SSL and TLS: Theory and Practice

Chapter 3 – TLS Protocol

Rolf Oppliger

May 28, 2023

Terms of Use

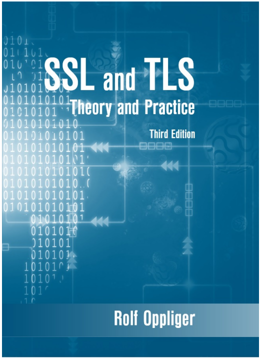
- This work is published with a CC BY-ND 4.0 license (🅭🅯⊜)
  - CC = Creative Commons (🅭)
  - BY = Attribution (🅯)
  - ND = No Derivatives (⊜)

whoami

rolf-oppliger.ch
rolf-oppliger.com

- eSECURITY Technologies Rolf Oppliger (founder and owner)
- Swiss National Cyber Security Centre NCSC (scientific employee)
- University of Zurich (adjunct professor)
- Artech House (author and series editor for information security and privacy)

Reference Book



© Artech House, 2023
ISBN 978-1-68569-015-1

https://www.esecurity.ch/Books/ssltls3e.html

Challenge Me

Outline

3. TLS Protocol

3. TLS Protocol

3. TLS Protocol

3.1 Introduction

- The TLS protocol is structurally identical to the SSL protocol
- It is a client/server protocol stacked on top of TCP
- It consists of the same two layers and (sub)protocols
  - The TLS record protocol fragments, optionally compresses, and cryptographically protects higher-layer protocol data
  - The TLS change cipher spec (20), alert (21), handshake (22), and application data (23) protocols are layered on top of the record layer and the respective protocol
  - New record types
    - 24 for the TLS/DTLS Heartbeat extension
    - 25 for DTLS ciphertext with a connection identifier (CID)
    - 26 for DTLS ACK messages

3. TLS Protocol

3.1 Introduction

- Like the SSL protocol, the TLS protocol employs sessions and connections (where multiple connections may belong to the same session)
- Four connection states
  - Current read and write states
  - Pending read and write states
- All TLS records are processed under the current (read and write) states, whereas the security parameters and elements for the pending states are negotiated by the handshake protocol

3. TLS Protocol

3.1 Introduction

Table 3.1
Security Parameters for a TLS Connection

connection end	Information about whether the entity is considered the "client" or the "server" in the connection
bulk encryption algorithm	Algorithm used for bulk data encryption (including its key size, how much of that key is secret, whether it is a block or stream cipher, and the block size if a block cipher is used)
MAC algorithm	Algorithm used for message authentication
compression algorithm	Algorithm used for data compression
master secret	48-byte secret shared between the client and the server
client random	32-byte value provided by the client
server random	32-byte value provided by the server

3. TLS Protocol

3.1 Introduction

Table 3.2
TLS Connection State Elements

compression state	The current state of the compression algorithm
cipher state	The current state of the encryption algorithm (this includes all values needed to execute the algorithm, such as a key and an IV if the cipher is operated in CBC mode)
MAC secret	MAC secret for this connection
sequence number	64-bit sequence number for the records transmitted under a particular connection state (initially set to zero)

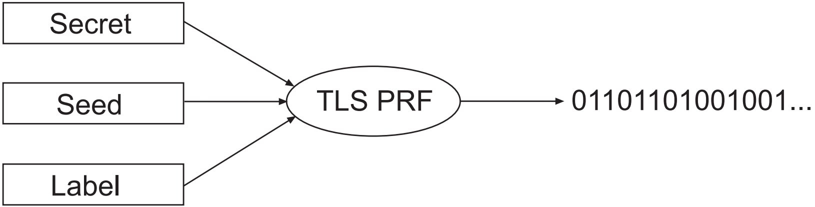
3. TLS Protocol

3.1 Introduction

- A major difference between SSL and TLS 1.0 refers to the way the keying material is generated
    - SSL uses an ad-hoc and hand-crafted construction to generate the master secret and key block
    - TLS 1.0 uses a slightly different construction known as TLS PRF
- The PRF for TLS versions 1.0 and 1.1 is different from the PRF for TLS versions 1.2 and 1.3

3. TLS Protocol

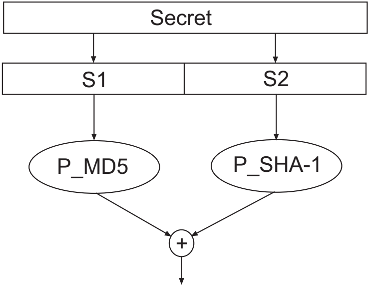
3.1 Introduction



3. TLS Protocol

3.1 Introduction

```
PRF(secret,label,seed) =
   P_MD5(S1,label + seed) XOR P_SHA-1(S2,label + seed)
```



3. TLS Protocol

3.1 Introduction

- The auxiliary data expansion function P_hash is defined as

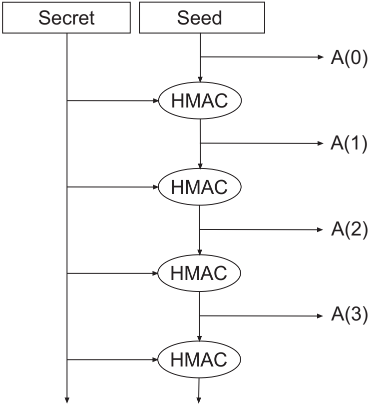
```
P_hash(secret,seed) = HMAC_hash(secret,A(1) + seed) +
                      HMAC_hash(secret,A(2) + seed) +
                      HMAC_hash(secret,A(3) + seed) +
                      ...
```

- The A-function is recursively defined as

```
A(0) = seed
A(i) = HMAC_hash(secret,A(i-1))
```

3. TLS Protocol

3.1 Introduction



3. TLS Protocol

3.1 Introduction

- While TLS 1.0 and 1.1 use MD5 and SHA-1 in the `P_hash`-function, TLS 1.2 uses SHA-256
- TLS 1.3 also uses SHA-256, but the TLS PRF is replaced with the HMAC-based key derivation function (HKDF)
- The HKDF is standardized by the IETF (RFC 5869) and heavily used for Internet applications

3. TLS Protocol

3.1 Introduction

```
master_secret =
   PRF(pre_master_secret,"master secret",
       client_random + server_random)

key_block =
   PRF(master_secret,"key expansion",
       server_random + client_random)

iv_block =
   PRF("","IV block",client_random + server_random)
```

3. TLS Protocol

3.1 Introduction

- Exported keying material (EKM) may be used to (cryptographically) bind an application to a TLS connection
- This may be used to mitigate MITM attacks (e.g., using the token binding mechanism and respective TLS extension)
- The mechanism to construct EKM is similar to the construction of keying material (i.e., it uses the same TLS PRF)

3. TLS Protocol

3.2 TLS 1.0

- TLS 1.0 was published in 1999 (RFC 2246)
- The version is 3,1 (0x0301)
- The cipher suites are inherited from SSL 3.0 (except FORTEZZA-based KEA)
- The MAC construction is more aligned with the "normal" HMAC construction (i.e., *version* field is also included)

$$HMAC_K(TLSCompressed) = h(K \parallel opad \parallel h(K \parallel ipad \parallel seq\_number \parallel \underbrace{type \parallel version \parallel length \parallel fragment}_{TLSCompressed}))$$

3. TLS Protocol

3.2 TLS 1.0

- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA is the only cipher suite that is mandatory in TLS 1.0
- In addition to 3DES, there are cipher suites that employ a Japanese block cipher named Camellia (cf. Table 3.3)
- While SSL 3.0 requires complete certificate chains, TLS 1.0 can handle certificate chains up to a trusted (intermediate) CA
- New alert messages are added (cf. Tables 3.5 and 3.6)
- The CERTIFICATEVERIFY and FINISHED messages are simplified

3. TLS Protocol

3.3 TLS 1.1

- TLS 1.1 was published in 2006 (RFC 4346)
- The version is 3,2 (0x0302)
- The major differences are motivated by cryptographic vulnerabilities that have been exploited by attacks against block ciphers operated in CBC mode (e.g., Vaudenay and BEAST attacks)
- Also, a new way of specifying parameters and parameter values was introduced
- The IANA maintains a number of registries

3. TLS Protocol

3.3 TLS 1.1

- TLS_RSA_WITH_3DES_EDE_CBC_SHA is mandatory (instead of TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA)
- All cipher suites that comprise an export-grade key exchange algorithm are discouraged (i.e., they should no longer be negotiated actively)
- There are a few changes in alert messaging and the way premature closures of sessions are handled (i.e., they can still be resumed)

3. TLS Protocol

3.4 TLS 1.2

- TLS 1.2 was published in 2008 (RFC 5246)
- The version is 3,3 (0x0303)
- It employs SHA-256 (instead of combining MD5 and SHA-1)

  ```
  PRF(secret,label,seed) = P_hash(secret,label+seed)
  ```
- Otherwise, the construction of the keying material remains the same
- The biggest change is the new extension mechanism (RFC 6066)
- There are many TLS extensions (cf. Appendix C)

3. TLS Protocol

3.4 TLS 1.2

- Each extension is structurally the same and consists of two fields
  - A two-byte *type* field
  - A variable-length *data* field of which the first two bytes specify its length
- If a client wants to signal support for the secure renegotiation extension, then it must append 0xFF, 0x01, 0x00, 0x01, and 0x00 to the CLIENTHELLO message
  - 0xFF and 0x01 refer to the extension type (= 65,281)
  - 0x00 and 0x01 refer to the length of the data (= 1)
  - 0x00 refers to the data of the extension

3. TLS Protocol

3.4 TLS 1.2

- TLS_RSA_WITH_AES_128_CBC_SHA is mandatory (instead of TLS_RSA_WITH_3DES_EDE_CBC_SHA)
- All cipher suites that employ DES or IDEA are no longer recommended and thus removed
- The use of authenticated encryption with associated data (AEAD) is addressed in RFC 5116
- Modes of operation for block ciphers to provide AEAD
  - Counter with CBC-MAC (CCM)
  - Galois/counter mode (GCM)
- AEAD requires only an encryption key (no MAC key is needed)

3. TLS Protocol

3.4 TLS 1.2

- There are situations in which public key operations are too expensive or have other disadvantages
- RFC 4279 provides three sets of cipher suites in which the key exchange is based on a preshared key (PSK), i.e., PSK, DHE_PSK, and RSA_PSK
- TLS 1.2 supports ECC-based certificates
- TLS 1.2 provides support for the DEFLATE compression algorithm (that combines LZ77 and Huffman encoding), but does not recommend its use (due to compression-related attacks, cf. Appendix A.6)

3. TLS Protocol

3.5 TLS 1.3

- TLS 1.3 was published in 2018 (RFC 8446)
- The version is 3,4 (0x0304)
- Design goals for TLS 1.3
  - Protection against all (known) types of attacks
  - Efficiency and performance (e.g., in terms of RTTs)

3. TLS Protocol

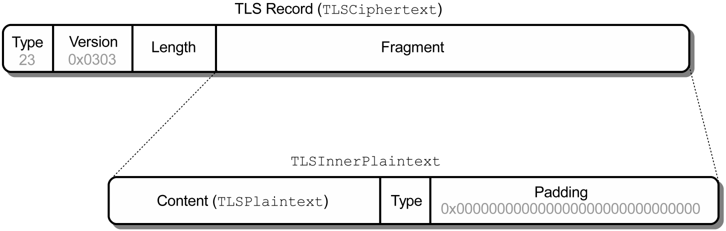
3.5 TLS 1.3

- Security issues
  - No compression
  - PSK-based mechanisms instead of session resumption and renegotiation
  - PSK and/or ephemeral Diffie-Hellman key exchange only (i.e., no static key exchange)
  - PSK or digital signature (i.e., RSA, ECDSA, or EdDSA) for authentication
  - AEAD cipher for data confidentiality and integrity
  - HDKF for key derivation (see below)

3. TLS Protocol

3.5 TLS 1.3

- To provide better privacy protection (i.e., type and length), TLS 1.3 provides a simple encapsulation mechanism for encrypted data (that is also used in DTLS)



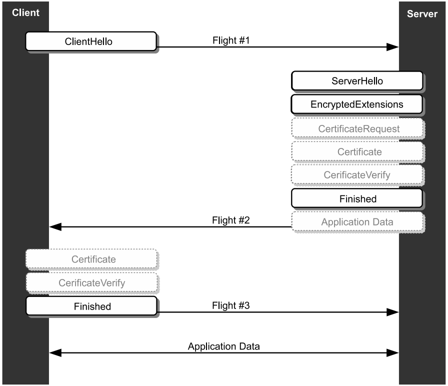
3. TLS Protocol

3.5 TLS 1.3

- With regard to efficiency, the designers of TLS 1.3 were influenced by technologies like Snap Start and False Start from Google, as well as a protocol named OPTLS
- The key idea is to have the client guess a Diffie-Hellman group supported by the server and already provide a Diffie-Hellman share in the first message (i.e., CLIENTHELLO message)
- This results in a 1-RTT handshake
- If there is a PSK, then a 0-RTT handshake is possible

3. TLS Protocol

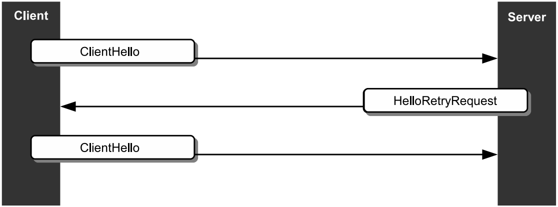
3.5 TLS 1.3



3. TLS Protocol

3.5 TLS 1.3

- If the client-provided parameters do not match, then the server sends back a HELLORETRYREQUEST message (that looks like a SERVERHELLO message with a fixed random value)



3. TLS Protocol

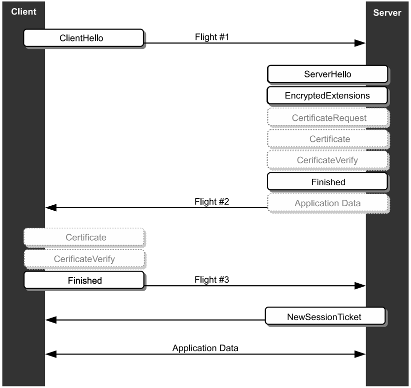
3.5 TLS 1.3

- Up to TLS 1.2, session IDs and session tickets are used to refer to previously established sessions and session keys
- In TLS 1.3, session IDs and session tickets are no longer available
- Instead, they are replaced with PSKs
- If forward secrecy is required, then a PSK can optionally be combined with an ephemeral (EC)DHE key exchange
- The use of a PSK is similar to the use of a session ticket

3. TLS Protocol

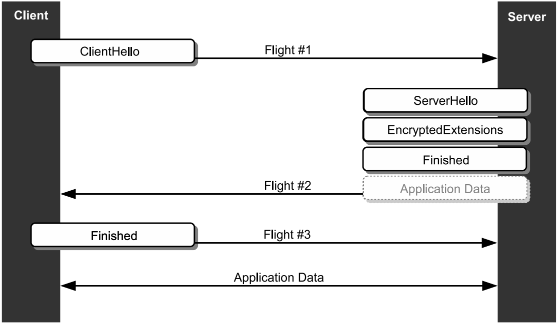
3.5 TLS 1.3

- A NEWSESSIONTICKET message to establish a PSK



3. TLS Protocol
3.5 TLS 1.3

- The PSK can then be used in a (shortened) TLS 1.3 handshake



3. TLS Protocol

3.5 TLS 1.3

- TLS 1.3 differs from its predecessors in terms of key derivation
- It uses a standardized KDF that is based on the HMAC construction and follows the extract-then-expand paradigm
- The HMAC construction is used for both the extraction of a uniform key from a source key ($HKDF_{extract}$) and the expansion of this key into a key stream ($HKDF_{expand}$)
- The resulting KDF is known as HMAC-based KDF (HKDF) and is specified in RFC 5869

3. TLS Protocol

3.5 TLS 1.3

- Extract function (for salt s and source key k)

$$HKDF_{extract}(s, k) = HMAC_s(k) = k'$$

- Expand function (for context string c and length l)

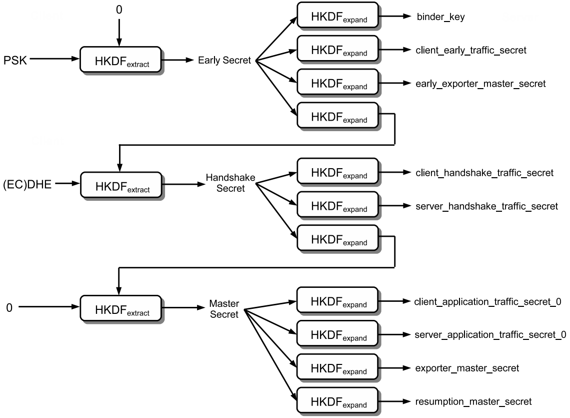
$$HKDF_{expand}(k', c, l) = T_1 \parallel T_2 \parallel \ldots \parallel T_n$$

with T_0 is the zero string and T_i is recursively computed as

$$T_i = HMAC_{k'}(T_{i-1} \parallel c \parallel i)$$

3. TLS Protocol

3.5 TLS 1.3



3. TLS Protocol

3.5 TLS 1.3

- Both the `client_application_traffic_secret_0` and `server_application_traffic_secret_0` keys are used to protect application data
- The distinction between keys to protect handshake traffic and keys to protect application data is new in TLS 1.3
- The postfix `_0` in the traffic keys' names suggests that there is a possibility to update the keys
- This is where the notion of a KEYUPDATE message comes into play (it allows either side to update its traffic secret)
- `Application_traffic_secret_N+1` can be generated from `application_traffic_secret_N` and a distinct label

3. TLS Protocol

3.6 HSTS

- At the 2009 BlackHat conference, Moxie Malinspike presented "new tricks for defeating SSL in practice"
- One of these tricks referred to a tool named SSLStrip
- The tool acts as a MITM and attempts to remove the use of SSL/TLS by modifying unencrypted data on the fly
- To mitigate the attack, it makes sense to strictly apply HTTPS (instead of HTTP)
- This is where **HTTP strict transport security (HSTS)** specified in RFC 6797 comes into play

3. TLS Protocol

3.6 HSTS

- HSTS uses a special HTTP response header named `Strict-Transport-Security`
- Using this header, a web server can inform the browser that SSL/TLS needs to be invoked
- Its use is governed by two directives
  - `Max-age` is mandatory and specifies how long (in seconds) HSTS applies
  - `IncludeSubDomains` is optional and valueless
- If the `includeSubDomains` directive is not used, then HSTS can be used to track users (using "HSTS supercookies")

3. TLS Protocol

3.7 Protocol Transcripts

- Refer to the reference book (pages 136 – 153) for TLS 1.0 and TLS 1.2 transcripts
- Wireshark PCAP files for TLS 1.2 and TLS 1.3 can be downloaded from the reference book's web site

3. TLS Protocol

3.8 Security Analysis

- The SSL and TLS protocols have a long history, and hence they have also been subject to many security analyses and attacks (cf. Appendix A)
- The lessons learned have been incorporated in TLS 1.3
- Two caveats
  - A secure protocol does not exclude the existence of implementation and configuration flaws that may be exploited in attacks (e.g., Hearbleed)
  - Even a highly secure protocol cannot disable cross-protocol attacks (e.g., DROWN, ALPACA, . . .)

3. TLS Protocol

3.9 Final Remarks

- While TLS 1.0 started as a simple protocol, TLS 1.1 and 1.2 added complexity and turned TLS into a protocol that supports many features and is involved
- This makes the protocol flexible and useful in many (maybe nonstandard) situations and use cases
- This includes situations in which the use of SSL/TLS is optional
- This practice is sometimes referred to as opportunistic security
- It is a dual-edged sword

3. TLS Protocol

3.9 Final Remarks

- TLS 1.2 is the typical result of standardization
- It supports almost all technologies and techniques the cryptographic community has come up with, including AES (GCM), ECC, HMAC, and SHA-2
- Whenever a new cryptographic technology or technique is proposed, there is incentive to write an RFC that specifies how to incorporate it into the TLS ecosystem
- Examples include SRP, Camellia and ARIA, Suite B cryptography, and quantum cryptography

3. TLS Protocol

3.9 Final Remarks

- The downside of TLS 1.2's flexibility and feature richness is that interoperability becomes an issue
- When people specified TLS 1.3, care was taken that the protocol is reduced to its core, and that only basic and undisputed functionalities are incorporated
- The result is considered to be highly secure, but also as simple as possible to enable interoperability
- The existence of middleboxes acting as interception proxies has turned out to be a major obstacle for interoperability and end-to-end deployment of TLS 1.3

Questions and Answers

Thank you for your attention