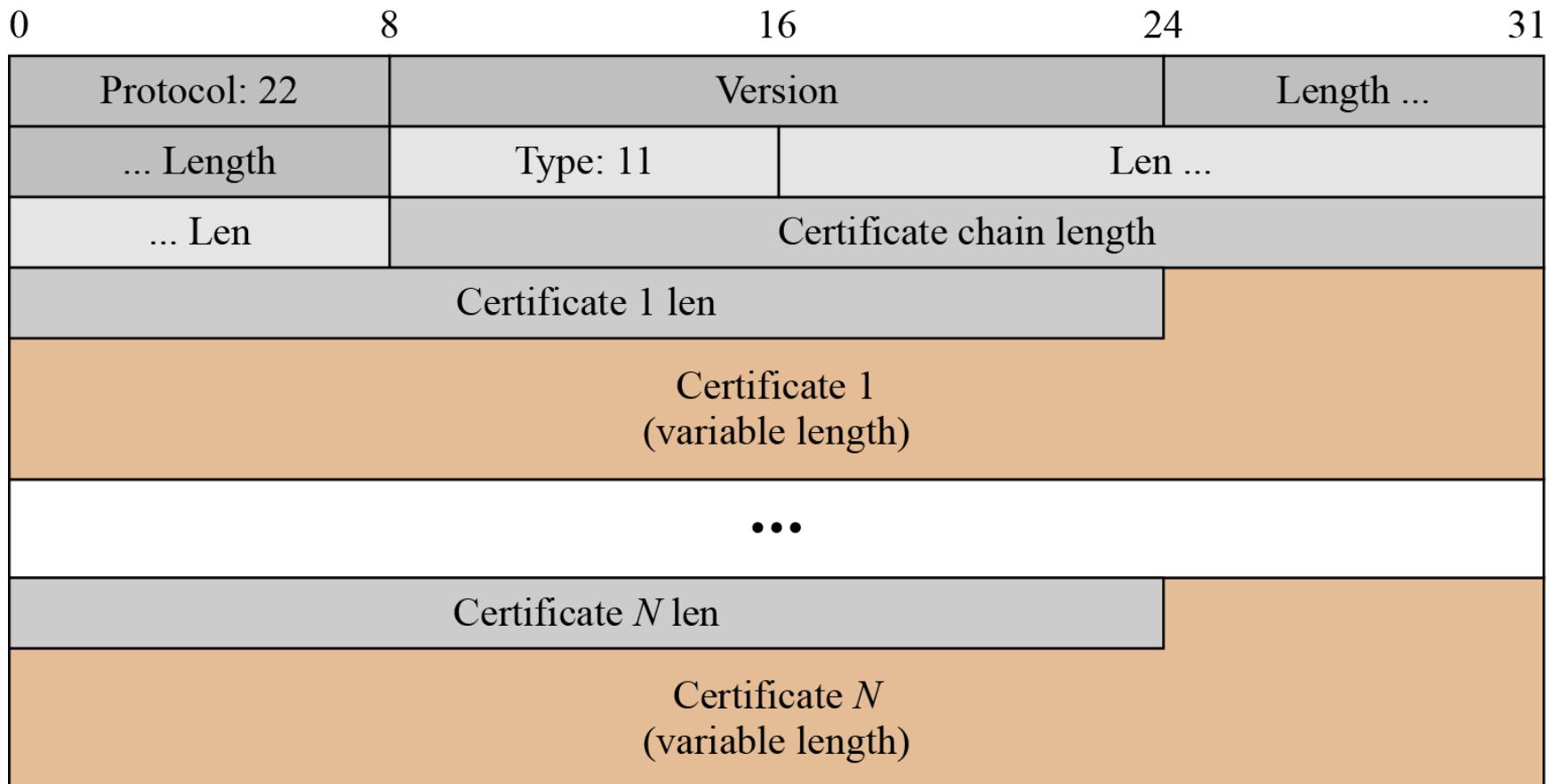
### 17.3.3 Continued

Figure 17.29 *ServerHello message*

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

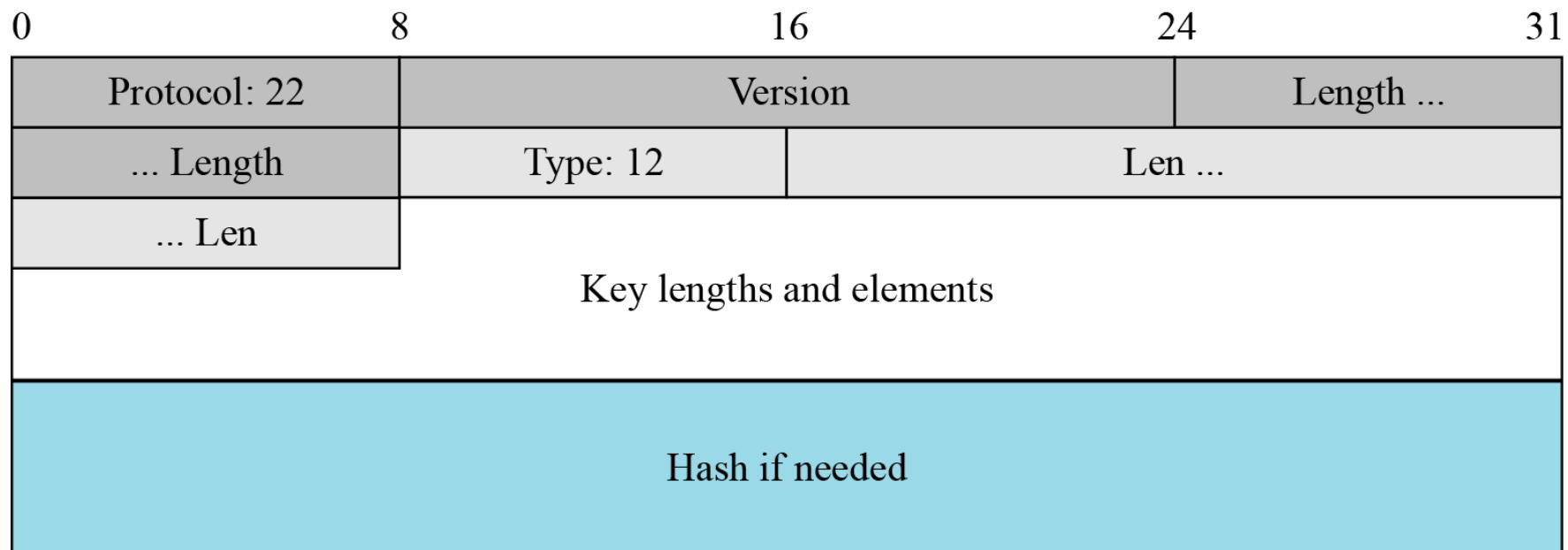
### 17.3.3 Continued

Figure 17.30 Certificate message



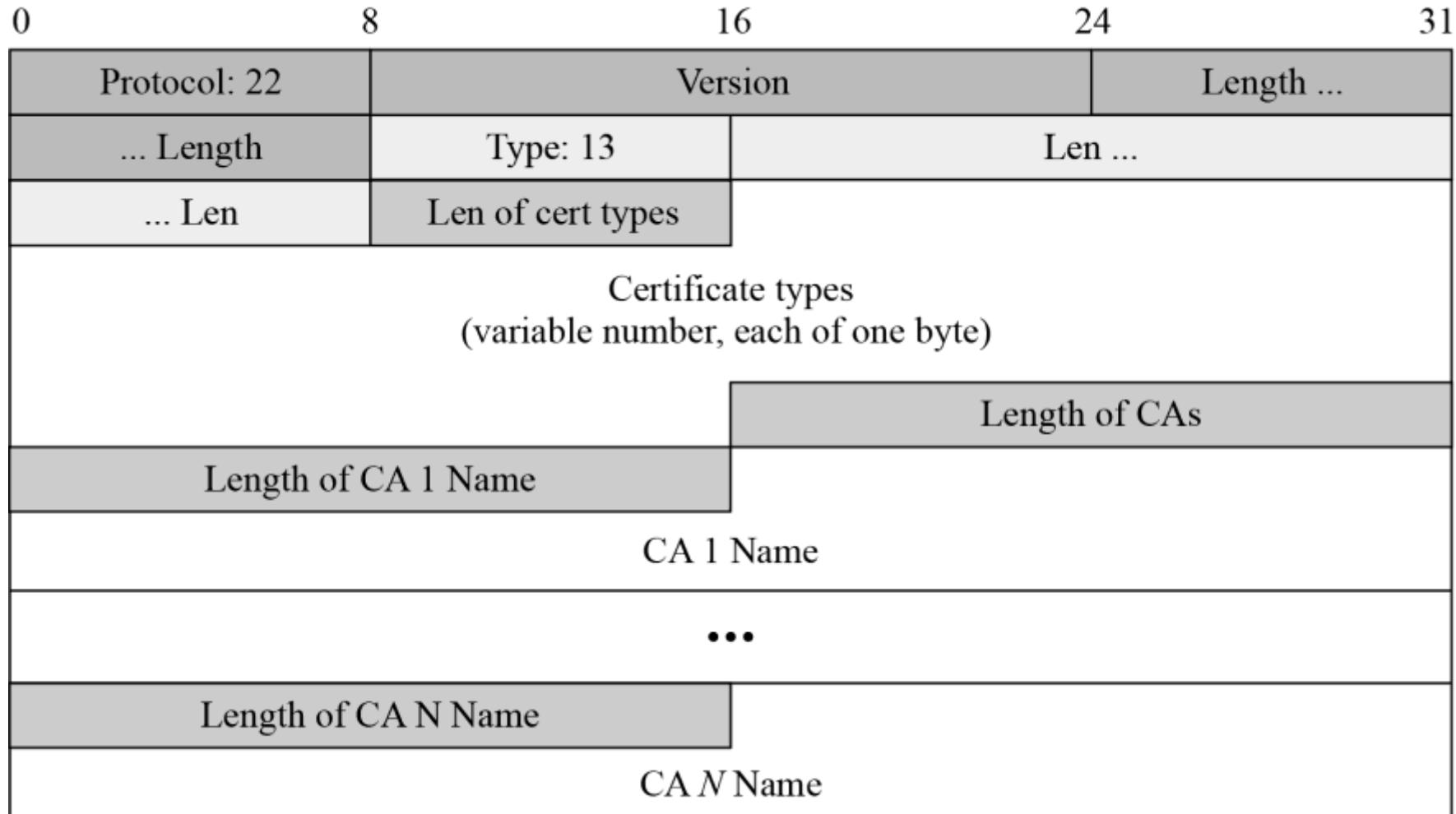
### 17.3.3 Continued

Figure 17.31 *ServerKeyExchange message*



## 17.3.3 Continued

Figure 17.32 CertificateRequest message



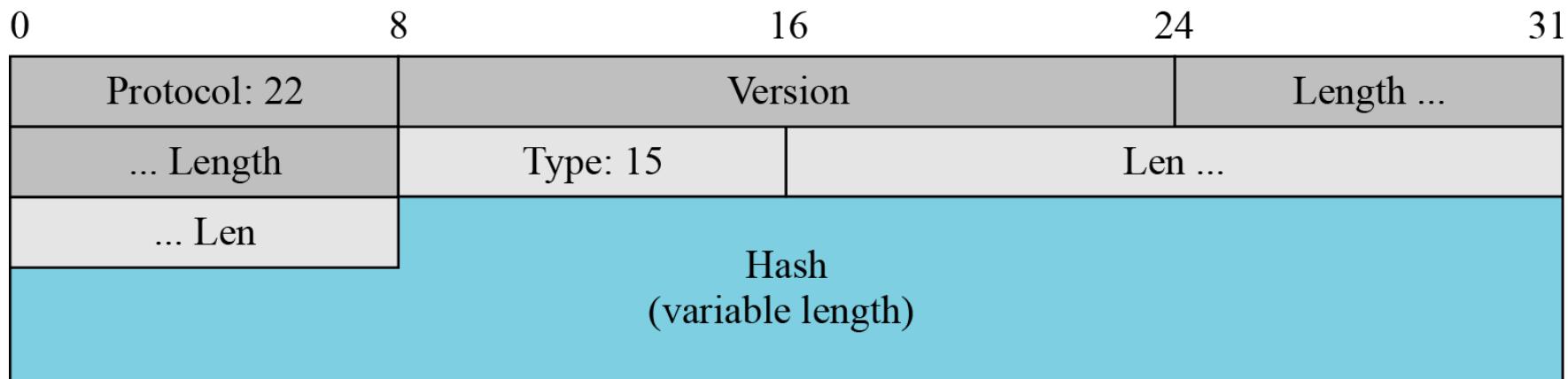
### 17.3.3 Continued

Figure 17.33 *ServerHelloDone message*

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

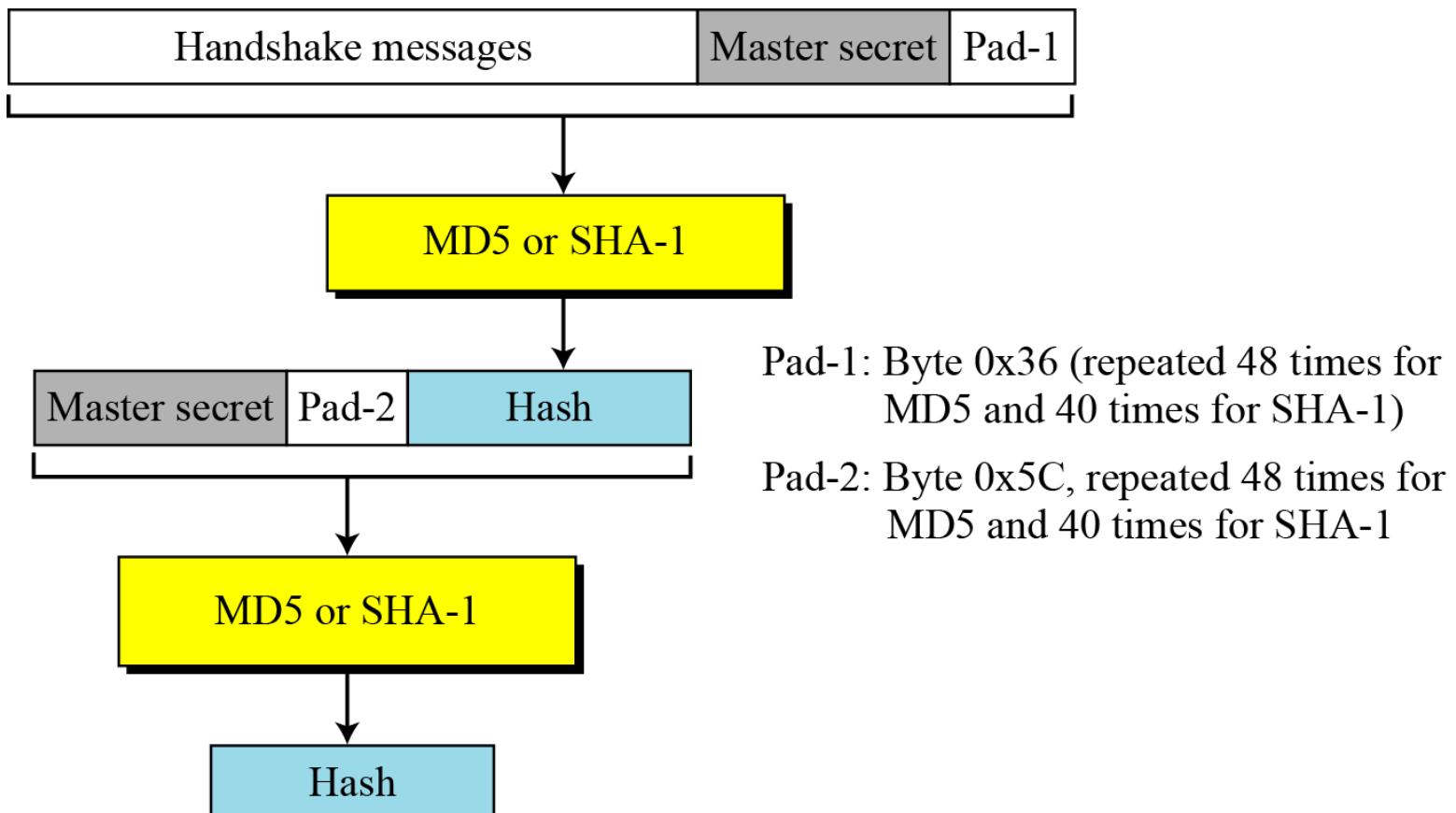
### 17.3.3 Continued

Figure 17.34 CertificateVerify message



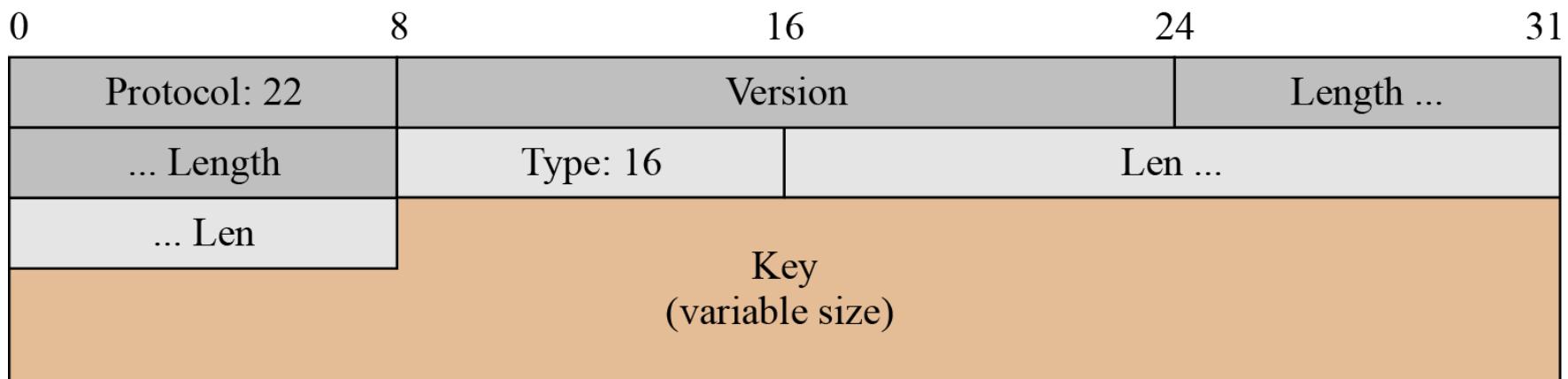
### 17.3.3 Continued

Figure 17.35 Hash calculation for CertificateVerify message



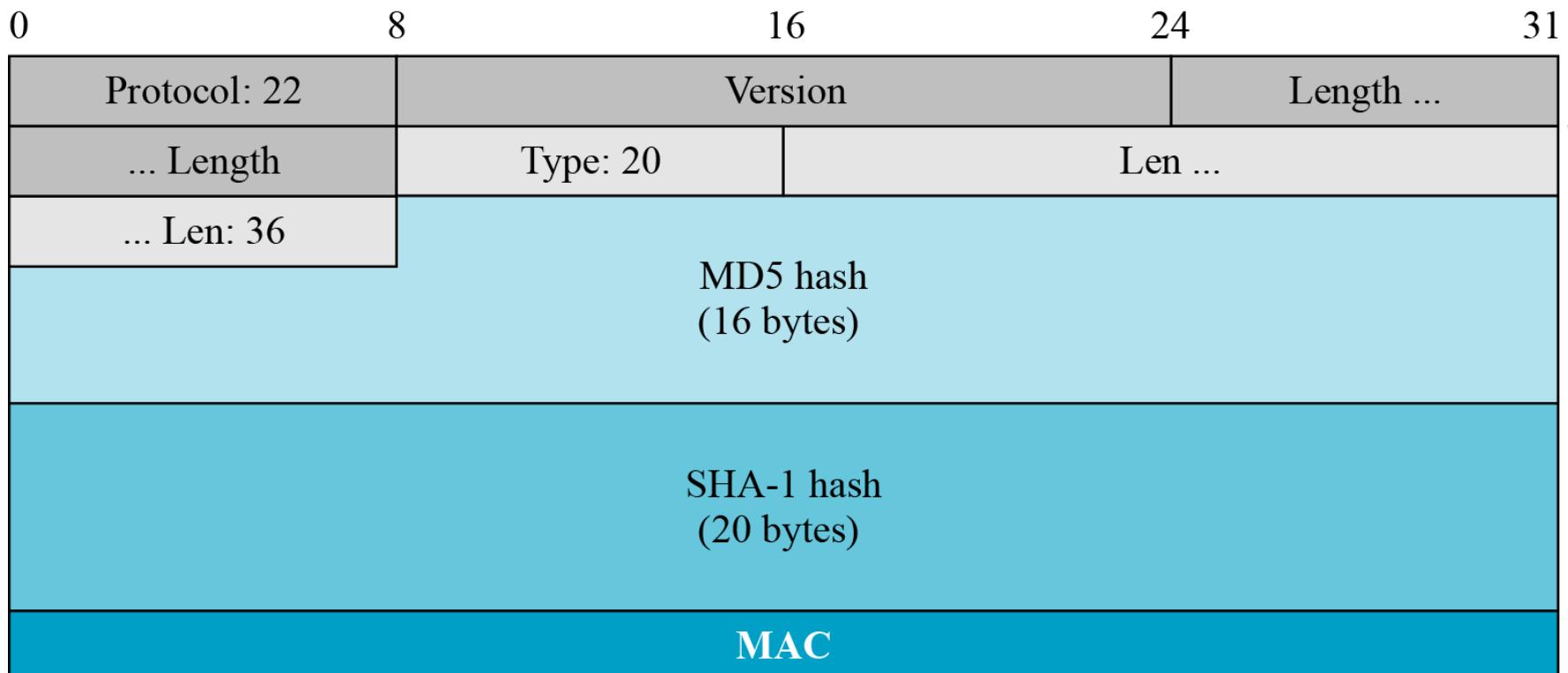
### 17.3.3 Continued

Figure 17.36 *ClientKeyExchange message*



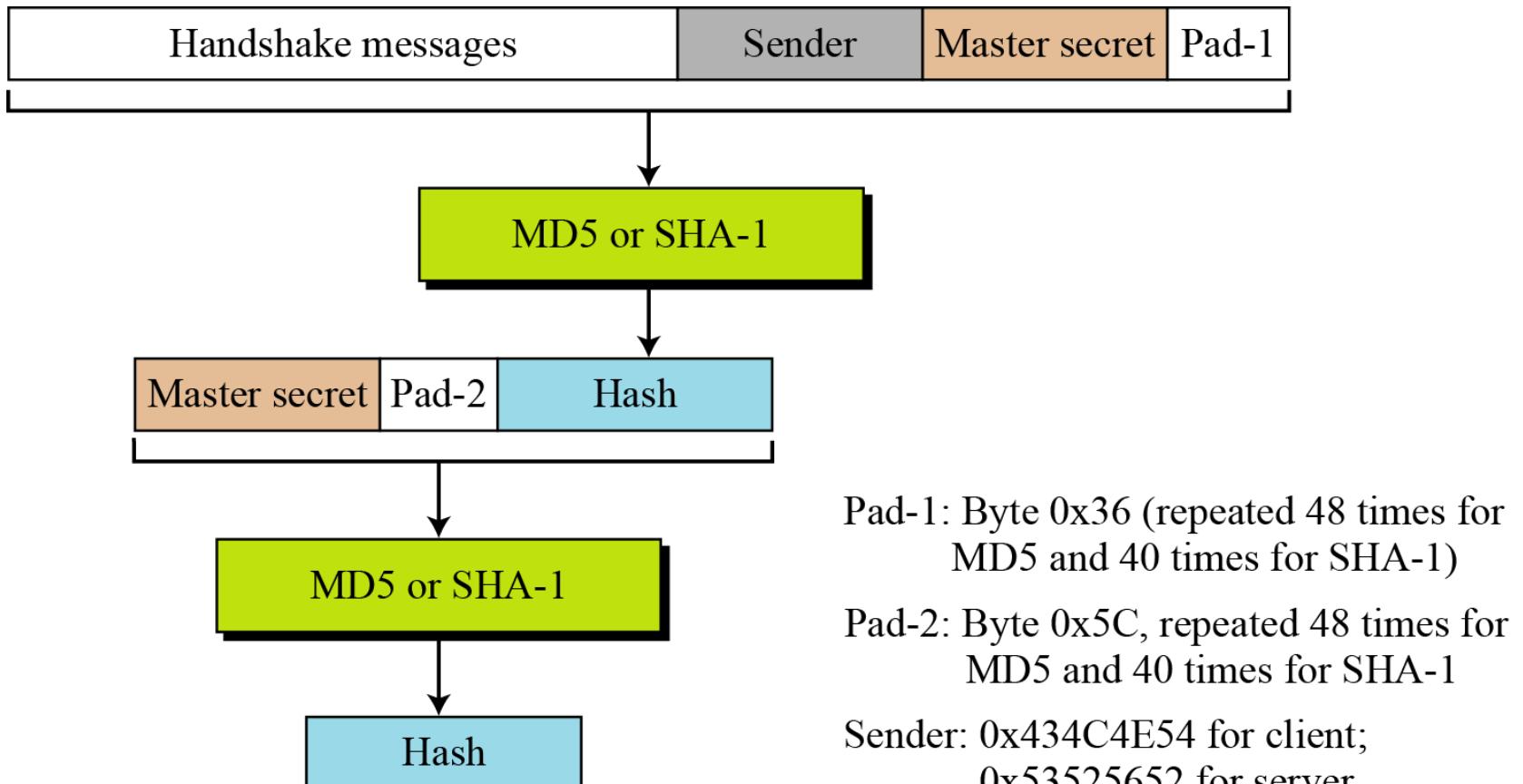
### 17.3.3 Continued

Figure 17.37 Finished message



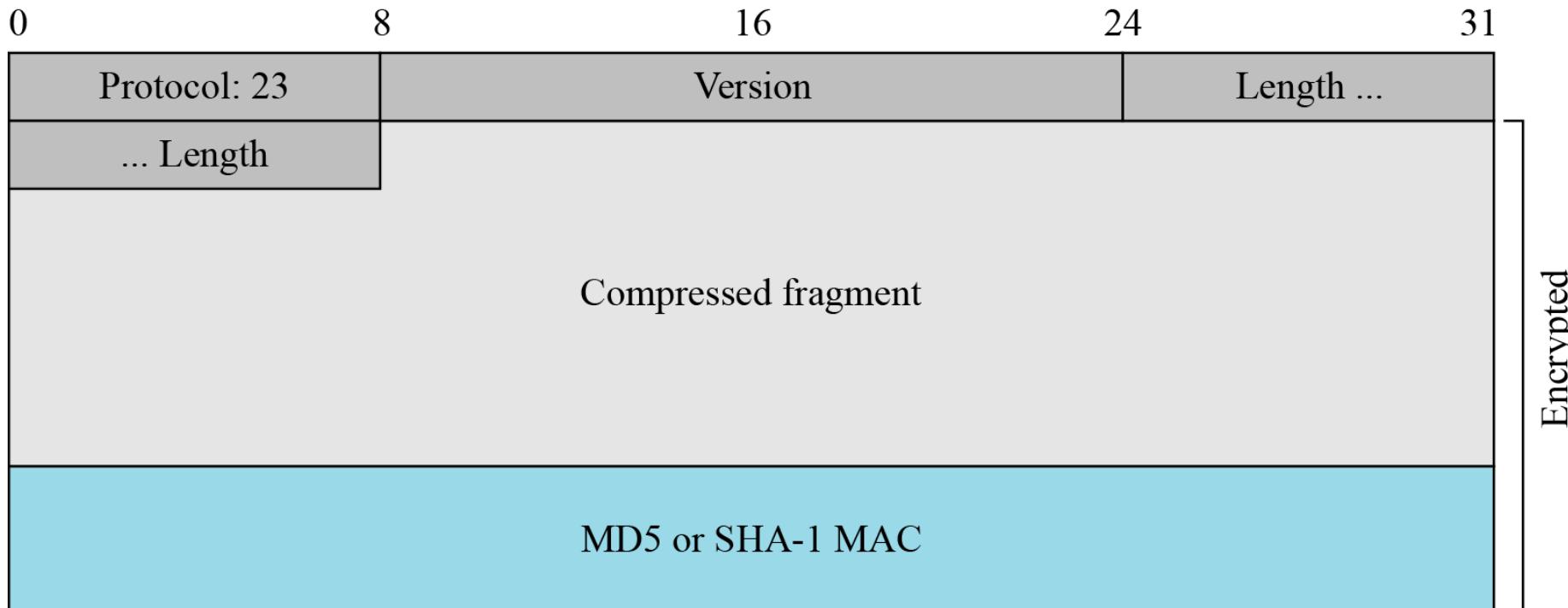
## 17.3.3 Continued

Figure 17.38 Hash calculation for Finished message



### 17.3.3 Application Data

**Figure 17.39 Record Protocol message for application data**



# 17-4 Transport Layer Security (TLS)

*The Transport Layer Security (TLS) protocol is the IETF standard version of the SSL protocol. The two are very similar, with slight differences.*

## Topics discussed in this section:

**17.4.1 Version**

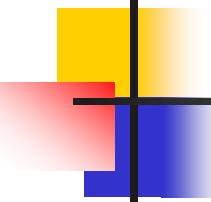
**17.4.2 Cipher Suite**

**17.4.3 Generation of Cryptographic Secrets**

**17.4.4 Alert Protocol**

**17.4.5 Handshake Protocol**

**17.4.6 Record Protocol**



## **17.4.1 Version**

*The first difference is the version number (major and minor). The current version of SSL is 3.0; the current version of TLS is 1.0. In other words, SSLv3.0 is compatible with TLSv1.0.*

## *17.4.2 Cipher Suite*

*Another minor difference between SSL and TLS is the lack of support for the Fortezza method. TLS does not support Fortezza for key exchange or for encryption/decryption. Table 17.6 shows the cipher suite list for TLS (without export entries).*

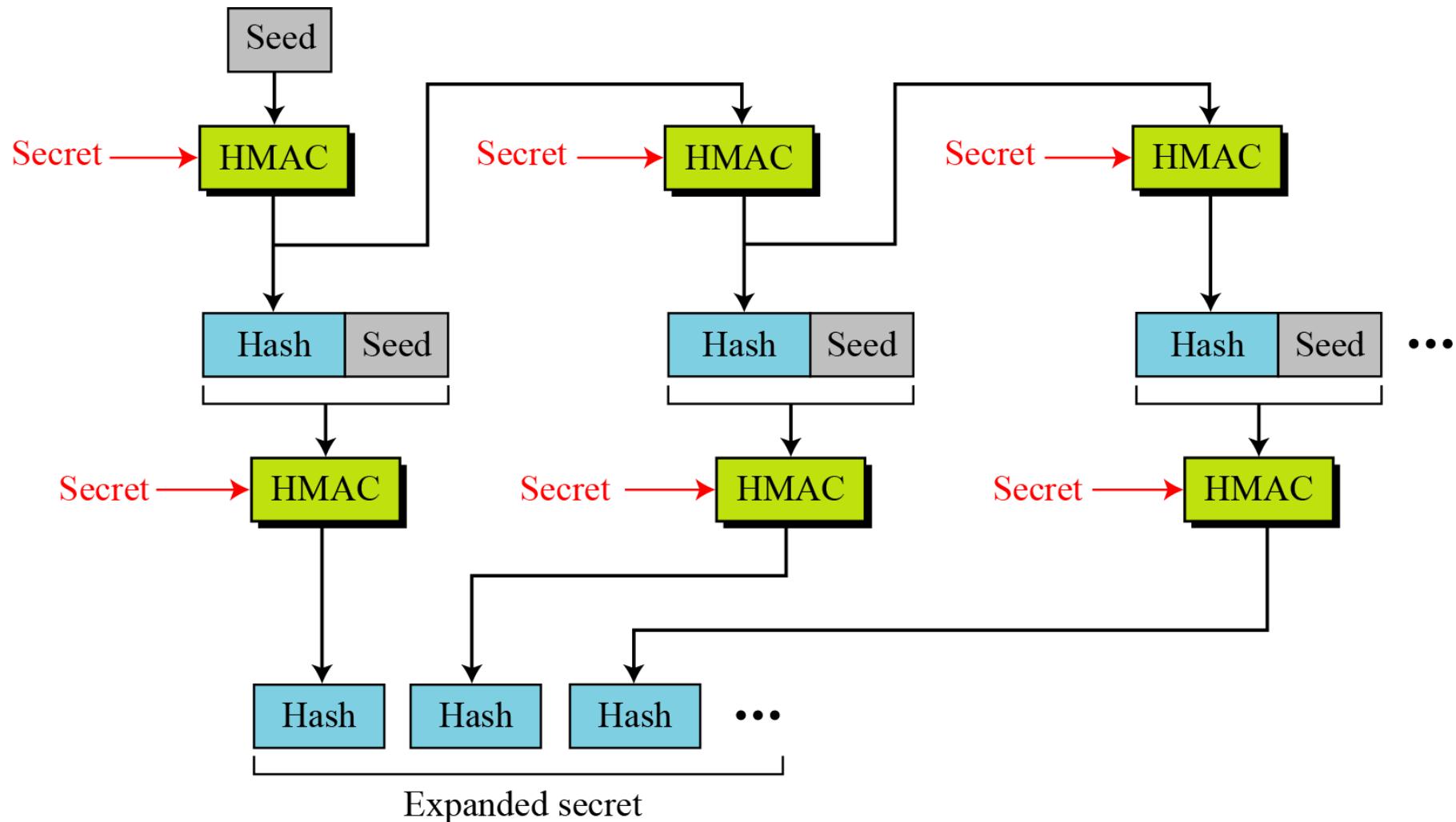
## 17.4.2 Continued

**Table 17.6 Cipher Suite for TLS**

<i>Cipher suite</i>	<i>Key Exchange</i>	<i>Encryption</i>	<i>Hash</i>
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_WITH DES_CBC_SHA	RSA	DES	SHA-1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
TLS_DH_anon_WITH DES_CBC_SHA	DH_anon	DES	SHA-1
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
TLS_DHE_RSA_WITH DES_CBC_SHA	DHE_RSA	DES	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
TLS_DHE_DSS_WITH DES_CBC_SHA	DHE_DSS	DES	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
TLS_DH_RSA_WITH DES_CBC_SHA	DH_RSA	DES	SHA-1
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
TLS_DH_DSS_WITH DES_CBC_SHA	DH_DSS	DES	SHA-1
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1

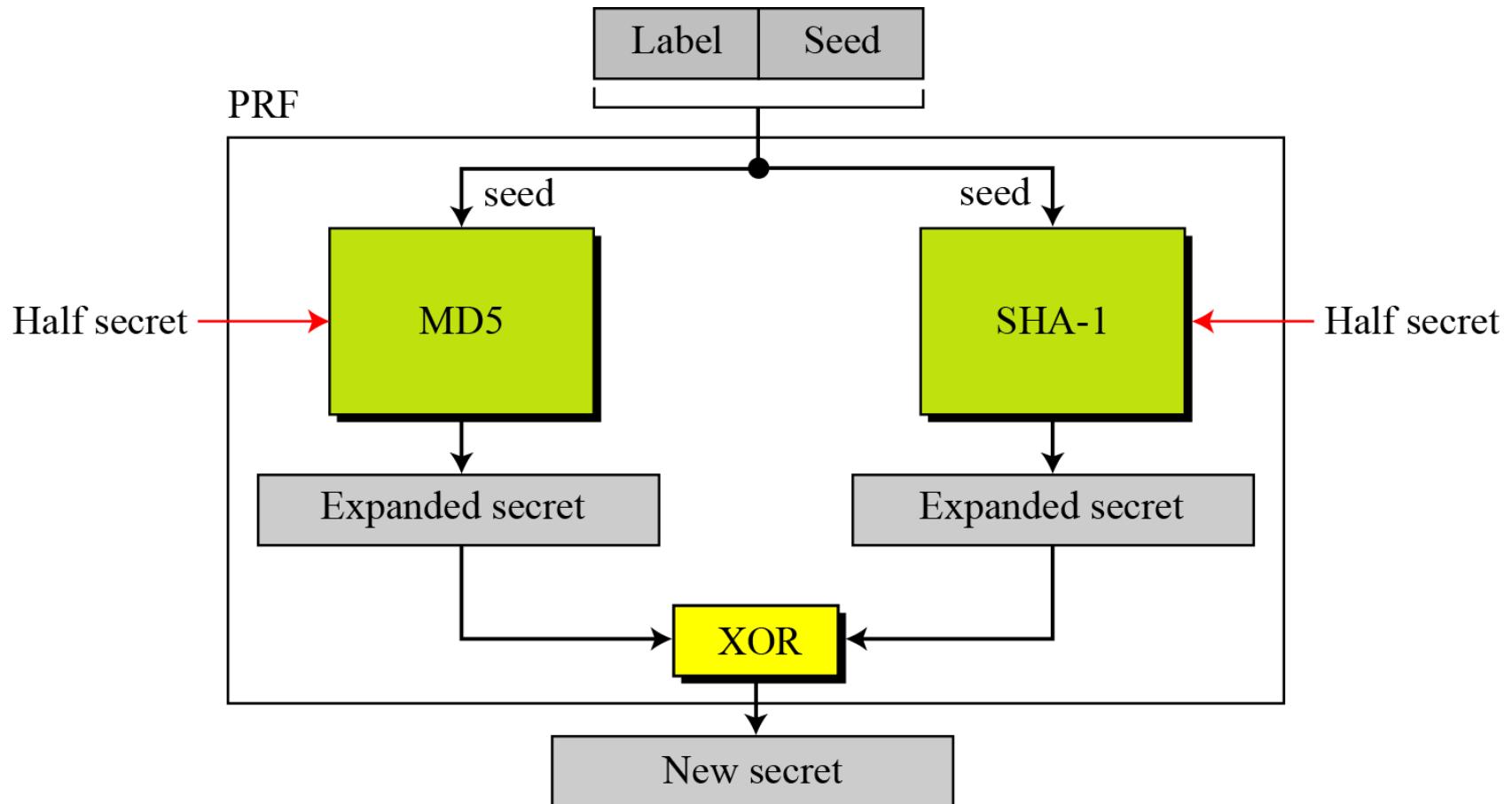
### ***17.4.3 Generation of Cryptographic Secrets***

## Figure 17.40 Data-expansion function



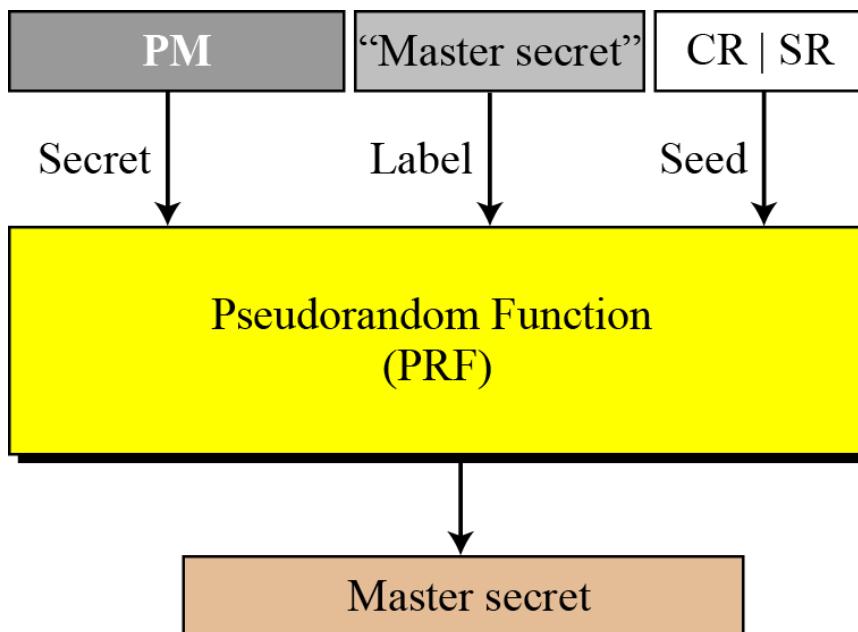
## 17.4.3 Continued

Figure 17.41 PRF



## 17.4.3 Continued

Figure 17.42 Master secret generation



PM: Pre-master Secret

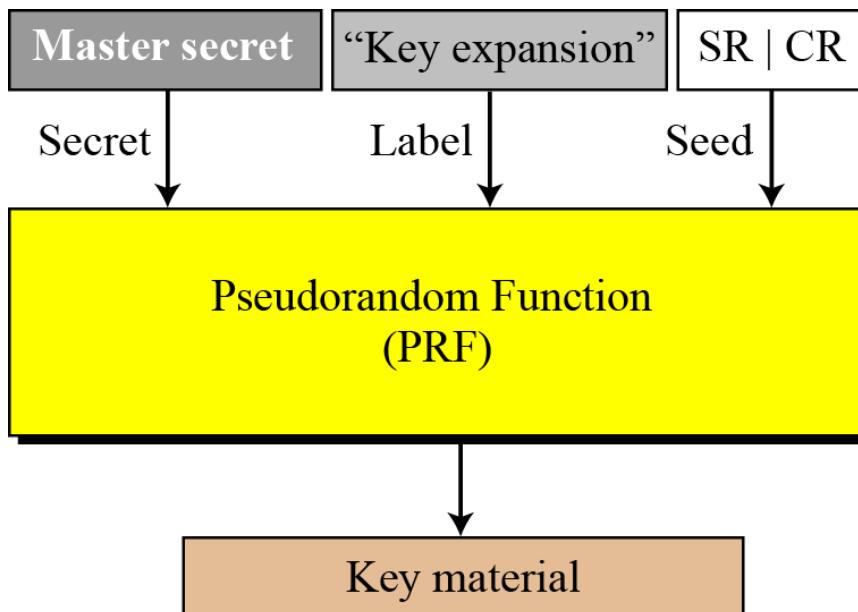
CR: Client Random Number

SR: Server Random Number

|: Concatenation

## 17.4.3 Continued

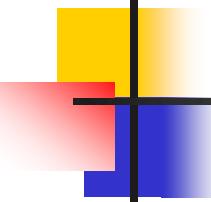
Figure 17.43 Key material generation



CR: Client Random Number

SR: Server Random Number

|: Concatenation



## **17.4.4 Alert Protocol**

*TLS supports all of the alerts defined in SSL except for NoCertificate. TLS also adds some new ones to the list. Table 17.7 shows the full list of alerts supported by TLS.*

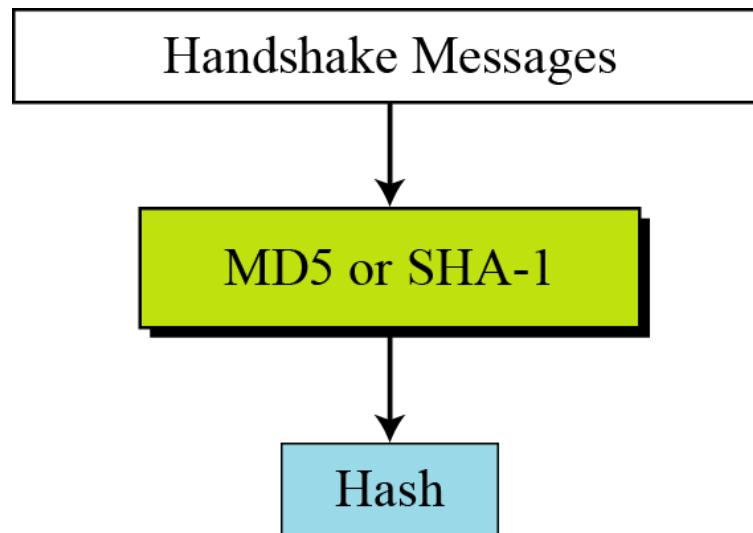
## 17.4.4 Continued

**Table 17.7 Alerts defined for TLS**

<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.
21	<i>DecryptionFailed</i>	Decrypted message is invalid.
22	<i>RecordOverflow</i>	Message size is more than $2^{14} + 2048$ .
30	<i>DecompressionFailure</i>	Unable to decompress appropriately.
40	<i>HandshakeFailure</i>	Sender unable to finalize the handshake.
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 has expired.
46	<i>CertificateUnknown</i>	Certificate unknown.
47	<i>IllegalParameter</i>	A field out of range or inconsistent with others.
48	<i>UnknownCA</i>	CA could not be identified.

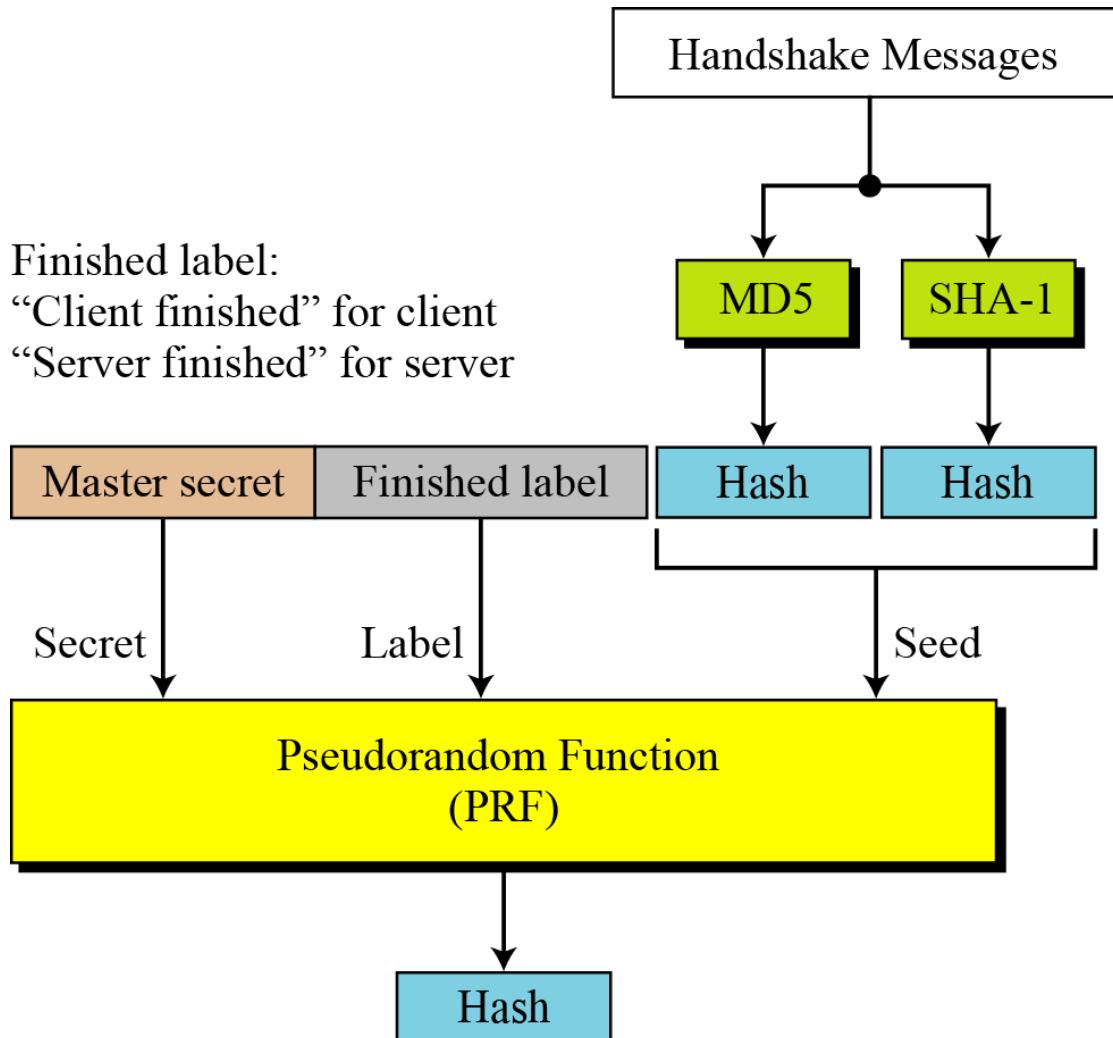
## 17.4.5 Handshake Protocol

**Figure 17.44 Hash for CertificateVerify message in TLS**



## 17.4.5 Continued

Figure 17.45 Hash for Finished message in TLS



## 17.4.6 Record Protocol

Figure 17.46 HMAC for TLS

