Protokol TLS (Transport Layer Security)

- cíl: po obousměrném proudovém spojení (třeba TCP) poskytovat bezpečné proudové spojení

HTTPS = HTTP nad TLS
viz též DTLS (Datagram TLS)

- evoluce: SSL 1 → SSL 2 → SSL 3 → TLS 1.0 → TLS 1.1 → TLS 1.2 → TLS 1.3

SSL 1: Secure Sockets Layer firmy Netscape (verze 1 nepublikována)
SSL 2: první HTTPS
TLS 1.1: obsolétní a děrové
TLS 1.2: ještě běžné (lze používat bezpečně)
TLS 1.3: aktuální

TLS 1.3 [RFC 8446]

Kombinuje:
1. výměnu klíčů – typicky (EC)DHE
2. autentikaci stran – typicky RSA podpis + veřejný klíč + certifikát
3. šifrování dat – šifra v režimu AEAD (tedy šifra a MAC v jednom)
   ↳ třeba AES-GCM nebo ChaCha20 + Poly1305
4. vyjednávání o parametrech
   - verze protokolu
   - 1 grupy pro DHE
   - 2 autentikační mechanismus včetně nešování se
   - 3 cipher suite

Základem je Record Protocol
- přenáší zprávy protokolů vyšší vrstvy – zprávy umí šifrovat – na počátku se nešifruje
  – pak dočasným klíčem
  – nakonec finálním klíčem

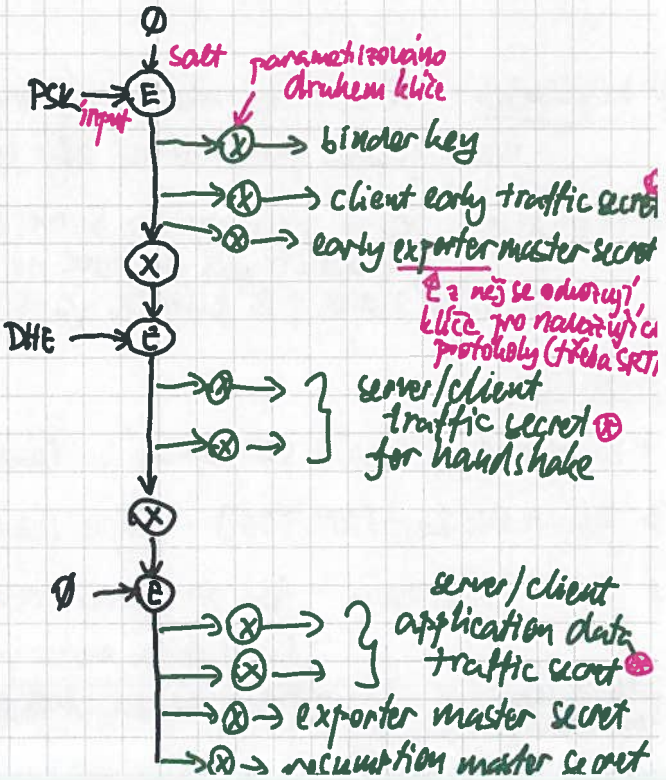
handshake (inicializace) → zprávy obsahují pevné položky + extensions
application data → proud dat libovolně rozsekaný na zprávy
alert (hlášení chyb) a EOF

přidává padding, lze padovat víc, než je nutné] zaručí délku plaintextu

- rozvrh klíčů

odvozený z HKDF [RFC 5869]
HMAC-based extract & expand Key Derivation Function
cíl: původní data "pomíchat" (extract) → (E)
a pak z nich vyrobit klíče žádané velikosti (Expand) → (X)

⊛ traffic secret: z něj se odvozuje klíč a IV pro cipher suite
↳ časem se přepočítá, abychom 1 klíč nepoužívali moc dlouho
↳ z tě a seq. no. se odvozuje nonce pro AEAD



• <u>Handshake Protocol</u> (1-RTT handshake)

ClientHello →
- key share
- nabízené množiny parametrů (grupy pro DHE apod. se std. číselníka)
- nabízený PSK exchange modes (PSK or PSK+DHE)
- nabízené PSK (jejich identity)
- early data

← ServerHello
- key share
- vybrané hodnoty parametrů
- vybraný PSK (identita) + exch. mode

Šifrováno handshake klíči:
- Encrypted extensions
- [Certificate request – pokud se má klient autentikovat]
- [Certificate (serveru)
- Cert Verify – transcript podepsaný soukr. klíčem k certifikátu]
- Finished – MAC transcriptu

[application data]

Šifr. HS klíči:
- [Certificate
- Cert Verify] →
- Finished

application data →

← app. data

• <u>Pre-Shared Keys</u> – lze používat s DHE i bez
- klient pošle seznam identit PSK + pro každou <u>binder</u> – MAC transcriptu pomocí binder key z rozvrhu klíčů pro daný PSK

↓

používají se také pro session resume:
- server pošle NewSessionTicket (kdykoli, může i víckrát) s noncí a identitou (do té zakódován stav spojení)
- oba z nonce a resumption master secretu spočítají klíč

} při založení dalšího spojení lze použít jako PSK (na 1 použití)
↓
ale server mi může dát víc klíčů, abych mohl založit paralelní spojení

• <u>KeyUpdate</u>: pošlu, pokud měním svůj traffic key na další v pořadí
- mohu požádat protistranu, aby udělala totéž v opačném směru

• <u>HelloRetry</u>: odpoví server místo ServerHello, pokud se mu nelíbí navržené parametry, a navrhne nové
- předání stavu přes klienta (cookie extension)

<u>Rozšíření</u>

• heartbeat – žádá o periodické zasílání "oživovacích" zpráv

• Raw Public Keys [RFC 7250] – podpis bez certifikátu, veřejný klíč validuji jinudy, třeba přes DNSSEC

• 0-RTT handshake – jen při session resume ... v ~~KeyUpdate~~ NewSessionTicket mi server dovolí posílat early data
- early data mohu přibalit k ClientHello
- pozor, nejsou chráněna proti replayování! – používat jen na žádost aplikace
  ↳ např. u HTTP GET bezpečné (nemá side-effekt)

server nemusí kontrolovat recyklaci PSK

- Server Name Indication (SNI) – host name pro servery obsluhující více domén
  - podle něj server typicky volí certifikát
  - pozor, není šifrované!
    - ↳ experimentální rozšíření: Encrypted SNI, Encrypted Hello } klíče bere z DNS
- Application-Level Protocol Id (ALPN) – umožňuje na 1 portu provozovat víc protokolů
  - např. HTTP/1 vs. HTTP/2
- post-handshake auth – lze dodatečně požádat klienta o certifikát

Dohadování na verzi protokolu

Problémy: ① downgradovací útoky
② kostnatění protokolu kvůli zastaralým middleboxům "protocol ossification"

Řešení: ② číslo verze v hlavičce posíláme jako TLS 1.2, skutečné je v extension & další zprávy připomínají v1.2, než začneme šifrovat

① starší verze posílají v obou hello nonce
- pokud server odpovídá starou verzí, ale umí novou, změní 8 z 32 bytů nonce na fixní string ⇒ klient to pozná (útočník to nemůže změnit zpět, neb by nesedly podpisy)

Útoky na starší verze

- useknutí spojení – "cookie cutting attack" – v hlavičce "Cookie" smažu "; secure" na konci, takže klient cookie pošle i po nešifrovaném HTTP
  - na to potřebuji umět vkládat seno do odpovědi serveru
  - trik: server často v redirectech přidává parametry za ? z přív. URL
  - (klienti běžně akceptují odpověď bez konce hned za hlavičkou?)
  - ↳ proto máme Close Alert, ale dodnes ho aplikace běžně ignorují
- Re-negotiation attack
  - staré TLS umí spustit dohadování znovu (třeba kvůli změně klíčů po několika GB dat) nebo dodatečné žádosti o klientův cert
  - ale nepodepisuje návaznost na předchozí stav
    - ↳ útočník naváže spojení se serverem, pošle data, požádá o re-nego a propojí s obětí ⇒ oběť si myslí, že má čerstvé spojení } umožňuje vložit data před klientova
  - TLS 1.2 má rozšíření Secure Re-negotiation
  - TLS 1.3 re-nego úplně ruší → nahrazeno KeyUpdate + post-HS auth.
- BEAST – TLS 1.0 a jeho nešikovné CBC
- CRIME – TLS ≤ 1.2 umělo kompresi, v 1.3 není
- Lucky 13 – bloková šifra s CBC měla padding oracle ⇒ TLS 1.3 podporuje jen AEAD
- POODLE – jiný útok na padding v SSL3
- DROWN, ROBOT – variace na Bleichenbacherův útok na RSA

} viz str. 52–53 starých zápisků

Shrnutí: TLS 1.3 se vyhýbá známým útokům
TLS 1.2 vyžaduje pečlivou konfiguraci + rozšíření, pak je též bezpečné
nic staršího nepoužívat!