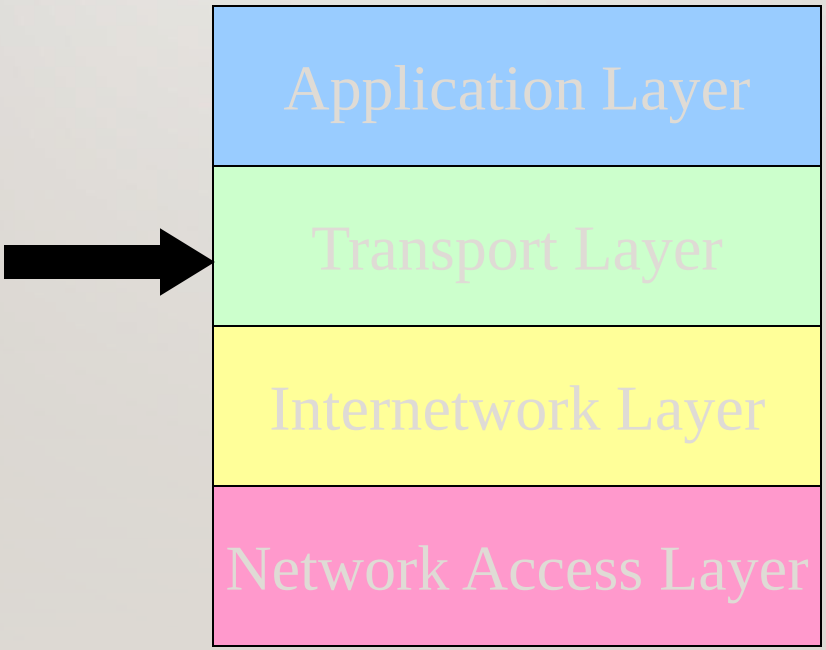
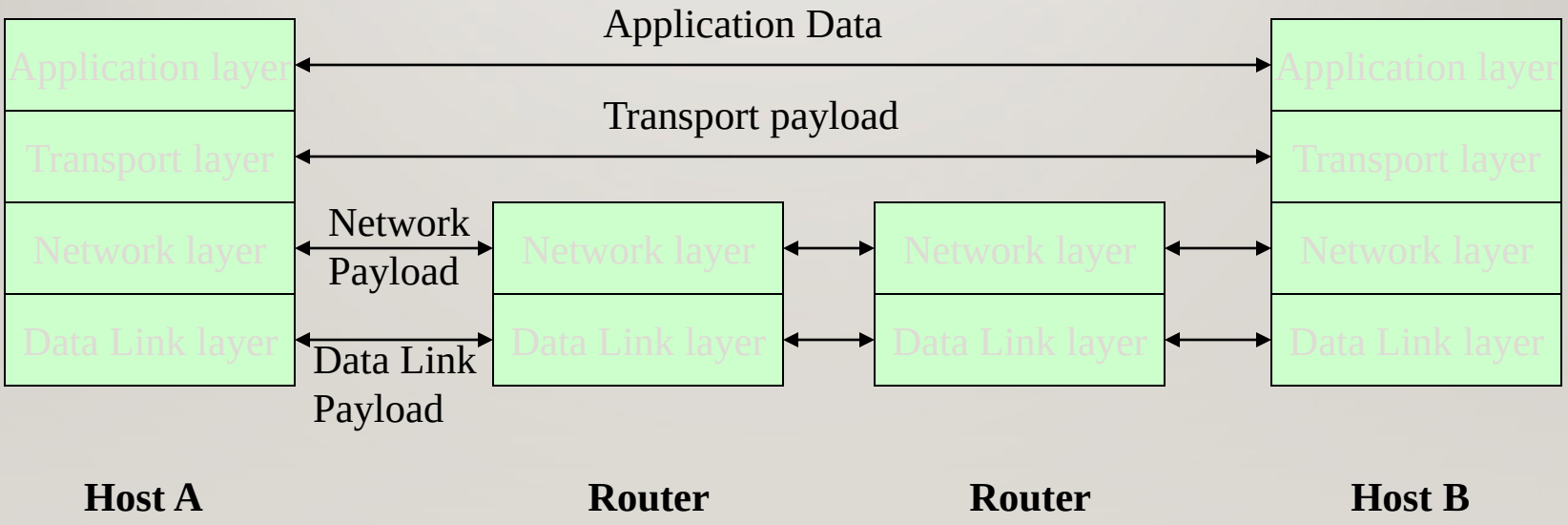
MODULE -4

TRANSPORT LAYER SECURITY

TCP/IP PROTOCOL STACK



COMMUNICATION BETWEEN LAYERS



TRANSPORT LAYER

- Provides services to the application layer
- Services:
  - Connection-oriented or connectionless transport
  - Reliable or unreliable transport
  - Security

SECURITY REQUIREMENTS

- Key management
- Confidentiality
- Repudiation
- Integrity/authentication
- Authorization

TRANSPORT LAYER SECURITY

- Advantages:
  - Does not require enhancement to each application
- Disadvantages:
  - Obtaining user context gets complicated
  - Protocol specific --> need to be duplicated for each transport protocol
  - Need to maintain context for connection (not currently implemented for UDP)

TRANSPORT LAYER SECURITY PROTOCOLS

- Connectionless and connection-oriented transport layer service:
  - Security Protocol 4 (SP4) – NSA, NIST
  - Transport Layer Security (TLSP) – ISO
- Connection-oriented transport layer service:
  - Encrypted Session Manager (ESM) – AT&T Bell Labs.
  - Secure Socket Layer (SSL) – Netscape Communications
  - Transport Layer Security (TLS) – IETF TLS WG

Most popular transport layer security protocols

SSL

- SSL versions:
  - 1.0: serious security flaws – never released to public
  - 2.0: some weaknesses – in Netscape Navigator 1.0-2.x
  - 3.0: no serious security flaws – in Netscape Navigator 3.0 and higher, MS Explorer 3.0 and higher
- RFC2246, http://www.ietf.org/rfc/rfc2246.txt
- Open-source implementation at http://www.openssl.org/

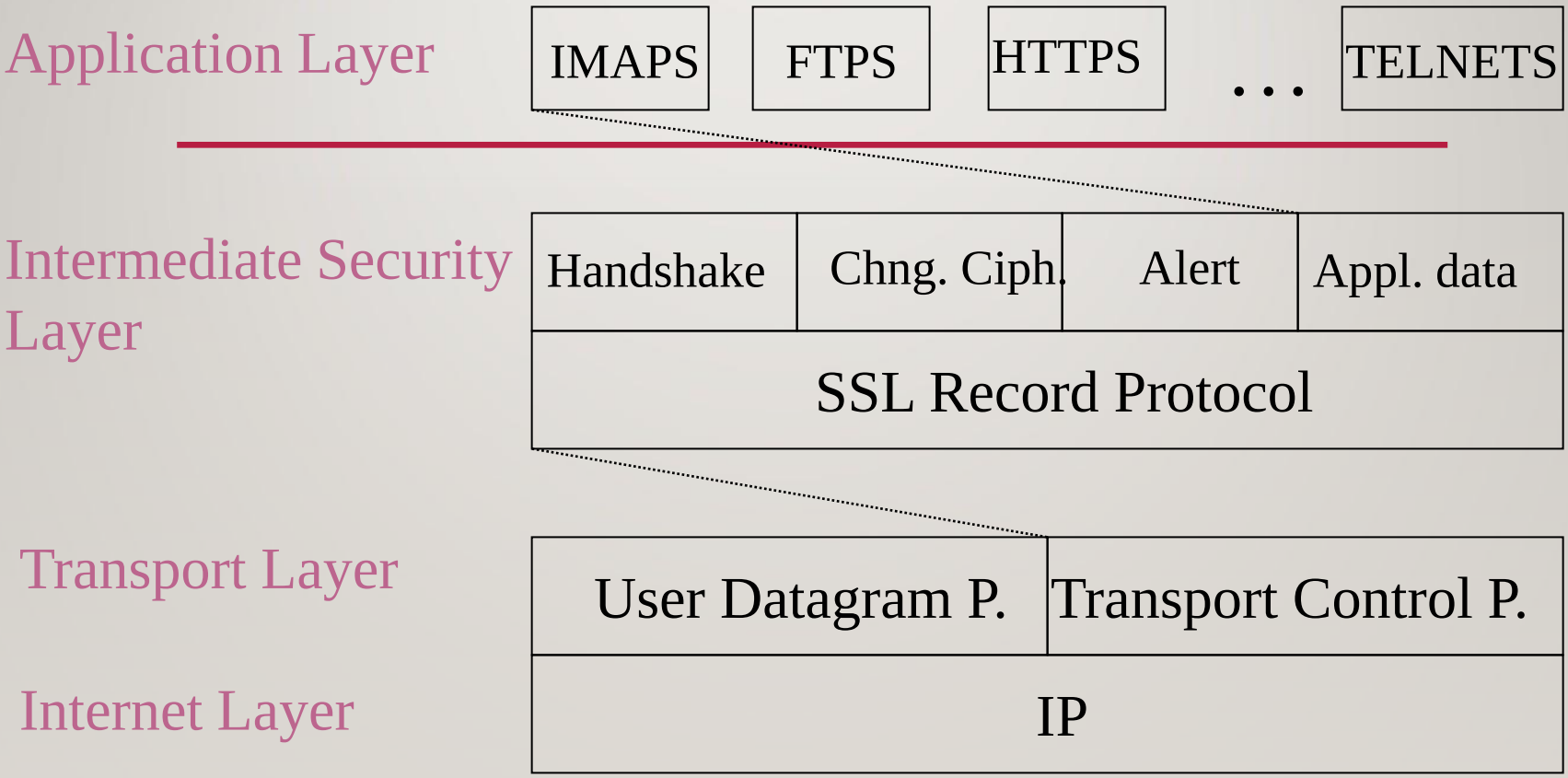
SSL 2.0 VULNERABILITIES

- Short key length
- Weak MAC construction
- Message integrity vulnerability
- Ciphersuite rollback attack

SSL

- Intermediate security layer between the transport layer and the application layer
- Based on connection-oriented and reliable service (e.g., TCP)
- Able to provide security services for any TCP-based application protocol, e.g., HTTP, FTP, TELNET, POP3, etc.
- Application independent

SSL ARCHITECTURE



SSL SERVICES

- SSL provides
  - Client- server authentication (public-key cryptography)
  - Data traffic confidentiality
  - Message authentication and integrity check
- SSL does not provide
  - Traffic analysis
  - TCP implementation oriented attacks

SSL USAGE

- **Both client and server must know that the other is using SSL** by either
  - Using dedicated port numbers – separate port number for every application protocol using SSL
    - May require two TCP connections if the client does not know what the server supports
  - Using normal port number but negotiate security options as part of the application protocol
    - Requires each application protocol to be modified
    - Will be needed for future applications
  - Using a TCP option to negotiate the use of a security protocol during TCP connection establishment
    - Hasn't been seriously discussed yet

SSL

- Connection: a transport that provides a service. Every connection is associated with a session
- Session: association between a client and a server. Created by the Handshake protocol.

SSL STATE INFORMATION

- SSL session is stateful → SSL protocol must initialize and maintain session state information on either side of the session
- SSL session can be used for several connections → connection state information

SSL SESSION STATE INFORMATION ELEMENTS

- Session ID: chosen by the server to identify an active or resumable session state
- Peer certificate: certificate for peer entity (X.509)
- Compression method: algorithm to compress data before encryption
- Cipher spec: specification of data encryption and Message Authentication Code (MAC) algorithms
- Master secret: 48-byte secret shared between client and server
- Is resumable: flag that indicates whether the session can be used to initiate new connections

SSL CONNECTION STATE INFORMATION ELEMENTS

- Server and client random: 32 bytes sequences that are chosen by server and client for each connection
- Server write MAC secret: secret used for MAC on data written by server
- Client write MAC secret: secret used for MAC on data written by client
- Server write key: key used for data encryption by server and decryption by client
- Client write key: key used for encryption by client and decryption by server
- Initialization vector: for CBC block ciphers
- Sequence number: for both transmitted and received messages, maintained by each party

SSL CONNECTION STATE

- Four parts to state
  - Current read state
  - Current write state
  - Pending read state
  - Pending write state
- Handshake:
  - Initial current state is empty
  - Pending state can be made current or reinitialized to empty

SSL PROTOCOL

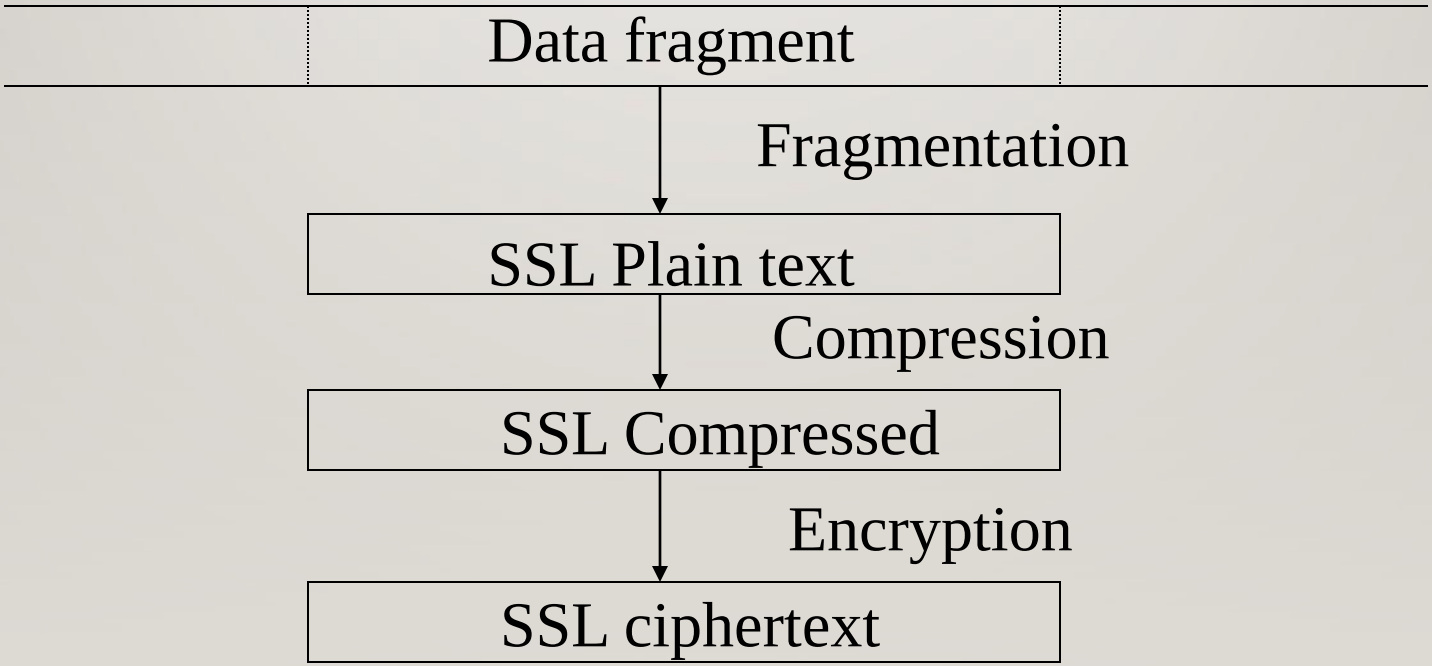
Components:

- **SSL Record Protocol**
  - Layered on top of a connection-oriented and reliable transport layer service
  - Provides message origin authentication, data confidentiality, and data integrity
- **SSL sub-protocols**
  - Layered on top of the SSL Record Protocol
  - Provides support for SSL session and connection establishment

SSL RECORD PROTOCOL

- Receives data from higher layer SSL sub-protocols
- Addresses
  - Data fragmentation
  - Compression
  - Authentication
  - Encryption

SSL RECORD PROTOCOL



SSL RECORD CONTENT

- Content type
  - Defines higher layer protocol that must be used to process the payload data (8 bits, only 4 defined)
- Protocol version number
  - Defines SSL version in use (8 bits major, 8 bits minor)
- Length: max 2^{14} + 2048
- Data payload
  - Optionally compressed and encrypted
  - Encryption and compression requirements are defined during SSL handshake
- MAC
  - Appended for each each record for message origin authentication and data integrity verification

SSL SUB-PROTOCOLS

- Alert Protocol
  - Used to transmit alerts via SSL Record Protocol
  - Alert message: (alert level, alert description)
- Handshake Protocol – Complex
  - Used to mutually authenticate client and server and exchange session key
  - Establish new session and connection together or
  - Uses existing session for new connection

SSL SUB-PROTOCOLS

- ChangeCipherSpec Protocol
  - Used to change cipher specifications
  - Can be changed at the end of the handshake or later
- Application Protocol
  - Used to directly pass application data to the SSL Record Protocol

SSL HANDSHAKE

- Phase 1: establish security capabilities
- Phase 2: server authentication and key exchange
- Phase 3: client authentication and key exchange
- Phase 4: finish

SSL HANDSHAKE

Phase 1

Security capabilities

1. C → S: **C**LIENT**H**ELLO
2. S → C: **S**ERVER**H**ELLO

Phase 2

Optional server messages

[**C**ERTIFICATE]
[**S**ERVER**K**EY**E**XCHANGE]
[**C**ERTIFICATE**R**EQUEST]
SERVER**H**ELLO**D**ONE

3. C → S: [**C**ERTIFICATE]

Phase 3

Client key exchange

CLIENT**K**EY**E**XCHANGE
[**C**ERTIFICATE**V**ERIFY]
CHANGE**C**IPHER**S**PEC
FINISH

Phase 4

Change cipher suite

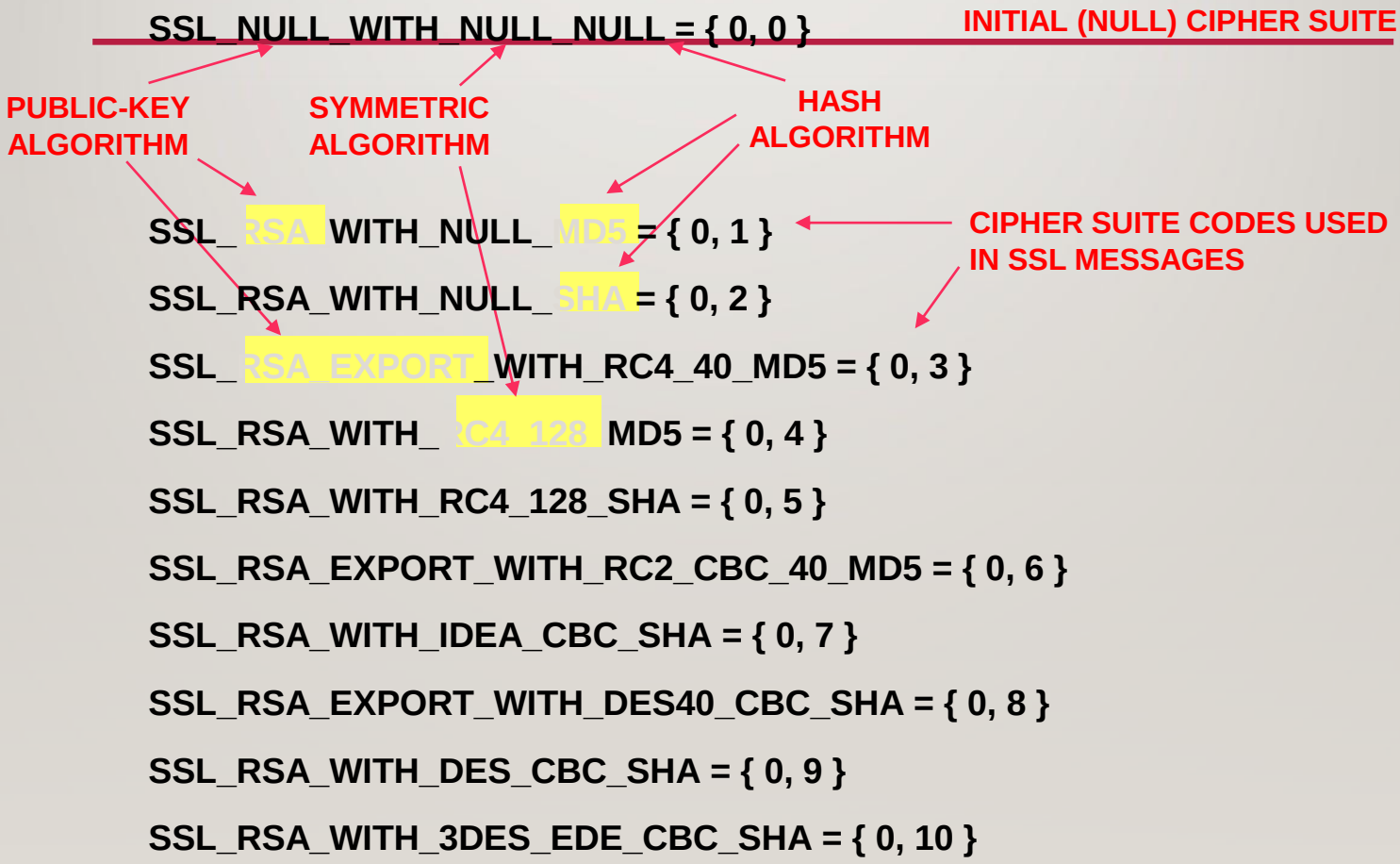
4. S → C: **C**HANGE**C**IPHER**S**PEC
FINISH

1. C → S: **C**LIENT**H**ELLO

SSL HANDSHAKE

- CLIENTHELLO message is sent by the client
  - When the **client wants to establish a TCP connection** to the server,
  - When a **HELLOREQUEST message is re**ceived, or
  - When client wants to **renegotiate security parameters** of an existing connection
- Message content:
  - Number of highest SSL understood by the client
  - Client's random structure (32-bit timestamp and 28-byte pseudorandom number)
  - Session ID client wishes to use (ID is empty for existing sessions)
  - List of cipher suits the client supports
  - List of compression methods the client supports

Cipher Suites



2. S → C: **S**ERVER**H**ELLO
[**C**ERTIFICATE]
[**S**ERVER**K**EY**E**XCHANGE]
[**C**ERTIFICATE**R**EQUEST]
SERVER**H**ELLO**D**ONE

SSL HANDSHAKE

- Server processes CLIENTHELLO message
- Server Respond to client with SERVERHELLO message:
  - Server version number: lower version of that suggested by the client and the highest supported by the server
  - Server's random structure: 32-bit timestamp and 28-byte pseudorandom number
  - Session ID: corresponding to this connection
  - Cipher suite: selected by the server for client's list
  - Compression method: selected by the server from client's list

2. S → C: **S**ERVER**H**ELLO
[**C**ERTIFICATE]
[**S**ERVER**K**EY**E**XCHANGE]
[**C**ERTIFICATE**R**EQUEST]
SERVER**H**ELLO**D**ONE

SSL HANDSHAKE

Optional messages: Phase 2– server authentication

- CERTIFICATE:
  - If the server is using certificate-based authentication
  - May contain RSA public key → good for key exchange
- SERVERKEYEXCHANGE:
  - If the client does not have certificate, has certificate that can only be used to verify digital signatures, or uses FORTEZZA token-based key exchange (not recommended)
- CERTIFICATEREQUEST:
  - Server may request personal certificate to authenticate a client

3. C → S: [**C**ERTIFICATE]
CLIENT**K**EY**E**XCHANGE
[**C**ERTIFICATE**V**ERIFY]
CHANGE**C**IPHER**S**PEC
FINISH

SSL HANDSHAKE

- Client processing:
  - Verifies site certification
    - Valid site certification if the server's name matches the host part of the URL the client wants to access
  - Checks security parameters supplied by the SERVERHELLO

3. C → S: [**C**ERTIFICATE]
CLIENT**K**EY**E**XCHANGE
[**C**ERTIFICATE**V**ERIFY]
CHANGE**C**IPHER**S**PEC
FINISH

SSL HANDSHAKE

- Client messages: Phase 3 – client authentication and key exchange
  - CERTIFICATE
    - If server requested a client authentication, client sends
  - CLIENTKEYEXCHANGE
    - Format depends on the key exchange algorithm selected by the server
      - RSA: 48-byte premaster secret encrypted by the server's public key
      - Diffie-Hellman: public parameters between server and client in SERVERKEYEXCHANGE and CLIENTKEYEXCHANGE msgs.
      - FORTEZZA: token-based key exchange based on public and private parameters
    - Premaster key is transformed into a 48-byte master secret, stored in the session state

3. C → S: [**C**ERTIFICATE]
CLIENT**K**EY**E**XCHANGE
[**C**ERTIFICATE**V**ERIFY]
CHANGE**C**IPHER**S**PEC
FINISH

SSL HANDSHAKE

- Client messages:
  - CERTIFICATEVERIFY
    - If client authentication is required
    - Provides explicit verification of the use's identity (personal certificate)
  - CHANGECIPHERSPEC
    - Completes key exchange and cipher specification
  - FINISH
    - Encrypted by the newly negotiated session key
    - Verifies that the keys are properly installed in both sites

4. S → C: **C**HANGE**C**IPHER**S**PEC
FINISH

SSL HANDSHAKE

- Phase 4: finish
- Server finishes handshake by sending CHANGECIPHERSPEC and FINISH messages

- After SSL handshake completed a secure connection is established to send application data encapsulated in SSL Record Protocol

SSL HANDSHAKE TO RESUME SESSION

1. C → S: **C**LIENT**H**ELLO
2. S → C: **S**ERVER**H**ELLO
   cHANGE**c**IPHER**s**PEC
   FINISH
3. C → S: **C**HANGE**C**IPHER**S**PEC
   FINISH

SSL PROTOCOL

- Provides secure TCP connection between client and server by
  - Server authentication
  - Optional client authentication
  - Key exchange services
  - Negotiation
  - Data confidentiality and integrity
  - Message authentication
  - Compression/decompression

SSL DELAY

- Slower than a TCP session (2-10 times)
- Causes:
  - Handshake phase
    - Client does public-key encryption
    - Server does private-key encryption (still public-key cryptography)
    - Usually clients have to wait on servers to finish
  - Data Transfer phase
    - Symmetric key encryption

FIREWALL TUNNELING

- SSL/TSL: end-to-end security → difficult to interoperate with application gateways
- Firewalls: man-in-the-middle
  - Application protocol being proxied
  - Application protocol being tunneled

PROXIED PROTOCOL

- Proxy server is aware of the specifics of the protocol and understand protocol level processing
- Support:
  - Protocol-level filtering
  - Access control
  - Accounting
  - Logging
- Usually proxied protocols: telnet, ftp, http

TUNNELED PROTOCOL

- Proxy server:
  - NOT aware of the specifics of the protocol → simply relaying the data between Client and Server
  - Does NOT have access to data being transferred
  - Knows: source and destination addresses (IP and port) and the requesting user (if authentication is supported)
- Cannot support: protocol –level filtering, access control, and logging at the same extend as the proxied version.
- Usually tunneled protocols: SSL-enhanced protocols

SUMMARY

- Advantages of SSL/TSL:
  - Simplicity
  - Wide deployment
- Disadvantages:
  - Do not secure UDP
  - Work poorly with applications gateways

TESTING FOR SSL-TLS (OWASP-CM-001)

- Problem: legacy and new web servers are often able and configured to handle weak cryptographic options
- Causes: historic export restrictions of high grade cryptography

SSL TESTING CRITERIA

HTTPS://WWW.OWASP.ORG/INDEX.PHP/TESTING_FOR_SSL-TLS_%28OWASP-CM-001%29

Minimum checklist:

- SSLv2, due to known weaknesses in protocol design
- Export (EXP) level cipher suites in SSLv3
- Cipher suites with symmetric encryption algorithm smaller than 128 bits
- X.509 certificates with RSA or DSA key smaller than 1024 bits
- X.509 certificates signed using MD5 hash, due to known collision attacks on this hash
- TLS Renegotiation vulnerability – patch available

THANK YOU