

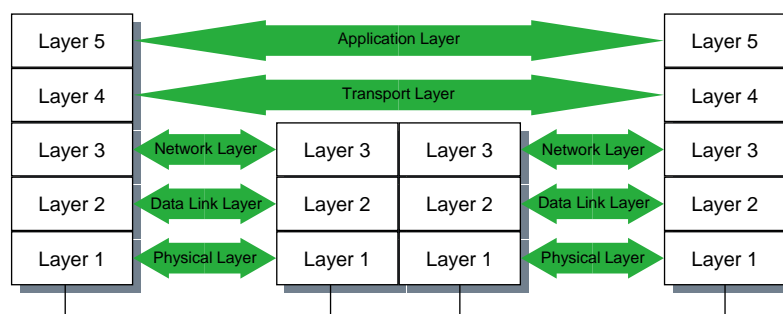
Network Security

Chapter 9

SSL/TLS – Transport Layer Security

Arno Wacker
IngWiss / Distributed Systems
University Duisburg-Essen

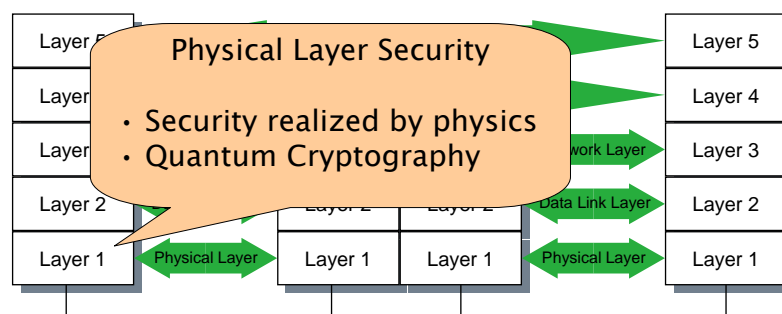
Protocol Layers (1)



Protocol Layers (2)

- Physical Layer
 - Physical connection between two processes
- Data-Link Layer
 - Communication between two connected computers
- Network Layer
 - Communication between two computers via the internet
- Transport Layer
 - Communication between two processes
 - Processes can reside on any computer in the internet
 - Multiple processes per computer

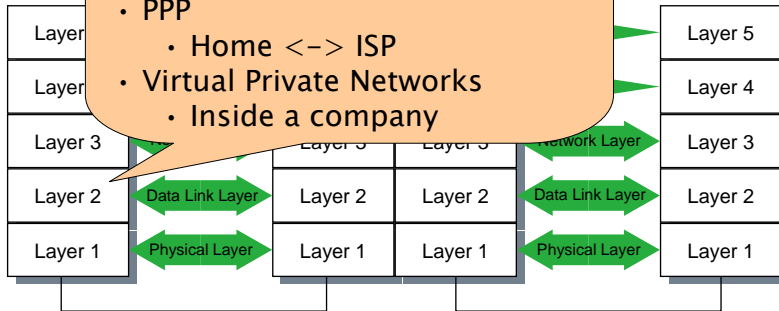
Protocol Layers (3)



Protocol Layers (4)

Data-Link Layer Security

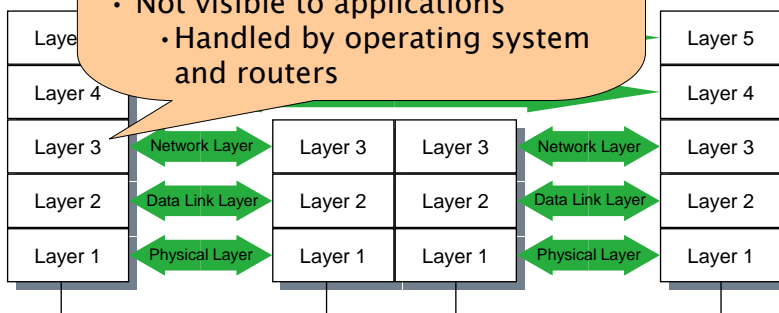
- Implemented by switches, operating system, etc.
- PPP
 - Home <-> ISP
- Virtual Private Networks
- Inside a company



Protocol

Network Layer Security

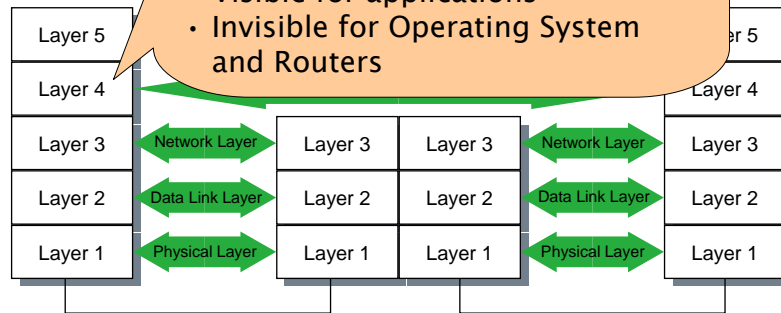
- Connection-less
- Security on the IP packet level
- IPSec
- Not visible to applications
 - Handled by operating system and routers



Protocol L

Transport Layer Security

- Connection-oriented
- Secure Socket Layer (SSL)
 - Works on Berkeley Sockets
- Visible for applications
- Invisible for Operating System and Routers



Transport Layer Security

- Prerequisites
 - A reliable transport layer, e.g. TCP
- Additional security properties
 - Authentication of users or services
 - Example: Is Alice talking to me?
Am I talking to www.fun-shop.com ?
 - Confidentiality
 - Transmitted data is encrypted
 - Data Integrity
 - Transmitted data has not been modified
 - No additional data inserted
 - No data removed

The Secure Socket Layer (SSL) Protocol (1)

- SSL was originally designed to protect HTTP sessions
 - SSL version 2.0 was included in browsers of Netscape
⇒ it quickly became predominant
 - SSL v.2 contained some flaws and so Microsoft Corporation developed a competing protocol called Private Communication Technology (PCT)
 - Netscape improved the protocol
 - SSL v.3 became the de-facto standard protocol for securing HTTP traffic

The Secure Socket Layer (SSL) Protocol (2)

- SSL without HTTP
- SSL can be deployed to secure arbitrary applications that run over TCP
- SSL adds security on-top of TCP
 - It does not replace the TCP implementation of the OS
- In 1996 the IETF decided to specify a generic *Transport Layer Security (TLS)*
 - Standardized in 1999
 - It is based on SSL
 - Further extensions: Kerberos, TLS + HTTP/1.1, ...

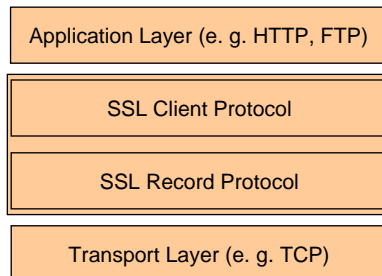
SSL Security Services (1)

- Peer entity **authentication**
 - Prior to any communications between a client and a server, an authentication protocol is run to authenticate the peer entities
 - Upon successful completion of the authentication dialogue an *SSL session* is established between the peer entities
- User data **confidentiality**
 - If negotiated upon session establishment, user data is encrypted
 - Different encryption algorithms can be negotiated: RC4, DES, 3DES, IDEA

SSL Security Services (2)

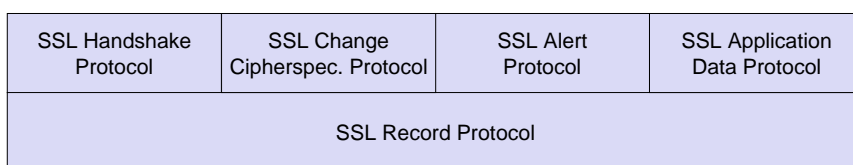
- User data **integrity**
 - A MAC based on a cryptographic hash function is appended to user data
 - The MAC is computed with a negotiated secret in prefix-suffix mode
 - Either MD5 or SHA can be negotiated for MAC computation

SSL Layer Hierarchy



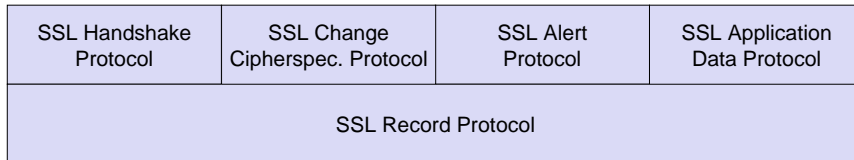
- Record Layer
 - fragments application data into blocks
 - handles encryption and compression
 - adds a MAC
- Client Layer
 - Handshake Protocol (session establishment)
 - Change Cipher Spec (as of SSL 3.0)
 - Alert Protocol (as of SSL 3.0)

SSL Protocol Architecture (1)



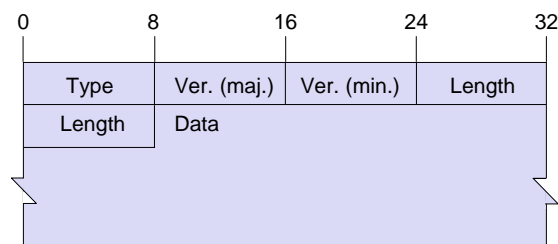
- Handshake
 - Authentication and negotiation of parameters
- Change Cipher Spec.
 - Signaling of transitions in ciphering strategy
- Alert: signaling of error conditions
- Application Data: For sending user data

SSL Protocol Architecture (2)



- Record:
 - Fragmentation of user data into plaintext records of length $< 2^{14}$
 - Compression (optional) of plaintext records
 - Encryption and integrity protection (both optional)

SSL Record Protocol



- Content Type:
 - Change Cipherspec. (20), Alert (21), Handshake (22)
Application Data (23)
- Version of SSL (major = 3, minor = 0)
- Length: the length of the data in bytes

SSL Record Protocol Processing: Sender

- Fragment user data into records
 - Maximum length of 2^{14} octets
- Compress record data
 - Default algorithm for this is null (~ no compression)
- Append message authentication code (HMAC)
 - $MAC = H(MAC-Secret + pad_2 + H(MAC-Secret + pad_1 + seqnum + length + data))$
 - Note, that seqnum is not transmitted
 - TCP takes care of the sequence
- Encrypt data and MAC
 - Use the encryption algorithm defined in cipherspec

SSL Record Protocol Processing: Receiver

- Decrypt
 - Now we have data and MAC
- Check integrity
 - Check MAC against the data
- Decompress
 - Now we have plaintext again
- Send to application
 - Deliver the plaintext to the application

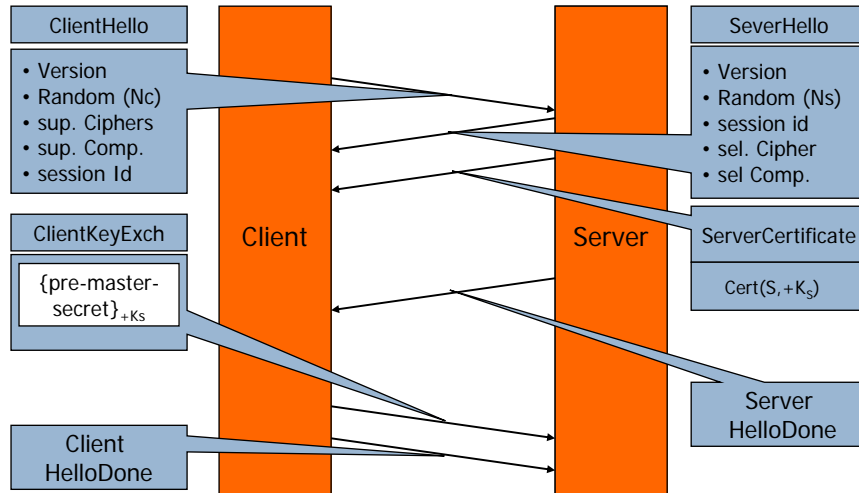
SSL Session State

- *Session identifier*: a byte sequence chosen by the server
- *Compression method*: algorithm to compress data prior to encryption
- *Cipher spec*: specifies cryptographic algorithms and parameters
- *Master secret*: a negotiated shared secret of length 48 byte
- *Is resumable*: a flag indicating if the session supports new connections

SSL Handshake Protocol: Introduction

- The SSL handshake protocol
 - Establishes peer authentication
 - Negotiates cryptographic parameters for SSL session
- An SSL session can be negotiated to be resumable
 - Resuming/Duplicating SSL sessions allows to re-use an established security context
 - This is very important for securing HTTP traffic, as usually every item on a web page is transferred in an individual TCP connection
 - When resuming/duplicating an existing session, an abbreviated handshake is performed

SSL Handshake (1)



SSL Handshake (2)

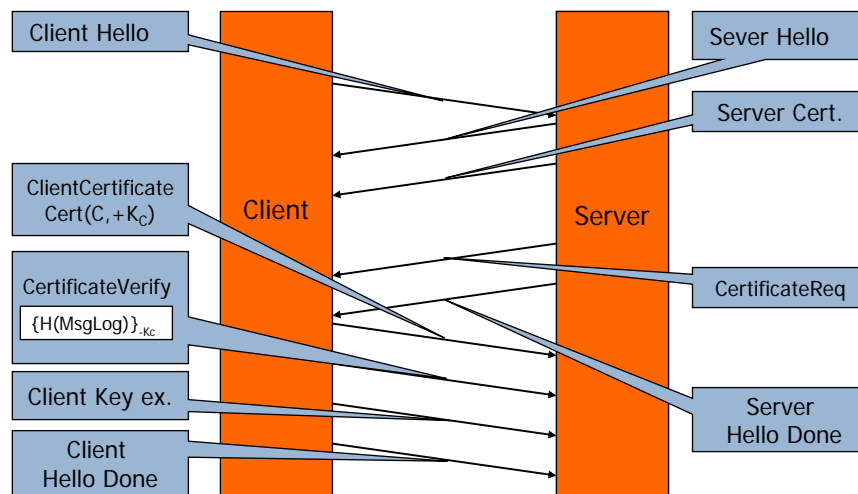
- **PreMasterSecret PMS**
 - Random number by client based on N_C and N_S
 - Prevents replay attacks
 - Sent to server encrypted with server's public key $+K_S$
 - PMS is now shared secret between client and server
 - Used by client and server to compute MasterSecret
- **MasterSecret**
 - Hash over PreMasterSecret (PMS) and client and server nonces (N_C and N_S):

$$MS = H(PMS, N_S, N_C)$$
 - Used to compute MAC-secret und Write-Secret keys for server and client

SSL Handshake (3)

- *Server and client random*
 - Byte sequences chosen by server and client
- *Server MAC secret*
 - Used in MAC computations by the server
- *Client MAC secret*
 - Used in MAC computations by the client
- *Server write key*
 - Encryption by server and decryption by client
- *Client write key*
 - Encryption by client and decryption by server

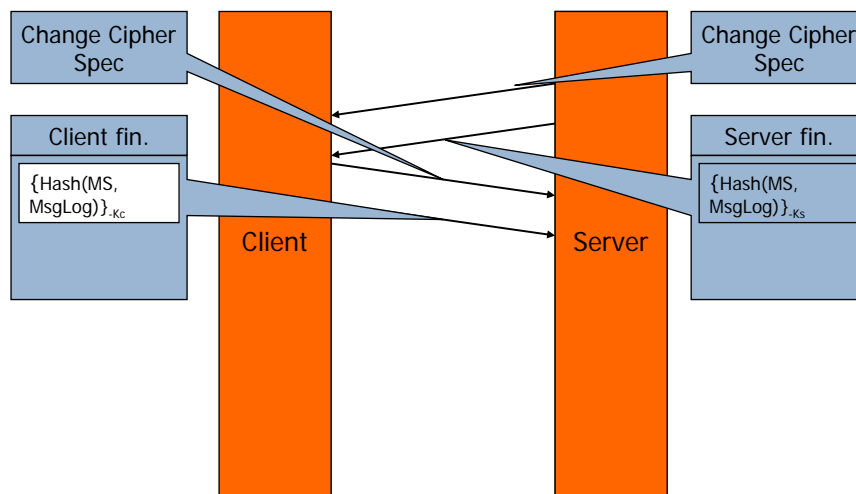
Handshake with Client Authentication (1)



Handshake with Client Authentication (2)

- Server may request client authentication by sending a CertificateRequest
- Client returns its certificate, which includes its public key $+K_C$
- CertificateVerify message may be sent allowing the server to (explicitly) verify the client's certificate
 - $H(\text{MS}, \text{MsgLog})$ is a hash over the master secret and all handshake messages sent so far
 - Hash value is encoded using the client's private key $-K_C$

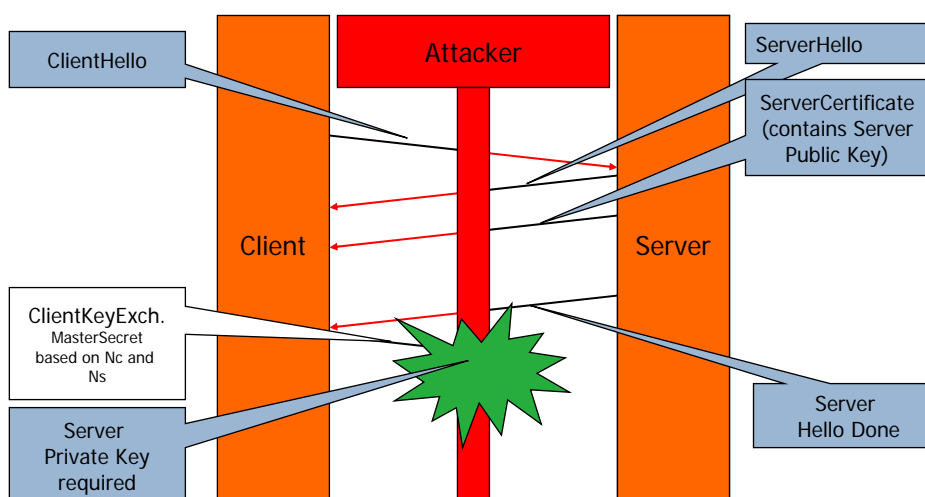
SSL Abbreviated Handshake (1)



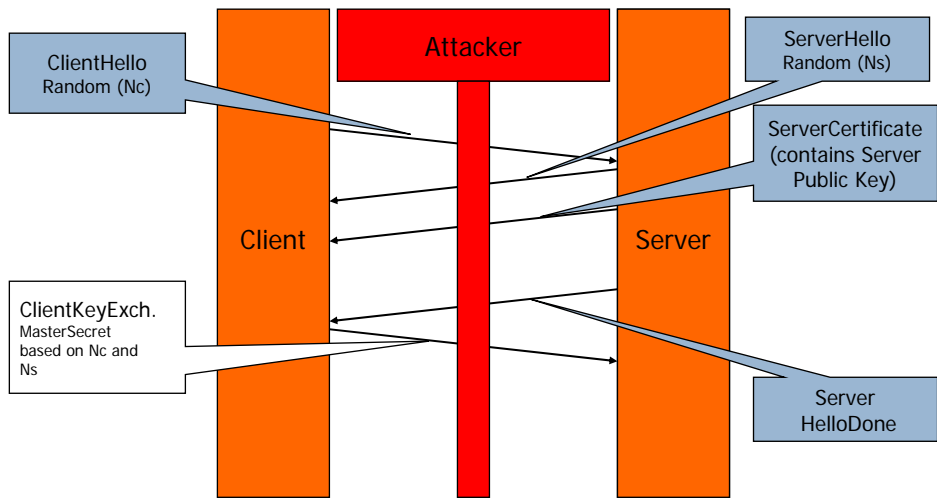
SSL Abbreviated Handshake (2)

- ChangeCipherSpec message
 - Indicates that for the following messages the newly negotiated cipher suite is to be used
- Finish message
 - Complete the handshake
 - Hash(MS,MsgLog) encodes the exchange of all handshake messages from HelloClient until now not including the finish messages
 - If both client and server compute the same hash value they have seen the same handshake messages and can start data exchange
 - Otherwise, they terminate the session

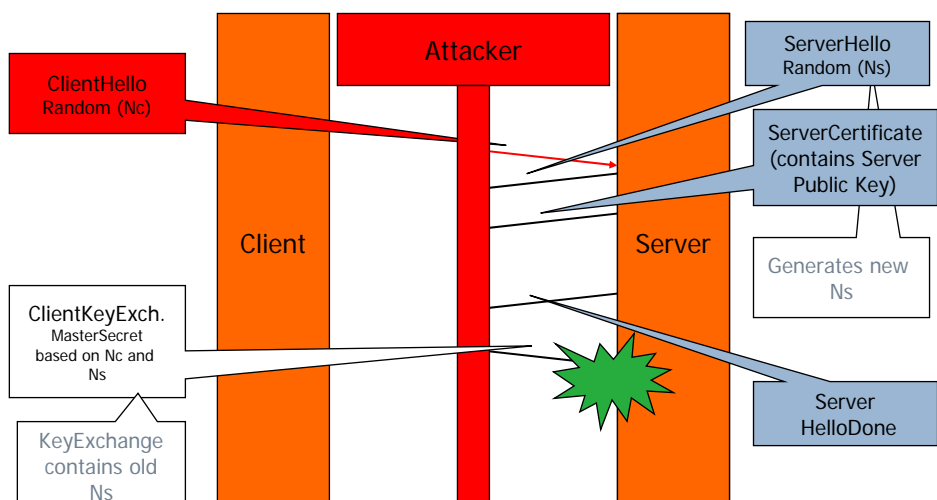
Man-in-the-Middle Attack



Replay Attack – Record Part



Replay Attack – Replay Part



The Transport Layer Security Protocol

- In 1996 the IETF started a working group to define a transport layer security (TLS) protocol:
 - Officially, the protocols SSL, SSH and PCT were announced to be taken as input
 - However, the TLS V.1.0 specification draft published in December 1996 was essentially the same as the SSL V.3.0 specification

The Secure Shell Protocol – SSH (1)

- SSH was originally designed to provide a secure replacement for the Unix r-tools
 - rlogin
 - rsh
 - rcp
- ⇒ SSH is an application or session-layer protocol
- However, as SSH also includes a generic transport layer security protocol
 - ⇒ It is discussed in this chapter as a transport layer security protocol

The Secure Shell Protocol – SSH (2)

- Secure Shell (SSH) Version 1 was originally developed by Tatu Ylönen at the Helsinki University of Finland
- The author also provided a free implementation
 - The protocol found widespread use
 - OpenSSH is an up-to-date and free implementation
- Later on, the development of SSH was commercialized by the author
- In 1997 SSH 2.0 has been published
 - SSH 1.0 is considered to be not secure any more

The Secure Shell Protocol – SSH (3)

- SSH follows a client-server approach
- Every SSH server has at least one host key
- SSH version 2 offers two different trust models
 - Every client has a local database that associates each host name with the corresponding public host key
 - The hostname to public key association is certified by a CA and every client knows the public key of the CA
- The protocol allows full negotiation of:
 - Encryption, integrity, key exchange, compression, and public key algorithms and formats

SSH Transport Protocol

- The SSH Transport Protocol runs on top of a transport protocol
 - TCP
- It provides the following services
 - Encryption of user data
 - Data integrity
 - Server authentication (host authentication only)
 - Compression of user data prior to encryption
- Comparable to SSL

SSH Connection Protocol (1)

- The SSH connection protocol runs on top of the SSH transport protocol
- Services provided by the connection protocol:
 - Interactive login sessions
 - Remote execution of commands
 - Forwarded TCP/IP connections
 - Forwarded X11 connections
- For each of the above services one or more *“channels”* are established
 - All channels are multiplexed into a single encrypted and integrity protected SSH transport protocol connection

Conclusion

- Both SSL and SSH are suited to secure Internet communications in (above) the transport layer:
 - Both security protocols operate upon and require a transport service, e.g. TCP
 - Even though SSH operates in / above the transport layer the server authentication is host-based and not application-based
 - Transport layer security protocols offer true end-to-end protection for user data exchanged between application processes