Web Security

Protocols:
SSL & TLS

Outline

- Secure Sockets Layer (SSL)
- Transport Layer Security (TLS)

Web Security Threats

One way to group those threats is in terms of passive and active attacks.

Passive attacks

- It include eavesdropping on network traffic between browser and server and gaining access to information on a web site that is supposed to be restricted.

Active attacks

- It include impersonating another user, altering messages in transit between client and server, and altering information on a web site.

A Comparison of Threats on The Web

	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

Web Security Threats

Another way to classify Web security threats is in terms of the location of the threat:

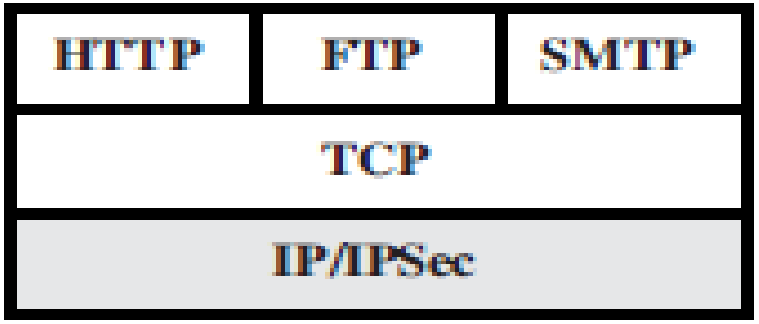
Web server, Web browser, and network traffic between browser and server.

- Issues of server and browser security fall into the category of computer system security.
- Issues of traffic security fall into the category of network security.

Web Traffic Security Approaches

A number of approaches to providing Web security are possible.

- One way to provide Web security is to use IP security (IPsec).



(a) Network level

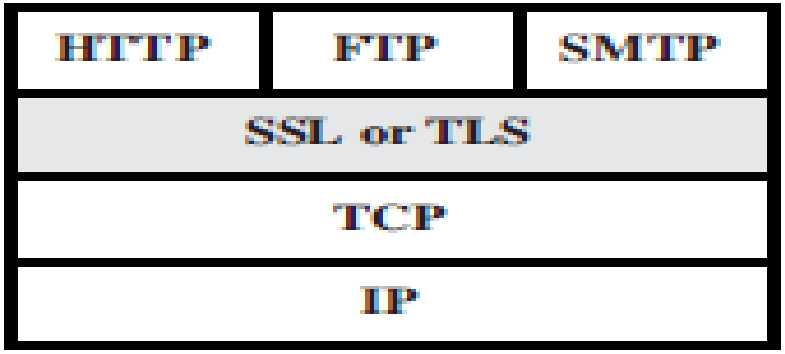
Web Traffic Security Approaches

- Another relatively general-purpose solution is to implement security just above TCP.

At this level, there are two implementation choices.

- SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications.
- Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers.

Web Traffic Security Approaches

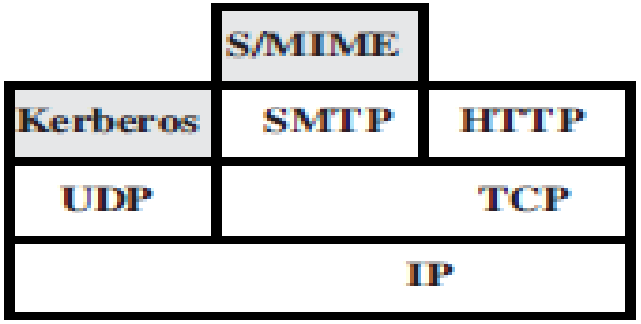


(b) Transport level

Web Traffic Security Approaches

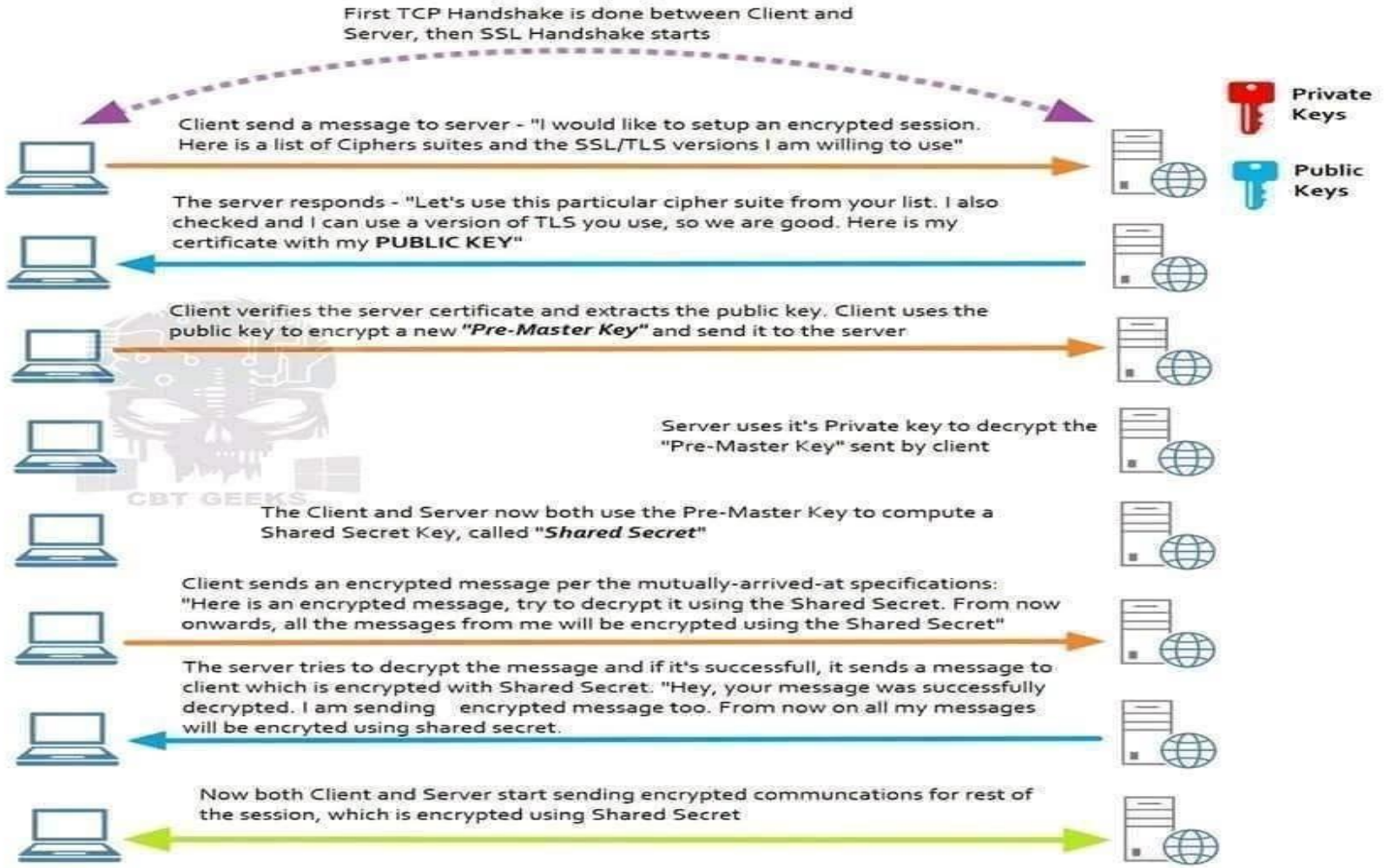
- Application-specific security services are embedded within the particular application.

Advantage: The service can be tailored to the specific needs of a given application.



(c) Application level

The SSL / TLS Handshake Process



Secure Socket Layer (SSL)

- Invented by Phil Karlton (CMU Ph.D.) and others at Netscape.
- It is a secure data exchange protocol providing
  - Privacy between two Internet applications.
  - Authentication of server (authentication of browser optional).
- It is a two layers protocols
  - ✓ SSL Handshake, Change Cipher Spec and Alert Protocols
    - They are used in the management of SSL exchanges, such as Negotiates symmetric encryption protocol, authentication, etc.
  - ✓ SSL Record Protocol
    - It is used to provide basic security services to various higher layer protocols, such as Packs/unpacks records, performs encryption/decryption.

Secure Socket Layer (SSL)

- Does not provide non-repudiation.

Two important concepts of SSL are:

Session:

- It is an association between a client and a server.
- It is created by the Handshake Protocol and defines cryptographic security parameters which can be shared among multiple connections.
- There may be multiple sessions between parties.

Connection:

- It makes use of TCP to provide a reliable end-to-end service.
- Every connection is associated with one session.
- There may be multiple secure connections between any pair of parties.

Secure Socket Layer (SSL)

- A session is defined by the following parameters:

Session Identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

Peer Certificate: X.509.v3 of the peer.

Compression Method: Algorithm to compress data.

Cipher Spec: Encryption algorithm, hash algorithm and cryptographic attributes such as hash size.

Master Secret: 48 bytes secret shared b/w client and server.

Is resumable: A flag to indicate whether the session can be used to initiate new connections.

- A connection is defined by the following parameters:

Server and client random: random bytes chosen by the server and client for each connection.

Secure Socket Layer (SSL)

Server write MAC secret: Secret key used in MAC operation

Client write MAC secret: Secret key used in MAC operation

Server write Key: Conventional encryption key

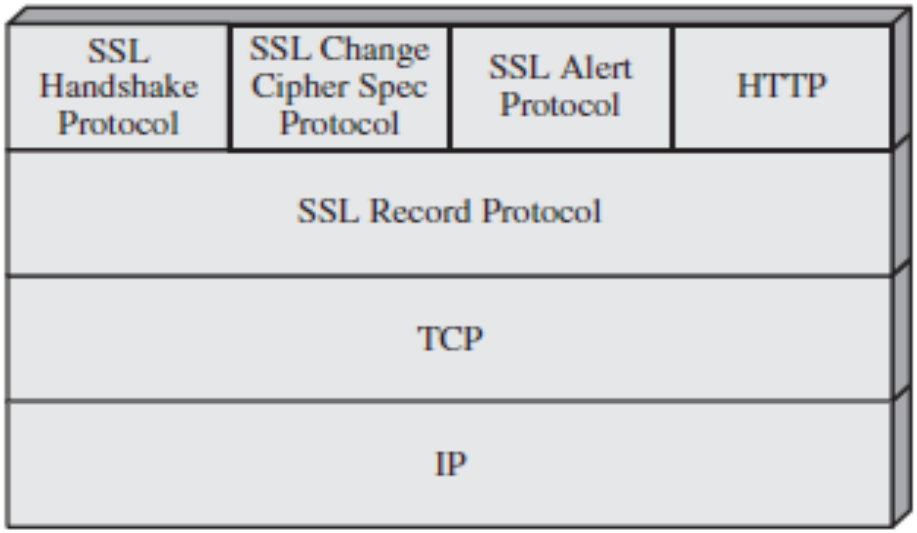
Client write Key: Conventional encryption key

Initialization vector: Is needed when block cipher in CBC mode is used.

Sequence numbers:

Each party maintains separate sequence numbers for transmitted and received messages for each connection.

SSL (Secure Socket Layer)



SSL Protocol Stack (Source: William Stalling)

SSL Record Protocol

- Record protocol provides two services for connections:

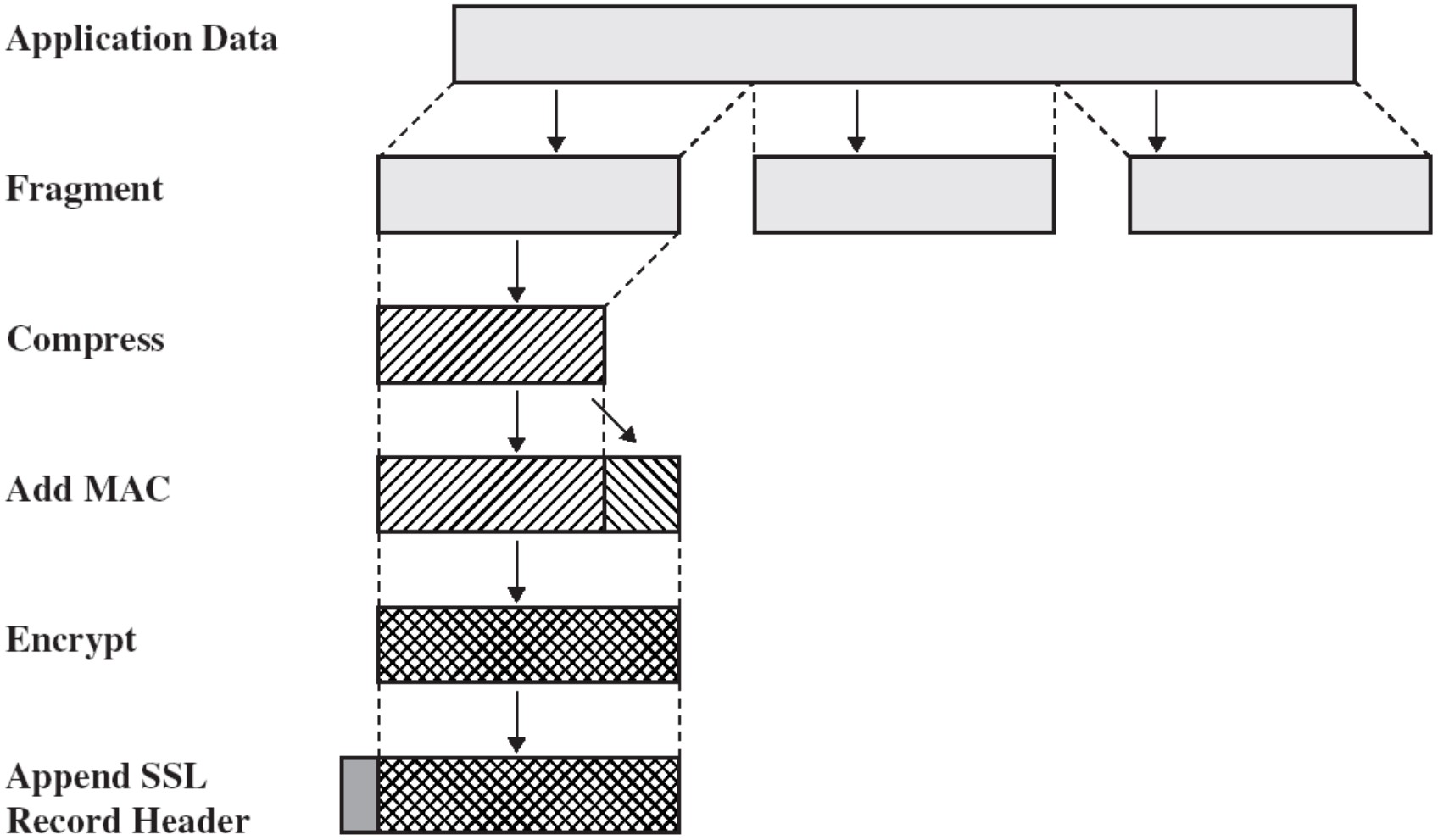
Confidentiality:

Block Cipher		Stream Cipher	
Algorithm	**Key Size**	**Algorithm**	**Key Size**
AES	128, 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Message Integrity: Message Authentication Code (MAC)

```
hash(MAC_write_secret || pad_2||
     hash(MAC_write_secret || pad_1||seq_num ||
     SSLCompressed.type || SSLCompressed.length ||
     SSLCompressed.fragment))
```

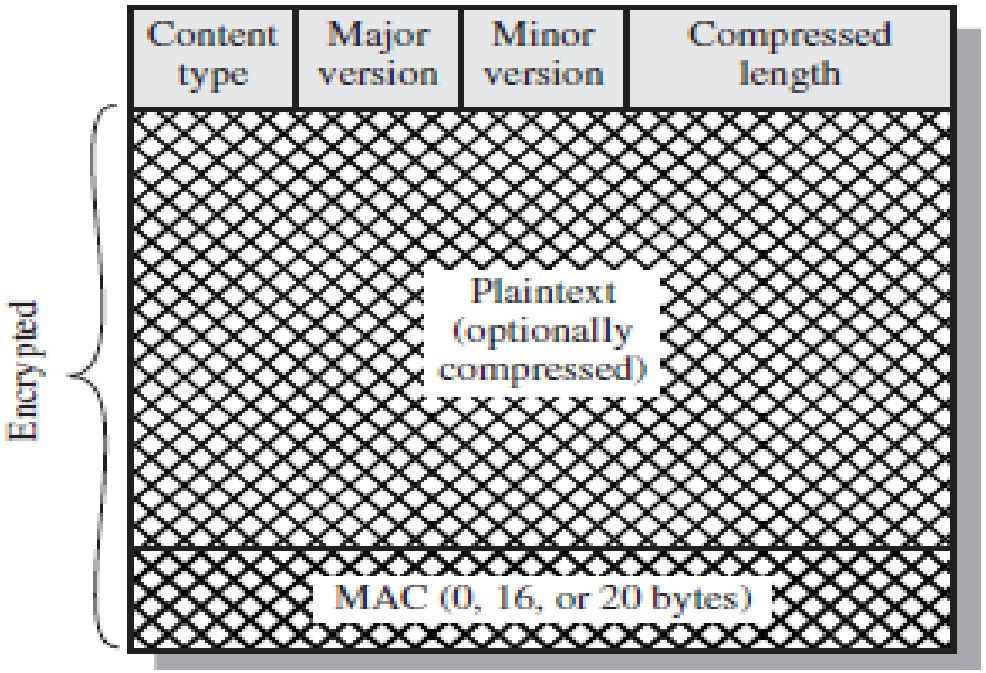
SSL Record Protocol Operation



SSL Record Protocol

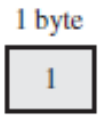
- The final step of SSL Record Protocol processing is to append a header which consists of the following fields:
  - **Content Type (8 bits):** The higher layer protocols, such as change_cipher_spec, alert, handshake and application data.
  - **Major Version (8 bits):**
  - **Minor Version (8 bits):** For SSL.v3, the value is 0.
  - **Compressed Length (16 bits):** Maximum value is 2^{14} + 2048.

SSL Record Format

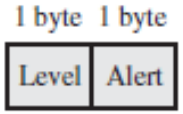


SSL Specific Protocol

- **Change Cipher Spec Protocol:**



- **Alert Protocol:** Levels (1) warning or (2) fatal



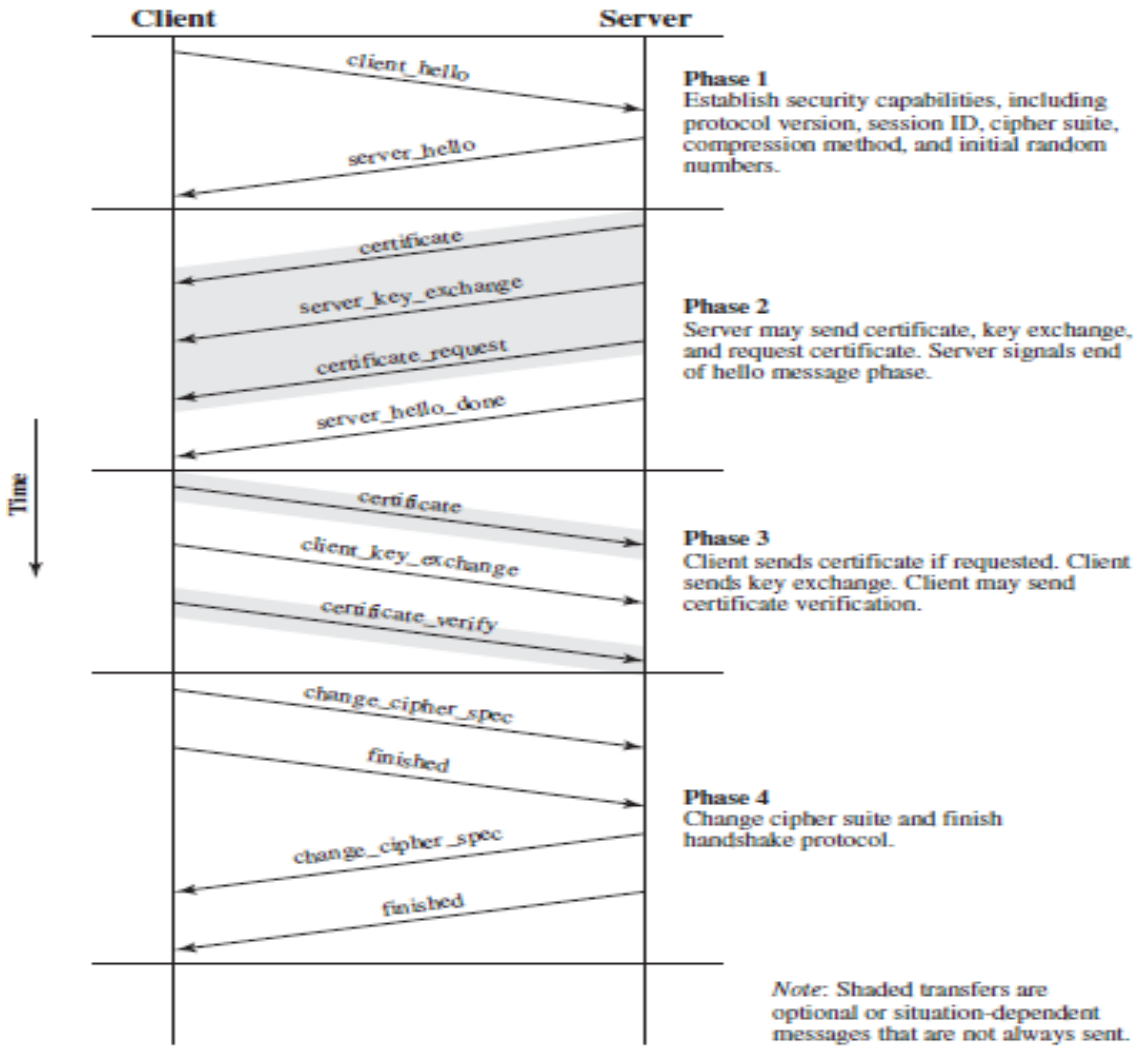
- **Handshake**

  - Type indicates one of 10 messages.
  - Length indicates message in bytes.
  - Content indicates parameters associated with the message.

SSL Handshake Protocol Message Types

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 Protocol



Initial Exchange to Setup a Logical Connection b/w peers

The exchange can be viewed as having four phases.

Phase 1: Establish Security Capabilities:

- **Version:** Highest SSL version supported by the client.
- **Random :**

  Client generated, consisting of a 32-bit timestamp and 28 bytes random number.

  These values serve as nonces and are used during key exchange to prevent replay attacks.
- **Session ID:**

  A variable length session identifier.

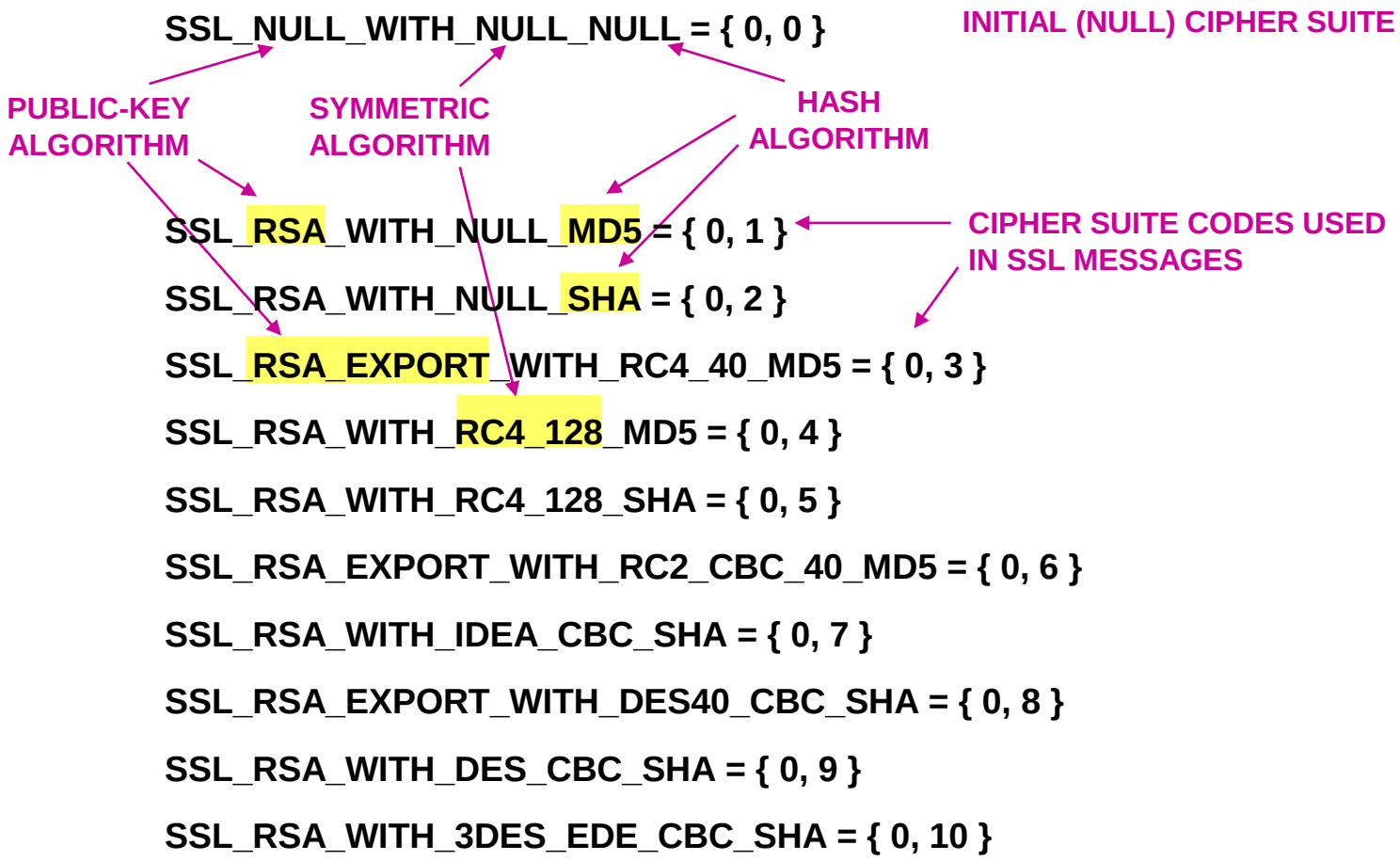
  A nonzero value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session.

  A zero vale indicate that the client wishes to establish a new connection on a new session.
- **Cipher Suite:**
- **Compression Method:** A list of compression methods the client supports.

Cipher Suite

- For public-key, symmetric encryption and certificate verification we need
  - public-key algorithm
  - symmetric encryption algorithm
  - message digest (hash) algorithm
- This collection is called a cipher suite
- SSL supports many different suites
- Client and server must decide on which one to use
- The client offers a choice; the server picks one

Cipher Suites



Initial Exchange to Setup a Logical Connection b/w peers

Phase 2: Server Authentication and Key Exchange:

- The server starts this phase by sending its certificates, if it needs to be authenticated.
- A **server_key_exchange** is not required

(1) If server has sent a certificate with fixed Diffie-Hellman parameters.

(2) RSA key is to be exchanged.

- A **server_key_exchange** is required

(1) If Anonymous Diffie-Hellman, Ephemeral Diffie-Hellman or Fortezza.

(2) RSA key exchange, in which the server is using RSA but has a signature-only RSA key.

Initial Exchange to Setup a Logical Connection b/w peers

- **Certificate_request_message**

  It includes two parameters:

  certificate_type and **certificate_authorities** that include public key algorithm along with its use and list of acceptable certificate authorities, respectively.

- **Server_done_message**

  It is sent by the server to indicate the end of the server hello and associated messages.

Phase 3: Client Authentication and Key Exchange

✓ Upon receiving **server_done_message,** the client should verify server's certificate and check that server hello parameters are acceptable.

Initial Exchange to Setup a Logical Connection b/w peers

- If server has requested a certificate, the client sends a certificate.

Client_key_exchange

- The content of the message depends on the type of key exchange.

RSA: The client generates 48 bytes pre-master secret and encrypts with the server's public key received through certificate or temporary RSA key from a **server_key_exchange** message.

Certificate_verify

- The client may send this message to provide explicit verification of a client certificate.

```
CertificateVerify.signature.md5_hash=
     MD5(master_secret || pad_2 || MD5(handshake_messages ||
         master_secret || pad_1));
CertificateVerify.signature.sha_hash=
     SHA(master_secret || pad_2 || SHA(handshake_messages ||
         master_secret || pad_1));
```

Initial Exchange to Setup a Logical Connection b/w peers

Phase 4: Finish

- The client sends a **change_cipher_spec** message and copies the pending CipherSpec into current CipherSpec.
- Next client immediately sends the finished message under the new algorithm, keys and secrets.
- **It verifies that the key exchange and authentication processes were successful.**

```
MD5(master_secret || pad2 || MD5(handshake_messages ||
    Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages ||
    Sender || master_secret || pad1))
```

Generation of Master Secret and Cryptographic Parameters

- **Premaster secret**
  - Created by client; used to “seed” calculation of encryption parameters.
  - Very simple: 2 bytes of SSL version + 46 random bytes.
  - Sent encrypted to server using server’s public key.
- **Master secret**
  - Generated by both parties from premaster secret and random values generated by both client and server.
- **Key material**
  - Generated from the master secret and shared random values.
- **Encryption keys**
  - Extracted from the key material.

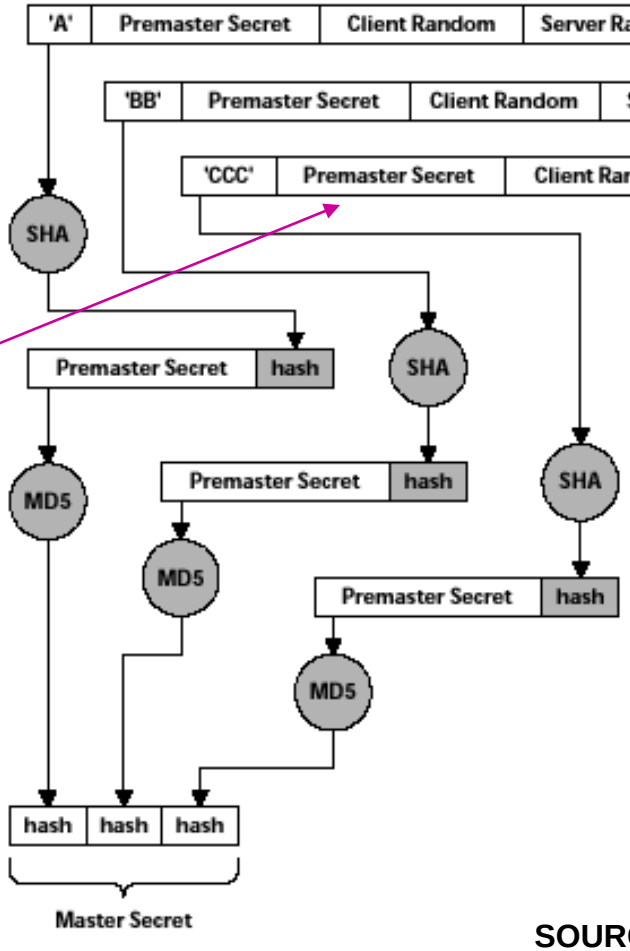
Forming the Master Secret

SERVER'S PUBLIC KEY IS SENT BY SERVER IN ServerKeyExchange

CLIENT GENERATES THE PREMASTER SECRET

ENCRYPTS WITH PUBLIC KEY OF SERVER

CLIENT SENDS PREMASTER SECRET IN ClientKeyExchange



SENT BY SERVER IN ServerHello

SENT BY CLIENT IN ClientHello

MASTER SECRET IS 3 MD5 HASHES CONCATENATED TOGETHER = 384 BITS

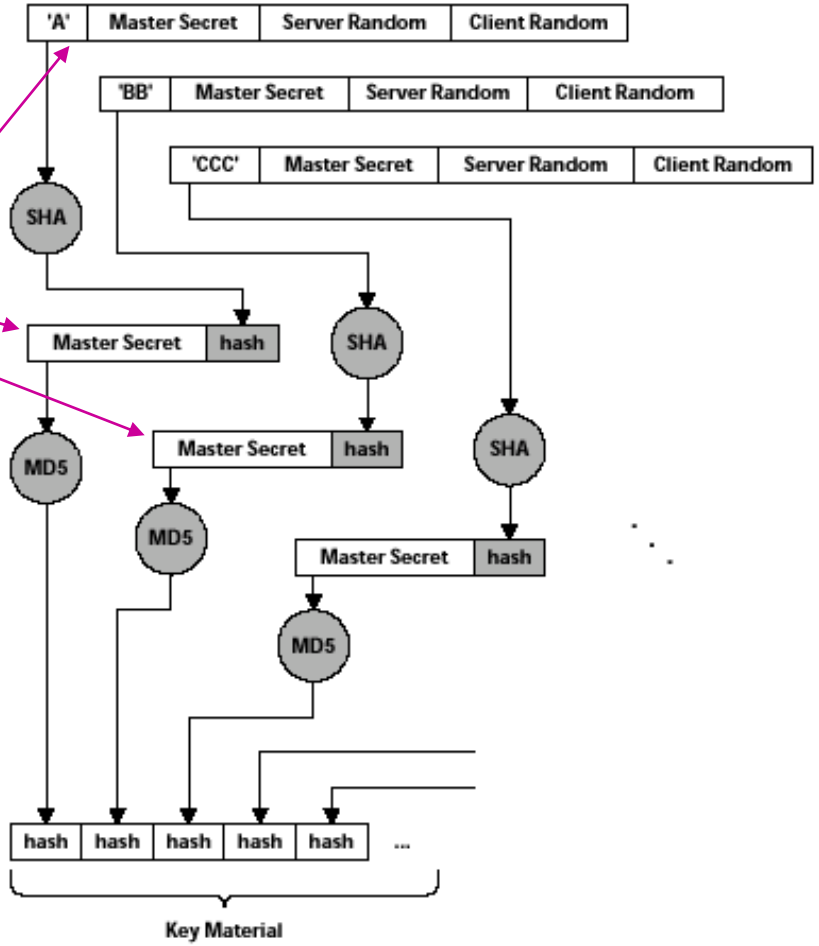
Note: "type strings" are included in calculations to prevent using a value from one computation as part of another

SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

Forming the Key Material

JUST LIKE FORMING THE MASTER SECRET

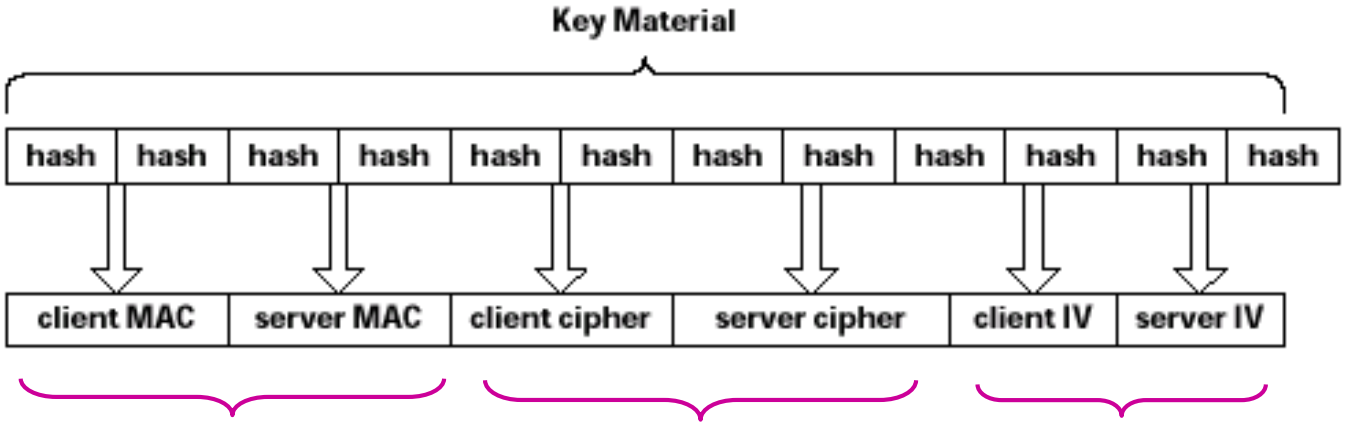
EXCEPT THE MASTER SECRET IS USED HERE INSTEAD OF THE PREMASTER SECRET



Note: "type strings" are included in calculations to prevent using a value from one computation as part of another

SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

Obtaining Keys from the Key Material



Note: These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters.

SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

SSL (Secure Sockets Layer)

Some payment services using SSL:

- Credit Card Network 

- Secure-Bank.Com 

- Web-Charge 

- SecureTrans 

Transport Layer Security (TLS)

- SSL is so important that it was adopted by the Internet Engineering Task Force (IETF) to produce an Internet version of it.
- TLS Protocol 1.0 (RFC 2246).
- TLS is very similar to SSLv3 but it does not interoperate.
- Goals
  - Separate record and handshaking protocols
  - Extensibility (add new cipher suites easily)
  - Efficiency (minimize network activity)

TLS

- **Version Number:** Major Version is 3 and the Minor Version is 1.
- **MAC:** There are two differences

  1. HMAC algorithm 2. Padding bytes are XOR with secret key
- **Pseudorandom Function:** It is used to expand secrets into blocks of data for purpose of key generation or validation.
- **Alert Codes:** Additional alerts codes are defined.
- **Client Certificate Types:** It doesn't include the Fortezza scheme.
- **Certificate Verify Message:**

  Hashes are calculated only over handshake messages.
- **Finished Message:**
  - It is a hash based on master secret, the previous handshake message, and a label that identifies client or server.

```
PRF(master_secret,finished_label,MD5(handshake_messages)
    SHA-1(handshake_messages))
```

TLS

- **Cryptographic Computation:**

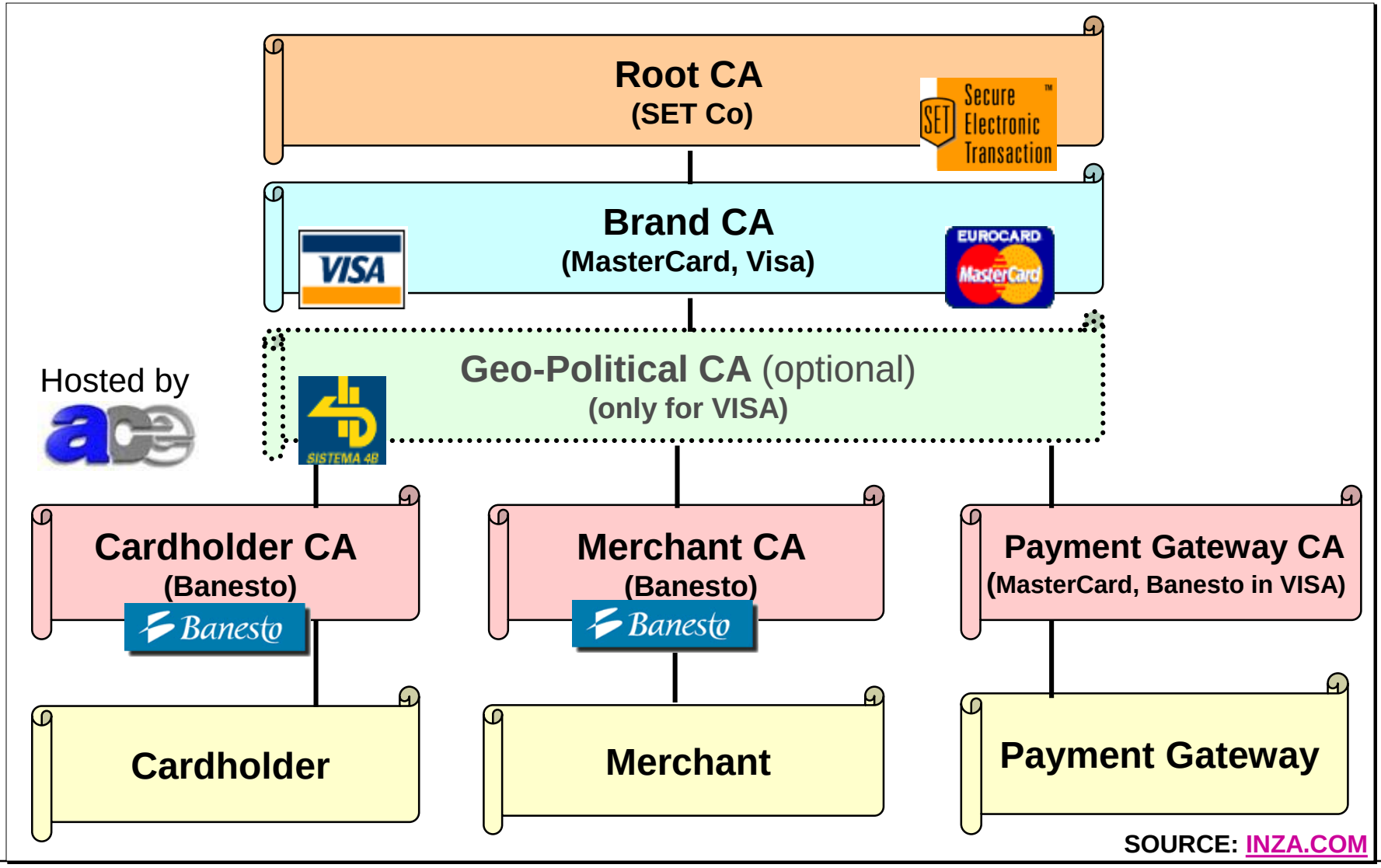
In it master secret (48 bytes pseudorandom output) is calculated as follows:

master secret = PRF(pre_master_secret, ClientHello.random||ServerHello.random)

key_block = PRF(master_secret, SecurityParameters.server_random || SecurityParameters.client_random)

- **Padding:**
  - ✓ It can be any amount (e.g., 1, 9, 17 and so on, up to 249 bytes) that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes.
  - ✓ A variable length padding is used to frustrate attacks.

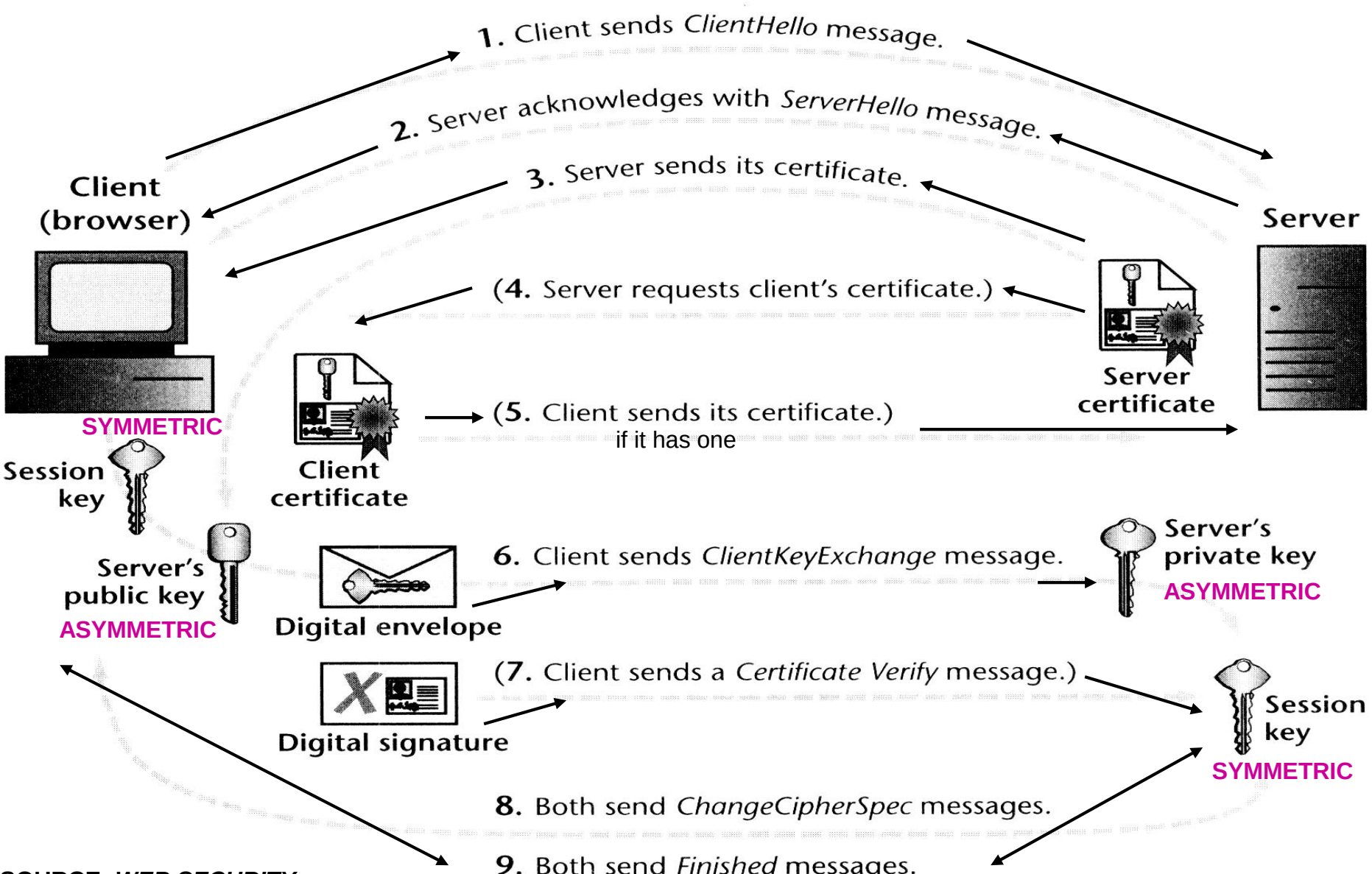
SET Certificate Hierarchy



Major Ideas

- SSL, TLS are secure message protocols, not payment protocols
- SSL requires the vendor to have a certificate.
- SSL is secure against breaking of any one form of encryption.
- SET is a payment protocol.
- SET requires all parties to have certificates.

Secure Sockets Layer (SSL) Handshake



SOURCE: ***WEB SECURITY***