DI-FCT-UNL
Segurança de Redes e Sistemas de Computadores
Network and Computer Systems Security

Mestrado Integrado em Engenharia Informática
MSc Course: Informatics Engineering
1º Semestre, 2019/2020

Transport Layer Security (TLS), HTTPS and WEB/HTTPS Security

TLS Primer (and the Basics)

TLS: We all 've Got You Under our Skin ;-))

Read …

- See W. Stallings, Network Security Essentials, Chapter 6:
  - Initial Web Security Considerations
    - Motivation (initially for SSL) and for TLS
    - Initial TLS presentation
    - HTTPS (how HTTPS use TLS)

  - For practical observations (tools, java programming with JSSE support and programming with TLS), please remember you have related LAB materials in:
    - LAB 7 (X509 Certificates and Certification Chains)
    - LAB 8 (Java Programming using TLS)

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

HTTP, Web Security, HTTPS and TLS

- Web Browsers, Web Servers, Web Apps and Web-Based Contents and Services
  - More and more easy to program, develop, configure, deploy and deploy, but … underlying software (runtime SW stack) can be complex and may hide many potential security flaws
    - Web Security Threats and Web Software Vulnerabilities

- More and more critical applications managing sensitive data and traffic are Web based: require Web Interaction Security not provided by HTTP
  - Web Traffic Security Protection (end-to-end security assumptions)

HTTPS / TLS Approach

TLS and the scope of HTTPS for "Web Encryption"

- More and more critical applications manage sensitive data
  - More and more Web Traffic Security, primarily supported by HTTPS (and TLS)
  - HTTPS is (and will be more and more) the unified application-level security support layer to protect web (http) traffic

See, Ex., Google, HTTPS Effort:
https://transparencyreport.google.com/https/overview?hl=en

TLS vs. Web Security Considerations

- Initial motivation: Protection of HTTP Communication
- … but designed as a generic solution (transport+session layer security) to support any application level protocol
- Usually implementations offer fast development and prototyping to migrate TCP/IP Based Applications and Protocols to adopt TLS

See provided bibliography: W. Stallings, Network Security Essentials, Chap.6 – Transport Layer Security, 6.1 – Web Security Considerations

Outline

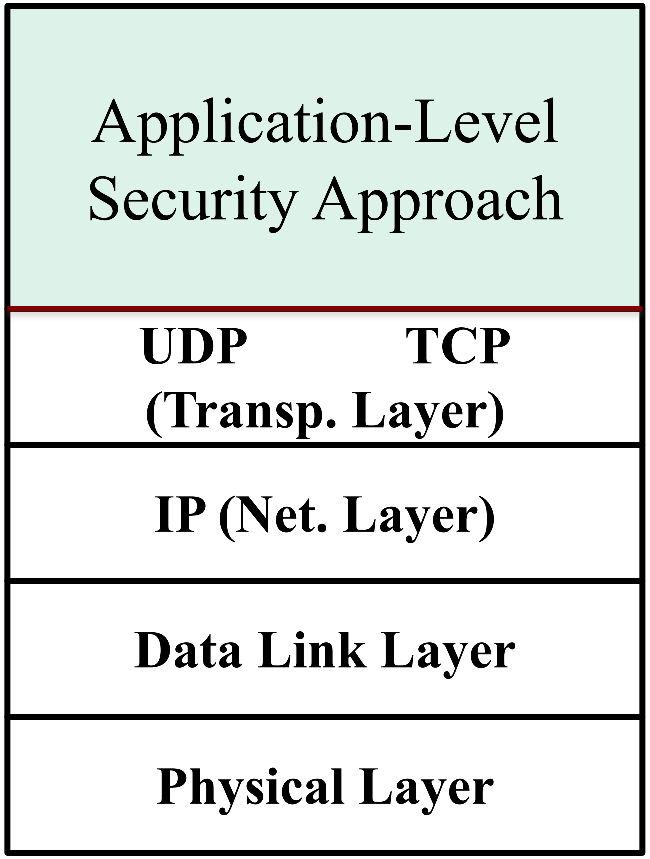
- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

Protection of Application-Level Protocols and TCP/IP Security Stack Approaches

- Protection at Application Level: App. Protocol + Session Control Services
- Some examples;
  - SSH, SCP
  - DNSSEC
  - Kerberos and Kerberized Applications
  - S/MIME, PGP
  - DMARC, DKIM
  - POP3-AUTH, POP3S, IMAP-S (ex., SASL, APOP Ext.)

Email Security Protocols

  - …… (many)



TLS Level Approach

Transport Layer Security (TLS) Approach

TLS/TCP: TLS
TLS/UDP: DTLS

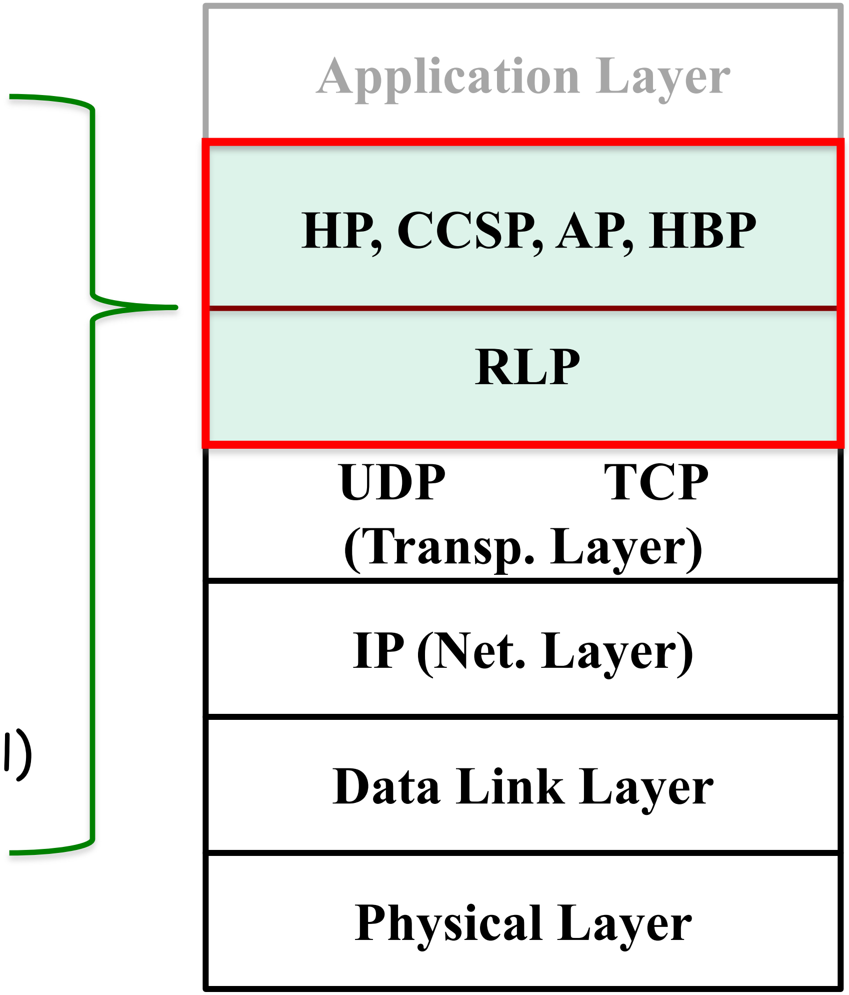
TLS as a Security (Sub)Stack providing:

Secure Transport

- RLP (Record Layer Protocol)

Session Control Services

- HP (Handshake Protocol)
- CCSP (Change Cipher Spec Protocol)
- AP (Alert Protocol)
- HBP (Heart Beat Protocol)



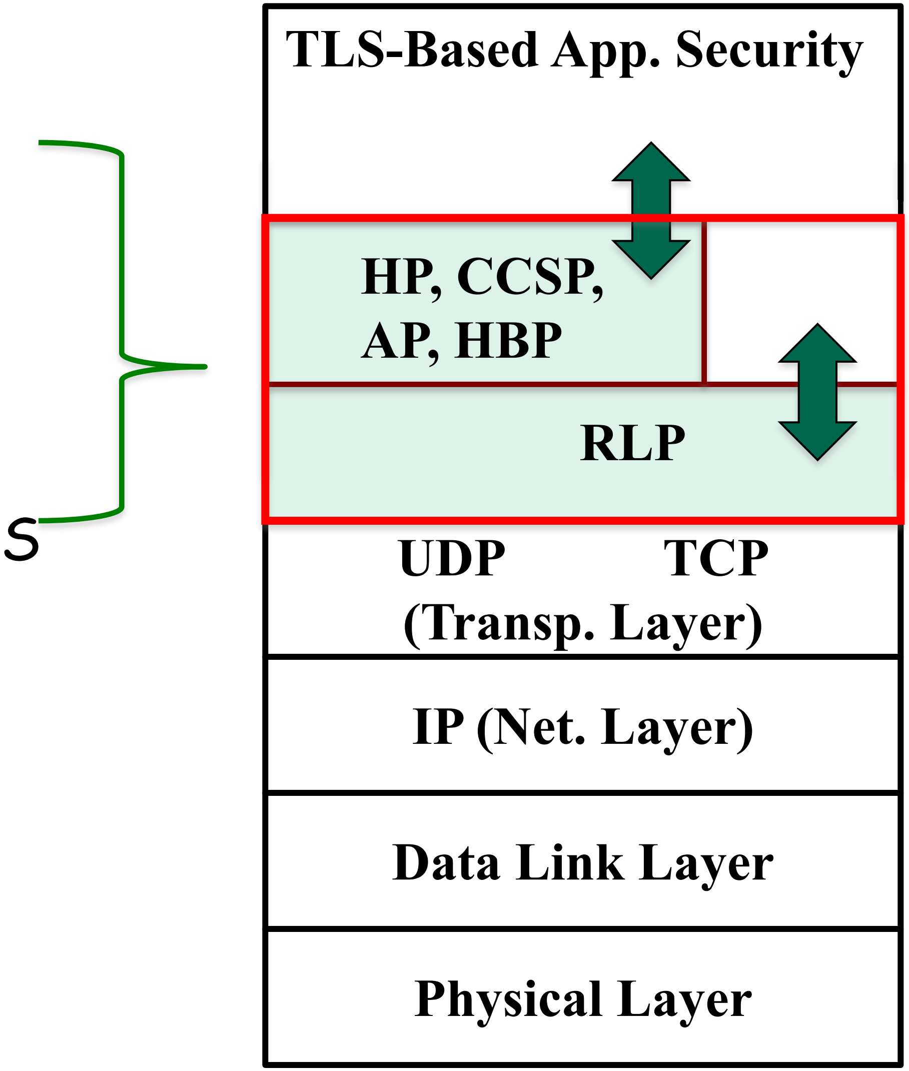
TLS-Based Application Security Approach

- TLS-Enabled Application Security

HTTPS

STARTTLS POP3S, IMAP and ACAP (…. > rfc 8314)

Kerberos V5 w/ STARTTLS Extension (rfc 6251)



Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

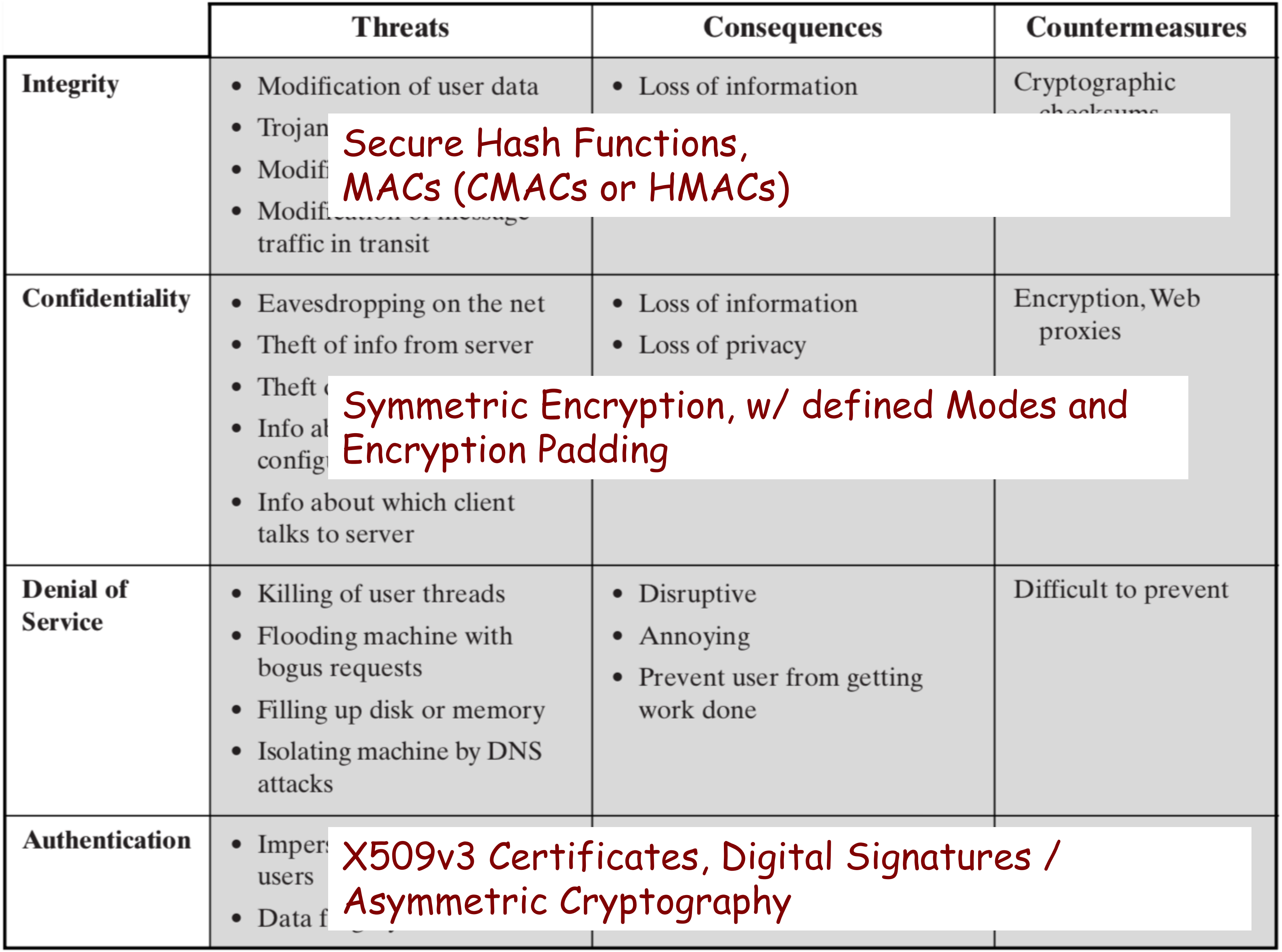
TLS: Protection provided in summary

Security Properties Addressed by TLS:

- Integrity (message and data flow-integrity)
  - Including msg ordering control and session (connection-oriented) integrity
- Confidentiality (message and data confidentiality)
  - Session or Connection Oriented Confidentiality
  - But not necessarily Traffic Confidentiality
- Authentication (peer authentication and message authentication)
- Secure establishment and management control of Session Keys and Security Association Parameters

- What about Availability protection ? (discussion)

	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



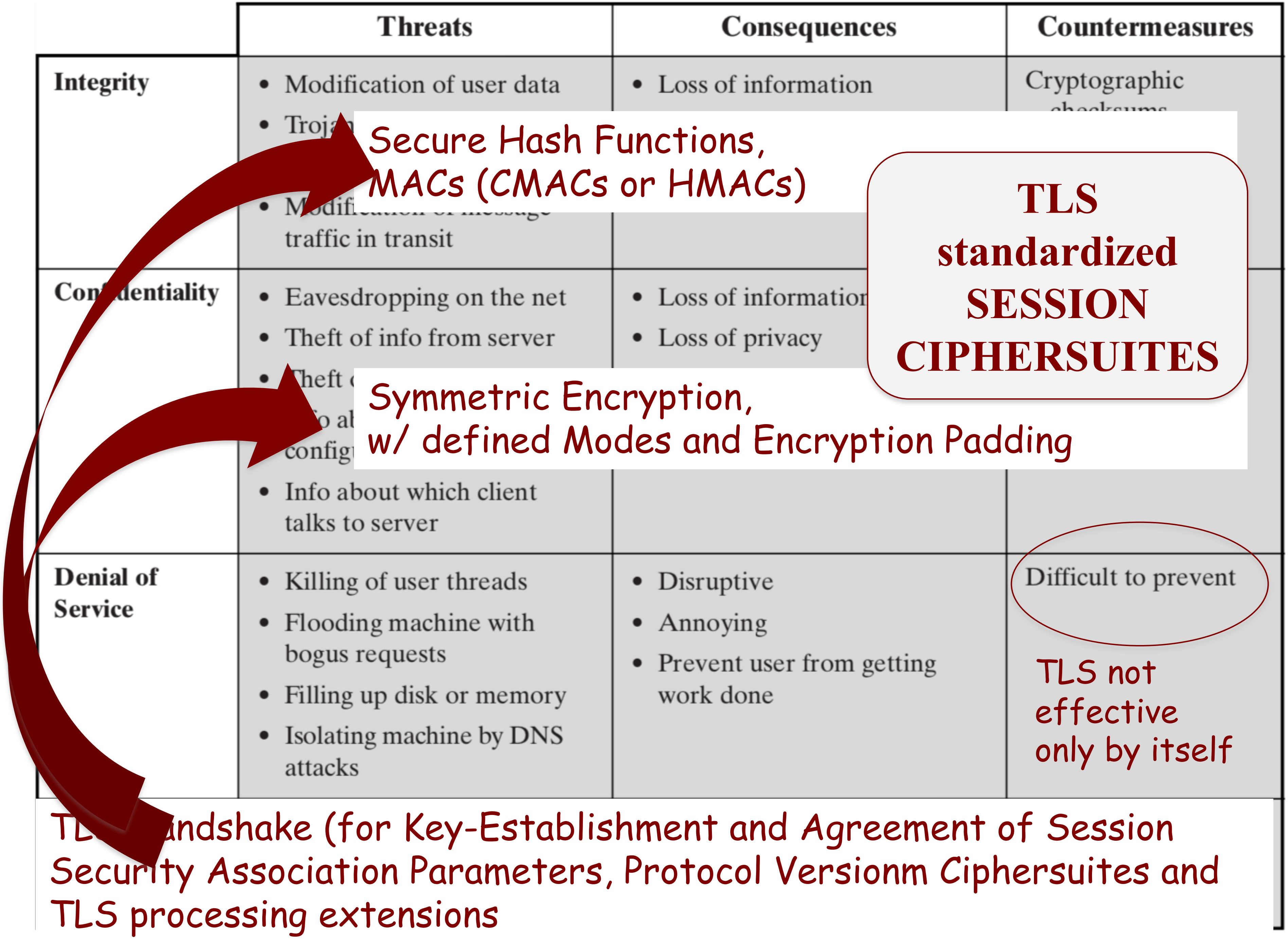
	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan • Modifi • Modification of message traffic in transit	• Loss of information	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft • Info ab config • Info about which client 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	• Impers users • Data f		

Secure Hash Functions,
MACs (CMACs or HMACs)

Symmetric Encryption, w/ defined Modes and Encryption Padding

X509v3 Certificates, Digital Signatures / Asymmetric Cryptography

	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 TLS not effective only by itself
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques



	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Troja • Modification of message traffic in transit	• Loss of information	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft c • o ab config • Info about which client talks to server	• Loss of information • Loss of privacy	
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

Secure Hash Functions,
MACs (CMACs or HMACs)

TLS standardized SESSION CIPHERSUITES

Symmetric Encryption,
w/ defined Modes and Encryption Padding

TLS not effective only by itself

TLS Handshake (for Key-Establishment and Agreement of Session Security Association Parameters, Protocol Versionm Ciphersuites and TLS processing extensions

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

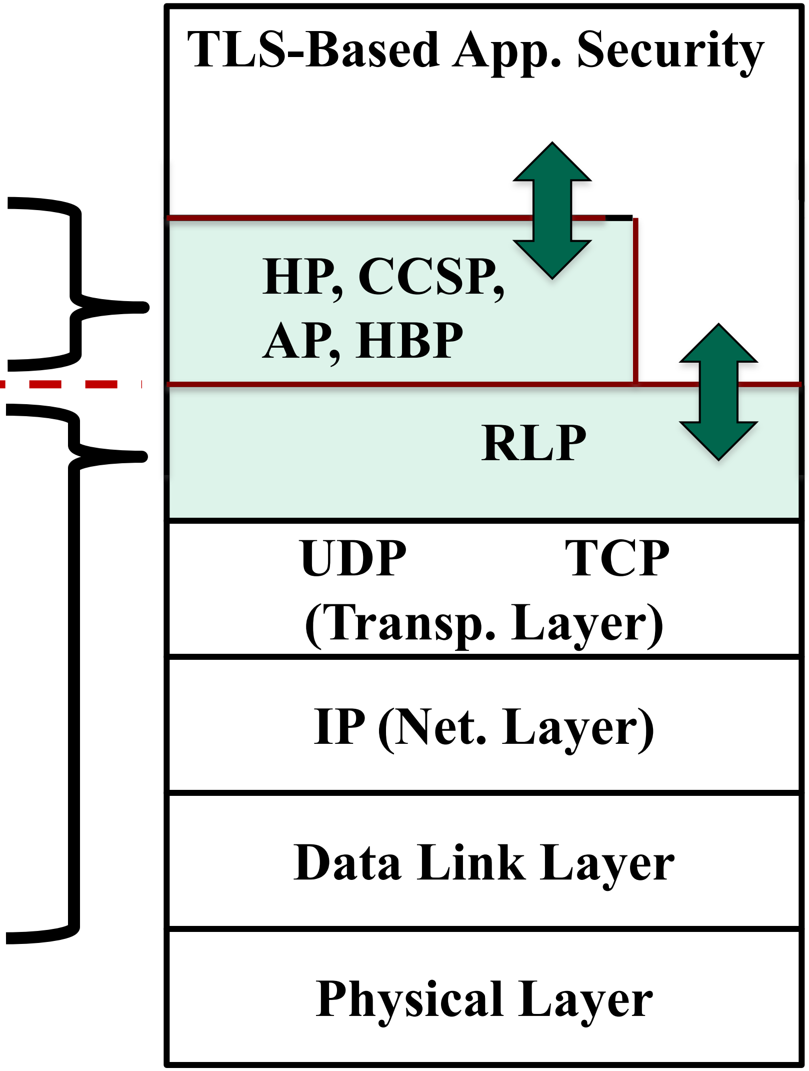
TLS-Stack and Role of TLS Sub-Protocols

Session:

- Establishment and Management of TLS Session Security Associations
  - Session-Context Parameters

Connection:

- Secure transport (for a peer-to-peer or client/server secure channel)
- Transient connections
- Connections are associated with one session



TLS-Stack and Role of TLS Sub-Protocols

HP: Handshake Protocol

- Authentication, Agreement and Establishment of Cryptographic Keys, Security Association Parameters and Extensions for TLS Sessions

AP: Alert Protocol

- Reaction to events and exceptions in TLS flows, aborting, resuming or restarting HP

CCSP: Change Cipher Spec. Protocol

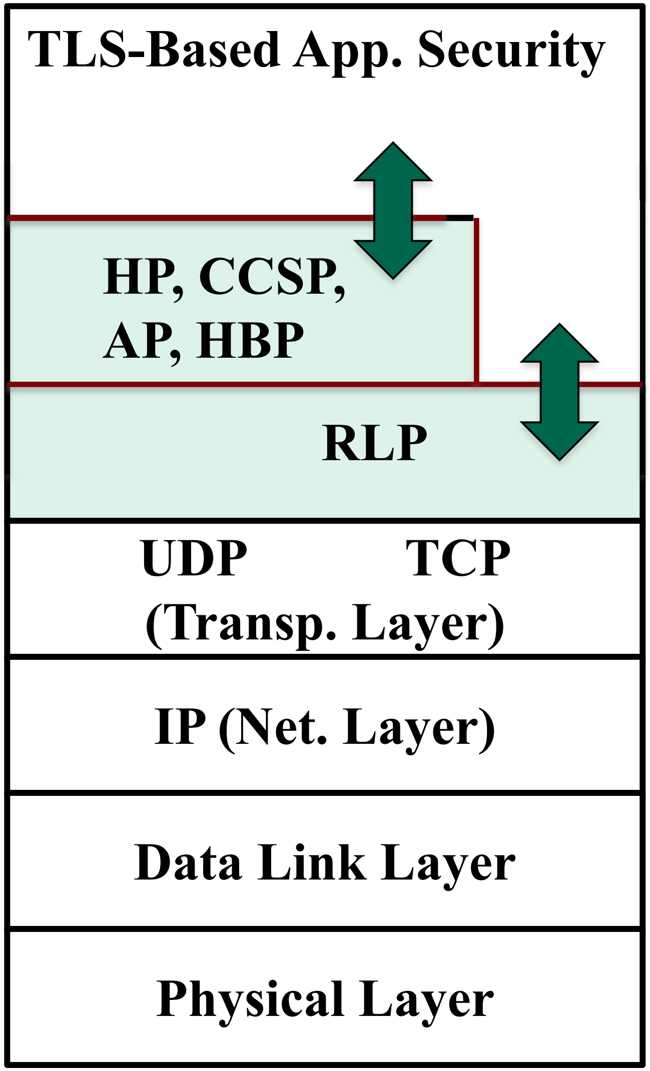
- Sync. of established session security parameters

Heartbeat Protocol

- Keep-Alive Control of established sessions

RLP: Record Layer Protocol

- Secure transport TLS payload format



TLS-Stack and Role of TLS Sub-Protocols

HP: Handshake Protocol

- Authentication, Agreement and Establishment of Cryptographic Keys, Security Association Parameters and Extensions for TLS Sessions

AP: Alert Protocol

- Reaction to events and exceptions in TLS flows, aborting, resuming or restarting HP

CCSP: Change Cipher Spec. Protocol

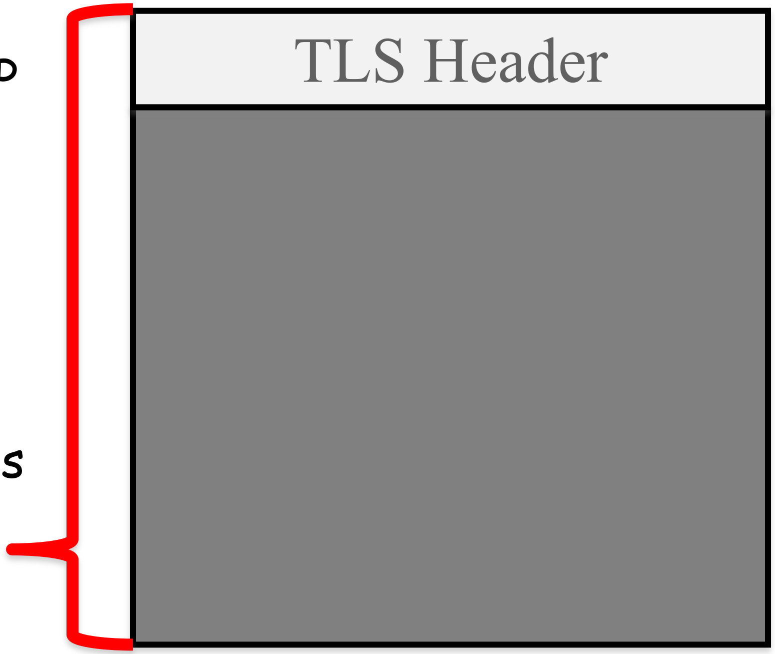
- Sync. of established session security parameters

Heartbeat Protocol

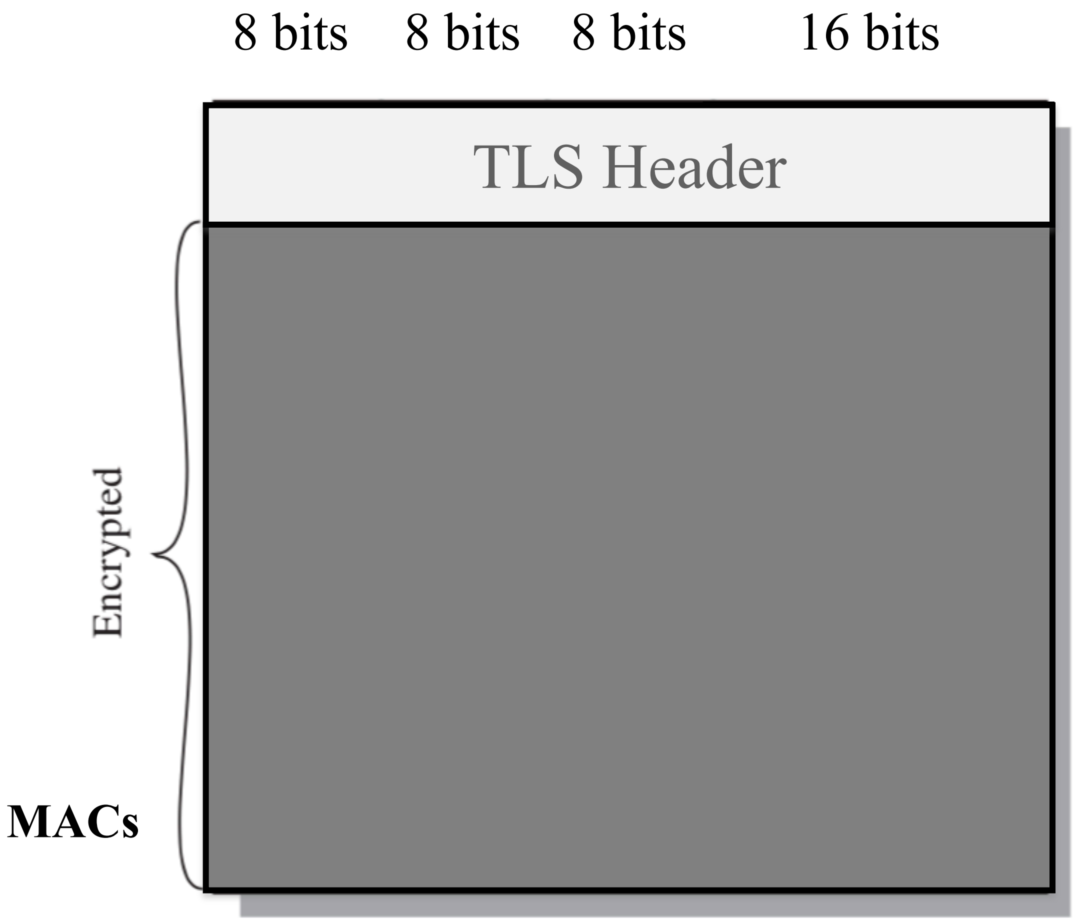
- Keep-Alive Control of established sessions

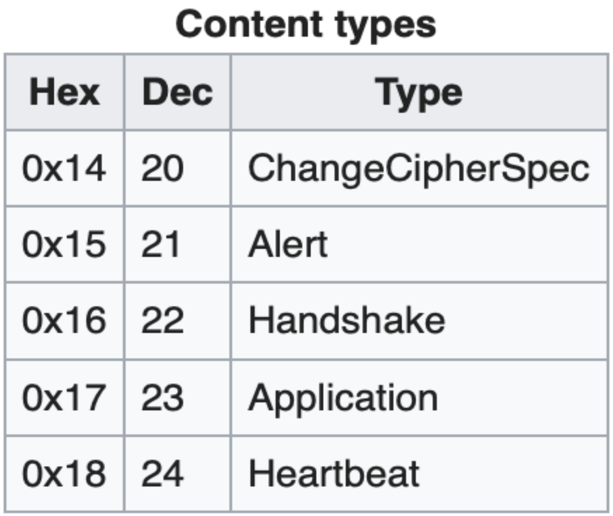
RLP: Record Layer Protocol

- Secure transport TLS payload format



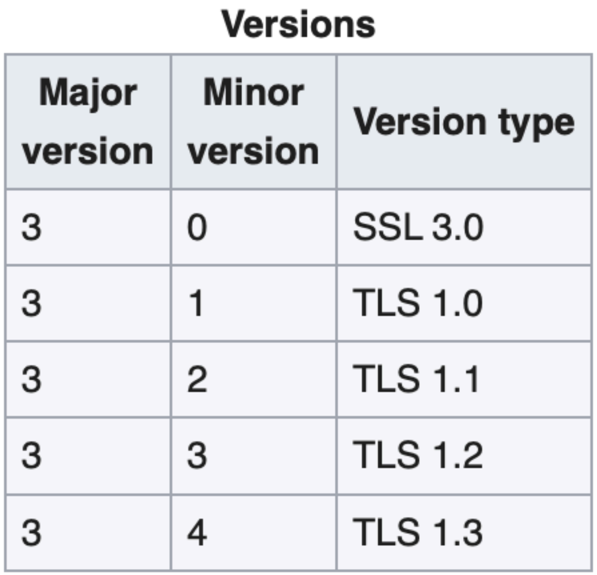
RLP Message Format





Content types

Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application
0x18	24	Heartbeat

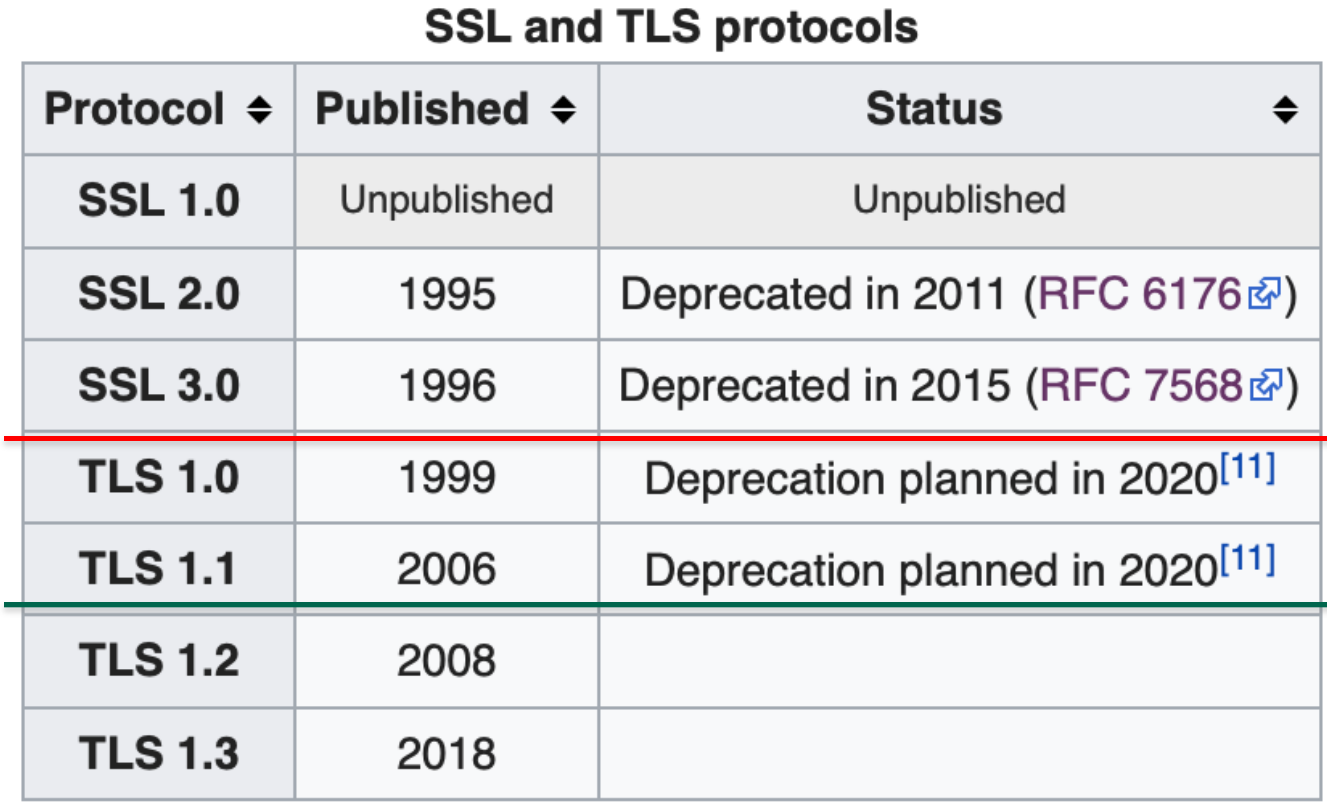


Versions

Major version	Minor version	Version type
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2
3	4	TLS 1.3

Generic Format:
TLS Header || { TLS Message Types || MAC }

Protocol Versions: TLS and SSL Protocols



SSL and TLS protocols

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecation planned in 2020[11]
TLS 1.1	2006	Deprecation planned in 2020[11]
TLS 1.2	2008	
TLS 1.3	2018	

Def. RFC 2246, Jan/99

Def. RFC 4346, Apr/06

Def. RFC 5246, Aug/08

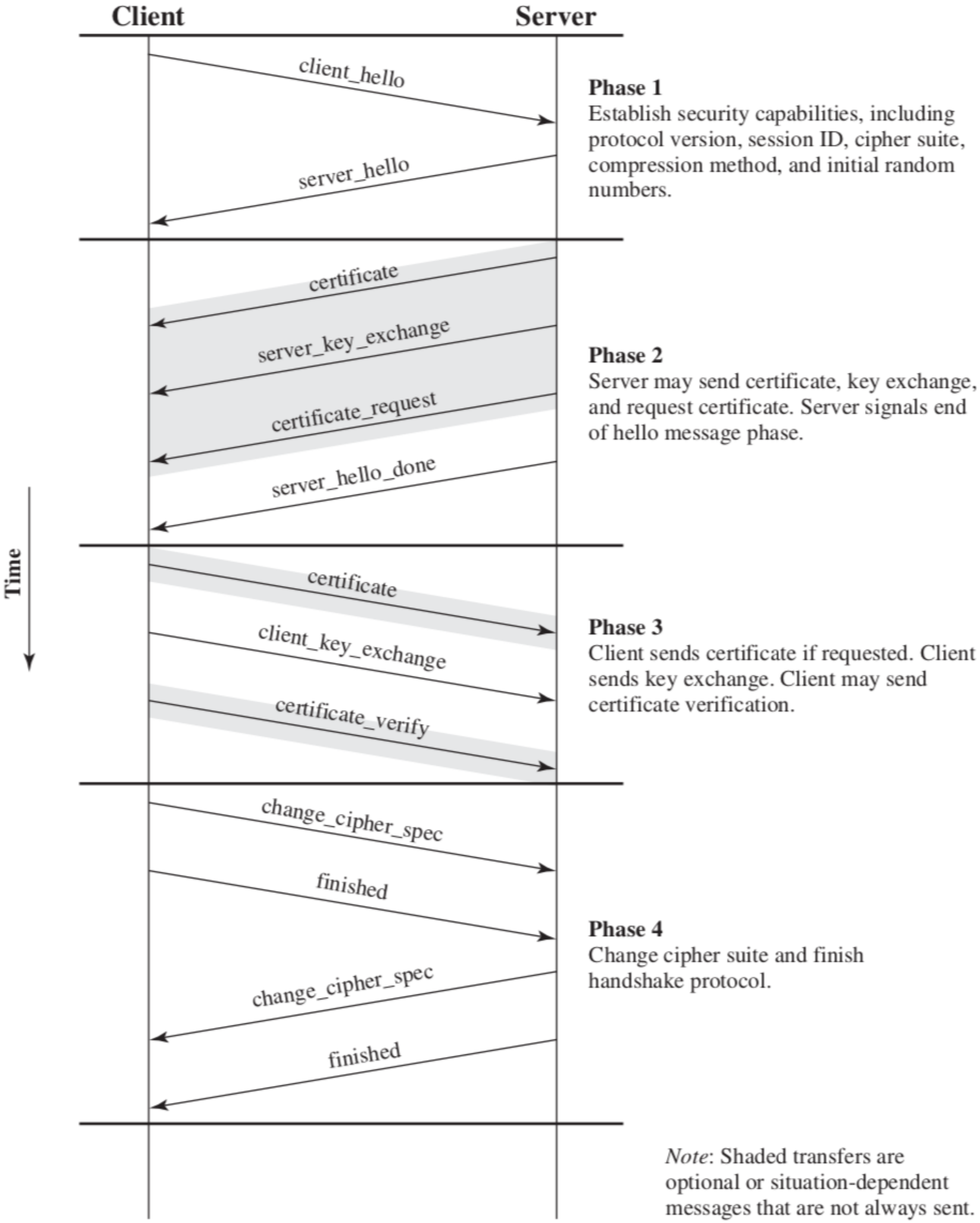
Def. RFC 8446, Aug/18

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - *Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB*
- TLS vs. HTTPS

TLS Handshake Flow

Generic Flow



Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

TLS Handshake Flow

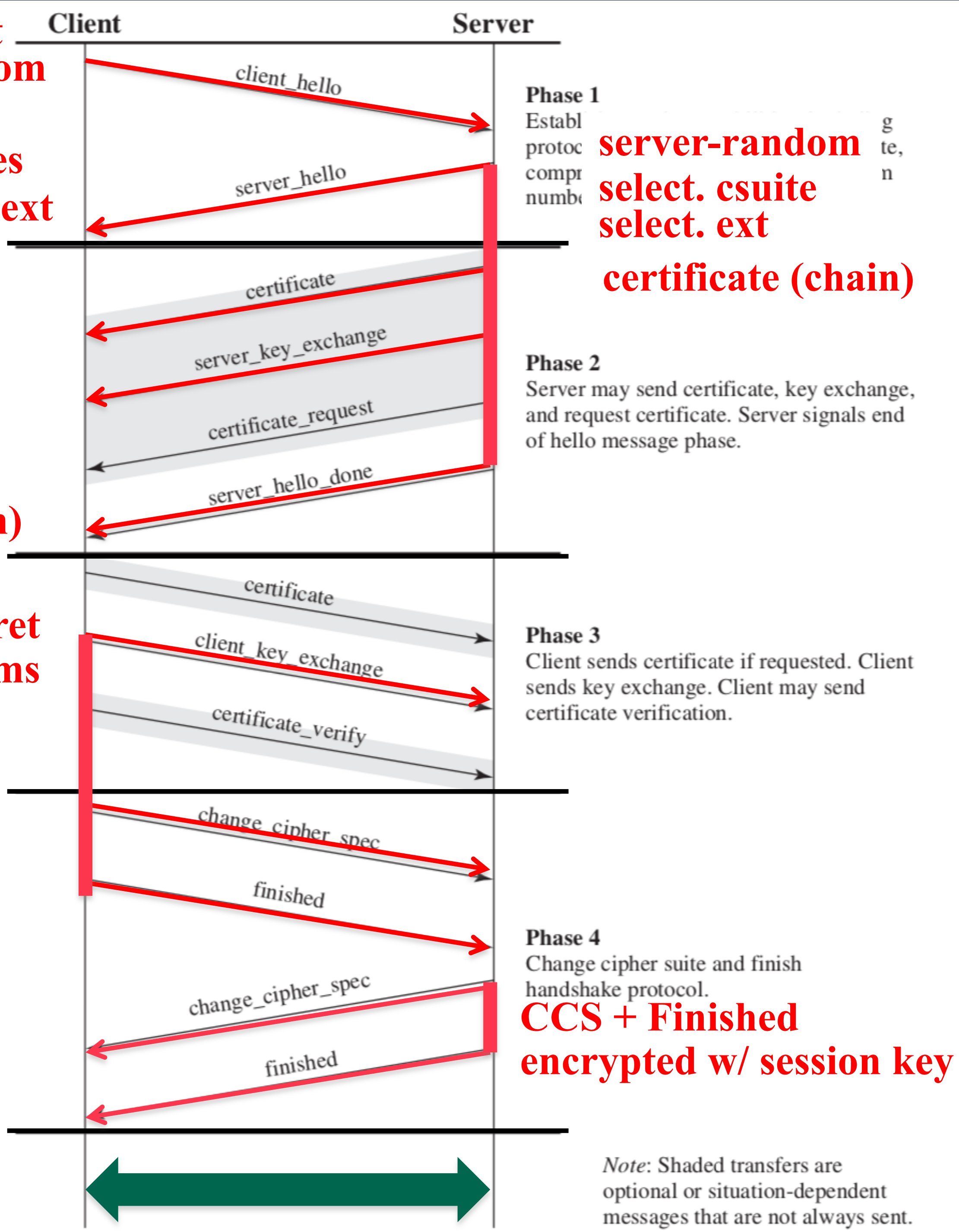
Server-Only Authentication

time

client random

prop. csuites

pref. ext



Phase 1
Establ
protoc
compr
numb

g
te,
n

server-random

select. csuite

select. ext

certificate (chain)

Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

certificate (chain) verification

PMS – Pre-Master-Secret or DH public params

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

CCS + Finished encrypted w/session key

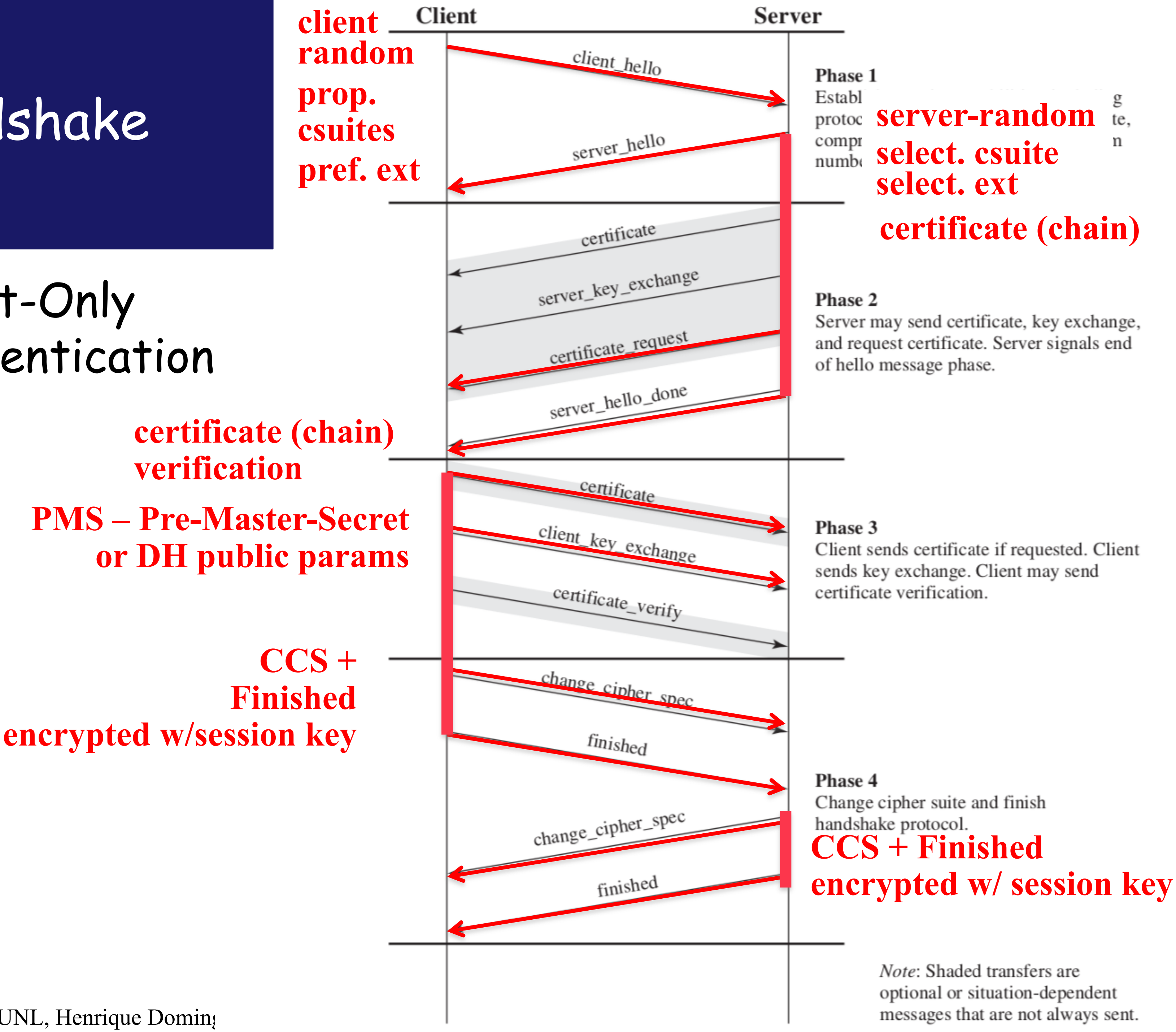
Phase 4
Change cipher suite and finish handshake protocol.

CCS + Finished encrypted w/ session key

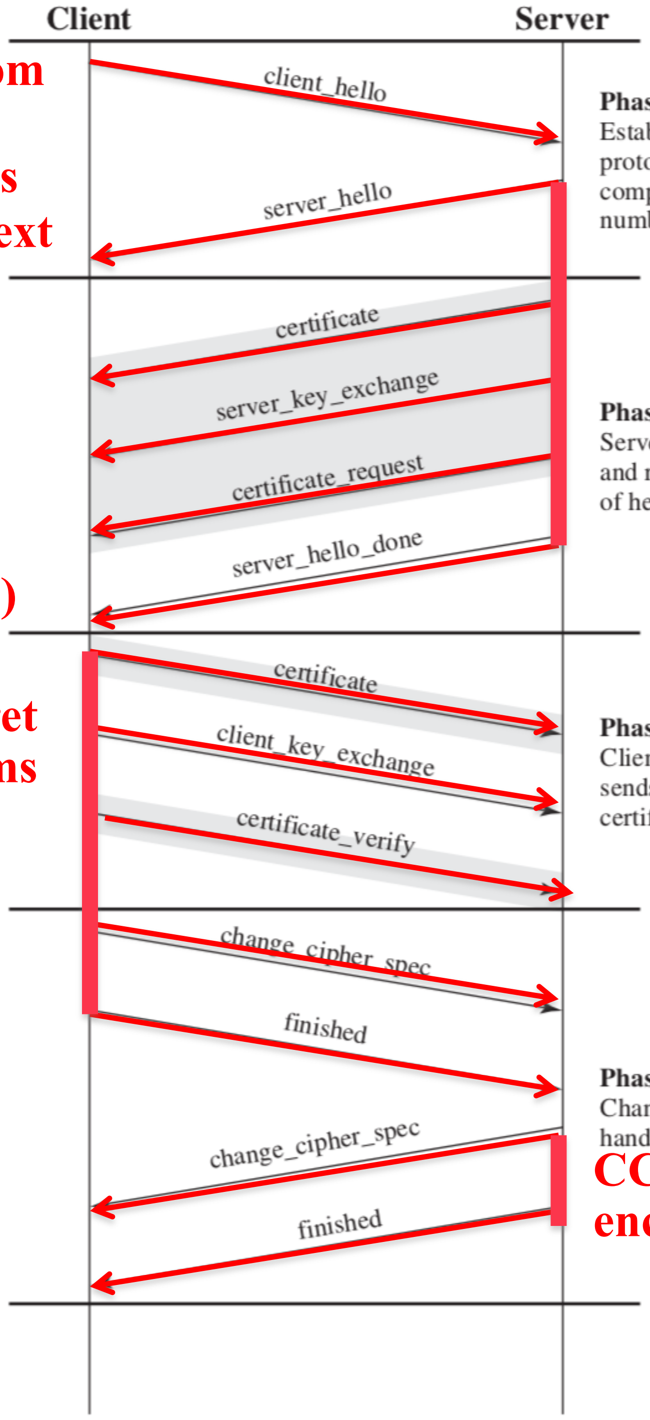
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

TLS Handshake Flow

Client-Only Authentication


Client
Server
client random
prop. csuites
pref. ext
client_hello
server_hello
Phase 1
Establ
g
protoc
te,
compr
n
numb
server-random
select. csuite
select. ext
certificate (chain)
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
certificate (chain) verification
PMS – Pre-Master-Secret or DH public params
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
CCS + Finished encrypted w/session key
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
CCS + Finished encrypted w/ session key
change_cipher_spec
finished
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

TLS Handshake Flow



client random
prop. csuites
pref. ext

Phase 1
Establ g
protoc te,
compr n
numb

server-random
select. csuite
select. ext

certificate (chain)

Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Mutual Authentication

certificate (chain) verification

PMS – Pre-Master-Secret or DH public params

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

CCS + Finished encrypted w/session key

Phase 4
Change cipher suite and finish handshake protocol.

CCS + Finished encrypted w/ session key

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - *Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB*
- TLS vs. HTTPS

TLS Level Programming Approach

TLS Programming Level APIs

Examples:

- Java JSSE (Java Secure Socket Extension)

Java 8 - https://docs.oracle.com/javase/10/security/java-secure-socket-extension-jsse-reference-guide.htm#JSSEC-GUID-93DEEE16-0B70-40E5-BBE7-55C3FD432345

…

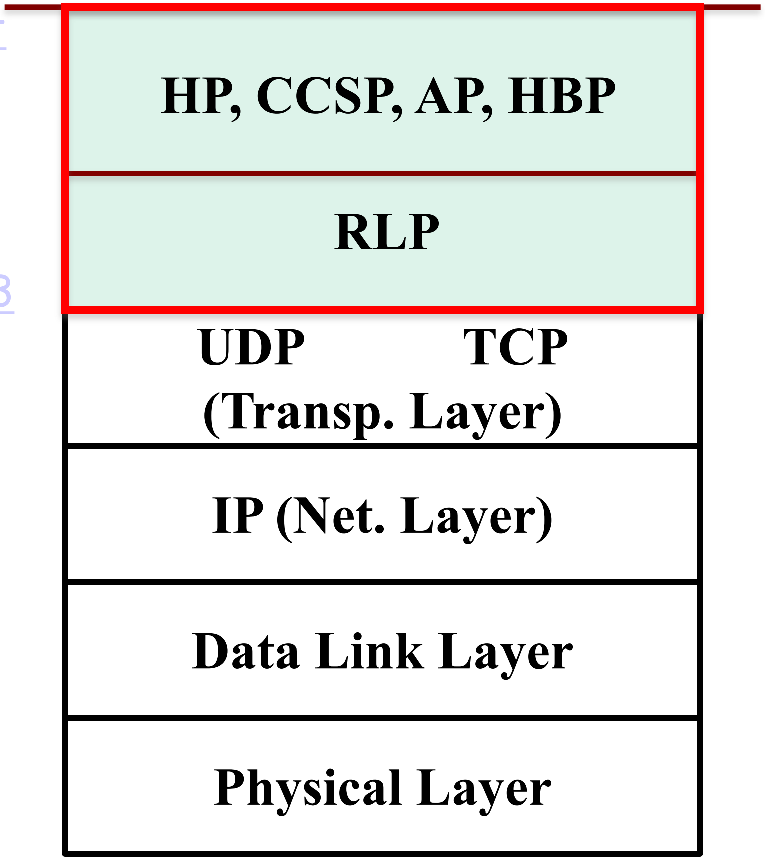
…

…

Java 13 - https://docs.oracle.com/en/java/javase/13/security/java-secure-socket-extension-jsse-reference-guide.html#GUID-93DEEE16-0B70-40E5-BBE7-55C3FD432345

- Openssl library for TLS Sockets (C, C++): https://www.openssl.org
- MS TLS .NET Framework https://docs.microsoft.com/en-us/dotnet/framework/network-programming/tls

TLS-Enabled Prigramming Abstraction:
TLS-Libraries,
Franmeworks and APIs



HP, CCSP, AP, HBP

RLP

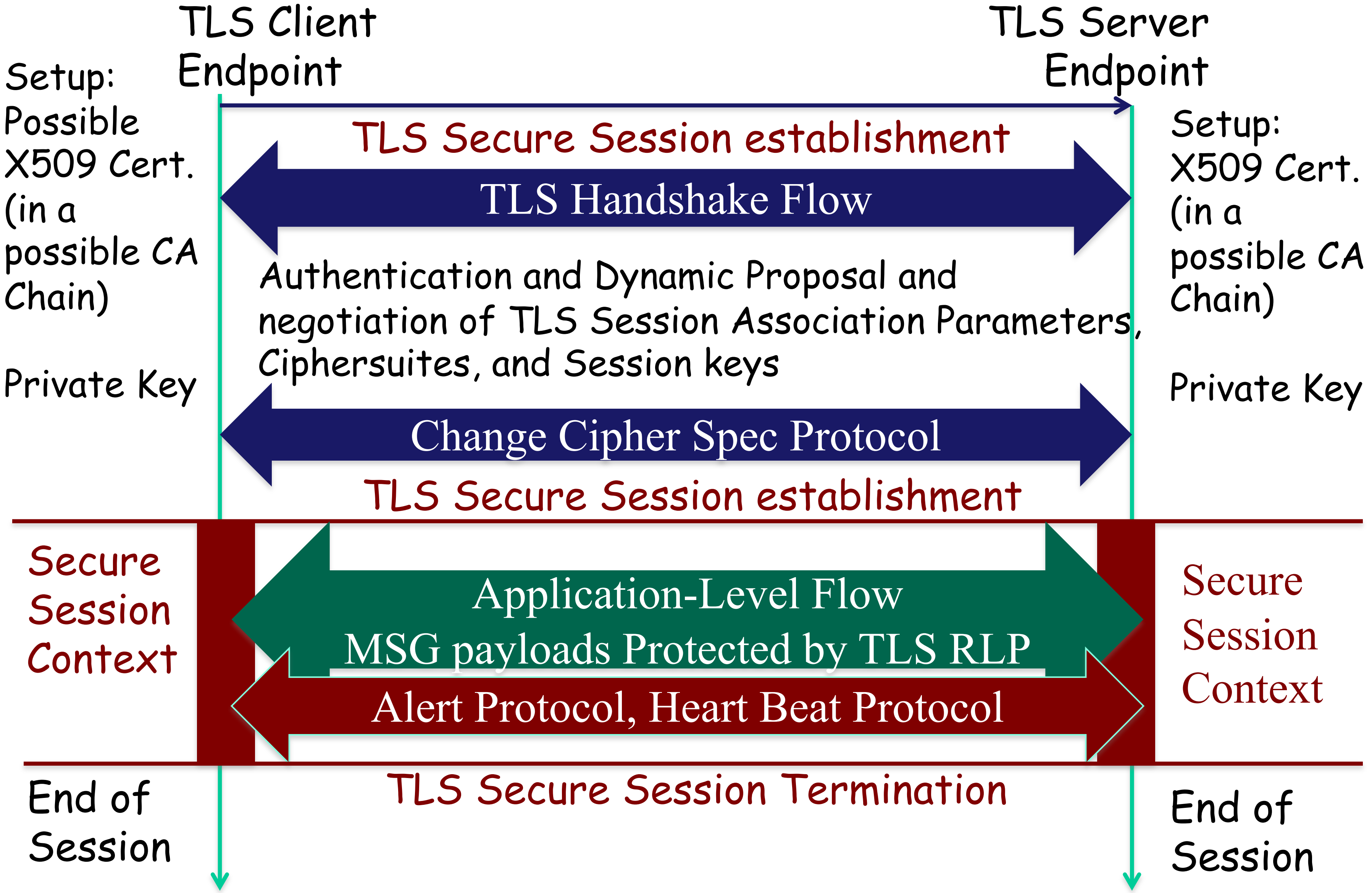
UDP TCP
(Transp. Layer)

IP (Net. Layer)

Data Link Layer

Physical Layer

TLS Operation and Generic Traffic Flow



TLS Client Endpoint

TLS Server Endpoint

Setup: Possible X509 Cert. (in a possible CA Chain)

Private Key

Setup: X509 Cert. (in a possible CA Chain)

Private Key

TLS Secure Session establishment

TLS Handshake Flow

Authentication and Dynamic Proposal and negotiation of TLS Session Association Parameters, Ciphersuites, and Session keys

Change Cipher Spec Protocol

TLS Secure Session establishment

Secure Session Context

Application-Level Flow

MSG payloads Protected by TLS RLP

Alert Protocol, Heart Beat Protocol

Secure Session Context

TLS Secure Session Termination

End of Session

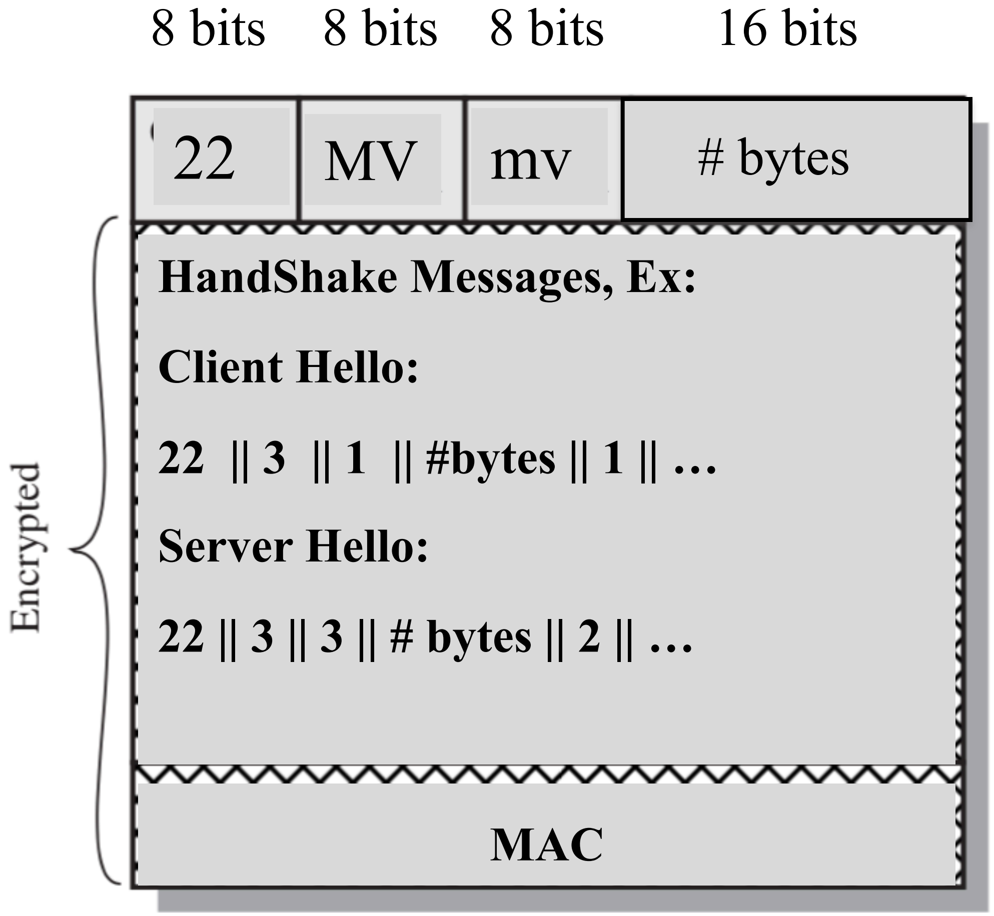
End of Session

TLS Operation and Flexibility Issues
(imply on possible different required setups)

- Client TLS and Server TLS endpoints can map or not Client Side and Server Side App. Endpoints
  - In TLS a Client TLS Endpoint initiates the Handshake Process

    … But it can be the Server Side App Endpoint
- TLS protocol can be supported in different versions
- Peer-Authentication of Endpoints can be:
  - Unilateral Authentication
    - Server Only or Client Only Authentication
  - Mutual Authentication
    - Client and Server mutually authenticated
- Peer-Authentication Type and Key + SA Establihment can be different, according to the negotiated handshake
- Agreed TLS ciphersuites (for all the cryptographic methids that will be used) depend on the handshake negotiation

Ex: Handshake / RLP Message Format



8 bits 8 bits 8 bits 16 bits

22 | MV | mv | # bytes

HandShake Messages, Ex:

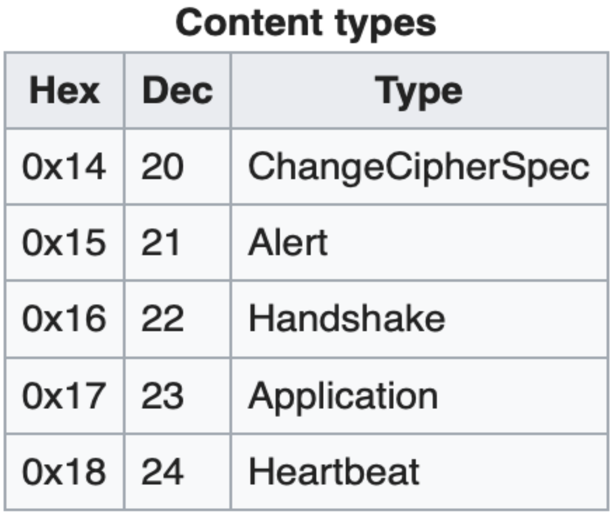
Client Hello:

22 || 3 || 1 || #bytes || 1 || …

Server Hello:

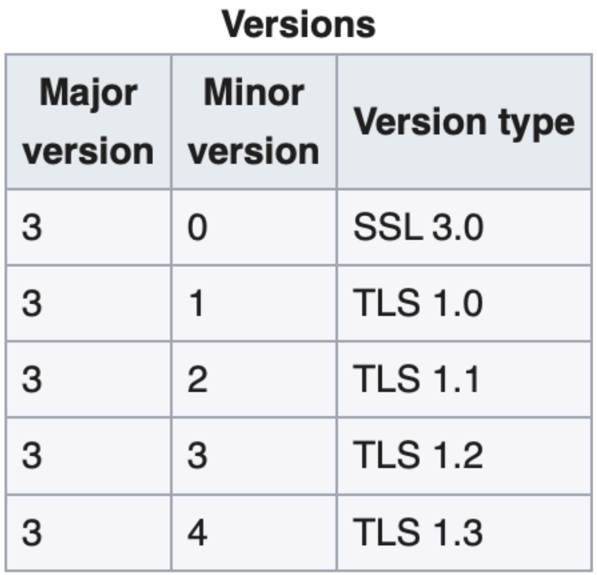
22 || 3 || 3 || # bytes || 2 || …

MAC



Content types

Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application
0x18	24	Heartbeat

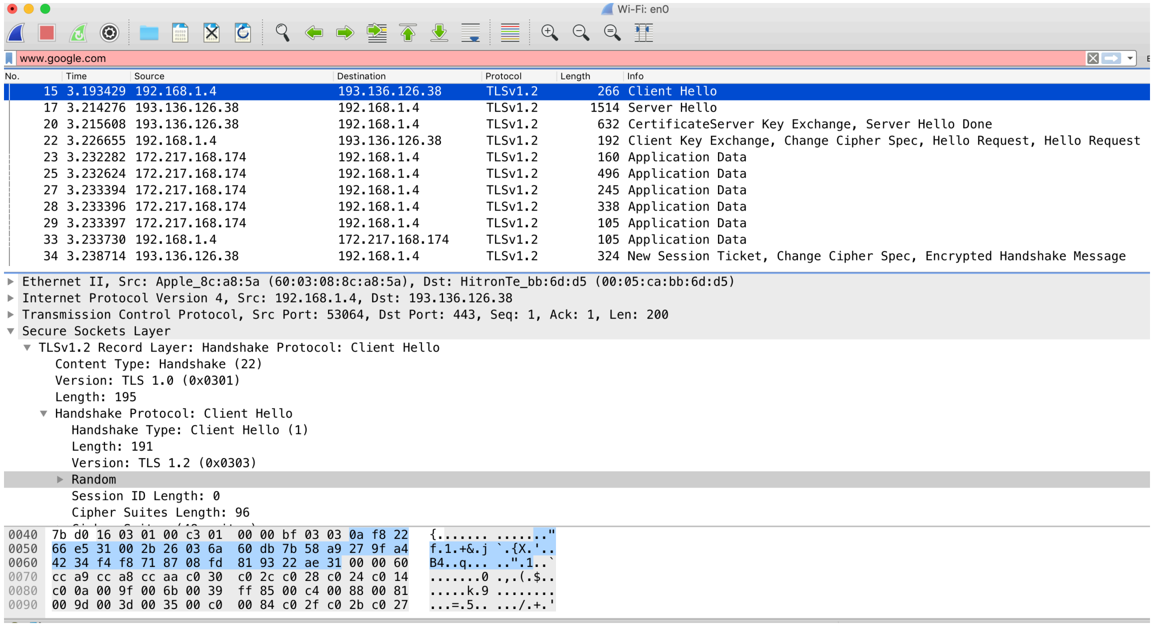


Versions

Major version	Minor version	Version type
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2
3	4	TLS 1.3

TLS Traffic Flow Analysis: Wireshark
(can use a TLS client: browser or openssl tool and TLS server)

Suggestion:
Analyze the TLS Traffic Flow in a Real TLS Trace: Ex: TLS 1.0, TLS 1.2, TLS 1.3 using the openssl and wireshark tools

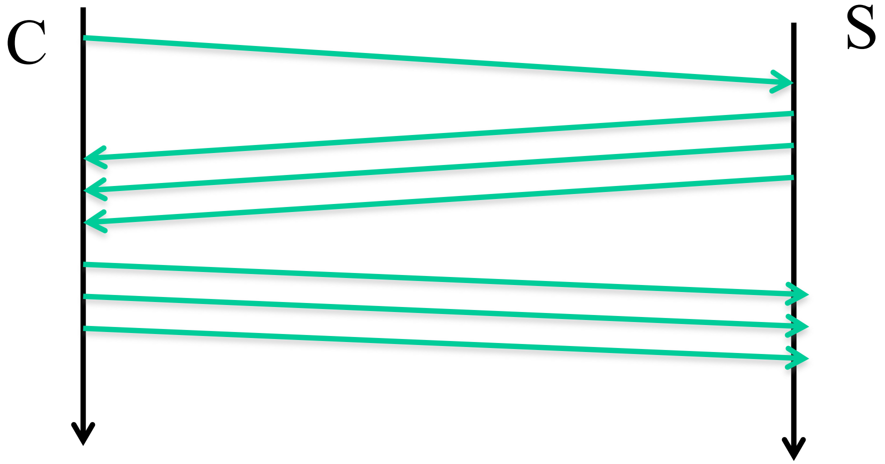


```
www.google.com
No.  Time      Source          Destination     Protocol  Length Info
  15 3.193429 192.168.1.4     193.136.126.38  TLSv1.2    266 Client Hello
  17 3.214276 193.136.126.38  192.168.1.4     TLSv1.2   1514 Server Hello
  20 3.215608 193.136.126.38  192.168.1.4     TLSv1.2    632 CertificateServer Key Exchange, Server Hello Done
  22 3.226655 192.168.1.4     193.136.126.38  TLSv1.2    192 Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
  23 3.232282 172.217.168.174 192.168.1.4     TLSv1.2    160 Application Data
  25 3.232624 172.217.168.174 192.168.1.4     TLSv1.2    496 Application Data
  27 3.233394 172.217.168.174 192.168.1.4     TLSv1.2    245 Application Data
  28 3.233396 172.217.168.174 192.168.1.4     TLSv1.2    338 Application Data
  29 3.233397 172.217.168.174 192.168.1.4     TLSv1.2    105 Application Data
  33 3.233730 192.168.1.4     172.217.168.174 TLSv1.2    105 Application Data
  34 3.238714 193.136.126.38  192.168.1.4     TLSv1.2    324 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
```

```
▶ Ethernet II, Src: Apple_8c:a8:5a (60:03:08:8c:a8:5a), Dst: HitronTe_bb:6d:d5 (00:05:ca:bb:6d:d5)
▶ Internet Protocol Version 4, Src: 192.168.1.4, Dst: 193.136.126.38
▶ Transmission Control Protocol, Src Port: 53064, Dst Port: 443, Seq: 1, Ack: 1, Len: 200
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 195
    ▼ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 191
        Version: TLS 1.2 (0x0303)
      ▶ Random
        Session ID Length: 0
        Cipher Suites Length: 96
```

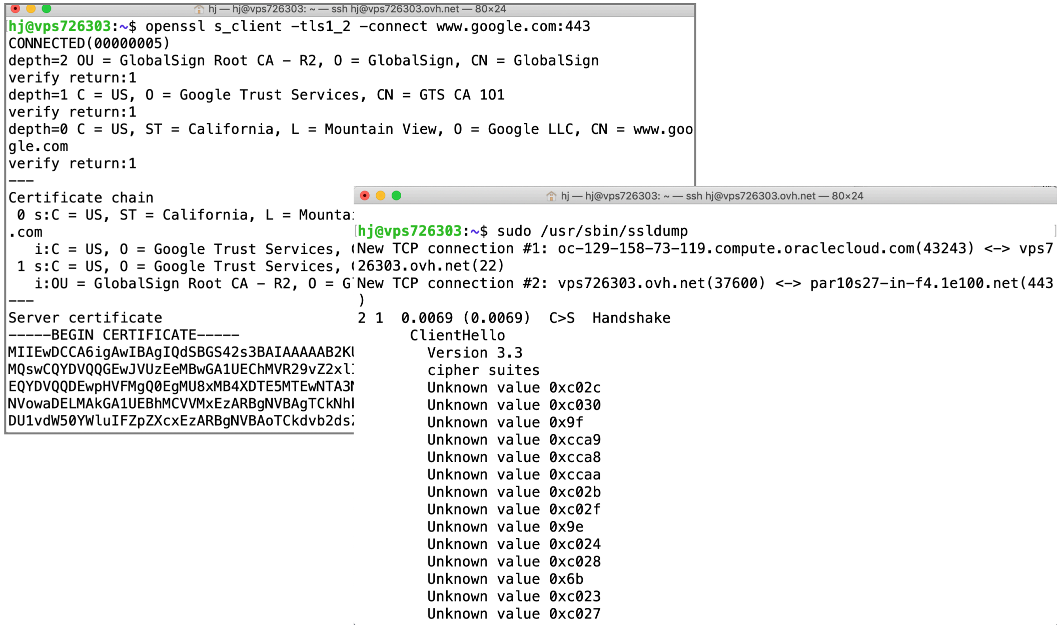
```
0040  7b d0 16 03 01 00 c3 01  00 00 bf 03 03 0a f8 22   {....... ......."
0050  66 e5 31 00 2b 26 03 6a  60 db 7b 58 a9 27 9f a4   f.1.+&.j `.{X.'..
0060  42 34 f4 f8 71 87 08 fd  81 93 22 ae 31 00 00 60   B4..q... ..".1..`
0070  cc a9 cc a8 cc aa c0 30  c0 2c c0 28 c0 24 c0 14   .......0 .,.(.$..
0080  c0 0a 00 9f 00 6b 00 39  ff 85 00 c4 00 88 00 81   .....k.9 ........
0090  00 9d 00 3d 00 35 00 c0  00 84 c0 2f c0 2b c0 27   ...=.5.. .../.+.'
```

SEE LAB 6
Will do this in LAB 6



TLS Traffic Flow Analysis: openssl + ssldump

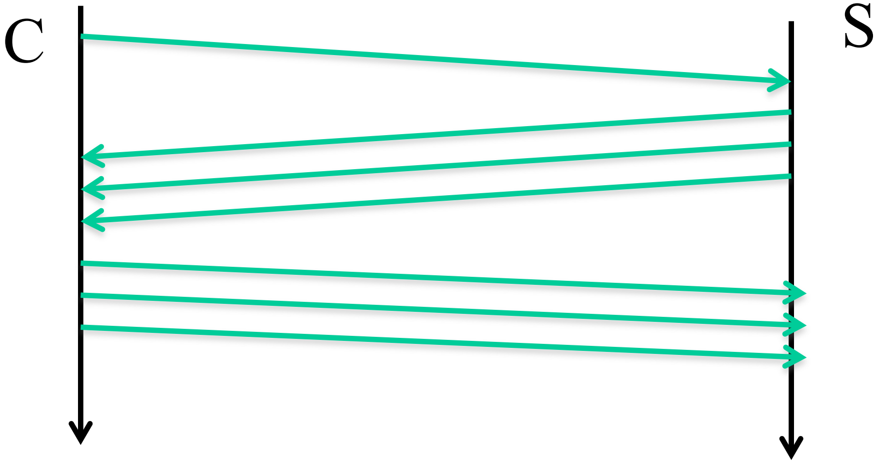
Suggestion:
Analyze the TLS Traffic Flow in a Real TLS Trace:
Ex: TLS 1.0, TLS 1.2, TLS 1.3 using the openssl and wireshark tools


hj — hj@vps726303: ~ — ssh hj@vps726303.ovh.net — 80×24
hj — hj@vps726303: ~ — ssh hj@vps726303.ovh.net — 80×24

```
hj@vps726303:~$ openssl s_client -tls1_2 -connect www.google.com:443
CONNECTED(00000005)
depth=2 OU = GlobalSign Root CA - R2, O = GlobalSign, CN = GlobalSign
verify return:1
depth=1 C = US, O = Google Trust Services, CN = GTS CA 1O1
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google LLC, CN = www.goo
gle.com
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = Mounta
.com
   i:C = US, O = Google Trust Services, 
 1 s:C = US, O = Google Trust Services, 
   i:OU = GlobalSign Root CA - R2, O = G
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEwDCCA6igAwIBAgIQdSBGS42s3BAIAAAAAB2K
MQswCQYDVQQGEwJVUzEeMBwGA1UEChMVR29vZ2xl
EQYDVQQDEwpHVFMgQ0EgMU8xMB4XDTE5MTEwNTA3
NVowaDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
DU1vdW50YWluIFZpZXcxEzARBgNVBAoTCkdvb2ds
```

```
hj@vps726303:~$ sudo /usr/sbin/ssldump
New TCP connection #1: oc-129-158-73-119.compute.oraclecloud.com(43243) <-> vps7
26303.ovh.net(22)
New TCP connection #2: vps726303.ovh.net(37600) <-> par10s27-in-f4.1e100.net(443
)
2 1  0.0069 (0.0069)  C>S  Handshake
      ClientHello
        Version 3.3
        cipher suites
        Unknown value 0xc02c
        Unknown value 0xc030
        Unknown value 0x9f
        Unknown value 0xcca9
        Unknown value 0xcca8
        Unknown value 0xccaa
        Unknown value 0xc02b
        Unknown value 0xc02f
        Unknown value 0x9e
        Unknown value 0xc024
        Unknown value 0xc028
        Unknown value 0x6b
        Unknown value 0xc023
        Unknown value 0xc027
```

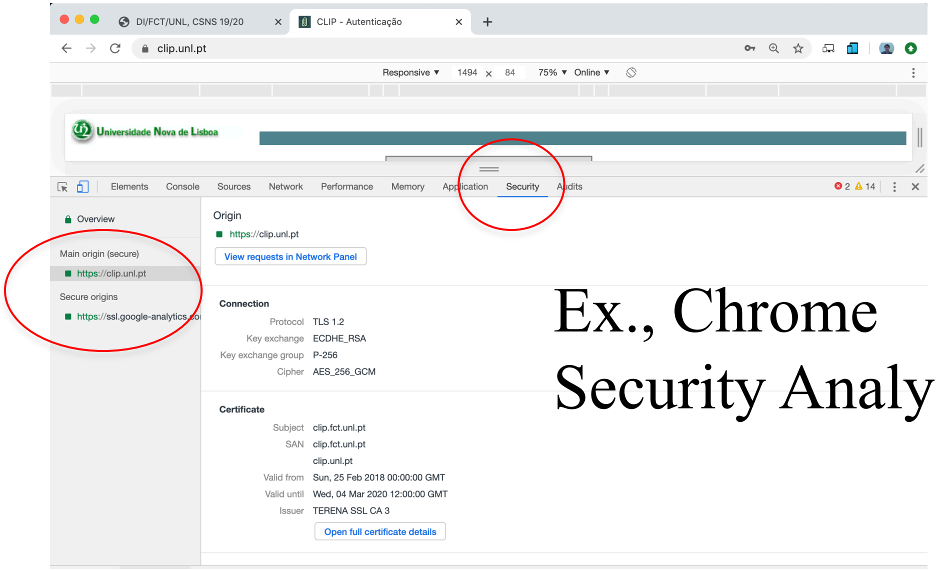
SEE LAB 6
Will do this in LAB 6



TLS Traffic Flow Analysis:

Security Analysis w/ your Browser Development Tools

Suggestion:
Analyze the TLS Traffic Flow in a Real TLS Trace: Ex: TLS 1.0, TLS 1.2, TLS 1.3 using the openssl and wireshark tools



Main origin (secure)
- https://clip.unl.pt

Secure origins
- https://ssl.google-analytics.co

Origin
- https://clip.unl.pt

Connection

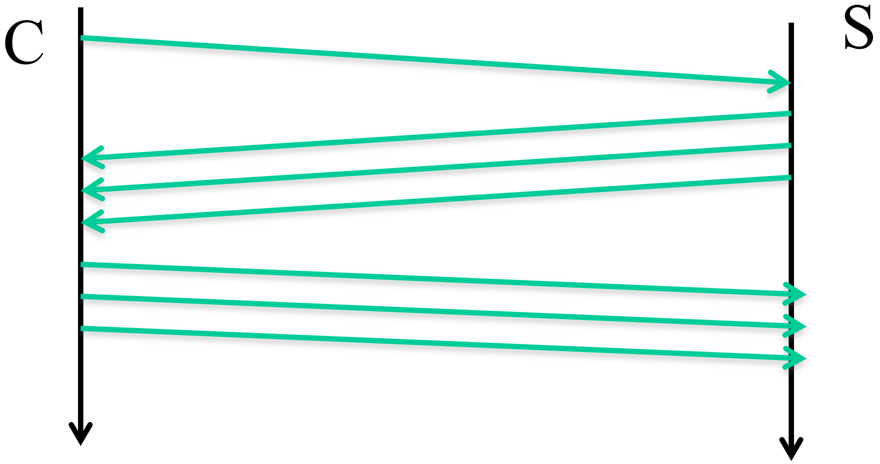
Protocol	TLS 1.2
Key exchange	ECDHE_RSA
Key exchange group	P-256
Cipher	AES_256_GCM

Certificate

Subject	clip.fct.unl.pt
SAN	clip.fct.unl.pt clip.unl.pt
Valid from	Sun, 25 Feb 2018 00:00:00 GMT
Valid until	Wed, 04 Mar 2020 12:00:00 GMT
Issuer	TERENA SSL CA 3

Ex., Chrome Security Analysis

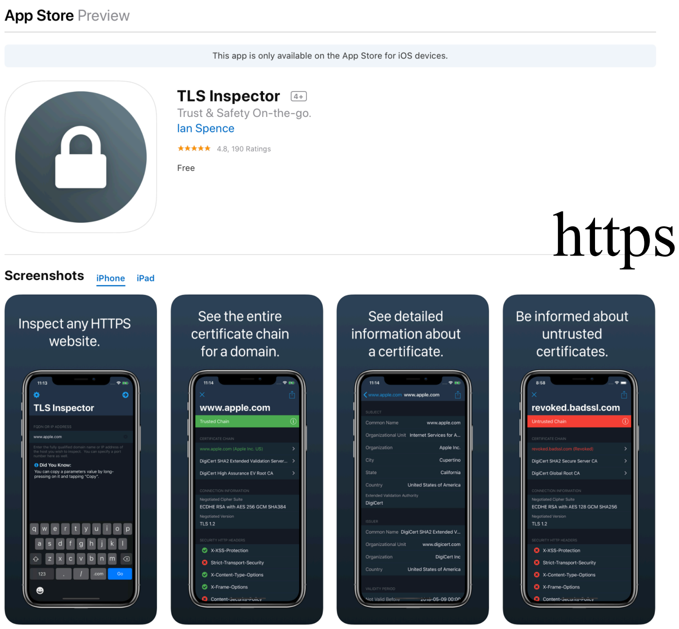
SEE LAB 6
Will do this in LAB 6



TLS Traffic Flow Analysis

Other interesting tools: mobile inspection

Suggestion:
Analyze the TLS Traffic Flow in a Real TLS Trace: Ex: TLS 1.0, TLS 1.2, TLS 1.3 using the openssl and wireshark tools



App Store Preview

This app is only available on the App Store for iOS devices.

TLS Inspector 4+
Trust & Safety On-the-go.
Ian Spence
★★★★★ 4.8, 190 Ratings
Free

Screenshots iPhone iPad

Inspect any HTTPS website.

See the entire certificate chain for a domain.

See detailed information about a certificate.

Be informed about untrusted certificates.

https://tlsinspector.com

AppStore
TLSInspector

https://github.com/google/nogotofai
l
https://source.android.com/security

GoogleStore
nogotofail

Handshake Types for Key & SA Establishment

- RSA: RSA Signatures + RSA encryption envelopes
- ECDSA: EC DSA Signatures + ECC Envelopes
- EDH: Ephemeral authenticated Diffie Hellman Agreement, w/ RSA or DSA Signatures
- EC-EDH or EC-DHE: Ephemeral authenticated Diffie Hellman Agreement, w/ EC-DSA Signatures

Usual methods

Very specific use

- SRP: Secure Remote Password Protocol
- PSK: Pre-Shared Keys

- FDH (Fixed Diffie Hellman): Fixed authenticated Diffie Hellman Agreement, w/ Certificates of DH-Public Numbers
- EC-FDH or EC-DH: Fixed authenticated Diffie Hellman Agreement, w/ EC-DSA Signatures

- No Authentication
- ADH (Anonymous Diffie Hellman)
- Fortezza

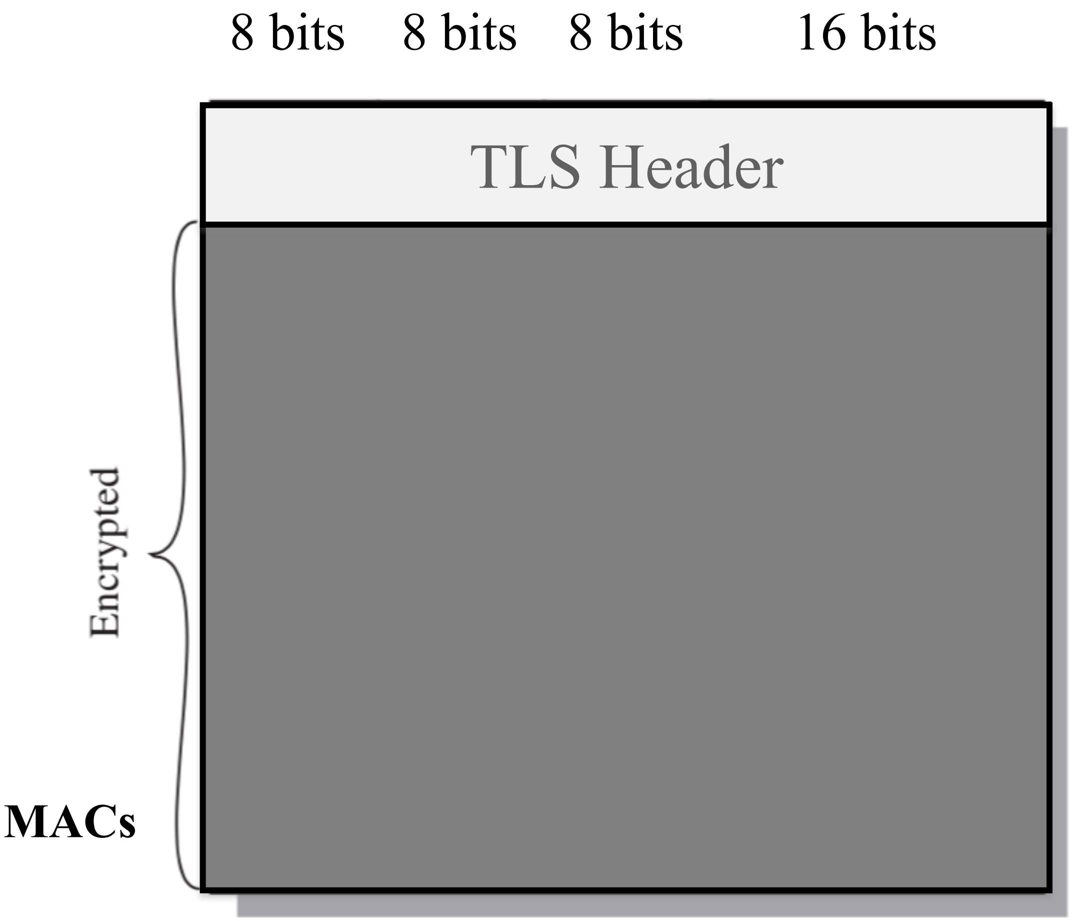
Not used today for Security and practical reasons

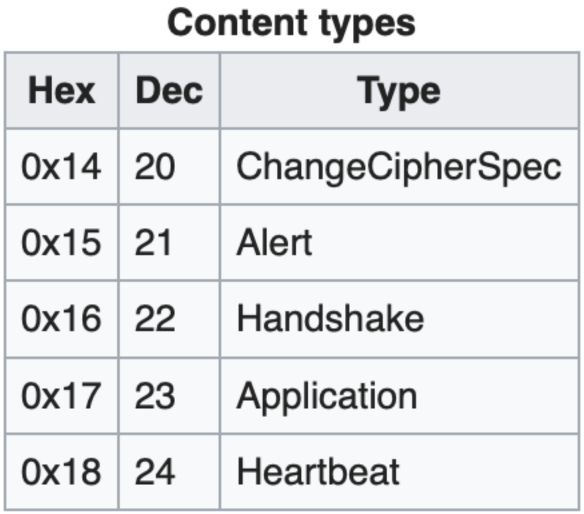
Standardized Ciphersuites: Support vs. Enabling
Ex., see openssl ciphers or TLS client proposed ciphersuites

- Combinations of the cryptographic methods for the handshake negotiation, usually represented in the following way (example):

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc14)
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc14)
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc13)
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc15)
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f)
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
… etc

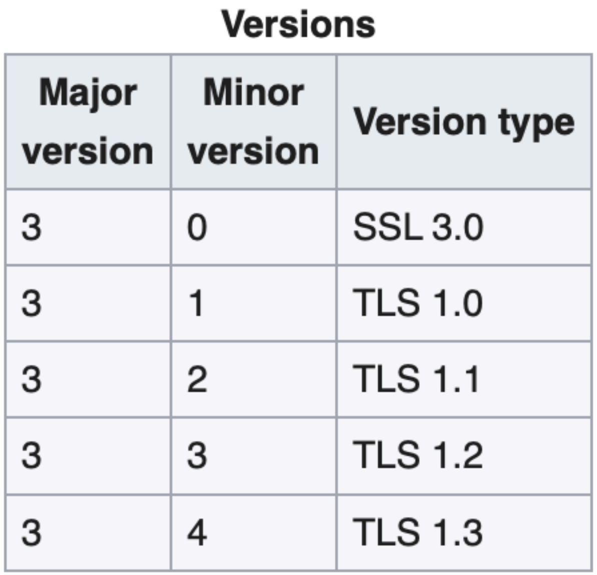
RLP Message Format





Content types

Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application
0x18	24	Heartbeat

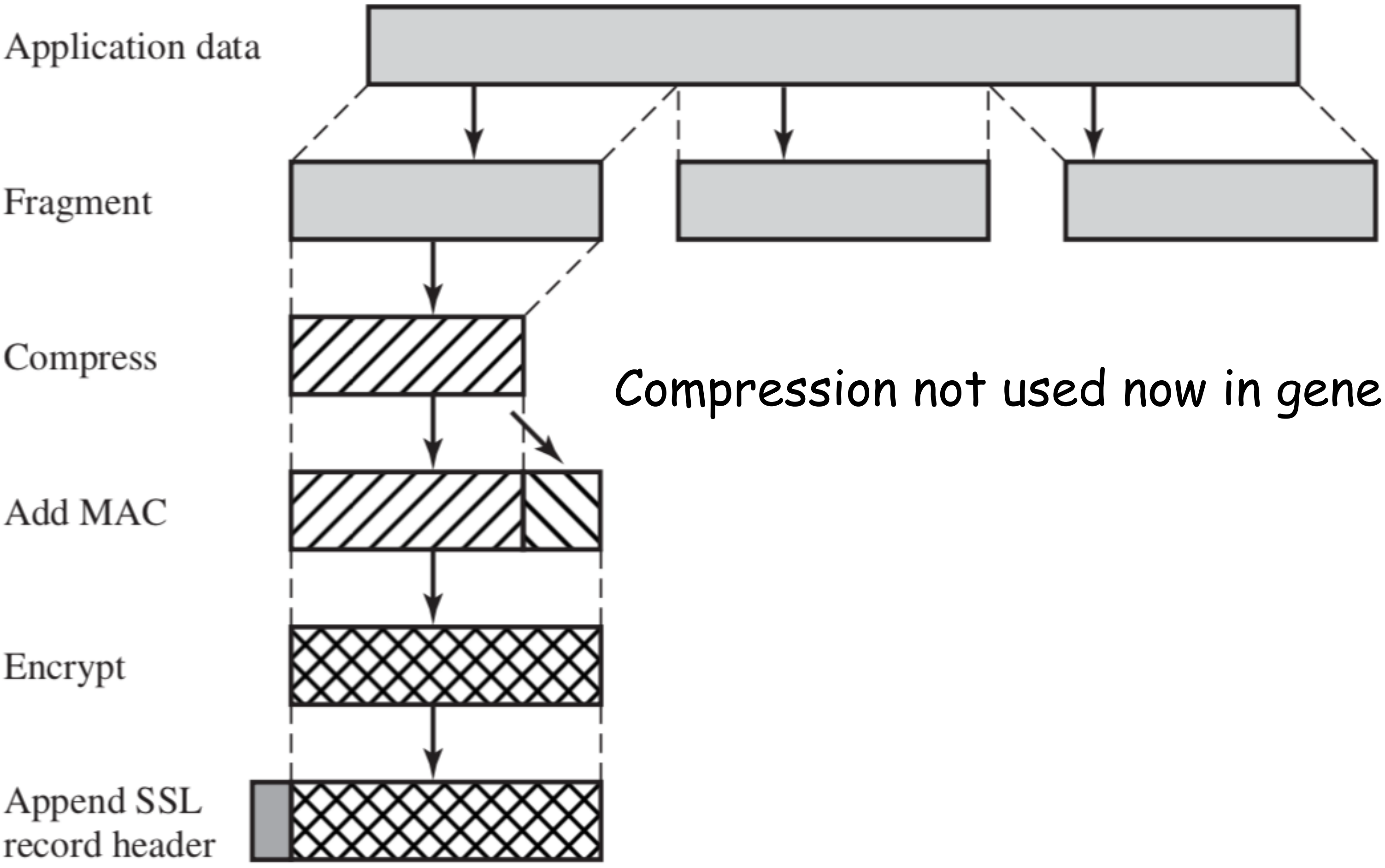


Versions

Major version	Minor version	Version type
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2
3	4	TLS 1.3

TLS: TLP - Record Layer Protocol

Message Processing in Endpoints



Compression not used now in general

RLP encapsulation format

Hands-On TLS Analysis

Hands-On TLS Sessions

Security Inspection and Traffic Analysis

- TKLS Traffic using Wireshark tool

(see also other practical observations in the context in Labs: Lab 8)

TLS Analysis: openssl tool and JRE instrumentation (examples, see also in LABs)

openssl tool (example):

```
$ openssl s_client -connect www.gmail.com:443
```

Security enforcement (ex., TLS protocol version, Client-enabled/proposed Ciphersuites)

```
$ openssl ciphers
$ openssl s_client -connect www.gmail.com:443 -tls1_3 –cipher
TLS_AES_256_GCM_SHA384
… etc
```

JRE / TLS Runtime Instrumentation

```
$ java -Djavax.net.debug=all    ...
```

Even more easy (Java) app. level programming …(hands-on: Lab 8)

Transparent support for base URL operations (URL/HTTP or URL/HTTPS): URL Class and URL Connections

Analysis with:

- openssl tool: TLS Session establishment inspection and observation of established ciphersuites
- wireshark: TLS protocol analysis

JSSE Programming Client/Server w/ detailed parameterization of TLS endpoints

JSSE-Based Rest Code

Java JSSE Programming (Lab, hands-on)

- See Lab 8 (Hands-On Exercises)
  - Debugging / TLS Traffic Analysis
    - Use of openssl, wireshark and browser/browser-dev. tools
  - Programming with JSSE (Demos/Exercises)
    - Fine-tuned TLS parameterizations and TLS session context control
    - Unilateral vs. Mutual authentication
    - TLS debug in java with -Djavax.net.debug=all

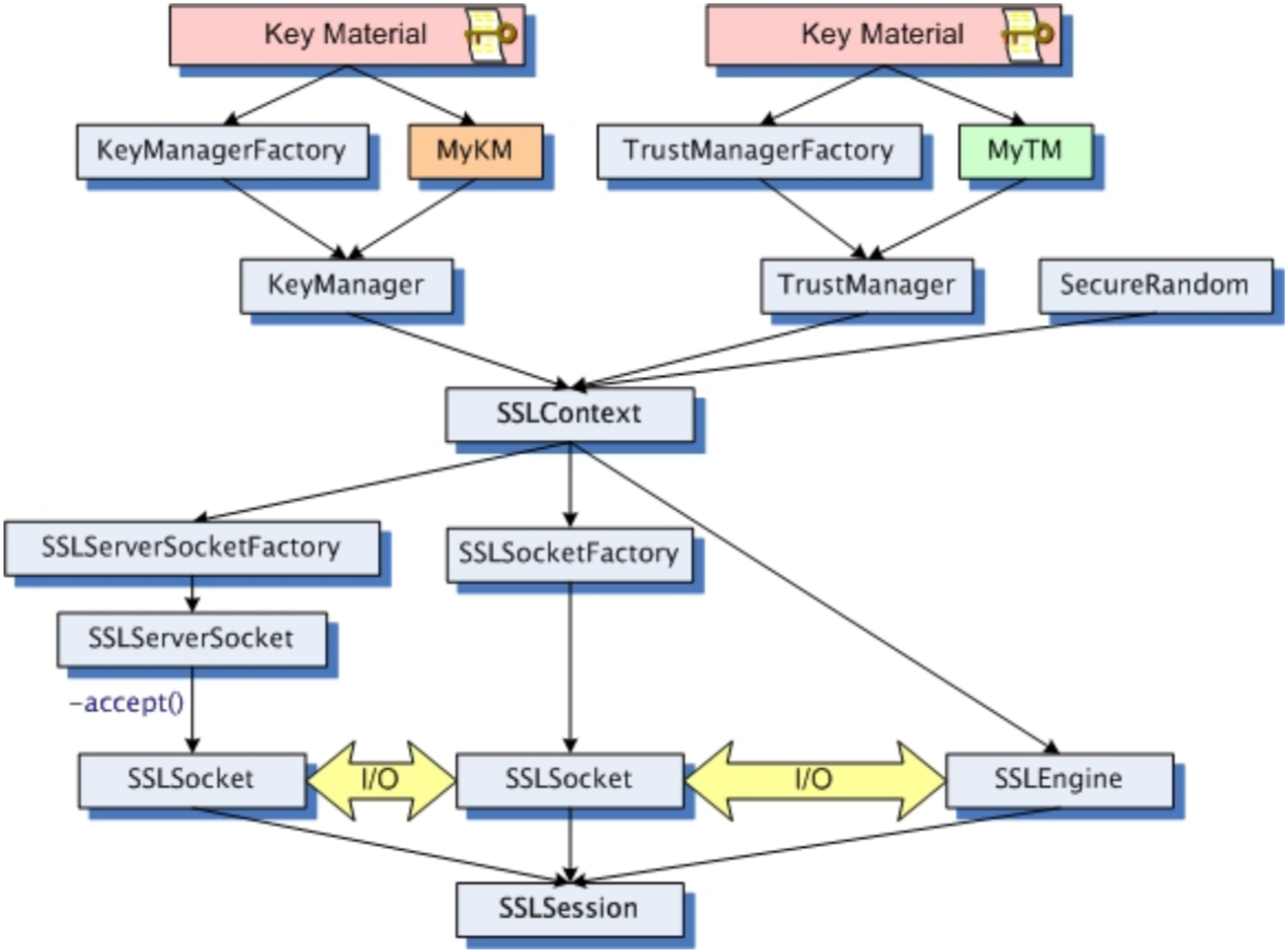
JSSE Programming; Base Server Skeleton

```
import java.io.*;
import javax.net.ssl.*;
. . .
int port = availablePortNumber;
SSLServerSocket s;
try {
SSLServerSocketFactory sslSrvFact =
(SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
s = (SSLServerSocket)sslSrvFact.createServerSocket(port);
SSLSocket c = (SSLSocket)s.accept();
OutputStream out = c.getOutputStream();
InputStream in = c.getInputStream();
// Send and Recv messages
} catch (IOException e) {     }
```

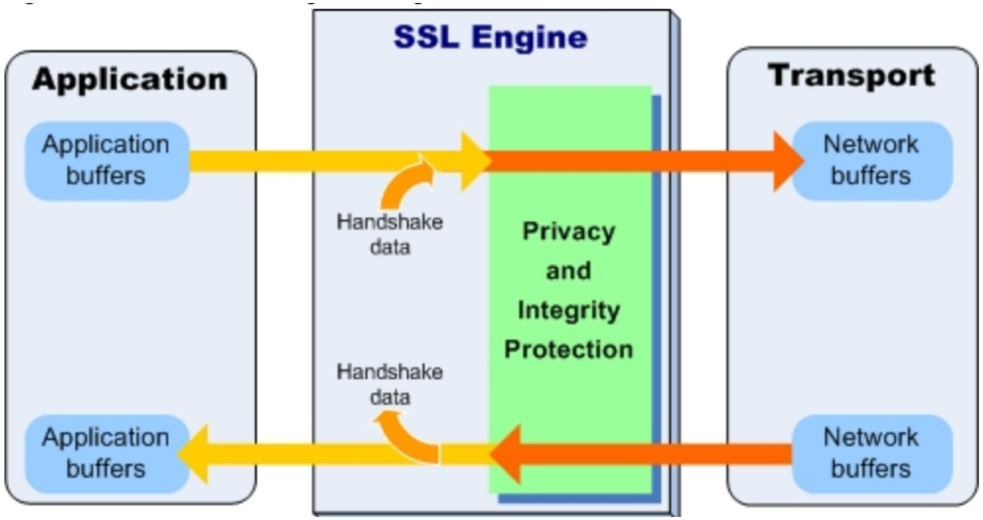
JSSE Programming: Base Client Skeleton

```
import java.io.*;
import javax.net.ssl.*;
. . .
int port = availablePortNumber;
String host = "hostname";
try {
  SSLSocketFactory sslFact =
     (SSLSocketFactory)SSLSocketFactory.getDefault();
  SSLSocket s = (SSLSocket)sslFact.createSocket(host, port);
  OutputStream out = s.getOutputStream();
  InputStream in = s.getInputStream();
  // Send / Recv messages from the server
}  catch (IOException e) {     }
```

JSSE Classes and Interfaces



Dataflows protected by JSSE TLS Engine



App-Level Session-Level Transport-Level

Engine (runtime) states (TLS session-level management):

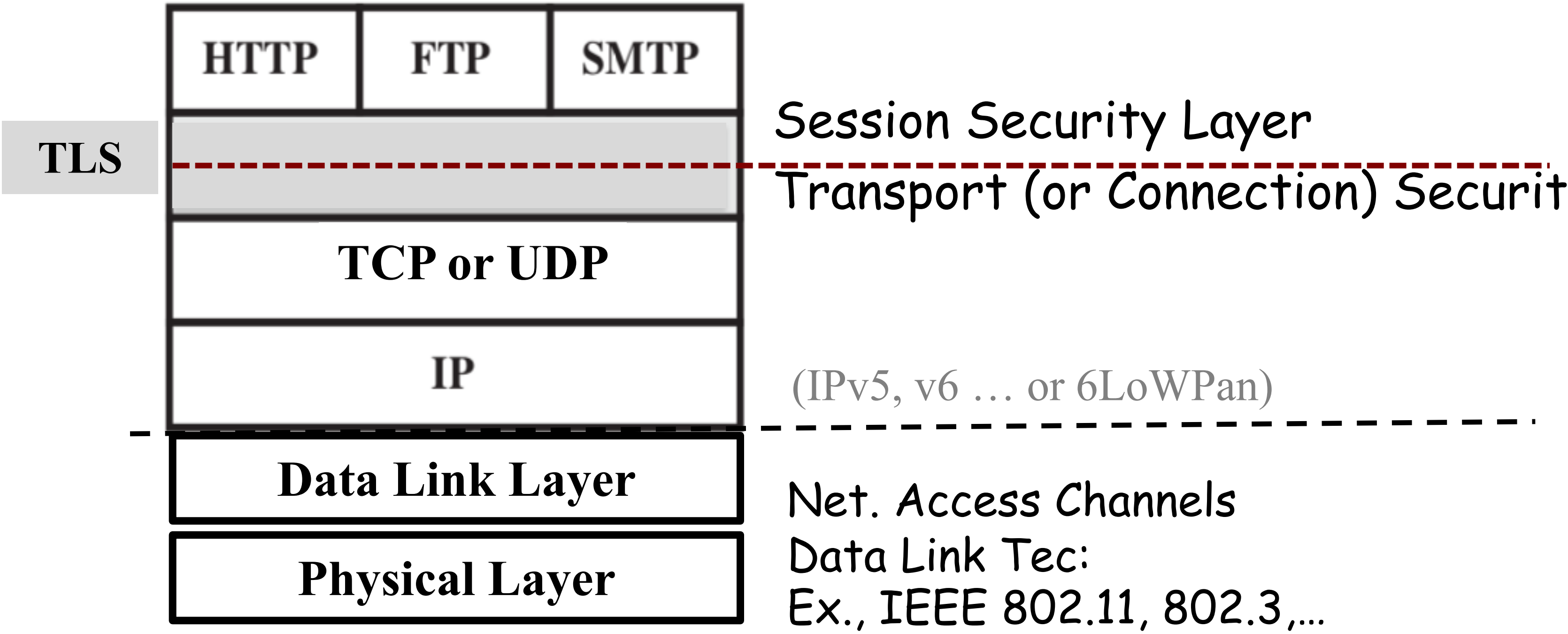
- **Creation:** Ready to be configured
- **Initial handshaking:** Perform authentication and negotiate communication parameters
- **Application data:** Ready for application exchange
- **Re-handshaking:** Renegotiate communications parameters/authentication; handshaking data may be mixed with application data
- **Closure:** Ready to shut down the connection

Outline

- WEB security issues
    - Web traffic security threats: the role of SSL and TLS
    - TCP/IP Stack and TLS
    - Security properties and services addressed by TLS
    - TLS Stack (TLS Sub-Protocols)
    - Overview of TLS Handshake
    - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
    - TLS architecture and protocol stack
    - TLS protocol versions
    - TLS configurability and flexibility issues
    - TLS Ciphersuites
    - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

TLS: Secure Session vs. Secure Transport

Transport-Level Security Service Levels



Session Security Layer

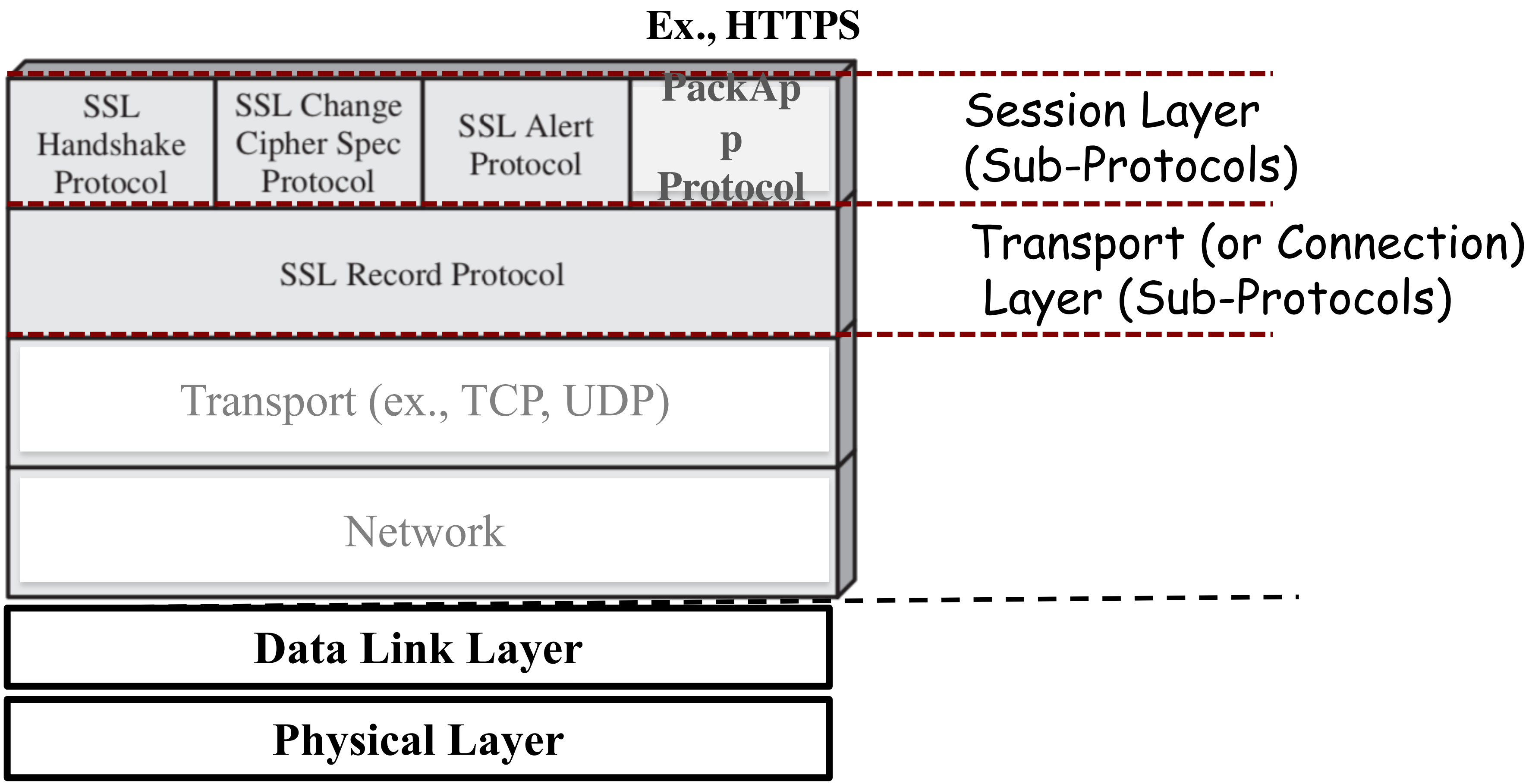
Transport (or Connection) Security Layer

(IPv5, v6 … or 6LoWPan)

Net. Access Channels
Data Link Tec:
Ex., IEEE 802.11, 802.3,…
802.1 to 802.15.x
https://en.wikipedia.org/wiki/IEEE_802

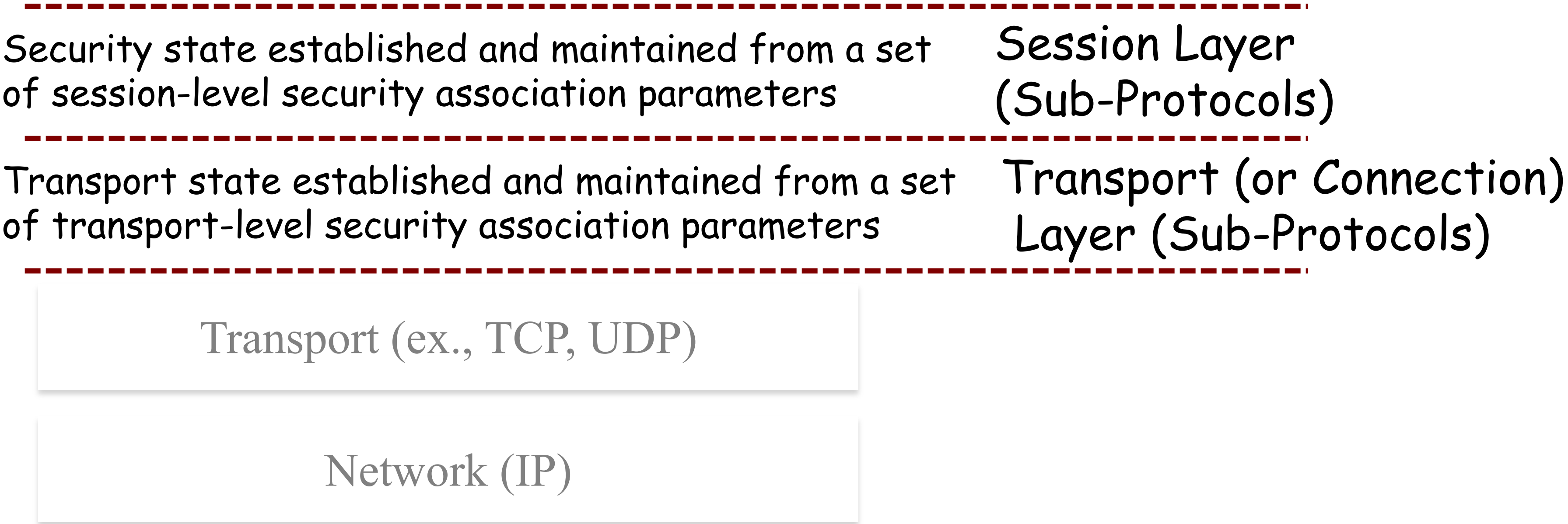
TLS: Secure Session vs. Secure Transport

Transport-Level Security Service Levels and related protocols in the TLS Stack


Ex., HTTPS
SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
PackAp p Protocol
SSL Record Protocol
Transport (ex., TCP, UDP)
Network
Data Link Layer
Physical Layer
Session Layer (Sub-Protocols)
Transport (or Connection) Layer (Sub-Protocols)

TLS: Secure Session vs. Secure Transport

TLS Security Association Parameters: Established and Setup from the Handshake Protocol



Security state established and maintained from a set of session-level security association parameters

Session Layer (Sub-Protocols)

Transport state established and maintained from a set of transport-level security association parameters

Transport (or Connection) Layer (Sub-Protocols)

Transport (ex., TCP, UDP)

Network (IP)

…

TLS: Transport Security Control Parameters

A transport or connection state is defined by a set of parameters, (transport or connection security association parameters) exchanged and initially established in the context of the Handshake protocol

- **Server and client random values.**
- **Server write MAC secrets** (Server MAC Key)
- **Client write MAC secret** (Client Mac Key)
- **Server write key** (Server Encryption Key)
- **Client write key** (Client Encryption Key)
- **Initialization vectors**: established from an initial IV
- **Sequence numbers**: From 0 to 2^{64} -1

TLS: Session Security Control Parameters

A session state is defined by a set of security association parameters, exchanged and initially established in the context of the Handshake protocol

Session identifier: An arbitrary byte sequence proposed bi the client but chosen by the server to identify an active or resumable session state.

Peer certificate: An X509.v3 certificate of the peer. This element of the state may be null, depending on different authentication modes

In general: a certification chain, validated during the handshake

Compression method: algorithm to compress data prior to encryption.

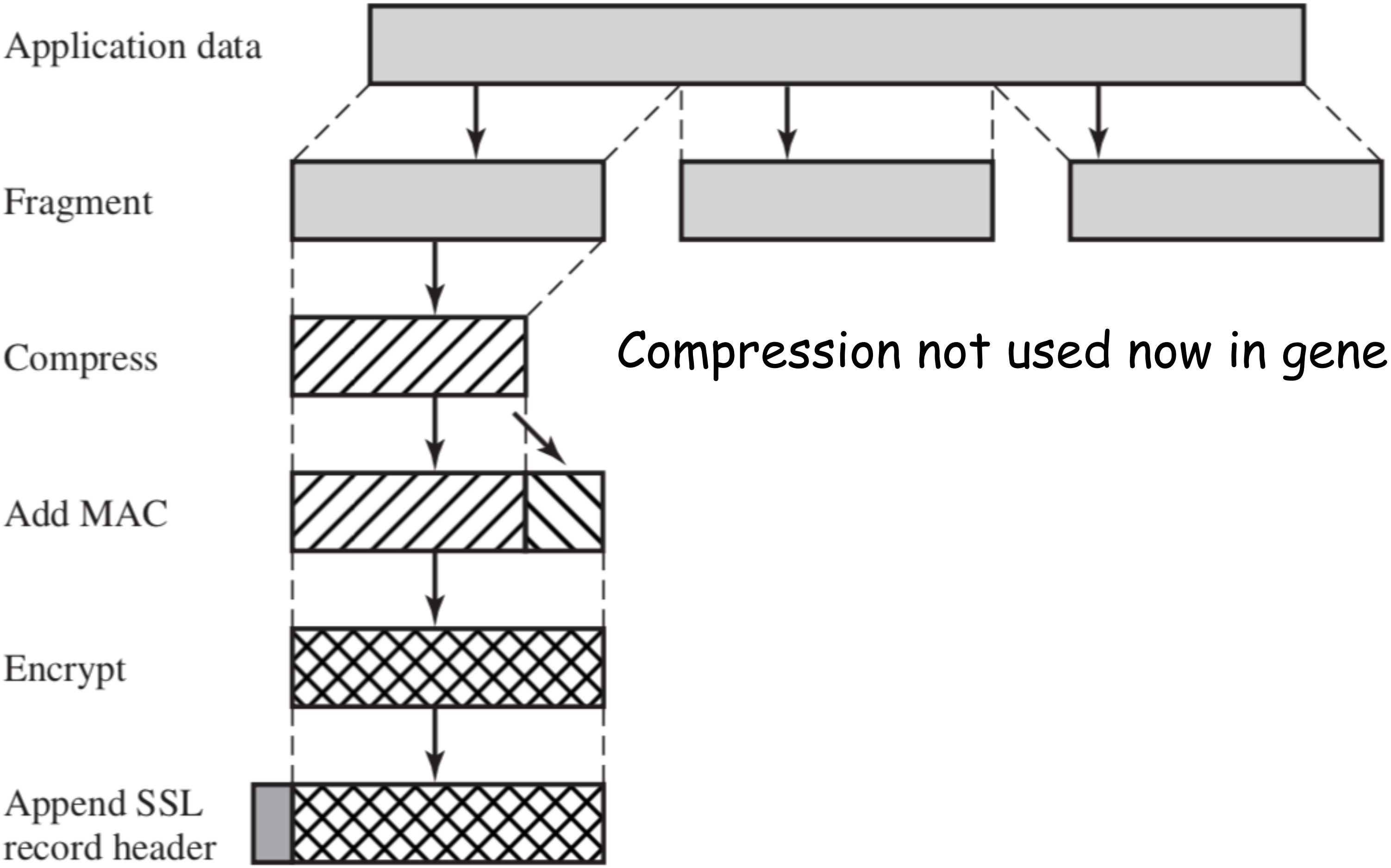
Cipher spec: Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.

Master secret: 48-byte secret shared between the client and server.

Is_resumable: A flag indicating whether the session can be used to initiate new connections

TLS: TLP - Record Layer Protocol

RLP Processing in Endpoints



Compression not used now in general

TLS Study consolidation

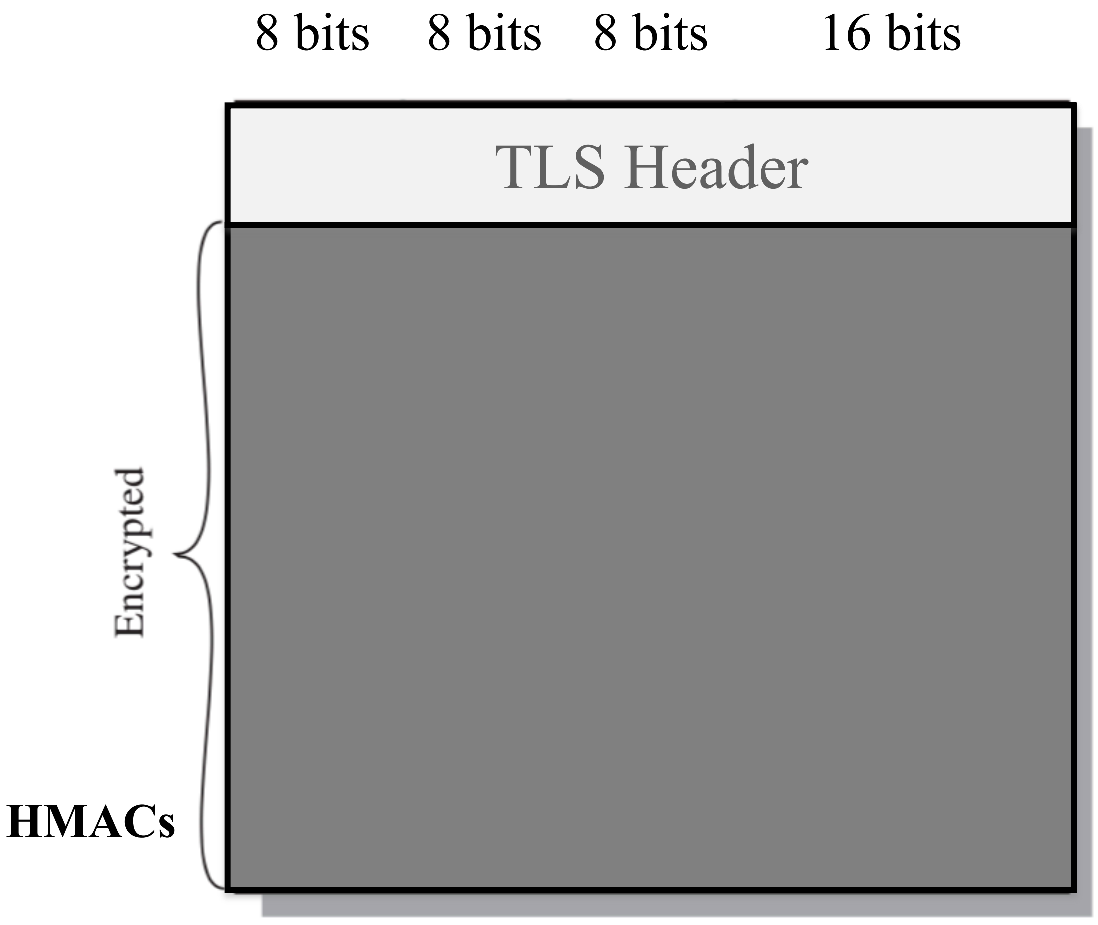
- Consolidate your TLS study:
  - TLS architecture
  - Connection-level: TLS RLP and RLP operation
  - Session-level (and subprotocols): Handshake, ChangeCipherSpec and Alert
  - Handshake modes: Key-exchanged methods, Handshake Setup and operation
  - TLS (and SSL): summary of possible attack vectores

See:
W. Stallings, Network Security Essentials, Chap. 6, 6.2 – Transport Layer Security

Outline

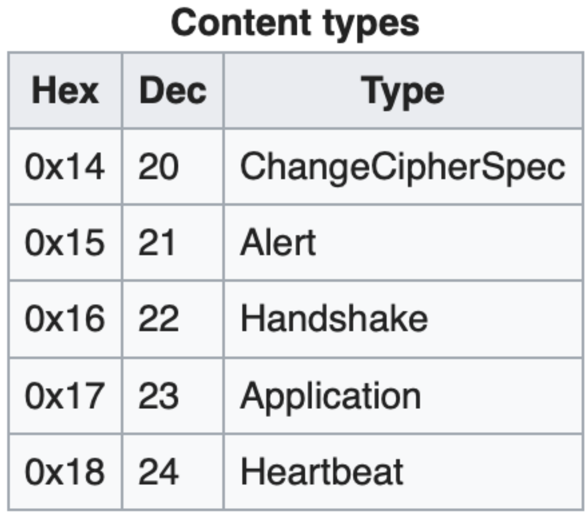
- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

RLP Message Format



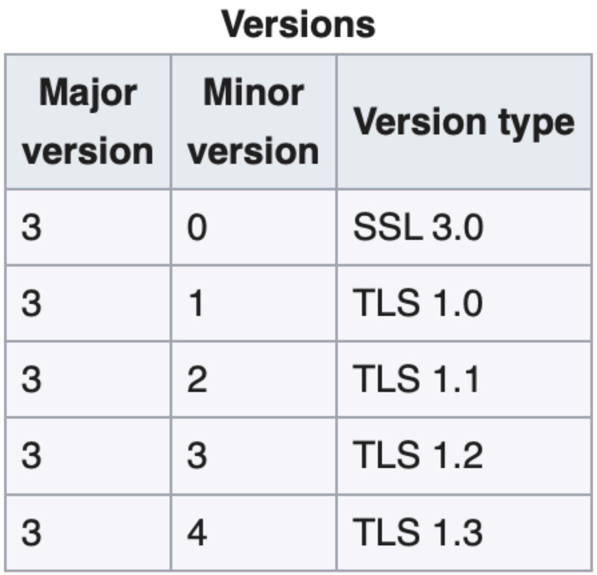
HMAC-MD5
HMAC-SHA-1

Also: HMAC-SHA256
HMAC-SHA384
and AEAD



Content types

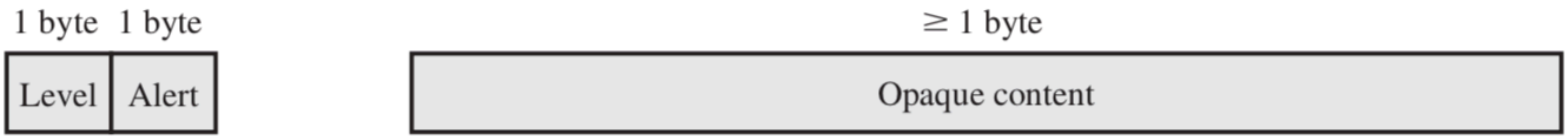
Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application
0x18	24	Heartbeat



Versions

Major version	Minor version	Version type
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2
3	4	TLS 1.3

TLS AP: Alert Protocol


1 byte 1 byte

Level Alert

(b) Alert Protocol

≥ 1 byte

Opaque content

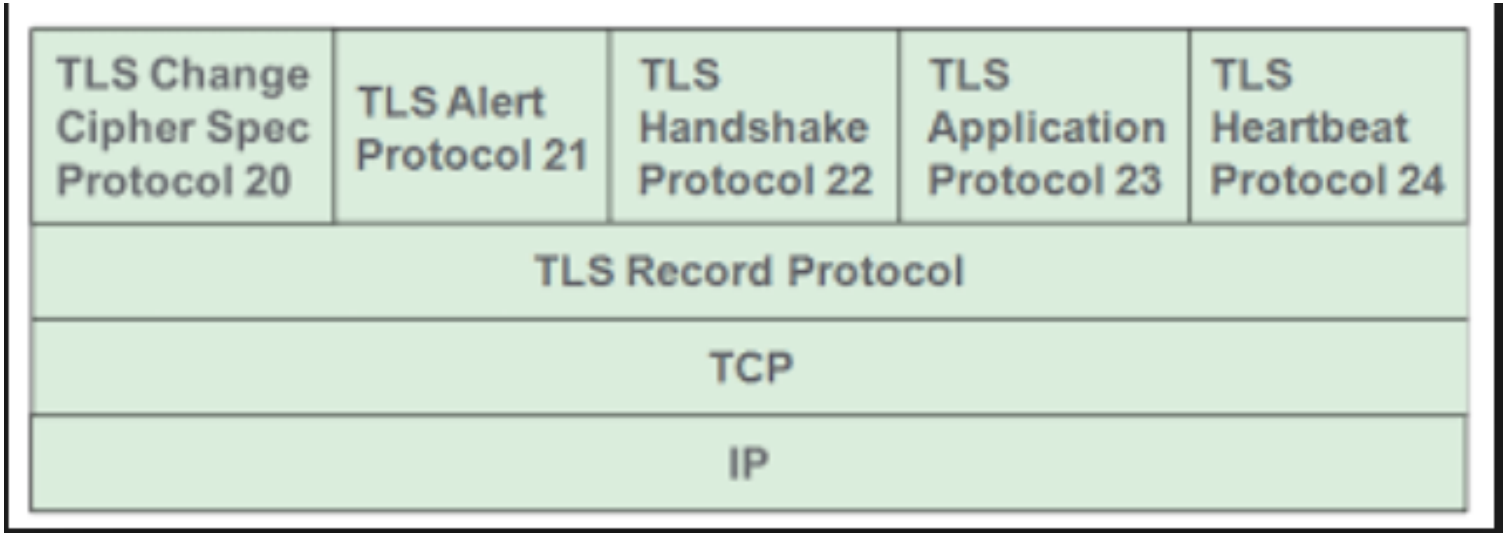
(d) Other Upper-Layer Protocol (e.g., HTTP)

Standardized Alert Control Messages and Encodings (see bibliography) are categorized in different levels: warning or fatal

Fatal alerts: close the session and remove all the security association parameters.

TLS - HB (Heartbeat Protocol Extension)

Introduced in 2012, RFC 6520 (as a keep-alive control to maintain the connection state)



TLS Change Cipher Spec Protocol 20

TLS Alert Protocol 21

TLS Handshake Protocol 22

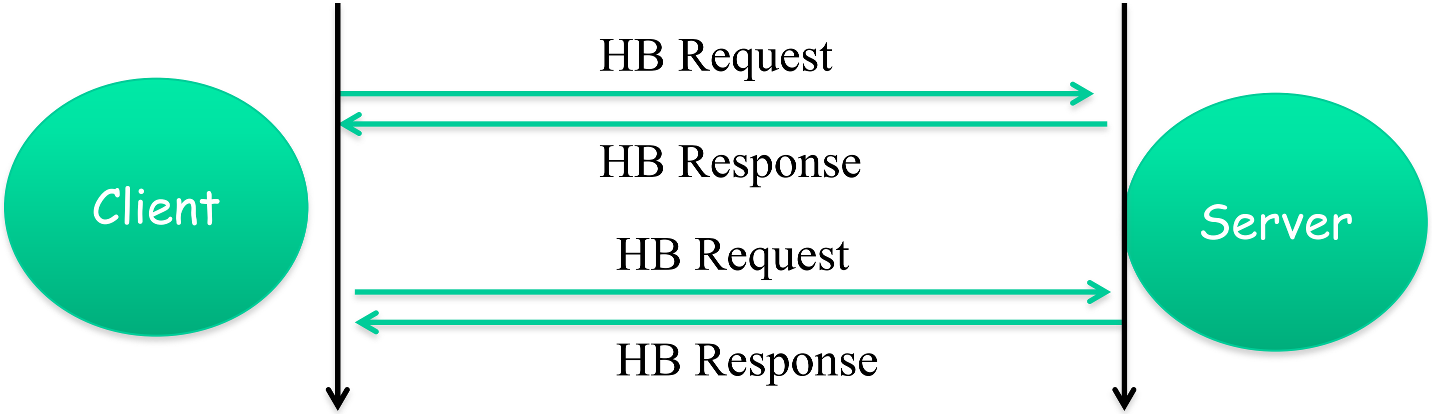
TLS Application Protocol 23

TLS Heartbeat Protocol 24

TLS Record Protocol

TCP

IP



Client

Server

HB Request

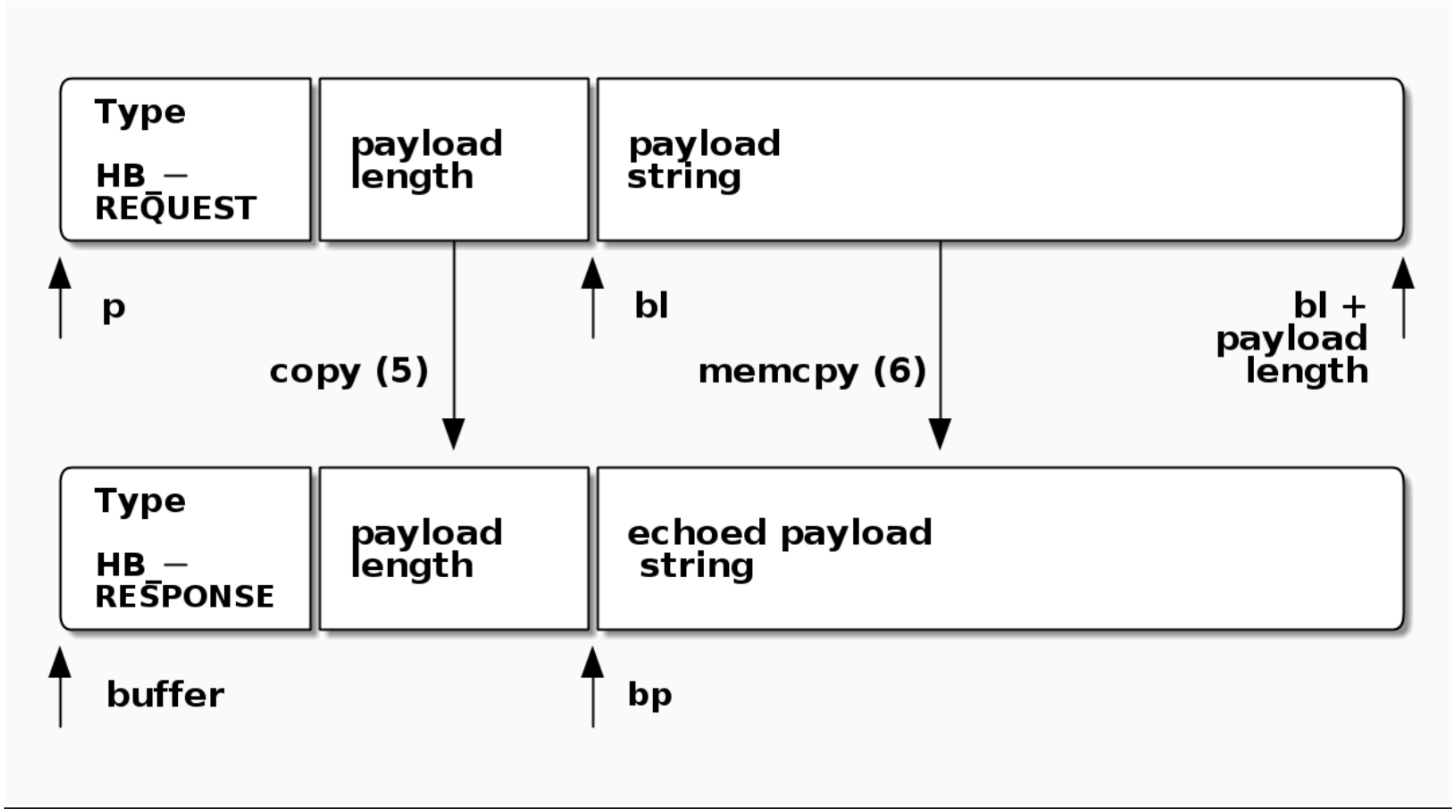
HB Response

HB Request

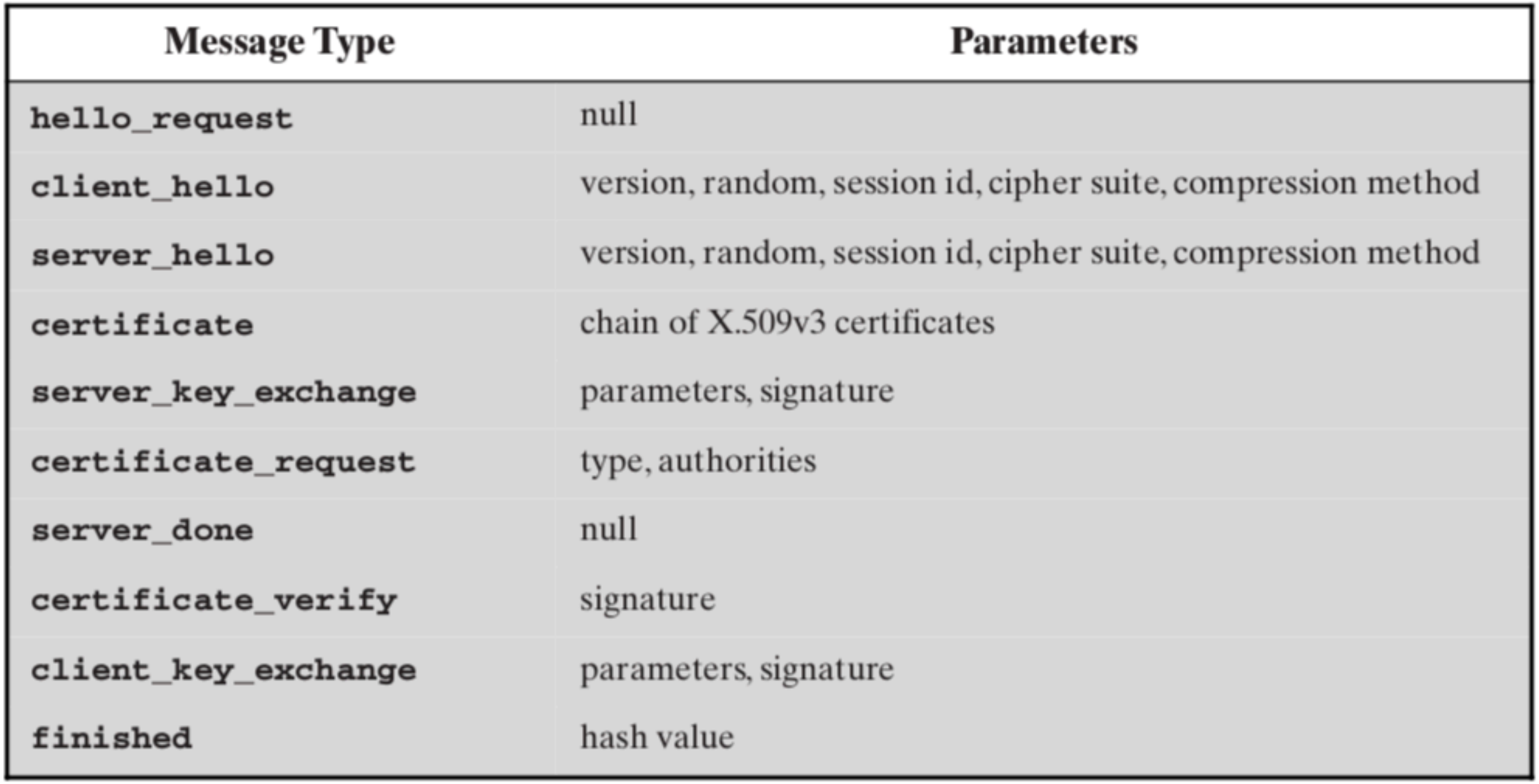
HB Response

TLS - HB (Heartbeat Protocol Extension)

Introduced in 2012, RFC 6520 (as a keep-alive control to maintain the connection state)



TLS Handshake - Handshake 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

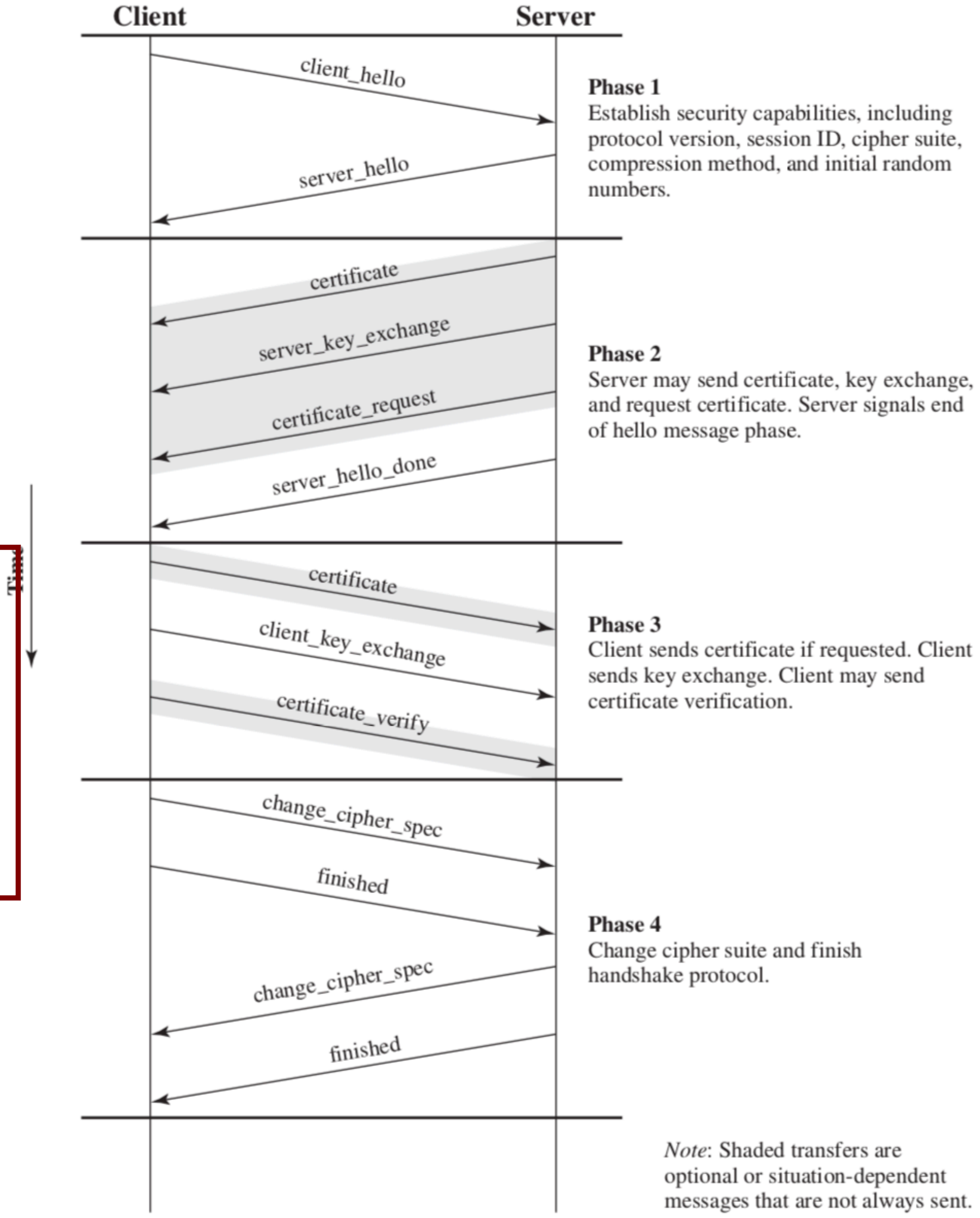
TLS Handshake Phases

- Four Phases:
  - Phase 1:
    - Establishment of Security Capabilities: Negotiation and Parameterization Phase
  - Phase 2:
    - Server Authentication and Key-Exchange (establishment of security parameters authenticated from the server side)
  - Phase 3:
    - Client Authentication and Key-Exchange (establishment of security parameters authenticated from the server side)
  - Phase 4: Finish Phase
    - Phase for establishment and setup of all the security association parameters
    - Includes the CCSP message exchanges

TLS Handshake:

Handshake Flow

The Better for Your detailed study: Use wireshark (or ssldump) and inspect TLS traffic to learn !


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Outline

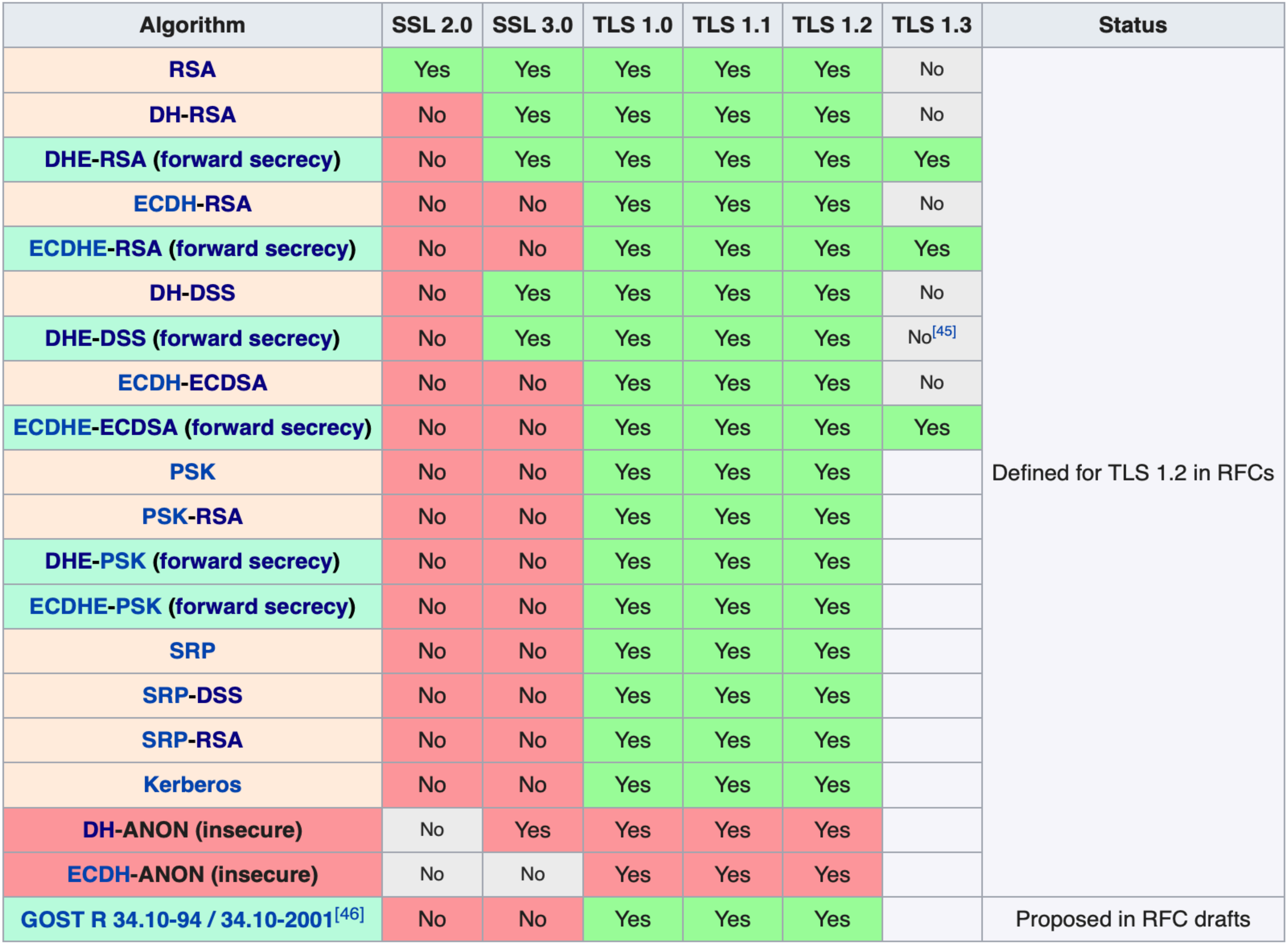
- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

TLS Key Exchanges in the Handshake

- Main Key-Exchange Methods in the Handshake
  - RSA Based (TLS_RSA)
  - FDH or Fixed Diffie-Hellman (TLS_DH, TLS_ECDH)
  - EDH or Ephemeral Diffie-Hellman (TLS_DHE, TÇS_ECDHE)
  - ADH or Anonymous Diffie-Hellman (TLS_DH_ANON, TLS_DHE_ANON)

- Flexibility and Authentication Modes for Key-Exchanges:
  - Server Only (Unilateral Server Authentication)
  - Client Only (Unilateral Client Authentication)
  - Mutual Authentication (Client and Server)
  - No Authentication (Anonymous)

Key exchange/agreement and authentication



Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
RSA	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
DH-RSA	No	Yes	Yes	Yes	Yes	No	
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes	
ECDH-RSA	No	No	Yes	Yes	Yes	No	
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
DH-DSS	No	Yes	Yes	Yes	Yes	No	
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No[45]	
ECDH-ECDSA	No	No	Yes	Yes	Yes	No	
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
PSK	No	No	Yes	Yes	Yes		
PSK-RSA	No	No	Yes	Yes	Yes		
DHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes		
ECDHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes		
SRP	No	No	Yes	Yes	Yes		
SRP-DSS	No	No	Yes	Yes	Yes		
SRP-RSA	No	No	Yes	Yes	Yes		
Kerberos	No	No	Yes	Yes	Yes		
DH-ANON (insecure)	No	Yes	Yes	Yes	Yes		
ECDH-ANON (insecure)	No	No	Yes	Yes	Yes		
GOST R 34.10-94 / 34.10-2001[46]	No	No	Yes	Yes	Yes		Proposed in RFC drafts

TLS Ciphersuites

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

TLS Ciphersuites

- See:
- https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml

- LAB 8:
  - See more and how to manage (set, get, enable disable) configurations for TLS protocol versions, authentication modes and setting/negotiation ciphersuites between TLS Client/Server endpoints
  - Java programming with JSSE (SSL Sockets)

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - Overview of TLS Handshake
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

TLS Ciphersuites

See in LAB 8:

- Use of Wireshark for TLS Traffic Analysis

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
  - Security properties and services addressed by TLS
  - TLS Stack (TLS Sub-Protocols)
  - TLS operation and TLS based programming
- TLS: Session-Security vs. Transport Security Layers
  - TLS architecture and protocol stack
  - TLS protocol versions
  - TLS configurability and flexibility issues
  - TLS Ciphersuites
  - Analysis of TLS Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS

HTTPS Connection Initiation

Connection Initiation:

- HTTPS Client maps on TLS Client endpoint
- TLS starts with the handshake
  - Implicitly after a TCP connection is established
  - When the TLS handshake has finished, the client may then initiate the first HTTP request.
  - All HTTP data is to be sent as TLS application data. Normal HTTP behavior, including retained connections, should be followed.

HTTPS Connection Closure

Connection Closure:

- An HTTP client or server can indicate the closing of a connection by including the following line in an HTTP record:

  Connection: close.

- This indicates that the connection will be closed after this record is delivered, terminating the TLS "Session" Control State

- The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve also closing the underlying TCP connection.
  - Double handshake FIN/ACK FIN in TCP connnection Closures

- Client sends a TLS alert protocol (**close_notify alert)**. Then, TLS implementations must initiate an exchange of closure alerts before closing a connection.

HTTPS Connection Closure w/ Incomplete Closes

- A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an "incomplete close".
  - Note that an implementation that does this may choose to reuse the session.
  - This should only be done if the application knows (typically through detecting HTTP message boundaries) that it has received all the message data that it cares about.

For more information (hands-on):

See HTTPS debug with wireshark and browser/https (web) server interaction

HTTPS Connection Closure without close_notify

HTTP clients must cope with a situation in which the underlying TCP connection is terminated without a prior close_notify alert and without a Connection: close indicator.

- Such a situation could be due to a programming error on the server or a communication error that causes the TCP connection to drop.

The unannounced TCP closure could be also evidence of some sort of attack.

- So the HTTPS client should issue some sort of security warning(typically awareness control and logging such situations) when this occurs.

See:
W. Stallings, Network Security Essentials, Chap. 6, 6.3 – HTTPS

Slides Revision and Suggested Readings and Study

Readings
(for frequency test):

W. Stallings, Network Security Essentials – Applications and Standards
-Ed.. 2017 Chap 6 Transport Layer Security, 6.1-6.4, pp. 187-208

Practical Study:
TLS and HTTPS Traffic Analysis with different tools (see the slides and "hands-on" traffic analysis in Labs)

- Particularly: Handshake, RLP exchanges and TLS flow depending on the Handshake negotiation and parameterizations
- See also the "fine-grain" parameterization when programing with TLS (ex., Java JSSE Lab Exercises)

Revision: Complementary Readings

See the other references on the slides and bibliog. references in the textbook

And revise also the available materials
Lab 7 – X509 Certificates
Certification Chains and Tools
Lab 8 – TLS Analysis, tools and programming support