TLS (Transport Layer Security)

Martin Stanek

Department of Computer Science
Comenius University
stanek@dcs.fmph.uniba.sk

Cryptology 1 (2023/24)

Content

History, goals, and current support

Structure of the TLS

- Handshake Protocol

Security

- Trust, Checking certificates, OCSP stapling
- Certificate pinning, Certificate Transparency
- Using TLS: HSTS, STARTTLS
- History of some SSL/TLS problems

SSL/TLS History

- SSL – Secure Socket Layer
- TLS – Transport Layer Security
- History:
  - 1995 SSL 2.0 (Netscape Communications)
  - 1996 SSL 3.0 (Netscape Communications)
  - 1999 TLS 1.0 (RFC 2246, “SSL 3.1”)
  - 2006 TLS 1.1 (RFC 4346)
  - 2008 TLS 1.2 (RFC 5246), updated by 10 other RFCs
  - 2018 TLS 1.3 (RFC 8446)

Goals of TLS

- According to TLS 1.2 (prioritized):
  1. Cryptographic security – to establish a secure connection between two parties (data confidentiality and integrity/authenticity)
  2. Interoperability
  3. Extensibility – to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary
  4. Relative efficiency – optional session caching scheme, reducing network activity

- According to TLS 1.3:
  - *The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream.*
  - authentication, confidentiality, integrity

Support: browsers and servers

- Browsers – default settings:
  - Chrome (119), Firefox (120): TLS 1.2, 1.3
  - removed/disabled by default – TLS 1.0 and 1.1
- Servers:

	XII/2017	XII/2018	X/2020	XI/2021	XI/2022	XI/2023
sites	150.000	139.000	138.100	135.500	135.600	134.928
TLS 1.0	91.0%	71.3%	51.5%	40.5%	34.4%	29.5%
TLS 1.1	84.9%	79.1%	58.5%	44.3%	37.5%	31.8%
TLS 1.2	89.4%	94.3%	99.0%	99.6%	99.9%	99.9%
TLS 1.3		10.5%	39.8%	50.4%	58.4%	66.2%

SSL Pulse (https://www.ssllabs.com/ssl-pulse/)

TLS applications

- TLS requires a reliable transport protocol (e.g. TCP)
  - see DTLS for using TLS with datagram protocols
  - the most recent DTLS 1.3 based on TLS 1.3:
    equivalent security guarantees with the exception of order protection / non-replayability
- almost transparent to higher level protocols
- various applications:
  - web: HTTPS ~ HTTP + TLS (the most frequently used application)
  - accessing mail: IMAP/POP3 + TLS
  - transferring mail: SMTP + TLS
  - building VPN over TLS

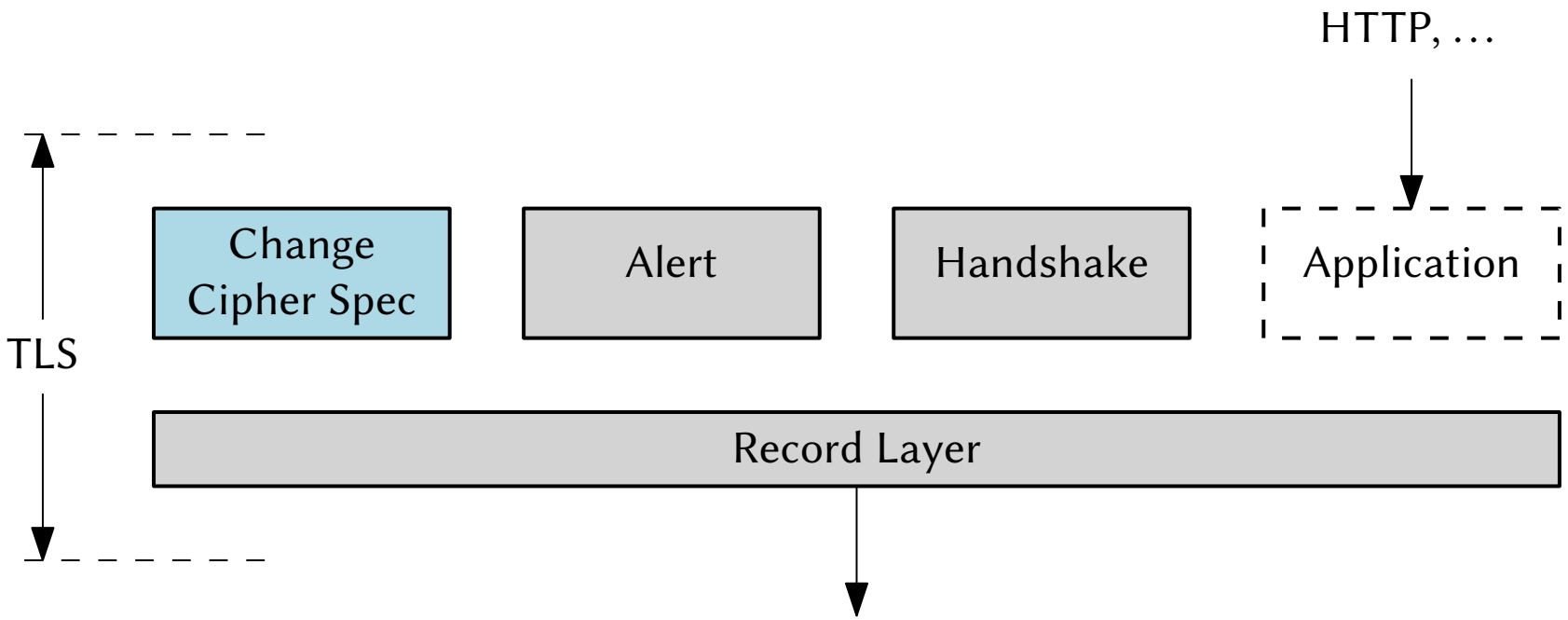
    ...

Limitations of the TLS

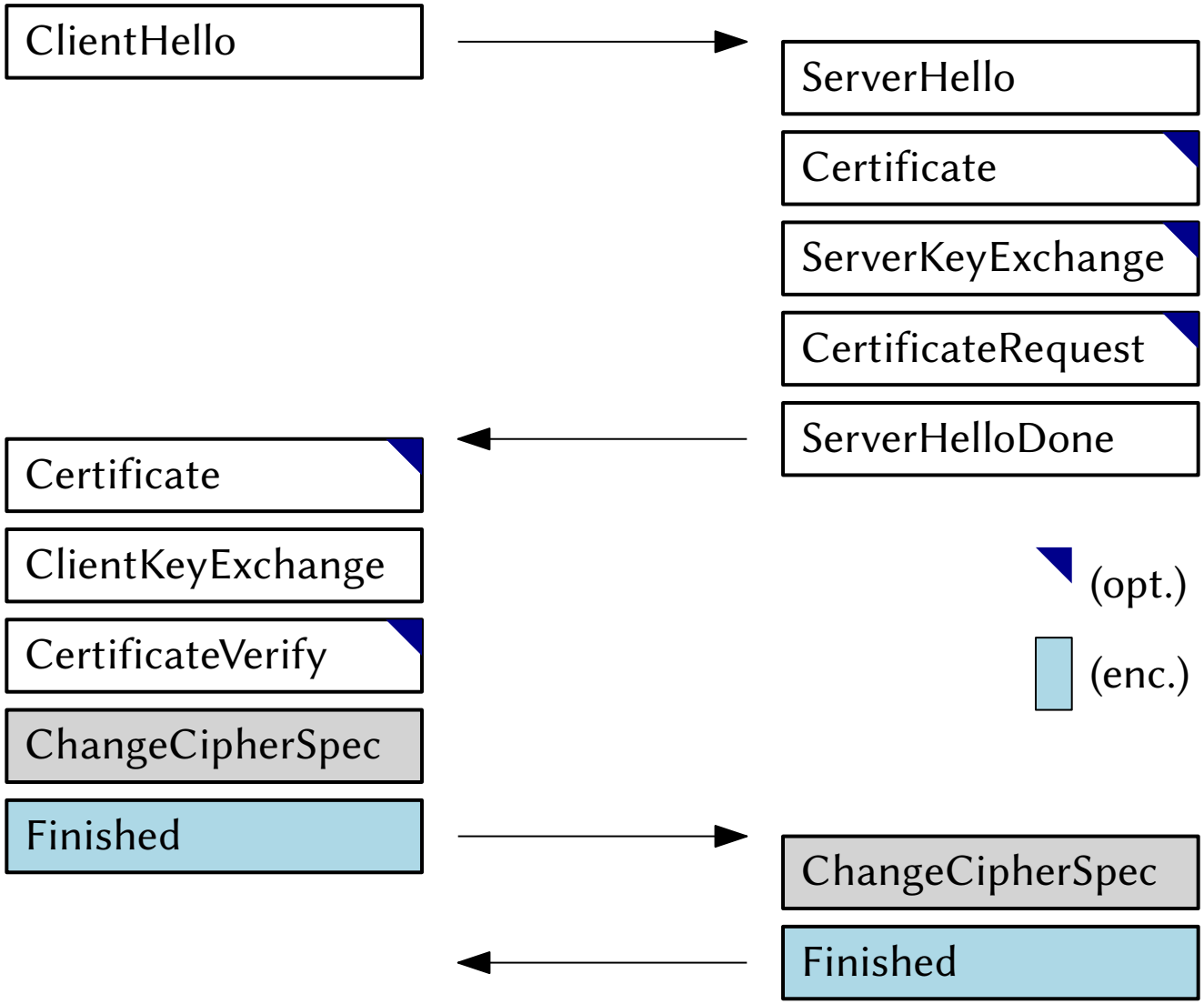
- no data non-repudiation
- depends on PKI
  - certificate management (trust, distribution, revocation, etc.)
- TLS does not provide solution for web application vulnerabilities
  - SQL injection, XSS, CSRF, ...
- TLS does not provide solution for weaknesses on user's side
  - weak passwords, accepting suspicious certificates, ...

Structure of the TLS

- client ↔ server (asymmetric communication)
- two layers, subprotocols
- ChangeCipherSpec – TLS 1.2 only (and middlebox compatibility when needed)



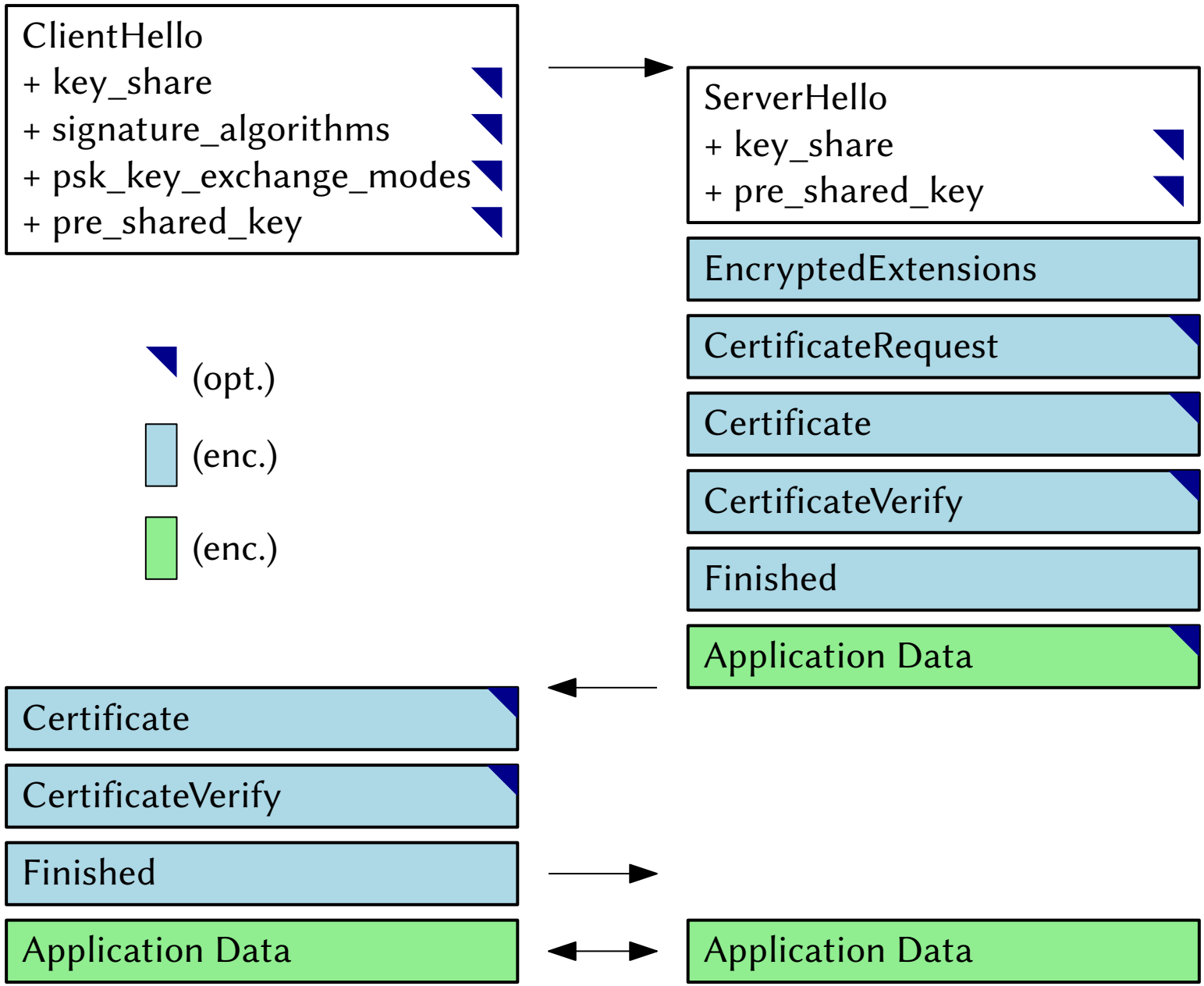
TLS 1.2 Handshake Protocol – overview



TLS 1.2 Handshake Protocol – brief description

1. Exchange hello messages, agree on algorithms, exchange random values (nonces), check for session resumption.
2. Exchange certificates to authenticate server (mandatory) and client (optional).
3. Exchange parameters and values to agree on a pre-master secret.
4. Calculate master secret from the pre-master secret and random values. Calculate necessary keys and other parameters.
5. Switch to agreed algorithms and keys.
6. Verify that the other communication end calculated the same parameters.

TLS 1.3 Handshake Protocol – overview



Basic cryptographic components

- key agreement schemes: DH, RSA (TLS 1.2 only)
- server authentication (certificates), client authentication optional
- symmetric encryption: block/stream ciphers
- authenticating data: AEAD (authenticated encryption with additional data), HMAC (TLS 1.2 only)
- PRF (pseudorandom function)
- PRNG (pseudorandom number generator)
- …

List of supported cipher suites – client-preferred order

User Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:120.0) Gecko/20100101 Firefox/120.0

TLS_AES_128_GCM_SHA256 (0x1301)	Forward Secrecy
TLS_CHACHA20_POLY1305_SHA256 (0x1303)	Forward Secrecy
TLS_AES_256_GCM_SHA384 (0x1302)	Forward Secrecy
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)	Forward Secrecy
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	Forward Secrecy
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)	Forward Secrecy
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)	Forward Secrecy
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)	Forward Secrecy
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	Forward Secrecy
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)	**WEAK**
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)	**WEAK**
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	**WEAK**
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	**WEAK**
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	**WEAK**
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	**WEAK**
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	**WEAK**
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	**WEAK**

List of supported signatures and curves

Firefox/120.0

- signature_algorithms:

  SHA256/ECDSA, SHA384/ECDSA, SHA512/ECDSA, RSA_PSS_SHA256, RSA_PSS_SHA384, RSA_PSS_SHA512, SHA256/RSA, SHA384/RSA, SHA512/RSA, SHA1/ECDSA, SHA1/RSA

  (*) PSS schemes defined in TLS 1.3

- named groups:

  x25519, secp256r1, secp384r1, secp521r1, ffdhe2048, ffdhe3072

List of supported cipher suits and secure TLS configuration

www.uniba.sk (November 2023), server-preferred order for TLS 1.2 (https://www.ssllabs.com/ssltest)

 # TLS 1.2 (suites in server-preferred order)

TLS_RSA_WITH_RC4_128_SHA (0x5) **INSECURE**

TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) **WEAK**

TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) DH 2048 bits FS **WEAK**

TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x45) DH 2048 bits FS **WEAK**

This server uses RC4 with modern protocols. Grade capped to C.

This server does not support Forward Secrecy with the reference browsers. Grade capped to B. MORE INFO »

This server does not support Authenticated encryption (AEAD) cipher suites. Grade capped to B. MORE INFO »

This server supports TLS 1.0 and TLS 1.1. Grade capped to B. MORE INFO »

recommened TLS configurations: https://ssl-config.mozilla.org/

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 – 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), and 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 (all keys can be recomputed from pre-master secret)
  - ephemeral non-anonymous DH – DHE, ECDHE (FS)
- TLS 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

Trust – certificates (PKI)

- trusted CA certificates distributed by browsers/OS
- example: Firefox ~ 150 CA certificates
- Do you trust them all?
- Certificate validation – chain, expiration, server name, signatures, check revocation, … *bugs are common*
- User – let's ignore 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?
- several 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 timestamp; 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; FF – 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.
- TLS Certificate Status Extension
- 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)
- 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

HTTP Public key pinning (HPKP)

- problem: compromised CA issues fake certificates
- bind host to known public-key (or keys)
- information in HTTP header
- “trust on first use” mechanism
- limitations
  - cannot detect MITM attack in the first connection
  - attacker can even insert own pinning info in this case
- now deprecated, replaced by Certificate Transparency

Certificate Transparency

- goals:
  - make hard for a CA to issue a certificate for domain that is not visible to domain owner
  - allow to monitor and audit issued certificates (e.g. by domain owners or CA)
  - protect users against certificates issued maliciously or mistakenly
- Certificate Transparency log
  - Merkle tree of certificate chains (or precertificate chains)
  - 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
  - new certificates contains SCT(s)

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 (HTTP Strict Transport Security, 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
- limitations
  - 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, readig 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)
- DRAWN (2016) – cross-protocol attack, old SSL version with shared RSA key
- ROBOT (2018) – Return Of Bleichenbacher's Oracle Threat