TLS part 2

Cryptology (1)

Martin Stanek

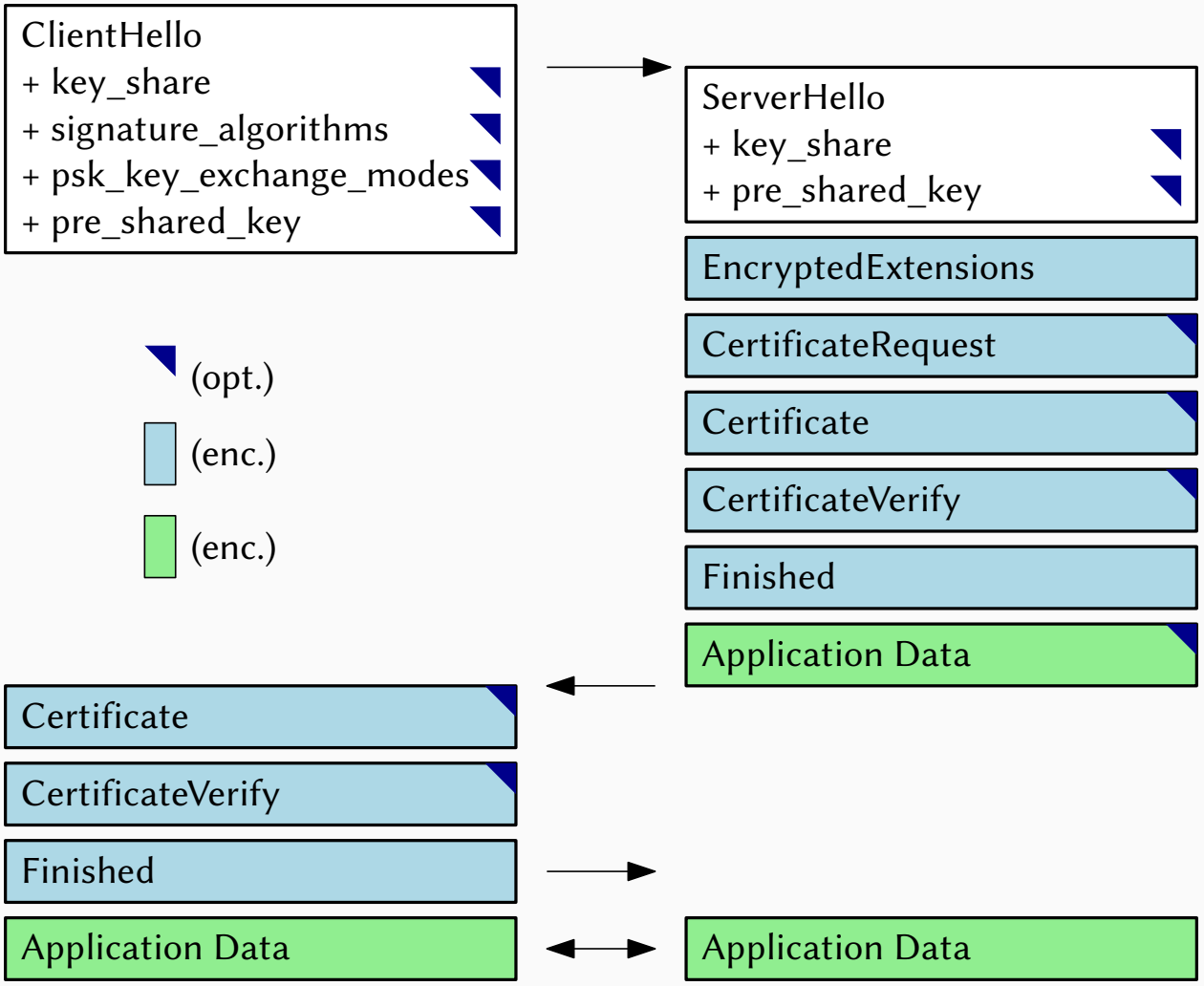
2025

KI FMFI UK Bratislava

TLS 1.3 – major changes from TLS 1.2

- AEAD ciphers only (support for non-AEAD ciphers removed)
- public-key key exchange with forward secrecy (static RSA and Diffie-Hellman removed)
- redesigned key derivation function: HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
- reworked handshake: 1-RTT (1 round trip time) mode
- new zero round-trip time (0-RTT) mode
- other things removed: custom DHE groups, compression support, DSA
- RSA-PSS is used instead of PKCS#1 v1.5 for handshake signatures
- key exchange modes: DH, PSK, PSK + DH

TLS 1.3 – overview



TLS 1.3 – mandatory cipher suites

- symmetric cipher suite (AEAD + hash function HKDF):
  - MUST: TLS_AES_128_GCM_SHA256
  - SHOULD: TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256
- digital signatures (MUST):
  - rsa_pkcs1_sha256 (for certificates)
  - rsa_pss_rsae_sha256 (for CertificateVerify and certificates)
  - ecdsa_secp256r1_sha256
- key exchange:
  - MUST: secp256r1 (NIST P-256)
  - SHOULD: X25519

Forward Secrecy (FS)

- previous session keys are not compromised even if the long term keys are
- desirable property of key agreement/distribution protocols
- TLS 1.2:
  - RSA: obtaining server's RSA private key reveals all previous and future pre-master secrets (and all keys can be recomputed from the pre-master secret)
  - ephemeral non-anonymous DH – DHE, ECDHE (FS)
- TLS 1.3 supports three basic key exchange modes:
  - Diffie-Hellman over the finite fields and or elliptic curves (FS)
  - pre-shared symmetric key (PSK) (not FS)
  - combination of PSK and DH (FS)

TLS 1.3 – Selfie

- Selfie (2019) – first protocol attack on TLS 1.3
  - rarely used case of (external) PSK authentication
  - scenario: client can also be a server
  - simple reflection: attacker resend all messages back to client
  - client establishes connection with itself
  - limited impact in practice

TLS Security

Trust – certificates (PKI)

- trusted CA certificates distributed by browsers/OS
- example: Firefox ≈ 180 CA certificates
- Do you trust them all?
- Certificate validation – chain, expiration, server name, signatures, check revocation, …
  bugs are common
- Users – ignoring warnings/errors

Trust – reality

- (2014-2015) Lenovo Superfish – self-signed CA pre-installed, automatic MITM attack (inserting ads to web pages), private key shared among installations
- (2011) DigiNotar (NL) – compromised since 2009, fake certificates (MITM), removed from the list od trusted CA, bankruptcy
- (2011) Comodo – registration authority account compromised, 9 fake certificates
- (2017-2018) distrust of Symantec CA (and its subordinates: Thawte, GeoTrust, RapidSSL) – business sold to DigiCert
- (2018) Trustico (former reseller for Symantec) – sending 23.000 private keys to DigiCert by e-mail … to revoke the certificates
- Serrano et al. *A complete study of P.K.I. (PKI's Known Incidents)*, 2019

Checking certificates

- checking certificate status: OK or revoked?
- two standard options:
  - CRL (Certificate revocation list) – a list signed by CA, issued frequently (e.g., at least every 24 hours); can be large (e.g., GlobalSign's CRL from 22 kB to 4.7 MB thanks to Heartbleed)
  - OCSP (Online Certificate Status Protocol) – requesting info from CA; response with a timestamp, and signed by CA
- non-standard approach:
  - CRLSet (Chrome), OneCRL (Firefox) – list of selected revoked certificates distributed as an update to the browser (Chrome – selected certificates; Firefox – intermediate certificates)

OCSP stapling

- problems with OCSP:
  - What to do if there is no response from CA – block or allow?
  - user privacy (CA learns what certificates client wants to check)
  - CA flooded with requests related to sites with high traffic.
  - slower user experience.
- idea: server requests OCSP response at regular intervals and adds it as Certificate Status message in the Handshake
  - the response cannot be forged (timestamp, signed by CA)
  - TLS Certificate Status Request Extension
  - Multiple Certificate Status Request Extension
    - providing status for all certificates in a chain
    - original extension: only for server's own certificate
- OCSP Must-staple
  - certificate extension – server must staple, otherwise the certificate is invalid

OCSP reality – Let’s Encrypt

- Let’s Encrypt ended OCSP Support in 2025

  “*We plan to end support for OCSP primarily because it represents a considerable risk to privacy on the Internet.*”

  “*… operating OCSP services has taken up considerable resources …*”

  “*… OCSP Must Staple has failed to get wide browser support after many years …*”

- revocation information provided exclusively via CRLs

Certificate Transparency

- goals:
  - make hard for a CA to issue a certificate that is not visible to domain owner
  - allow to monitor and audit issued certificates (by domain owners, CA, etc.)
  - protect users against certificates issued maliciously or mistakenly
- Certificate Transparency log
  - Merkle tree of certificates (or precertificates) issued by CAs
  - publicly verifiable
  - signed root
- CA publishes certificates (precertificates) to public logs
- SCT – Signed Certificate Timestamp – log's promise to incorporate the certificate in the Merkle tree
  - SCT(s) included in the certificate

HTTP Strict Transport Security (HSTS)

- SSL Stripping
  - attacker: MITM proxy replacing links https with http links
  - user clicks on a link …
  - victim communicates with attacker via http
  - attacker communicates with the web server via https
- HSTS (RFC 6797)
  - HSTS headers over https – instructing browser to use only https for all future requests
  - browser transforms all http links into https links
  - browser does not allow unsecured connections to the web server
- limitation: HSTS header stripped in first visit (pre-loaded list of HSTS sites in browsers – does not scale)
- supported: Firefox, Chrome, Safari

STARTTLS

- Opportunistic TLS, switch from plaintext to TLS connection
- STARTTLS command
  - supported: SMTP, POP3, IMAP, LDAP, etc.
- STRIPTLS attack – removing STARTTLS

History of some SSL/TLS problems (1)

- Apple “goto” fail (2014) – not verifying server’s signature
- Heartbleed (2014) – OpenSSL bug, reading server’s memory
- PKCS #1 v1.5 padding RSA (Bleichenbacher 1998) – decrypt anything
- timing attacks (2003)
- PRNG
  - initialization problems (Debian, OpenSSL 2006–2008)
  - hardcoded keys (DUHK, 2017)
- renegotiation problem (2009)
  - victim’s handshake as a renegotiation
  - insert arbitrary data as a prefix of victim’s communication

History of some SSL/TLS problems (2)

- BEAST (2011) – CBC mode problem
- CRIME (2012) – compression leaks plaintext information
- POODLE (2014) – padding oracle attack
  - variants: GOLDENDOODLE, Zombie POODLE, Sleeping POODLE, …
- FREAK (2015) – attacking export grade cryptography (512-bit RSA)
- Logjam (2015) – attacking export grade cryptography (short DH groups)
- DROWN (2016) – cross-protocol attack, old SSL version with shared RSA key
- ROBOT (2018) – Return Of Bleichenbacher's Oracle Threat

Exercises

1. *Find all CT logs that include current certificate for domain* `www.uniba.sk`*.*
2. *Find an “*`sk`*” domain that*
   a) *uses HSTS but is not in the HSTS preload list of your browser (check the parameters)*
   b) *uses HSTS and is in the HSTS preload list of your browser*
3. *Find the purpose of Encrypted Client Hello extension of TLS 1.3.*