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

Mestrado Integrado em Engenharia Informática
MSc Course: Informatics Engineering
2º Semestre, 2018/2019

7. Transport Layer Security (TLS) and HTTPS

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

Web Security, HTTPS and TLS

- Web Browsers (and WebApps), Web Servers and Web-Based Contents and Services: easy to use, easy to configure easy to develop/deploy

- But … underlying software (and runtime) is complex and nay hide many potential security flaws: lots of evidences in the real life

- HTTPS is (and will be more and more) the unified application-level security layer to protect web (http) traffic

- And more and more critical applications managing sensitive data and traffic are Web based (primarily supported by HTTPS and TLS)

Web Security vs. Web Traffic Threats

- Initial motivation: Protection of HTTP Communication (C/S interaction),
- … but with a generic solution to support any other TCP supported application protocol

Implementations:
SSL Socket-Library (SSLSockets), openssl
Java: JSSE Package
https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html
MS C#: SSLStreams
https://docs.microsoft.com/en-us/dotnet/api/system.net.security.sslstream?redirectedfrom=MSDN&view=netframework-4.8

- Usually implementations offer fast development and prototyping to migrate TCP/IP Based Applications and Protocols to adopt TLS

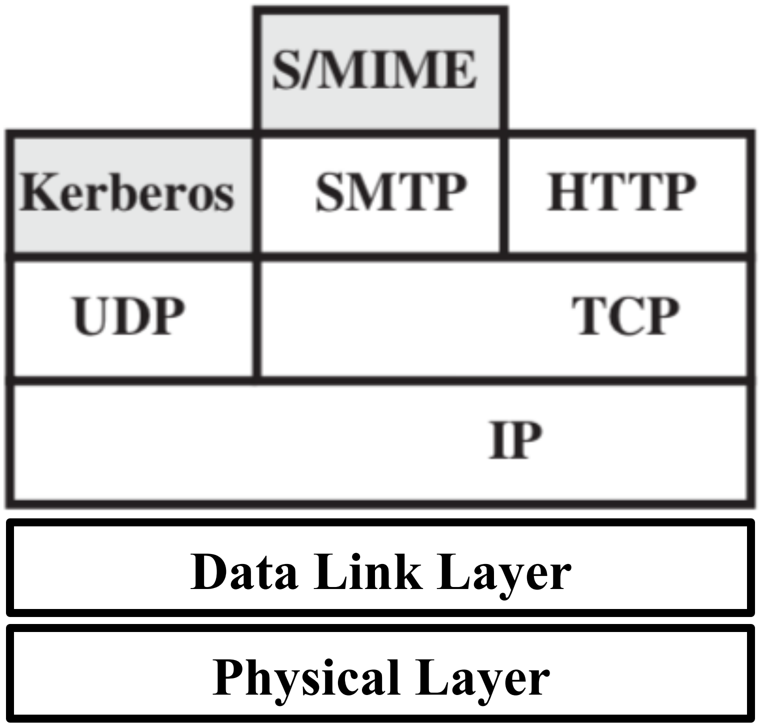
TLS: Protection provided

- Security Properties Addressed):
  - Integrity (message and data flow-integrity)
  - Confidentiality (message and data confidentiality)
  - Authentication (peer authentication + message authentication)
  - 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

TCP/IP Security Stack

YCP/IP Security Stack:
Application-Level Security Services and Standards



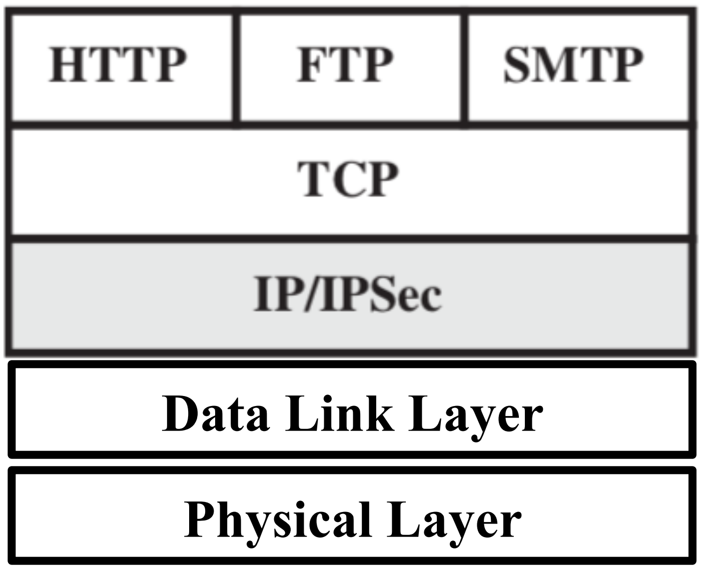
Email Security

S/MIME

PGP is another solution

TCP/IP Security Services Stack

Network-Level Security

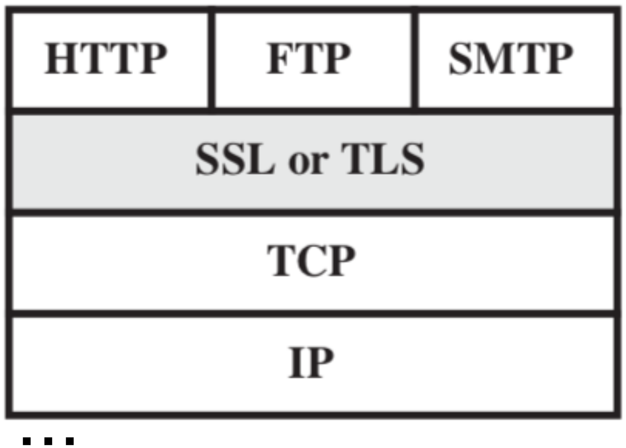


Ex.,
IPSec Protocol Stack

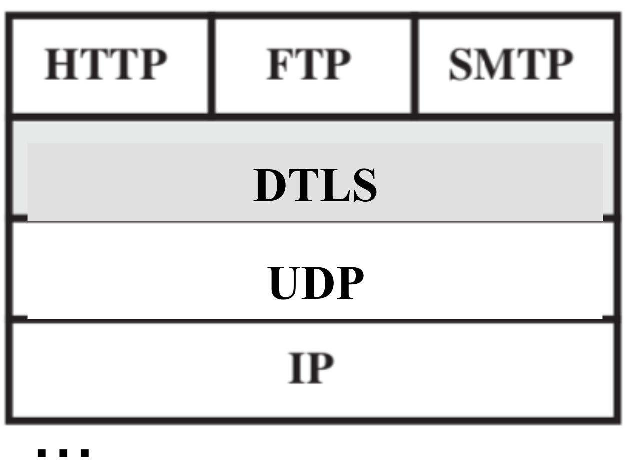
Relevance in supporting
VPNs and Secure VLANs

TCP/IP Security Services Stack

Transport-Level Security



Relevance in support for HTTPS and as a generic Secure transport to protect many other Application-Level Protocols

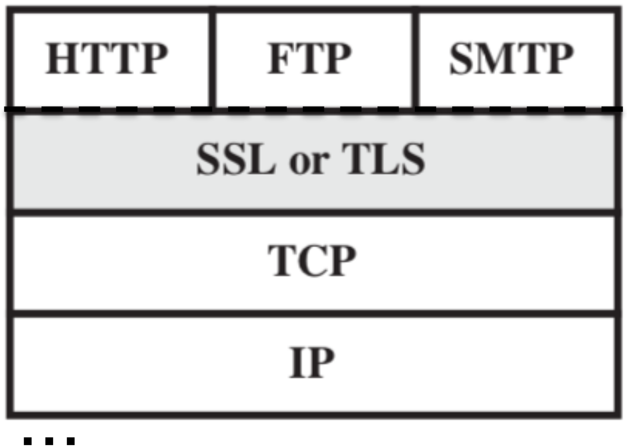


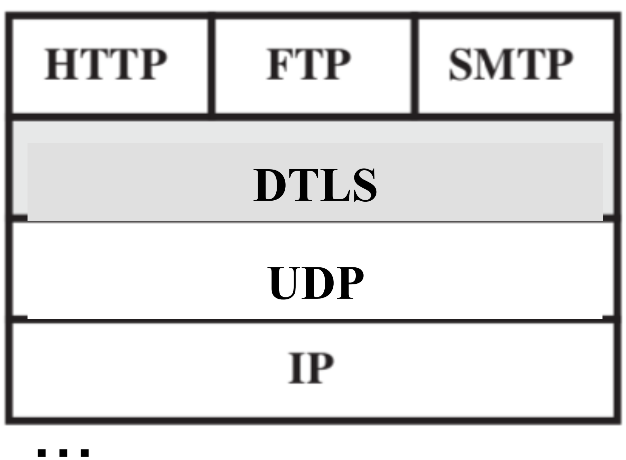
Also defined for UDP Support: DTLS

Implementations: as a generic protocol suite (in library) or embedded packaging (ex., browsers)

TCP/IP Security Services Stack

TCP/IP Programming w/ Transport-Level Security





TLS (SSL) Sockets, Ex., Java JSSE

https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html

Also: TLS Packaging (lot of libraries) for Application-Level Support
See:
https://en.wikipedia.org/wiki/Transport_Layer_Security

Ex., URL operations,
java.lang.Object / java.net.URL

- Rest or WS Runtime Libraries / TLS stubs and skeletons

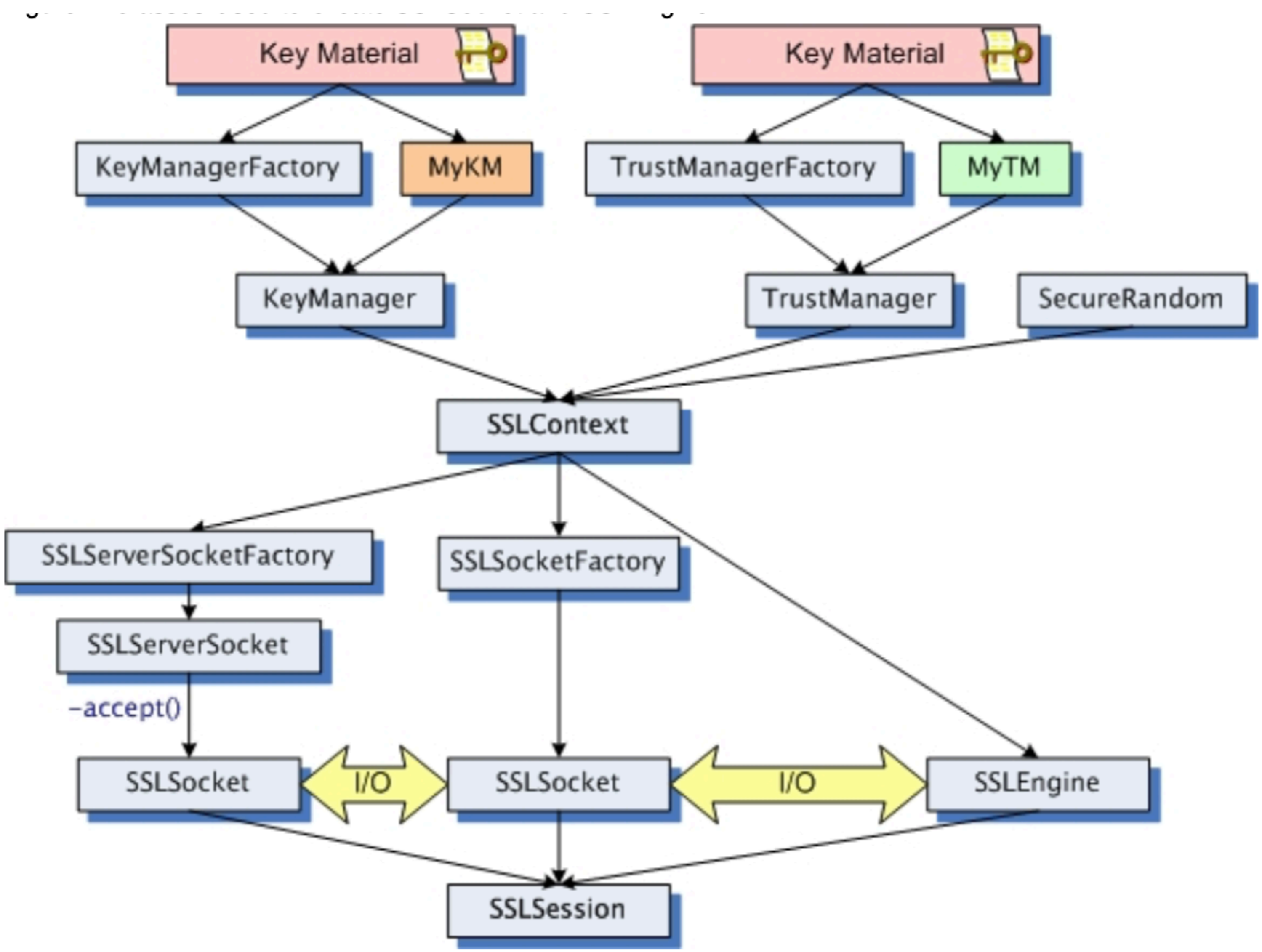
JSSE Programing; Base Server Skeleton

```
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) {     }
```

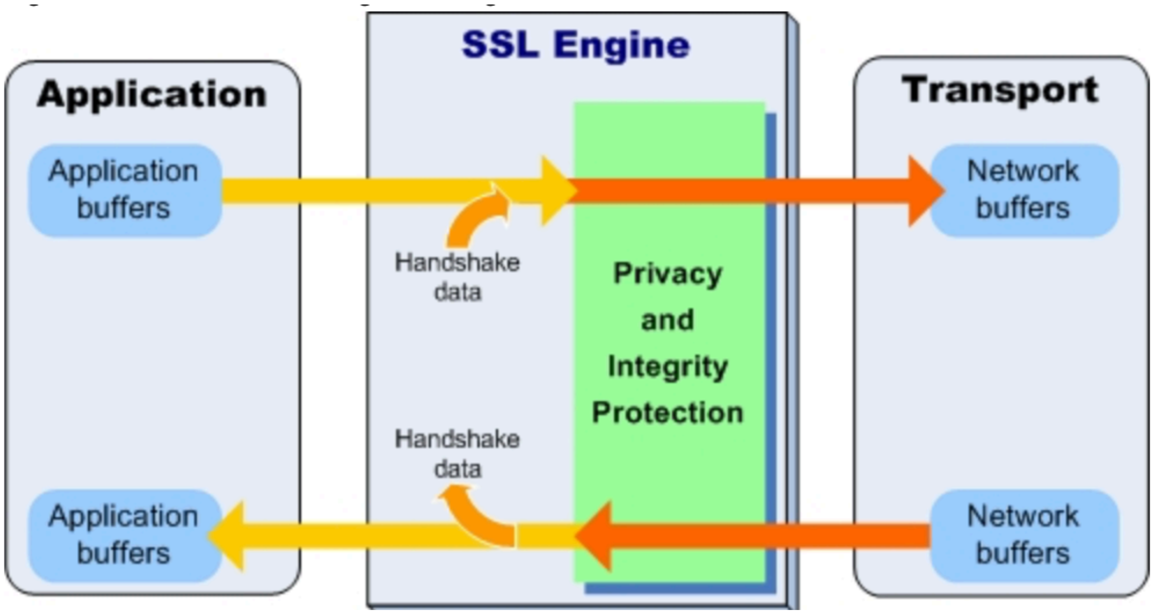
JSSE Programing; Base Client Skeleton

```
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



Engine (runtime) States:

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

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

URL based: Trabsparent support for base URL operations (URL/HTTP or URL/HTTPS) : https://docs.oracle.com/javase/8/docs/api/java/net/URL.html

JSSE Programming Client/Server

JSSE-Based Rest Code

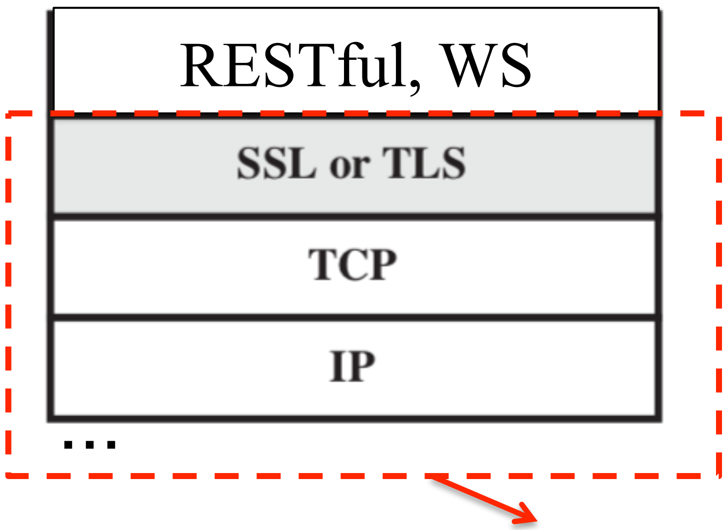
Etc ..

More on Lab8 Materials and Lab8 Class

Also for protection/parameterization of HTTPS endpoints and communications in the TP2 (Work Assignment #2) Requirements

Web-Programing / TLS

Web Programming / HTTPS / Transport-Level Security



Relevance in supporting Secure REST and Web Services endpoints

Usage:
"Transparent Layered Security" for application-level programming ?
… or "Why you should know about the undelaying details ?"

...

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

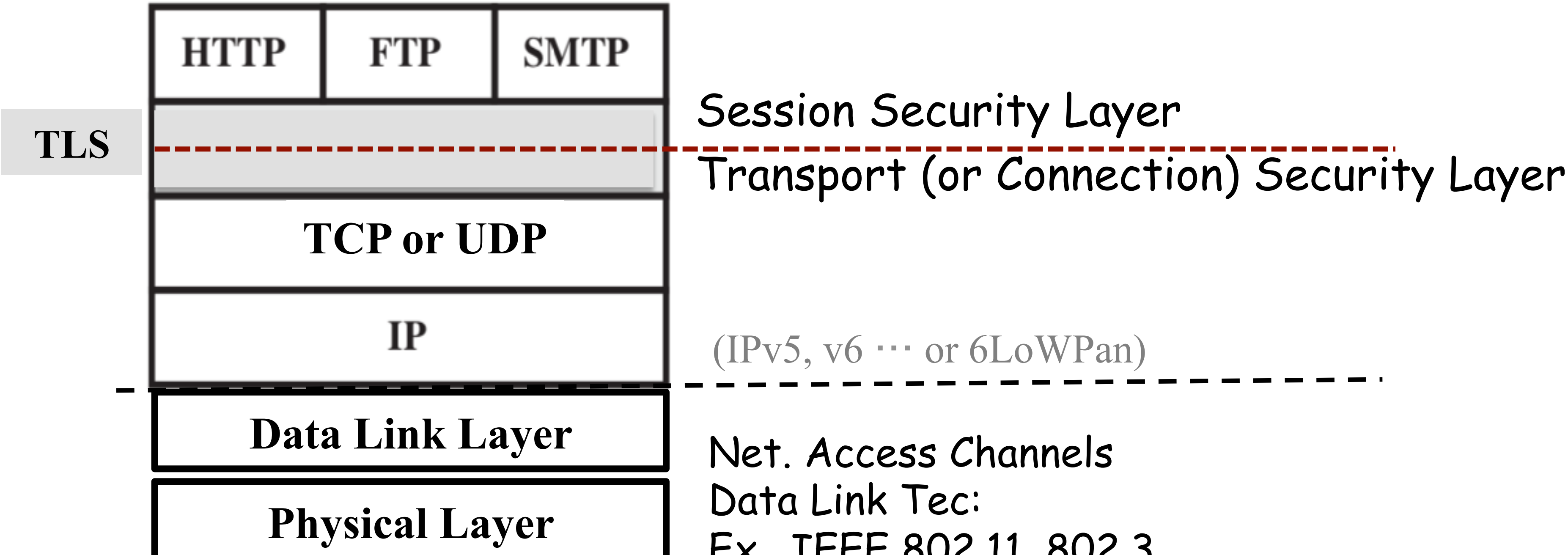
Def. RFC 8446, Aug/18

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

TLS: Secure Session vs. Secure Transport

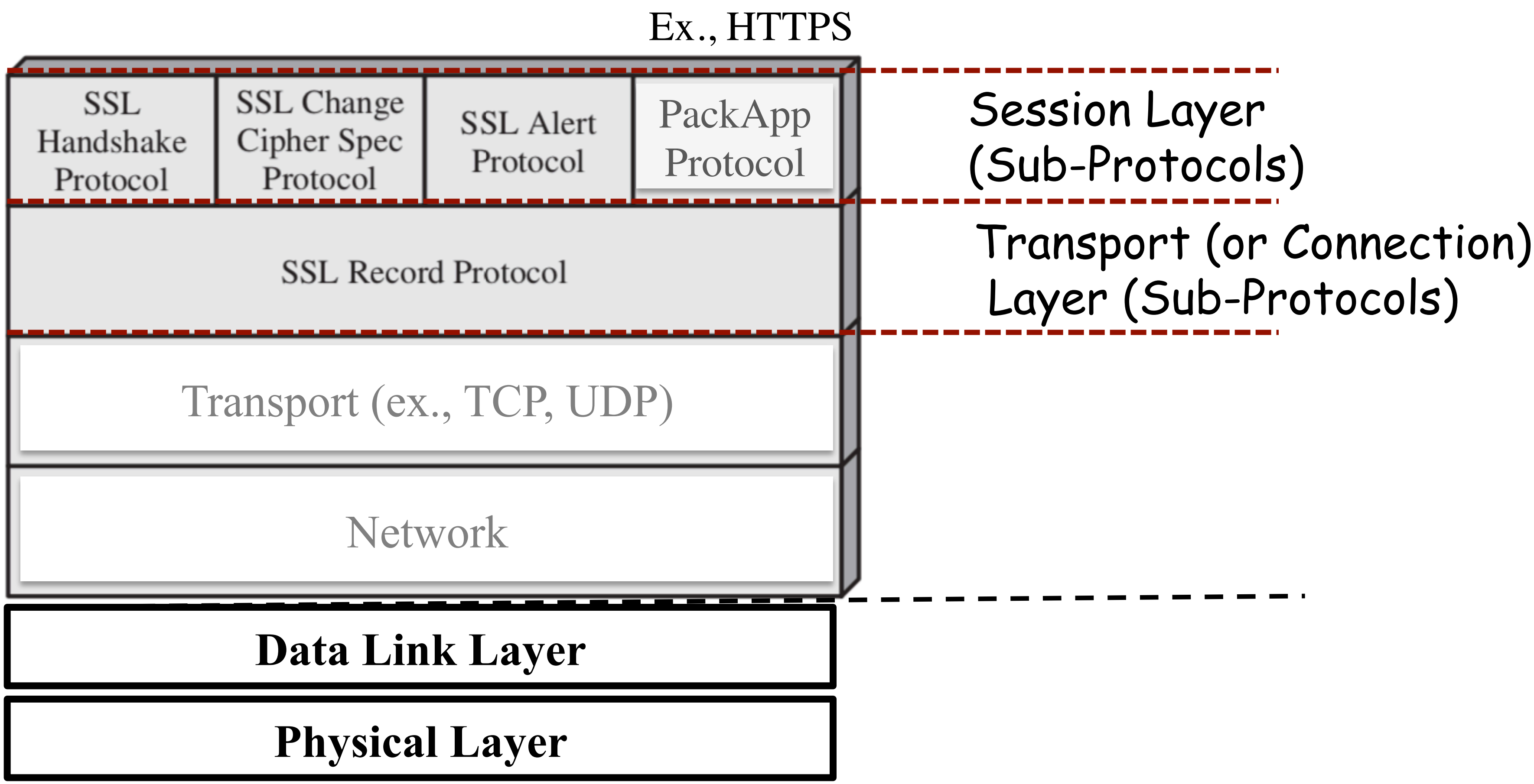
Transport-Level Security Service Levels



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



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.

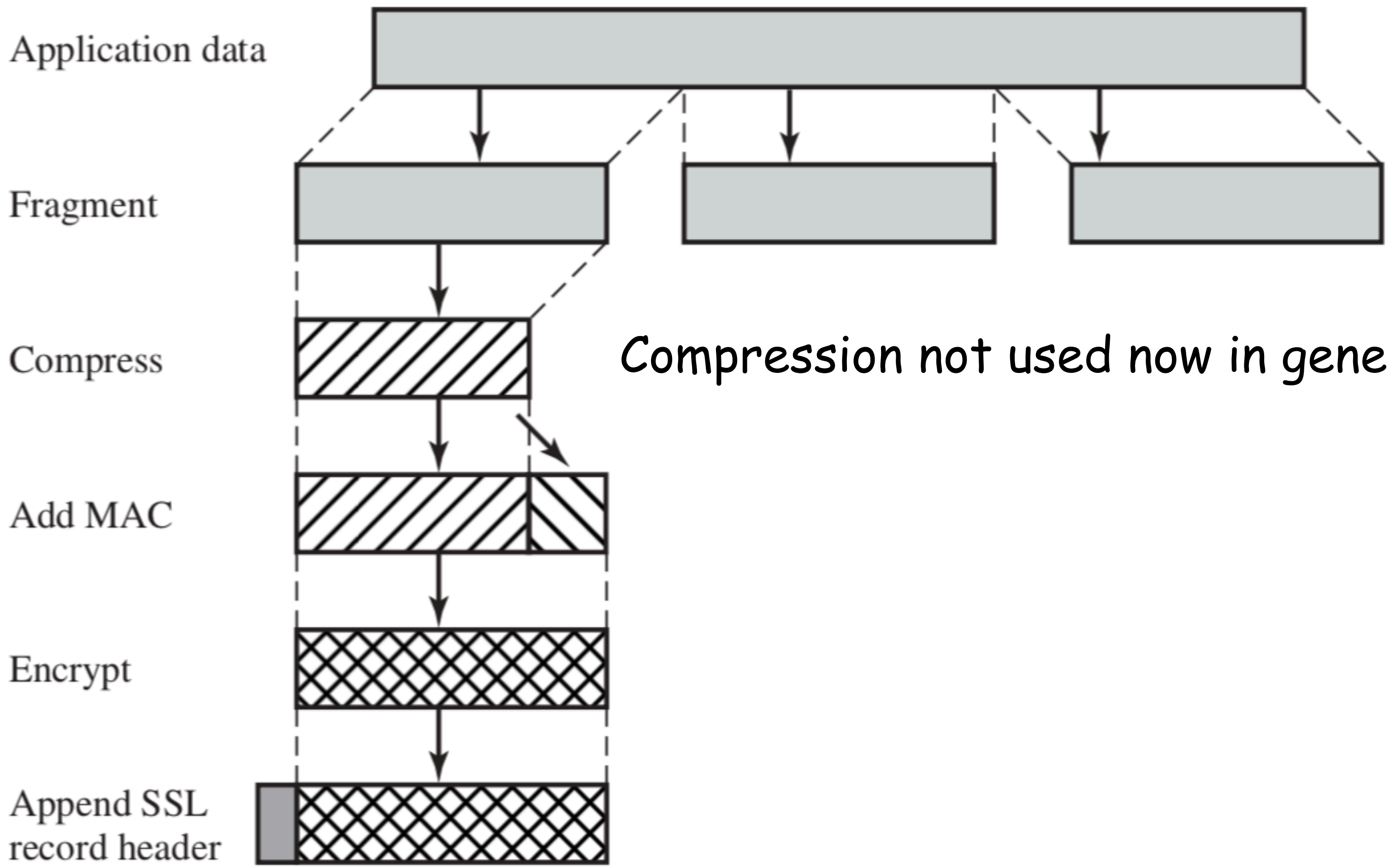
Is_resumable: A flag indicating whether the session can be used to initiate new connections

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

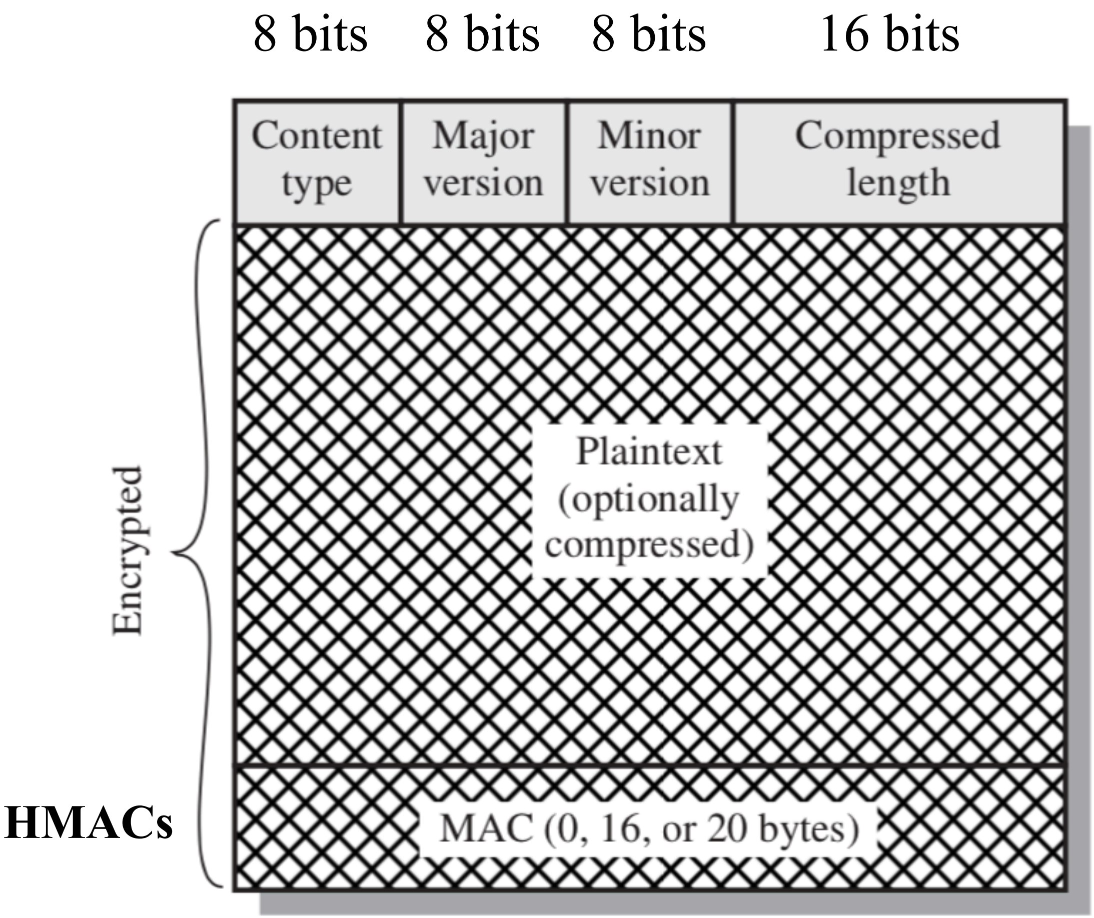
TLS: TLP - Record Layer Protocol

RLP Processing in Endpoints



Compression not used now in general

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

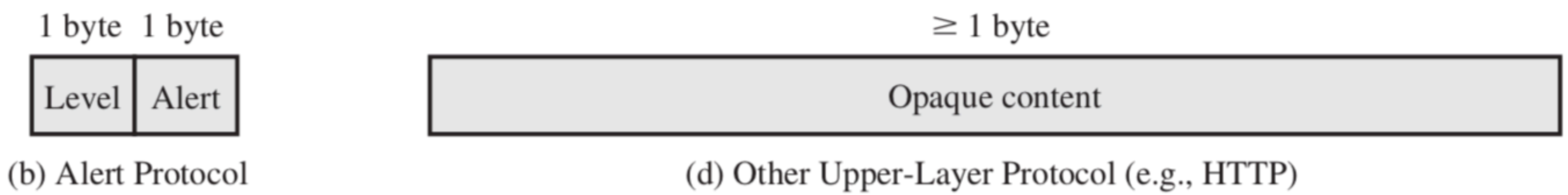
Versions

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

RLP Message Format w/ Additional Padding

<table>
<tr><th>+</th><th>Byte +0</th><th>Byte +1</th><th>Byte +2</th><th>Byte +3</th></tr>
<tr><th>Byte
0</th><td>Content type</td><td colspan="3"></td></tr>
<tr><th rowspan="2">Bytes
1..4</th><td colspan="2">Legacy version</td><td colspan="2">Length</td></tr>
<tr><td>(Major)</td><td>(Minor)</td><td>(bits 15..8)</td><td>(bits 7..0)</td></tr>
<tr><th>Bytes
5..(m–1)</th><td colspan="4">Protocol message(s)</td></tr>
<tr><th>Bytes
m..(p–1)</th><td colspan="4">MAC (optional)</td></tr>
<tr><th>Bytes
p..(q–1)</th><td colspan="4">Padding (block ciphers only)</td></tr>
</table>

TLS AP: Alert Protocol



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 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

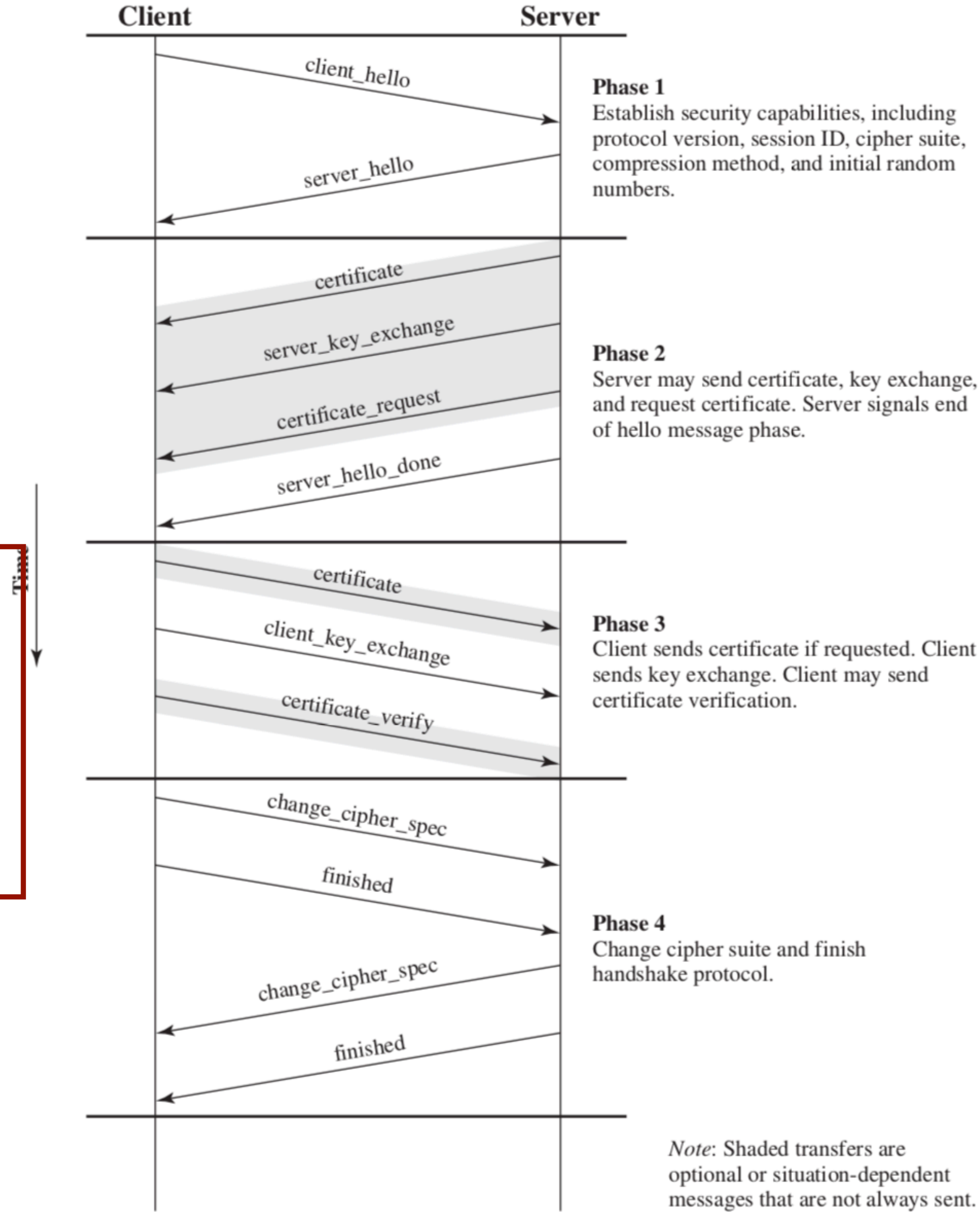
TLS Handshake Phases

- 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 !



For your practical (hands-on) self-study

- Use wireshark
  - Capture filter to filter traffic between your localhost machine and an HTTPS (TLS) server endpoint
  - In client try to use:
    - Openssl to debug TLS client connections

      ex., openssl s_client -connect example.com:443
    - Can also use simple Java-based TLS Client Programs as you can fond in Lab 8 materials
  - Inspect the TLS traffic (mainly the Handshake) traffic between the client side and the server side

  - This is how you will learn a lot ! ;-))

TLS in more detail …

Details on TLS: Flexibility, Security and Detailed End-Point Parameterizations for Handshake and TLS Session-Establishment

TLS Key Exchanges in the Handshake

- 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)

  - TLS_PSK and TLS_SRP
  - Fortezza (not used now is TLS)

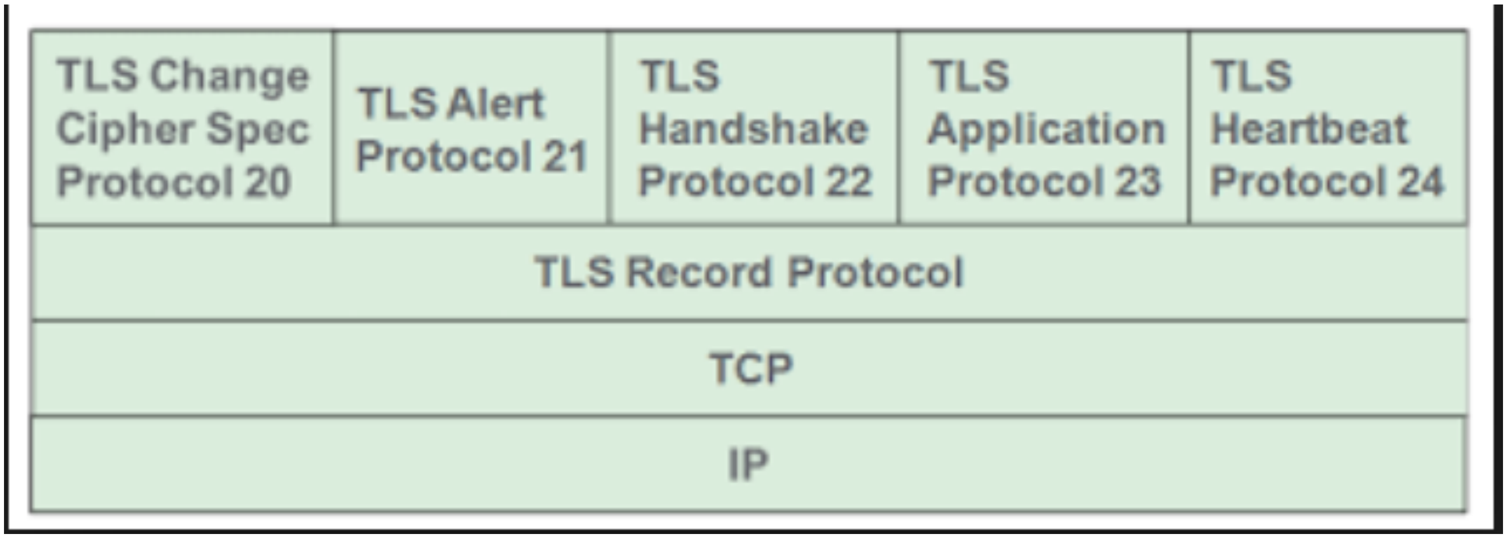
- Flexibility and Authentication Modes for Key-Exchanges:
  - Server Only (Unilateral Server Authentication)
  - Client Only (Unilateral Client Authentication)
  - Mutual Authentication (Client and Server)
  - No Authentication (Anonymous)

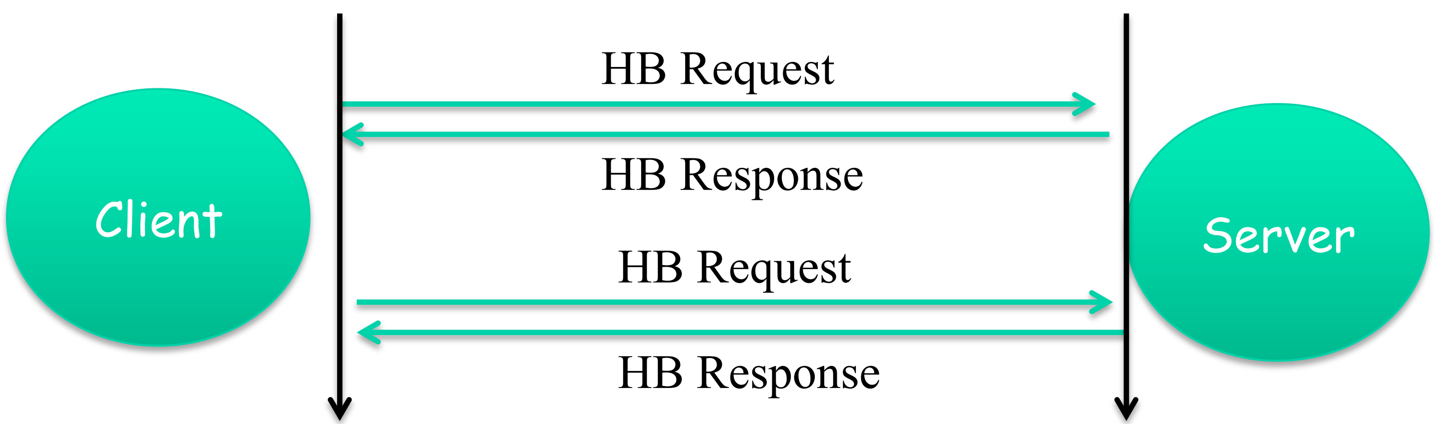
Key exchange/agreement and authentication

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 - HB (Heartbeat Protocol Extension)

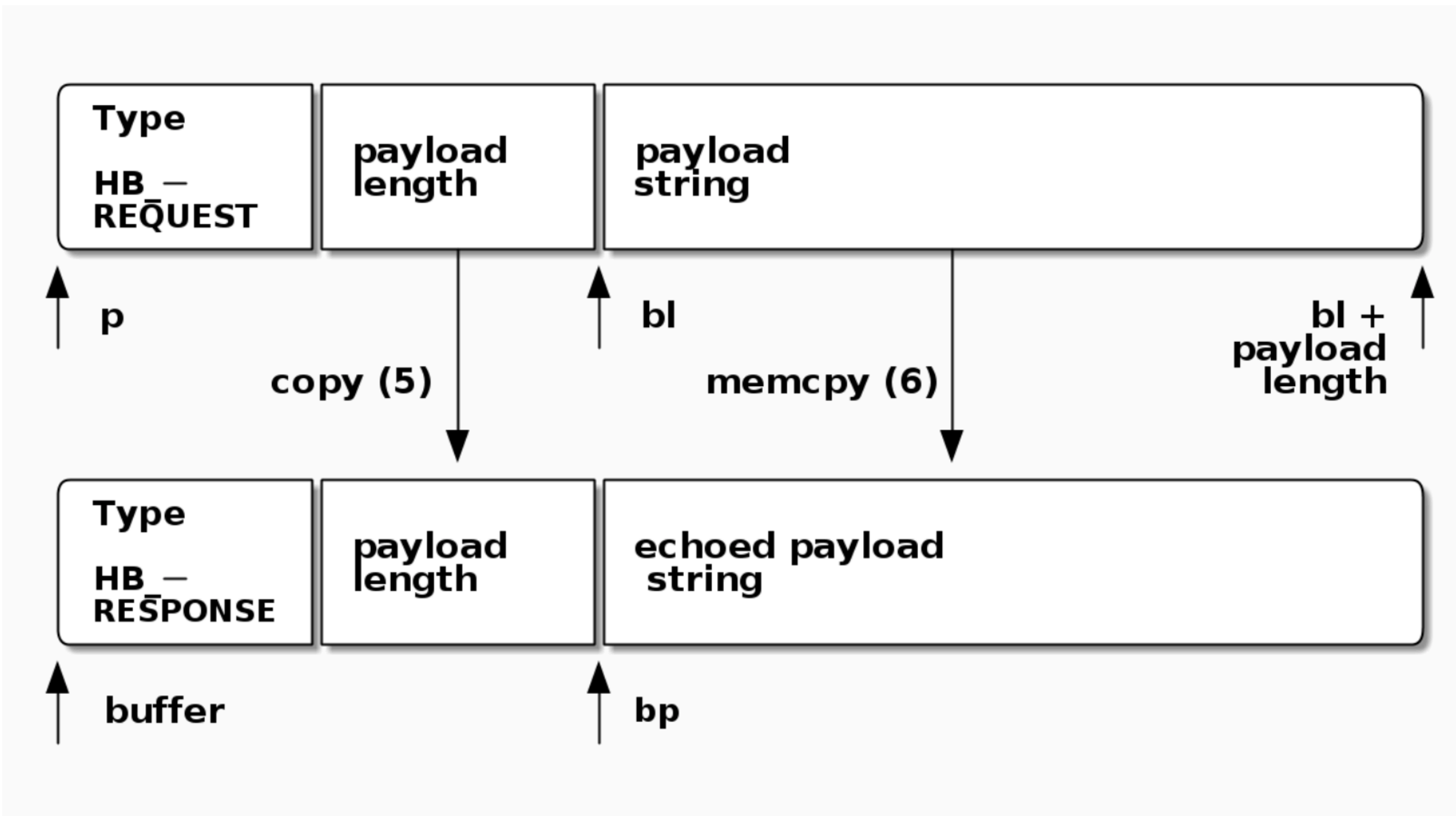
Introduced in 2012, RFC 6520 (as a keep-alive control to maintain the connection state)





TLS - HB (Heartbeat Protocol Extension)

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



Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

HTTPS (HTTP/SSL), RFC 2818 (HTTP over TLS)

- URLs: https://...
- HTTPS Server Port: 443 (standard port)
- HTTPS uses TLS/TCP
- When HTTPS is used, the following elements of the communication are encrypted:
  - Client Requests: URLs of requested resources
  - Contents of HTTP headers, Request/Response HTTP Contents
  - All contents of browser forms (filled in by browser user)
  - Cookies from browsers to server and vice-versa

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.

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

Web insecurity vs. TLS Cryptosuites

TLS Cryptographic Suites:

Negotiation options (handshake), flexibility, complexity (design vs. implementation)
vs. Security vs. Insecurity

One relevant issue for Web Security concerns:
See (ex.):
OWASP (Open Web Application Security Project) Foundation
> Top Ten Vulnerability Rank (2010, 2013, 2017)
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

OWASP:
Ten Most Critical Web App Security Risks

1. Injection
2. Broken Authentication
3. Sensitive Data Exposures (*)

   Weak-Ciphers, No PBS/PFS provisioning, unsecure PWD-encryption/hashing w/ impact on TLS misconfigurations
4. XML External Entities (or XEE Attacks)
5. Broken Access Control
6. Security Misconfigurations
7. Cross Site Scripting (or XSS Attacks)
8. Unsecure Deserialization
9. Layer-Below App. Components w/ Unknown Vulnerabilities
10. Insufficient Logging and Monitoring

TLS and SSL Versions (Installation Base)

After Apr/2016, latest versions of major browsers adopt TLS V1.1, 1.2, 1.3
… but many vulnerabilities are induced by old browsers and old versions of OSes and many implementations (libraries or App packaged implementations)

TLS v1.3 is recent: in Safari 12.0, Opera v60, Firefox v66, Google Chrome v73
See here: https://en.wikipedia.org/wiki/Transport_Layer_Security

Protocol version	Website support[59]	Security[59][60]
SSL 2.0	1.9%	Insecure
SSL 3.0	7.8%	Insecure[61]
TLS 1.0	68.8%	Depends on cipher[n 1] and client mitigations[n 2]
TLS 1.1	77.9%	Depends on cipher[n 1] and client mitigations[n 2]
TLS 1.2	95.0%	Depends on cipher[n 1] and client mitigations[n 2]
TLS 1.3	13.6%	Secure

Client and Server Endpoints must agree In the protocol version

Ciphersuites and related parameterizations

- The established *ciphersuites* (standardized cryptography) are defined in different versions of SSL and TLS
  - Dynamically negotiable in different TLS and SSL versions and Handshake Sub-protocols, between clients (ex., browsers) and servers (ex., HTTPS servers):
    - Clients: propose supported ciphersuites (typically in a set) and Keysizes
    - Servers: accept the ciphersuite (from the client set)
    - Relevant issue: possible bad default settings

- Standardization of different client or server certificate types, digital signatures supported: correct verification in implementations and operational trust assumptions are very important issues !

- Padding processing and insufficient mitigation of DoS/DDoS is another security standardization issue (remember the base RLP message format and design implications)

TLS Authentication and Key-Exhange Methods

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

Cipher security against publicly known feasible attacks

Cipher			Protocol version						Status
Type	Algorithm	Nominal strength (bits)	SSL 2.0	SSL 3.0 [n 1][n 2][n 3][n 4]	TLS 1.0 [n 1][n 3]	TLS 1.1 [n 1]	TLS 1.2 [n 1]	TLS 1.3	
Block cipher with mode of operation	AES GCM[47][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	AES CCM[48][n 5]		N/A	N/A	N/A	N/A	Secure	Secure	
	AES CBC[n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	Camellia GCM[49][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	Camellia CBC[50][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	ARIA GCM[51][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	ARIA CBC[51][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	SEED CBC[52][n 6]	128	N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	3DES EDE CBC[n 6][n 7]	112[n 8]	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	
	GOST 28147-89 CNT[46][n 7]	256	N/A	N/A	Insecure	Insecure	Insecure	N/A	Defined in RFC 4357
	IDEA CBC[n 6][n 7][n 9]	128	Insecure	Insecure	Insecure	Insecure	N/A	N/A	Removed from TLS 1.2
	DES CBC[n 6][n 7][n 9]	56	Insecure	Insecure	Insecure	Insecure	N/A	N/A	
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	Forbidden in TLS 1.1 and later
	RC2 CBC[n 6][n 7]	40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
Stream cipher	ChaCha20-Poly1305[57][n 5]	256	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	RC4[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	Prohibited in all versions of TLS by RFC 7465
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
None	Null[n 12]	–	N/A	Insecure	Insecure	Insecure	Insecure	N/A	Defined for TLS 1.2 in RFCs

HMACs

HMACs standardized by RFC 2104

Data integrity

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
HMAC-MD5	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
HMAC-SHA1	No	Yes	Yes	Yes	Yes	No	
HMAC-SHA256/384	No	No	No	No	Yes	No	
AEAD	No	No	No	No	Yes	Yes	
GOST 28147-89 IMIT[46]	No	No	Yes	Yes	Yes		Proposed in RFC drafts
GOST R 34.11-94[46]	No	No	Yes	Yes	Yes		

Standardized Functions in TLS endpoints

- Cryptographic computations are different in different SSL and TLS versions

- Key and MAC Generation for Cryptogrphic Computations MACs: TLS v1.2 (RFC 5246)

- Other critical cryptographic computations:
  - PRE-MASTER Secrets
  - MASTER SECRET CREATION : 48 bytes (384 bits)
  - KEY-BLOCK generation by PRF Pseudorandom Function based on HMACs from previous random seeds and shared secrets along the handshake exchanged parameters

See the bibliography

Ciphersuites and related parameterizations

- Important security measures (default baseline):
  - Avoidance of SSL versions and TLS 1.0
  - Avoidance of considered "Weak Cryptosuites"
  - Appropriate key sizes (RSA, DSA keys >= 2048 bits) for the proper protection of secure envelopes for the establishment of session or MAC keys and security transport and session association parameters
  - The problem of "possibly unsecure ECCs" (on going problem)
  - Only Ephemeral Diffie Hellman Agreements with parameterizations for public and private numbers >= 2048 bits
    - Trade-off for Efficiency: fixed shared initialization parameters (primitive root and prime number for the modular operations)
  - Problem: scale, installation base vs. "relaxed" TLS server configurations

See the bibliography and also

- LABs (hands-on study and verifications in tracing Handshake Protocol)
- Security auditing on possible weak ciphersuites and vulnerabilities

SSL/TLS Attacks

SSL/TLS Attacks and Impact

- Design implications
- Implementation implications

TLS and SSL Attacks vs. Countermeasures

The history of SSL (versions 1., 2., 3) and TLS (versions 1.1 and 1.2) attacks and related countermeasures (as many other protocols) that the "perfect secure protocol" and "the perfect implementation strategy for security vs. flexibility vs. usability tradeoffs" have not been achieved.

Constant back-and-forth between threats and counter-measures has been a constant struggle ….

New complexities and tradeoffs =>
New threats and threat models =>
New adversarial conditions =>
New counter-measures (patching ?) =>
Evolution/Revision of standardization =>
Evolution/Revision of Implementations

TLS and SSL Attacks

Attacks involving PKI and X509 Certificates' Management and Validation

Attacks against the Handshake Protocol

Attacks on the record layer protocol

- **BEAST** (Browser-Exploit Against SSL/TLS): Crypto Attack (Chosen-Plaintext Crypto. Attack)
- **CRIME** Attack (Compression Ratio Info-Leak Cookies): Session Hijacking on TLS protected cookies and compression/decompression processing, can break the authentication of TLS sessions
- Attacks on PKIs and Certification-Chain validations in many libraries, overtime:
  - OpenSSL, GnuTLS, JSSE, ApacjeHttpCLient, Weberknetch, cURL, PHP, Python, and other Applications with integrated Packaged TLS processing
- **HackersChoice Attack**: DoS against the Handshake Proecessing Computations for usual Server-Only Authentication Modes currently used

TLS and SSL Attacks

Heatbleed Attack:

Endpoint from client side TLS negotiation of Heartbeat messages

Attack against TLS SW implementations (Bad TLS Heartbeat implementation) causing access to "memory mapped" security association parameters

https://en.wikipedia.org/wiki/Heartbleed

POODLE (Padding Oracle On Downgraded Legacy Encryption)

Man in the Middle Attack: exploit which takes advantage of Internet and security software clients' fallback to "weak-ciphersuites' negotitated and accepted by the HTTPS server endpoint

https://en.wikipedia.org/wiki/POODLE

More information on different TLS Attacks

Recent and On-Going Research on TLS vulnerabilities:

- BEAST: T. Duong, J. Rizzo, BEAST Proof of concept
- CRIME and Breach Attacks
- PKI Attacks:
- Timming Attacks on Padding
- Poodle Attack
- RC4 Attacks
- Trubcation Attacks
- Unholy Attack
- Sweet32 Attack
- Heartbleed

TLS vulnerabilities and impact

Attacks	Security				
	Insecure		Depends	Secure	Other
Renegotiation attack	1.2% (−0.1%) support insecure renegotiation		0.4% (±0.0%) support both	96.2% (+0.1%) support secure renegotiation	2.2% (±0.0%) no support
RC4 attacks	<0.1% (±0.0%) support only RC4 suites	6.0% (−0.3%) support RC4 suites used with modern browsers	28.5% (−0.7%) support some RC4 suites	65.5% (+1.0%) no support	N/A
CRIME attack	2.4% (−0.1%) vulnerable		N/A	N/A	N/A
Heartbleed	0.1% (±0.0%) vulnerable		N/A	N/A	N/A
ChangeCipherSpec injection attack	0.8% (±0.0%) vulnerable and exploitable		4.7% (−0.2%) vulnerable, not exploitable	92.6% (+0.4%) not vulnerable	1.9% (+0.1%) unknown
POODLE attack against TLS (Original POODLE against SSL 3.0 is not included)	2.1% (−0.1%) vulnerable and exploitable		N/A	97.1% (+0.2%) not vulnerable	0.8% (−0.1%) unknown
Protocol downgrade	23.2% (−0.4%) TLS_FALLBACK_SCSV not supported		N/A	67.6% (+0.7%) TLS_FALLBACK_SCSV supported	9.1% (−0.4%) unknown

Current relevance of TLS 1.3

TLS 1.3, IETF Defined in 2014
(Today coexisting w/ TLS 1.2 ...)

TLS 1.3 removes:

- Compression
- Not Authenticated Modes and Handshake Exchanges
- Considered Weak Chiphers
- Static RSA and DH Key Exchange Methods
- 32 bit timestamps as part of Random parameters in Client/Server Hello Handshake Messages
- Renegotiation of secrets from previous established parameters
- Heartbeat Protocol
- Change Cipher Spec Protocol
- RC4
- Use of MD5, SHA-1 and SHA-224

Current relevance of TLS 1.3

TLS 1.3, IETF Defined in 2014
(Today coexisting w/ TLS 1.2)

TLS 1.3 includes (for improving the tradeoff security and efficiency):

- DH and EC-DH for Key Exchanges (no RSA Key Exchange)
- Simplification of "one-shot" Handshake rounds (one round trip time handshake), by reordering/piggybacking (or pipelining) the handshake sequence
- Client side must send authenticated parameters, before the negotiation of cipher suites when client-authentication or mutual-aiuthentication is adopted

Classic bibliography on TLS dangerous issues

- The most dangerous code in the world: validating SSL certificates in non-browser software, M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh and V. Shmatikov, ACM CCS 2012
- Forward Secrecy and TLS Renegotiation: F. Giesen et al., On the Security of TLS Renegotiation, ACM CCS 2013

- T. Jager et al., On the Security of TLS v1.3 and QUIC against Weaknesses in PKCS#1 .5 Encryption, ACM CCS 2015

- The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations , Eyal Ronen, Robert Gillham, Daniel Genkin, Adi Shamir, David Wong, and Yuval Yarom, Dec 2018

See also:

- https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2019/february/downgrade-attack-on-tls-1.3-and-vulnerabilities-in-major-tls-libraries/ , Nov 2018

- Selfie: reflections on TLS 1.3 with PSK, Nir Drucker and Shay Gueron , https://eprint.iacr.org/2019/347.pdf,

A Recent Research Bibliog. … (TLS Vulnerabilities and Proposed Solutions)

ACM CCS 2018

- Pseudo Constant Time Implementations of TLS Are Only Pseudo Secure
  Eyal Ronen (Weizmann Institute of Science), Kenny Paterson (Royal Holloway, University of London), Adi Shamir (Weizmann Institute of Science)
- Partially specified channels: The TLS 1.3 record layer without elision
  Christopher Patton (University of Florida), Thomas Shrimpton (University of Florida)
- The Multi-user Security of GCM, Revisited: Tight Bounds for Nonce Randomization
  Viet Tung Hoang (Florida State University), Stefano Tessaro (University of California Santa Barbara), Aishwarya Thiruvengadam (University of California Santa Barbara)

Usenix Sec. Symp. 2018:

- Return Of Bleichenbacher's Oracle Threat (ROBOT), H. Bock et al.,

IEEE Sympo. On Security and Privacy 2018

- A Formal Treatment of Accountable Proxying over TLS, Karthikeyan Bhargavan at al.

IEEE Synp. On Sec and Privacy 2019:

- The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations, E. Ronen et al.

NDSS 2018:

- Removing Secrets from Android's TLS. Jaeho Lee *(Rice University)* and Dan S. Wallach *(Rice University).*
- TLS-N: Non-repudiation over TLS Enablign Ubiquitous Content Signing. Hubert Ritzdorf *(ETH Zurich)*, Karl Wust *(ETH Zurich)*, Arthur Gervais *(Imperial College London)*, Guillaume Felley *(ETH Zurich)*, and Srdjan Capkun *(ETH Zurich).*

NDSS 2019:

- The use of TLS in Censorship Circumvention. Sergey Frolov, Eric Wustrow

TLS in current practice …

- TLS v1.2 and v 1.3 is the base of current baseline security
- A strict control on considered secure ciphersuites, and parameterizations must be addressed as baseline countermeasures against the more prevalent attacks:

Hands-on (Ref. Example):
https://www.ssllabs.com/ssltest/

Outline

- WEB security issues
  - Web traffic security threats: the role of SSL and TLS
  - TCP/IP Stack and TLS
- TLS: Session-Security vs. Transport Security Layers
- TLS architecture and protocol stack
  - Sub-Protocols: RLP, CSP, AP, HP and HB
- TLS vs. HTTPS
- TLS Practical Security: Weak Ciphersuites and Security Tradeoffs
- Web Security and Threats beyond TLS

Threats beyond TLS

- Remember: TLS is designed to protect transport-based communication channels (UDP or TCP)

- TLS and HTTPS don't means WEB Security: it is just one of the security elements for WEB Security
  - See: OWASP Web Security Attacks and Top-Ten Vulnerabilities
  - OWASP: See https://www.owasp.org/index.php/Main_Page

- Relates with communication security properties, not considering intrusions on endpoints

- The required secure processing in implementing the TLS endpoints (transport and session states and sensitive security association parameters and correct and trusted TLS state-machine execution control) is out of scope of TLS protocols' security standardization effort

Threats beyond TLS

- SW and Application Level Security
  - Can use TLS but with Application-Level Vulnerabilities
  - Bad or unmatched use of TLS Parameterizations
- PKI SW based vulnerabilities
- Related Attacks: Attacks against Time Synchronization Protocols
- Unsecure management of X509 certificates and incorrect verification and validation of x509 (namely X509v3 extension attributes) in the TLS handshake of Certification chains: Recurrent vulnerabilities in many TLS libraries
  - This included deficient management of the "trusted root assumption" in acceptance or pre-installed X509 certificates (including CA certificates)
  - Incorrect operation and management of X509 certificates' life-cycles – include lack of proper control for CRLs and management of OCSP endpoints
- DoS or DDoS
  - No effective protection on TLS…. It Can be aggravated w/ TLS

Revision: Suggested Readings and Study

Readings:

W. Stallings, Network Security Essentials – Applications and Standards

- Ed.. 2017 Chap 6 Transport Layer Security, 6.1-6.4, pp. 187-208
- Ed. 2011 Chap 5 Transport Layer Security, 5.1-5.4, pp. 139-162

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