TSVWG
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

M. Westerlund
J. Preu Mattsson
C. Porfiri
Ericsson

7 July 2025

Datagram Transport Layer Security (DTLS) based key-management of the Stream Control Transmission Protocol (SCTP) DTLS Chunk
draft-westerlund-tsvwg-sctp-dtls-handshake-05

Abstract

This document defines a key-management solution based on Datagram Transport Layer Security 1.3 (DTLS) to protect the content of Stream Control Transmission Protocol (SCTP) packets using the packet protection framework provided by the SCTP DTLS chunk. The combination provides encryption, source authentication, integrity and replay protection for the SCTP association with in-band DTLS based key-management and mutual authentication of the peers. The specification is enabling very long-lived sessions of weeks and months and supports mutual re-authentication and rekeying with ephemeral key exchange. The key-management solution does not require any additional defined features or implementation support beyond core DTLS 1.3. This is intended as a replacement to using DTLS/SCTP (RFC6083) and SCTP-AUTH (RFC4895).

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-westerlund-tsvwg-sctp-dtls-handshake/.

Discussion of this document takes place on the Transport Area Working Group (tsvwg) Working Group mailing list (mailto:tsvwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tsvwg/. Subscribe at https://www.ietf.org/mailman/listinfo/tsvwg/.

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

Status of This Memo

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

This Internet-Draft will expire on 8 January 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

This document describes the usage of the Datagram Transport Layer Security version 1.3 (DTLS) [RFC9147] protocol for key-management of the SCTP DTLS Chunk packet protection [I-D.westerlund-tsvwg-sctp-dtls-chunk] securing Stream Control Transmission Protocol (SCTP) [RFC9260]. This combination of specifications is intended as a replacement to DTLS/SCTP [RFC6083] and usage of SCTP-AUTH [RFC4895]. The combination of SCTP DTLS Chunk and the key-management defined in this document we refer to as DTLS in SCTP.

DTLS in SCTP provides mutual authentication of endpoints, data confidentiality, data origin authentication, data integrity protection, and data replay protection of SCTP packets. Ensuring these security services to the application and its upper layer protocol over SCTP. Thus, it allows client/server applications to communicate in a way that is designed with communications privacy and preventing eavesdropping and detect tampering or message forgery.

Applications using DTLS in SCTP can use all currently existing transport features provided by SCTP and its extensions, in some cases with some limitations, as specified in [I-D.westerlund-tsvwg-sctp-dtls-chunk]. DTLS in SCTP supports:

* preservation of message boundaries.

* no limitation on number of unidirectional and bidirectional streams.

* ordered and unordered delivery of SCTP user messages.

* the partial reliability extension as defined in [RFC3758].

* multi-homing of the SCTP association per [RFC9260].

* the dynamic address reconfiguration extension as defined in [RFC5061] (Limitations apply).

* User messages of any size.

* SCTP Packets with a protected set of chunks up to a size of 2^14 (16384) bytes.

The main benefit of this key-management solution over the solution proposed by the WG is that this does not require any extensions to DTLS 1.3 to be implemented. It solely relies on the core DTLS handshake to do mutual authentication, creates a main secret, and then relies on the TLS exporter to export necessary secrets for the DTLS Chunk.

1.2. Protocol Overview

DTLS in SCTP is a key management specification for the SCTP DTLS 1.3 chunk [I-D.westerlund-tsvwg-sctp-dtls-chunk] that together utilizes DTLS 1.3 for the security functions like key exchange, authentication, encryption, integrity protection, and replay protection. All key management message exchange happens inband over the SCTP assocation.

In this document we use the terms DTLS Key context for indicating the pair of keys, derived from a DTLS connection, and all relevant data that needs to be provided to the SCTP DTLS Chunk Protection Operator for DTLS encryption and decryption. DTLS Key context includes Keys for sending and receiving, replay window, and last used sequence number. Each DTLS key context is associated with a four value tuple identifying the context, consisting of SCTP Association, the restart indicator, the DTLS Connection ID (if used), and the DTLS epoch.

The basic functionalities and how things are related is described below.

The process starts with a SCTP association where DTLS 1.3 Chunk usage has been negotiated and this key-management method has been agreed in the SCTP INIT and INIT-ACK. To initialize and authenticate the peer the DTLS handshake is exchanged as SCTP user messages with the DTLS Chunk Key-Management Messages PPID (see section 10.6 of [I-D.westerlund-tsvwg-sctp-dtls-chunk]) until an initial DTLS connection has been established. If the DTLS handshake fails, the SCTP association is aborted. With successful handshake and authentication of the peer the key material exported from the DTLS connection and configured for the DTLS 1.3 chunk. From that point until SCTP association termination the DTLS chunk will protect the SCTP packets. When the DTLS connection has been established and the DTLS Chunk configured with DTLS Key context the PVALID message is exchanged to verify that no downgrade attack between any offered protection solutions has occurred. To prevent manipulation, the PVALID message are sent as SCTP user messages (using DATA chunks) encapsulated in DTLS chunks.

Assuming that the PVALID validation is successful the SCTP association is established and the Upper Layer Protocol (ULP) can start sending messages over the SCTP association. All chunks are protected by encapsulating them in DTLS chunks as defined in [I-D.westerlund-tsvwg-sctp-dtls-chunk]. Using the current DTLS Key context the DTLS Chunk Protection operator protects the plain text, which is all chunks to be sent in one SCTP packet, producing a DTLS Record that is encapsulated in the DTLS chunk and then transmitted as a SCTP packet with a common header.

The DTLS Chunk specifies that in the receiving SCTP endpoint each incoming SCTP packet on any of its interfaces and ports are matched to the SCTP association based on ports and VTAG in the common header. Using the indicated DTLS Key context(s) for that SCTP association the content of the DTLS chunk is attempted to be processed, including replay protection, decryption, and integrity checking. And if decryption and integrity verification was successful the produced plain text of one or more SCTP chunks are provided for normal SCTP

processing in the identified SCTP association along with associated per-packet meta data such as path received on, original packet size, and ECN bits.

When mutual re-authentication or rekeying is needed or desired by either endpoint a new DTLS connection handshake is performed between the SCTP endpoints and a new DTLS Key context is created. When the handshake has completed the DTLS in SCTP implementation can simply switch to use the new DTLS Key contexts in the DTLS chunk. All rekeying will be using ephemeral key exchange and shall not use the DTLS Key-Update mechanism to avoid confusion about the properties of the DTLS Key Contexts for the DTLS chunk. After a short while (no longer than 2 min) to enable any outstanding packets to drain from the network path between the endpoints, the old key-management DTLS connection is closed and that signal that the corresponding key context can be deleted from the DTLS chunk's key store.

The DTLS connection may send alerts, handshake messages, or other non-application data to its peer at any point in time. All DTLS message will be sent by means of SCTP user messages with the DTLS Chunk Key-Management Messages PPID as specified in [I-D.westerlund-tsvwg-sctp-dtls-chunk]. However, only the DTLS close_notify is expected to be used after the handshake has been completed in this solution.

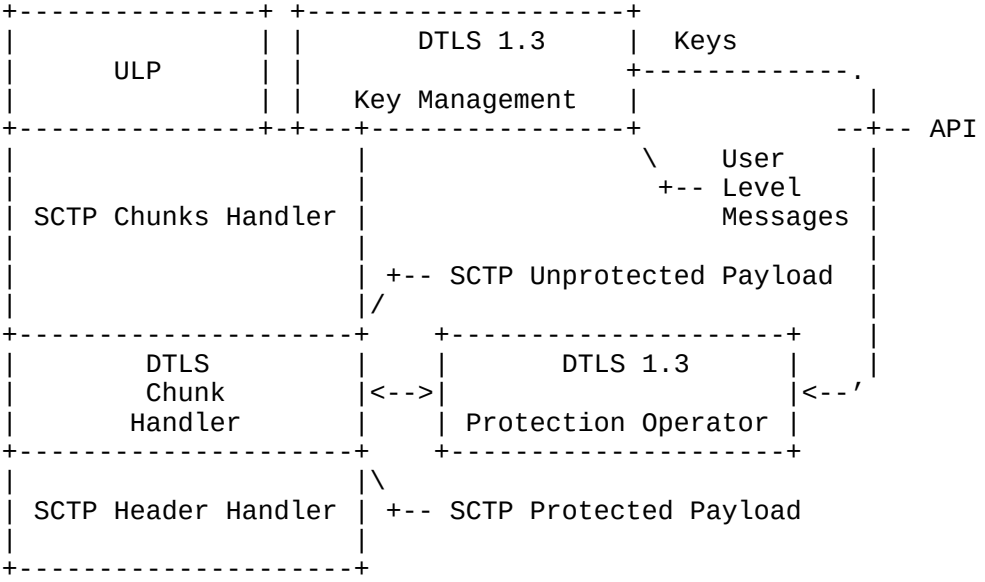

Figure 1: DTLS in SCTP layer in regard to SCTP and upper layer protocol

1.3. Properties of DTLS in SCTP

DTLS in SCTP (as the combination of the DTLS chunk and the in-band authentication and key-management using DTLS handshakes defined in this document) has a number of properties that are attractive.

* Provides confidentiality, integrity protection, and source authentication for each SCTP packet.

* Provides replay protection on SCTP packet level preventing malicious replay attacks on SCTP, both protecting the data as well as the SCTP functions themselves.

* Provides mutual authentication of the endpoints based on any authentication mechanism supported by DTLS.

* Uses parallel DTLS connections to enable mutual re-authentication and rekeying with ephemeral key-exchange. Thus, enabling SCTP association lifetimes without known limitations and without needing to drain the SCTP association.

* Uses core of DTLS as it is and updates and fixes to DTLS security properties; can be implemented without further changes to this specification.

* No reliance on DTLS implementation used for key-management having to support any features beyond core DTLS specification and the TLS exporter.

* Secures all SCTP packets exchanged after SCTP association has reached the established state and the initial key-exchange has completed. Making targeted attacks against the SCTP protocol and implementation much harder.

* DTLS in SCTP results in no limitations on user message transmission or message sizes, those properties are the same as for an unprotected SCTP association.

* Limited overhead on a per packet basis, with 4 bytes for the DTLS chunk plus the DTLS record overhead. The DTLS overhead is dependent on the DTLS version and cipher suit.

* Support of SCTP packet plain text payload sizes up to 2^14 bytes.

1.4. Terminology

This document uses the following terms:

Association: An SCTP association.

Connection: A DTLS connection.

DTLS Key context: Keys, derived from a DTLS connection, and all relevant data that needs to be provided to the SCTP DTLS Chunk. Each DTLS key context is associated with a four value tuple identifying the context, consisting of SCTP Association, the restart indicator, the DTLS Connection ID (if used), and the DTLS epoch

Restart DTLS Key context: A DTLS Key context to be used for an SCTP Association Restart

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

Traffic DTLS Key context: A DTLS Key context used to protect the regular SCTP traffic, i.e. not a restart DTLS Key context.

1.5. Abbreviations

AEAD: Authenticated Encryption with Associated Data

DKC: DTLS Key Context

DTLS: Datagram Transport Layer Security

MTU: Maximum Transmission Unit

PPID: Payload Protocol Identifier

SCTP: Stream Control Transmission Protocol

ULP: Upper Layer Protocol

1.6. 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.

2. DTLS usage of DTLS Chunk

DTLS in SCTP uses the DTLS chunk as specified in [I-D.westerlund-tsvwg-sctp-dtls-chunk]. The chunk if just repeated here for the reader's convenience.

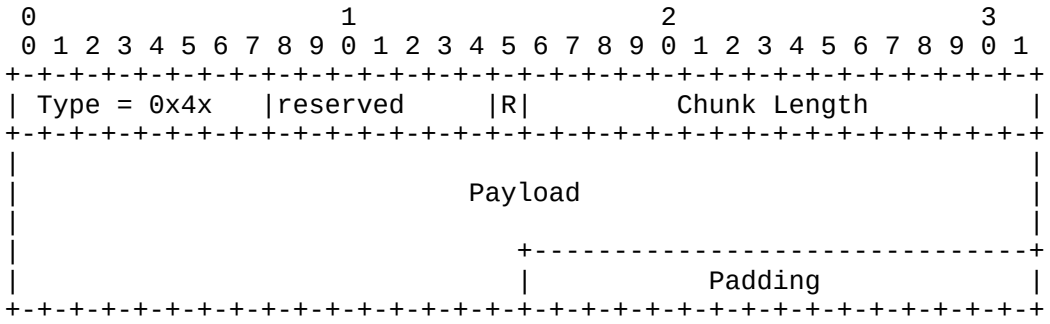

Figure 2: DTLS Chunk Structure

Type: 8 bits The Chunk Type indicate that this is the DTLS chunk.

reserved: 7 bits Reserved bits for future use. Sender MUST set these bits to 0 and MUST be ignored on reception.

R: 1 bit (boolean) Restart indicator. If this bit is set this DTLS chunk is protected with by a Restart DTLS Key context.

Chunk Length: 16 bits (unsigned integer) This value holds the length of the Payload in bytes plus 4.

Payload: variable length This holds the encrypted data as one DTLS 1.3 Record [RFC9147].

Padding: 0, 8, 16, or 24 bits To ensure the Chunk is whole 32-bit words long.

3. DTLS messages over SCTP User Messages

DTLS messages for the Handshake DTLS connection, i.e. that are not DTLS records containing protected SCTP chunk payloads, will be sent as SCTP user messages using the format defined below. A DTLS handshake message may be fragmented by DTLS to a set of DTLS records of a maximum configured fragment size. Each DTLS message fragment is sent as a SCTP user message on the same stream where each message is configured for reliable and in-order delivery with the PPID set to DTLS Chunk Key-Management Messages [I-D.westerlund-tsvwg-sctp-dtls-chunk]. These user messages MAY

contain one or more DTLS records. The SCTP stream ID used MAY be any stream ID that the ULP already uses, and if not know Stream 0. Note that all fragments of a handshake message MUST be sent with the same stream ID to ensure the in-order delivery.

The DTLS instance SHOULD NOT use DTLS retransmission to repair any packet losses of handshake message fragment. Note: If the DTLS implementation supports configuring a MTU larger than the actual IP MTU it MAY be used as SCTP provides reliability and fragmentation.

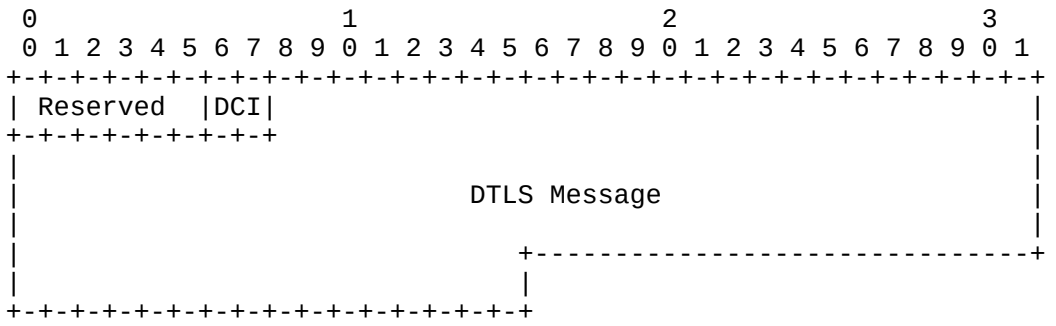

Figure 3: DTLS User Message Structure

Reserved: 6 bits Each bit MUST be set to zero (0) on transmission and MUST be ignored on reception. These bits MAY in the future be assigned a meaning when set to one (1).

DCI: DTLS Connection Index 2 bits (unsigned integer) DTLS Connection Index is the lower two bits of an DTLS Connection Index counter which corresponds to the epoch used in the SCTP DTLS Chunk. This is a counter implemented in DTLS in SCTP that is used to identify which DTLS connection instance that is capable of processing the DTLS message over a user message. This index is recommended to be the lower part of a 64-bit unsigned integer variable as this is how DTLS epoch counter is defined. DCI is unrelated to the DTLS Connection ID [RFC9147]. The counters initial value SHALL be three (3).

DTLS Message: variable length One or more DTLS records. In cases more than one DTLS record is included all DTLS records except the last MUST include a length field. Note that this matches what is specified in DTLS 1.3 [RFC9147] will always include the length field in each record.

4. Protection Valid Message

The Protection Valid message is sent from the responder to the Initiator when the responder has confirmed reception of DTLS chunk protected DTLS messages. This message triggers requiring protection at the Initiator when it has been received.

The message SHALL be sent protected.

The message is 2 bytes and is 0x4F4B. This SCTP user message MUST be sent reliable and on Stream 0 with in order delivery in relatio to any DTLS ACK message.

5. DTLS Chunk Integration

The [I-D.westerlund-tsvwg-sctp-dtls-chunk] contains a high-level description of the basic DTLS in SCTP architecture, this section deals with details related to the DTLS 1.3 inband key-establishment integration with SCTP.

5.1. SCTP Association Life Cycle.

DTLS in SCTP uses inband key-establishment, thus the DTLS handshake for a Key-Management DTLS Connection establishes shared keys with the remote peer. As soon as the SCTP State Machine enters established state, DTLS in SCTP is responsible for progressing to where the DTLS Chunk is fully configured and the ULP will be protected.

5.1.1. PROTECTION INITILIZATION

When the SCTP Association enter ESTABLISHED state, the initator will start the handshake according to Section 7.2.

When a successful handshake has been completed, the Primary DTLS Key Context and the Restart DTLS Key Context will be created by deriving the keys and IVs from the key-management DTLS connection. These will be installed in the DTLS Chunks as defined in this document to avoid dead lock and ensure successful protection enabling the ULP

5.1.2. SCTP Association Ongoing

When an SCTP Association is protected the establishe primary DTLS key context is used for Chunk protection operation of the payload of SCTP chunks in each packet per the DTLS Chunk specification [I-D.westerlund-tsvwg-sctp-dtls-chunk].

When necessary to meet requirements on key life time or periodic re-authentication of the peer and establishment of new forward secrecy keys, the existing DTLS 1.3 key-managment connection is being replaced with a new one by first opening a new parallel DTLS connection as further specified in Section 8, derive and install new DTLS Key Contexts and then close the old DTLS connection and remove the old DTLS Key contexts.

5.1.3. SHUTDOWN states

When the SCTP association leaves the ESTABLISHED state per [RFC9260] to be shutdown the DTLS Chunk’s key contexts are kept and continues to protect the SCTP packet payloads through the shutdown process.

When the association reaches the CLOSED state as part of the SCTP association closing process all DTLS connections existing (traffic and restart) for this association are terminated without further transmissions, i.e. DTLS close_notify is not transmitted.

5.2. DTLS Connection Handling

It’s up to DTLS key-establishment function to manage the DTLS connections and their related DTLS Key context in the DTLS chunk.

5.2.1. Add a New DTLS Connection

Either peer can add a new DTLS connection to the SCTP association at any time, but no more than 2 DTLS connections can exist at the same time. Details of the handshake are described in Section 7.2.

As either endpoint can initiate a DTLS handshake at the same time, either endpoint may receive a DTLS ClientHello message 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 previous DTLS connection shall be continued to be processed and the other dropped.

When the handshake has been completed successfully, the new DTLS Key contexts are established by exporting keys and installing the in the DTLS Chunk, and the new DTLS Key context are immediately started to be used. If the handshake is not completed successfully, a new DTLS handshake attempt will be tried using the same DCI.

5.2.2. Remove an existing DTLS Connection

A DTLS connection is removed when a newer DTLS connection is in use. It is RECOMMENDED to not initiate removal until at least one SCTP packet protected by the new DTLS Key Context has been received, and any transmitted packets protected using the new DTLS Key Context has been acknowledge, alternatively one Maximum Segment Lifetime (120 seconds) has passed since the last SCTP packet protected by the old DTLS connection was transmitted.

Either peers can initialize the removal of a DTLS connection from the current SCTP association when needed when a new have been established. The closing of the DTLS connection when the SCTP association is in PROTECTED and ESTABLISHED state is done by having the DTLS connection send a DTLS close_notify. When the DTLS closure of a DTLS connection is completed, the related DTLS Key Contexts (Traffic and Restart) in the DTLS chunk are released.

5.2.3. Considerations about removal of DTLS Connections

Removal of a DTLS connection may happen under circumstances as described above in Section 5.2.2 in different states of the Association. This section describes how the implementation should take care of the DTLS connection removal in details.

The initial DTLS connection exists as soon as Association reaches the PROTECTED state. As long as one DTLS connection only exists, that DTLS connection SHALL NOT be removed as it won’t be possible for the Association to proceed further.

In general a DTLS connection can be removed when there’s another active DTLS connection with valid DTLS Key Contexts that can be used for negotiating further key-management DTLS 1.3 connections. In case the DTLS connection is removed and no useable DTLS Key Context exist for key-management DTLS 1.3 negotiation, the Association SHALL be ABORTED.

It is up to the implementation to guarantee that a DTLS Key Context exists all the time, for avoiding that undesired DTLS connection closure causes the Association abortion.

5.3. DTLS Key Update

DTLS Key Update MUST NOT be used. DTLS Key Context replacement MUST be used instead, by means creating a new DTLS connection as specified in Section 8, deriving the new Traffic DTLS Key Context, the new Restart DTLS Key Context and then closing the old DTLS connection.

5.4. Error Cases

As DTLS has its own error reporting mechanism by exchanging DTLS alert messages no new DTLS related cause codes are defined to use the error handling defined in [I-D.westerlund-tsvwg-sctp-dtls-chunk].

When Handshake DTLS connection encounters an error it may report that issue using DTLS alert message to its peer by putting the created DTLS record in a SCTP user message (Section 3) with the Handshake PPID. This is independent of what to do in relation to the SCTP association. Depending on the severance of the error different paths can be the result:

However, as there is not expected that the key-management DTLS connection will at all have any activity between completing the handshake, and the DTLS connection closing, there is unlikely that any error will occur.

6. DTLS Considerations

6.1. Version of DTLS

This document defines the usage of DTLS 1.3 [RFC9147]. Earlier versions of DTLS MUST NOT be used (see [RFC8996]). It is expected that DTLS in SCTP as described in this document will work with future versions of DTLS.

Only one version of DTLS MUST be used during the lifetime of an SCTP Association, meaning that the procedure for replacing the DTLS version in use requires the existing SCTP Association to be terminated and a new SCTP Association with the desired DTLS version to be instantiated.

6.2. Configuration of Key-Management DTLS

6.2.1. General

The DTLS Connection ID SHOULD NOT be used in the Key-Management DTLS Connection avoiding overhead and addition implementation requirements on DTLS implementation.

The DTLS record length field is normally not needed as the DTLS Chunk provides a length field unless multiple records are put in same DTLS chunk payload or user message. If multiple DTLS records are included in one DTLS chunk payload or user message the DTLS record length field MUST be present in all but the last.

DTLS record replay detection MUST be used.

Sequence number size can be adapted based on how quickly it wraps.

Many of the TLS registries have a "Recommended" column. Parameters not marked as "Y" are NOT RECOMMENDED to support. Non-AEAD cipher suites or cipher suites without confidentiality MUST NOT be supported. Cipher suites and parameters that do not provide ephemeral key-exchange MUST NOT be supported.

The Cipher suites negotiated in the Key-Management DTLS Connection SHALL only include those supported by the DTLS Chunk. The DTLS Chunk is expected to have an API capability to determine the Cipher Suit Capabilities, see Abstract API in Section 10.1 of [I-D.westerlund-tsvwg-sctp-dtls-chunk].

6.2.2. Authentication and Policy Decisions

DTLS in 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 in SCTP MUST perform identity authentication. It is RECOMMENDED that DTLS in SCTP is used with certificate-based authentication.

When certificates are used the application using DTLS in 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 in SCTP defines 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 [RFC9525]. DTLS in 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.

6.2.3. New Connections

Implementations MUST set up a new DTLS connection using a full handshake 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 MUST limit the number of simultaneous connections to two.

To force attackers to do dynamic key exfiltration and limit the amount of compromised data due to key compromise, implementations MUST have policies for how often to set up new connections with ephemeral key exchange such as ECDHE. Implementations SHOULD set up new connections frequently to force attackers to dynamic key extraction. E.g., at least every hour and every 100 GB of data which is a common policy for IPsec [ANSSI-DAT-NT-003]. See [I-D.ietf-tls-rfc8446bis] for a more detailed discussion on key compromise and key exfiltration in (D)TLS. As recommended in [KTH-NCSA], resumption can be used to chain the connections, increasing security by forcing an adversary to break them in sequence.

For many DTLS in 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 by setting up a new connection using a full handshake. TLS Certificate lifetimes significantly shorter than a year are common which is shorter than many expected SCTP associations protected by DTLS in SCTP.

6.2.4. DTLS 1.3

DTLS 1.3 is used instead of 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. Implementations setup a new DTLS connection to handle the need for new keys, alternatively terminate the SCTP Assocation.

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. Resumption can have significant latency benefits for quickly restarting a broken DTLS/SCTP association. If tickets and resumption are used it is enough to issue a single ticket per connection.

The PSK key exchange mode : psk_ke MUST NOT be used as it does not provide ephemeral key exchange.

7. Establishing DTLS in SCTP

This section specifies how DTLS in SCTP is established using Key-Management DTLS Connections and the DTLS Chunk [I-D.westerlund-tsvwg-sctp-dtls-chunk].

A DTLS in SCTP Association is built up with a Key-Management DTLS connection, from that DTLS connection Traffic DTLS Key Context and Restart DTLS Key Context are derived and the configures the DTLS Context.

The Key-Management DTLS connection is established as part of extra procedures for the DTLS chunk initial handshake (see Section 7.2.1).

7.1. DTLS Key Context derivation

This section describes how DTLS Key Contexts are derived from the DTLS handshake using the TLS Exporter as defined by [RFC9147]. The TLS exporter label specifications below is following [RFC5705].

There are two sets of keys one for the primary DTLS key context and one for the restart DTLS Key Context. Each set consists of one client and one server side write key. In addition each key needs an Initilization Vector (IV) that is used by the record processing in TLS to create the nonce, See Section 5.3 of [RFC8446]. The client and server roles are here in relation to key-management DTLS session roles. So the DTLS Client will install the key derived using the EXPORTER_DTLS_IN_SCTP_PRIMARY_CLIENT_KEY label as its write key for the traffic context, and use the EXPORTER_DTLS_IN_SCTP_PRIMARY_SERVER_KEY as its traffic DTLS context read key. Correspondlingly the EXPORTER_DTLS_IN_SCTP_RESTART_CLIENT_KEY is used to export the key used by the endpoint that acted as DTLS Client as write key for the restart DTLS key context. And the EXPORTER_DTLS_IN_SCTP_RESTART_SERVER_KEY as the DTLS client’s read key for the restart DTLS Key Context. Correspondligy the IV values needs to be exported using the corresponding _IV label.

The following labels are defined:

* EXPORTER_DTLS_IN_SCTP_PRIMARY_CLIENT_KEY

* EXPORTER_DTLS_IN_SCTP_PRIMARY_CLIENT_IV

* EXPORTER_DTLS_IN_SCTP_PRIMARY_SERVER_KEY

* EXPORTER_DTLS_IN_SCTP_PRIMARY_SERVER_IV

* EXPORTER_DTLS_IN_SCTP_RESTART_CLIENT_KEY

* EXPORTER_DTLS_IN_SCTP_RESTART_CLIENT_IV

* EXPORTER_DTLS_IN_SCTP_RESTART_SERVER_KEY

* EXPORTER_DTLS_IN_SCTP_RESTART_SERVER_IV

To ensure that downgrade attack on the protection solution offered is not possible the context used will be the full sequence of Protection Solution Identifiers as include in the DTLS 1.3 Chunk Protected Association (Section 4.1 of [I-D.westerlund-tsvwg-sctp-dtls-chunk]) sent by the SCTP assocation initiator. Thus, any downgrade attack on this will result in a mismatch in produced keys as the initiator will use what it actually offered and the responder a truncated or modified sequence.

The length of the exported key or IV material depends on the need for the negotiatated cipher suit for the protection.

7.2. DTLS Handshake

7.2.1. Protection Initilization using DTLS connection

The handshake of the initial DTLS connection is part of the DTLS in SCTP Association initialization. The key-management DTLS connections progress interacts with the key installation for the SCTP associations DTLS chunk.

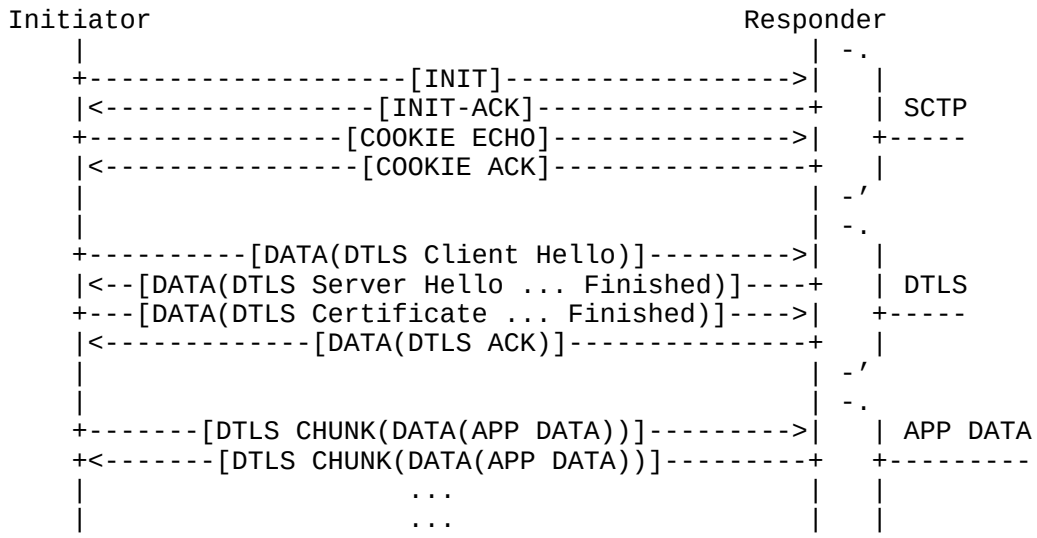

Figure 4: Handshake of initial DTLS connection

SCTP Handshake is strictly compliant to [RFC9260]. The DTLS 1.3 Chunk Protected Association parameter (Section 4.1 of [I-D.westerlund-tsvwg-sctp-dtls-chunk]) is included containing the Protection Solution identifier (See Section 10.1) for this documents key-management at a suitable preference position depending on local policy. And in case this key-management solution is the most preferred then the process continues as stated below and depicted in Figure 5.

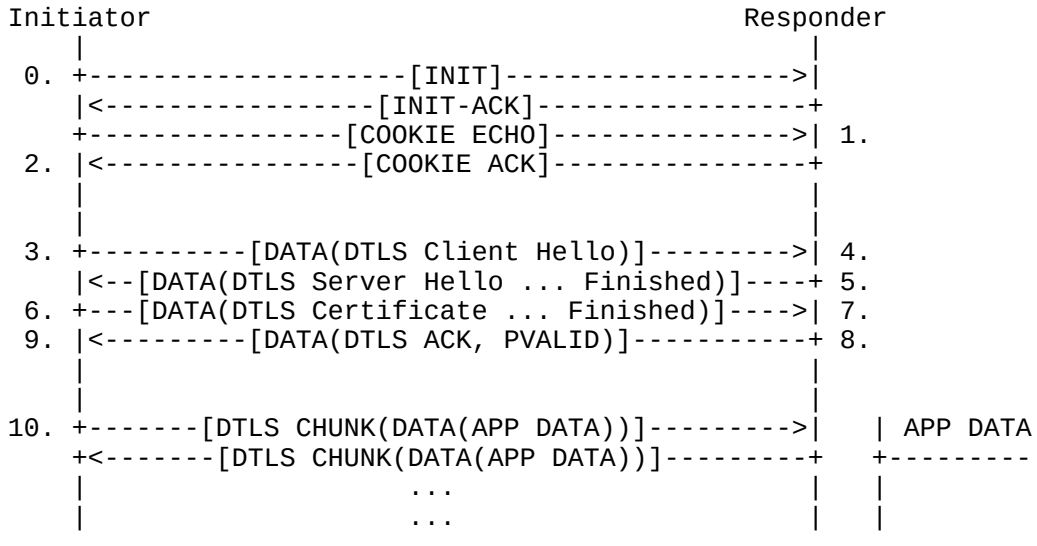

Figure 5: The steps of Interaction between Key-Management and DTLS Chunk API

1. The Initiator initiates an SCTP Association and provides the DTLS 1.3 Chunk Protected Association parameter preference ordered list of supporter Protection Solutions. The offered parameter list is remembered by the Key-Management.

2. The Responder peer enter SCTP Established, and the Key-Management is provided with the full ordered list of Protection Solutions offered in the INIT Chunk.

3. The Initiator enters SCTP Assocationa Established and the Key-Management is triggered to perform the next step.

4. Key-Management initiated a DTLS handshake with the the supported configuration. Taking supported Cipher-suits in the DTLS Chunk implementation into account when creating its DTLS Client-Hello mesage. The DTLS messages are sent per Section 3

5. Responder receives DTLS Client-Hello and generates the DTLS Server Hello, etc response message(s) for the DTLS handshake. In case the DTLS server in the responder requires the use of the retry message an additional message exchange between DTLS Client and DTLS server is needed before one can progress to 5.

6. Responder uses its the TLS Exporter on the DTLS Connection’s state to derive the primary client write key and IV Section 7.1 and install them into the DTLS Chunk’s primary Key Context. Then it sends the DTLS Server’s response message(s).

7. The DTLS client receives the DTLS server’s messages (Server Hello etc.) and can now export both the client and server write key for the primary and restart Key Contexts, however their usage is not yet required and SCTP packets without DTLS chunks are still accepted. Then the DTLS Client next handshake message is sent. This message SHALL be protected by the DTLS Chunk using the primary key Context (Client Write key and IV).

8. The responder’s DTLS chunk will receive the SCTP packets containing the DTLS chunk protected DTLS messages. Concluding the main process of the DTLS handshake.

9. The responder export the remaining keys and IVs and installs all primary and restart Server Write Key and IV, as well as restart client write key and IV. After that it requires all future SCTP Packets to be protected by DTLS Chunk. If any DTLS ACK message is to be sent, it SHOULD be sent next. Then it sends a

protected Protection Valid Message Section 4 to the SCTP Association Initiators Key-Management. The key-management can now inform the ULP that the SCTP association is protected.

10. Upon successful reception of the Protection Valid Message the Initiator's Key-Management configure the DTLS Chunk to only accept DTLS Chunk Protected SCTP packets.

11. The Initiator's ULP is notified that the SCTP association is Established and protected and it may generate user messages.

If the DTLS handshake failed the SCTP association SHALL be aborted and an ERROR chunk with the Error in Protection error cause, with the appropriate extra error causes is generated, the right selection of "Error During Protection Handshake" or "Timeout During Protection Handshake or Validation".

7.2.2. Handshake of further DTLS connections

When the SCTP Association has entered the PROTECTED state, each of the endpoint can initiate a DTLS handshake for rekeying when necessary.

The DTLS endpoint will if necessary fragment the handshake into multiple records. Each DTLS handshake message fragment is sent as a SCTP user message Section 3. The DTLS instance SHOULD NOT use DTLS retransmission to repair any packet losses of handshake message fragment. Note: If the DTLS implementation support configuring a MTU larger than the actual IP MTU it could be used as SCTP provides reliability and fragmentation.

If the DTLS handshake failed the SCTP association SHALL generate an ERROR chunk with the Error in Protection error cause, with extra error causes "Error During Protection Handshake".

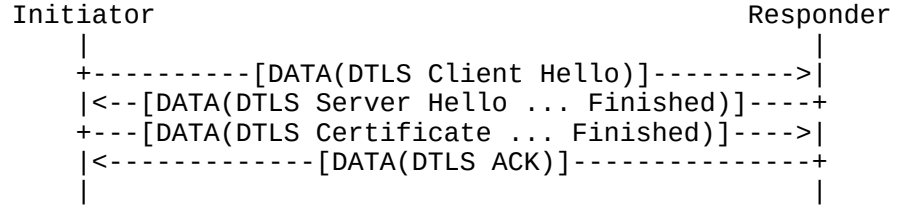

Figure 6: Handshake of further DTLS connection

The Figure 6 shows a successful handshake of a further DTLS connection. Such connections can be initiated by any of the peers. Same as during the initial handshake, DTLS handshake messages are transported by means of DATA chunks with the DTLS Chunk Key-Management Messages PPID.

7.3. SCTP Association Restart

In order to achieve an Association Restart as described in [I-D.westerlund-tsvwg-sctp-dtls-chunk], a Restart DTLS Key Context dedicated to Restart SHALL exist and be available. Furthermore, both peers SHALL have safely stored both the current Restart DTLS Key Context. Here we assume that Restart DTLS Key Context is maintained across the events leading to SCTP Restart request.

7.3.1. Installation of initial Restart DTLS Key Context

As soon as the Association has reached the PROTECTED INITILIZATION state, after the Traffic DTLS Key context have been installed a Restart DTLS Key Context SHALL be derived and then installed.

It MAY exist a short time gap where the Association has entered in PROTECTED state but no Restart DTLS Key Context has been installed yet. If a SCTP Restart procedure will be initiated during that time, it will fail and the Association will also fail. However, this is unlikely as the restart Init will be sent multiple times following a exponential back-off timer and in that time the Restart DTLS Key Context is expected to be in place.

Once installed, no traffic will be sent over the Restart DTLS Key Context so that both endpoints will have a known DTLS context state, i.e. the Sequence number and replay window are both just initialized to default values for the epoch=3.

7.3.2. Installation of Restart DTLS Key Context for further DTLS Connections

As each subsequent Key-Management DTLS connection complete the DTLS handshake and the Traffic DTLS Key context has been derived and installed also the Restart DTLS Key context SHALL be installed. The closing of the previous DTLS connection SHALL NOT be initiated or completed until the Restart DTLS Key Context is in place.

7.3.3. SCTP Association Restart Procedure

The DTLS in SCTP Association Restart is meant to preserve the security characteristics.

In order the Association Restart to proceed both Initiator and Responder SHALL use the same Restart DTLS Key Context for COOKIE-ECHO/COOKIE-ACK handshake, that implies that the Initiator must preserve the Restart DTLS Key Context and that the Responder SHALL NOT change the Restart DTLS Key Context during the Restart procedure.

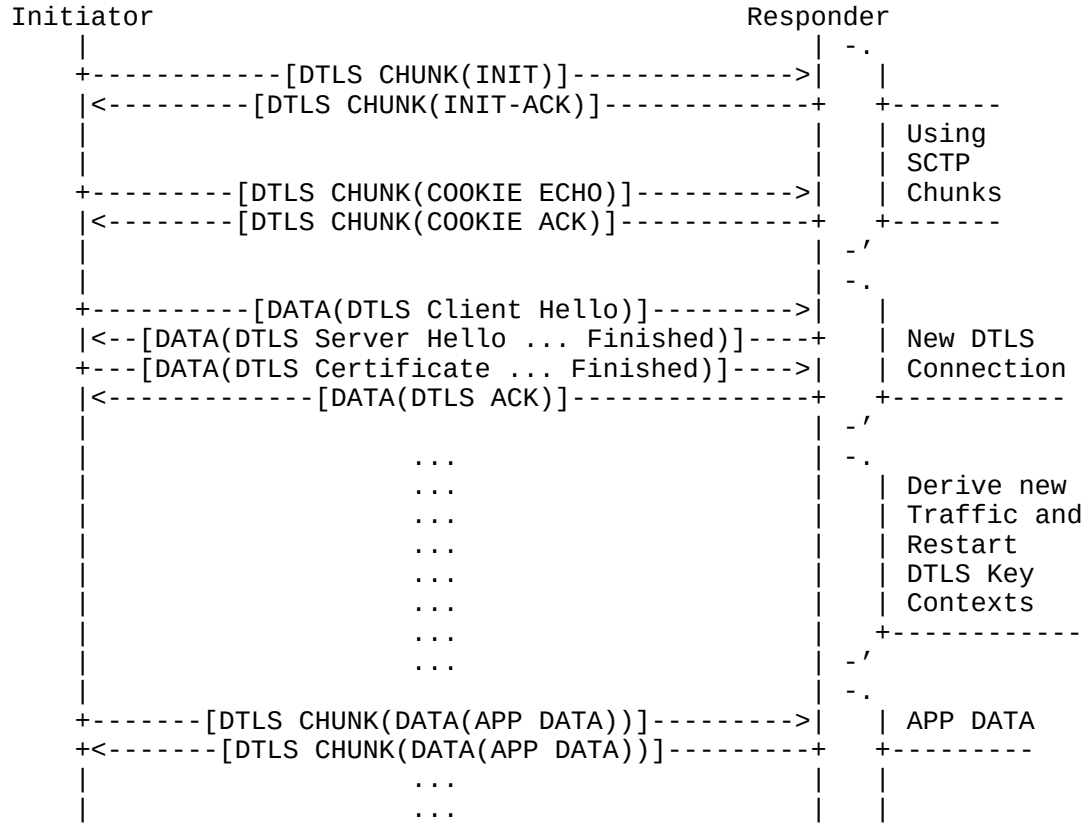

Figure 7: SCTP Restart sequence for DTLS in SCTP

The Figure 7 shows a successful SCTP Association Restart.

From procedure viewpoint the sequence is the following:

* Initiator sends INIT (VTag=0), Responder replies INIT-ACK both encrypted using Restart DTLS Key Context

* Initiator sends COOKIE-ECHO using DTLS CHUNK encrypted with the Key tied to the Restart DTLS Key Context

* Responder replies with COOKIE-ACK using DTLS CHUNK encrypted with the Restart DTLS Key Context

* Initiator sends handshakes for new Traffic DTLS connection as well as new Restart DTLS connection.

* The SCTP Association goes into Established state and jumps directly to PROTECTED state, and the ULP can resume communication protected using the Restart DTLS Key Context.

* A new key-management DTLS handshake is initiated as soon as PROTECTED state has been reached. When the Key-management DTLS connection has been completed, new Traffic and Restart DTLS Key Contexts are derived and installed using Epoch=3. The new Traffic DTLS Key Context is being used for traffic as soon as the context have been installed. When Traffic Key Context has been installed the new Restart DTLS Key Context for epoch=3 are installed.

User Data for any ULP traffic MAY be initiated immediately after COOKIE-ECHO/COOKIE-ACK handshake using the current Restart DTLS Key Context, that is even before a new Traffic DTLS Key Context or a Restart DTLS Key Context have been derived. If a problem occurs before the new Restart DTLS Key Context has been installed, the Association cannot be Restarted, thus it’s RECOMMENDED the new Restart DTLS Key Context to be installed as early as possible.

Note that, different than the initial Association establishment, the ULP traffic is permitted immediately after the COOKIE-ECHO/COOKIE-ACK handshake, the reason is that the validation has already been performed prior to the restart DTLS Key Context was created.

8. Parallel DTLS Rekeying

Rekeying in this specification is implemented by replacing the DTLS connection getting old with a new one by first creating the new DTLS connection, derive the new DTLS Key contexts, start using it, then closing the old one.

8.1. Criteria for Rekeying

The criteria for rekeying may vary depending on the ULP requirement on security properties, chosen cipher suits etc. Therefore it is assumed that the implementation will be configurable by the ULP to meet its demand.

Likely criteria to impact the need for rekeying through the usage of new DTLS connection are:

* Time duration since last authentication of the peer

* Amount of data transferred since last forward secrecy preserving rekeying

* The cipher suit’s maximum key usage being reached.

8.2. Procedure for Rekeying

This specification allows up to 2 DTLS connection to be active at the same time for the current SCTP Association. The following state machine applies.

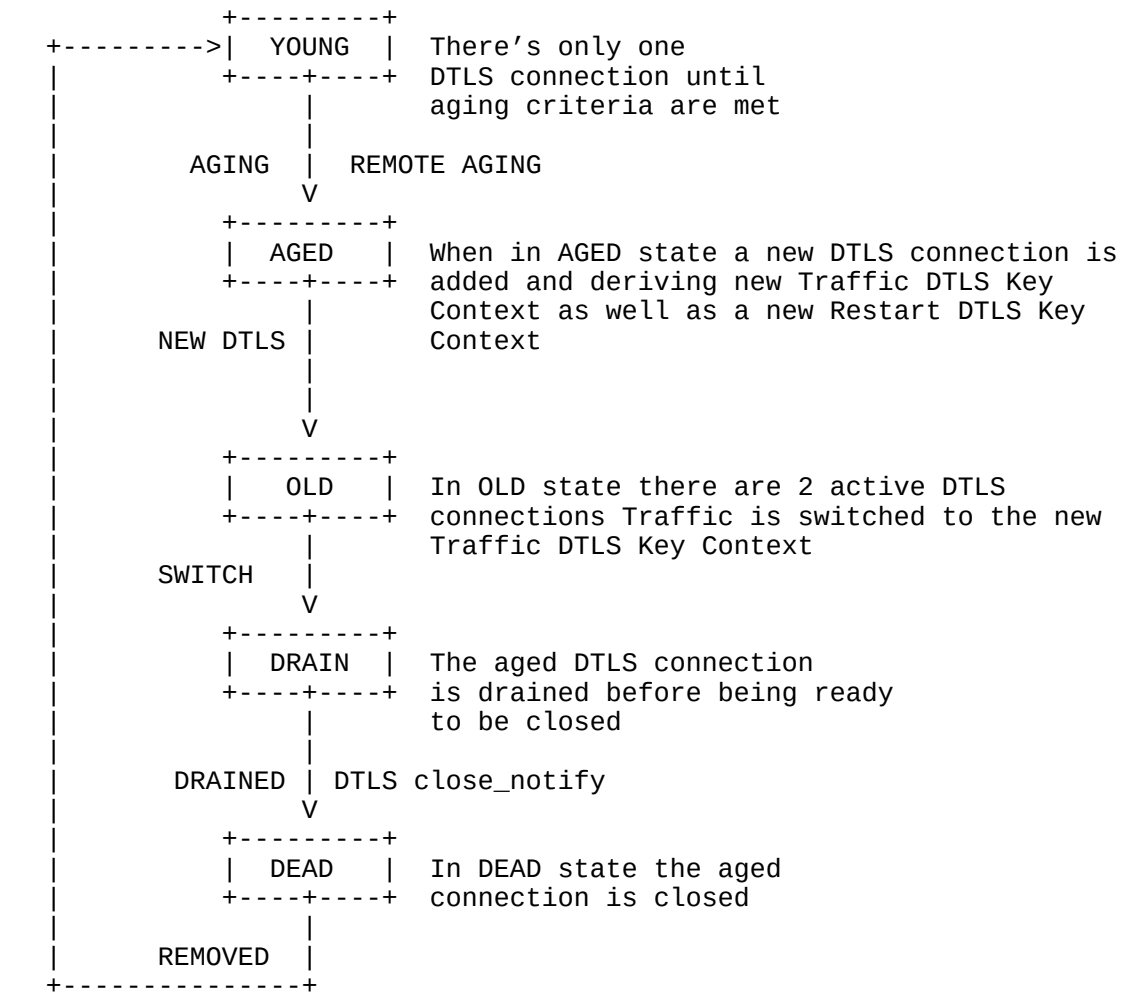

Figure 8: State Diagram for Rekeying

Trigger for rekeying can either be a local AGING event, triggered by the DTLS connection meeting the criteria for rekeying, or a REMOTE AGING event, triggered by receiving a DTLS Connection handshake on the DTLS Connection Index (next value) that would be used for new DTLS connection. In such case a new DTLS connection shall be added according to Section 5.2.1.

As soon as the new DTLS connection completes handshaking, and the Traffic and Restart DTLS Key Contexts have been derived and installed, the protection of the SCTP packets is moved from the old Traffic DTLS Key Context, then the procedure for closing the old DTLS connection is initiated, see Section 5.2.2.

8.3. Race Condition in Rekeying

A race condition may happen when both peers experience local AGING event at the same time and start creation of a new DTLS connection.

The race condition is solved as specified in Section 5.2.1.

9. Security Considerations

9.1. General

The security considerations given in [RFC9147], [RFC6347], and [RFC9260] also apply to this document. BCP 195 [RFC9325] [RFC8996] provides recommendations and requirements for improving the security of deployed services that use DTLS. BCP 195 MUST be followed which implies that DTLS 1.0 SHALL NOT be supported and are therefore not defined.

9.2. Privacy Considerations

Although DTLS in SCTP provides privacy for the actual user message as well as almost all chunks, some fields are not confidentiality protected. In addition to the DTLS record header, the SCTP common header and the DTLS chunk header are not confidentiality protected. An attacker can correlate DTLS connections over the same SCTP association using the SCTP common header.

To provide identity protection it is RECOMMENDED that DTLS in SCTP is used with certificate-based authentication in DTLS 1.3 [RFC9147] and to not reuse tickets. DTLS 1.3 with external PSK authentication does not provide identity protection.

By mandating ephemeral key exchange and cipher suites with confidentiality DTLS in SCTP effectively mitigate many forms of passive pervasive monitoring. By recommending implementations to frequently set up new DTLS connections with (EC)DHE force attackers to do dynamic key exfiltration and limits the amount of compromised data due to key compromise.

10. IANA Consideration

This document requests the following registration.

10.1. SCTP Protection Solution Identifier

IANA is requested to assign one SCTP Protection Solution Identifier to identify the key-management defined in this document.

Identifier	Solution Name	Reference	Contact
4096	DTLS in SCTP Handshake	RFC-TBD	Draft Authors

Table 1: SCTP Protection Solution Indicators

10.2. TLS Exporter Labels

IANA is requested to register the following values in the TLS Exporter Label Registry [RFC5705] with Reference RFC-TO-BE and empty Comment. The registry was at the time of writing located at: https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#exporter-labels

Value	DTLS-OK	Recommended
EXPORTER_DTLS_IN_SCTP_PRIMARY_CLIENT_KEY	Y	N
EXPORTER_DTLS_IN_SCTP_PRIMARY_CLIENT_IV	Y	N
EXPORTER_DTLS_IN_SCTP_PRIMARY_SERVER_KEY	Y	N
EXPORTER_DTLS_IN_SCTP_PRIMARY_SERVER_IV	Y	N
EXPORTER_DTLS_IN_SCTP_RESTART_CLIENT_KEY	Y	N
EXPORTER_DTLS_IN_SCTP_RESTART_CLIENT_IV	Y	N
EXPORTER_DTLS_IN_SCTP_RESTART_SERVER_KEY	Y	N
EXPORTER_DTLS_IN_SCTP_RESTART_SERVER_IV	Y	N

Table 2: TLS Exporter Labels

11. References

11.1. Normative References

[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>.

[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>.

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

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[I-D.westerlund-tsvwg-sctp-dtls-chunk] Westerlund, M., Preu Mattsson, J., Porfiri, C., and M. Txen, "Stream Control Transmission Protocol (SCTP) DTLS chunk", July 2025, <https://datatracker.ietf.orghttps://datatracker.ietf.org/doc/draft-westerlund-tsvwg-sctp-dtls-chunk/>.

[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>.

[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>.

11.2. Informative References

[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>.

[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>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/info/rfc9525>.

[I-D.ietf-tls-rfc8446bis] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-12, 17 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-rfc8446bis-12>.

[I-D.ietf-uta-rfc6125bis] Saint-Andre, P. and R. Salz, "Service Identity in TLS", Work in Progress, Internet-Draft, draft-ietf-uta-rfc6125bis-15, 10 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-rfc6125bis-15>.

[ANSSI-DAT-NT-003] Agence nationale de la scurit des systmes 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>>.

[KTH-NCSA] Eker, M., "On factoring integers, and computing discrete logarithms and orders, quantumly", KTH, School of Electrical Engineering and Computer Science (EECS), Computer Science, Theoretical Computer Science, TCS. Swedish NCSA, Swedish Armed Forces. , October 2024, <http://kth.diva-portal.org/smash/get/diva2:1902626/FULLTEXT01.pdf>.

Authors’ Addresses

Magnus Westerlund
Ericsson
Email: magnus.westerlund@ericsson.com

John Preu Mattsson
Ericsson

Email: john.mattsson@ericsson.com

Claudio Porfiri
Ericsson
Email: claudio.porfiri@ericsson.com