session identifier
An arbitrary byte sequence chosen by the client and server.

master secret
48-byte secret shared between the client and server.

attributes such as the hash size, MAC algorithm (such as MD5 or SHA), and also defines cryptographic attributes such as the hash size.

cipher spec
Specified the bulk data encryption algorithm (such as null, DES, etc.) and a MAC algorithm (such as MD5 or SHA).

peer certificate
X509 v3 [X509] certificate of the peer. This element of the state may be null.

session identifier
An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
When a block cipher in CBC mode is used, an initialization vector (IV) is
chosen by the server and client for each connection.

**Initialization Vectors**

- **Server Write MAC Secret**
  - The secret used in MAC operations on data written by the server.
- **Client Write MAC Secret**
  - The secret used in MAC operations on data written by the client.

**Server Write Key**

- The bulk cipher key for data encrypted by the server and decrypted by the
  client.

**Client Write Key**

- The bulk cipher key for data encrypted by the client and decrypted by the
  server.

**Initialization Vectors**

- **Server**
  - Byte sequences that are chosen by the server and client for each connection.
- **Client**
  - Random sequences generated by client and server.
Always send
Indicates optional or situation-dependent messages that are not

Application Data

Client

Server

Cert

Server

Server certificate

Client certificate

Client certificate

Any other

Change cleartext

Change cleartext

Change cleartext

Change cleartext
The client hello message includes a random structure, which is used later in the protocol.

```
struct {
  uint32 gmt_unix_time;
  opaque random_bytes[28];
} Random;
```

gmt_unix_time

The current time and date in standard UNIX 32-bit

random_bytes

28 bytes generated by a secure random number

define additional requirements.

Protocol; higher level or application protocols may
are not required to be set correctly by the basic SSL
format according to the sender's internal clock. Clocks
The current time and date in standard UNIX 32-bit

gmt_unix_time

Random

opaque random_bytes[28];

uint32 gmt_unix_time;

struct

which is used later in the protocol.

The client hello message includes a random structure,
The server processes the client hello message and responds with either a handshake_failure alert or server hello message.

```
struct ServerHello {
    ProtocolVersion protocol_version;
    CompressionMethod compression_method;
    CipherSuite cipher_suite;
    SessionID session_id;
    Random random;
    ProtocolVersion server_version;
}
```

The server responds with either
If RSA is being used for key agreement and authentication, the client generates a 48-byte pre-master secret, encrypts it with server public key, and sends it encrypted with server public key. The 48 bytes = 384 bits. This random value is generated by the client and is used to generate the master secret.
master_secret

\[\text{MD5}(\text{pre-master-secret} + \text{SHA}(\text{CCC} + \text{pre-master-secret} + \text{serverHello.random} + \text{ClientHello.random} + \text{ServerHello.random})) + \text{MD5}(\text{pre-master-secret} + \text{SHA}(\text{BB} + \text{pre-master-secret} + \text{serverHello.random} + \text{ClientHello.random} + \text{ServerHello.random})) + \text{MD5}(\text{pre-master-secret} + \text{SHA}(\text{AA} + \text{pre-master-secret} + \text{serverHello.random} + \text{ClientHello.random} + \text{ServerHello.random})) \]

= \text{master-secret}
To generate the key material, compute

\[
\text{MasterSecret} + \text{ServerHello.random} + \text{ClientHello.random} + \text{SHA}('A' + \text{MasterSecret} + \text{ServerHello.random} + \text{ClientHello.random}) + \text{MD5(MasterSecret)} + \text{ServerHello.random} + \text{ClientHello.random} + \text{SHA}('BB' + \text{MasterSecret} + \text{ServerHello.random} + \text{ClientHello.random}) + \text{MD5(MasterSecret)} + \text{ServerHello.random} + \text{ClientHello.random} + \text{SHA}('CC' + \text{MasterSecret} + \text{ServerHello.random} + \text{ClientHello.random}) + \text{MD5(MasterSecret)} = \text{KeyBlock}
\]

until enough output has been generated.

...
Then the key block is partitioned as follows.

Then the key block is partitioned as follows.

server-write-key[cipher_spec.key_material]
client-write-key[cipher_spec.key_material]
server-write-MAC-secret[cipher_spec.hash_size]
client-write-MAC-secret[cipher_spec.hash_size]
The MAC is generated as:

$$\text{hash (MAC-write-secret + pad_1 + } \text{seq-num + length + content + pad_2 + pad_3 + } \text{seq-num)}$$

The hashing algorithm derived from the cipher suite.

The sequence number for this message.

The character 0x36 repeated the same number of times.

The character 0x36 repeated 48 times for MD5 or 40 times for SHA.