Lecture:

Transport Layer Security (secure Socket Layer)

Recommended reading:
Stephen Thomas, "SSL and TLS essentials", Wiley, 2000
Very old and in some parts obsolete, but very well written

Lecture's twofold goal

- **Dissection of well known and widely employed security protocol**
  - Rasonable in-depth analysis of TLS details
    - Devil is in details ☺
    - Although many more details had to be skipped (this is not a full course on TLS…)

- **Understand how a long-to-live security protocol should be designed**
  - Show good design choices vs bad ones
    - TLS shows several examples for both!

Introduction to TLS

History of SSL/TLS

transport layer Security = HTTPS

SSL = TLS
- SSL: proprietario
- TLS: standard

SSL v1
by Netscape
never released
1994

SSL v2
Integrated in
netscape 1.1
Badly broken!
1995

SSL v3
Redesigned
from scratch
by Netscape
1996

. . .

separare protocollo da crypto algoritmo

TLS v1.0
First IETF design
(versus Netscape)
1996-1999
RFC 2246, jan 1999
TLS1.0=SSLv3.1

TLS v 1.1
RFC 4346
Apr 2006

DTLS
RFC 4347
Apr 2006
UDP support
(progettato per TCP)
DATAGRAM

TLS v 1.2
RFC 5246
Aug 2008
Get rid of weak
MD5/SHA-1hash
(negotiated PRF,
default SHA-256)

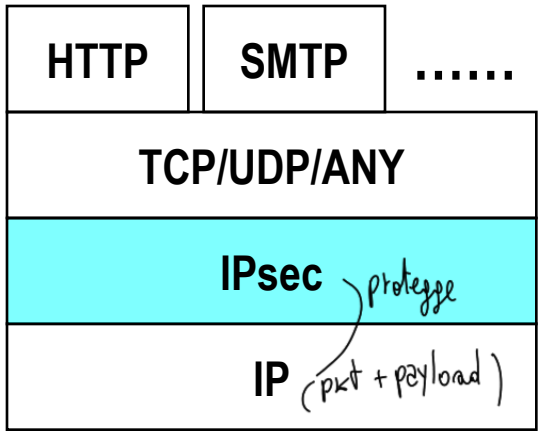
TLS v 1.3
RFC 8446
Aug 2018
Three-way handshake!
Only AEAD ciphers
Major differences
(a «new» protocol?!)
rimuove errori di TLS (compress → encrypt)

. . .

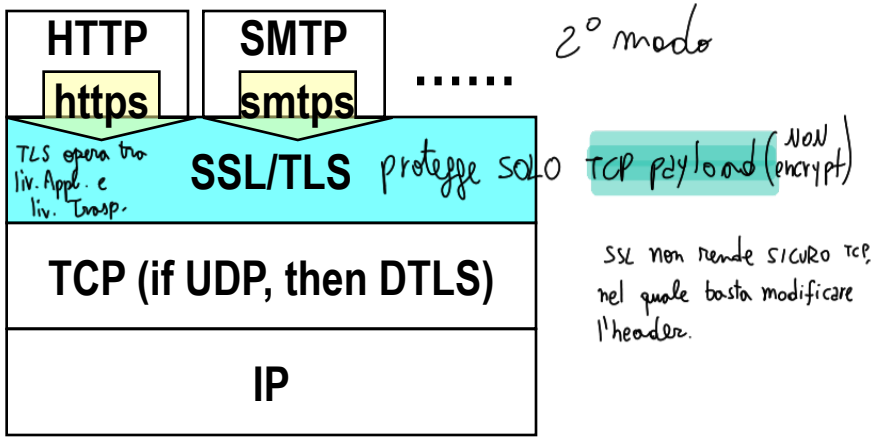
Public Domain implementation
available @ www.openssl.org

come incapsulo pkt?

SSL/TLS: layered view



Network layer security



Transport layer security

➔ **SSL/TLS: on top of TCP, but below application layer**

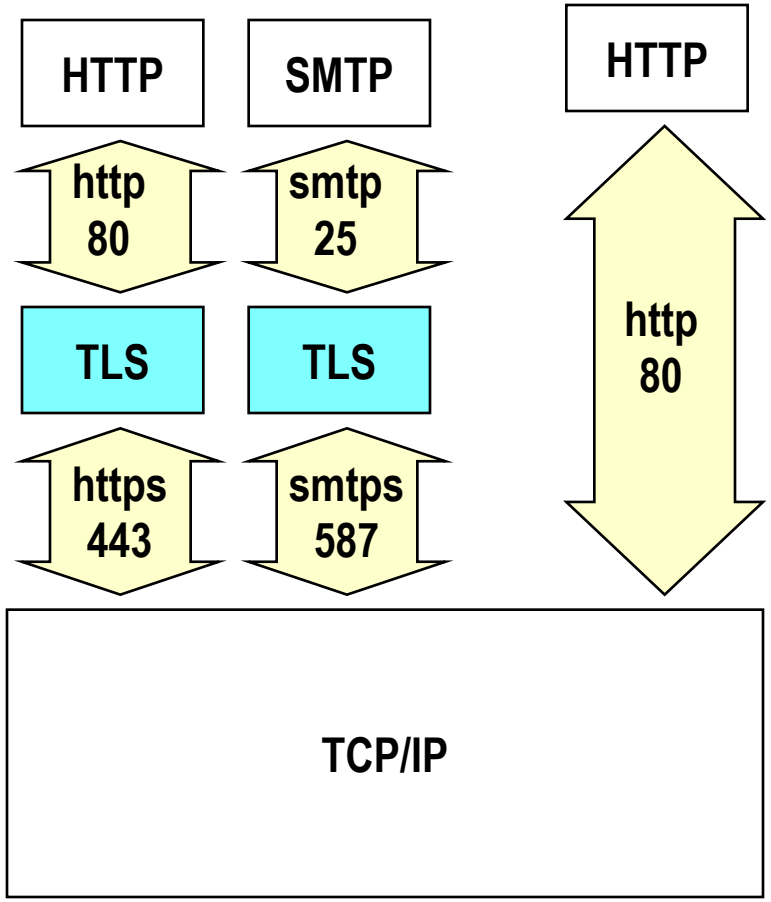
- ⇨ (can be considered as top sublayer for L4 or bottom for L7)
- ⇨ SSL/TLS: NOT a security enhancement of TCP!

➔ **Not necessarily limited to Internet transport (L4)!**

- ⇨ Devised for point-to-point relationships in general
- ⇨ E.g. EAP-TLS (RFC 2716)
  - →TLS-based authentication and integrity protection over L2 EAP

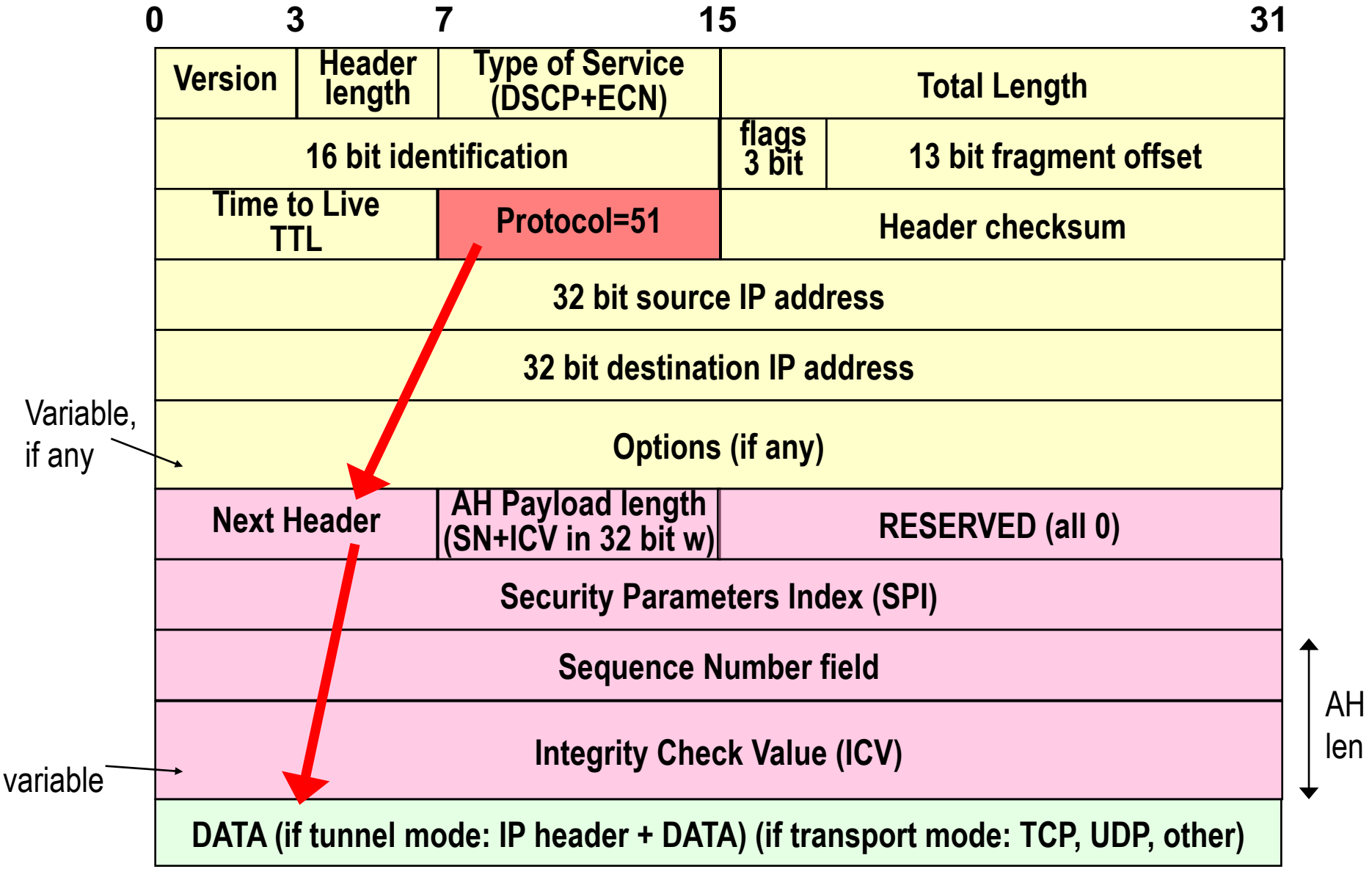
L'applicazione che uso è descritta nella porta cifrata, non so quale uso, parlo di Traffic Flow Confidentiality in IPsec, in SSL posso capirlo dalla porta.

Application support



- **Bad historical idea: reserve special port number for HTTP over SSL/TLS**
  - HTTP=80, HTTPS=443
- **But what if TLS used for other applications? Special port # here as well!**
  - smtps 465 (MS) or 587 (others)
  - spop3 995
  - imaps 991
  - telnets 992
  - …
  - Pros
    - works well; de facto standard
    - Straightforward application support!!
  - Cons:
    - 2 reserved port numbers for same service
    - deprecated by IETF (but still here…)
- **Alternative approach: slightly adapt application's internals**
  - App reuses same port number
  - Example: HTTPv1.1: upgrade: TLS/1.0 new http command (see RFC 2817)

Compare with IPsec (e.g. AH)…



TLS Goals

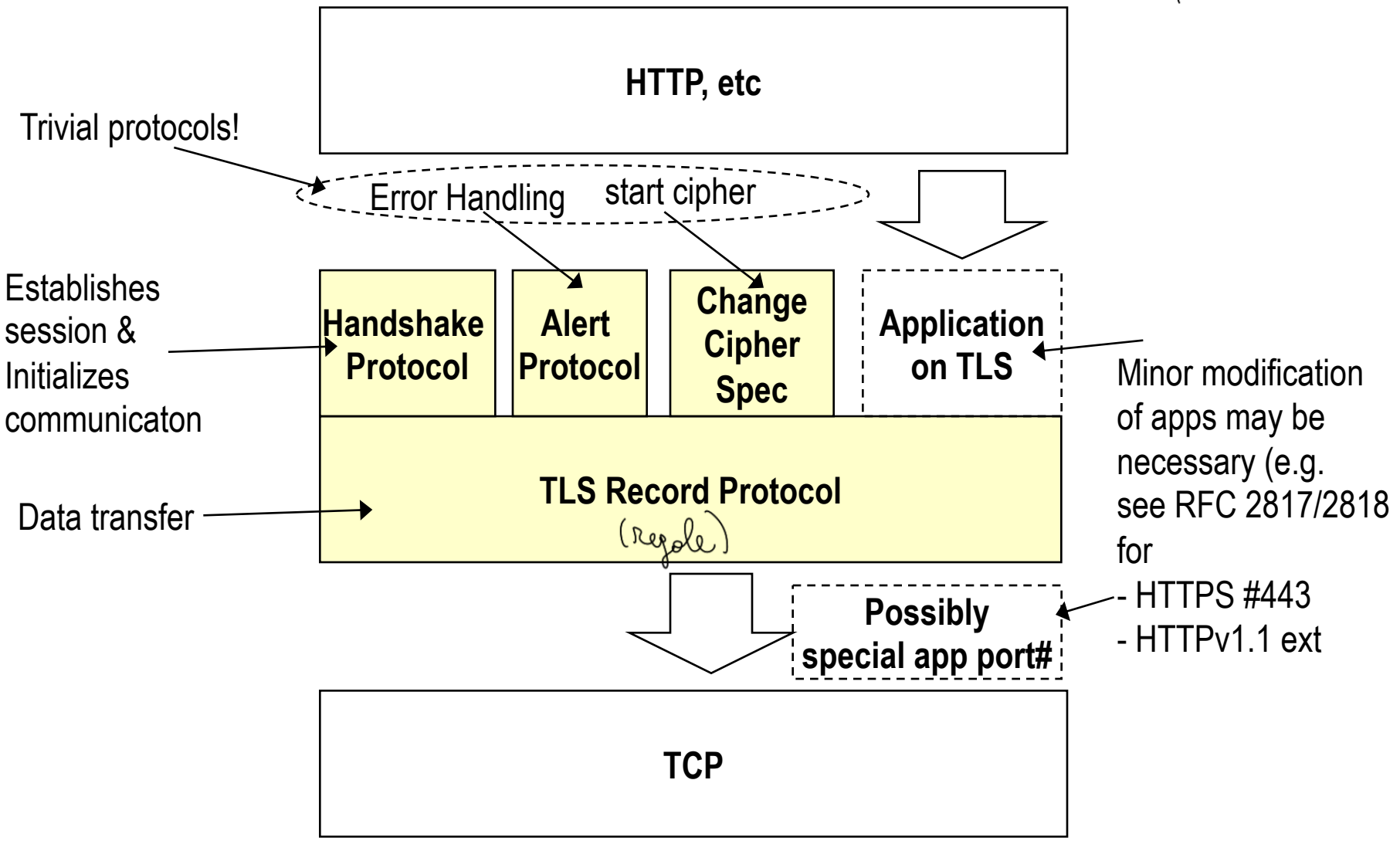
1) ➔ **Establish a session (TLS Handshake phase)**
   - ⇨ Agree on algorithms
   - ⇨ Share secrets
   - ⇨ Perform authentication

2) ➔ **Transfer application data**
   - ⇨ Communication privacy
     - → Symmetric encryption
   - ⇨ Data integrity
     - → Keyed Message Authentication Code (HMAC)

➔ **TLS approach: two-in-one**
   - ⇨ Other Internet security protocols may clearly distinguish the protocol for establishing a session (e.g., IPsec IKE) from the protocol that delivers data and enforces security services (e.g., IPsec ESP/AH)

TLS protocol stack

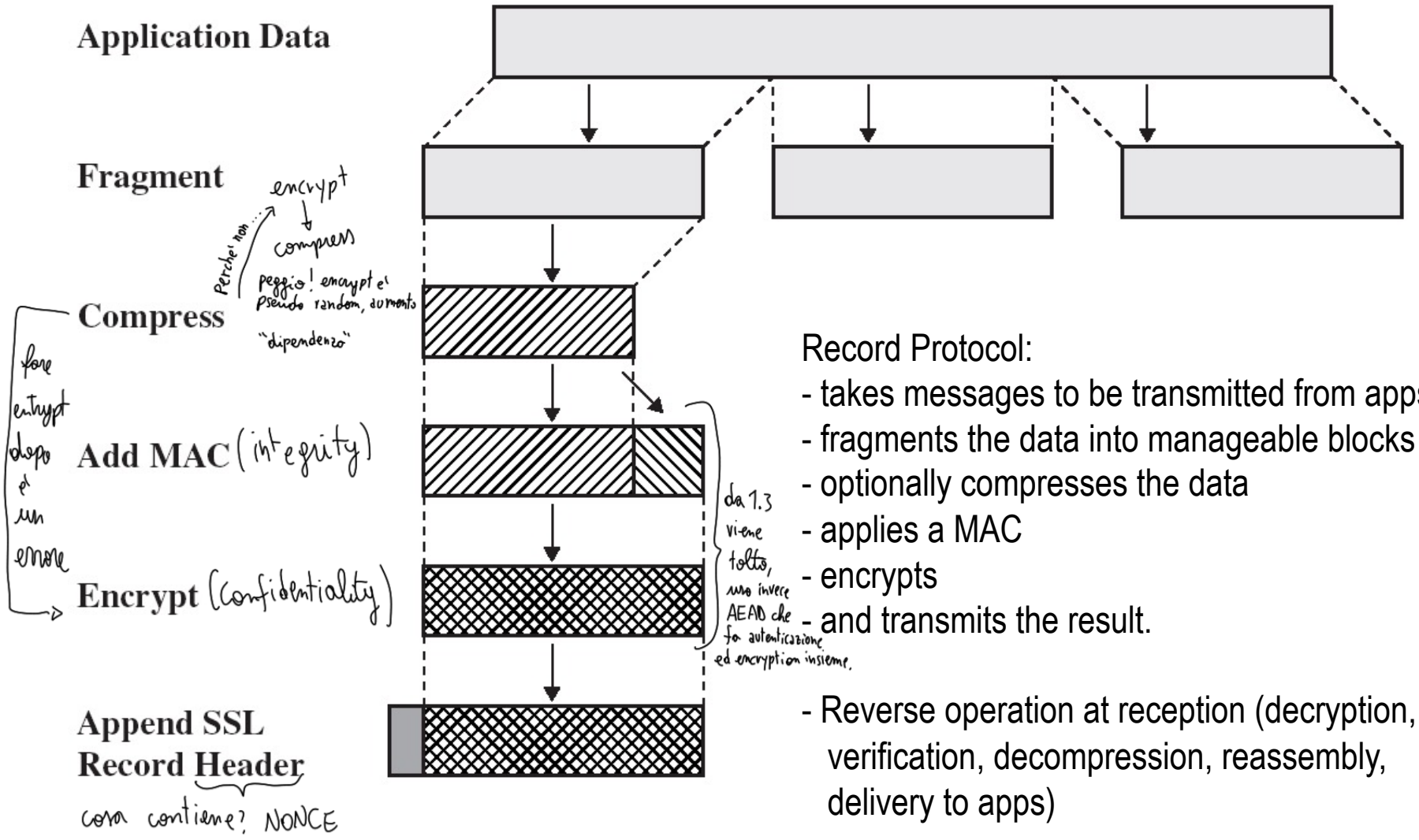
Skip



TLS Record Protocol
(up to TLSv1.2 – v1.3 differs!)

(cosa TLS fa)

Record Protocol operation



Record Protocol:

- takes messages to be transmitted from apps
- fragments the data into manageable blocks
- optionally compresses the data
- applies a MAC
- encrypts
- and transmits the result.

- Reverse operation at reception (decryption, verification, decompression, reassembly, delivery to apps)

Fragmentation

➔At application ←→TLS interface

⇨DON'T get confused with TCP segmentation!!

➔Input: block message of arbitrary size

⇨possibly multiple aggregated messages of SAME protocol

➔Fragment size: 2^{14}=16384 bytes

Compression

➔ **Lossless compression** (banda lenta al tempo)
- ⇨ Formerly used in SSLv2

➔ **Considered in TLSv1.0 but not specified**
- ⇨ «null» compression method in SSLv3 and TLSv1.0

➔ **2004, RFC 3749: introduces DEFLATE (RFC 2951)**
- ⇨ RFC 3943 – support for Lempel-Ziv-Stac

➔ **More recently: considered appealing**
- ⇨ Widespread diffusion of "verbose" languages – e.g. XML
- ⇨ @ early 2012:
  - →45% of the browsers (including Chrome & Firefox)
  - →42% of the servers

➔ **Suddenly died ☺ in september 2012** , compressione rimossa.
- ⇨ **CRIME attack, more later**

Message Authentication Code

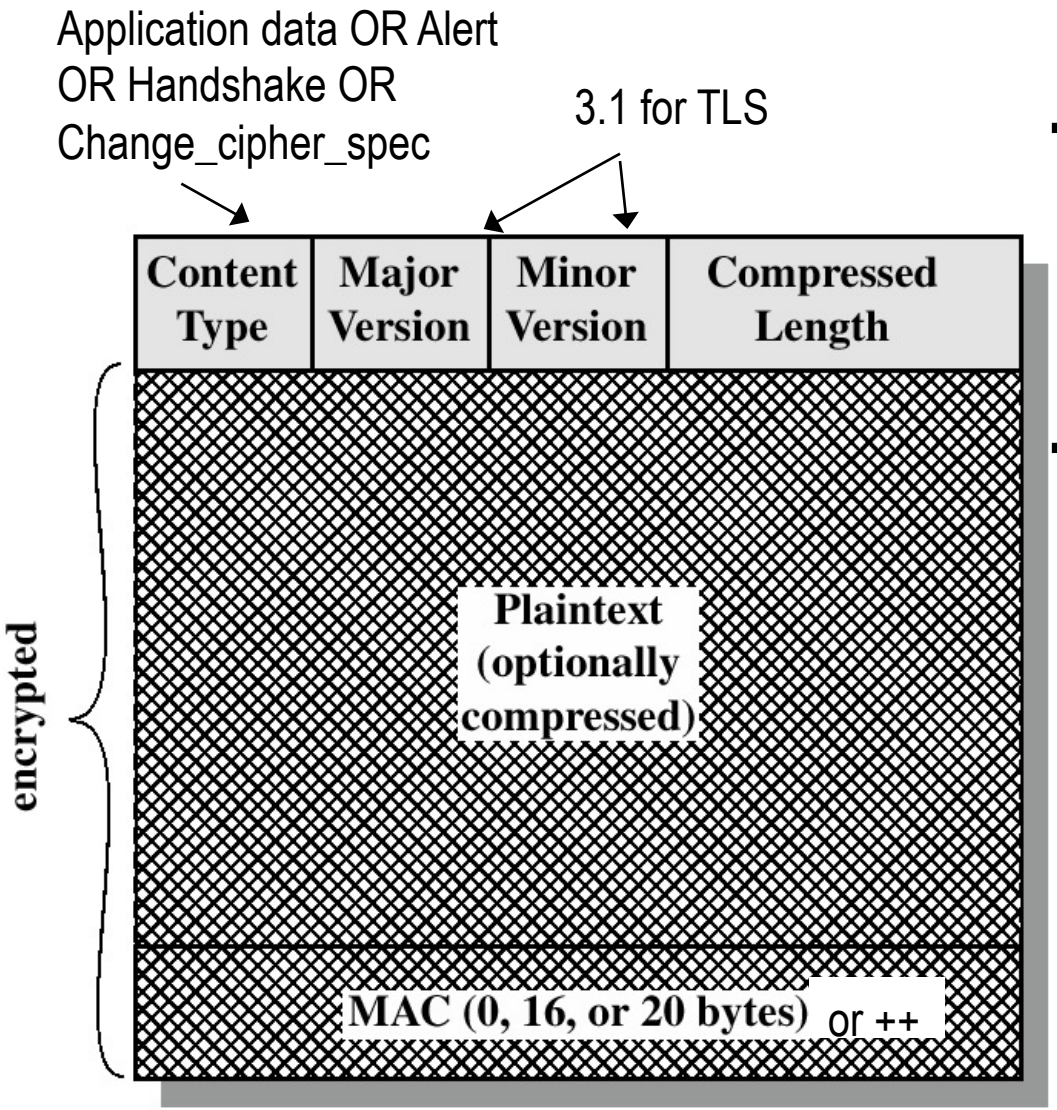
➔MAC computation

- ⇨Secret (symmetric) key derived from security parameters negotiated during handshake
- ⇨Hash function: negotiated during handshake
- ⇨Computation: uses HMAC construction ~> e il reply ?
  - ➔keyed-hashing Message Authentication, RFC 2104

Encryption

- **Fragment Encryption**
  - Applies to both (compressed) fragment and MAC
  - No encryption possible
    - If no encryption negotiated
    - Or in early handshake phases
- **Symmetric encryption algorithm**
  - Block or stream cipher
    - Algorithm (RC4, 3DES, AES, etc) negotiated during handshake, too
    - If block cipher, padding necessary to achieve block size
    - Secret key derived from security parameters negotiated during handshake
    - Differs from key used in MAC
  - Encryption algorithm CANNOT increase size of more than 1024 bytes

Record Protocol Data Unit



➔ **5 bytes header**

- ⇨ Content type = 1
- ⇨ Version = 1+1
- ⇨ Length = 2

➔ **Content Type**

- ⇨ higher layer protocol
  - →20 = 0x14 = Change Cipher Spec
  - →21 = 0x15 = Alert
  - →22 = 0x16 = Handshake
  - →23 = 0x17 = Application Data

e il nonce?

What about replay?

I really want to pay 2000$

Pay 1000 $	HMAC

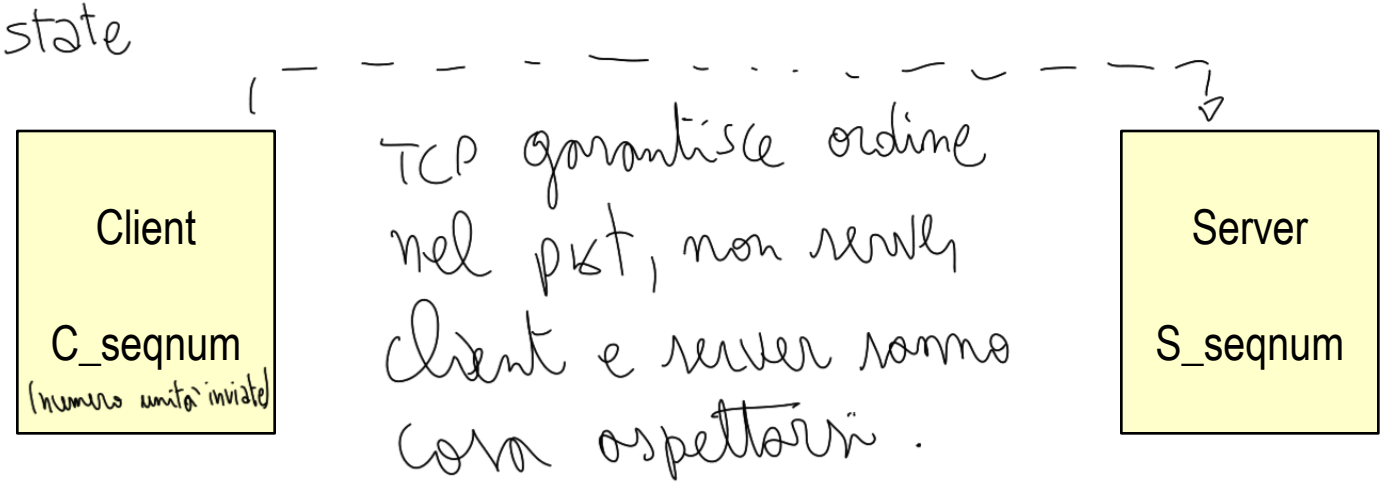
I want to pay 1000$

Pay 1000 $	HMAC

Pay 1000 $	HMAC

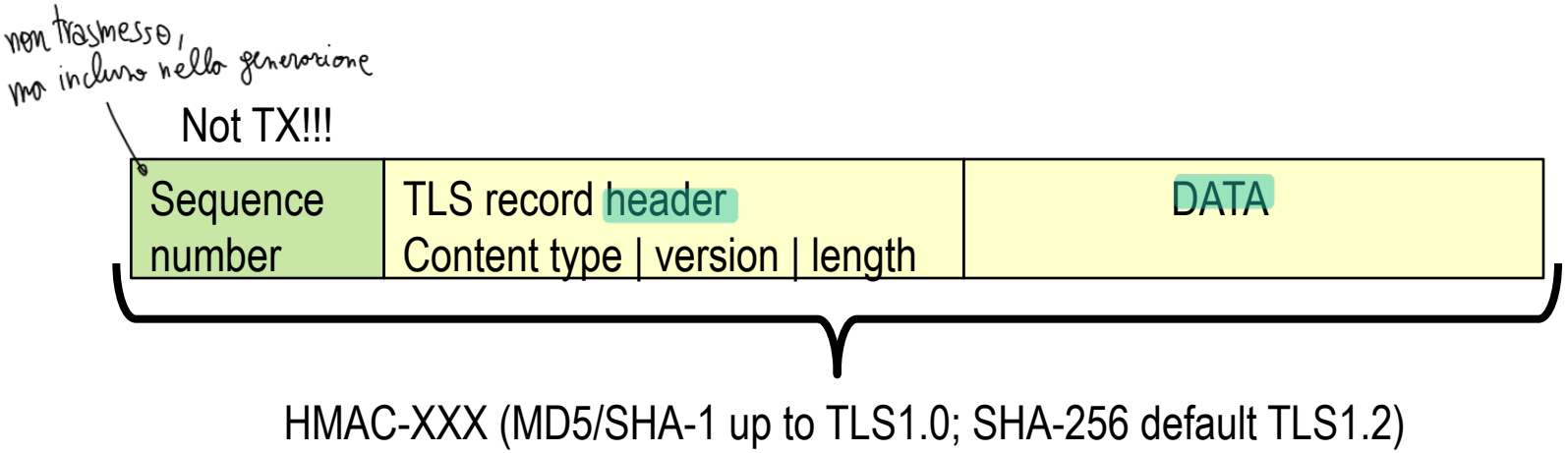
But I'll make you pay twice!

Sequence numbers



- **kept at both connection extremes**
  - Distinct per-direction
  - Initialized to 0; up to 2^{64}-1; do NOT wrap
- **Not transmitted**
  - Remember: reliable transport, hence no "holes"
  - Aftermath: NO WAY TO AVOID TCP!
    - Explicit 2+6=8 bytes sequence number added in DTLS

MAC generation details



➔ **Negotiation may decide not to use MAC**

⇨ In practice, always present

➔ **Sequence number:**

⇨ Not transmitted but included in the MAC

→ to detect missing/extra data

→ and to prevent replay/reordering attacks