SSL and TLS: Theory and Practice

Chapter 4 – DTLS Protocol

Rolf Oppliger

May 28, 2023

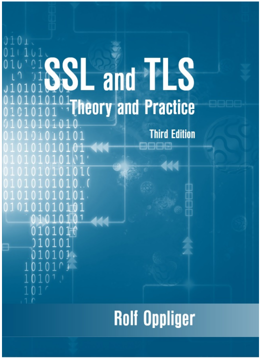
Terms of Use

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

4. DTLS Protocol

4. DTLS Protocol

4. DTLS Protocol

4.1 Introduction

- The SSL/TLS protocols are layered on top of TCP
- There are increasingly many applications and respective protocols that are layered on top of UDP
  - Low-latency media streaming
  - Real-time communications (e.g., IP telephony and videoconferencing)
  - Multicast communications
  - Online gaming
  - Application protocols used for the IoT (e.g., CoAP)
  - ...

4. DTLS Protocol

4.1 Introduction

- In contrast to TCP, UDP only provides a best-effort datagram delivery service that is connectionless and unreliable
- So neither the SSL protocol nor any of the TLS protocol versions can be used to secure UDP-based applications
- The same is true for other connectionless transport layer protocols
  - Datagram congestion control protocol (DCCP)
  - Stream control transmission protocol (SCTP)
  - ...

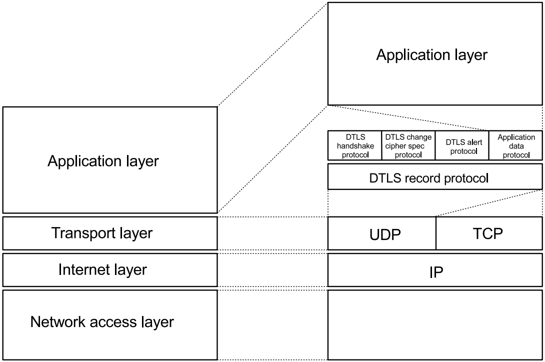
4. DTLS Protocol

4.1 Introduction

- In the early 2000s, the IETF TLS WG started an activity to construct something like "TLS over datagram transport"
- The term **Datagram TLS (DTLS)** was coined and adopted by the WG
- The goal was to define a protocol that derives as little as possible from TLS but is layered on top of UDP
- The placement of DTLS in the TCP/IP protocol stack is similar to the one SSL/TLS

4. DTLS Protocol

4.1 Introduction



4. DTLS Protocol

4.1 Introduction

- The only differences refer to the names and the fact that DTLS is layered on top of UDP
- This also means that the DTLS protocol must be able to deal with datagrams that are not reliably transmitted (i.e., they can be delayed, reordered, lost, or replayed)
- The "normal" TLS protocol cannot handle this type of unreliability
- This should be different with DTLS, but DTLS should be similar to TLS (to minimize new security invention and to maximize the amount of code and infrastructure reuse)

4. DTLS Protocol

4.1 Introduction

- The resulting DTLS protocol provides a solution for the security requirements of UDP-based applications (as well as DCCP- and SCTP-based applications)
- One can reasonably expect that many applications will make use of DTLS
- It can be used alone or in conjunction with other protocols (e.g., SRTP/SRTCP in the case of IP telephony)
- There is no well-known UDP port for DTLS, i.e., the port number depends on the application protocol in use

4. DTLS Protocol

4.1 Introduction

- Three official versions of DTLS
  - DTLS 1.0 (2006) specified in RFC 4347
  - DTLS 1.2 (2012) specified in RFC 6347
  - DTLS 1.3 (2022) specified in RFC 9147
- There is no DTLS version 1.1
- The DTLS protocols are specified as deltas from their TLS counterparts

4. DTLS Protocol

4.1 Introduction

- Even if layered on top of UDP, DTLS must still use some shared state between the communicating entities
- The state defines a security context and is established in a handshake
- In the realm of IPSec/IKE, such a security context is called (security) association
- As with SSL/TLS, the preferred name in DTLS is a connection
- Hence, the terms DTLS connection and (security) association are sometimes used synonymously and interchangeably

4. DTLS Protocol

4.1 Introduction

- Until DTLS 1.2, a DTLS connection is referenced by the IP addresses and port numbers of the entities involved
- In RFC 9146, the notion of a connection identifier (CID) is introduced for DTLS 1.2
- A CID is a logical identifier that refers to a security context
- To improve privacy, there can be multiple CIDs that identify a security context, and it is recommended to use a new CID whenever an entity changes its IP address or port number

4. DTLS Protocol

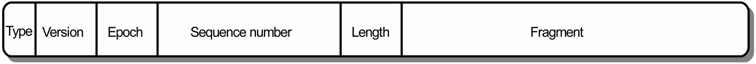
4.1 Introduction

- Most things said for SSL/TLS also apply to DTLS (e.g., cipher suites, alert messages, . . .)
- Two major problems (caused by UDP) must be resolved
  - Due to the connectionless nature of UDP, each DTLS record must stand for itself (and must be encrypted and decrypted independently from other records)
    → *No interrecord dependency*
  - Due to the unreliability of UDP, some reliability features (of TCP) must be retrofitted into DTLS
    → *Message sequence numbers to handle fragmentation/ reassembly and in-oder delivery*

4. DTLS Protocol

4.2 DTLS 1.0

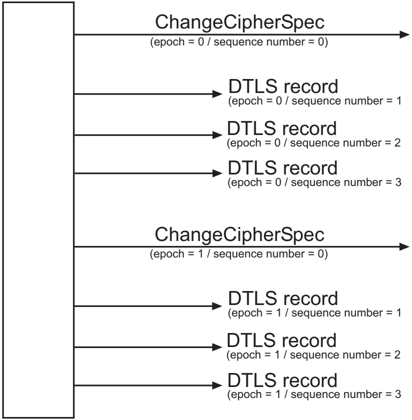
- In addition to the "normal" header fields, a DTLS 1.0 record header comprises a 2-byte epoch field and a 6-byte sequence number field



- These fields can be used to number and thus uniquely identify DTLS records

4. DTLS Protocol

4.2 DTLS 1.0



4. DTLS Protocol

4.2 DTLS 1.0

- In a DTLS record header, the version field always comprises the 1's complement of the DTLS version in use
- This is to ensure that records from TLS and DTLS can be easily separated and told apart
- Actual values
    - In DTLS 1.0, the version field comprises 254,255 (or 0xFEFF)
    - In DTLS 1.2, the version field comprises 254,253 (or 0xFEFD)
    - In DTLS 1.3, the version field comprises 254,252 (or 0xFEFC)

4. DTLS Protocol

4.2 DTLS 1.0

- To deal with the second problem area (i.e., unreliability of UDP), the DTLS handshake protocol must be changed
- The changes refer to
  - The possibility of message fragmentation and reassembly
  - Message retransmission
  - Replay detection
  - Protection against some DoS attacks

4. DTLS Protocol

4.2 DTLS 1.0

- The header of a TLS handshake message comprises a 3-byte length field, meaning that such a message can be up to 2^{24} bytes long
- Such a message is then transmitted in multiple records of at most 2^{14} bytes
- Normally, handshake messages are much shorter and transmitted in a single record
- Otherwise, message fragmentation and reassembly are needed

4. DTLS Protocol

4.2 DTLS 1.0

- To enable message fragmentation and reassembly, each DTLS handshake message header comprises three new fields

Table 4.1
Exemplary Header Fields Used for Fragmentation and Reassembly

Header field	R_1	R_2	R_3	R_4
Message sequence [2 bytes]	i	i	i	i
Fragment offset [3 bytes]	0	n_1	$n_1 + n_2$	$n_1 + n_2 + n_3$
Fragment length [3 bytes]	n_1	n_2	n_3	n_4

4. DTLS Protocol

4.2 DTLS 1.0

- Due to the fact that DTLS is layered on top of UDP, UDP datagrams and respective DTLS records and handshake messages may get lost
- On the record layer, sequence numbers deal with this issue
- On the handshake protocol layer, there is no mechanism in TLS to handle packet loss, and hence DTLS must deal with it
- The standard way is to start a retransmission timer when a message is sent, and to resend the message if no acknowledgment is received before it expires

4. DTLS Protocol

4.2 DTLS 1.0

- In addition to the reordering and loss of DTLS records and messages, one may also be worried about records or messages being replayed
- DTLS optionally provides support for replay detection
- The technique is borrowed from IPsec/IKE, where a bitmap window of received packets is maintained
- Packets that are too old to fit in the window and packets that have previously been received are silently discarded
- DTLS supports a similar technique for DTLS records (optional)

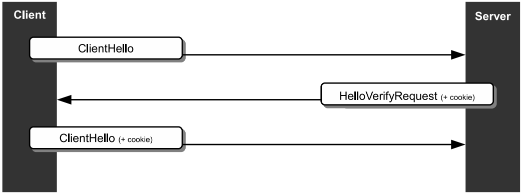
4. DTLS Protocol

4.2 DTLS 1.0

- To mitigate some DoS attacks, the DTLS protocol uses a cookie exchange mechanism that is borrowed from Photuris
- Before the proper handshake begins, the server must provide a stateless cookie in a HelloVerifyRequest message
- The client must replay this cookie in the ClientHello message
- A cookie should be generated in such a way that it can be verified without retaining per-client state on the server
- Ideally, it is a keyed one hash value of some client-specific parameters, such as the client IP address

4. DTLS Protocol

4.2 DTLS 1.0



- The key in use needs to be known only to the server
- While DTLS 1.0 supports cookies up to 32 bytes, this maximum size is enlarged in DTLS 1.2 to 255 bytes

4. DTLS Protocol

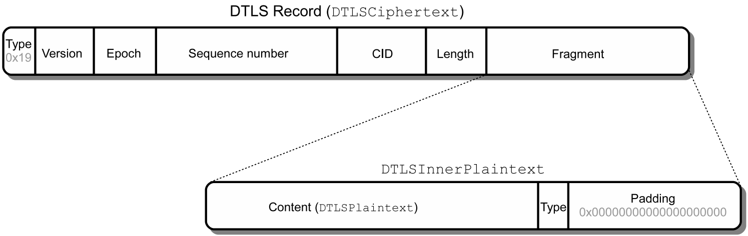
4.3 DTLS 1.2

- DTLS 1.2 is very similar to DTLS 1.0
- There is no (noticeable) change in the handshake protocol
- A major change refers to the record protocol and the possibility of using CIDs to improve the performance of DTLS
- CIDs are important, because the use of dynamically changing IP addresses and port numbers has increased over time

4. DTLS Protocol

4.3 DTLS 1.2

- DTLS 1.2 CID-enhanced ciphertext record format
- The type is set to `tls12_cid` that refers to 25 (0x19)



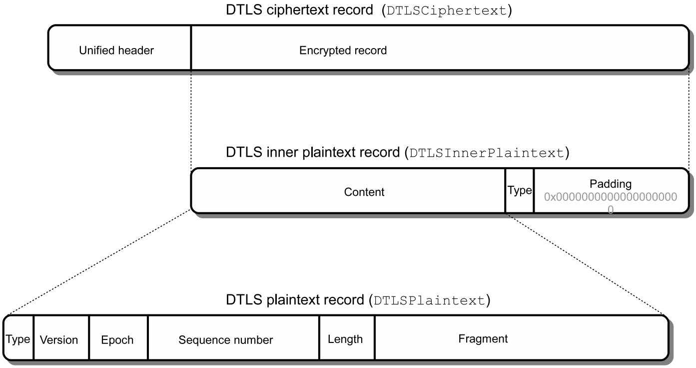
4. DTLS Protocol

4.4 DTLS 1.3

- Due to the alignment with TLS 1.3, there are fundamental changes in the DTLS 1.3 record and handshake protocols
- The DTLS 1.3 record format is optimized with a variable length encoding of the header
- If a record is not encrypted or does not make use of the CID mechanism, then the "normal" record format still applies
- Otherwise, a new ciphertext record format is used

4. DTLS Protocol

4.4 DTLS 1.3



4. DTLS Protocol
4.4 DTLS 1.3

- DTLS 1.3 uses epoch and sequence number fields that are 8 bytes long each
- The CID is part of the unified header (if present), but most other header fields appear in the plaintext record
  - Type (1 byte)
  - Version (2 bytes, `legacy_record_version` = 254,253 = DTLS 1.2)
  - Epoch (2 least significant bytes)
  - Sequence number (6 least significant bytes)
  - Length (2 bytes)

4. DTLS Protocol

4.4 DTLS 1.3

0	0	1	C	S	L	E	E

- The unified header has a variable length and a distinct format
- It starts with a bit mask
  - 3-bit sequence 001
  - C-bit is set if a CID is used
  - S-bit indicates the length of the sequence number (8/16 bits)
  - L-bit is set if the length is present
  - E-bits comprise the two least significant bits of the epoch

4. DTLS Protocol
4.4 DTLS 1.3

- After the bit mask, the unified header comprises one or several fields
  - If the C-bit is set, then the header comprises a variable-length CID
  - The low-order 8 or 16 bits from the record's sequence number
  - If the L-bit is set, then the header comprises the record length (otherwise the record consumes the rest of the UDP datagram)
- Bit mask for minimum-length header: 001000EE
- Bit mask for maximum-length header: 001111EE

4. DTLS Protocol

4.4 DTLS 1.3

- In DTLS 1.0 and 1.2, the content type of a record always appears in the first byte of the header
- This makes it easy to process an incoming record
- In DTLS 1.3, the first byte still determines how a record must be processed, but the rules are more involved
- Note that a record may not be encrypted, in which case the "normal" record format applies

4. DTLS Protocol

4.4 DTLS 1.3

- In this case, the content type is provided in the first byte of the header
  - Value 20 suggests that the record comprises a CHANGECIPHERSPEC message that only occurs in DTLS 1.0 or 1.2
  - Values 21, 22, or 26 suggest that the record comprises an alert, handshake, or ACK message that is not encrypted (otherwise the message would be sent in a record that uses the new format)
  - Values 23 or 24 suggest that the record comprises application data or a heartbeat message in DTLS 1.0 or 1.2
  - Value 25 suggests that the record comprises a DTLS 1.2 CID-enhanced ciphertext record

4. DTLS Protocol

4.4 DTLS 1.3

- If the new record format applies, then the first byte of the unified header must begin with 001
- After decryption, the content type suggests whether the record refers to
  - An alert message (21)
  - A handshake message (22)
  - Application data (23)
  - A heartbeat message (24)
  - An ACK message (26)

4. DTLS Protocol

4.4 DTLS 1.3

- The DTLS 1.3 handshake protocol is aligned with TLS 1.3
- It uses the same message formats, flows, and logic
- This applies to

- ClientHello (1)
- ServerHello (2)
- NewSessionTicket* (4)
- EndOfEarlyData (5)
- EncryptedExtensions (8)
- Certificate (11)
- CertificateRequest (13)
- CertificateVerify (15)
- Finished (20)
- KeyUpdate* (24)

4. DTLS Protocol

4.4 DTLS 1.3

- A few messages have been removed in DTLS 1.3
  - ServerHelloDone message
  - ChangeCipherSpec message
  - ServerKeyExchange message
  - ClientKeyExchange message
- The ServerHelloDone and ChangeCipherSpec messages have been removed without any replacement
- The contents of the ServerKeyExchange and ClientKeyExchange messages have moved to extensions

4. DTLS Protocol

4.4 DTLS 1.3

- There are also two new messages related to CIDs that have been introduced in DTLS 1.3
  - RequestConnectionID* (9)
  - NewConnectionID* (10)
- If a client and server have negotiated the use of CIDs with the `connection_id` extension, then either side can request a CID
- All messages marked with a superscript star (*) refer to post-handshake messages

4. DTLS Protocol

4.4 DTLS 1.3

- DTLS 1.3 assigns dedicated epoch values to handshake messages to identify the correct cipher state
  - Epoch value 0 is used for unencrypted messages, i.e., ClientHello, ServerHello, and HelloRetryRequest
  - Epoch value 1 is used for messages protected with keys derived from `client_early_traffic_secret`
  - Epoch value 3 is used for payloads protected with keys derived from `*_application_traffic_secret_0`
  - Epoch values 4 to $2^{64} - 1$ are used for payloads protected with keys derived from `*_application_traffic_secret_N`
- $*$ refers to either `client` or `server`

4. DTLS Protocol

4.4 DTLS 1.3

- DTLS 1.0 and 1.2 use a timer to detect lost messages and start a message retransmission (if the timer expires)
- This applies to all messages in a flight
- DTLS 1.3 uses ACK messages that allow an entity to acknowledge individual records
- ACK messages are not handshake messages, but messages with a separate content type 26 (instead of 22 as for handshake messages)

4. DTLS Protocol
4.4 DTLS 1.3

- DTLS 1.3 reuses the HELLORETRYREQUEST message from TLS 1.3 (to replace the HELLOVERIFYREQUEST message from DTLS 1.0 and 1.2)
- The HELLORETRYREQUEST has the same format as a SERVERHELLO message
- It is used for message retransmission and the cookie exchange (in lieu of a HELLOVERIFYREQUEST message)
- The cookie must be carried in a `cookie` extension originally defined for TLS 1.3 but mainly used for DTLS 1.3

4. DTLS Protocol

4.5 Security Analysis

- Given the fact that the DTLS protocol was designed to be as similar as possible to the TLS protocol, one can reasonably expect that the security analyses of TLS, at least in principle, still apply to DTLS
- Taking into account that TLS/DTLS 1.3 only employs state of the art cryptography, such as AEAD ciphers and no static key exchange, people have a comfortable gut feeling about the security of DTLS 1.3
- It is just a gut feeling, and fairly little is known about the real security of the protocol

4. DTLS Protocol

4.5 Security Analysis

- From all attacks that can be mounted against the SSL/TLS protocols, there are some attacks that work against DTLS, and there are some attacks that don't work
- For example, compression-related attacks and padding oracle attacks generally work against DTLS, whereas attacks against RC4 don't work
- Padding oracle attacks are more subtle to mitigate in the case of DTLS (than in the case of TLS)

4. DTLS Protocol

4.5 Security Analysis

- Maybe the biggest worry with regard to the security of the DTLS protocol is related to the fact that DTLS is layered on top of UDP instead of TCP
- There may be entirely new attacks that take advantage of this fact
- In network security, UDP-based applications are more difficult to secure than TCP-based ones
- We (have to) live with the comfortable gut feeling and assume that the DTLS protocol provides a reasonable level of security

4. DTLS Protocol

4.6 Final Remarks

- DTLS is very comparable to SSL/TLS
- But DTLS is still a relatively new protocol that is not yet widely deployed
- The lack of implementation experience goes hand in hand with the fact that there are only a few studies about the optimal deployment of DTLS
- Further studies are needed
- The same is true for firewall traversal and the security of it

Questions and Answers

Thank you for your attention