TLS: ПРЕДЫИСТОРИЯ

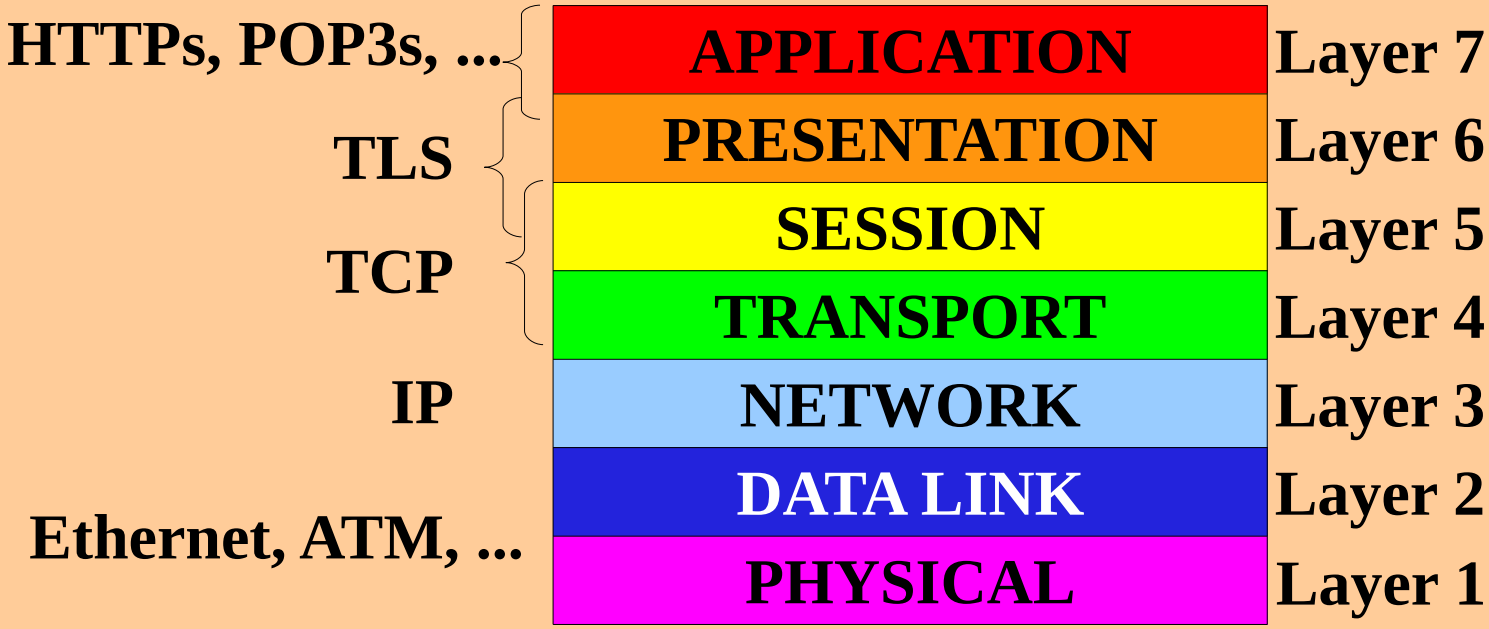
Secure Sockets Layer (SSL) – протокол защищённых сокетов первоначально разрабатывался Netscape. Версия 2.0 была опубликована в 1994 г. Однако из-за обнаруженных уязвимостей в 1996 г. была разработана модификация SSLv3. Эта версия была взята за основу IETF для разработки свободного протокола **Transport Layer Security (TLS)** – протокол безопасности транспортного уровня.

В TLS для идентификации версий используются следующие значения:

«3.0» = SSLv3 — 1996 г.
«3.1» = TLS v.1.0 — RFC 2246 — 1999 г.
«3.2» = TLS v.1.1 — RFC 4346 — 2006 г.
«3.3» = TLS v.1.2 — RFC 5246 — 2008 г.
«3.3»* = TLS v.1.3 — RFC 8446 — август, 2018 г.

Основное назначение: аутентификация сервера (+клиента) и защищённая передача данных (предотвращение подслушивания, подмены данных при передаче, контроль целостности данных) на основе криптоалгоритмов.

TLS vs. OSI/RM



TLS прозрачен для приложения, однако...

Как приложение инициализирует процедуру квитирования?

Как приложение интерпретирует полученные для аутентификации сертификаты?

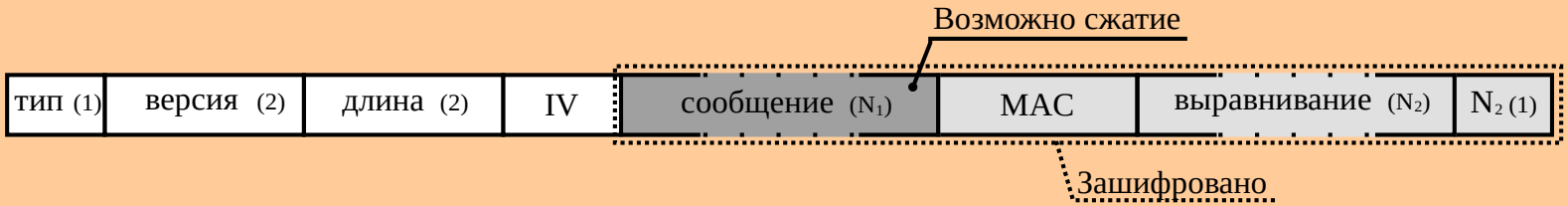
Для разработчиков: OpenSSL, GnuTLS

TLS: ПРОТОКОЛ ФОРМАТА

Формат TLS-записи:

(а) незашифрованной или для поточного шифра

(б) для блочного шифра



Разбиение потока данных (прикладного протокола) на фрагменты (<16Кб), границы сообщений могут не соблюдаться.

Типы: 0x14 (ChangeCipherSpec), 0x15 (Alert), 0x16 (Handshake), 0x17 (Application)

IV — случайное число

$MAC = \text{hmac}(K_{MAC}, seqno||type||ver||len||IV||payload)$

$\text{len}(payload||MAC||padding||N_2)$ кратна блоку шифра (N_2 подбирается)

TLS: ПРОТОКОЛ КВИТИРОВАНИЯ

Cipher Suite

Стороны должны согласовать *криптографический комплект*:

* протокол (TLS или SSL);
* алгоритм обмена ключами и проверки аутентичности (NULL, RSA, DHE_RSA, DH_anon, …);
* алгоритм шифрования, размер ключа, режим шифрования (NULL, RC4_128, AES_128_GCM, CAMELLIA_256_CBC, …);
* алгоритм хэширования для имитовставки и PRF (NULL, MD5, SHA, SHA256, …).

Примеры:

TLS_NULL_WITH_NULL_NULL (0x0000)
TLS_RSA_WITH_RC4_128_MD5 (0x0004)
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
TLS_DH_anon_WITH_AES_256_CBC_SHA256 (0x006D)

*Запрещены в TLS 1.3

http://www.iana.org/assignments/tls-parameters

TLS: ПРОТОКОЛ КВИТИРОВАНИЯ

Pseudo-Random Function (PRF) в TLS 1.2

Генерация ключей

$seed$ = "key expansion"||RND_s||RND_c

$A_0 = seed$, $A_1 = \text{hmac}(MS, A_0)$, …, $A_i = \text{hmac}(MS, A_{i-1})$

$\text{PRF}(MS, seed) = \text{hmac}(MS, A_1||seed)||\text{hmac}(MS, A_2||seed)||...$

Результат PRF делится на 4 сеансовых ключа: K^c_{MAC}, K^s_{MAC}, K^c_{CIPH}, K^s_{CIPH}

$MS = \text{PRF}(PMS, seed)$, где $seed$ = "master secret"||RND_c||RND_s

длина MS — 48 байт

PMS — pre-master secret из результата процедуры квитирования

В TLS 1.3 изменена схема расчёта ключей – на основе HKDF (RFC 5869)

TLS: ПРОТОКОЛ КВИТИРОВАНИЯ

Процедура квитирования (полная) в TLS 1.2

C → S: ClientHello (*ver*, *ts*, RND_c, *sess_id*, *cs_list*, *cmp_list*, *ext*)

S → C: ServerHello (*ver*, *ts*, RND_s, *sess_id*, *cs*, *cmp*, *ext*)

S → C: Certificate* (X.509) — необязательно для DH_anon_*

S → C: ServerKeyExchange* (*p*, *g*, *A* для DHE, ЭЦП(K^s_{pri}, *msg*))

— необязат. для DH_RSA, DH_DSS (p, g, A уже в сертификате) и для RSA_*

S → C: CertificateRequest* — необязательно, если клиент анонимный

S → C: ServerHelloDone

C → S: Certificate* (X.509) — необязательно, если клиент анонимный

C → S: ClientKeyExchange (RSA(K^s_{pub}, *PMS*) или *B* для DH или пусто)

C → S: CertificateVerify* (ЭЦП(K^c_{pri}, *prev_msg*)) — необяз. для анон. клиента

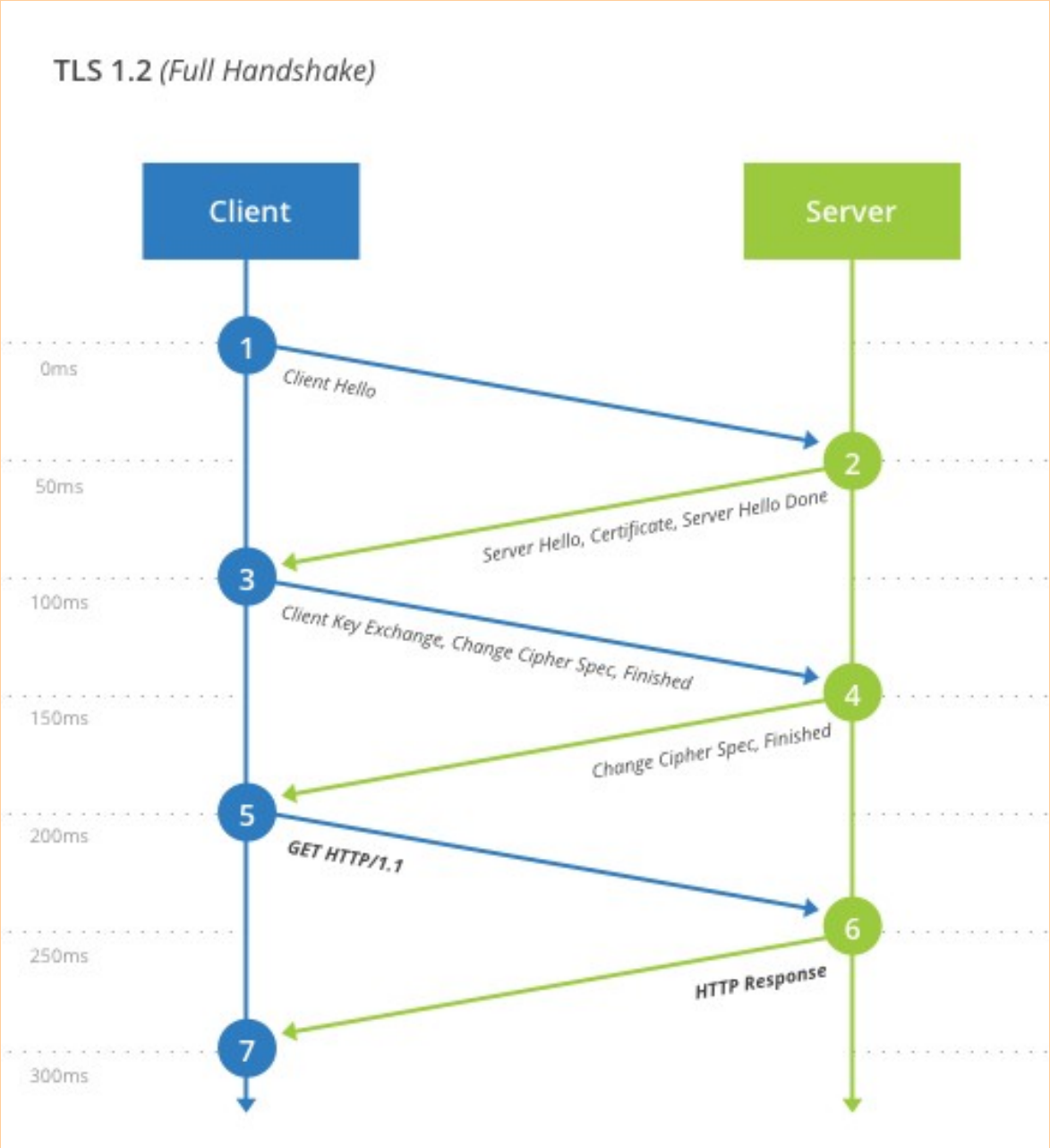
C → S: ChangeCipherSpec

C → S: $Finished_ш$ (PRF(*MS*, "client finished"||hash(*handshake_messages*)))

S → C: ChangeCipherSpec

S → C: $Finished_ш$ (PRF(*MS*, "server finished"||hash(*handshake_messages*)))

TLS: ПРОТОКОЛ КВИТИРОВАНИЯ



TLS: ПРОТОКОЛ КВИТИРОВАНИЯ

Процедура квитирования (сокращённая) в TLS 1.2

C → S: ClientHello (…, *sess_id*, ...)
S → C: ServerHello (…, *sess_id*, ...)
C → S: ChangeCipherSpec
C → S: $\text{Finished}_{ш}$
S → C: ChangeCipherSpec
S → C: $\text{Finished}_{ш}$

Каждому TLS-сеансу назначается идентификатор (sess_id).
Возможно возобновление TLS-сеанса после разрыва соединения транспортного уровня (TCP).

Если сервер разрешает возобновление сеанса, то *sess_id* в ServerHello совпадёт с указанным клиентом в ClientHello.

TLS: ПРОТОКОЛ КВИТИРОВАНИЯ

Процедура квитирования в TLS 1.3

Режим 0-RTT:

Данные приложения включаются в ClientHello и ServerHello как расширение EarlyData, для шифрования используется ключ предыдущего сеанса или PSK. Уязвим к replay-атакам (необходима защита на уровне приложения).