

Workgroup: TSVWG

Internet-Draft:

draft-ietf-tsvwg-dtls-over-sctp-bis-04

Published: 23 June 2022

Intended Status: Standards Track

Expires: 25 December 2022

Authors: M. Westerlund J. Preuß Mattsson C. Porfiri

Ericsson

Ericsson

Ericsson

**Datagram Transport Layer Security (DTLS) over Stream Control
Transmission Protocol (SCTP)**

Abstract

This document describes the usage of the Datagram Transport Layer Security (DTLS) protocol to protect user messages sent over the Stream Control Transmission Protocol (SCTP). It is an improved alternative to the existing rfc6083.

DTLS over SCTP provides mutual authentication, confidentiality, integrity protection, and replay protection for applications that use SCTP as their transport protocol and allows client/server applications to communicate in a way that is designed to give communications privacy and to prevent eavesdropping and detect tampering or message forgery.

Applications using DTLS over SCTP can use almost all transport features provided by SCTP and its extensions. This document is an improved alternative to RFC 6083 and removes the 16 kB limitation on protected user message size by defining a secure user message fragmentation so that multiple DTLS records can be used to protect a single user message. It further updates the DTLS versions to use, as well as the HMAC algorithms for SCTP-AUTH, and simplifies secure implementation by some stricter requirements on the establishment procedures.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at <https://github.com/gloinul/draft-westerlund-tsvwg-dtls-over-sctp-bis>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 December 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- 1. [Introduction](#)
 - 1.1. [Overview](#)
 - 1.1.1. [Comparison with TLS over SCTP](#)
 - 1.1.2. [Changes from RFC 6083](#)
 - 1.2. [DTLS Version](#)
 - 1.3. [Terminology](#)
 - 1.4. [Abbreviations](#)
- 2. [Conventions](#)
- 3. [DTLS Considerations](#)
 - 3.1. [Version of DTLS](#)
 - 3.2. [Cipher Suites and Cryptographic Parameters](#)
 - 3.3. [Message Sizes](#)
 - 3.4. [Replay Protection](#)
 - 3.5. [Path MTU Discovery](#)
 - 3.6. [Retransmission of Messages](#)
- 4. [SCTP Considerations](#)
 - 4.1. [Mapping of DTLS Records](#)
 - 4.2. [DTLS Connection Handling](#)
 - 4.3. [Payload Protocol Identifier Usage](#)
 - 4.4. [Stream Usage](#)
 - 4.5. [Chunk Handling](#)
 - 4.6. [SCTP-AUTH Hash Function](#)

- [4.7. Parallel DTLS connections](#)
- [4.8. Renegotiation and KeyUpdate](#)
 - [4.8.1. DTLS 1.2 Considerations](#)
 - [4.8.2. DTLS 1.3 Considerations](#)
- [4.9. Handling of Endpoint-Pair Shared Secrets](#)
 - [4.9.1. DTLS 1.2 Considerations](#)
 - [4.9.2. DTLS 1.3 Considerations](#)
- [4.10. Shutdown](#)
- [5. DTLS/SCTP Control Message](#)
 - [5.1. SHUTDOWN-Request](#)
 - [5.2. Ready To Close Indication](#)
- [6. DTLS over SCTP Service](#)
 - [6.1. Adaptation Layer Indication in INIT/INIT-ACK](#)
 - [6.2. DTLS over SCTP Initialization](#)
 - [6.3. Client Use Case](#)
 - [6.4. Server Use Case](#)
 - [6.5. RFC 6083 Fallback](#)
 - [6.5.1. Client Fallback](#)
 - [6.5.2. Server Fallback](#)
- [7. SCTP API Consideration](#)
- [8. IANA Considerations](#)
 - [8.1. TLS Exporter Label](#)
 - [8.2. SCTP Adaptation Layer Indication Code Point](#)
 - [8.3. SCTP Payload Protocol Identifiers](#)
- [9. Security Considerations](#)
 - [9.1. Cryptographic Considerations](#)
 - [9.2. Downgrade Attacks](#)
 - [9.3. Targeting DTLS Messages](#)
 - [9.4. Authentication and Policy Decisions](#)
 - [9.5. Resumption and Tickets](#)
 - [9.6. Privacy Considerations](#)
 - [9.7. Pervasive Monitoring](#)
- [10. Contributors](#)
- [11. Acknowledgments](#)
- [12. References](#)
 - [12.1. Normative References](#)
 - [12.2. Informative References](#)
- [Appendix A. Motivation for Changes](#)
- [Authors' Addresses](#)

1. Introduction

1.1. Overview

This document describes the usage of the Datagram Transport Layer Security (DTLS) protocol, as defined in DTLS 1.2 [[RFC6347](#)], and DTLS 1.3 [[RFC9147](#)], over the Stream Control Transmission Protocol (SCTP), as defined in [[RFC9260](#)] with Authenticated Chunks for SCTP (SCTP-AUTH) [[RFC4895](#)].

This specification provides mutual authentication of endpoints, confidentiality, integrity protection, and replay protection of user messages for applications that use SCTP as their transport protocol. Thus, it allows client/server applications to communicate in a way that is designed to give communications privacy and to prevent eavesdropping and detect tampering or message forgery. DTLS/SCTP uses DTLS for mutual authentication, key exchange with forward secrecy for SCTP-AUTH, and confidentiality of user messages. DTLS/SCTP use SCTP and SCTP-AUTH for integrity protection and replay protection of all SCTP Chunks that can be authenticated, including user messages.

Applications using DTLS over SCTP can use almost all transport features provided by SCTP and its extensions. DTLS/SCTP supports:

- *preservation of message boundaries.
- *a large number of unidirectional and bidirectional streams.
- *ordered and unordered delivery of SCTP user messages.
- *the partial reliability extension as defined in [[RFC3758](#)].
- *the dynamic address reconfiguration extension as defined in [[RFC5061](#)].
- *User messages of any size.

The method described in this document requires that the SCTP implementation supports the optional feature of fragmentation of SCTP user messages as defined in [[RFC9260](#)]. The implementation is required to have an SCTP API (for example the one described in [[RFC6458](#)]) that supports partial user message delivery and also recommended that I-DATA chunks as defined in [[RFC8260](#)] is used to efficiently implement and support larger user messages.

To simplify implementation and reduce the risk for security holes, limitations have been defined such that STARTTLS as specified in [[RFC3788](#)] is no longer supported.

1.1.1. Comparison with TLS over SCTP

TLS, from which DTLS was derived, is designed to run on top of a byte-stream-oriented transport protocol providing a reliable, in-sequence delivery. TLS over SCTP as described in [[RFC3436](#)] has some serious limitations:

- *It does not support the unordered delivery of SCTP user messages.
- *It does not support partial reliability as defined in [[RFC3758](#)].

- *It only supports the usage of the same number of streams in both directions.

- *It uses a TLS connection for every bidirectional stream, which requires a substantial amount of resources and message exchanges if a large number of streams is used.

1.1.2. Changes from RFC 6083

The DTLS over SCTP solution defined in RFC 6083 had the following limitations:

- *The maximum user message size is 2^{14} (16384) bytes, which is a single DTLS record limit.

- *DTLS 1.0 has been deprecated for RFC 6083 requiring at least DTLS 1.2 [[RFC8996](#)]. This creates additional limitation as discussed in [Section 1.2](#).

- *DTLS messages that don't contain protected user message data were limited to only be sent on Stream 0 and requiring that stream to be in-order delivery which could potentially impact applications.

This specification defines the following changes compared with RFC 6083:

- *Removes the limitations on user messages sizes by defining a secure fragmentation mechanism. It is optional to support message sizes over $2^{64}-1$ bytes.

- *Enable DTLS key-change without requiring draining all inflight user message from SCTP.

- *Mandates that more modern DTLS version are used (DTLS 1.2 or 1.3)

- *Mandates support of modern HMAC algorithm (SHA-256) in the SCTP authentication extension [[RFC4895](#)].

- *Recommends support of [[RFC8260](#)] to enable interleaving of large SCTP user messages to avoid scheduling issues.

- *Applies stricter requirements on always using DTLS for all user messages in the SCTP association.

- *Requires that SCTP-AUTH is applied to all SCTP Chunks that can be authenticated.

- *Requires support of partial delivery of user messages.

1.2. DTLS Version

Using DTLS 1.2 instead of using DTLS 1.0 limits the lifetime of a DTLS connection and the data volume which can be transferred over a DTLS connection. This is caused by:

- *The number of renegotiations in DTLS 1.2 is limited to 65534 compared to unlimited in DTLS 1.0.
- *While the AEAD limits in DTLS 1.3 does not formally apply to DTLS 1.2 the mathematical limits apply equally well to DTLS 1.2.

DTLS 1.3 comes with a large number of significant changes.

- *Renegotiations are not supported and instead partly replaced by KeyUpdates. The number of KeyUpdates is limited to 2^{48} .
- *Strict AEAD significantly limits on how much many packets can be sent before rekeying.

Many applications using DTLS/SCTP are of semi-permanent nature and use SCTP associations with expected lifetimes of months or even years, and where there is a significant cost of bringing down the SCTP association in order to restart it. Such DTLS/SCTP usages that need:

- *Periodic re-authentication and transfer of revocation information of both endpoints (not only the DTLS client).
- *Periodic rerunning of Diffie-Hellman key-exchange to provide forward secrecy and mitigate static key exfiltration attacks.
- *Perform SCTP-AUTH rekeying.

At the time of publication DTLS 1.3 does not support any of these, where DTLS 1.2 renegotiation functionality can provide this functionality in the context of DTLS/SCTP. To address these requirements from semi-permanent applications, this document use several overlapping DTLS connections with either DTLS 1.2 or 1.3. Having uniform procedures reduces the impact when upgrading from 1.2 to 1.3 and avoids using the renegotiation mechanism which is disabled by default in many DTLS implementations.

To address known vulnerabilities in DTLS 1.2 this document describes and mandates implementation constraints on ciphers and protocol options. The DTLS 1.2 renegotiation mechanism is forbidden to be used as it creates need for additional mechanism to handle race conditions and interactions between using DTLS connections in parallel.

Secure negotiation of the DTLS version is handled by the DTLS handshake. If the endpoints do not support a common DTLS version the DTLS handshake will be aborted.

In the rest of the document, unless the version of DTLS is specifically called out the text applies to both versions of DTLS.

1.3. Terminology

This document uses the following terms:

Association: An SCTP association.

Connection: An DTLS connection. It is uniquely identified by a connection identifier.

Stream: A unidirectional stream of an SCTP association. It is uniquely identified by a stream identifier.

1.4. Abbreviations

AEAD: Authenticated Encryption with Associated Data

DTLS: Datagram Transport Layer Security

HMAC: Keyed-Hash Message Authentication Code

MTU: Maximum Transmission Unit

PPID: Payload Protocol Identifier

SCTP: Stream Control Transmission Protocol

SCTP-AUTH: Authenticated Chunks for SCTP

TCP: Transmission Control Protocol

TLS: Transport Layer Security

ULP: Upper Layer Protocol

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

3. DTLS Considerations

3.1. Version of DTLS

This document defines the usage of either DTLS 1.3 [[RFC9147](#)], or DTLS 1.2 [[RFC6347](#)]. Earlier versions of DTLS MUST NOT be used (see [[RFC8996](#)]). DTLS 1.3 is RECOMMENDED for security and performance reasons. It is expected that DTLS/SCTP as described in this document will work with future versions of DTLS.

3.2. Cipher Suites and Cryptographic Parameters

For DTLS 1.2, the cipher suites forbidden by [[RFC9113](#)] MUST NOT be used. For all versions of DTLS, cryptographic parameters giving confidentiality and forward secrecy MUST be used.

3.3. Message Sizes

DTLS/SCTP, automatically fragments and reassembles user messages. This specification defines how to fragment the user messages into DTLS records, where each DTLS record allows a maximum of 2^{14} protected bytes. Each DTLS record adds some overhead, thus using records of maximum possible size are recommended to minimize the transmitted overhead. DTLS 1.3 has much less overhead than DTLS 1.2 per record.

The sequence of DTLS records is then fragmented into DATA or I-DATA Chunks to fit the path MTU by SCTP. These changes ensure that DTLS/SCTP has the same capability as SCTP to support user messages of any size. However, to simplify implementations it is OPTIONAL to support user messages larger than $2^{64}-1$ bytes. This is to allow implementation to assume that 64-bit length fields and offset pointers will be sufficient.

The SCTP-API defined in [[RFC6458](#)] results in an implementation limitation when it comes to support any user message sizes. That API does not allow the changing of the SCTP-AUTH key used to protect the sending of a particular user message. Thus, user messages that will be transmitted over periods of time on the order or longer than the interval between rekeying can't be supported. Beyond delaying the completion of a rekeying until the message has been transmitted, the session can deadlock if the DTLS connection used to protect this long user message reaches the limit of number of bytes transmitted with a particular key. However, this is not an interoperability issue as it is the sender side's API that limits what can be sent and thus the sender implementation will have to address this issue.

The security operations and reassembly process requires that the protected user message, i.e., with DTLS record overhead, is stored in the receiver's buffer. This buffer space will thus put a limit on

the largest size of plain text user message that can be transferred securely. However, by mandating the use of the partial delivery of user messages from SCTP and assuming that no two messages received on the same stream are interleaved (as it is the case when using the API defined in [\[RFC6458\]](#)) the minimally required buffering prior to DTLS processing is a single DTLS record per used incoming stream. This enables the DTLS/SCTP implementation to provide the Upper Layer Protocol (ULP) with each DTLS record's content when it has been decrypted and its integrity been verified enabling partial user message delivery to the ULP. Implementations can trade-off buffer memory requirements in the DTLS layer with transport overhead by using smaller DTLS records. However, for efficient operation and avoiding flow control stalls if user message fragments are not frequently and expiendtlly moved to upper layer memory buffers, the receiver buffer needs to be larger.

The DTLS/SCTP implementation is expected to behave very similar to just SCTP when it comes to handling of user messages and dealing with large user messages and their reassembly and processing. Making it the ULP responsible for handling any resource contention related to large user messages.

3.4. Replay Protection

SCTP-AUTH [\[RFC4895\]](#) does not have explicit replay protection. However, the combination of SCTP-AUTH's protection of DATA or I-DATA chunks and SCTP user message handling will prevent third party attempts to inject or replay SCTP packets resulting in impact on the received protected user message. In fact, this document's solution is dependent on SCTP-AUTH and SCTP to prevent reordering, duplication, and removal of the DTLS records within each protected user message. This includes detection of changes to what DTLS records start and end the SCTP user message, and removal of DTLS records before an increment to the epoch. Without SCTP-AUTH, these would all have required explicit handling.

DTLS optionally supports record replay detection. Such replay detection could result in the DTLS layer dropping valid messages received outside of the DTLS replay window. As DTLS/SCTP provides replay protection even without DTLS replay protection, the replay detection of DTLS MUST NOT be used.

3.5. Path MTU Discovery

DTLS Path MTU Discovery MUST NOT be used. Since SCTP provides Path MTU discovery and fragmentation/reassembly for user messages, and specified in [Section 3.3](#), DTLS can send maximum sized DTLS Records.

3.6. Retransmission of Messages

SCTP provides a reliable and in-sequence transport service for DTLS messages that require it. See [Section 4.4](#). Therefore, DTLS procedures for retransmissions MUST NOT be used.

4. SCTP Considerations

4.1. Mapping of DTLS Records

The SCTP implementation MUST support fragmentation of user messages using DATA [[RFC9260](#)], and optionally I-DATA [[RFC8260](#)] chunks.

DTLS/SCTP works as a shim layer between the user message API and SCTP. On the sender side a user message is split into fragments m_0 , m_1 , m_2 , each no larger than $2^{14} = 16384$ bytes.

$m_0 \mid m_1 \mid m_2 \mid \dots = \text{user_message}$

The resulting fragments are protected with DTLS and the records are concatenated

$\text{user_message}' = \text{DTLS}(m_0) \mid \text{DTLS}(m_1) \mid \text{DTLS}(m_2) \mid \dots$

The new $\text{user_message}'$, i.e., the protected user message, is the input to SCTP.

On the receiving side, the length field in each DTLS record can be used to determine the boundaries between DTLS records. DTLS can decrypt individual records or a concatenated sequence of records. The last DTLS record can be found by subtracting the length of individual records from the length of $\text{user_message}'$. Whether to decrypt individual records, sequences of records, or the whole $\text{user_message}'$ is left to the implementation. The output from the DTLS decryption(s) is the fragments m_0 , m_1 , $m_2 \dots$. The user_message is reassembled from decrypted DTLS records as $\text{user_message} = m_0 \mid m_1 \mid m_2 \dots$. There are three failure cases an implementation needs to detect and then act on:

1. Failure in decryption and integrity verification process of any DTLS record. Due to SCTP-AUTH preventing delivery of injected or corrupt fragments of the protected user message this should only occur in case of implementation errors or internal hardware failures or the necessary security context has been prematurely discarded.
2. In case the SCTP layer indicates an end to a user message, e.g., when receiving a MSG_EOR in a `recvmsg()` call when using the API described in [[RFC6458](#)], and the last buffered DTLS

record length field does not match, i.e., the DTLS record is incomplete.

3. Unable to perform the decryption processes due to lack of resources, such as memory, and have to abandon the user message fragment. This specification is defined such that the needed resources for the DTLS/SCTP operations are bounded for a given number of concurrent transmitted SCTP streams or unordered user messages.

The above failure cases all result in the receiver failing to recreate the full user message. This is a failure of the transport service that is not possible to recover from in the DTLS/SCTP layer and the sender could believe the complete message have been delivered. This error **MUST NOT** be ignored, as SCTP lacks any facility to declare a failure on a specific stream or user message, the DTLS connection and the SCTP association **SHOULD** be terminated. A valid exception to the termination of the SCTP association is if the receiver is capable of notifying the ULP about the failure in delivery and the ULP is capable of recovering from this failure.

Note that if the SCTP extension for Partial Reliability (PR-SCTP) [[RFC3758](#)] is used for a user message, user message may be partially delivered or abandoned. These failures are not a reason for terminating the DTLS connection and SCTP association.

The DTLS Connection ID **MUST** be negotiated ([[RFC9146](#)] or Section 9 of [[RFC9147](#)]). If DTLS 1.3 is used, the length field in the record layer **MUST** be included in all records. A 16-bit sequence number **SHOULD** be used rather than 8-bit to minimize issues with DTLS record sequence number wrapping.

The ULP may use multiple messages simultaneous, and the progress and delivery of these messages are progressing independently, thus the receiving DTLS/SCTP implementation may not receive records in order in case of packet loss. Assuming that the sender will send the DTLS records in order the DTLS records were created (which may not be certain in some implementations), then there is a risk that DTLS sequence number have wrapped if the amount of data in flight is more than the sequence number covers. Thus, for 8-bit sequence number space with 16384 bytes records the receiver window only needs to be $256 * 16384 = 4,194,304$ bytes for this risk to definitely exist. While a 16-bit sequence number should not have any sequence number wraps for receiver windows up to 1 GB. The DTLS/SCTP may not be tightly integrated and the DTLS records may not be requested to be sent in strict sequence order, in these case additional guard ranges are needed.

Also, if smaller DTLS records are used, this limit will be correspondingly reduced. The DTLS/SCTP Sender needs to choose sequence number length and DTLS Record size so that the product is larger than the used receiver window, preferably twice as large. Receiver implementations that are offering receiver windows larger than the product 65536*16384 bytes MUST be capable of handling sequence number wraps through trial decoding with a lower values in the higher bits of the extended sequence number.

Section 4 of [[RFC9146](#)] states "If, however, an implementation chooses to receive different lengths of CID, the assigned CID values must be self-delineating since there is no other mechanism available to determine what connection (and thus, what CID length) is in use.". As this solution requires multiple connection IDs, using a zero-length CID will be highly problematic as it could result in that any DTLS records with a zero length CID ends up in another DTLS connection context, and there fail the decryption and integrity verification. And in that case to avoid losing the DTLS record, it would have to be forwarded to the zero-length CID using DTLS Connection and decryption and validation must be tried. Resulting in higher resource utilization. Thus, it is REQUIRED to use non-zero length CID values, and instead RECOMMENDED to use a single common length for the CID values. A single byte should be sufficient, as reuse of old CIDs is possible as long as the implementation ensure they are not used in near time to the previous usage.

4.2. DTLS Connection Handling

DTLS/SCTP is negotiated on SCTP level as an adaptation layer ([Section 6](#)). After a succesful negotiation of the DTLS/SCTP during SCTP association establishment, a DTLS connection MUST be established prior to transmission of any ULP user messages. All DTLS connections are terminated when the SCTP association is terminated. A DTLS connection MUST NOT span multiple SCTP associations.

As it is required to establish the DTLS connection at the beginning of the SCTP association, either of the peers should never send any SCTP user messages that are not protected by DTLS. So, the case that an endpoint receives data that is not either DTLS messages or protected user messages in the form of a sequence of DTLS Records on any stream is a protocol violation. The receiver MAY terminate the SCTP association due to this protocol violation. Implementations that do not have a DTLS endpoint immediately ready on SCTP handshake completion will have to ensure correct caching of the messages until the DTLS endpoint is ready.

Whenever a mutual authentication, updated security parameters, rerun of Diffie-Hellman key-exchange, or SCTP-AUTH rekeying is needed, a new DTLS connection is instead setup in parallel with the old

connection (i.e., there may be up to two simultaneous DTLS connections within one association).

4.3. Payload Protocol Identifier Usage

SCTP Payload Protocol Identifiers are assigned by IANA. Application protocols using DTLS over SCTP SHOULD register and use a separate Payload Protocol Identifier (PPID) and SHOULD NOT reuse the PPID that they registered for running directly over SCTP.

Using the same PPID does no harm as DTLS/SCTP requires all user messages being DTLS protected and knows that DTLS is used. However, for protocol analyzers, for example, it is much easier if a separate PPID is used and avoids different behavior from [[RFC6083](#)].

Messages that are exchanged between DTLS/SCTP peers not containing ULP user messages shall use PPID = 0 according to section 3.3.1 of [[RFC9260](#)] as no application identifier can be specified by the upper layer for this payload data. With the exception for the DTLS/SCTP Control Messages ([Section 5](#)) that uses its own PPID.

4.4. Stream Usage

DTLS 1.3 protects the actual content type of the DTLS record and have therefore omitted the non-protected content type field. Thus, it is not possible to determine which content type the DTLS record has on SCTP level. For DTLS 1.2 ULP user messages will be carried in DTLS records with content type "application_data".

DTLS Records carrying protected user message fragments MUST be sent in the by ULP indicated SCTP stream and user message. The ULP has no limitations in using SCTP facilities for stream and user messages. DTLS records of other types MAY be sent on any stream. It MAY also be sent in its own SCTP user message as well as interleaved with other DTLS records carrying protected user messages. Thus, it is allowed to insert between protected user message fragments DTLS records of other types as the DTLS receiver will process these and not result in any user message data being inserted into the ULP's user message. However, DTLS messages of other types than protected user message MUST be sent reliable, so the DTLS record can only be interleaved in case the ULP user message is sent as reliable.

DTLS is capable of handling reordering of the DTLS records. However, depending on stream properties and which user message DTLS records of other types are sent in may impact in which order and how quickly they are possible to process. Using the same stream with in-order delivery for the different messages will ensure that the DTLS Records are delivered in the order they are sent in user messages. Thus, ensuring that if there are DTLS records that need to be delivered in particular order it can be ensured. Alternatively, if

it is desired that a DTLS record is delivered as early as possible avoiding in-order streams with queued messages and considering stream priorities can result in faster delivery.

A simple solution avoiding any protocol issue are to send all DTLS messages that are not protected user message fragments is to pick a stream not used by the ULP, send the DTLS messages in their own user messages with in order delivery. That mimics the RFC 6083 behavior without impacting the ULP. However, it assumes that there are available streams to be used based on the SCTP association handshake allowed streams (Section 5.1.1 of [[RFC9260](#)]).

4.5. Chunk Handling

DATA chunks of SCTP MUST be sent in an authenticated way as described in SCTP-AUTH [[RFC4895](#)]. All other chunks that can be authenticated, i.e., all chunk types that can be listed in the Chunk List Parameter [[RFC4895](#)], MUST also be sent in an authenticated way. This makes sure that an attacker cannot modify the stream in which a message is sent or affect the ordered/unordered delivery of the message.

If PR-SCTP as defined in [[RFC3758](#)] is used, FORWARD-TSN chunks MUST also be sent in an authenticated way as described in [[RFC4895](#)]. This makes sure that it is not possible for an attacker to drop messages and use forged FORWARD-TSN, SACK, and/or SHUTDOWN chunks to hide this dropping.

I-DATA chunk type as defined in [[RFC8260](#)] is RECOMMENDED to be supported to avoid some of the down sides that large user messages have on blocking transmission of later arriving high priority user messages. However, the support is not mandated and negotiated independently from DTLS/SCTP. If I-DATA chunks are used, then they MUST be sent in an authenticated way as described in [[RFC4895](#)].

4.6. SCTP-AUTH Hash Function

When using DTLS/SCTP, the SHA-256 Message Digest Algorithm MUST be supported in the SCTP-AUTH [[RFC4895](#)] implementation. SHA-1 MUST NOT be used when using DTLS/SCTP. [[RFC4895](#)] requires support and inclusion of SHA-1 in the HMAC-ALGO parameter, thus, to meet both requirements the HMAC-ALGO parameter will include both SHA-256 and SHA-1 with SHA-256 listed prior to SHA-1 to indicate the preference.

4.7. Parallel DTLS connections

To enable SCTP-AUTH rekeying, periodic authentication of both endpoints, and force attackers to dynamic key extraction [[RFC7624](#)], DTLS/SCTP per this specification defines the usage of parallel DTLS connections over the same SCTP association. This solution ensures

that there are no limitations to the lifetime of the SCTP association due to DTLS, it also avoids dependency on version specific DTLS mechanisms such as renegotiation in DTLS 1.2, which is disabled by default in many DTLS implementations, or post-handshake messages in DTLS 1.3, which does not allow periodic mutual endpoint re-authentication or re-keying of SCTP-AUTH. Parallel DTLS connections enable opening a new DTLS connection performing a handshake, while the existing DTLS connection is kept in place. In DTLS 1.3 the handshake MAY be a full handshake or a resumption handshake and resumption can be done while the original connection is still open. In DTLS 1.2 the handshake MUST be a full handshake. The new parallel connection MUST use the same DTLS version as the existing connection. On handshake completion switch to the security context of the new DTLS connection for protection of user message and then ensure delivery of all the SCTP chunks using the old DTLS connections security context. When that has been achieved close the old DTLS connection and discard the related security context.

As specified in [Section 4.1](#) the usage of DTLS connection ID is required to ensure that the receiver can correctly identify the DTLS connection and its security context when performing its de-protection operations. There is also only a single SCTP-AUTH key exported per DTLS connection ensuring that there is clear mapping between the DTLS connection ID and the SCTP-AUTH security context for each key-id.

Application writers should be aware that establishing a new DTLS connections may result in changes of security parameters. See [Section 9](#) for security considerations regarding rekeying.

A DTLS/SCTP Endpoint MUST NOT have more than two DTLS connections open at the same time. Either of the endpoints MAY initiate a new DTLS connection by performing a full DTLS handshake. As either endpoint can initiate a DTLS handshake on either side at the same time, either endpoint may receive a DTLS ClientHello when it has sent its own ClientHello. In this case the ClientHello from the endpoint that had the DTLS Client role in the establishment of the existing DTLS connection shall be continued to be processed and the other dropped.

When performing the DTLS handshake the endpoint MUST verify that the peer identifies using the same identity as in the previous DTLS connection.

When the DTLS handshake has been completed, the new DTLS connection MUST be used for the DTLS protection of any new ULP user messages, and SHOULD be switched to for protection of not yet protected user message fragments of partially transmitted user messages. Also after the completion of the DTLS handshake a new SCTP-AUTH key will be

exported per [Section 4.9](#). To enable the sender and receiver to correctly identify when the old DTLS connection is no longer in use, the SCTP-AUTH key used to protect a SCTP packet MUST NOT be from a newer DTLS connection than produced any included DTLS record fragment.

The SCTP API defined in [[RFC6458](#)] has limitation in changing the SCTP-AUTH key until the whole SCTP user message has been delivered. However, the DTLS/SCTP implementation can switch the DTLS connection used to protect the user message fragments to a newer, even if the older DTLS connections exported key is used for the SCTP-AUTH. And for SCTP implementations where the SCTP-AUTH key can be switched in the middle of a user message the SCTP-AUTH key should be changed as soon as all DTLS record fragments included in an SCTP packet have been protected by the newer DTLS connection. Any SCTP-AUTH receiver implementation is expected to be able to select key on SCTP packet basis.

The DTLS/SCTP endpoint timely indicates to its peer when the previous DTLS connection and its context are no longer needed for receiving any more data from this endpoint. This is done by sending a DTLS/SCTP Control Message [Section 5](#) of type "Ready_To_Close" [Section 5.2](#) to its peer. The endpoint MUST NOT send the Ready_To_Close until the following two conditions are fulfilled:

1. All SCTP packets containing part of any DTLS record or message protected using the security context of this DTLS connection have been acknowledged in a non-renegable way.
2. All SCTP packets using the SCTP-AUTH key associated with the security context of this DTLS connection have been acknowledged in a non-renegable way.

A DTLS/SCTP endpoint that fulfills the above conditions for the SCTP packets it sends and have received a Ready_To_Close message SHALL immediately initiate closing of this DTLS connection by sending a DTLS close_notify. Then when it have received the peer's close_notify terminate the DTLS connection and expunge the associated security context and SCTP-AUTH key. Note that it is not required for a DTLS/SCTP implementation that has received a Ready_To_Close message to send that message itself when it fulfills the conditions. However, in some situation both endpoints will fulfill the conditions close enough in time that both endpoints will send its Ready_To_Close prior to receiving the indication from its peer, that works as both endpoints will then initiate DTLS close_notify and terminate the DTLS connections upon the reception of the peers close_notify.

SCTP implementations exposing APIs like [\[RFC6458\]](#) fulfilling these conditions require draining the SCTP association of all outstanding data after having completed all the user messages using the previous SCTP-AUTH key identifier. Relying on the SCTP_SENDER_DRY_EVENT to know when delivery has been accomplished. A richer API could also be used that allows user message level tracking of delivery, see [Section 7](#) for API considerations.

For SCTP implementations exposing APIs like [\[RFC6458\]](#) where it is not possible to change the SCTP-AUTH key for a partial SCTP message initiated before the change of security context will be forced to track the SCTP messages and determine when all using the old security context has been transmitted. This maybe be impossible to do completely reliable without tighter integration between the DTLS/SCTP layer and the SCTP implementation. This type of implementations also has an implicit limitation in how large SCTP messages it can support. Each SCTP message needs have completed delivery and enabling closing of the previous DTLS connection prior to the need to create yet another DTLS connection. Thus, SCTP messages can't be larger than that the transmission completes in less than the duration between the rekeying or re-authentications needed for this SCTP association.

The consequences of sending a DTLS close_notify alert in the old DTLS connection prior to the receiver having received the data can result in failure case 1 described in [Section 4.1](#), which likely result in SCTP association termination.

4.8. Renegotiation and KeyUpdate

DTLS 1.2 renegotiation enables rekeying (with ephemeral Diffie-Hellman) of DTLS as well as mutual reauthentication and transfer of revocation information inside an DTLS 1.2 connection. Renegotiation has been removed from DTLS 1.3 and partly replaced with post-handshake messages such as KeyUpdate. The parallel DTLS connection solution was specified due to lack of necessary features with DTLS 1.3 considered needed for long lived SCTP associations, such as rekeying (with ephemeral Diffie-Hellman) as well as mutual reauthentication.

This specification does not allow usage of DTLS 1.2 renegotiation to avoid race conditions and corner cases in the interaction between the parallel DTLS connection mechanism and the keying of SCTP-AUTH. In addition renegotiation is also disabled in some implementations, as well as dealing with the epoch change reliable have similar or worse application impact.

This specification also recommends against using DTLS 1.3 KeyUpdate and instead rely on parallel DTLS connections. For DTLS 1.3 there

isn't feature parity. It also has the issue that a DTLS implementation following the RFC may assume a too limited window for SCTP where the previous epoch's security context is maintained and thus changes to epoch handling would be necessary. Thus, unless the below specified more application impacting draining is used there exist risk of losing data that the sender will have assumed has been reliably delivered.

4.8.1. DTLS 1.2 Considerations

The endpoint MUST NOT use DTLS 1.2 renegotiation.

4.8.2. DTLS 1.3 Considerations

Before sending a KeyUpdate message, the DTLS endpoint MUST ensure that all DTLS messages have been acknowledged by the SCTP peer in a non-revokable way. After sending the KeyUpdate message, it stops sending DTLS messages until the corresponding Ack message has been processed.

Prior to processing a received KeyUpdate message, all other received SCTP user messages that are buffered in the SCTP layer and can be delivered to the DTLS layer MUST be read and processed by DTLS.

4.9. Handling of Endpoint-Pair Shared Secrets

SCTP-AUTH [[RFC4895](#)] is keyed using Endpoint-Pair Shared Secrets. In SCTP associations where DTLS is used, DTLS is used to establish these secrets. The endpoints MUST NOT use another mechanism for establishing shared secrets for SCTP-AUTH. The endpoint-pair shared secret for Shared Key Identifier 0 is empty and MUST be used when establishing the first DTLS connection.

The initial DTLS connection will be used to establish a new shared secret as specified per DTLS version below, and which MUST use shared key identifier 1. After sending the DTLS Finished message for the initial DTLS connection, the active SCTP-AUTH key MUST be switched from key identifier 0 to key identifier 1. Once the initial Finished message from the peer has been processed by DTLS, the SCTP-AUTH key with Shared Key Identifier 0 MUST be removed.

When a subsequent DTLS connection is setup, a new a 64-byte shared secret is derived using the TLS-Exporter. The shared secret identifiers form a sequence. If the previous shared secret used Shared Key Identifier n , the new one MUST use Shared Key Identifier $n+1$, unless $n = 65535$, in which case the new Shared Key Identifier is 1.

After sending the DTLS Finished message, the new SCTP-AUTH key can be used according to [Section 4.7](#). When the endpoint has both sent

and received a `close_notify` on the old DTLS connection then the endpoint SHALL remove the shared secret and the SCTP-AUTH key related to old DTLS connection.

4.9.1. DTLS 1.2 Considerations

Whenever a new DTLS connection is established, a 64-byte endpoint-pair shared secret is derived using the TLS-Exporter described in [\[RFC5705\]](#).

The 64-byte shared secret MUST be provided to the SCTP stack as soon as the computation is possible. The exporter MUST use the label given in [Section 8](#) and no context.

4.9.2. DTLS 1.3 Considerations

When the `exporter_secret` can be computed, a 64-byte shared secret is derived from it and provided as a new endpoint-pair shared secret by using the TLS-Exporter described in [\[RFC8446\]](#).

The 64-byte shared secret MUST be provided to the SCTP stack as soon as the computation is possible. The exporter MUST use the label given in [Section 8](#) and no context.

4.10. Shutdown

To prevent DTLS from discarding DTLS user messages while it is shutting down, the below procedure has been defined. Its goal is to avoid the need for APIs requiring per user message data level acknowledgments and utilizes existing SCTP protocol behavior to ensure delivery of the protected user messages data.

To support DTLS 1.2 `close_notify` behavior and avoid any uncertainty related to rekeying, a DTLS/SCTP protocol message ([Section 5](#)) sent as protected SCTP user message is defined with its own PPID to enable the DTLS/SCTP layer to know that it is targeting the remote DTLS/SCTP function and act on the request to close in a controlled fashion.

The interaction between peers (local and remote) and protocol stacks is as follows:

1. Local instance of ULP asks for terminating the DTLS/SCTP Association.
2. Local DTLS/SCTP acknowledge the request, from this time on no new data from local instance of ULP will be accepted. In case a DTLS connection handshake is ongoing this needs to be aborted conclusively at this step to ensure that the necessary DTLS

message exchange happens prior to draining any outstanding data in the SCTP association from this endpoint.

3. Local DTLS/SCTP finishes any protection operation on buffered user messages and ensures that all protected user message data has been successfully transferred to the remote peer.
4. Local DTLS/SCTP sends a DTLS/SCTP Control Message [Section 5](#) of type "SHUTDOWN_Request" [Section 5.1](#) to its peer.
5. The remote DTLS/SCTP when receiving the SHUTDOWN-Request informs its ULP that shutdown has been initiated. No more ULP user message data to be sent to peer can be accepted by DTLS/SCTP. In case this endpoint has initiated a DTLS connection handshake this MUST be aborted as the peer is unable to respond to avoid additional case of draining.
6. Remote DTLS/SCTP finishes any protection operation on buffered user messages and ensures that all protected user message data has been successfully transferred to the remote ULP.
7. Remote DTLS/SCTP sends DTLS close_notify to Local DTLS/SCTP for each and all DTLS connections. Then it initiates the SCTP shutdown procedure (section 9.2 of [[RFC9260](#)]).
8. When the local DTLS/SCTP receives a close_notify on a DTLS connection, in case it is DTLS 1.3 it SHALL send its corresponding DTLS close_notify on each open DTLS connection. When the last open DTLS connection has received close_notify and any if needed corresponding close_notify have been sent the local DTLS/SCTP initiates the SCTP shutdown procedure (section 9.2 of [[RFC9260](#)]).
9. Upon receiving the information that SCTP has closed the Association, independently the local and remote DTLS/SCTP entities destroy the DTLS connection completing the shutdown.

The verification in step 3 and 6 that all user data message has been successfully delivered to the remote ULP can be provided by the SCTP stack that implements [[RFC6458](#)] by means of SCTP_SENDER_DRY event (section 6.1.9 of [[RFC6458](#)]).

A successful SCTP shutdown will indicate successful delivery of all data. However, in cases of communication failures and extensive packet loss the SCTP shutdown procedure can time out and result in SCTP association termination where its unknown if all data has been delivered. The DTLS/SCTP should indicate to ULP successful completion or failure to shutdown gracefully.

5. DTLS/SCTP Control Message

The DTLS/SCTP Control Message is defined as its own upper layer protocol for DTLS/SCTP identified by its own PPID. The control message is single 32-bit unsigned integer value in network byte order. Each message is sent as its own SCTP user message after having been protected by an open DTLS connection on any SCTP stream and MUST be marked with SCTP Payload Protocol Identifier (PPID) value TBD1 [Section 8.3](#).

The DTLS/SCTP implementation MUST consume all SCTP messages received with the PPID value of TBD1. If the message is not 32-bit long the message MUST be discarded and the error SHOULD be logged. In case the message has an unknown value the message is discarded and the event SHOULD be logged.

Two control messages are defined in this specification.

5.1. SHUTDOWN-Request

The value "1" is defined as a request to the peer to initiate controlled shutdown. This is used per step 4 and 5 in [Section 4.10](#).

5.2. Ready To Close Indication

The value "2" is defined as an indication to the peer that from its perspective all SCTP packets with user message or using the SCTP-AUTH key associated with the oldest DTLS connection has been sent and acknowledged as received in a non-renegable way. This is used per [Section 4.7](#) to initiate the closing of the DTLS connections during rekeying.

6. DTLS over SCTP Service

The adoption of DTLS over SCTP according to the current specification is meant to add to SCTP the option for transferring encrypted data. When DTLS over SCTP is used, all data being transferred MUST be protected by chunk authentication and DTLS encrypted. Chunks that need to be received in an authenticated way will be specified in the CHUNK list parameter according to [\[RFC4895\]](#). Error handling for authenticated chunks is according to [\[RFC4895\]](#).

6.1. Adaptation Layer Indication in INIT/INIT-ACK

At the initialization of the association, a sender of the INIT or INIT ACK chunk that intends to use DTLS/SCTP as specified in this specification MUST include an Adaptation Layer Indication Parameter [\[RFC5061\]](#) with the IANA assigned value TBD ([Section 8.2](#)) to inform

its peer that it is able to support DTLS over SCTP per this specification.

6.2. DTLS over SCTP Initialization

Initialization of DTLS/SCTP requires all the following options to be part of the INIT/INIT-ACK handshake:

RANDOM: defined in [[RFC4895](#)]

CHUNKS: defined in [[RFC4895](#)]

HMAC-ALGO: defined in [[RFC4895](#)]

ADAPTATION-LAYER-INDICATION: defined in [[RFC5061](#)]

When all the above options are present and having acceptable parameters, the Association will start with support of DTLS/SCTP. The set of options indicated are the DTLS/SCTP Mandatory Options. No data transfer is permitted before DTLS handshake is complete. Chunk bundling is permitted according to [[RFC9260](#)]. The DTLS handshake will enable authentication of both the peers.

The extension described in this document is given by the following message exchange.

```
--- INIT[RANDOM; CHUNKS; HMAC-ALGO; ADAPTATION-LAYER-IND] --->
<- INIT-ACK[RANDOM; CHUNKS; HMAC-ALGO; ADAPTATION-LAYER-IND] -
----- COOKIE-ECHO ----->
<----- COOKIE-ACK -----
----- AUTH; DATA[DTLS Handshake] ----->
...
...
<----- AUTH; DATA[DTLS Handshake] -----
```

6.3. Client Use Case

When a client initiates an SCTP Association with DTLS protection, i.e., the SCTP INIT containing DTLS/SCTP Mandatory Options, it can receive an INIT-ACK also containing DTLS/SCTP Mandatory Options, in that case the Association will proceed as specified in the previous [Section 6.2](#) section. If the peer replies with an INIT-ACK not containing all DTLS/SCTP Mandatory Options, the client SHOULD reply with an SCTP ABORT.

6.4. Server Use Case

If a SCTP Server supports DTLS/SCTP, i.e., per this specification, when receiving an INIT chunk with all DTLS/SCTP Mandatory Options it will reply with an INIT-ACK also containing all the DTLS/SCTP

Mandatory Options, following the sequence for DTLS initialization [Section 6.2](#) and the related traffic case. If a SCTP Server that supports DTLS and configured to use it, receives an INIT chunk without all DTLS/SCTP Mandatory Options, it SHOULD reply with an SCTP ABORT.

6.5. RFC 6083 Fallback

This section discusses how an endpoint supporting this specification can fallback to follow the DTLS/SCTP behavior in RFC6083. It is recommended to define a setting that represents the policy to allow fallback or not. However, the possibility to use fallback is based on the ULP can operate using user messages that are no longer than 16384 bytes and where the security issues can be mitigated or considered acceptable. Fallback is NOT RECOMMEND to be enabled as it enables downgrade attacks to weaker algorithms and versions of DTLS.

An SCTP endpoint that receives an INIT chunk or an INIT-ACK chunk that does not contain the SCTP-Adaptation-Indication parameter with the DTLS/SCTP adaptation layer codepoint, see [Section 8.2](#), may in certain cases potentially perform a fallback to RFC 6083 behavior. However, the fallback attempt should only be performed if policy says that is acceptable.

If fallback is allowed, it is possible that the client will send plain text user messages prior to DTLS handshake as it is allowed per RFC 6083. So that needs to be part of the consideration for a policy allowing fallback.

6.5.1. Client Fallback

A DTLS/SCTP client supporting this specification encountering an server not compatible with this specification MAY attempt RFC 6083 fallback per this procedure.

1. Fallback procedure, if enabled, is initiated when receiving an SCTP INIT-ACK that does not contain the DTLS/SCTP Adaptation Layer indication. If fallback is not enabled the SCTP handshake is aborted.
2. The client checks that the SCTP INIT-ACK contained the necessary chunks and parameters to establish SCTP-AUTH per RFC 6083 with this endpoint. If not all necessary parameters or support algorithms don't match the client MUST abort the handshake. Otherwise it completes the SCTP handshake.
3. Client performs DTLS connection handshake per RFC 6083 over established SCTP association. If successful authenticating the targeted server the client has successful fallen back to use RFC 6083. If not terminate the SCTP association.

6.5.2. Server Fallback

A DTLS/SCTP Server that supports both this specification and RFC 6083 and where fallback has been enabled for the ULP can follow this procedure.receiving

1. When receiving an SCTP INIT message without the DTLS/SCTP adaptation layer indication fallback procedure is initiated.
2. Verify that the SCTP INIT contains SCTP-AUTH parameters required by RFC 6083 and compatible with this server. If that is not the case abort the SCTP handshake.
3. Send an SCTP INIT ACK with the required SCTP-AUTH chunks and parameters to the client.
4. Complete the SCTP Handshake. Await DTLS handshake per RFC 6083. Plain text SCTP messages MAY be received.
5. Upon successful completion of DTLS handshake successful fallback to RFC 6083 have been accomplished.

7. SCTP API Consideration

DTLS/SCTP needs certain functionality on the API that the SCTP implementation provide to the ULP to function optimally. A DTLS/SCTP implementation will need to provide its own API to the ULP, while itself using the SCTP API. This discussion is focused on the needed functionality on the SCTP API.

The following functionality is needed:

- *Controlling SCPT-AUTH negotiation so that SHA-256 algorithm is included, and determine that SHA-1 is not selected when the association is established.
- *Determine when all SCTP packets that uses an SCTP-auth key or contains DTLS records associated to a particular DTLS connection has been acknowledged non-renegable.
- *Determine when all SCTP packets have been acknowledged non-renegable.
- *Negotiate the adaptation layer indication that indicates DTLS/SCTP and determine if it was agreed or not.
- *Partial user messages transmission and reception.

8. IANA Considerations

8.1. TLS Exporter Label

RFC 6083 defined a TLS Exporter Label registry as described in [RFC5705]. IANA is requested to update the reference for the label "EXPORTER_DTLS_OVER_SCTP" to also include this specification.

8.2. SCTP Adaptation Layer Indication Code Point

[RFC5061] defined an IANA registry for Adaptation Code Points to be used in the Adaptation Layer Indication parameter. The registry was at time of writing located: <https://www.iana.org/assignments/sctp-parameters/sctp-parameters.xhtml#sctp-parameters-27> IANA is requested to assign one Adaptation Code Point for DTLS/SCTP per the below proposed entry in [Table 1](#).

Code Point (32-bit number)	Description	Reference
0x00000002	DTLS/SCTP	[RFC-TBD]

Table 1: Adaptation Code Point

RFC-Editor Note: Please replace [RFC-TBD] with the RFC number given to this specification.

8.3. SCTP Payload Protocol Identifiers

This document registers one Payload Protocol Identifier (PPID) to be used to identify the DTLS/SCTP control messages ([Section 5](#)).

Value	SCTP PPID	Reference
TBD1	DTLS/SCTP Control Message	[RFC-TBD]

Table 2: SCTP Payload Protocol Identifier

RFC-Editor Note: Please replace [RFC-TBD] with the RFC number given to this specification.

9. Security Considerations

The security considerations given in [RFC9147], [RFC4895], and [RFC9260] also apply to this document.

9.1. Cryptographic Considerations

Over the years, there have been several serious attacks on earlier versions of Transport Layer Security (TLS), including attacks on its most commonly used ciphers and modes of operation. [RFC7457] summarizes the attacks that were known at the time of publishing and

BCP 195 [[RFC7525](#)] [[RFC8996](#)] provide recommendations for improving the security of deployed services that use TLS.

When DTLS/SCTP is used with DTLS 1.2 [[RFC6347](#)], DTLS 1.2 MUST be configured to disable options known to provide insufficient security. HTTP/2 [[RFC9113](#)] gives good minimum requirements based on the attacks that were publicly known in 2022. DTLS 1.3 [[RFC9147](#)] only define strong algorithms without major weaknesses at the time of publication. Many of the TLS registries have a "Recommended" column. Parameters not marked as "Y" are NOT RECOMMENDED to support. DTLS 1.3 is preferred over DTLS 1.2 being a newer protocol that addresses known vulnerabilities and only defines strong algorithms without known major weaknesses at the time of publication.

DTLS 1.3 requires rekeying before algorithm specific AEAD limits have been reached. The AEAD limits equations are equally valid for DTLS 1.2 and SHOULD be followed for DTLS/SCTP, but are not mandated by the DTLS 1.2 specification.

HMAC-SHA-256 as used in SCTP-AUTH has a very large tag length and very good integrity properties. The SCTP-AUTH key can be used longer than the current algorithms in the TLS record layer. The SCTP-AUTH key is rekeyed when a new DTLS connection is set up at which point a new SCTP-AUTH key is derived using the TLS-Exporter.

(D)TLS 1.3 [[RFC8446](#)] discusses forward secrecy from EC(DHE), KeyUpdate, and tickets/resumption. Forward secrecy limits the effect of key leakage in one direction (compromise of a key at time T_2 does not compromise some key at time T_1 where $T_1 < T_2$). Protection in the other direction (compromise at time T_1 does not compromise keys at time T_2) can be achieved by rerunning EC(DHE). If a long-term authentication key has been compromised, a full handshake with EC(DHE) gives protection against passive attackers. If the resumption_master_secret has been compromised, a resumption handshake with EC(DHE) gives protection against passive attackers and a full handshake with EC(DHE) gives protection against active attackers. If a traffic secret has been compromised, any handshake with EC(DHE) gives protection against active attackers.

The document "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement" [[RFC7624](#)] defines key exfiltration as the transmission of cryptographic keying material for an encrypted communication from a collaborator, deliberately or unwittingly, to an attacker. Using the terms in RFC 7624, forward secrecy without rerunning EC(DHE) still allows an attacker to do static key exfiltration. Rerunning EC(DHE) forces an attacker to do dynamic key exfiltration (or content exfiltration).

When using DTLS 1.3 [[RFC9147](#)], AEAD limits and forward secrecy can be achieved by sending post-handshake KeyUpdate messages, which triggers rekeying of DTLS. Such symmetric rekeying gives significantly less protection against key leakage than re-running Diffie-Hellman as explained above. After leakage of application_traffic_secret_N, an attacker can passively eavesdrop on all future data sent on the connection including data encrypted with application_traffic_secret_N+1, application_traffic_secret_N+2, etc. Note that KeyUpdate does not update the exporter_secret.

DTLS/SCTP is in many deployments replacing IPsec. For IPsec, NIST (US), BSI (Germany), and ANSSI (France) recommends very frequent re-run of Diffie-Hellman to provide forward secrecy and force attackers to dynamic key extraction [[RFC7624](#)]. ANSSI writes "It is recommended to force the periodic renewal of the keys, e.g., every hour and every 100 GB of data, in order to limit the impact of a key compromise." [[ANSSI-DAT-NT-003](#)].

For many DTLS/SCTP deployments the SCTP association is expected to have a very long lifetime of months or even years. For associations with such long lifetimes there is a need to frequently re-authenticate both client and server. TLS Certificate lifetimes significantly shorter than a year are common which is shorter than many expected DTLS/SCTP associations.

SCTP-AUTH re-rekeying, periodic authentication of both endpoints, and frequent re-run of Diffie-Hellman to force attackers to dynamic key extraction is in DTLS/SCTP per this specification achieved by setting up new DTLS connections over the same SCTP association. Implementations SHOULD set up new connections frequently to force attackers to dynamic key extraction. Implementations MUST set up new connections before any of the certificates expire. It is RECOMMENDED that all negotiated and exchanged parameters are the same except for the timestamps in the certificates. Clients and servers MUST NOT accept a change of identity during the setup of a new connections, but MAY accept negotiation of stronger algorithms and security parameters, which might be motivated by new attacks.

Allowing new connections can enable denial-of-service attacks. The endpoints SHOULD limit the frequency of new connections.

When DTLS/SCTP is used with DTLS 1.2 [[RFC6347](#)], the TLS Session Hash and Extended Master Secret Extension [[RFC7627](#)] MUST be used to prevent unknown key-share attacks where an attacker establishes the same key on several connections. DTLS 1.3 always prevents these kinds of attacks. The use of SCTP-AUTH then cryptographically binds new connections to the old connection. This together with mandatory mutual authentication (on the DTLS layer) and a requirement to not

accept new identities mitigates MITM attacks that have plagued renegotiation [[TRISHAKE](#)].

9.2. Downgrade Attacks

A peer supporting DTLS/SCTP according to this specification, DTLS/SCTP according to [[RFC6083](#)] and/or SCTP without DTLS may be vulnerable to downgrade attacks where an on-path attacker interferes with the protocol setup to lower or disable security. If possible, it is RECOMMENDED that the peers have a policy only allowing DTLS/SCTP according to this specification.

9.3. Targeting DTLS Messages

The DTLS handshake messages and other control messages, i.e., not application data can easily be identified when using DTLS 1.2 as their content type is not encrypted. With DTLS 1.3 there is no unprotected content type. However, they will be sent with an PPID of 0 if sent in their own SCTP user messages. [Section 4.4](#) proposes a basic behavior that will still make it easy for anyone to detect the DTLS messages that are not protected user messages.

9.4. Authentication and Policy Decisions

DTLS/SCTP MUST be mutually authenticated. Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. DTLS only provides proof of possession of a key. DTLS/SCTP MUST perform identity authentication. It is RECOMMENDED that DTLS/SCTP is used with certificate-based authentication. When certificates are used the application using DTLS/SCTP is responsible for certificate policies, certificate chain validation, and identity authentication (HTTPS does for example match the hostname with a subjectAltName of type `dnsName`). The application using DTLS/SCTP MUST define what the identity is and how it is encoded and the client and server MUST use the same identity format. Guidance on server certificate validation can be found in [[RFC6125](#)]. DTLS/SCTP enables periodic transfer of mutual revocation information (OSCP stapling) every time a new parallel connection is set up. All security decisions MUST be based on the peer's authenticated identity, not on its transport layer identity.

It is possible to authenticate DTLS endpoints based on IP addresses in certificates. SCTP associations can use multiple IP addresses per SCTP endpoint. Therefore, it is possible that DTLS records will be sent from a different source IP address or to a different destination IP address than that originally authenticated. This is not a problem provided that no security decisions are made based on the source or destination IP addresses.

9.5. Resumption and Tickets

In DTLS 1.3 any number of tickets can be issued in a connection and the tickets can be used for resumption as long as they are valid, which is up to seven days. The nodes in a resumed connection have the same roles (client or server) as in the connection where the ticket was issued. In DTLS/SCTP, there are no significant performance benefits with resumption and an implementation can choose to never issue any tickets. If tickets and resumption are used it is enough to issue a single ticket per connection.

9.6. Privacy Considerations

[[RFC6973](#)] suggests that the privacy considerations of IETF protocols be documented.

For each SCTP user message, the user also provides a stream identifier, a flag to indicate whether the message is sent ordered or unordered, and a payload protocol identifier. Although DTLS/SCTP provides privacy for the actual user message, the other three information fields are not confidentiality protected. They are sent as cleartext because they are part of the SCTP DATA chunk header.

It is RECOMMENDED that DTLS/SCTP is used with certificate based authentication in DTLS 1.3 [[RFC9147](#)] to provide identity protection. DTLS/SCTP MUST be used with a key exchange method providing forward secrecy.

9.7. Pervasive Monitoring

As required by [[RFC7258](#)], work on IETF protocols needs to consider the effects of pervasive monitoring and mitigate them when possible.

Pervasive Monitoring is widespread surveillance of users. By encrypting more information including user identities, DTLS 1.3 offers much better protection against pervasive monitoring.

Massive pervasive monitoring attacks relying on key exchange without forward secrecy has been reported. By mandating forward secrecy, DTLS/SCTP effectively mitigate many forms of passive pervasive monitoring and limits the amount of compromised data due to key compromise.

An important mitigation of pervasive monitoring is to force attackers to do dynamic key exfiltration instead of static key exfiltration. Dynamic key exfiltration increases the risk of discovery for the attacker [[RFC7624](#)]. DTLS/SCTP per this specification encourages implementations to frequently set up new DTLS connections with (EC)DHE over the same SCTP association to force attackers to do dynamic key exfiltration.

In addition to the privacy attacks discussed above, surveillance on a large scale may enable tracking of a user over a wider geographical area and across different access networks. Using information from DTLS/SCTP together with information gathered from other protocols increase the risk of identifying individual users.

10. Contributors

Michael Tuexen contributed as co-author to the initial versions of this draft. Michael's contributions include:

- *The use of the Adaptation Layer Indication.

- *Many editorial improvements.

11. Acknowledgments

The authors of RFC 6083 which this document is based on are Michael Tuexen, Eric Rescorla, and Robin Seggelmann.

The RFC 6083 authors thanked Anna Brunstrom, Lars Eggert, Goran Fairhurst, Ian Goldberg, Alfred Hoenes, Carsten Hohendorf, Stefan Lindskog, Daniel Mentz, and Sean Turner for their invaluable comments.

The authors of this document want to thank Daria Ivanova, Li Yan, and GitHub user vanrein for their contribution.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <<https://www.rfc-editor.org/info/rfc3758>>.
- [RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <<https://www.rfc-editor.org/info/rfc4895>>.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP)

Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <<https://www.rfc-editor.org/info/rfc5061>>.

- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <<https://www.rfc-editor.org/info/rfc5705>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <<https://www.rfc-editor.org/info/rfc7627>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8260] Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol", RFC 8260, DOI 10.17487/RFC8260, November 2017, <<https://www.rfc-editor.org/info/rfc8260>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <<https://www.rfc-editor.org/info/rfc8996>>.
- [RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <<https://www.rfc-editor.org/info/rfc9113>>.
- [RFC9146] Rescorla, E., Ed., Tschofenig, H., Ed., Fossati, T., and A. Kraus, "Connection Identifier for DTLS 1.2", RFC 9146, DOI 10.17487/RFC9146, March 2022, <<https://www.rfc-editor.org/info/rfc9146>>.
- [RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <<https://www.rfc-editor.org/info/rfc9147>>.

[RFC9260]

Stewart, R., Tüxen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <<https://www.rfc-editor.org/info/rfc9260>>.

12.2. Informative References

[RFC3436]

Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, DOI 10.17487/RFC3436, December 2002, <<https://www.rfc-editor.org/info/rfc3436>>.

[RFC3788]

Loughney, J., Tuexen, M., Ed., and J. Pastor-Balbas, "Security Considerations for Signaling Transport (SIGTRAN) Protocols", RFC 3788, DOI 10.17487/RFC3788, June 2004, <<https://www.rfc-editor.org/info/rfc3788>>.

[RFC6083]

Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, DOI 10.17487/RFC6083, January 2011, <<https://www.rfc-editor.org/info/rfc6083>>.

[RFC6125]

Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.

[RFC6458]

Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <<https://www.rfc-editor.org/info/rfc6458>>.

[RFC6973]

Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.

[RFC7258]

Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.

[RFC7457]

Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <<https://www.rfc-editor.org/info/rfc7457>>.

[RFC7525]

Sheffer, Y., Holz, R., and P. Saint-Andre,
"Recommendations for Secure Use of Transport Layer
Security (TLS) and Datagram Transport Layer Security
(DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
2015, <<https://www.rfc-editor.org/info/rfc7525>>.

[RFC7624]

Barnes, R., Schneier, B., Jennings, C., Hardie, T.,
Trammell, B., Huitema, C., and D. Borkmann,
"Confidentiality in the Face of Pervasive Surveillance: A
Threat Model and Problem Statement", RFC 7624, DOI
10.17487/RFC7624, August 2015, <<https://www.rfc-editor.org/info/rfc7624>>.

[ANSSI-DAT-NT-003] Agence nationale de la sécurité des systèmes
d'information, "Recommendations for securing networks
with IPsec", ANSSI Technical Report DAT-NT-003 , August
2015, <https://www.ssi.gouv.fr/uploads/2015/09/NT_IPsec_EN.pdf>.

[TRISHAKE] Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Pironti,
A., and P. Strub, "Triple Handshakes and Cookie Cutters:
Breaking and Fixing Authentication over TLS", IEEE
Symposium on Security & Privacy , April 2016, <<https://hal.inria.fr/hal-01102259/file/triple-handshakes-and-cookie-cutters-oakland14.pdf>>.

Appendix A. Motivation for Changes

This document proposes a number of changes to RFC 6083 that have various different motivations:

Supporting Large User Messages: RFC 6083 allowed only user messages that could fit within a single DTLS record. 3GPP has run into this limitation where they have at least four SCTP using protocols (F1, E1, Xn, NG-C) that can potentially generate messages over the size of 16384 bytes.

New Versions: Almost 10 years has passed since RFC 6083 was written, and significant evolution has happened in the area of DTLS and security algorithms. Thus, DTLS 1.3 is the newest version of DTLS and also the SHA-1 HMAC algorithm of RFC 4895 is getting towards the end of usefulness. Use of DTLS 1.3 with long lived associations require parallel DTLS connections. Thus, this document mandates usage of relevant versions and algorithms.

Allowing DTLS Messages on any stream: RFC6083 requires DTLS messages that are not user message data to be sent on stream 0 and that this stream is used with in-order delivery. That can actually limit the applications that can use DTLS/SCTP. In addition with DTLS 1.3

encrypting the actual message type it is anyway not available.
Therefore a more flexible rule set is used that relies on DTLS
handling reordering.

Clarifications: Some implementation experiences have been gained
that motivates additional clarifications on the specification.

*Avoid unsecured messages prior to DTLS handshake have completed.

*Make clear that all messages are encrypted after DTLS handshake.

Authors' Addresses

Magnus Westerlund
Ericsson

Email: magnus.westerlund@ericsson.com

John Preuß Mattsson
Ericsson

Email: john.mattsson@ericsson.com

Claudio Porfiri
Ericsson

Email: claudio.porfiri@ericsson.com