# SECURE SOCKET LAYER(SSL)/TRANSPORT LAYER SECURITY (TLS)

DR. O.I. ADELAIYE O.I.

# WHY SSL/TLS?

- Modification of message/traffic in transit
- Impersonation of users
  - Fake users
- Data forgery
  - Data created by an intruder are considered to be genuine
- Eavesdropping on the net
  - Loss of privacy

# WHY? WEB SECURITY REQUIREMENT

- Data Integrity
  - Ensures that the received data is the same as when sent by the sender
  - Using checksums, message authentication codes
- Confidentiality
  - Protection of data from unauthorized disclosure
  - Using encryption
- Authentication
  - The proof of the communicating entity is the one it pretends to be
  - Essential on the server side, optional on the client one
  - Using challenge-response, username/password, certificates, …
- Objectives of SSL/TLS
  - Allow two entities to authenticate
  - Establish session keys

# WEB SECURITY APPROACHES

- Question: where should we put the security mechanisms?
  - Under TCP: IPSec, between IP and TCP
    - Requires changes to OS, applications over TCP do not change
- Over TCP: SSL/TLS
  - Does not require changes to OS, applications over TCP need to change
  - TCP is a reliable service: SSL does not need timing out controls or data retransmission techniques

# HISTORY

- Secure Network Programming (SNP) API
  - Encapsulating sensitive information in a secure layer (1993)
  - The SNP project received the 2004 ACM software system award
- Secure Socket Layer (SSL)
  - Originally proposed by Netscape
  - SSL v1 contained loads of security flaws, never made public
  - SSL v2 was released in 1994 and implemented in Netscape Navigator 1.1 in 1995,
    - Had a number of security flaws which were pointed out by the experts
    - Microsoft version: Private Communication Technology (PCT)
  - SSL v3 was released in 1996
  - SSL v3 uses RSA which had a patent on it, it could not be standardised in its original form, so …
- Transport Layer Security (TLS)
  - Internet standard variation of SSL
  - TLS 1.0 was published in 1999 by the IETF

# BASIC PROTOCOL

- A client wants to securely connect to a server
  - At least authentication of the server as well as data integrity and confidentiality are required
- The server has a X.509 certificate from a trusted Certification Authority (CA)
  - The root CA is built (or manually added) into the web browser of the client
- The server returns its certificate when contacted by the client's web browser
- The web browser encrypts a random number using the public key of the certificate
  - The encrypted random number is sent to the server as a challenge
- Once the server responds correctly, a secure channel is established between the server and the web browser

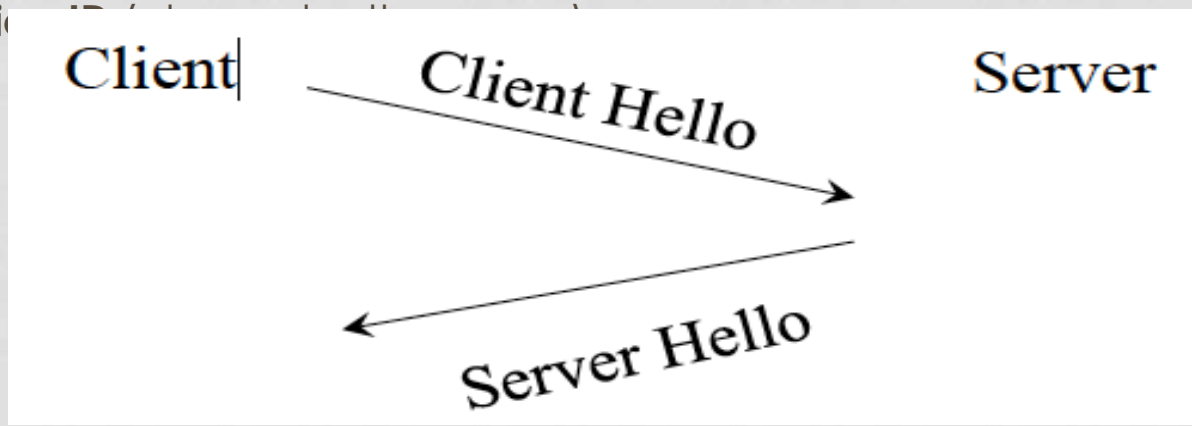
# SSL HANDSHAKE PROTOCOL

- Comprises 4 phases
  - **Phase 1** Establishes the capabilities of the client and server
  - **Phase 2** Server authentication and key exchange
  - **Phase 3** Client key exchange and optional client authentication
  - **Phase 4** Change Cipher Specification Protocol and Finish

# SSL HANDSHAKE PHASE 1

- The **client hello** message contains
  - The **versions of SSL** supported by the client
  - The **algorithms** supported by the client
  - Nonces to protect against replay (a **time stamp** and **random number**)
  - **Session ID** (initially set to **zero**)

- The **server hello** message contains
  - The **SSL version** chosen by the server (highest one)
  - The set of **algorithms** chosen by the server
  - Server nonces to stop replay (a **time stamp** and **random number**)
  - **Sessi**



# SSL HANDSHAKE PHASE 2

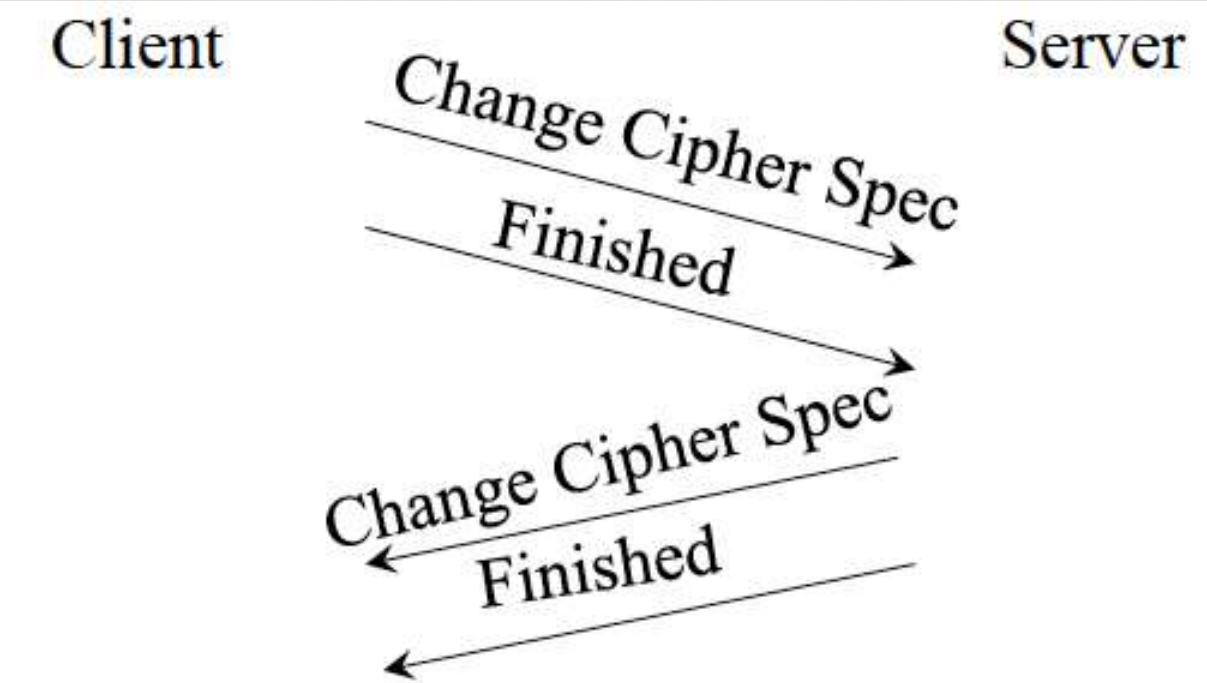
- The server sends to the client
  - Its X.509 public key **certificate** chain up to the root CA
  - If the server supports client authentication, a **client certificate request**
  - If **Diffie Helman** key exchange is being used, the key parameters
  - The Server Hello Done message

# SSL HANDSHAKE PHASE 3

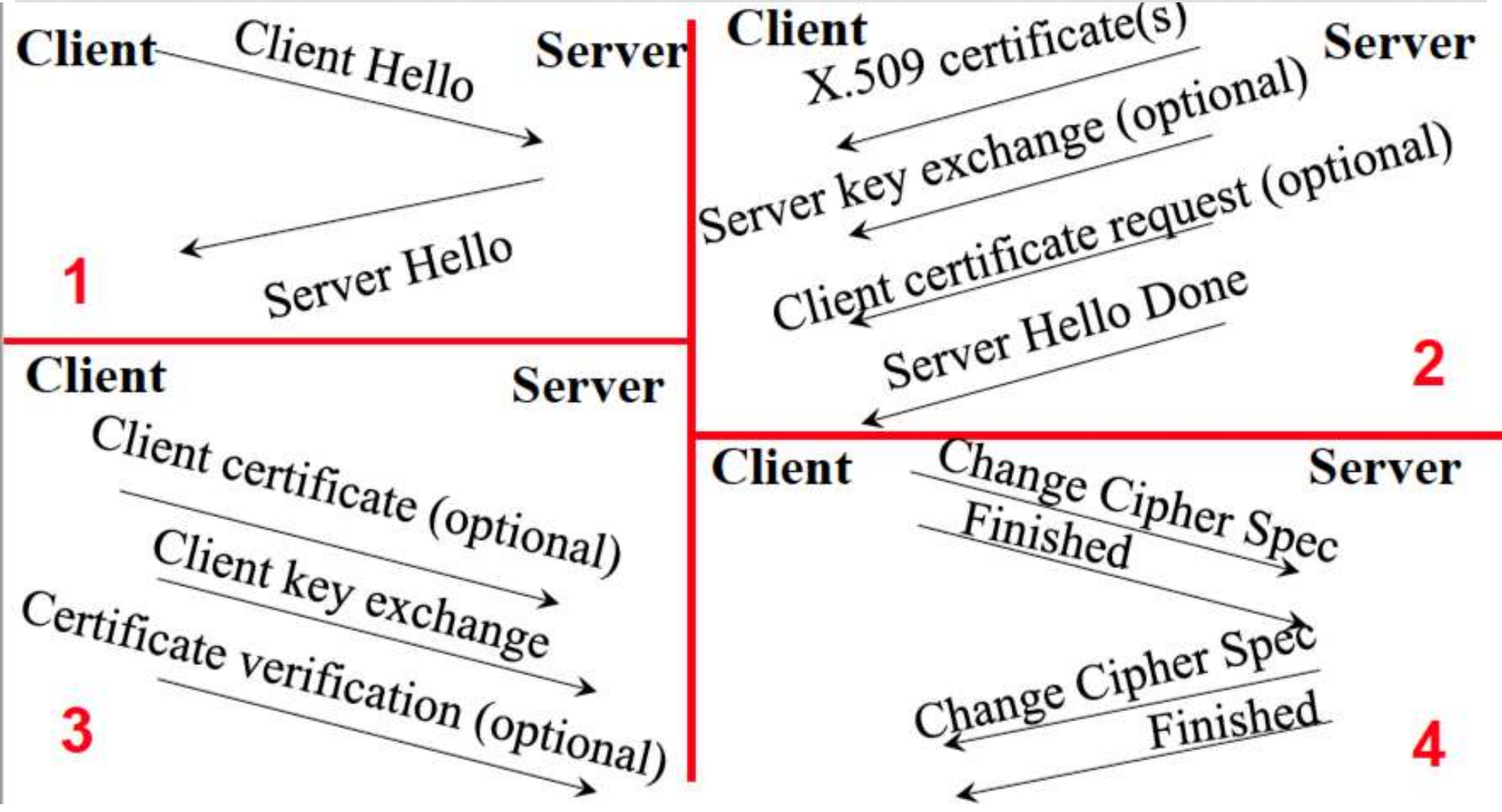
- The client sends to the server
  - Key exchange parameters encrypted with server's public key
    - Diffie-Helman: the client sends **its half** of the secret parameter
    - RSA: the client sends a **48 bytes pre-master** secret
- Optionally its **public key certificate** and a verification message (signature on the hash of the exchanged messages)

# SSL HANDSHAKE PHASE 4

- Phase 4 The client tells the server to change ciphers to the agreed set and then it is finished
- The server sends a keyed hash of all the handshake messages
- The server agrees to both

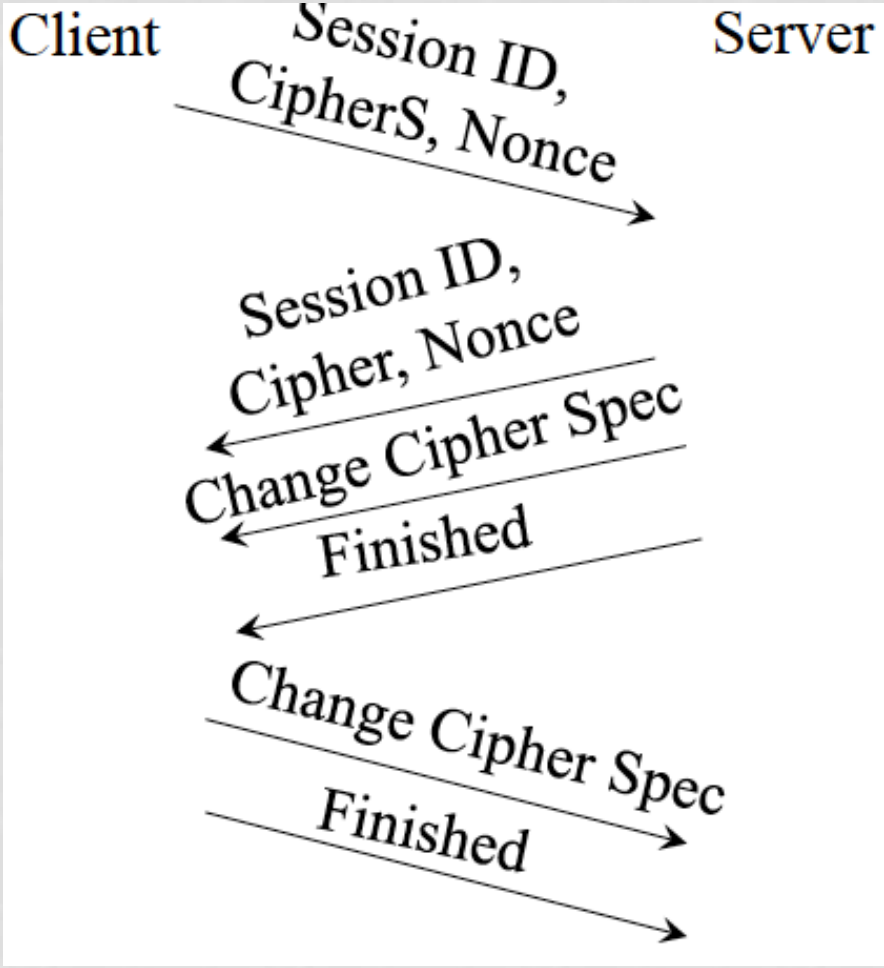


# SSL HANDSHAKE SUMMARY



# RESUMING SESSIONS

- In Phase 1 the Session ID is set to the Session ID previously returned by the server at start of session
- If the server accepts, it proceeds straight to the Finished phase
- Nonces are exchanged, so the session keys are unique



# CALCULATION OF SHARED SECRET

- The pre-master secret from Phase 3 is concatenated with the client and server random numbers to provide a master secret, which is then hashed to produce
  - A shared secret for message authentication codes (MAC) created by the client for data integrity
  - A shared secret for MACs created by the server
  - A symmetric encryption key for messages sent by the client
  - A symmetric encryption key for messages sent by the server
- The reason the client and server use different keys is to make it more difficult to break the messages
- In SSLv2 there were 2 keys, one on each side for both encryption and MACs

# CLIENT AUTHENTICATION

- Theoretically, SSL/TLS allows client authentication by sending the client certificate
- In practice, the client sends their username/password to the server over the established session

# SUMMARY: SSL SECURITY SERVICE

- Data Integrity
  - A keyed hash (MAC) is appended to the message
- Confidentiality
  - Symmetric encryption of the message and MAC
- Authentication
  - The server is always authenticated
  - The client optionally sends its certificate to the server

# CRYPTO ALGO. SUPPORTED BY SSL

- Encryption
  - Stream Encryption: RC-4 (40 and 128 bits)
  - Block Encryption: DES (40 and 56 bits), Triple DES (168 bits), or IDEA (128 bits)
- Message Authentication Codes
  - MD5 or SHA1
- Key Generation
  - Diffie-Helman and RSA (Ron Rivest, Adi Shamir, and Leonard Adleman)

# SSL MESSAGES

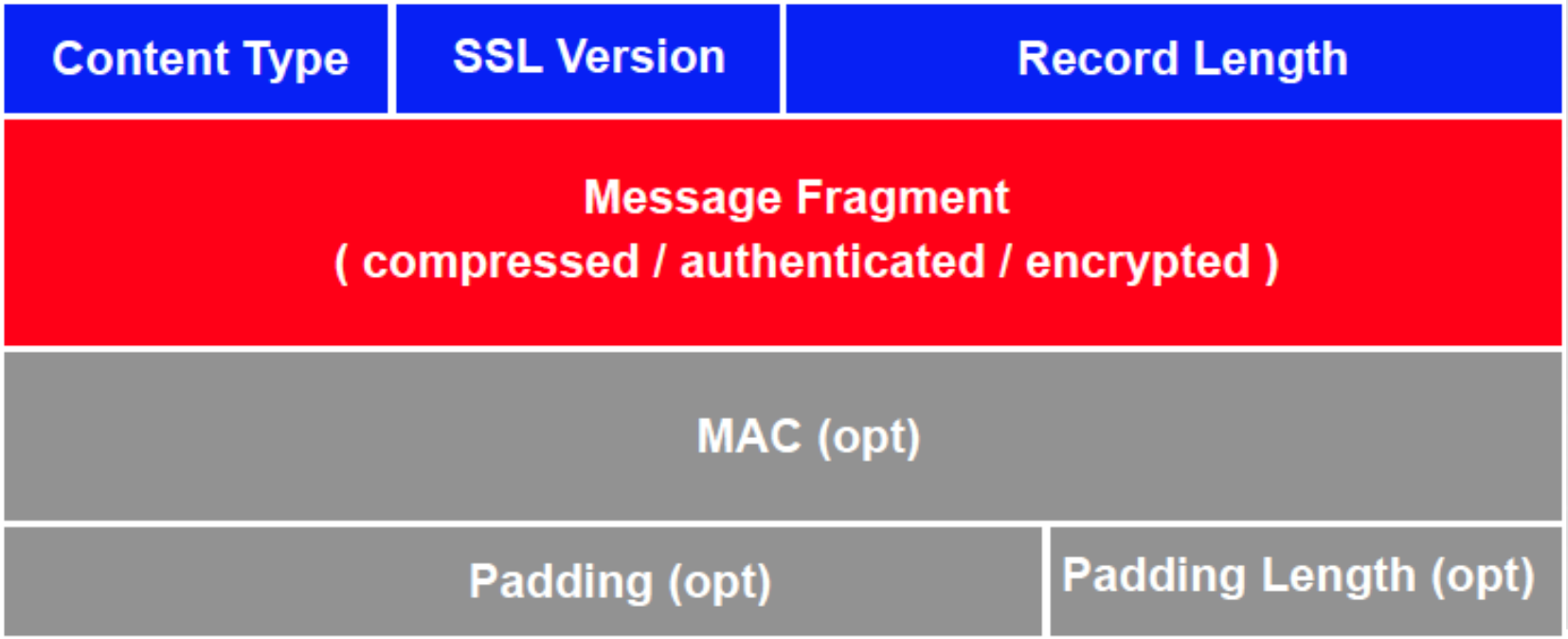
- SSL/TLS partitions the reliable octet stream of TCP into records, a record can be either
  - Handshake: session establishment and initialisation
  - Change cipher: switches to a given set of security algorithms
    - Could simply have been a handshake message
- User data: the application protocol (SMTP, FTP, LDAP, HTTP, …). HTTP is the most common, runs at port 443 (HTTPS)
- Alerts: error messages and notification of connection closure

# SSL ALERT PROTOCOL MESSAGE

- Used to convey alerts and errors to the client
- The alert messages are protected according to the ciphers agreed for the SSL session
- Each message consists of two bytes
  - The first byte indicates the severity of the alert
    - 1- warning, 2- fatal
    - If the level is fatal SSL terminates the connection
  - The second byte contains the code that indicates the specific type of alert

# SSL RECORD PROTOCOL

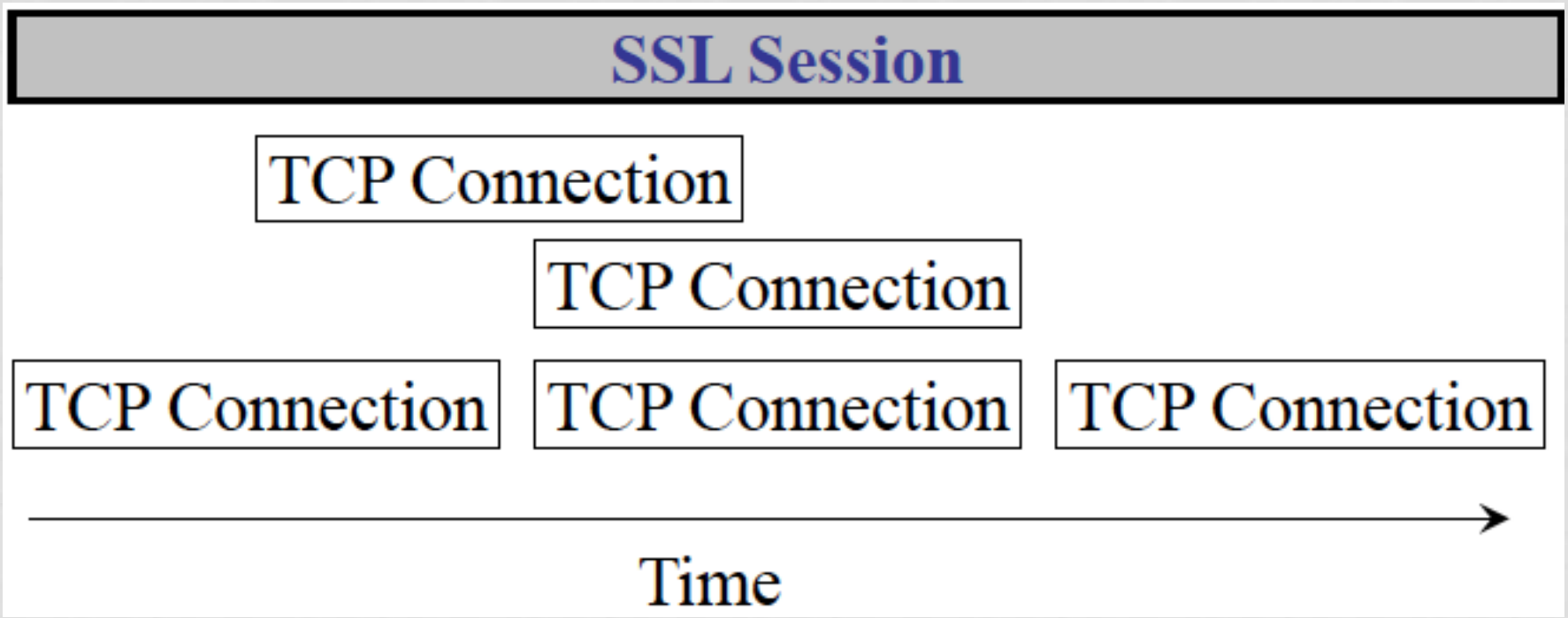
- The application message is broken up into fragments
- The SSL Record Protocol is applied to each fragment
  - Provides confidentiality via encryption
  - Provides message integrity with a MAC
  - Optional compression



# TCP AND SSL SESSIONS

- Connections between client and server are provided by TCP
- A SSL session is an association between the client and the server
  - SSL sessions run over TCP connections
- SSL was designed to work with HTTP1.0
  - Opens a lot of TCP connections (one for each item)
  - An SSL session should be able to span multiple TCP connections, both sequentially and in parallel
  - A client and server can disconnect then reconnect and continue using the same SSL session
- SSL sessions are created using the SSL Handshake Protocol
- The generated master key could be reused to resume the session in a "cheap" way.

# TCP CONNECTIONS AND SSL



# SHORT COMINGS: SSL

- User authentication is not available in v2
  - Optional in v3
- Web based CAs do not always authenticate the customer strongly
  - e.g. Verisign Class 1,
  - The server can not trust the user's certificate
- Poor support for certificate revocation in SSL products
  - Most web clients would not know if a server's certificate had been revoked
- If the system is configured poorly, it is possible for SSL to negotiate a NULL cipher suite so that no protection is carried out at all

# TLS VS. SSL

- TLS and SSL are similar but incompatible
- TLS algorithms
  - DSA (Digital Signature Algorithm), RSA is optional
  - Message authentication code: keyed-Hash Message Authentication Code (HMAC)

$$\mathrm{HMAC}_K(m) = h\Big( (K \oplus \mathrm{opad}) \| h\big((K \oplus \mathrm{ipad}) \| m\big) \Big)$$

  - SSL uses its own keyed hash algorithm
- Secret key generation: MD5 and SHA-1
- Signature: MD5 and SHA-1

# SUMMARY

- TLS is the Internet standard version of SSL
- SSL/TLS provides application-level security over TCP
- SSL/TLS provide
  - Confidentiality, using symmetric encryption
  - Data Integrity, using message authentication codes
  - Optional client authentication, using public key certificates
- SSL/TLS allow the negotiation of security mechanisms between two users

END