TLS / SSL

19CSE311 Computer Security

Jevitha KP

Department of CSE

TLS / SSL

- Transport layer security provides end-to-end security services for applications that use a reliable transport layer protocol such as TCP.

- **Transport Layer Security (TLS)**, the successor of the now-deprecated **Secure Sockets Layer (SSL)**, is a cryptographic protocol designed to provide **communications security** over a computer network.

- TLS is a proposed Internet Engineering Task Force (IETF) standard, first defined in 1999, and the current version is TLS 1.3, defined in RFC 8446 (August 2018).

- TLS builds on the earlier SSL specifications (1994, 1995, 1996) developed by Netscape Communications

- Although the SSL protocol was deprecated with the release of TLS 1.0 in 1999, it is still common to refer to these related technologies as **“SSL” or “SSL/TLS.”**

SSL

- SSL provides a secure channel between two machines or devices operating over the internet or an internal network.

- Used for establishing **authenticated** and **encrypted links** between networked computers.

- One common example is when SSL is used to secure communication between a web browser and a web server.

- This turns a website's address from HTTP to HTTPS, the ‘S’ standing for ‘secure’.

- The protocol is widely used in applications such as email, instant messaging, and voice over IP, but its use in securing HTTPS remains the most publicly visible.

SSL

- The protocol aims primarily to provide
- cryptography - including privacy (**confidentiality**),
- **integrity**, and
- **authenticity** through the use of certificates, between two or more communicating computer applications.

SSL

- For example, when a customer shops online, the following security services are desired:

    - The customer needs to be sure that the server belongs to the actual vendor, not an impostor. The customer does not want to give an impostor her credit card number (**entity authentication**).

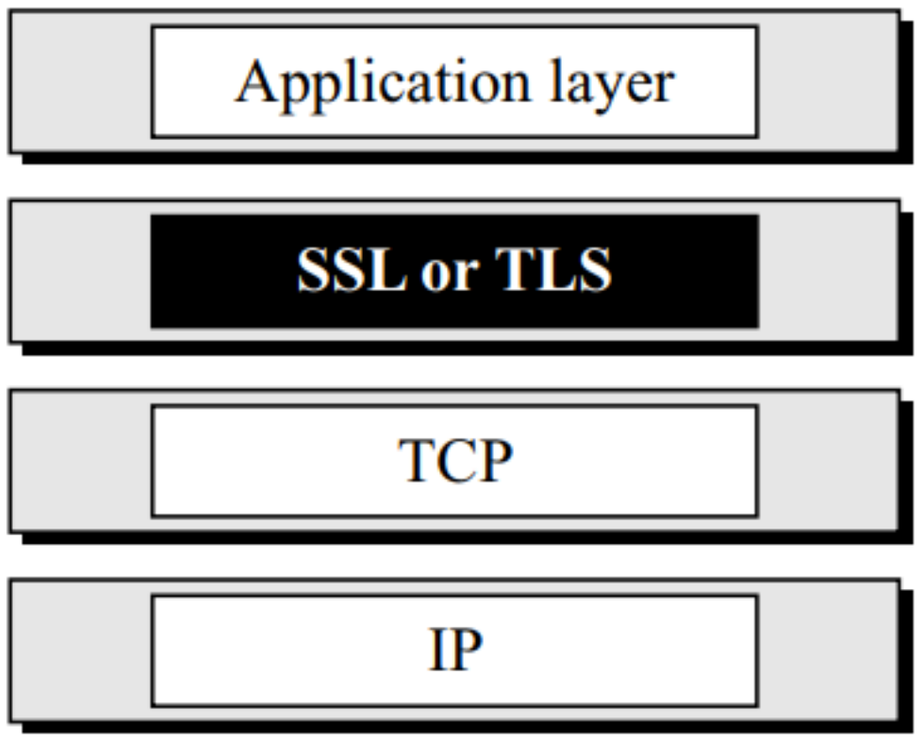
    - The customer and the vendor need to be sure that the contents of the message are not modified during transmission (**message integrity**).

    - The customer and the vendor need to be sure that an impostor does not intercept sensitive information such as a credit card number (**confidentiality**)

SSL Architecture

- Application-layer client/server programs, such as Hypertext Transfer Protocol (HTTP), that use the services of TCP can encapsulate their data in SSL packets.

- If the server and client are capable of running SSL (or TLS) programs then the client can use the 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.

- Typically, SSL can receive data from any application layer protocol, but usually the protocol is HTTP.

- The data received from the application is compressed (optional), signed, and encrypted.

- The data is then passed to a reliable transport layer protocol such as TCP.

SSL Services

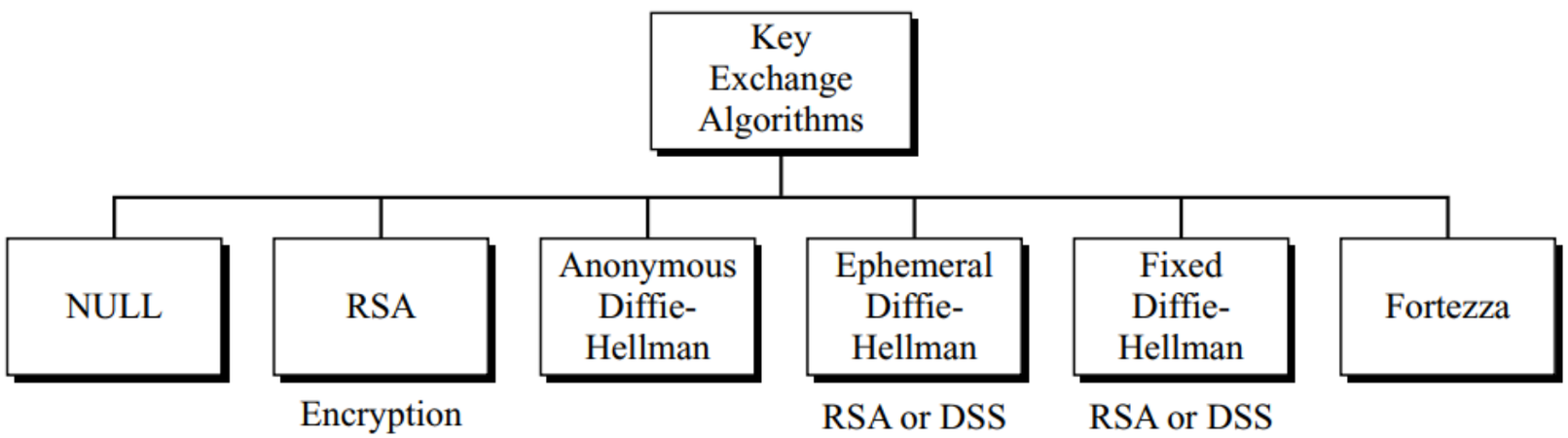
- SSL provides several services on data received from the application layer.
- **Fragmentation -** SSL divides the data into blocks of 214 bytes or less**.**
- **Compression -** Each fragment of data is compressed using one of the lossless compression methods negotiated between the client and server. This service is optional.
- **Message Integrity -** To preserve the integrity of data, SSL uses a keyed-hash function to create a Message Authentication Code(MAC).
- **Confidentiality -** To provide confidentiality, the original data and the MAC are encrypted using symmetric key cryptography.
- **Framing -** A header is added to the encrypted payload. The payload is then passed to a reliable transport layer protocol.

Key Exchange Algorithms

- To exchange an authenticated and confidential message, 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.

Key Exchange Algorithms

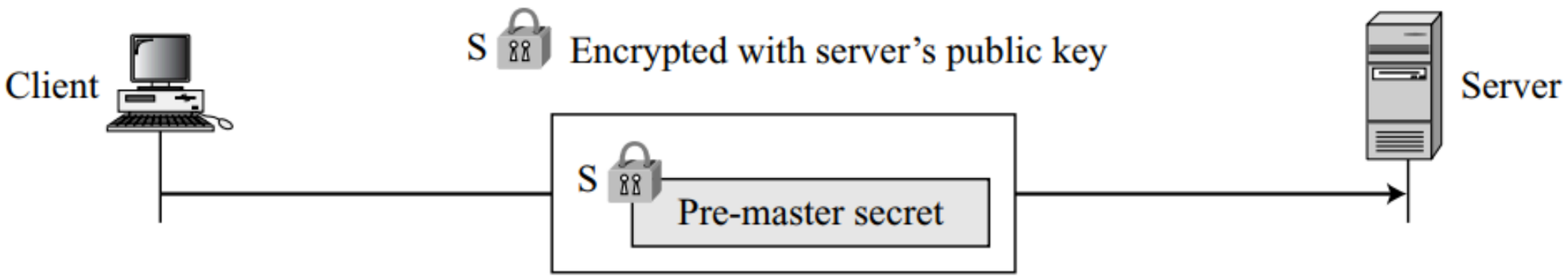
- SSL defines **six key-exchange methods** to establish this premaster secret:
  - NULL, RSA, anonymous Diffie-Hellman,
  - ephemeral Diffie-Hellman, fixed Diffie-Hellman, and Fortezza,



Key Exchange Algorithms

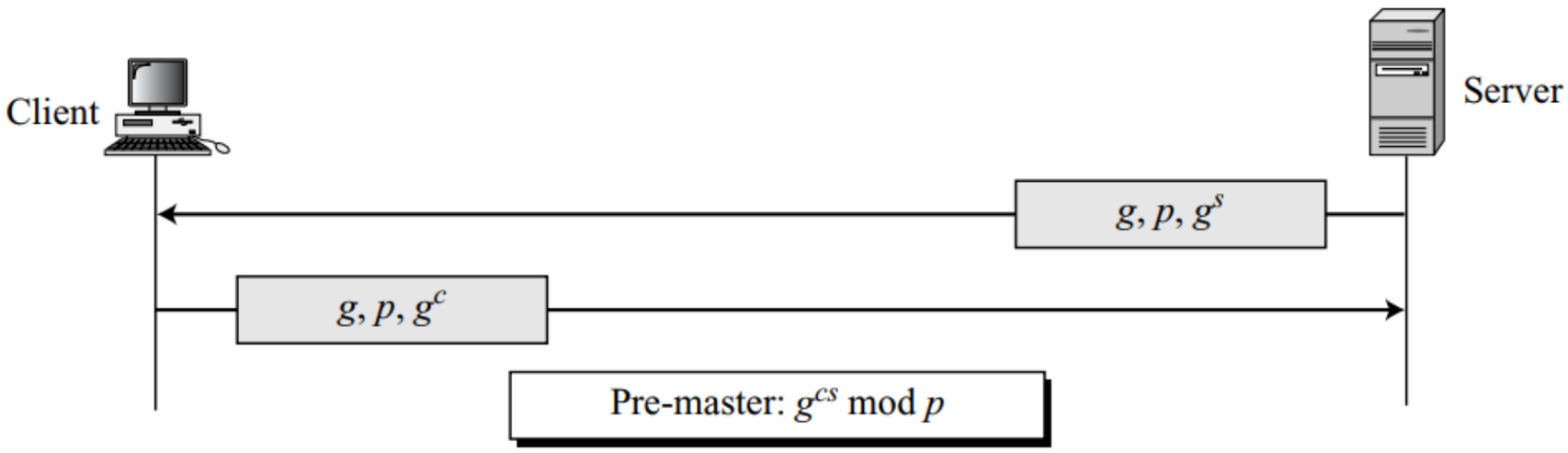
- **NULL** - There is no key exchange in this method. No pre-master secret is established between the client and the server.

- **RSA** - In this method, the pre-master secret is a 48-byte random number created by the client, encrypted with the server's RSA public key, and sent to the server. The server needs to send its RSA encryption/decryption certificate.



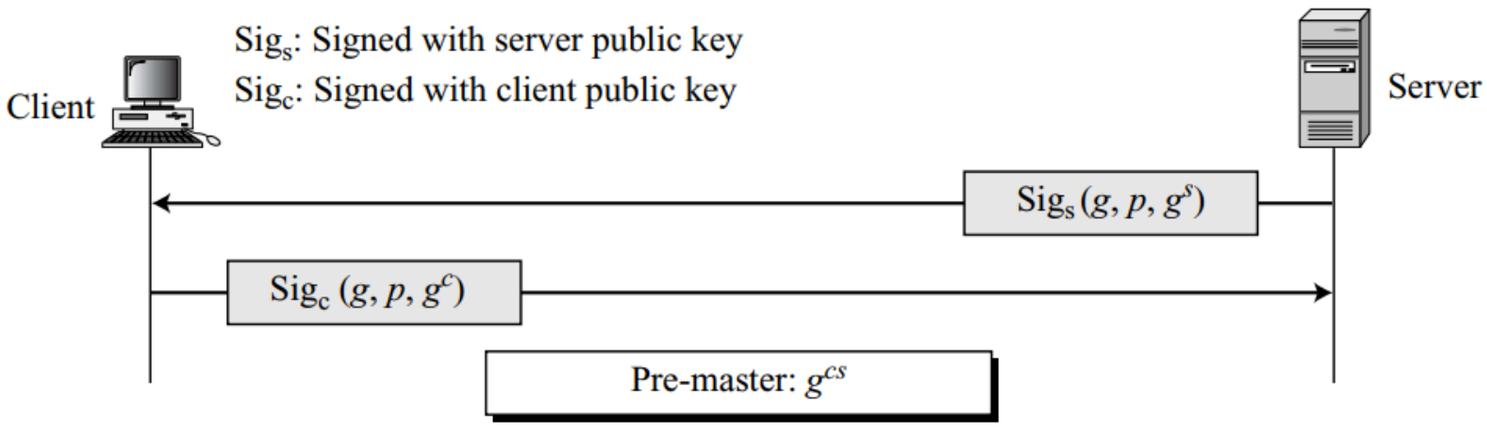
Key Exchange Algorithms

- **Anonymous Diffie-Hellman**
    - This is the simplest and most insecure method.
    - The pre-master secret is established between the client and server using the Diffie-Hellman (DH) protocol.
    - The DiffieHellman half-keys are sent in plaintext.
    - It is called anonymous Diffie-Hellman because neither party is known to the other.
    - The most serious disadvantage of this method is the man-in-the-middle attack



Key Exchange Algorithms

- **Ephemeral Diffie-Hellman**
- To thwart the man-in-the-middle attack, the ephemeral Diffie-Hellman key exchange can be used.
- Each party sends a Diffie-Hellman key signed by its private key.
- The receiving party needs to verify the signature using the public key of the sender.
- The public keys for verification are exchanged using either RSA or DSS digital signature certificates.



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).
- Then each entity can create a fixed Diffie-Hellman half-key (g^x).
- For additional security, each individual half-key is inserted into a certificate verified by a certification authority (CA).
- Two parties do not directly exchange the half-keys; the CA sends the half-keys in an RSA or DSS special certificate.
- When the client needs to calculate the pre-master, it uses its own fixed half-key and the server half-key received in a certificate.
- The server does the same, but in the reverse order.
- Note that no key-exchange messages are passed in this method; only certificates are exchanged.

Key Exchange Algorithms

- **Fortezza**

- Fortezza (derived from the Italian word for fortress) is a registered trademark of the U.S. National Security Agency (NSA).

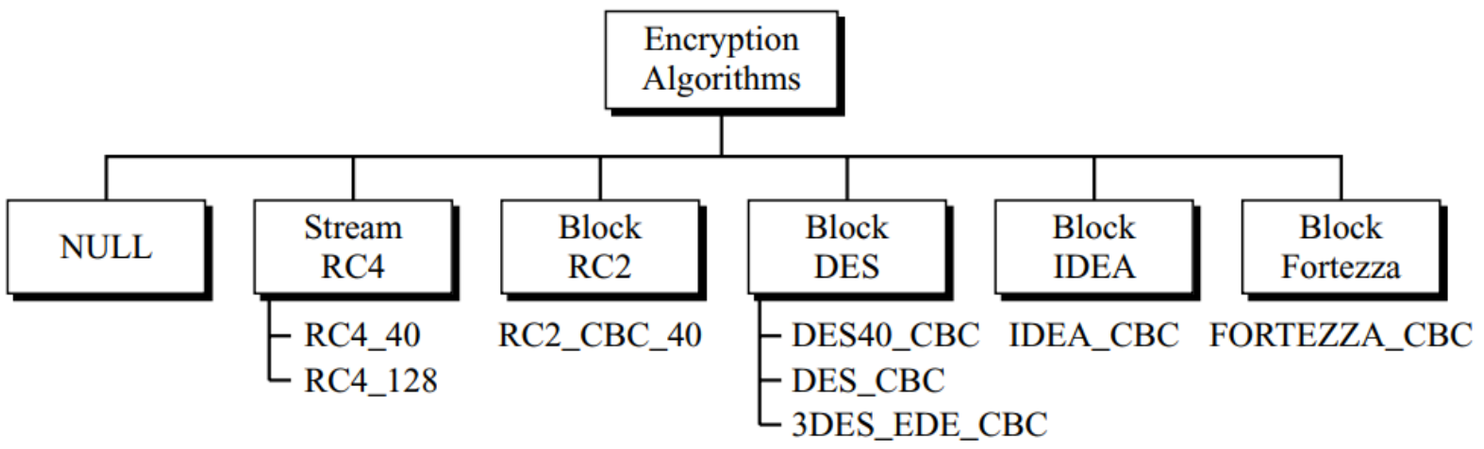
- It is a family of security protocols developed for the Defense Department.

Encryption/Decryption Algorithms

- There are several choices for the encryption/decryption algorithm.

- We can divide the algorithms into 6 groups

- All block protocols use an 8-byte initialization vector (IV) except for Fortezza, which uses a 20-byte IV



Encryption/Decryption Algorithms

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

- **Stream RC** - Two RC algorithms are defined in stream mode: RC4-40 (40-bit key) and RC4-128 (128-bit key).

- **Block RC -** One RC algorithm is defined in block mode: RC2_CBC_40 (40-bit key).

- **DES -** All DES algorithms are defined in block mode. DES40_CBC uses a 40-bit key. Standard DES is defined as DES_CBC. 3DES_EDE_CBC uses a 168-bit key.

- **IDEA** - The one 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, with a 96-bit key

Hash Algorithms

- SSL uses hash algorithms to provide message integrity (message authentication).

- Three hash functions are defined:

- **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.

Cipher Suite

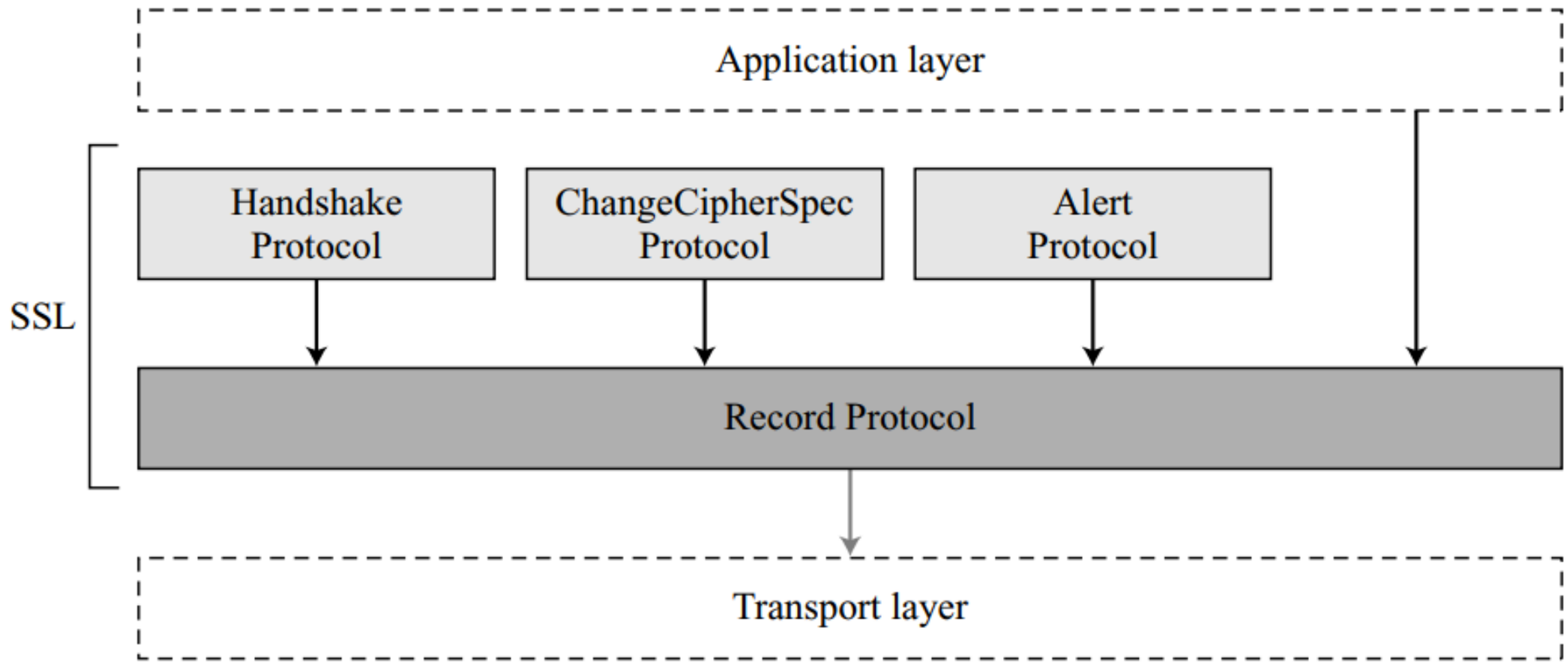
- The combination of key exchange, hash, and encryption algorithms defines a cipher suite for each SSL session.
- Each suite starts with the term “SSL” followed by the key exchange algorithm.
- The word “WITH” separates the key exchange algorithm from the encryption and hash algorithms.
- SSL_DHE_RSA_WITH_DES_CBC_SHA - 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.**
- DH is fixed Diffie-Hellman,
- DHE is ephemeral Diffie-Hellman, and
- DH-anon is anonymous Diffie-Hellman.

Cipher Suite

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

SSL Protocols

- SSL defines **four** protocols in **two layers** :



SSL Protocols

- **The Record Protocol**
    - It is the carrier protocol.
    - It carries messages from three other protocols as well as the data coming from the application layer.
        - Messages from the Record Protocol are payloads to the transport layer, normally TCP.
- **The Handshake Protocol**
    - provides security parameters for the Record Protocol.
    - It establishes a cipher set and provides keys and security parameters.
    - It also authenticates the server to the client and the client to the server if needed.

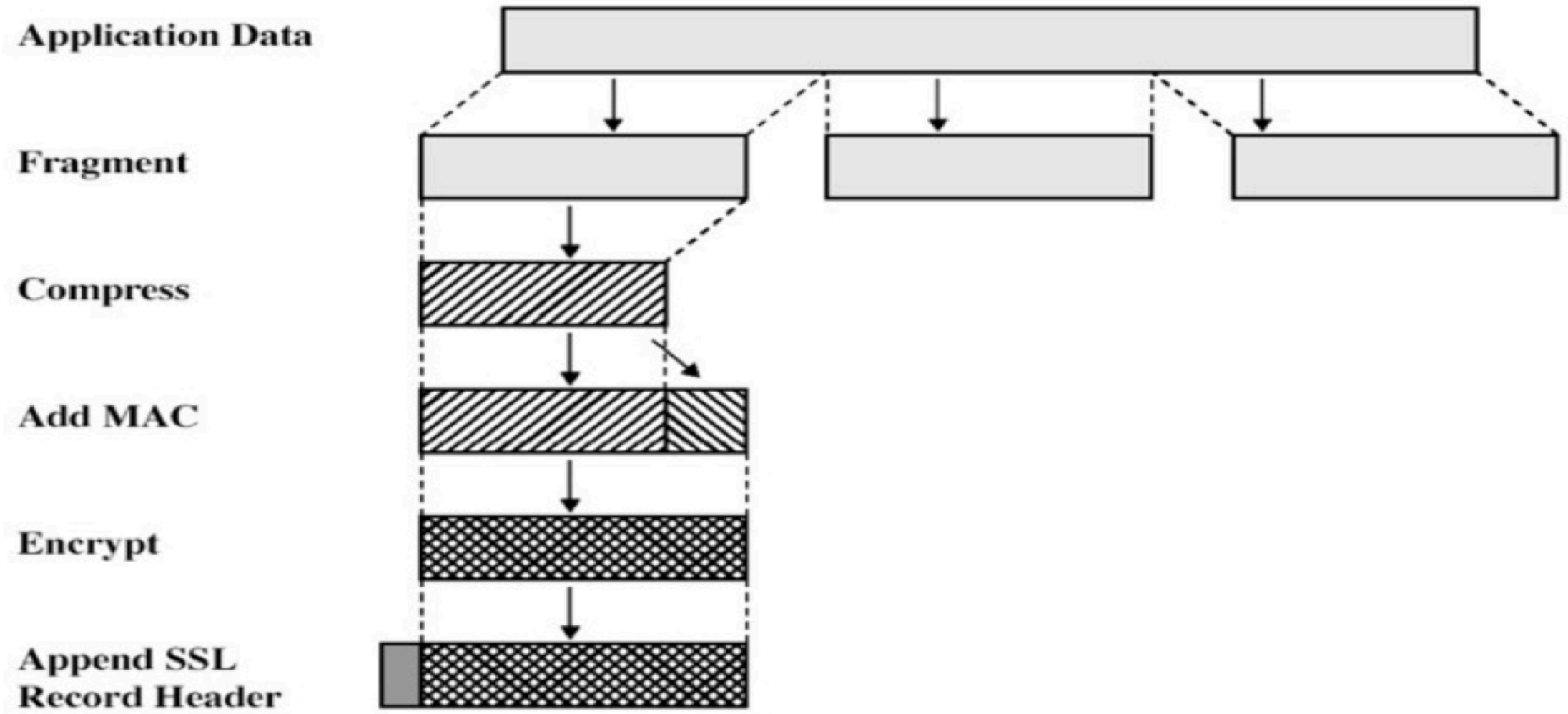
SSL Protocols

- **The ChangeCipherSpec Protocol**
    - It is used for signalling the readiness of cryptographic secrets.
- **The Alert Protocol**
    - It is used to report abnormal conditions.

SSL Record Protocol

- This protocol provides two services for SSL connections:
  - **Confidentiality** - using conventional encryption.
  - **Message Integrity** - using a Message Authentication Code (MAC).

SSL Record Protocol



SSL Record Protocol

- In order to operate on data the protocol performs the following actions:

  - It takes an application message to be transmitted and fragments it into manageable blocks. These blocks are 2^{14} = 16, 384 bytes or less.

  - These blocks are then optionally compressed which must be lossless and may not increase the content length by more than 1024 bytes.

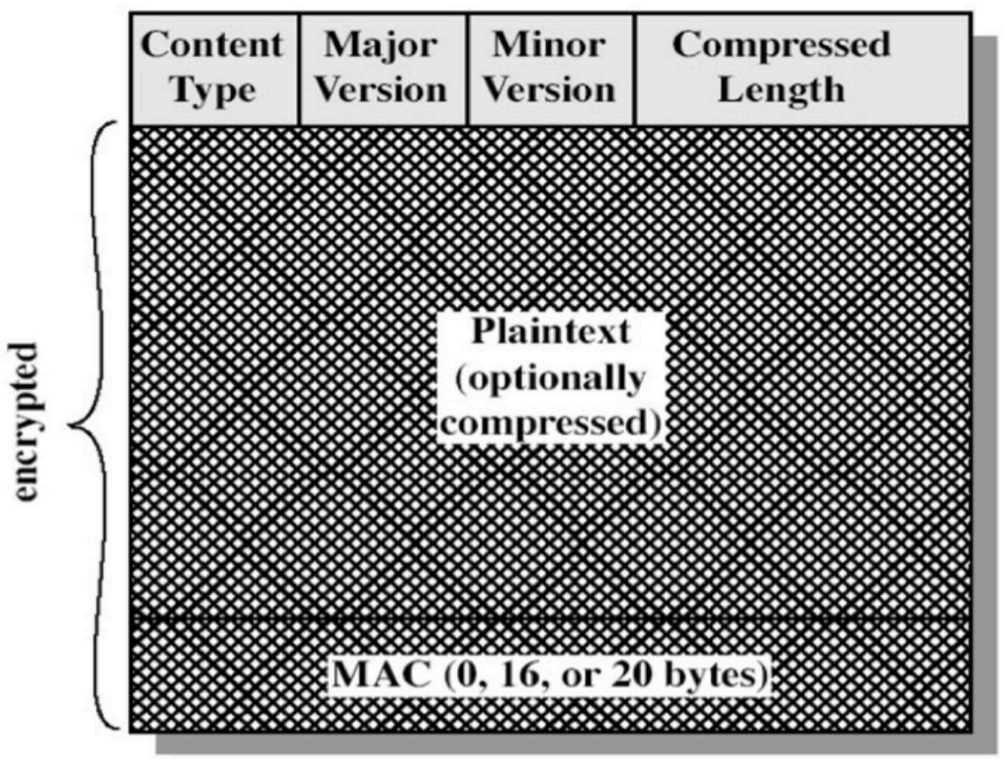
  - A message authentication code is then computed over the compressed data using a shared secret key.

SSL Record Protocol

- This is then appended to the compressed (or plaintext) block.
- The compressed message plus MAC are then encrypted using symmetric encryption.
- A number of different encryption algorithms are permitted.
- The final step is to prepend a header with the following :
  - **Content type (8 bits)** - The higher layer protocol used to process the enclosed fragment.
  - **Major Version (8 bits)** - Indicates major version of SSL in use. For SSLv3, the value is 3.
  - **Minor Version (8 bits)** - Indicates minor version in use. For SSLv3, the value is 0.
  - **Compressed Length (16 bits)** - The length in bytes of the compressed (or plaintext) fragment.

SSL Record Protocol

- The “**content type**” above is one of four types;
    - the three higher level protocols given that make use of the SSL record, and a fourth known as “application data”.
- The first three are described next as they are SSL specific protocols.



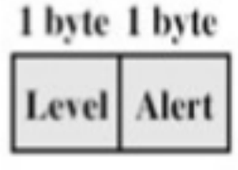
Change Cipher Spec Protocol

- This consists of a single message which consists of a single byte with the value 1.

- This is used to cause the pending state to be copied into the current state which updates the cipher suite to be used on this connection.

Alert Protocol

- This protocol is used to convey SSL-related alerts to the peer entity.
- It consists of two bytes the first of which takes the values 1 (warning) or 2 (fatal).
- If the level is fatal SSL immediately terminates the connection.
- The second byte contains a code that indicates the specific alert and describes the error.



Alert Protocol

- **Warning (level = 1):**

- This Alert has no impact on the connection between sender and receiver.

- Some of them are:
  - **Bad certificate:** When the received certificate is corrupt.
  - **No certificate:** When an appropriate certificate is not available.
  - **Certificate expired:** When a certificate has expired.
  - **Certificate unknown:** When some other unspecified issue arose in processing the certificate, rendering it unacceptable.
  - **Close notify:** It notifies that the sender will no longer send any messages in the connection.

Alert Protocol

- **Fatal Error (level = 2):**

- This Alert breaks the connection between sender and receiver.

- Some of them are :
  - **Handshake failure:** When the sender is unable to negotiate an acceptable set of security parameters given the options available.

  - **Decompression failure**: When the decompression function receives improper input.

  - **Illegal parameters:** When a field is out of range or inconsistent with other fields.

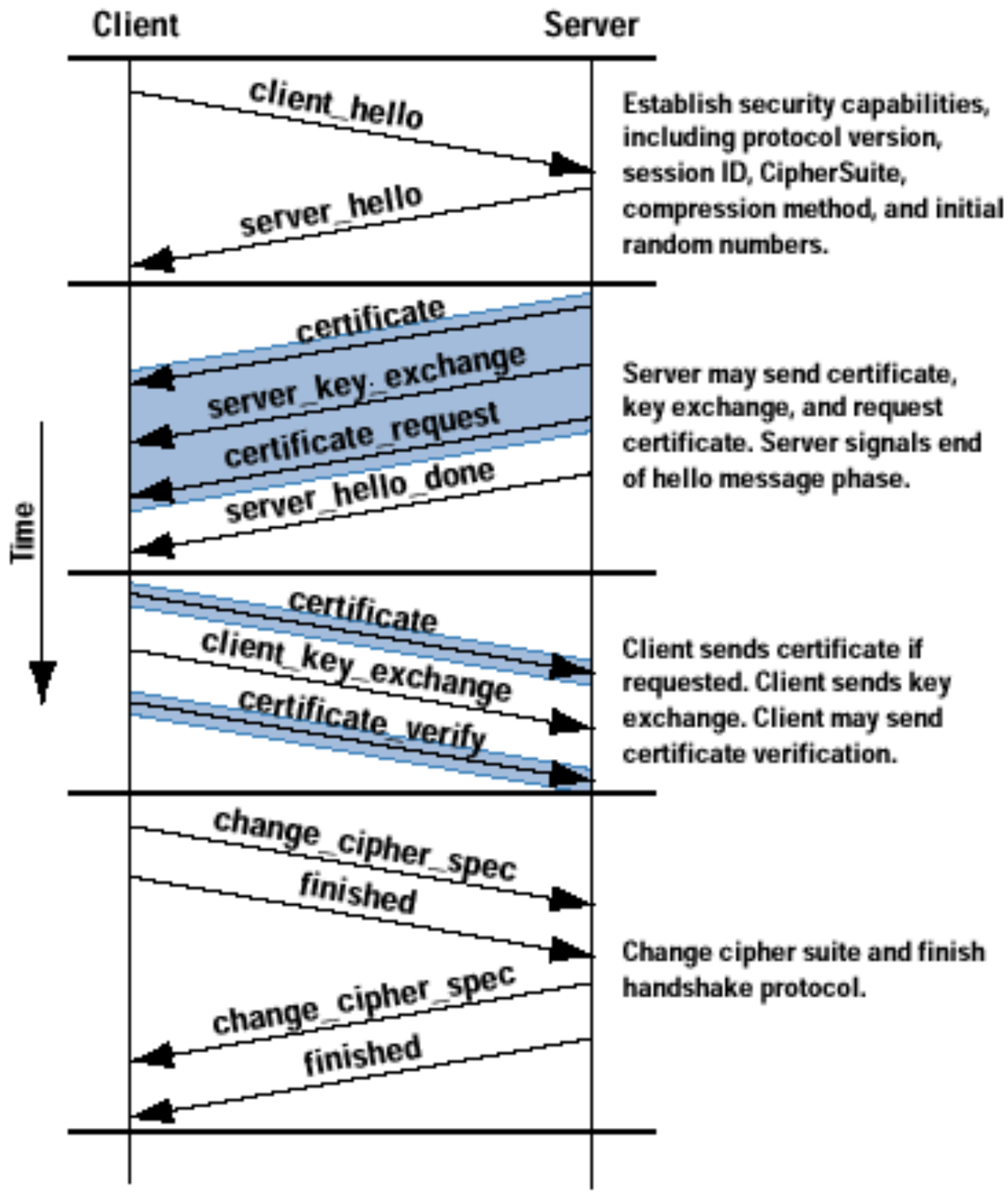
  - **Bad record MAC:** When an incorrect MAC was received.

  - **Unexpected message:** When an inappropriate message is received.

SSL Handshake

- This is the most complex part of SSL
- It allows the server and client to authenticate each other
- negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record.
- This protocol is used before any application data is sent.
- It consists of a series of messages exchanged by the client and server

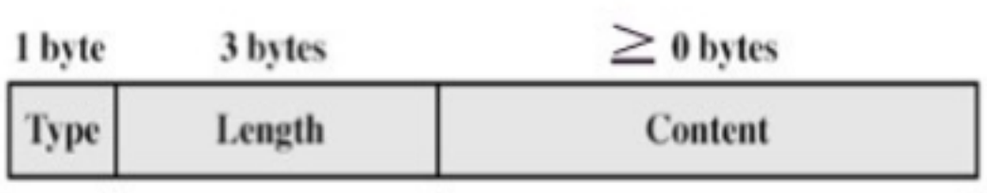
SSL Handshake



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

SSL Handshake

- Each message has three fields:
  - Type (1 byte): Indicates one of 10 messages such as “hello request”.
  - Length (3 bytes): The length of the message in bytes.
  - Content(≥ 0 byte): The parameters associated with this message such version of SSL being used.



SSL Handshake

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

SSL Handshake

- The Handshake Protocol consists of four phases:
  - Phase 1 :
    - Establish security capabilities including protocol version, session ID, cipher suite, compression method and initial random numbers.
    - This phase consists of the **client hello and server hello** messages

SSL Handshake

- The "client hello" message includes :
    - Version: client's SSL version number - The highest SSL version understood by client
    - Random: 32-bit timestamp and 28 byte nonce.
    - Session ID: A variable length session identifier.
    - CipherSuite: List of crypto algorithms supported by client in decreasing order of preference. Both key exchange and CipherSpec (this includes fields such as CipherAlgorithm, MacAlgorithm, CipherType, HashSize, Key Material and IV Size) are defined.
    - Compression Method: List of methods supported by client

SSL Handshake

- The server responds with a "server hello" message which includes :
    - server's SSL version number,
    - cipher settings,
    - session-specific data,
    - an SSL certificate with a public key and
    - other information that the client needs to communicate with the server over SSL.

SSL Handshake

- **Phase 2 :**

- Server may send **certificate**, **key exchange**, and **request certificate**

- it also signals end of hello message phase(**Server_hello_done**).

- The certificate sent is one of a chain of X.509 certificates.

- The server key exchange is sent only if required.

- A certificate may be requested from the client, if needed, by certificate request.

SSL Handshake

- Upon receipt of the server_done message, the client should verify that the server provided a valid certificate, if required, and check that the server_hello parameters are acceptable.

- The client verifies the server's SSL certificate from CA (Certificate Authority) and authenticates the server.

- If the authentication fails, then the client refuses the SSL connection and throws an exception.

- If the authentication succeeds, then proceed to next step.

SSL Handshake

- **Phase 3:**

- If all is satisfactory, the client sends one or more messages back to the server.

- The client sends **certificate** if requested (if none available then it sends a no certificate alert instead).

- Next the client sends **client key exchange message** .

- Finally, the client may send **certificate verification**.

SSL Handshake

- The client creates a session key, encrypts it with the server's public key and sends it to the server.

- If the server has requested client authentication (mostly in server to server communication), then the client sends his own certificate to the server.

- The server decrypts the session key with its private key and sends the acknowledgement to the client encrypted with the session key.

SSL Handshake

- **Phase 4:**
  - **Change cipher suite** and finish handshake protocol.
  - The secure connection is now setup and the client and server may begin to exchange application layer data.

To Read

- Chapter 17.1-17.2 from Forouzan text book