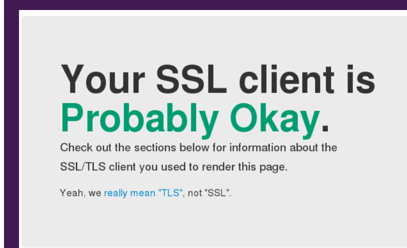
+



UCL
Université catholique de Louvain

WAP | SSL/TLS

INGI2347: COMPUTER SYSTEM SECURITY (Spring 2014)

Marco Canini

Lecture slides adapted from UCL INGI2347 by Gildas Avoine
Reproduced with permission

+

Plan for today

Lecture 10

- WiFi Protection Access (WPA) NEXT
  - Protocol phases
  - WPA attacks
- SSL / TLS
  - Protocol details

Recall: Weaknesses of WEP

1. IV is too short and not protected from reuse
2. The way keys are constructed from the IV makes it susceptible to weak key attacks (FMS attack)
3. There is no effective detection of message tampering (message integrity)
4. It directly uses the master key and has no built-in provision to update the keys
5. There is no protection against message replay

WiFi Protected Access (WPA)

- Goal: replace WEP
- WPA was developed as a urgent patch before the publication of the 802.11i standard (WPA2)
- WPA was designed to be deployed on old WiFi-compliant through a firmware update
- WiFi-compliant devices now must implement WPA2 since 2006

WPA Features

- A seq. number is used to prevent replay attacks
- Initialization Vector (IV) is 48-bit long
- User authentication vs. device-only authentication in WEP
- Keys are dynamically refreshed
- WPA2 uses AES instead of RC4 as in WEP and WPA

WPA Data Encryption

Two major suites of algorithms:

- Temporal Key Integrity Protocol (TKIP)
  - RC4 + Message Integrity Check (MIC)
- Counter Mode CBC-MAC Protocol (CCMP)
  - AES in Counter Mode (WPA2 only)

Changes from WEP to TKIP

Purpose	Change	Weakness addressed
Message Integrity	Add a message integrity protocol to prevent tampering (implementable in software on a low-power microprocessor)	(3)
IV selection and use	Change the rules for how IV values are selected and reuse the IV as a replay counter	(1)(5)
Per-Packet Key Mixing	Change the encryption key for every frame	(1)(2)(4)
IV Size	Increase the size of the IV to avoid ever reusing the same IV	(1)(4)
Key Management	Add a mechanism to distribute and change the broadcast keys	(4)

Modes: Personal vs. Enterprise

- Personal WPA utilizes pre-shared keys (PSK): all devices use same passphrase
  - User need a 256-bit passphrase (64 hex digits or 8 to 63 ASCII characters)
  - Authentication between client and Access Point (based on EAP-MD5)
  - Suited for home or small office infrastructure
- Enterprise WPA uses an IEEE 802.1X Authentication Server that distributes different keys to users
  - User authentication
  - Requires an authentication server (e.g. Radius)
  - Centralizes management of user credentials

IEEE 802.11i Operational Phases

Supplicant — Party being authenticated

Authenticator — Access Point

Authentication server — Radius

1. Security Policy Agreement

2. 802.1X Authentication

3. Key derivation and distribution

Master Key (MK) by Radius

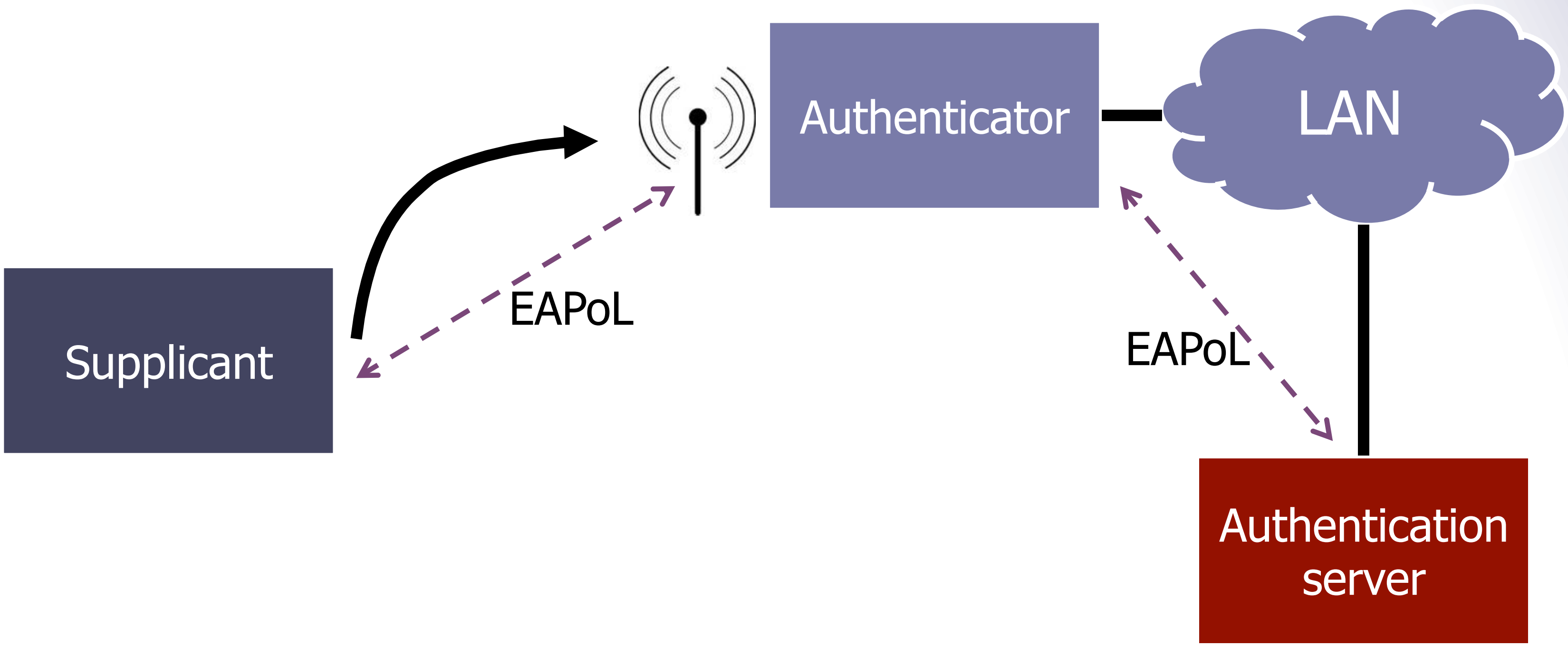
4. Data confidentiality and integrity

Phase 2

802.1X Authentication

IEEE 802.1X Authentication

- Extensible Authentication Protocol (EAP) over LAN: EAPoL



Extensible Authentication Protocol (EAP)

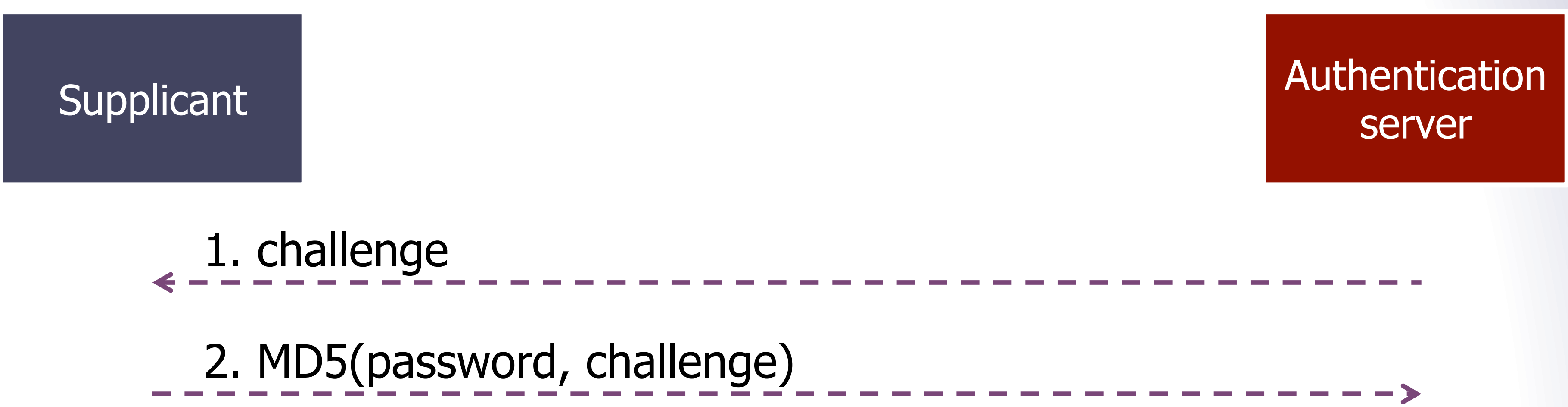
- A framework for transporting authentication protocols
  - Not really an authentication protocol itself
- Four types of packet:
  - request, response, success and failure
- Request packets are issued by the authenticator and solicit a response from the supplicant
- Any number of request-response exchanges may be used to complete the authentication
- A success (resp. failure) packet is sent to the supplicant if the authentication succeeded (resp. failed)

EAP variants

- Legacy based methods
  - EAP-MD5
- Certificate based methods
  - EAP-TLS, EAP-TTLS, PEAP
- Password based methods
  - LEAP, SPEKE
- And many others …

EAP-MD5 (not secure!) [RFC2284]

- Authentication of the client only
- MD5 message hashing algorithm
- Very simple EAP method



EAP-TLS [RFC5216]

- Mutual authentication and key exchange
- Public key certificates, also at the client
- Strong authentication but requires PKI

EAP-TTLS (Tunneled TLS) [RFC5281]

- Mutual authentication and key exchange
- Public key certificates, only at the AS
- Allows less secure authentication methods through a secure channel

Phase 3

Key derivation and distribution

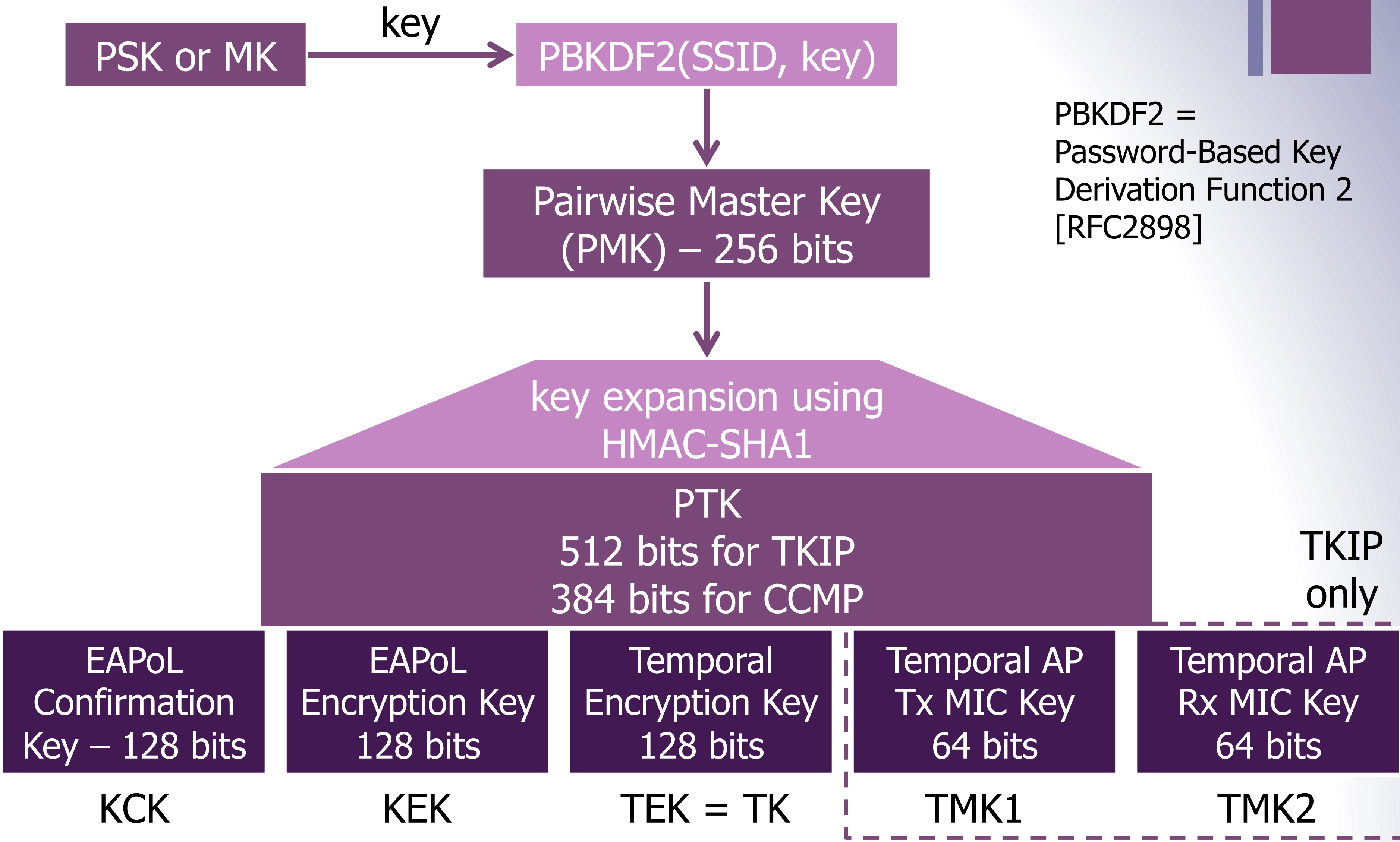
Key derivation and distribution

1. Master Key (MK) transmission

2. 4-way handshake for key derivation and distribution:
Pairwise Transient Key (PTK), used for unicast
Group Transient Key (GTK), used for multicast

3. Group Key handshake for GTK renewal

Pairwise Key Hierarchy



PBKDF2 =
Password-Based Key Derivation Function 2
[RFC2898]

Pairwise Key Hierarchy

KCK (Key Confirmation Key)

- Key for authenticating messages during the 4-way handshake and Group Key handshake

KEK (Key Encryption Key)

- Key for ensuring data confidentiality during the 4-way handshake and Group Key Handshake

Pairwise Key Hierarchy

TK (Temporary Key)

- Key for data encryption (used by TKIP or CMMP)

TMK (Temporary MIC Key)

- Key for data authentication (used only by Michael with TKIP)
- A dedicated key is used for each side of the communication

Further Reading

- Details of 4-way handshake and Group Key handshake:

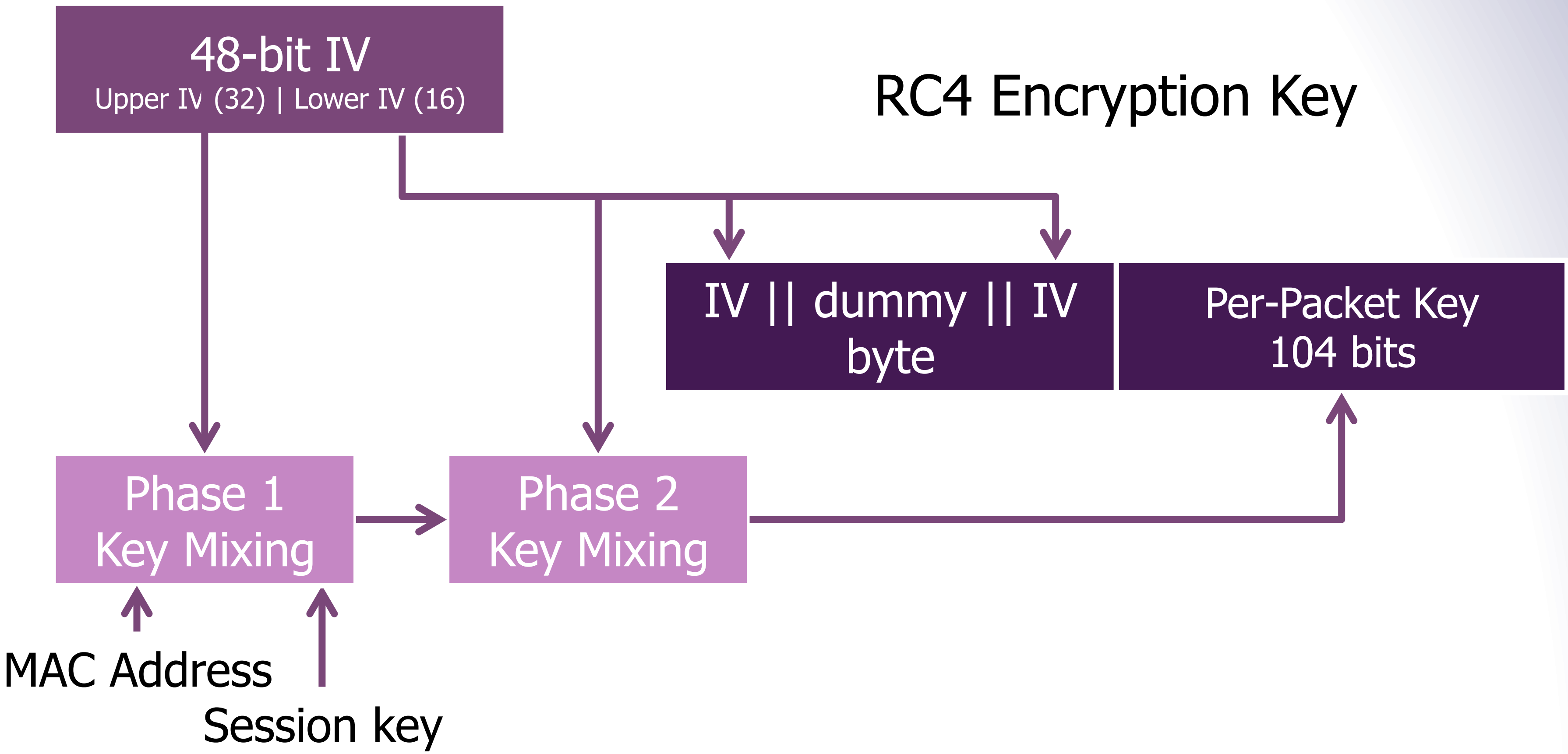
  http://bit.ly/OQooma

Phase 4

Data confidentiality and integrity

TKIP Key-Mixing

RC4 stream cipher used with a 128-bit per-packet key



TKIP MAC

- Message Integrity Code (MIC)
- Michael algorithm instead of CRC32
- Michael is keyed
- Strongest MIC that was available with most older network cards
- Due to weaknesses of Michael, the network is shut down during one minute if two frames fail to pass Michael's check
  - Generation of new keys and re-authentication are required

Further Reading

- Details of AES-CCMP:

  http://bit.ly/NIXhZk

Attacks on WPA

Attack on Weak Password

- All the keys derive from PSK

PMK = PBKDF2(SSID, PSK) then 4-way handshake to derive the other keys

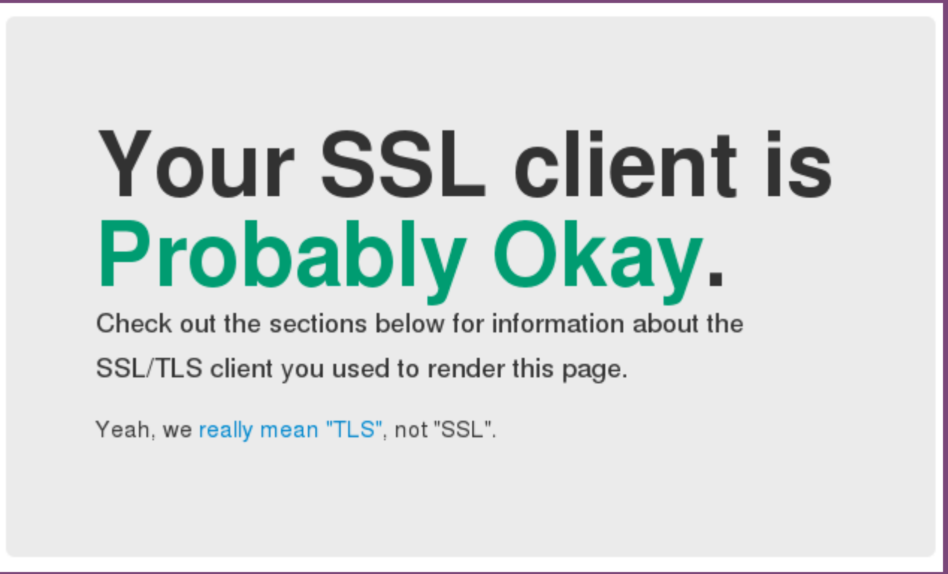
- Handshake is eavesdropped and used to check a candidate passphrase:
  - For every candidate passphrase, compute the associated PMK
  - Compute the PTK (4 HMAC-SHA1 computed on PMK and random values)
  - Compute the MIC (1 HMAC-SHA1 or MD5) and compare with the eavesdropped one
- To mitigate the problem, the network's SSID should not match any entry in the top 1000 SSIDs [https://wigle.net/gps/gps/main/ssidstats]

Attack on TKIP

- Packet injection
  - E.Tews and M.Beck (2008)
  - T.Ohigashi and M.Morii (2009)
  - F.Halvorsen, O.Haugen, M.Eian, S. Mjølsnes (2009)
  - 596 bytes within 18 min 25
- Decryption of packets from AP to client
  - M.Beck (2010)

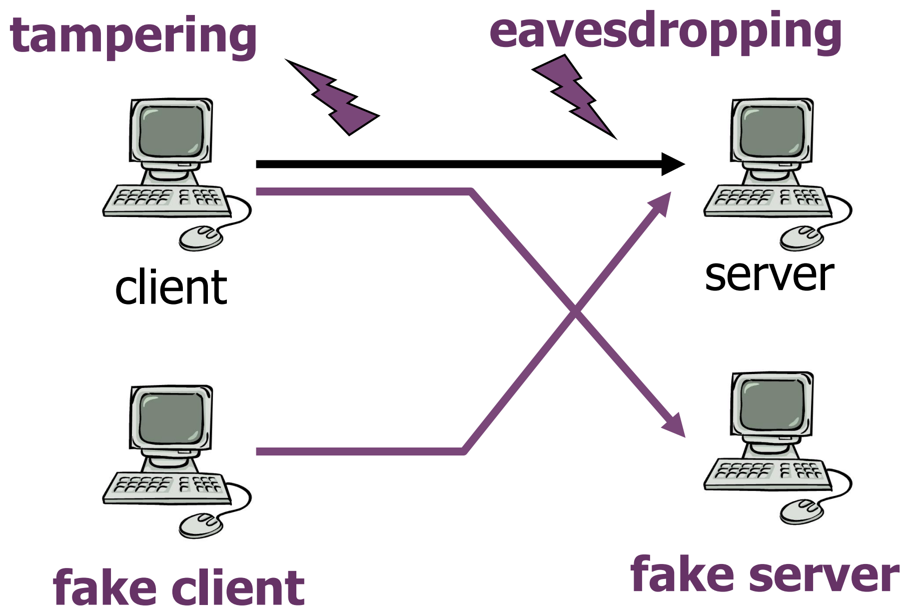
Recap WPA

	WPA	WPA2
Enterprise Mode	Authentication **IEEE 802.1X/EAP** Encryption **TKIP/MIC**	Authentication **IEEE 802.1X/EAP** Encryption **AES-CCMP**
Personal Mode	Authentication **PSK** Encryption **TKIP/MIC**	Authentication **PSK** Encryption **AES-CCMP**


Your SSL client is
Probably Okay.
Check out the sections below for information about the
SSL/TLS client you used to render this page.
Yeah, we really mean "TLS", not "SSL".

SSL / TLS

SSL Primer



- Authentication of server based on public key
- Trusted third party: certification authority (CA)

Secure Sockets Layer (SSL)

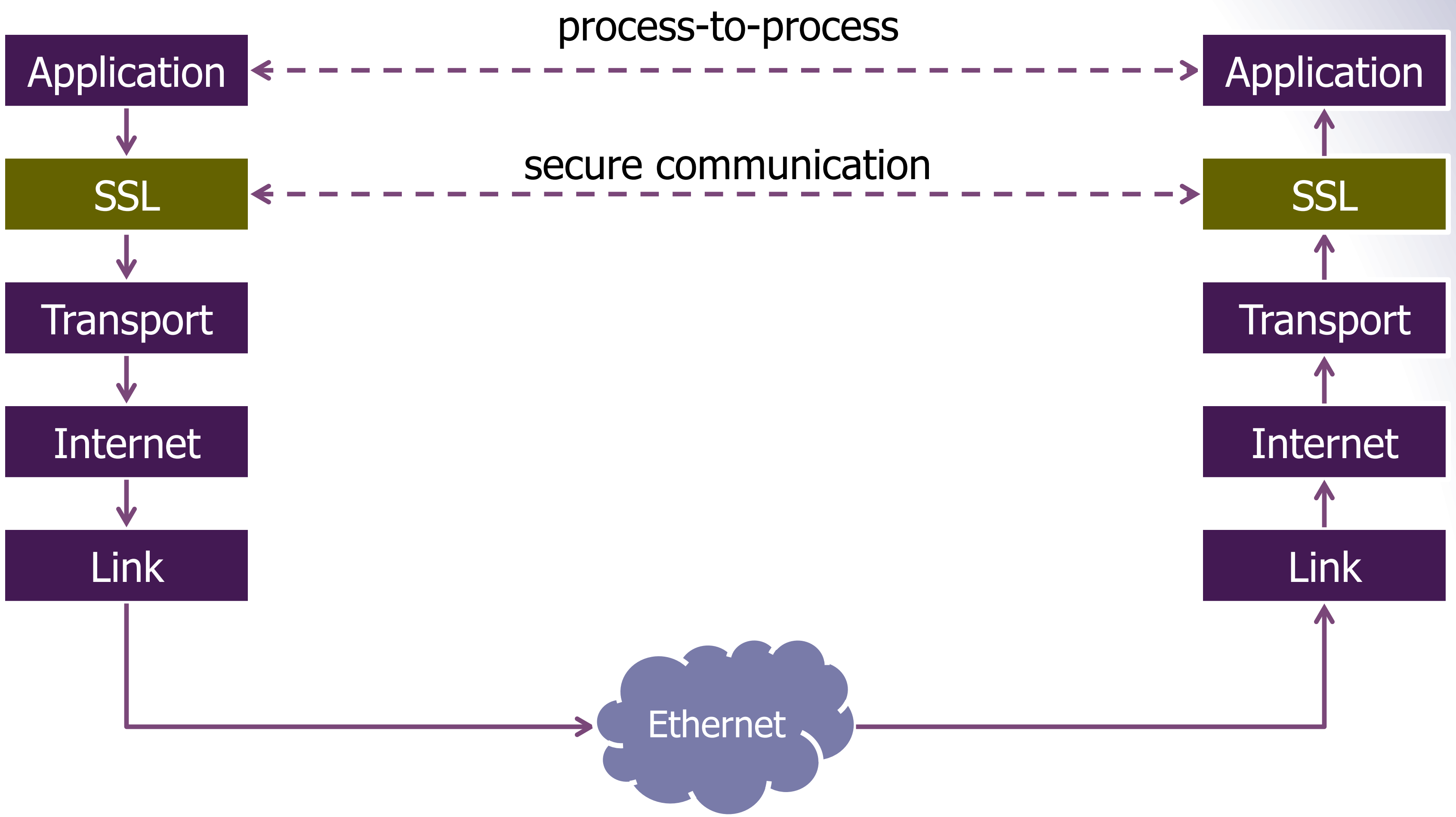
- Most widely deployed security protocol in the world
- SSL was developed by Netscape to offer secure access to web servers (HTTPS)
- History
  - SSL v1.0 never publicly released
  - SSL v2.0 released in 1994 (flawed)
  - SSL v3.0 released in 1996, leads to TLS 1.0 in 1999

Transport Layer Security (TLS)

- TLS is an IETF standard based on SSL v3.0
  - Slight modifications compared to SSL v3.0
  - TLS v1.0 and SSL v3.0 do not interoperate
  - TLS v1.0 sometimes called SSL v3.1
  - TLS v1.0 defined in RFC 2246
  - TLS v1.2 updated in RFC 5246 (August 2008)

- Current version (March 2011)
  - TLS v1.2 (prohibits SSL v2.0)
  - RFC 6176

SSL in the layered model



Approaches

Create a new protocol from an existing protocol

- Examples:
  HTTP (80) / HTTPS (443), FTP (21) / FTPS (990), SMTP (25) / SMTPS (995), POP3 (110) / POP3S (995), IMAP (143) / IMAPS (993)
- Disadvantage: only clients supporting TLS can connect
- Advantage: we are sure that communications are secure

Extend a protocol to negotiate SSL/TLS

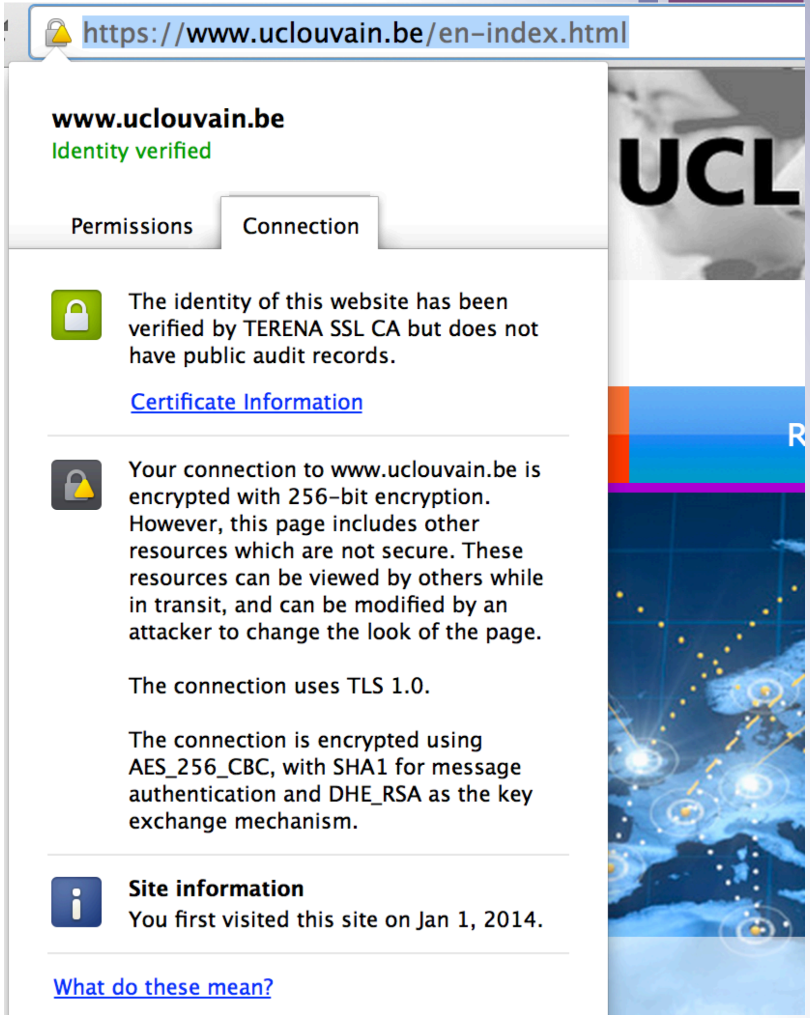
- Examples: (E)SMTP, POP3, IMAP, with the help of the STARTTLS command the client can ask to use TLS
- Advantage: the client is not required to support TLS to use the service

OpenSSL


```
bash$ openssl s_client -connect www.uclouvain.be:443
CONNECTED(00000003)
[...]
Certificate chain
 0 s:/C=BE/L=Louvain-la-Neuve/O=Universit\xC3\xA9 Catholique de Louvain/OU=Portail UCL/CN=www.uclouvain.be
   i:/C=NL/O=TERENA/CN=TERENA SSL CA
[...]
Server certificate
-----BEGIN CERTIFICATE-----
MIIErDCCA5SgAwIBAgIRAOjy08jirG7k+6k8Ln7bZxQwDQYJKoZIhvcNAQEFBQAw
[...]
-----END CERTIFICATE-----
[...]
SSL handshake has read 5258 bytes and written 328 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 2048 bit
[...]
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: C0FE449DC7345355B4119A095C27DA72691326880FE52271FB2CB3B0DCF29FE0
    Master-Key: 7A8DE9425505930A2F11AFC241F9236ABA61DAC7BFC0A9709C6F887D819BAA42C5F1B7A9E01CC26945A[...]
```

Example: HTTPS

- TLS guarantees data confidentiality and authenticity (server, possibly client)
  - The server must have a certificate
  - The client can have one
    - e.g. e-banking

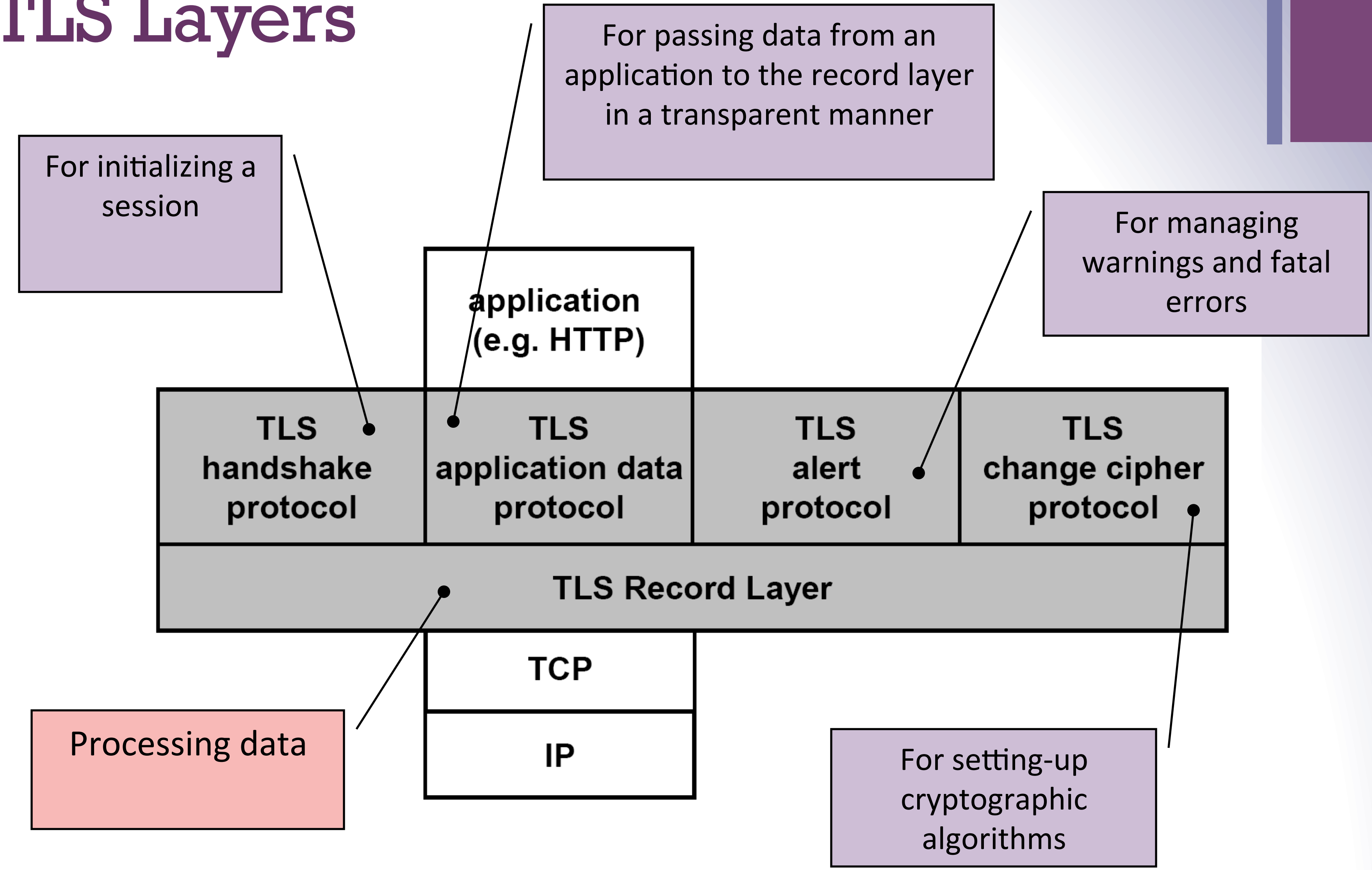


Example: Mail

- ESMTP (sending mail), POP3 (mailbox access), IMAP (better mailbox access)
  - TLS is implemented as a protocol extension
  - The use of TLS is optional (needs to be configured)
- By default these protocols send cleartext passwords
- TLS protects passwords and email contents

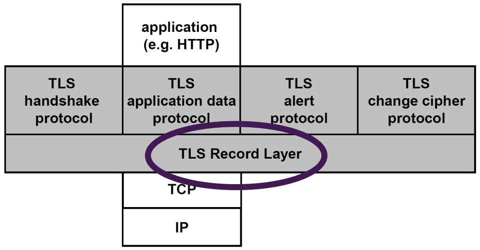
TLS Protocol

TLS Layers



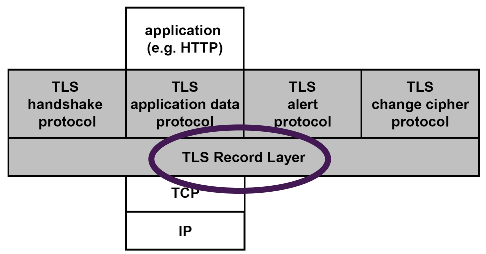
For initializing a session
For passing data from an application to the record layer in a transparent manner
For managing warnings and fatal errors
application (e.g. HTTP)
TLS handshake protocol
TLS application data protocol
TLS alert protocol
TLS change cipher protocol
TLS Record Layer
TCP
IP
Processing data
For setting-up cryptographic algorithms

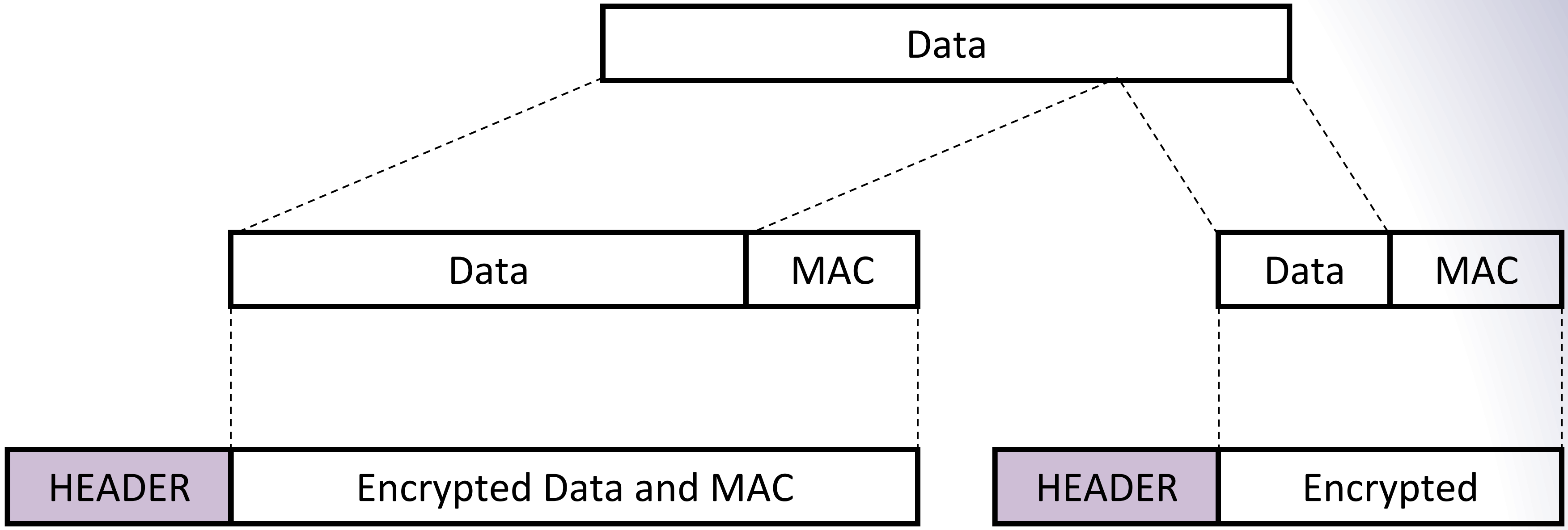
TLS Record Layer



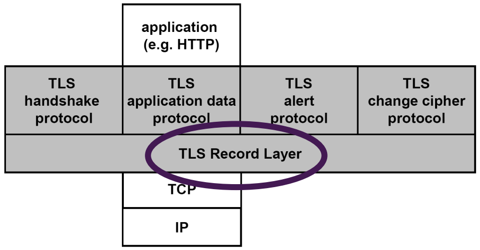
- Processing of data
  - Fragmentation
  - Compression (optional)
  - Authentication
  - Encryption
- It delivers processed fragments to the transport layer (TCP)
- At the receiving end, the inverse operations are carried out

Record Layer Summary





MAC Computation

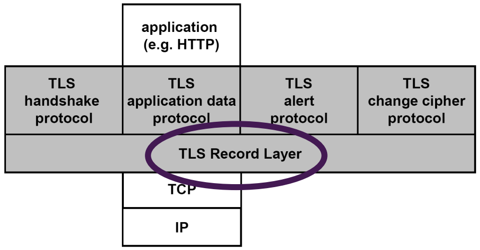


MAC = **Hash** (MAC_key || Pad2 ||

Hash (MAC_key || Pad1 || Seq || Length || Content))

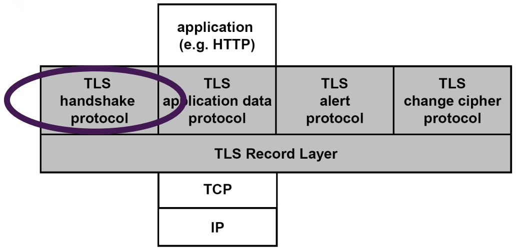
- MAC_key: secret shared by client and server
- Pad1, Pad2: pre-defined constants
- Seq: sequence number of this message
- **Hash**: Either HMAC-MD5 or HMAC-SHA1
- Length: Length in bytes of the compressed record
- Content: Compressed record

Encryption



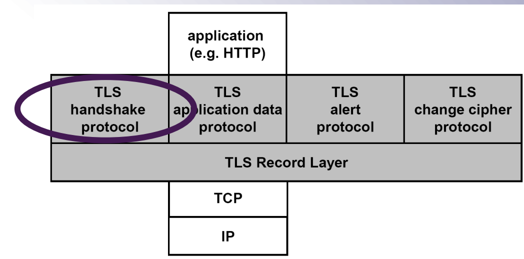
- Encryption is performed on compressed and authenticated records
- Block ciphers:
  - DES (40 bits or 56 bits), 3DES, IDEA, RC2 (40 bits)
  - AES (128 bits or 256 bits) in TLS v1.1
- Stream ciphers:
  - NULL, RC4 (40 bits or 128 bits)
- The client should refuse 40-bit keys if such a cipher is suggested by the server (warning enforced in TLS 1.1)

Handshake in Brief



- Negotiation of:
  - Protocol version (SSL 3.0, TLS 1.0, TLS 1.1)
  - Algorithms:
    - Key exchange (RSA, Diffie-Hellman)
    - Encryption (DES, 3DES, IDEA, RC4, RC2, AES)
    - MAC (HMAC-MD5, HMAC-SHA)
    - The client proposes the desired algorithms in order of preference, the server chooses
- Optional authentication of the partner using a certificate
- Messages are not encrypted
- Last messages authenticate the exchange

Handshake Exchanges



Client_Hello (crypto, random)

Server_Hello (crypto, random)

Server Certificate

Server_Hello_Done

Client_Key_Exchange

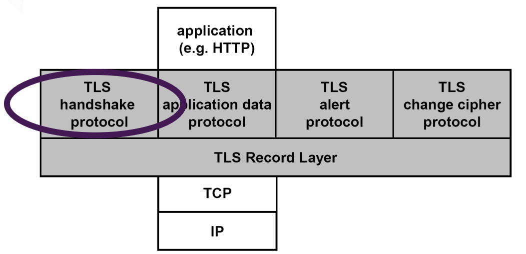
Change_Cipher_Spec

Handshake_Finished

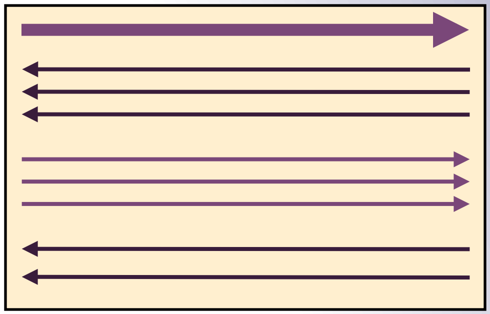
Change_Cipher_Spec

Handshake_Finished

Client_Hello Content

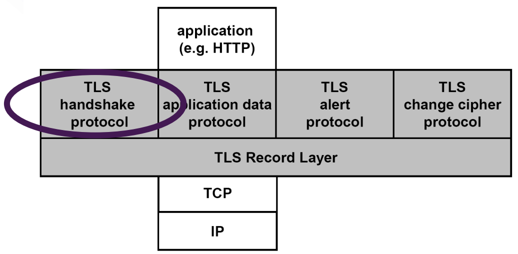


- Goal
  - Used by the client to initiate SSL session
  - Sent in clear without signature

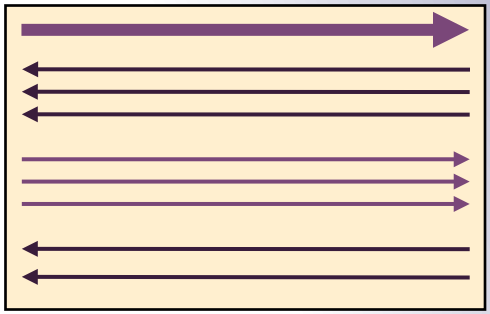


- Content
  - Protocol Version
  - 32 bytes long random number
  - Composed of two parts:
    - 4 bytes Unix timestamp (number of seconds since 01/01/1970)
    - 28 bytes random number
  - Optional Session Identifier
    - Each SSL session has an identifier which can be used later to restart a session
  - List of supported Ciphers
  - List of supported Compression Methods

Client_Hello Crypto

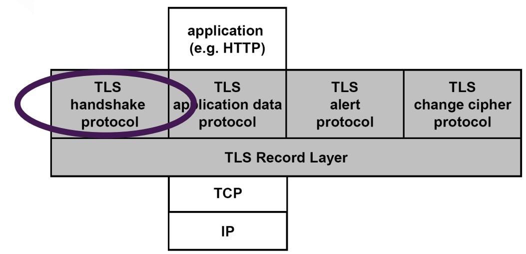


- List of supported cryptographic algorithms
  - Authentication + key exchange + cipher + hash



- Authentication
  - RSA or DSS
- Key Exchange
  - RSA, Diffie Hellman
- Encryption
- Hash

Cipher Suite Examples

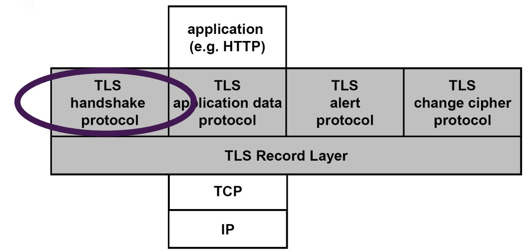



```
CipherSuite TLS_DH_DSS_WITH_DES_CBC_SHA = { 0x00,0x0C };

CipherSuite TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA = { 0x00,0x0D };

CipherSuite TLS_DH_RSA_WITH_DES_CBC_SHA = { 0x00,0x0F };

CipherSuite TLS                                        10};

CipherSuite TLS                                        ;

CipherSuite TLS                                        x13};

CipherSuite TLS                                        ;

CipherSuite TLS                                        x16};
```

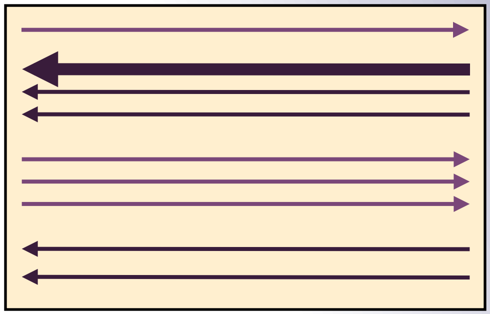
DH: Diffie-Hellman
DSS: Digital Signature Standard
DES: Data Encryption Standard
CBC: Cipher Block Chaining
SHA: Secure Hash Algorithm
EDE: Encrypt-Decrypt-Encrypt

Source: RFC4346.

Server_Hello Content

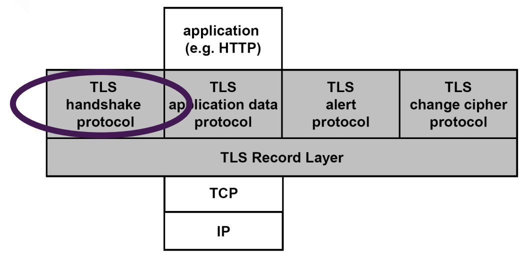


- Goal
  - Used by the server to reply to Client_Hello
  - Sent in the clear without signature



- Content
  - Protocol version: highest version of the protocol supported by both client and server
  - Random number
  - Optional Session Identifier, if it allows sessions to be resumed
  - Cipher Suite: One of the cipher suites proposed by client
  - Compression Method

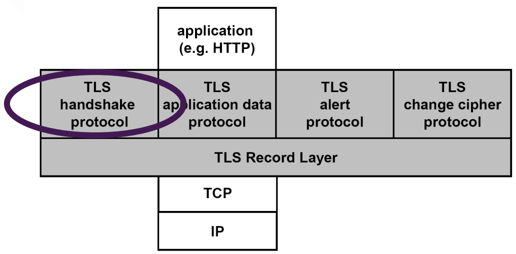
Server Certificate



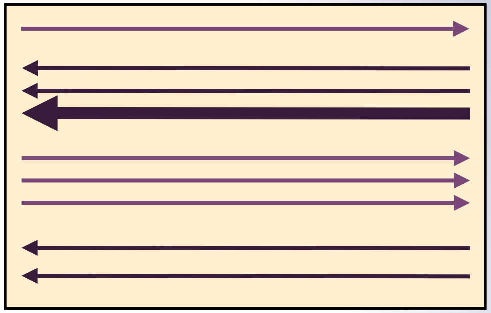
- Goal
  - Sent by the server to authenticate itself
  - A server may have several certificates from different certification authorities

- Content
  - A list of X.509 certificates:
    - Server certificate
    - Certificates of certification authorities
- Certificate can also be sent by the client when client authentication is requested by the server with Certificate_Request

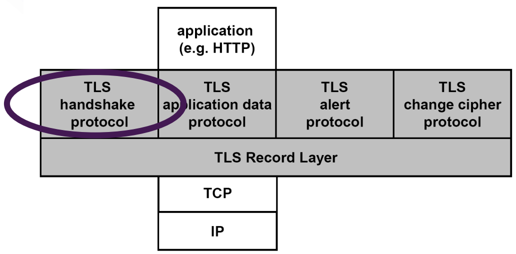
Server_Hello_Done



- Goal
  - Indicates that server has finished its handshake first phase
  - Sent unencrypted

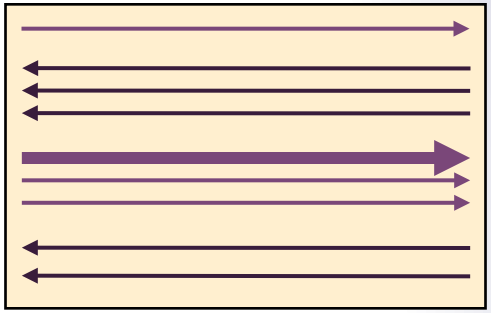


Client_Key_Exchange

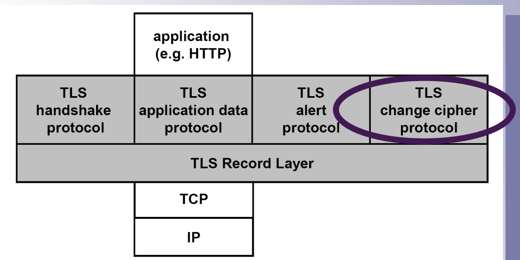


- Goal
  - Used by the client to send the PreMasterSecret, which is used to derive session keys
  - Encrypted with the server's public key

- Content
  - Encrypted PreMasterSecret with the public key of the server

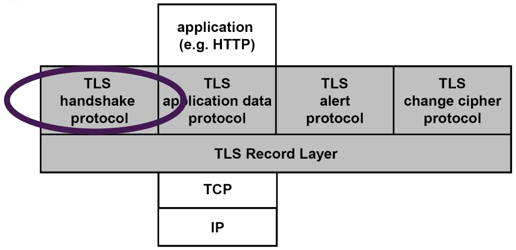


Change_Cipher_Spec



- Goal
  - Used by client and server to indicate that they start using a (new) key
  - During handshake, indicates that next message will be encrypted with the appropriate key

Handshake_Finished



- Goal
  - Sent by both client and server to confirm the establishment of the secure SSL session
    - Session is established only if client received expected Finished message from server and vice-versa
  - Allows to detect man in the middle attacks on Client_Hello and Server_Hello messages
    - Example: Attacker changes cipher list to propose weaker ciphers
  - First encrypted message on each direction

- Contents
  - Keyed hash (MD5 or SHA-1) of all the handshake messages and the MasterSecret

Any questions?

Stay tuned

Next time you will learn about

Passwords | Time-memory trade-offs