Transport-level and Web Security (SSL / TLS)

Web Security Considerations

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets

The following characteristics of Web usage suggest the need for tailored security tools:

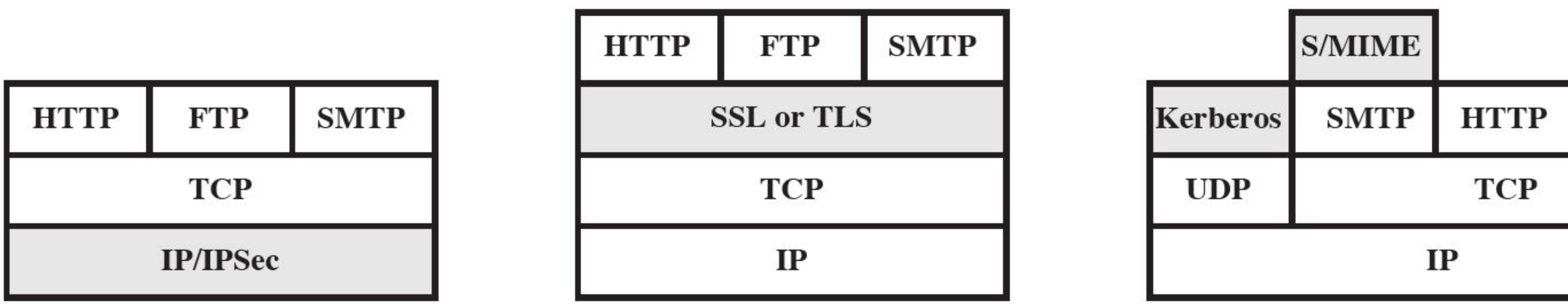
- Web servers are relatively easy to configure and manage
- Web content is increasingly easy to develop
- The underlying software is extraordinarily complex
  - May hide many potential security flaws
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex
- Casual and untrained (in security matters) users are common clients for Web-based services
  - Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures

	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Table 17.1 A Comparison of Threats on the Web

Where to provide security?

Long-lasting discussion, no ultimate answer



(a) Network Level (b) Transport Level (c) Application Level

have seen

this lecture

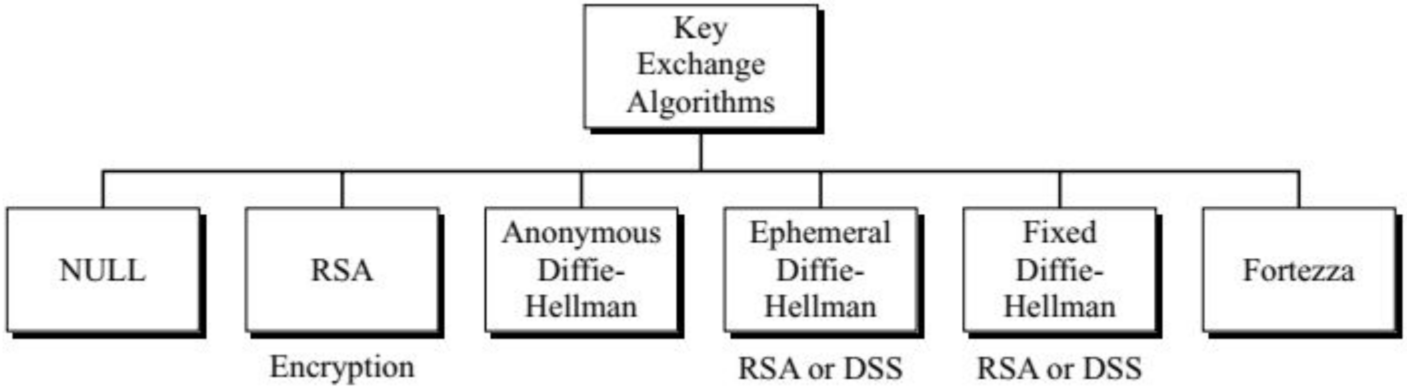
have seen and will see

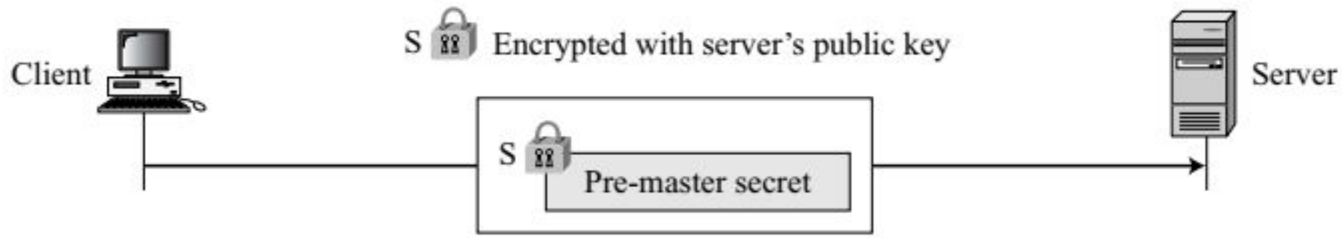
SSL Services

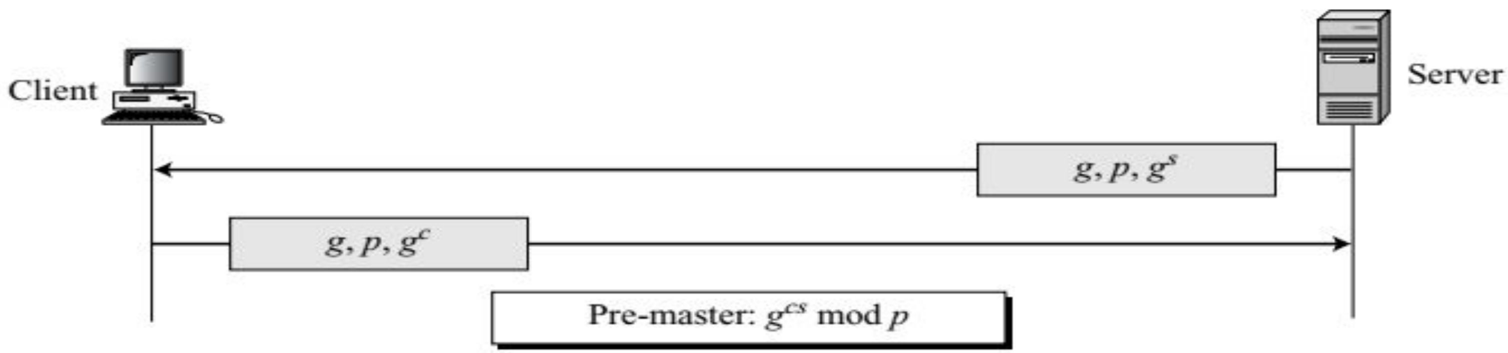
Fragmentation
Compression
Message Integrity
Confidentiality
Framing

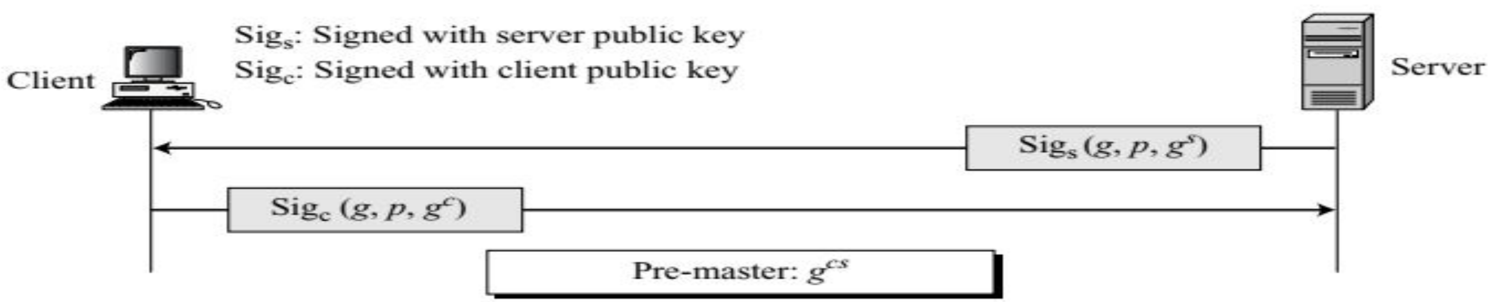
Key Exchange Algorithms

- The client and the server each need six cryptographic secrets (four keys and two initialization vectors).
- However, to create these secrets, one pre-master secret must be established between the two parties.
- SSL defines six key-exchange methods to establish this premaster secret: NULL, RSA, anonymous Diffie-Hellman, ephemeral Diffie-Hellman, fixed Diffie-Hellman, and Fortezza,

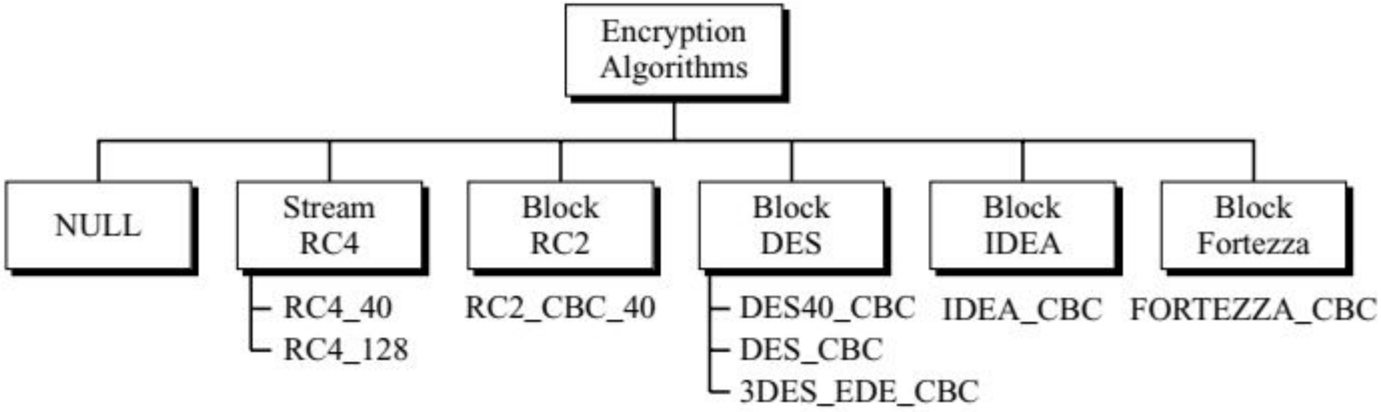



Client
S Encrypted with server's public key
Server
S
Pre-master secret

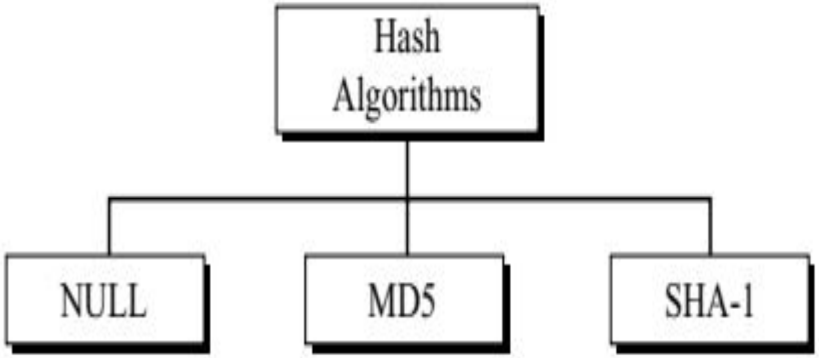

Client
Server
g, p, g^s
g, p, g^c
Pre-master: g^{cs} mod p


Client
Sig_s: Signed with server public key
Sig_c: Signed with client public key
Server
$Sig_s (g, p, g^s)$
$Sig_c (g, p, g^c)$
Pre-master: g^{cs}

Encryption/Decryption Algorithms



Hash Algorithms



Cipher Suite

- Each suite starts with the term “SSL” followed by the key exchange algorithm.
- The word “WITH” separates the key exchange algorithm from the encryption and hash algorithms.
  - Ex: SSL_DHE_RSA_WITH_DES_CBC_SHA
- Defines DHE_RSA (ephemeral Diffie-Hellman with RSA digital signature) as the key exchange with DES_CBC as the encryption algorithm and SHA as the hash algorithm.

SSL cipher suite list

Cipher suite	*Key Exchange*	*Encryption*	*Hash*
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

Cryptographic Parameter Generation

- To achieve message integrity and confidentiality, SSL needs six cryptographic secrets, four keys and two IVs.
- The client needs one key for message authentication (HMAC), one key for encryption, and one IV for block encryption.
- The server needs the same.
- 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.

1. The client and server exchange two random numbers; one is created by the client and the other by the server.
2. The client and server exchange one pre-master secret using one of the key-exchange algorithms we discussed previously.
3. A 48-byte master secret is created from the pre-master secret by applying two hash functions (SHA-1 and MD5),
4. The master secret is used to create variable-length key material by applying the same set of hash functions and prepending with different constants
5. Six different keys are extracted from the key material

Figure 17.8 *Calculation of master secret from pre-master secret*

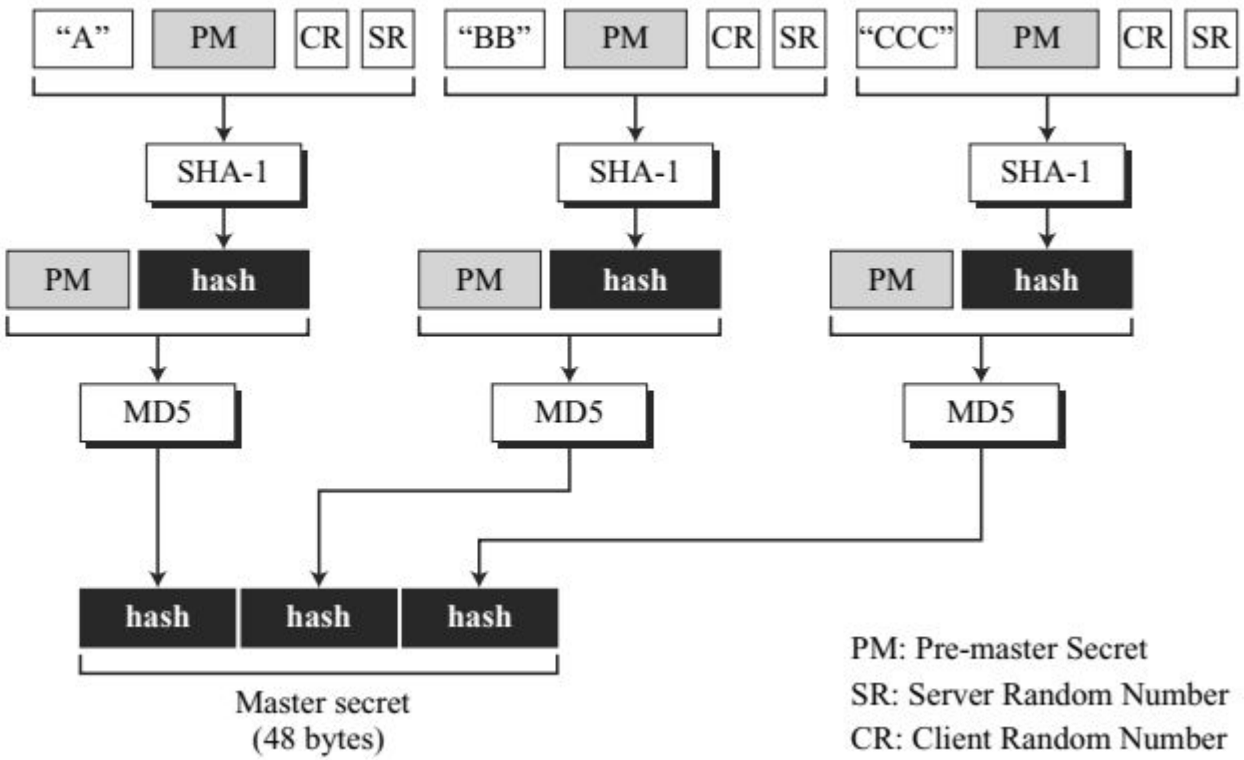


Figure 17.9 *Calculation of key material from master secret*

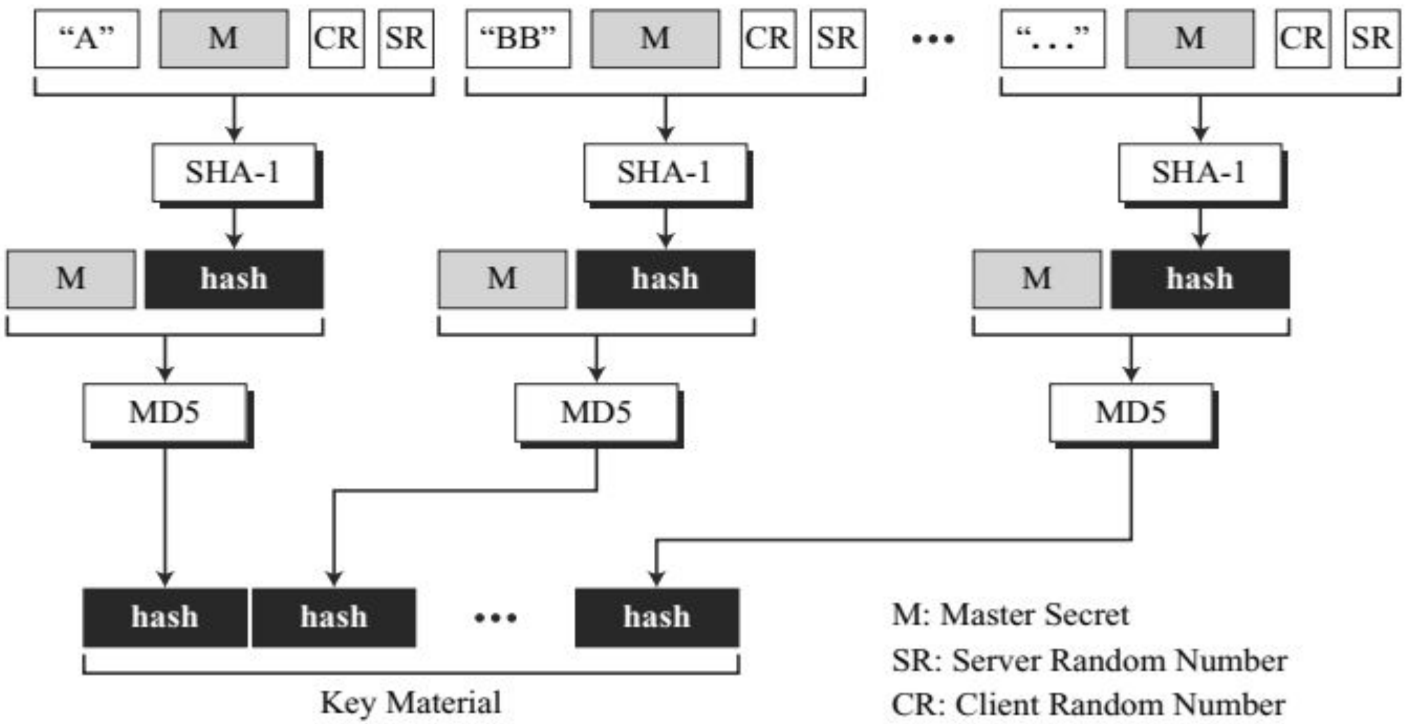
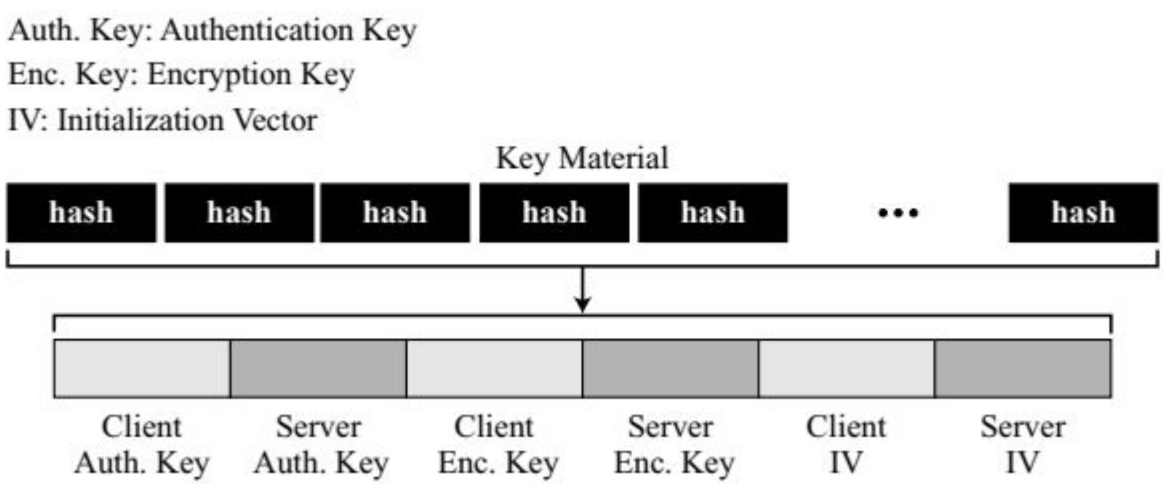


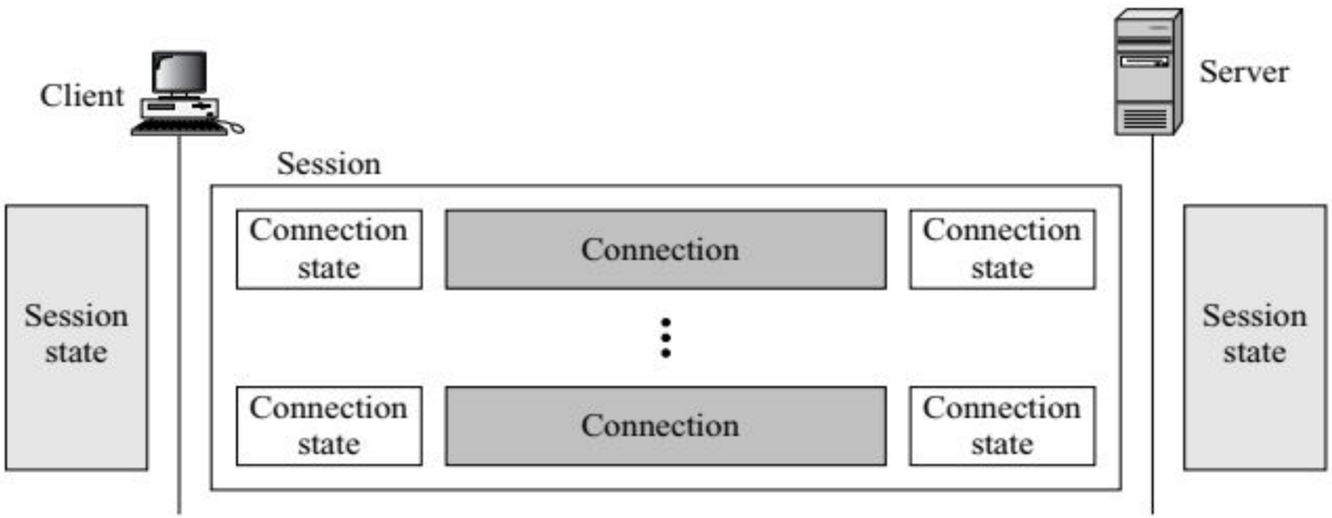
Figure 17.10 *Extractions of cryptographic secrets from key material*



Sessions and Connections

- SSL differentiates a connection from a session.
- A session is an association between a client and a server.
- After a session is established, the two parties have common information such as the session identifier, the certificate authenticating each of them (if necessary), the compression method (if needed), the cipher suite, and a master secret that is used to create keys for message authentication encryption.
- A session can consist of many connections.
- A connection between two parties can be terminated and reestablished 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.
- The separation of a session from a connection prevents the high cost of creating a master secret. By allowing a session to be suspended and resumed, the process of the master secret calculation can be eliminated.

Figure 17.11 *A session and connections*



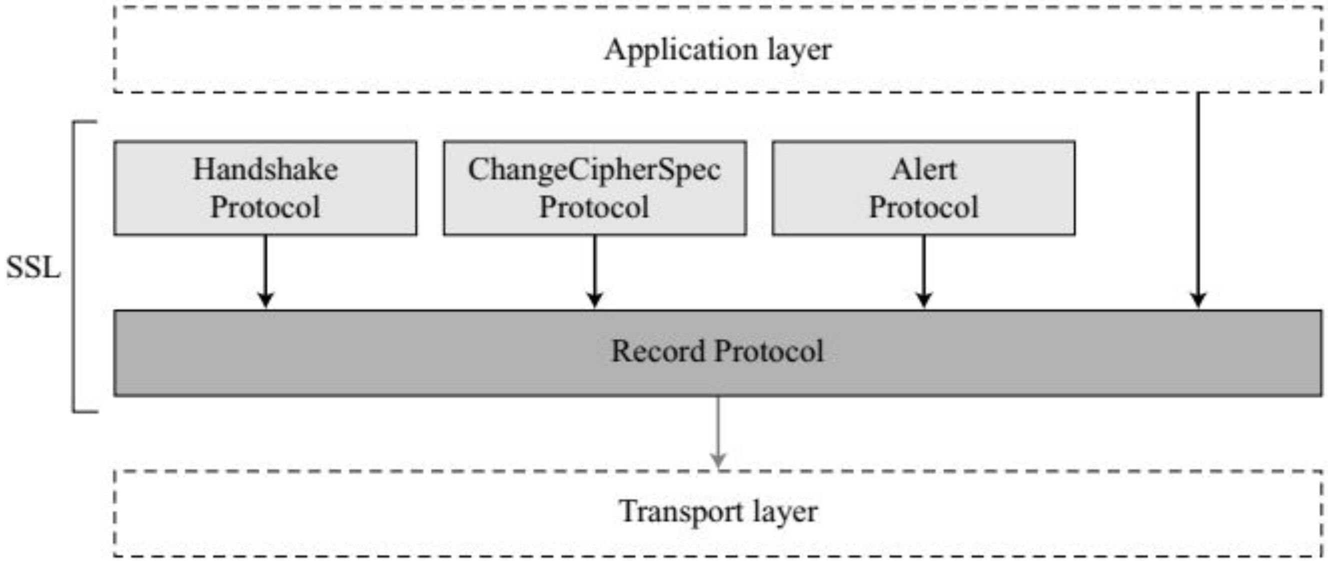
Session State

Parameter	*Description*
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.

Connection State

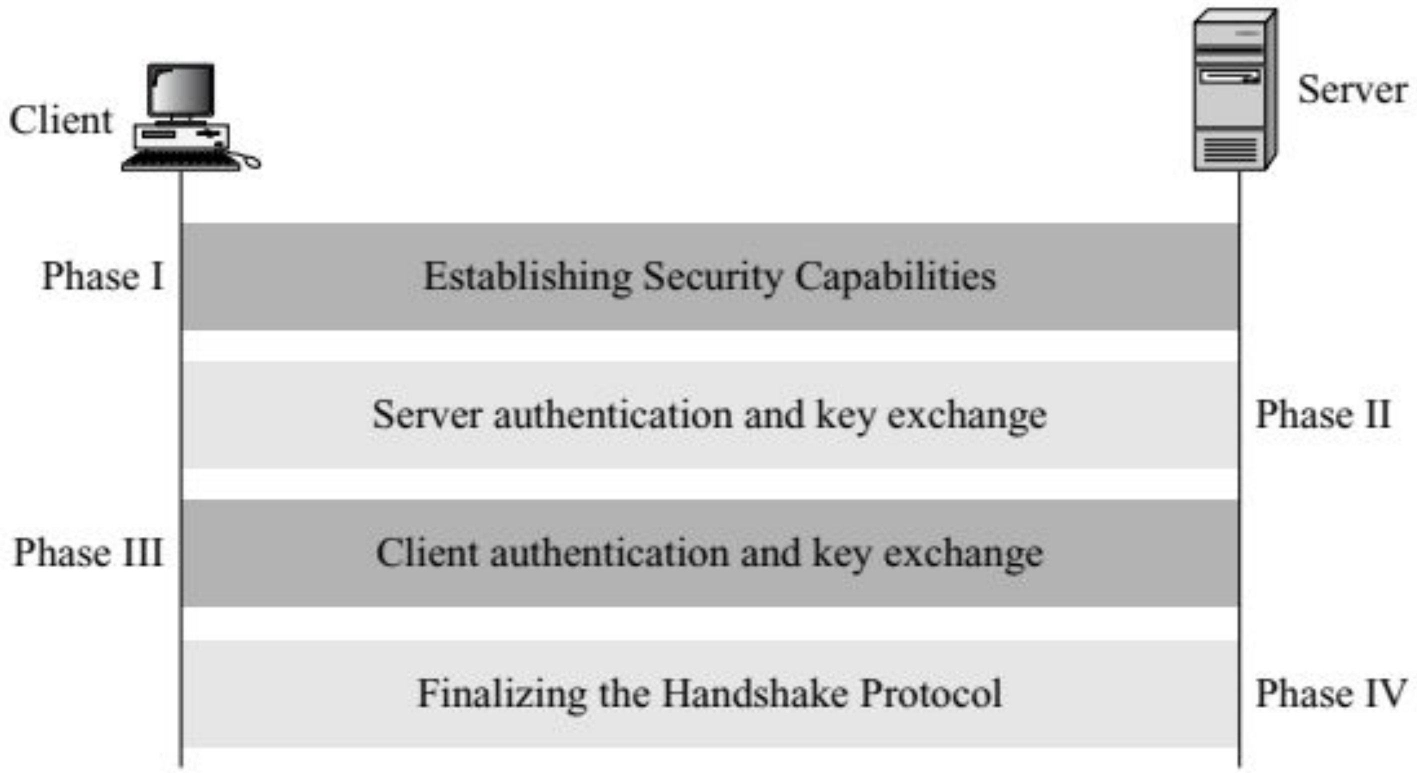
Parameter	*Description*
Server and client random numbers	A sequence of bytes chosen by the server and client for each connection.
Server write MAC secret	The outbound server MAC key for message integrity. The server uses it to sign; the client uses it to verify.
Client write MAC secret	The outbound client 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 $2^{64} - 1$.

PROTOCOLS



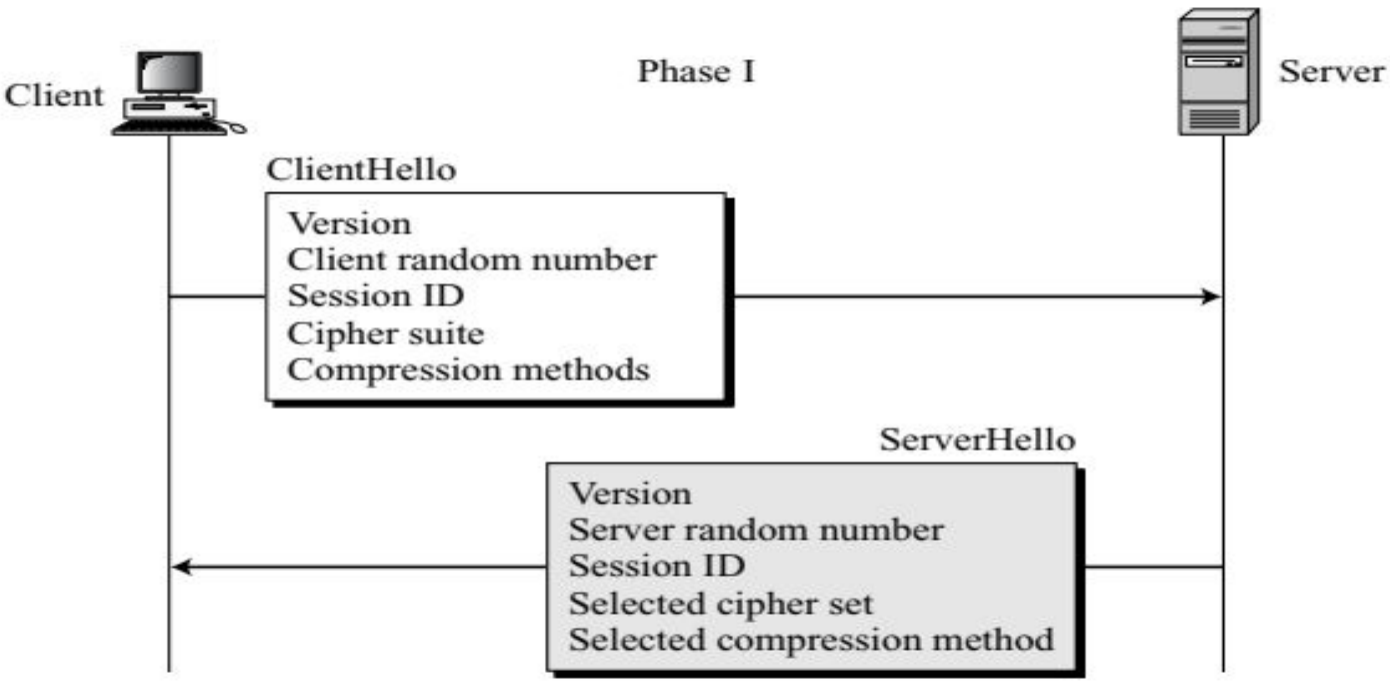
Application layer
Handshake Protocol
ChangeCipherSpec Protocol
Alert Protocol
SSL
Record Protocol
Transport layer

Handshake Protocol



Phase-I: Establishing Security Capability

- In Phase I, the client and the server announce their security capabilities and choose those that are convenient for both.
- In this phase, a session ID is established and the cipher suite is chosen. The parties agree upon a particular compression method.
- Finally, two random numbers are selected, one by the client and one by the server, to be used for creating a master secret as we saw before.
- Two messages are exchanged in this phase: ClientHello and ServerHello messages.

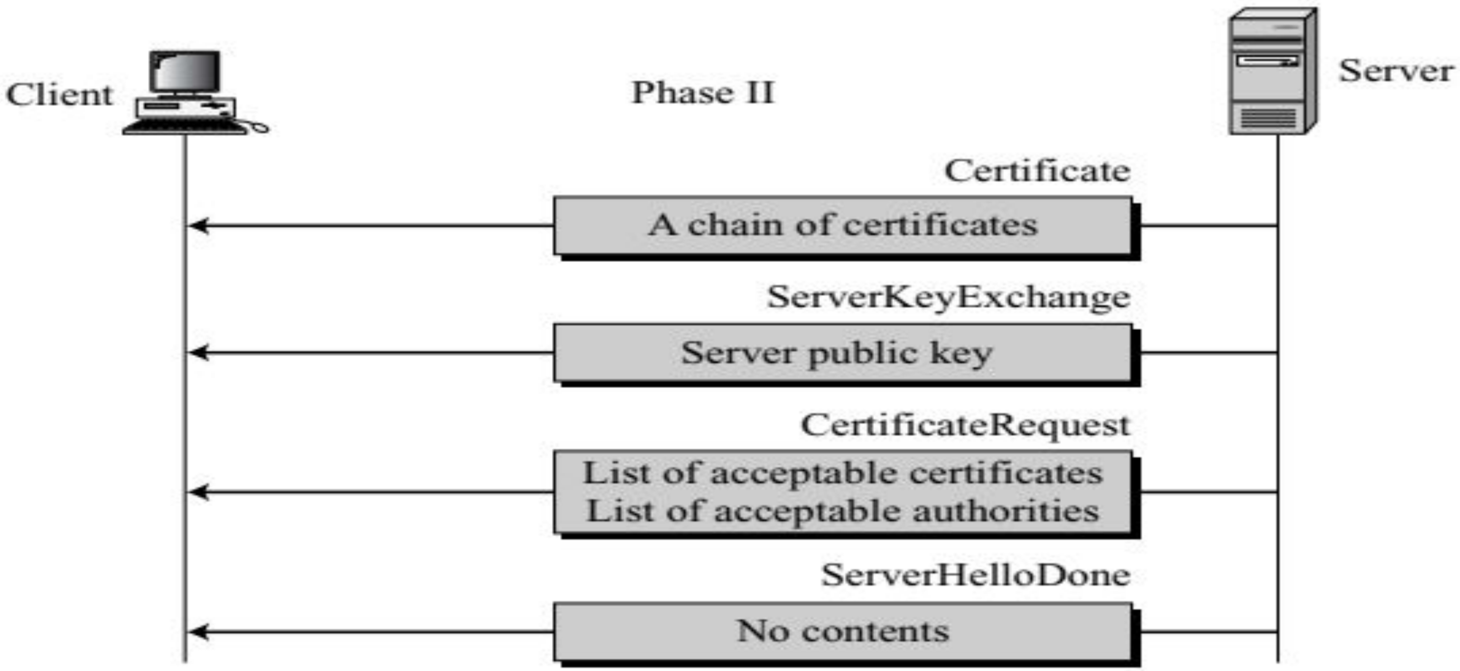

Client
Phase I
Server
ClientHello
Version
Client random number
Session ID
Cipher suite
Compression methods
ServerHello
Version
Server random number
Session ID
Selected cipher set
Selected compression method

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*

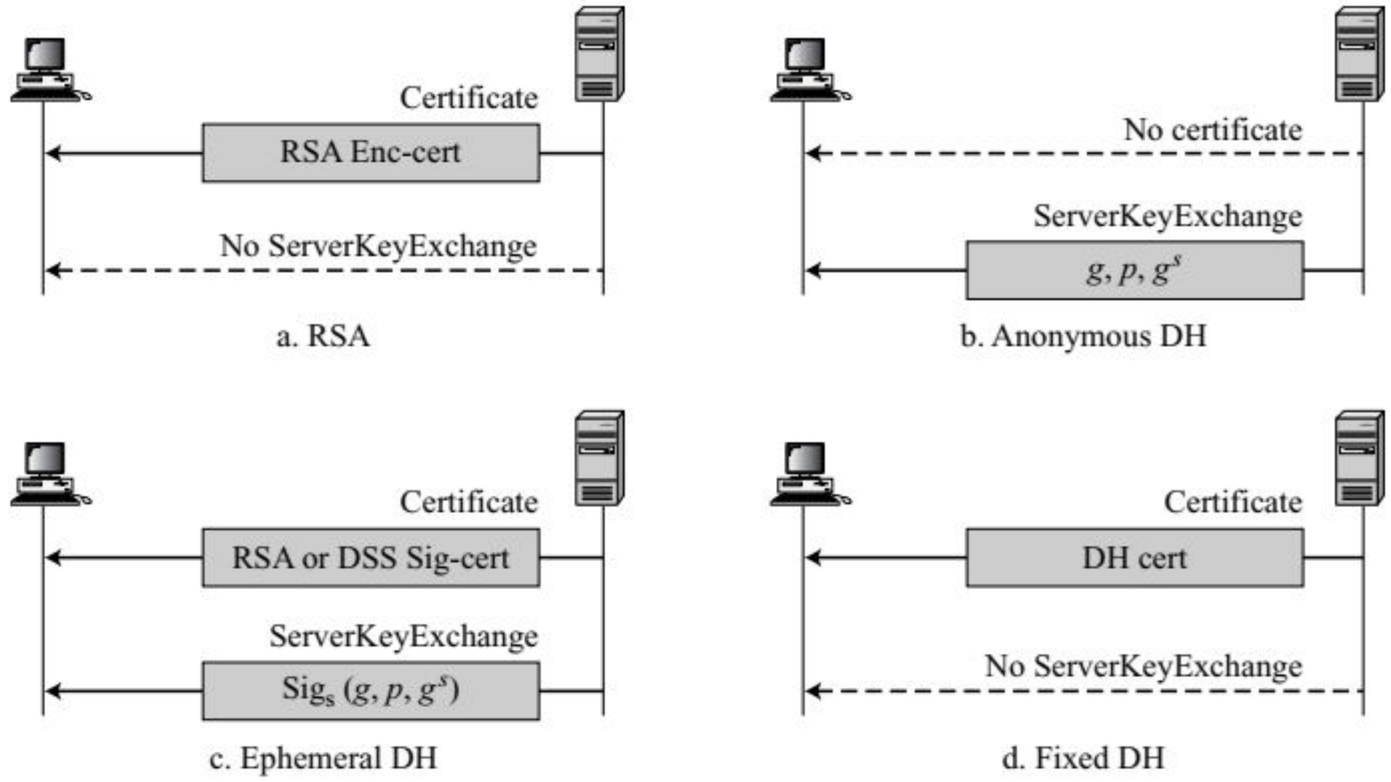
Phase II: Server Key Exchange and Authentication

- In phase II, the server authenticates itself if needed.
- The sender may send its certificate, its public key, and may also request certificates from the client.
- At the end, the server announces that the serverHello process is done.

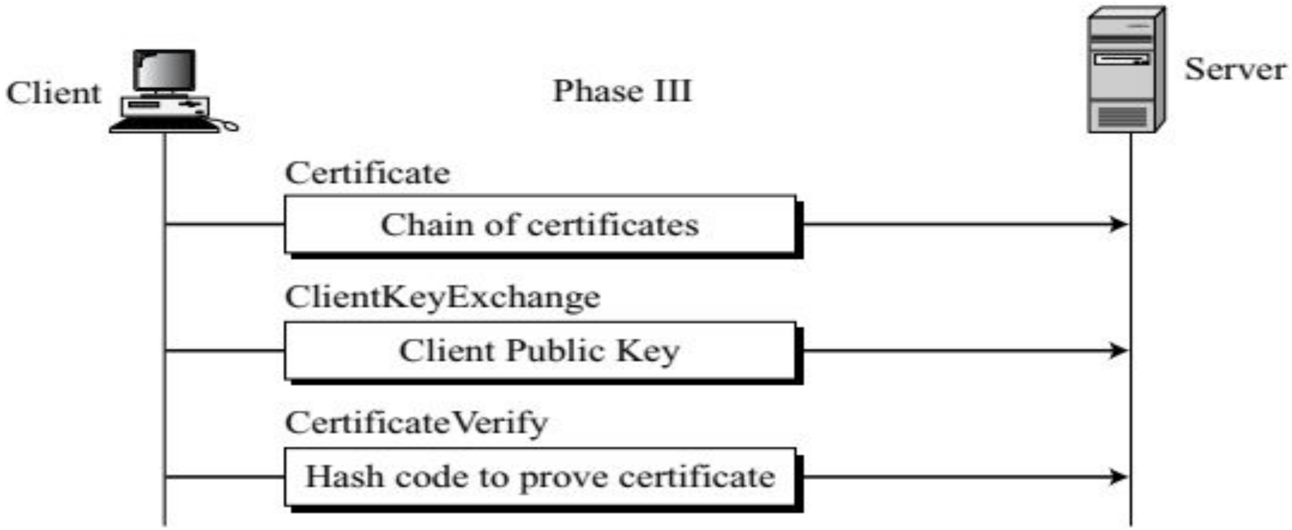

Client
Phase II
Server
Certificate
A chain of certificates
ServerKeyExchange
Server public key
CertificateRequest
List of acceptable certificates
List of acceptable authorities
ServerHelloDone
No contents

After Phase II,

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

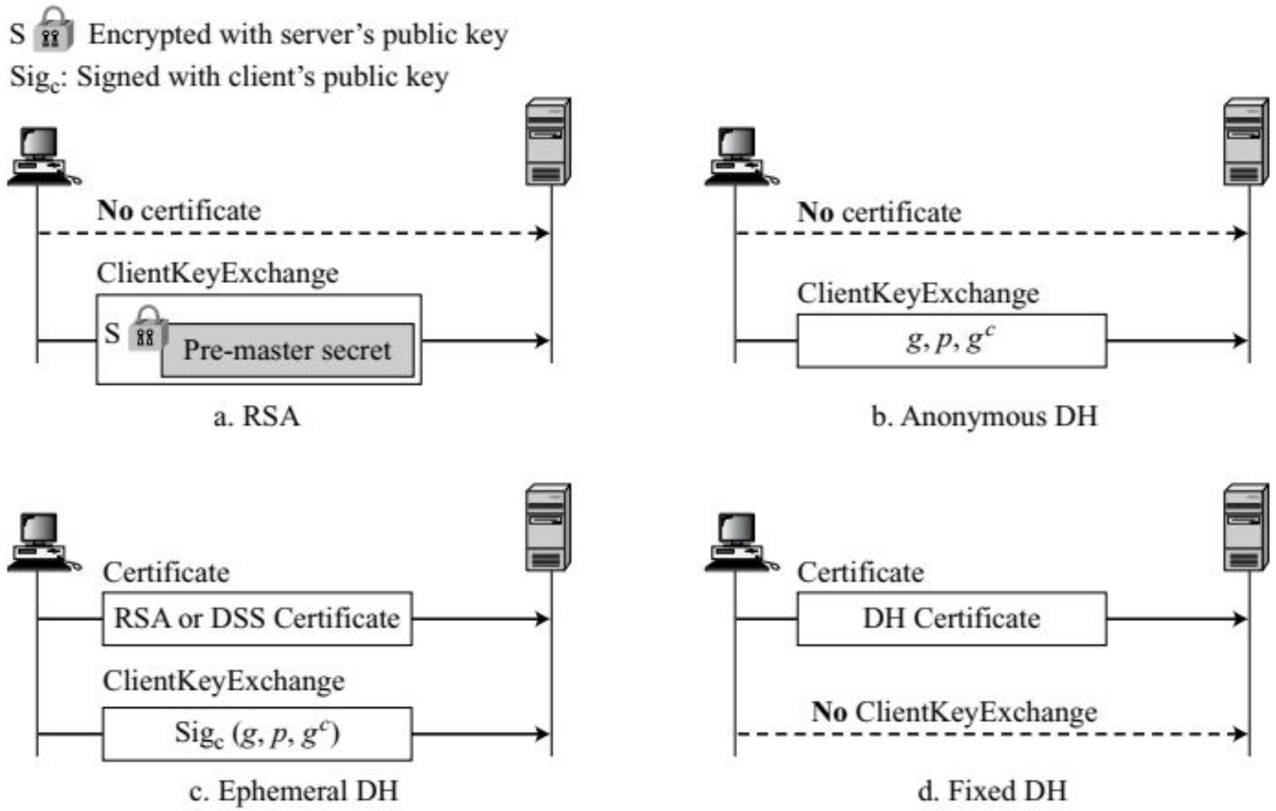

Certificate
RSA Enc-cert
No ServerKeyExchange
a. RSA
No certificate
ServerKeyExchange
g, p, g^s
b. Anonymous DH
Certificate
RSA or DSS Sig-cert
ServerKeyExchange
Sig_s (g, p, g^s)
c. Ephemeral DH
Certificate
DH cert
No ServerKeyExchange
d. Fixed DH

Phase III: Client Key Exchange and Authentication

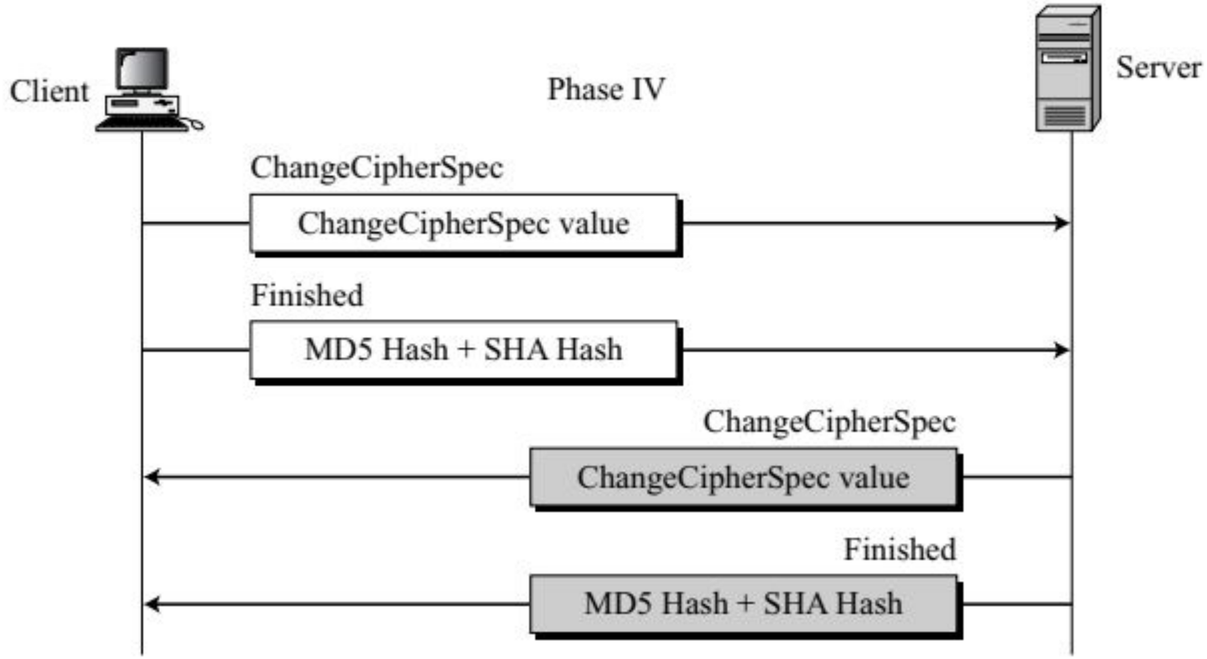


After Phase III,

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


S Encrypted with server's public key
Sig$_c$: Signed with client's public key
No certificate
ClientKeyExchange
S Pre-master secret
a. RSA
No certificate
ClientKeyExchange
g, p, g^c
b. Anonymous DH
Certificate
RSA or DSS Certificate
ClientKeyExchange
Sig$_c$ (g, p, g^c)
c. Ephemeral DH
Certificate
DH Certificate
No ClientKeyExchange
d. Fixed DH

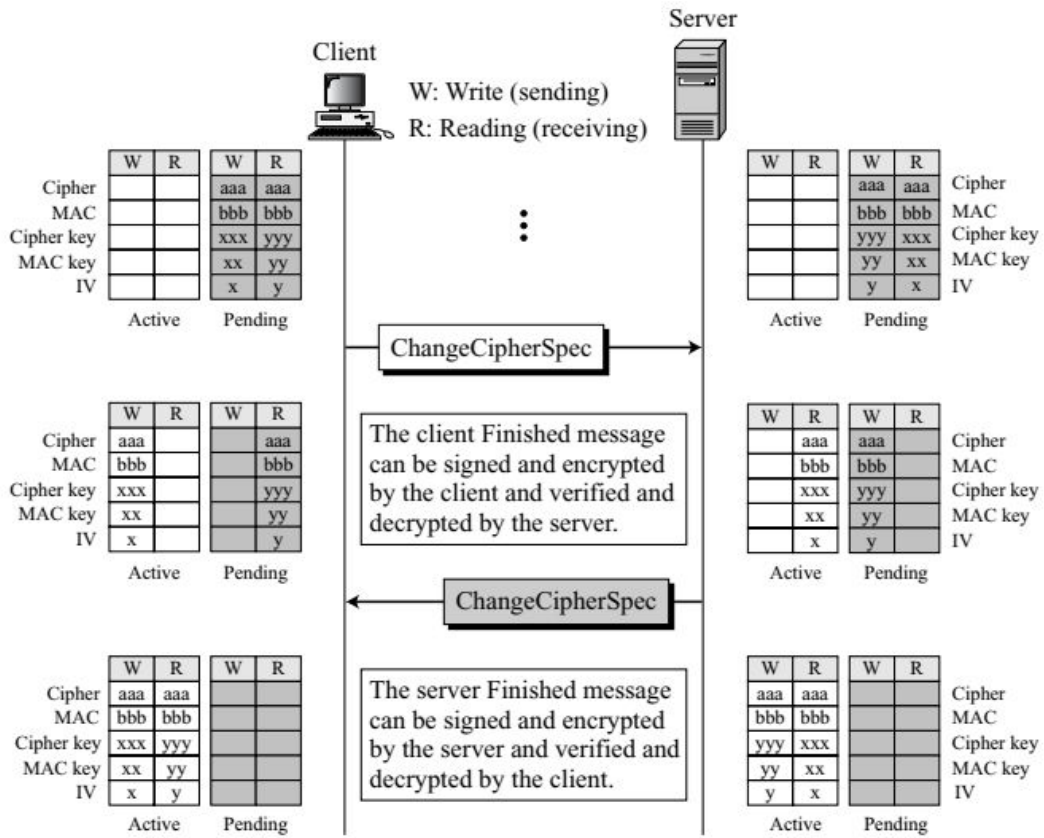
Finalizing and Finishing



Change Cipher Spec Protocol

- We have seen that the negotiation of the cipher suite and the generation of cryptographic secrets are formed gradually during the Handshake Protocol.
- The question now is: When can the two parties use these parameter secrets?
- SSL mandates that the parties cannot use these parameters or secrets until they have sent or received a special message, the ChangeCipherSpec message, which is exchanged during the Handshake protocol and defined in the ChangeCipherSpec Protocol.
- The sender and the receiver need two states, not one.
- One state, the pending state, keeps track of the parameters and secrets.
- The other state, the active state, holds parameters and secrets used by the Record Protocol to sign/verify or encrypt/decrypt messages.
- In addition, each state holds two sets of values: read (inbound) and write (outbound).

Change Cipher Spec Protocol

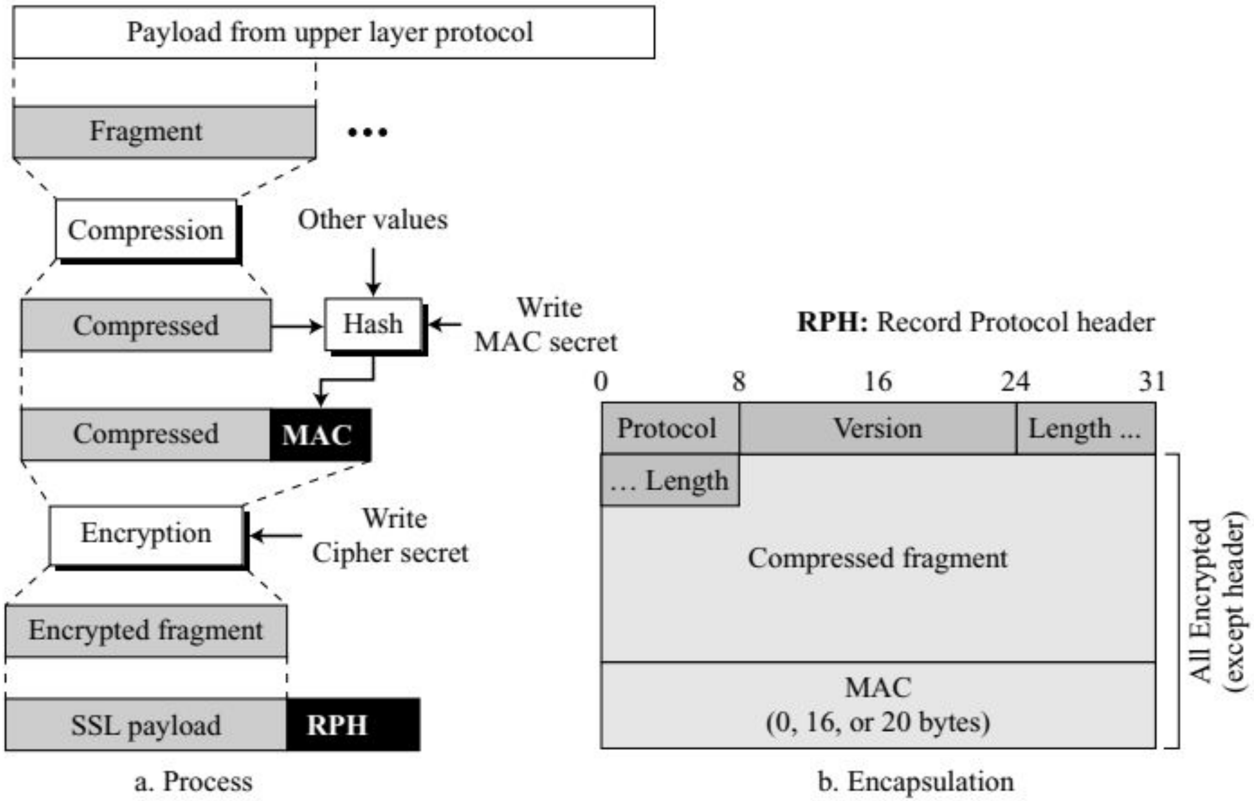


Alert Protocol

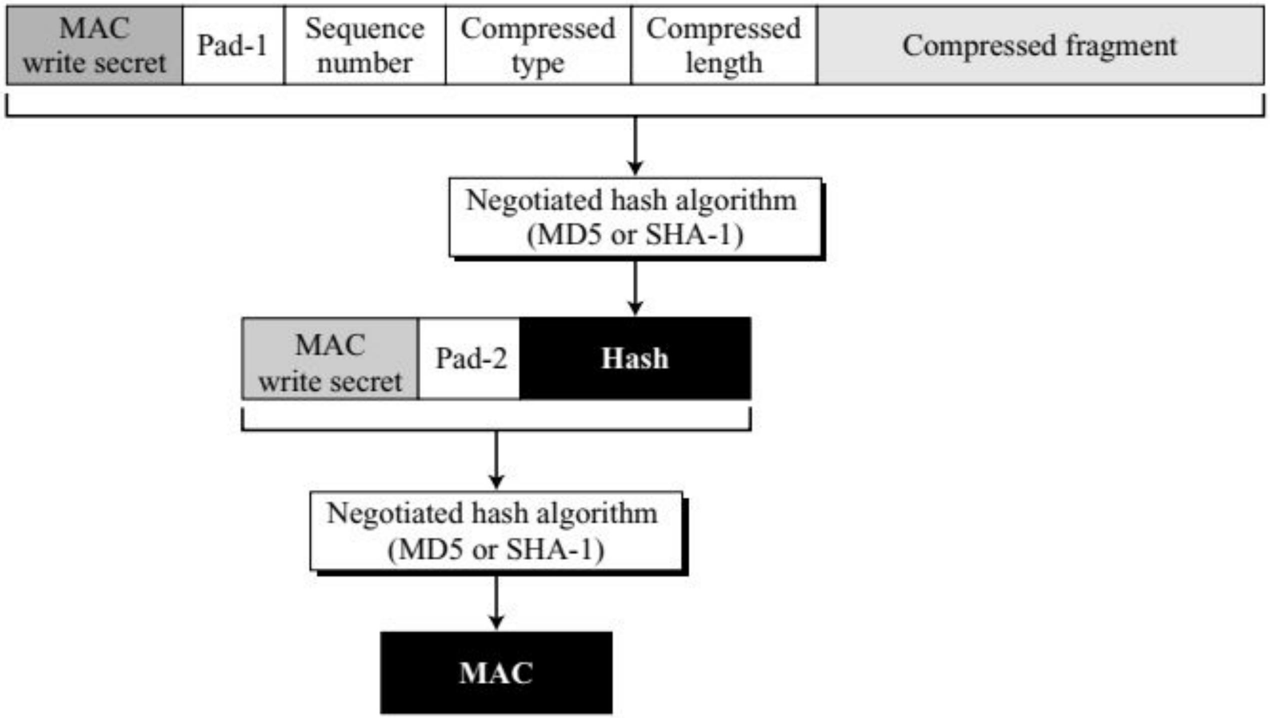
Table 17.4 *Alerts defined for SSL*

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

Record Protocol



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

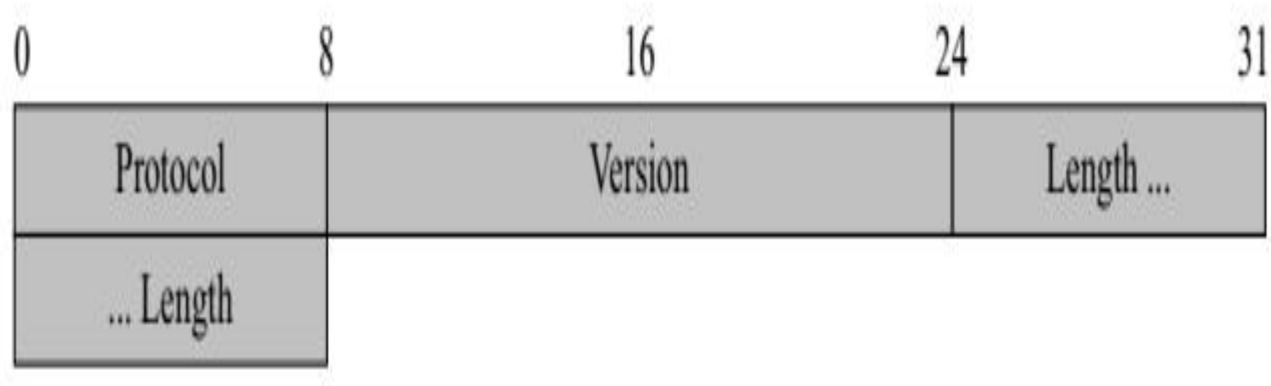


- The hash algorithm is applied twice. First, a hash is created from the concatenations of the following values:
- The MAC write secret (authentication key for the outbound message)
- Pad-1, which is the byte 0x36 repeated 48 times for MD5 and 40 times for SHA-1
- The sequence number for this message
- The compressed type, which defines the upper-layer protocol that provided the compressed fragment
- The compressed length, which is the length of the compressed fragment
- The compressed fragment itself

Second, the final hash (MAC) is created from the concatenation of the following values:

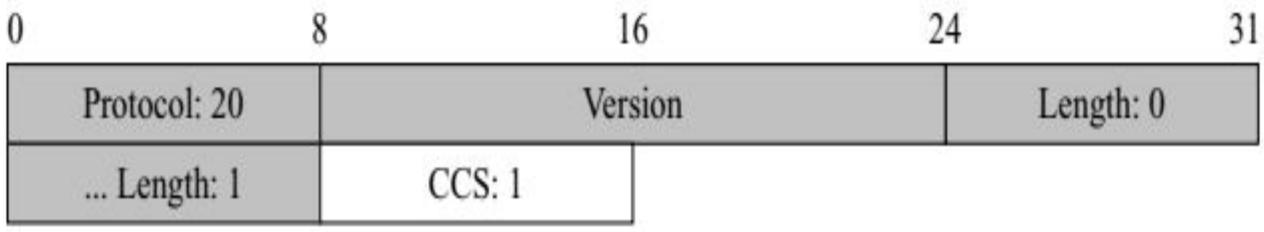
- a. The MAC write secret
- b. Pad-2, which is the byte 0x5C repeated 48 times for MD5 and 40 times for SHA-1
- c. The hash created from the first step

SSL MESSAGE FORMATS

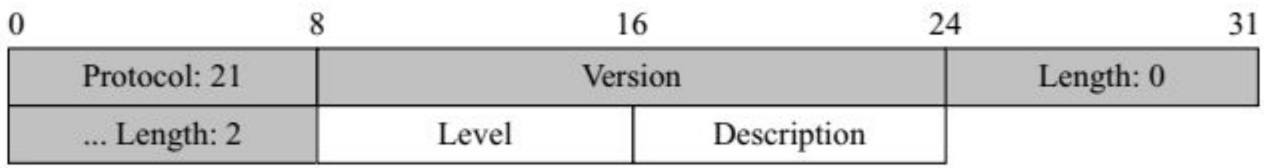


- **Protocol:** This 1-byte field defines the source or destination of the encapsulated message. It is used for multiplexing and demultiplexing. The values are 20 (ChangeCipherSpec Protocol), 21 (Alert Protocol), 22 (Handshake Protocol), and 23 (data from the application layer).
- **Version:** This 2-byte field defines the version of the SSL; one byte is the major version and the other is the minor. The current version of SSL is 3.0 (major 3 and minor 0).
- **Length:** This 2-byte field defines the size of the message (without the header) in bytes

ChangeCipherSpec Protocol

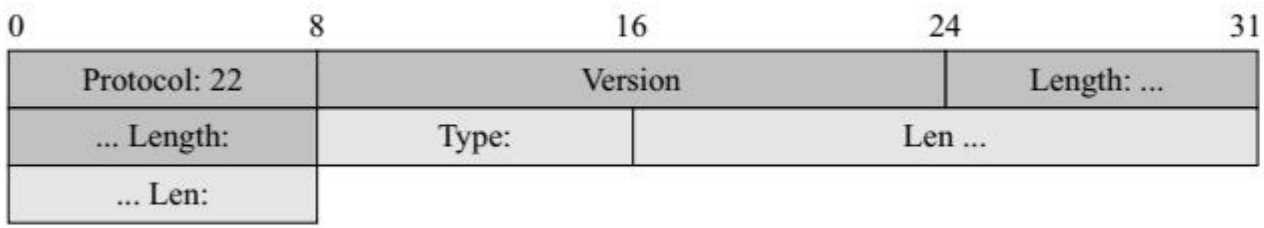


Alert Protocol



- Level. This one-byte field defines the level of the error. Two levels have been defined so far: warning and fatal.
- Description. The one-byte description defines the type of error.

Handshake Protocol

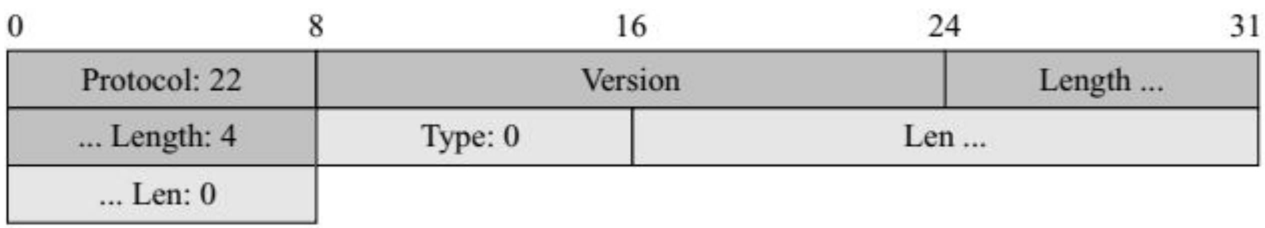


Type: This one-byte field defines the type of message. So far ten types have been defined as listed in Table 17.5.

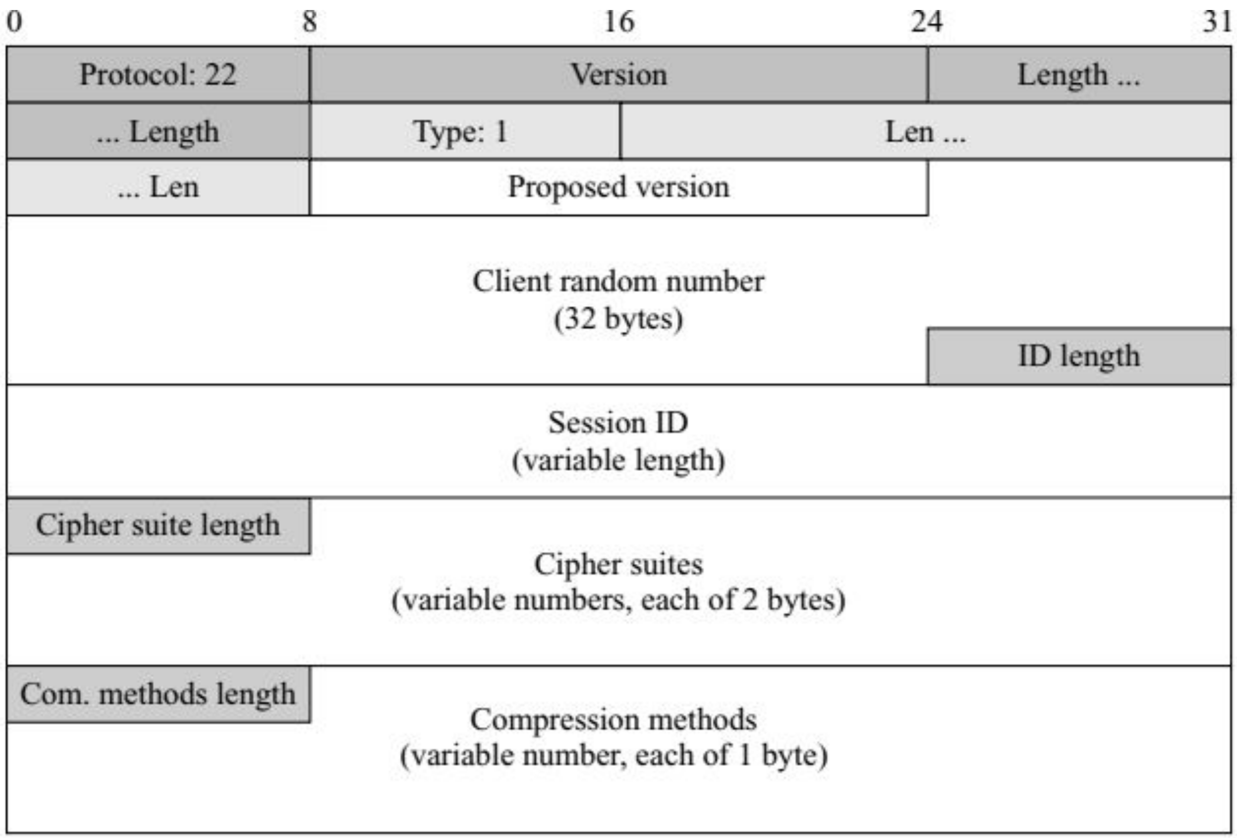
Length (Len): This three-byte field defines the length of the message (excluding the length of the type and length field). The reader may wonder why we need two length fields, one in the general Record header and one in the generic header for the Handshake messages. The answer is that a Record message may carry two Handshake messages at the same time if there is no need for another message in between.

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

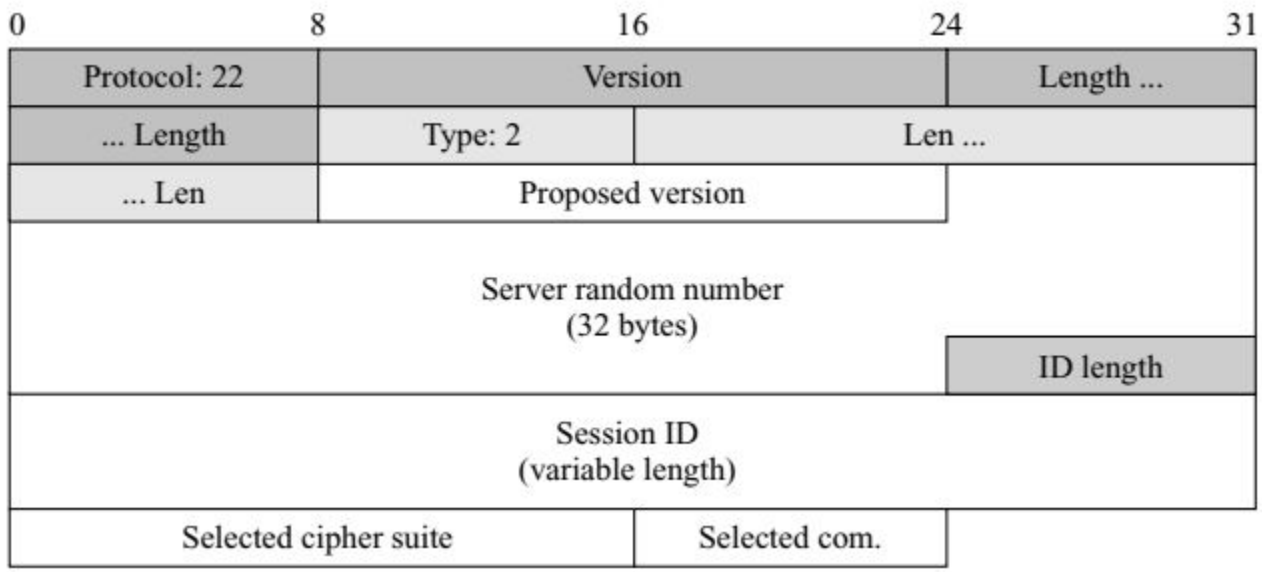
HelloRequest Message



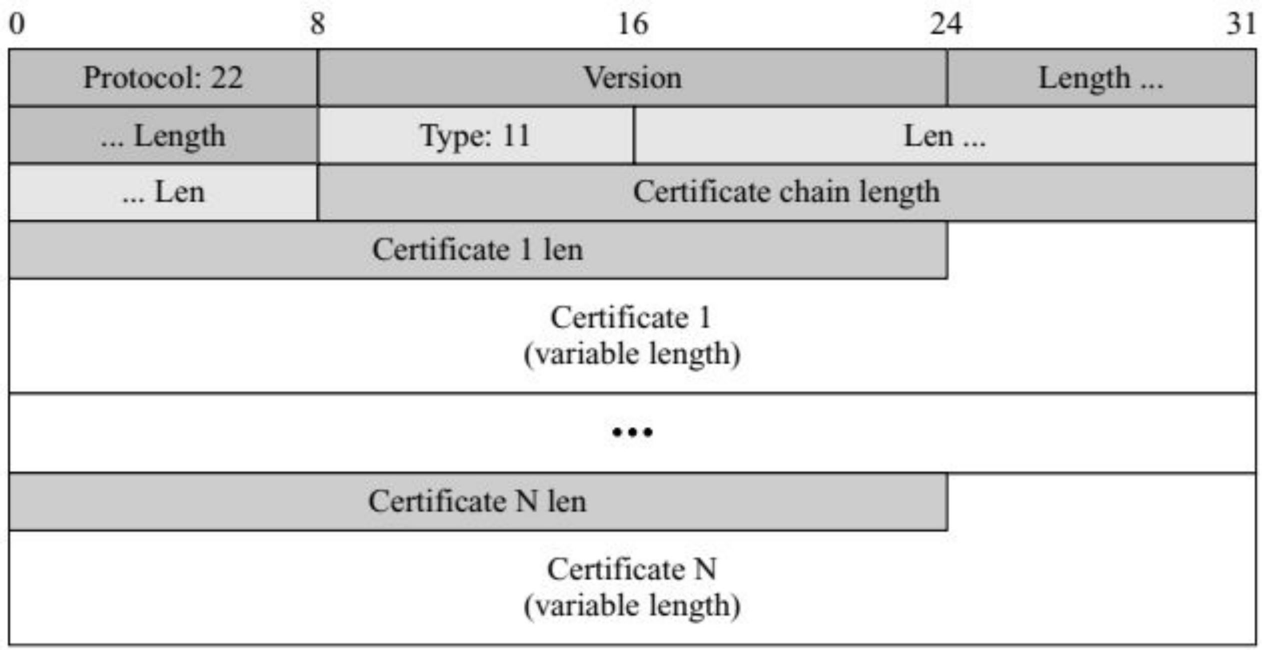
ClientHello Message



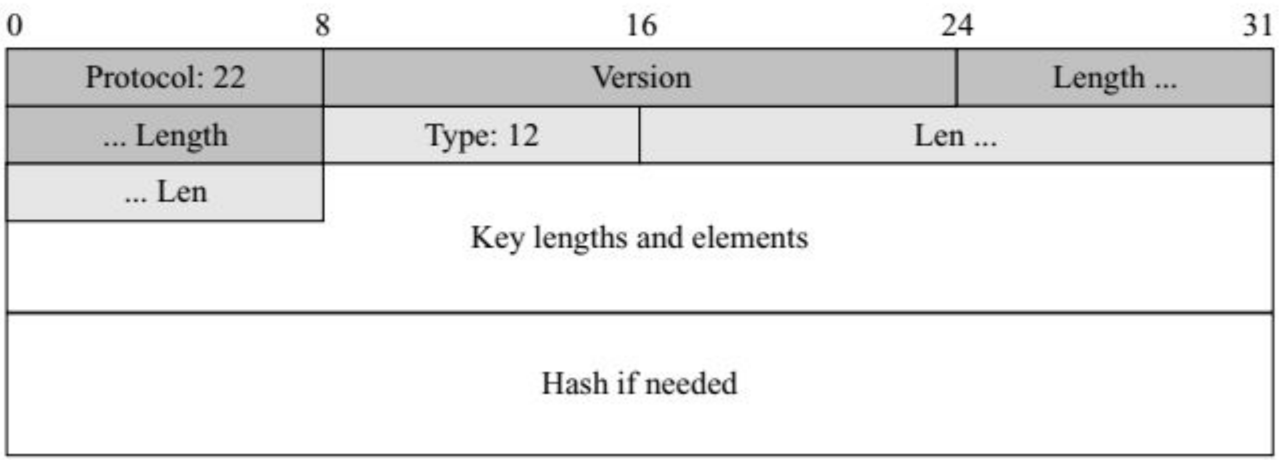
ServerHello Message



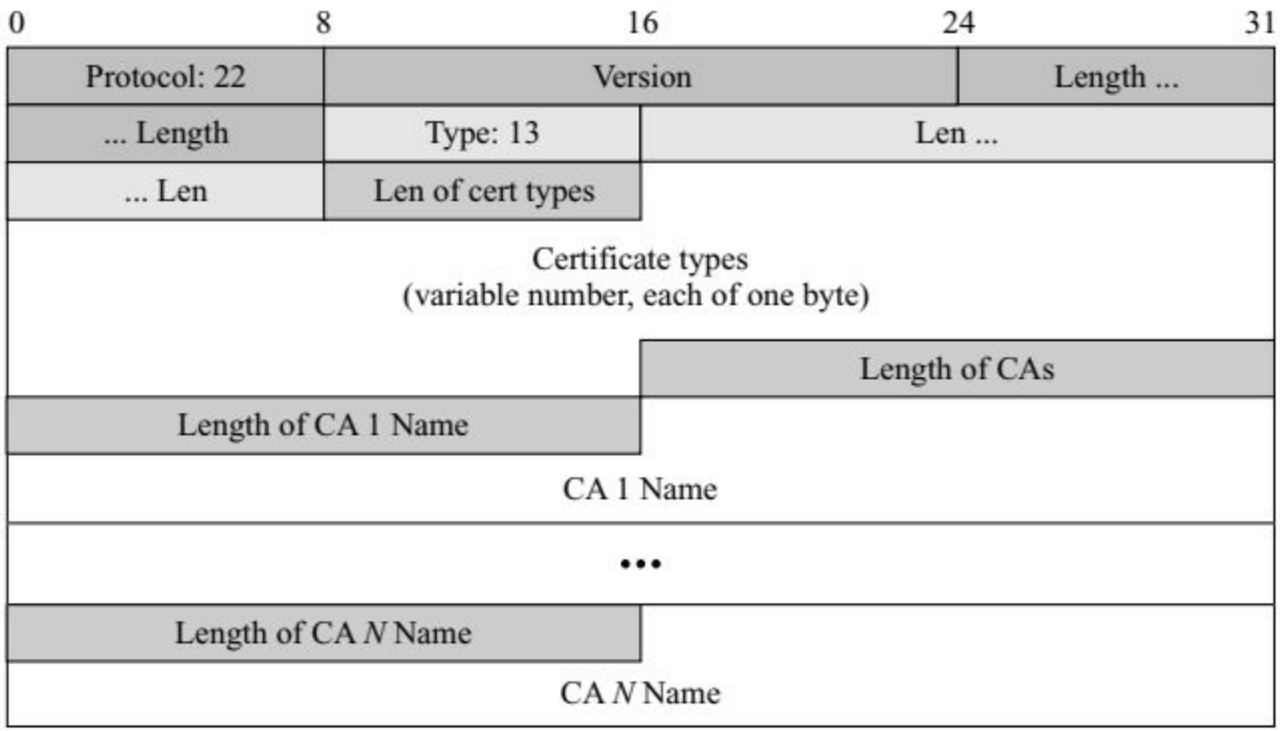
Certificate Message



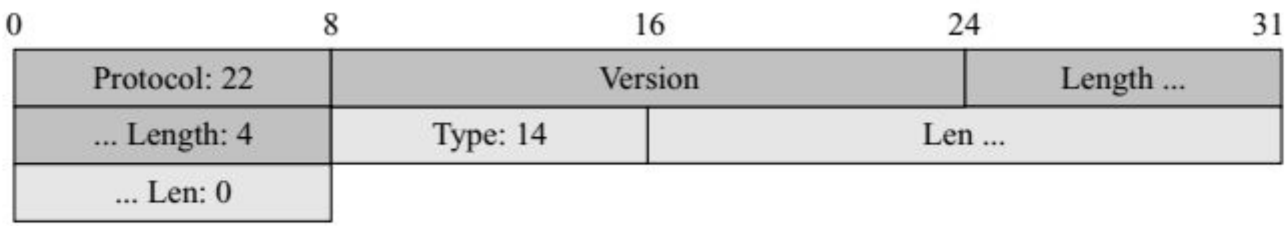
ServerKeyExchange Message



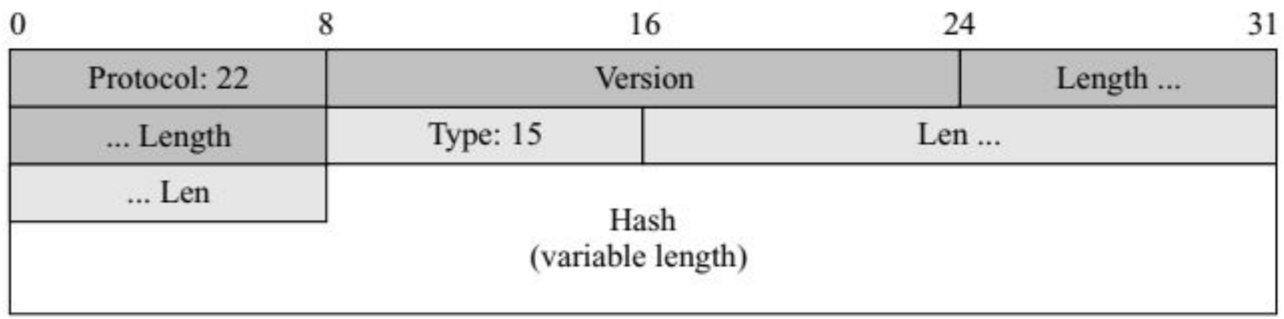
CertificateRequest Message



ServerHelloDone Message



CertificateVerify Message



ClientKeyExchange Message

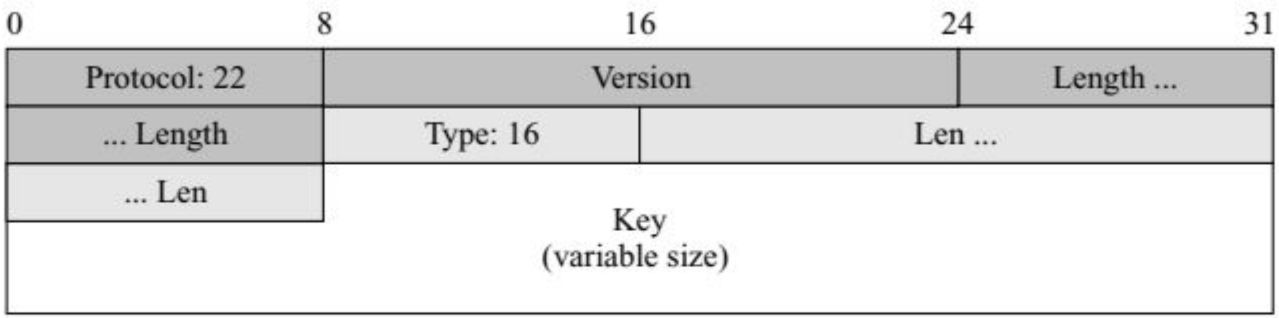
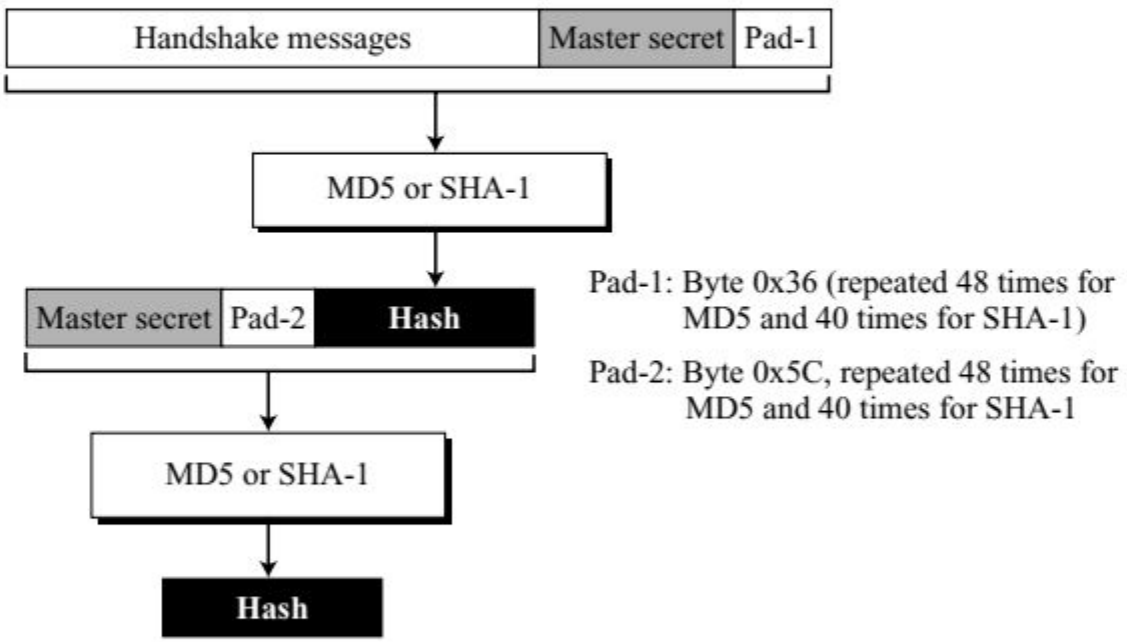


Figure 17.35 *Hash calculation for CertificateVerify message*



Finished Message

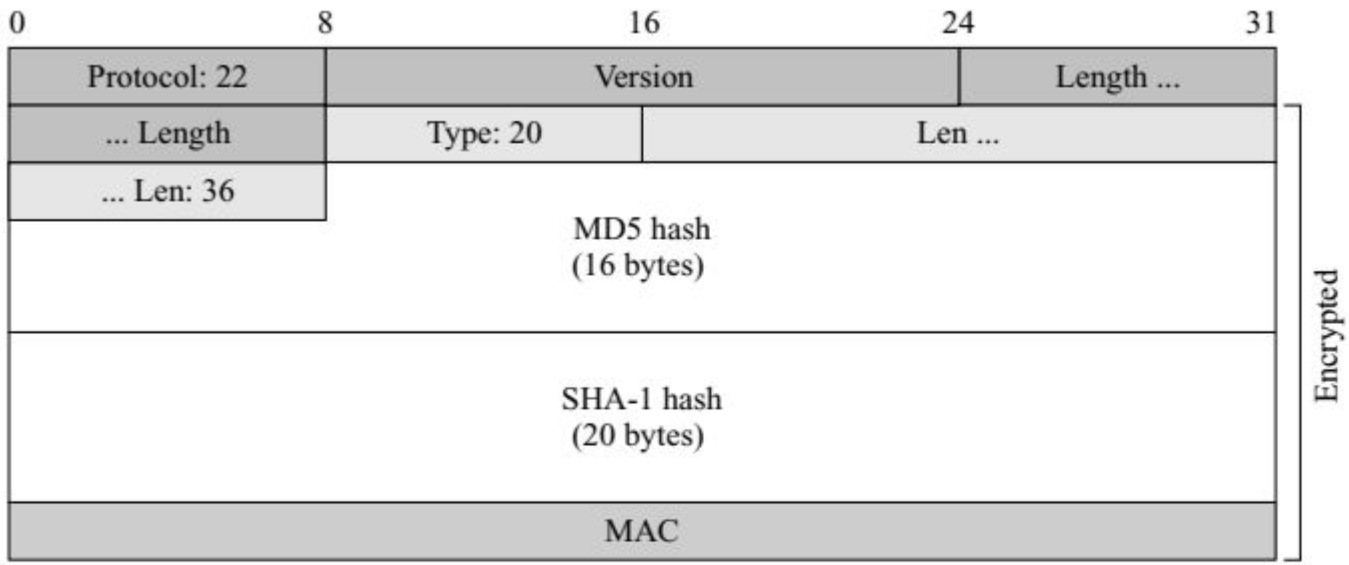
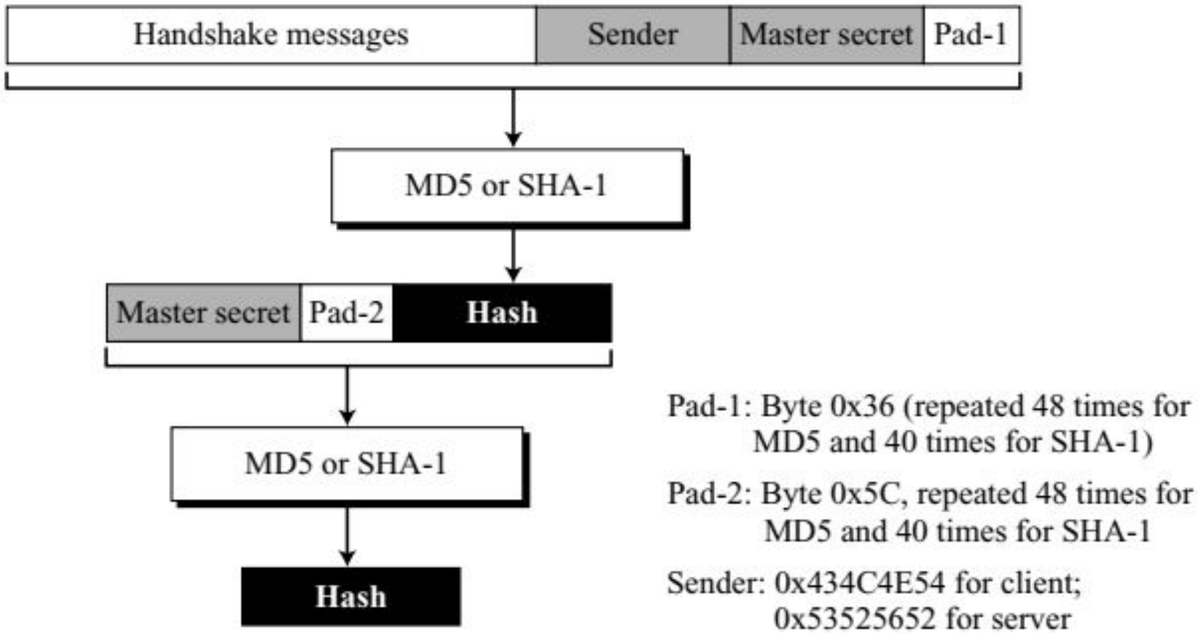
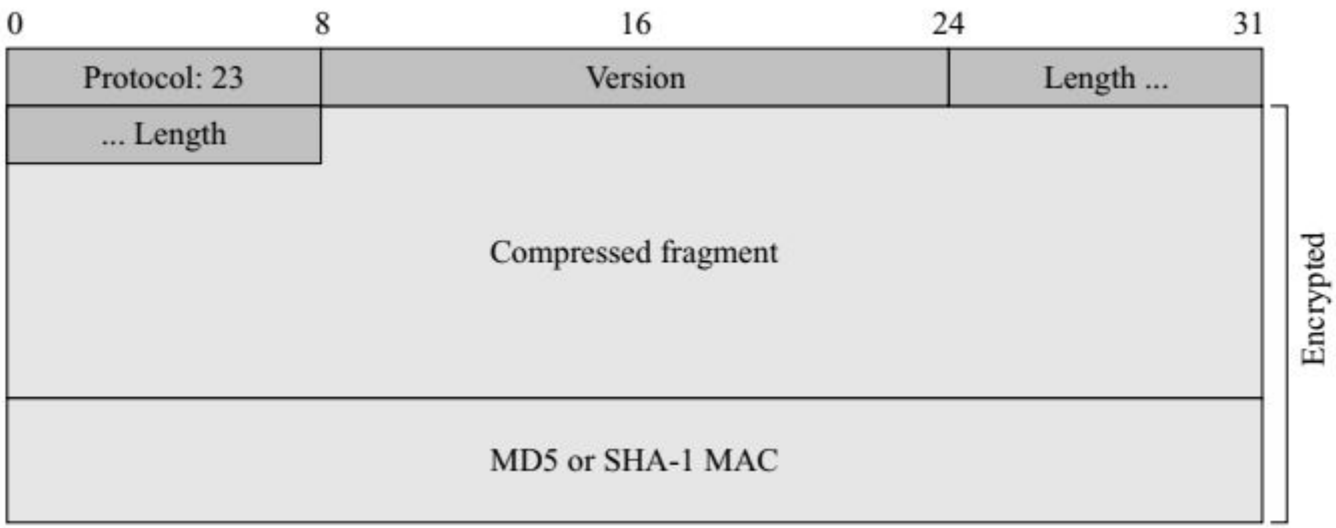


Figure 17.38 *Hash calculation for Finished message*



TRANSPORT LAYER SECURITY

The Transport Layer Security (TLS) protocol is the IETF standard version of the SSL protocol. The two are very similar, with slight differences. Instead of describing TLS in full, we highlight the differences between TLS and SSL protocols in this section.



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.

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.

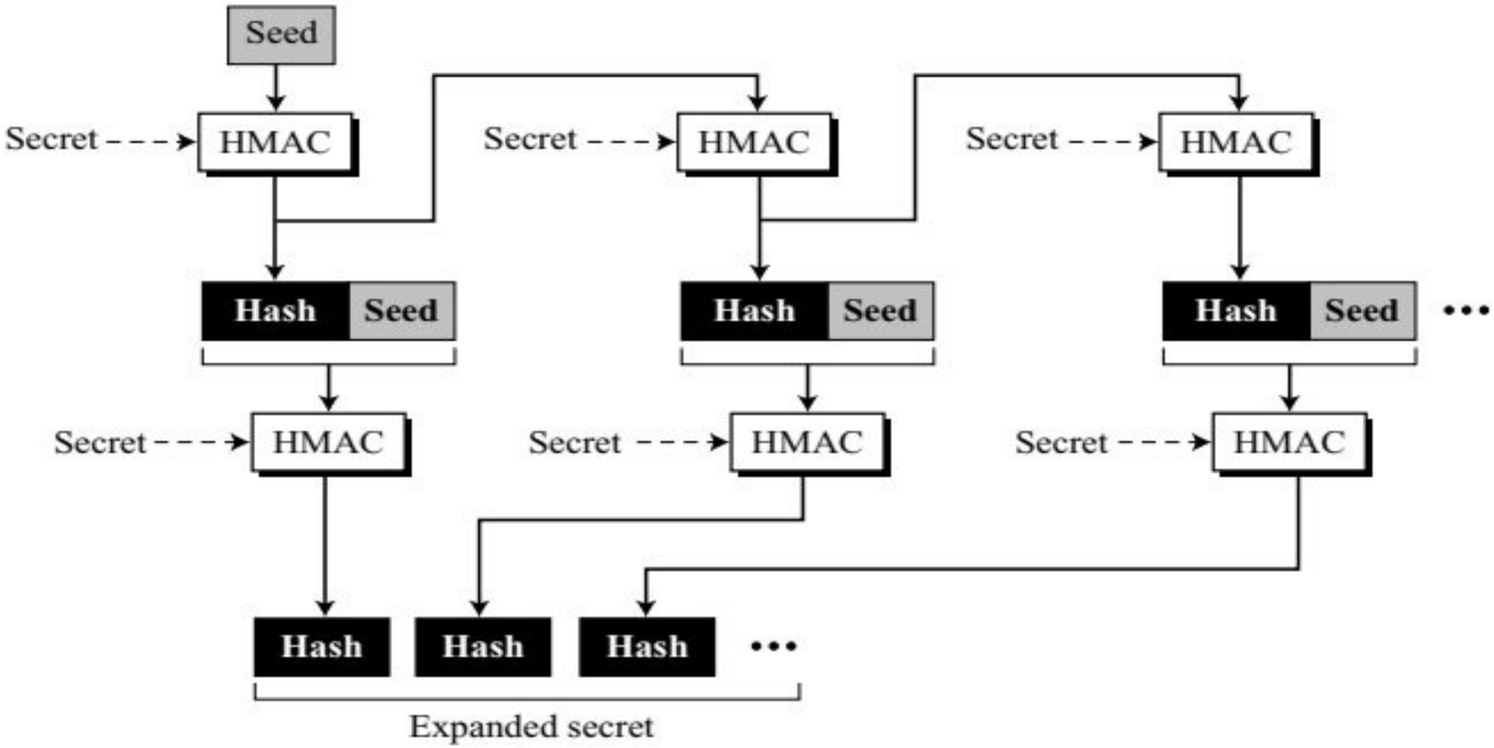
Cipher suite	*Key Exchange*	*Encryption*	*Hash*
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_WITH_DES_CBC_SHA	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

Generation of Cryptographic Secrets

The generation of cryptographic secrets is more complex in TLS than in SSL. TLS first defines two functions: the data-expansion function and the pseudorandom function.

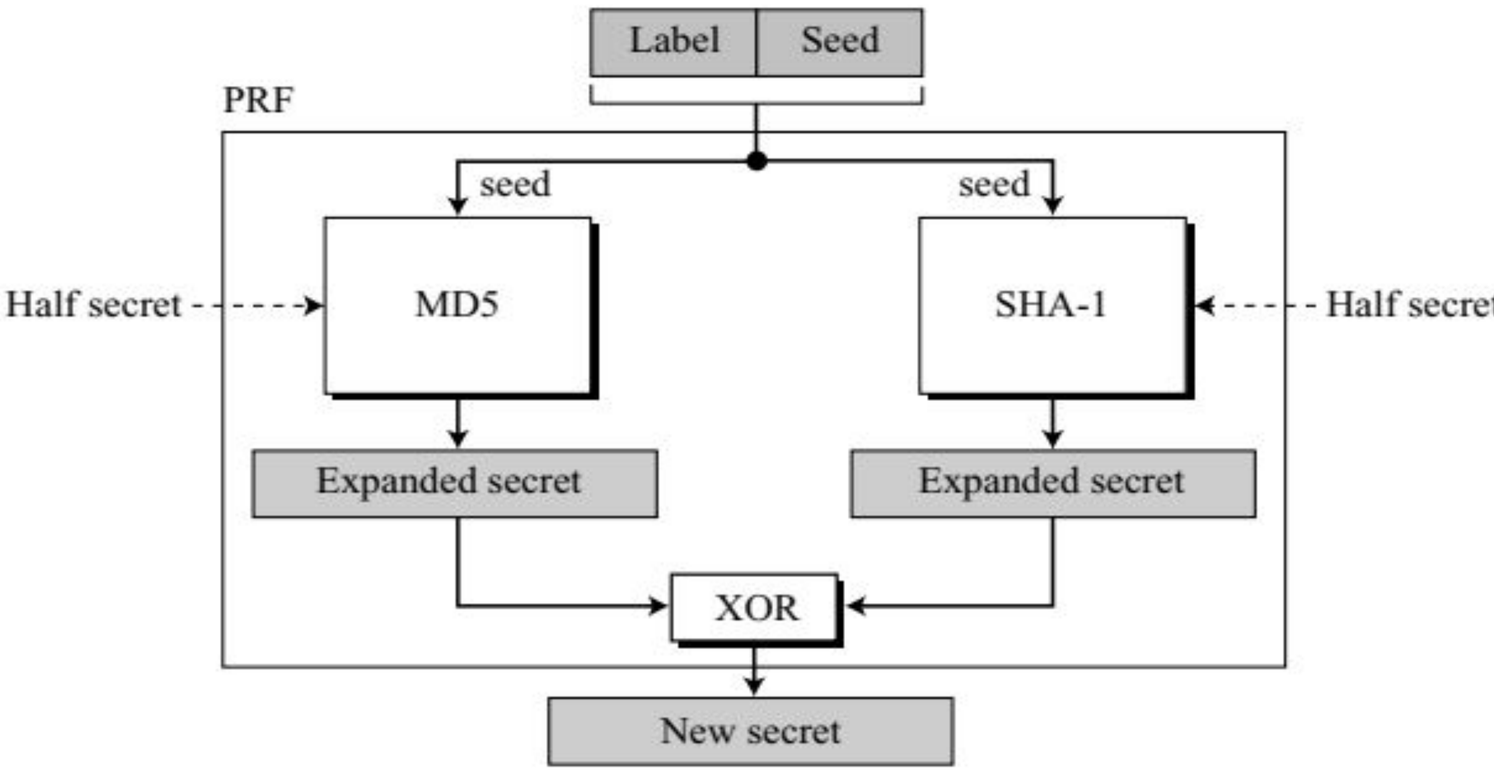
Data-Expansion Function

- The data-expansion function uses a predefined HMAC (either MD5 or SHA-1) to expand a secret into a longer one.
- This function can be considered a multiple section function, where each section creates one hash value.
- The extended secret is the concatenation of the hash values.
- Each section uses two HMACs, a secret and a seed.
- The data-expansion function is the chaining of as many sections as required.
- However, to make the next section dependent on the previous, the second seed is actually the output of the first HMAC of the previous section.


Seed
Secret
HMAC
Secret
HMAC
Secret
HMAC
Hash
Seed
Hash
Seed
Hash
Seed
•••
Secret
HMAC
Secret
HMAC
Secret
HMAC
Hash
Hash
Hash
•••
Expanded secret

Pseudorandom Function (PRF)

TLS defines a pseudorandom function (PRF) to be the combination of two data-expansion functions, one using MD5 and the other SHA-1. PRF takes three inputs, a secret, a label, and a seed. The label and seed are concatenated and serve as the seed for each dataexpansion function. The secret is divided into two halves; each half is used as the secret for each data-expansion function. The output of two data-expansion functions is exclusiveored together to create the final expanded secret. Note that because the hashes created from MD5 and SHA-1 are of different sizes, extra sections of MD5-based functions must be created to make the two outputs the same size. Figure 17.41 shows the idea of PRF

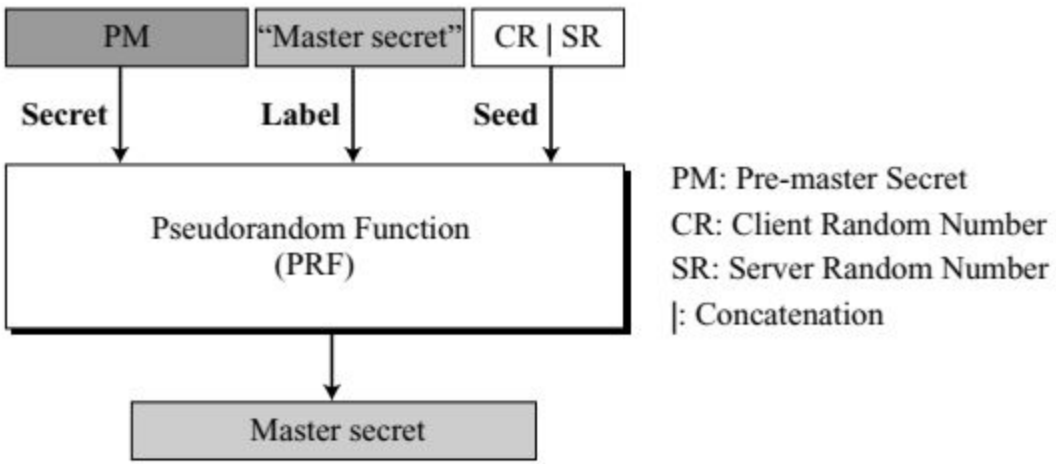

Label
Seed
PRF
seed
seed
Half secret
MD5
SHA-1
Half secret
Expanded secret
Expanded secret
XOR
New secret

Pre-master Secret

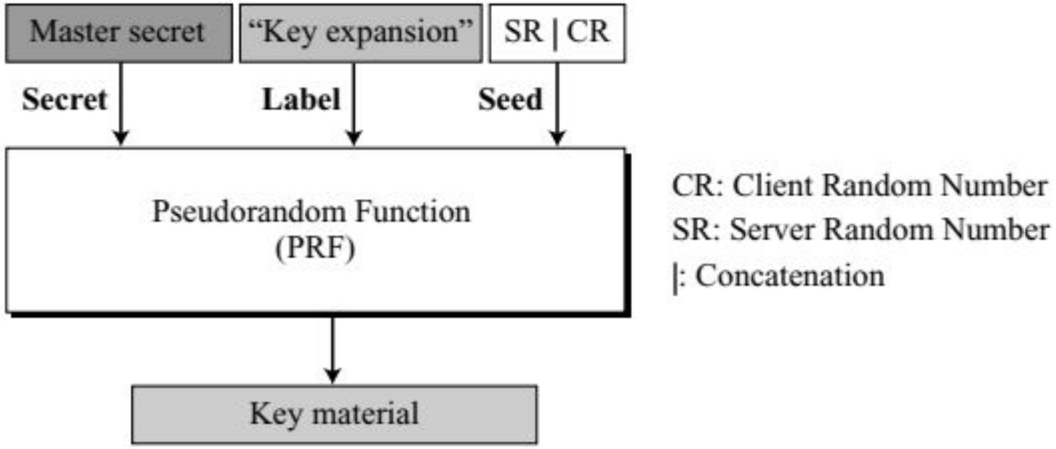
The generation of the pre-master secret in TLS is exactly the same as in SSL.

Master Secret

TLS uses the PRF function to create the master secret from the pre-master secret. This is achieved by using the pre-master secret as the secret, the string “master secret” as the label, and concatenation of the client random number and server random number as the seed. Note that the label is actually the ASCII code of the string “master secret”. In other words, the label defines the output we want to create, the master secret. Figure 17.42 shows the idea.


PM
"Master secret"
CR | SR
Secret
Label
Seed
Pseudorandom Function
(PRF)
Master secret
PM: Pre-master Secret
CR: Client Random Number
SR: Server Random Number
|: Concatenation

Key Material



Alert Protocol

TLS supports all of the alerts defined in SSL except for NoCertificate.

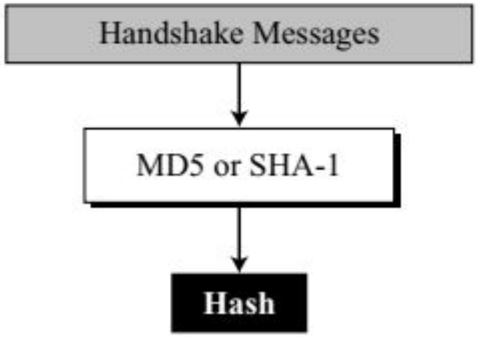
Value	*Description*	*Meaning*
0	*CloseNotify*	Sender will not send any more messages.
10	*UnexpectedMessage*	An inappropriate message received.
20	*BadRecordMAC*	An incorrect MAC received.
21	*DecryptionFailed*	Decrypted message is invalid.
22	*RecordOverflow*	Message size is more than $2^{14} + 2048$.
30	*DecompressionFailure*	Unable to decompress appropriately.
40	*HandshakeFailure*	Sender unable to finalize the handshake.
42	*BadCertificate*	Received certificate corrupted.
43	*UnsupportedCertificate*	Type of received certificate is not supported.
44	*CertificateRevoked*	Signer has revoked the certificate.
45	*CertificateExpired*	Certificate has expired.
46	*CertificateUnknown*	Certificate unknown.
47	*IllegalParameter*	A field out of range or inconsistent with others.
48	*UnknownCA*	CA could not be identified.

Handshake Protocol

TLS has made some changes in the Handshake Protocol. Specifically, the details of the CertificateVerify message and the Finished message have been changed.

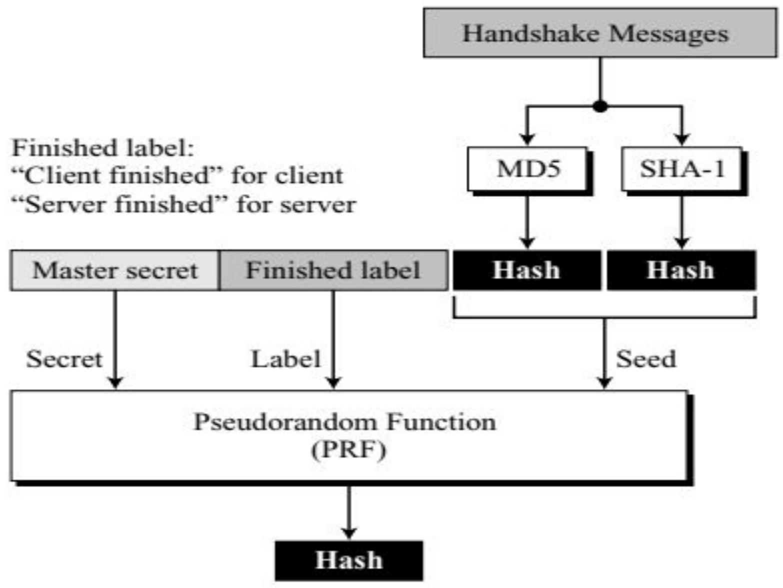
CertificateVerify Message

In SSL, the hash used in the CertificateVerify message is the two-step hash of the handshake messages plus a pad and the master secret. TLS has simplified the process. The hash in the TLS is only over the handshake messages.



Finished Message

The calculation of the hash for the Finished message has also been changed. TLS uses the PRF to calculate two hashes used for the Finished message, as shown in Figure 17.45.



Record Protocol

The only change in the Record Protocol is the use of HMAC for signing the message. TLS uses the MAC, as defined in Chapter 11, to create the HMAC. TLS also adds the protocol version (called Compressed version) to the text to be signed.