

Workgroup: TLS
Internet-Draft:
draft-ietf-tls-dtls-connection-id-13
Updates: [6347](#) (if approved)
Published: 22 June 2021
Intended Status: Standards Track
Expires: 24 December 2021
Authors: E. Rescorla, Ed. H. Tschofenig, Ed. T. Fossati
 RTFM, Inc. Arm Limited Arm Limited
 A. Kraus
 Bosch.IO GmbH

Connection Identifiers for DTLS 1.2

Abstract

This document specifies the Connection ID (CID) construct for the Datagram Transport Layer Security (DTLS) protocol version 1.2.

A CID is an identifier carried in the record layer header that gives the recipient additional information for selecting the appropriate security association. In "classical" DTLS, selecting a security association of an incoming DTLS record is accomplished with the help of the 5-tuple. If the source IP address and/or source port changes during the lifetime of an ongoing DTLS session then the receiver will be unable to locate the correct security context.

The new ciphertext record format with CID also provides content type encryption and record-layer padding.

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 24 December 2021.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

- [1. Introduction](#)
- [2. Conventions and Terminology](#)
- [3. The "connection_id" Extension](#)
- [4. Record Layer Extensions](#)
- [5. Record Payload Protection](#)
 - [5.1. Block Ciphers](#)
 - [5.2. Block Ciphers with Encrypt-then-MAC processing](#)
 - [5.3. AEAD Ciphers](#)
- [6. Peer Address Update](#)
- [7. Examples](#)
- [8. Privacy Considerations](#)
- [9. Security Considerations](#)
- [10. IANA Considerations](#)
 - [10.1. Extra Column to TLS ExtensionType Values Registry](#)
 - [10.2. Entry to the TLS ExtensionType Values Registry](#)
 - [10.3. Entry to the TLS ContentType Registry](#)
- [11. References](#)
 - [11.1. Normative References](#)
 - [11.2. Informative References](#)
- [Appendix A. History](#)

[Appendix B. Working Group Information](#)
[Appendix C. Contributors](#)
[Appendix D. Acknowledgements](#)
[Authors' Addresses](#)

1. Introduction

The Datagram Transport Layer Security (DTLS) [[RFC6347](#)] protocol was designed for securing connection-less transports, like UDP. DTLS, like TLS, starts with a handshake, which can be computationally demanding (particularly when public key cryptography is used). After a successful handshake, symmetric key cryptography is used to apply data origin authentication, integrity and confidentiality protection. This two-step approach allows endpoints to amortize the cost of the initial handshake across subsequent application data protection. Ideally, the second phase where application data is protected lasts over a long period of time since the established keys will only need to be updated once the key lifetime expires.

In DTLS as specified in RFC 6347, the IP address and port of the peer are used to identify the DTLS association. Unfortunately, in some cases, such as NAT rebinding, these values are insufficient. This is a particular issue in the Internet of Things when devices enter extended sleep periods to increase their battery lifetime. The NAT rebinding leads to connection failure, with the resulting cost of a new handshake.

This document defines an extension to DTLS 1.2 to add a Connection ID (CID) to the DTLS record layer. The presence of the CID is negotiated via a DTLS extension.

Adding a CID to the ciphertext record format presents an opportunity to make other changes to the record format. In keeping with the best practices established by TLS 1.3, the type of the record is encrypted, and a mechanism provided for adding padding to obfuscate the plaintext length.

2. Conventions and Terminology

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.

This document assumes familiarity with DTLS 1.2 [[RFC6347](#)]. The presentation language used in this document is described in Section 3 of [[RFC8446](#)].

3. The "connection_id" Extension

This document defines the "connection_id" extension, which is used in ClientHello and ServerHello messages.

The extension type is specified as follows.

```
enum {  
    connection_id(TBD1), (65535)  
} ExtensionType;
```

The extension_data field of this extension, when included in the ClientHello, MUST contain the ConnectionId structure. This structure contains the CID value the client wishes the server to use when sending messages to the client. A zero-length CID value indicates that the client is prepared to send using a CID but does not wish the server to use one when sending.

```
struct {  
    opaque cid<0..2^8-1>;  
} ConnectionId;
```

A server willing to use CIDs will respond with a "connection_id" extension in the ServerHello, containing the CID it wishes the client to use when sending messages towards it. A zero-length value indicates that the server will send using the client's CID but does not wish the client to include a CID when sending.

Because each party sends the value in the "connection_id" extension it wants to receive as a CID in encrypted records, it is possible for an endpoint to use a deployment-specific constant length for such connection identifiers. This can in turn ease parsing and connection lookup, for example by having the length in question be a compile-time constant. Such implementations MUST still be able to send CIDs of different length to other parties. Since the CID length information is not included in the record itself, implementations that want to use variable-length CIDs are responsible for constructing the CID in such a way that its length can be determined on reception.

In DTLS 1.2, CIDs are exchanged at the beginning of the DTLS session only. There is no dedicated "CID update" message that allows new CIDs to be established mid-session, because DTLS 1.2 in general does not allow TLS 1.3-style post-handshake messages that do not themselves begin other handshakes. When a DTLS session is resumed or renegotiated, the "connection_id" extension is negotiated afresh.

If DTLS peers have not negotiated the use of CIDs, or a zero-length CID has been advertised for a given direction, then the RFC 6347-

defined record format and content type MUST be used to send in the indicated direction(s).

If DTLS peers have negotiated the use of a non-zero-length CID for a given direction, then once encryption is enabled they MUST send with the record format defined in [Figure 3](#) with the new MAC computation defined in [Section 5](#) and the content type `tls12_cid`. Plaintext payloads never use the new record format or the CID content type.

When receiving, if the `tls12_cid` content type is set, then the CID is used to look up the connection and the security association. If the `tls12_cid` content type is not set, then the connection and security association is looked up by the 5-tuple and a check MUST be made to determine whether a non-zero length CID is expected. If a non-zero-length CID is expected for the retrieved association, then the datagram MUST be treated as invalid, as described in Section 4.1.2.1 of [[RFC6347](#)].

When receiving a datagram with the `tls12_cid` content type, the new MAC computation defined in [Section 5](#) MUST be used. When receiving a datagram with the RFC 6347-defined record format, the MAC calculation defined in Section 4.1.2 of [[RFC6347](#)] MUST be used.

4. Record Layer Extensions

This specification defines the DTLS 1.2 record layer format and [[I-D.ietf-tls-dtls13](#)] specifies how to carry the CID in DTLS 1.3.

To allow a receiver to determine whether a record has a CID or not, connections which have negotiated this extension use a distinguished record type `tls12_cid`(TBD2). Use of this content type has the following three implications:

- *The CID field is present and contains one or more bytes.
- *The MAC calculation follows the process described in [Section 5](#).
- *The real content type is inside the encryption envelope, as described below.

Plaintext records are not impacted by this extension. Hence, the format of the `DTLSPayload` structure is left unchanged, as shown in [Figure 1](#).

```

struct {
    ContentType type;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    uint16 length;
    opaque fragment[DTLSPlaintext.length];
} DTLSPlaintext;

```

Figure 1: DTLS 1.2 Plaintext Record Payload.

When CIDs are being used, the content to be sent is first wrapped along with its content type and optional padding into a `DTLSInnerPlaintext` structure. This newly introduced structure is shown in [Figure 2](#).

```

struct {
    opaque content[length];
    ContentType real_type;
    uint8 zeros[length_of_padding];
} DTLSInnerPlaintext;

```

Figure 2: New `DTLSInnerPlaintext` Payload Structure.

content Corresponds to the fragment of a given length.

real_type The content type describing the cleartext payload.

zeros An arbitrary-length run of zero-valued bytes may appear in the cleartext after the type field. This provides an opportunity for senders to pad any DTLS record by a chosen amount as long as the total stays within record size limits. See Section 5.4 of [\[RFC8446\]](#) for more details. (Note that the term `DTLSInnerPlaintext` in RFC 8446 refers to `DTLSInnerPlaintext` in this specification.)

The `DTLSInnerPlaintext` byte sequence is then encrypted. To create the `DTLSCiphertext` structure shown in [Figure 3](#) the CID is added.

```

struct {
    ContentType outer_type = tls12_cid;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    opaque cid[cid_length]; // New field
    uint16 length;
    opaque enc_content[DTLSCiphertext.length];
} DTLSCiphertext;

```

Figure 3: DTLS 1.2 CID-enhanced Ciphertext Record.

outer_type The outer content type of a DTLSCiphertext record carrying a CID is always set to `tls12_cid`(TBD2). The real content type of the record is found in `DTLSInnerPlaintext.real_type` after decryption.

cid The CID value, `cid_length` bytes long, as agreed at the time the extension has been negotiated. Recall that (as discussed previously) each peer chooses the CID value it will receive and use to identify the connection, so an implementation can choose to always receive CIDs of a fixed length. 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.

enc_content The encrypted form of the serialized `DTLSInnerPlaintext` structure.

All other fields are as defined in RFC 6347.

5. Record Payload Protection

Several types of ciphers have been defined for use with TLS and DTLS and the MAC calculations for those ciphers differ slightly.

This specification modifies the MAC calculation as defined in [\[RFC6347\]](#) and [\[RFC7366\]](#), as well as the definition of the additional data used with AEAD ciphers provided in [\[RFC6347\]](#), for records with content type `tls12_cid`. The modified algorithm MUST NOT be applied to records that do not carry a CID, i.e., records with content type other than `tls12_cid`.

The following fields are defined in this document; all other fields are as defined in the cited documents.

cid Value of the negotiated CID (variable length).

cid_length 1 byte field indicating the length of the negotiated CID.

length_of_DTLSInnerPlaintext The length (in bytes) of the serialized `DTLSInnerPlaintext` (two-byte integer). The length MUST NOT exceed 2^{14} .

seq_num_placeholder 8 bytes of `0xff`

Note "+" denotes concatenation.

5.1. Block Ciphers

The following MAC algorithm applies to block ciphers that do not use the Encrypt-then-MAC processing described in [[RFC7366](#)].

```
MAC(MAC_write_key,
    seq_num_placeholder +
    tls12_cid +
    cid_length +
    tls12_cid +
    DTLSCiphertext.version +
    epoch +
    sequence_number +
    cid +
    length_of_DTLSInnerPlaintext +
    DTLSInnerPlaintext.content +
    DTLSInnerPlaintext.real_type +
    DTLSInnerPlaintext.zeros
);
```

The rationale behind this construction is to separate the MAC input for DTLS without the connection ID from the MAC input with the connection ID. The former always consists of a sequence number followed by some other content type than `tls12_cid`; the latter always consists of the `seq_num_placeholder` followed by `tls12_cid`. Although $2^{64}-1$ is potentially a valid sequence number, `tls12_cid` will never be a valid content type when the connection ID is not in use. In addition, the `epoch` and `sequence_number` are now fed into the MAC in the same order as they appear on the wire.

5.2. Block Ciphers with Encrypt-then-MAC processing

The following MAC algorithm applies to block ciphers that use the Encrypt-then-MAC processing described in [[RFC7366](#)].

```
MAC(MAC_write_key,
    seq_num_placeholder +
    tls12_cid +
    cid_length +
    tls12_cid +
    DTLSCiphertext.version +
    epoch +
    sequence_number +
    cid +
    DTLSCiphertext.length +
    IV +
    ENC(content + padding + padding_length));
```


5.3. AEAD Ciphers

For ciphers utilizing authenticated encryption with additional data the following modification is made to the additional data calculation.

```
additional_data = seq_num_placeholder +  
                  tls12_cid +  
                  cid_length +  
                  tls12_cid +  
                  DTLSCiphertext.version +  
                  epoch +  
                  sequence_number +  
                  cid +  
                  length_of_DTLSInnerPlaintext;
```

6. Peer Address Update

When a record with a CID is received that has a source address different from the one currently associated with the DTLS connection, the receiver MUST NOT replace the address it uses for sending records to its peer with the source address specified in the received datagram, unless the following three conditions are met:

- *The received datagram has been cryptographically verified using the DTLS record layer processing procedures.
- *The received datagram is "newer" (in terms of both epoch and sequence number) than the newest datagram received. Reordered datagrams that are sent prior to a change in a peer address might otherwise cause a valid address change to be reverted. This also limits the ability of an attacker to use replayed datagrams to force a spurious address change, which could result in denial of service. An attacker might be able to succeed in changing a peer address if they are able to rewrite source addresses and if replayed packets are able to arrive before any original.
- *There is a strategy for ensuring that the new peer address is able to receive and process DTLS records. No strategy is mandated by this specification but see note (*) below.

The conditions above are necessary to protect against attacks that use datagrams with spoofed addresses or replayed datagrams to trigger attacks. Note that there is no requirement for use of the anti-replay window mechanism defined in Section 4.1.2.6 of DTLS 1.2. Both solutions, the "anti-replay window" or "newer" algorithm, will prevent address updates from replay attacks while the latter will only apply to peer address updates and the former applies to any application layer traffic.

Note that datagrams that pass the DTLS cryptographic verification procedures but do not trigger a change of peer address are still valid DTLS records and are still to be passed to the application.

(*) Note: Application protocols that implement protection against spoofed addresses depend on being aware of changes in peer addresses so that they can engage the necessary mechanisms. When delivered such an event, an application layer-specific address validation mechanism can be triggered, for example one that is based on successful exchange of a minimal amount of ping-pong traffic with the peer. Alternatively, an DTLS-specific mechanism may be used, as described in [[I-D.ietf-tls-dtls-rrc](#)].

DTLS implementations MUST silently discard records with bad MACs or that are otherwise invalid.

7. Examples

[Figure 4](#) shows an example exchange where a CID is used unidirectionally from the client to the server. To indicate that a zero-length CID is present in the "connection_id" extension we use the notation 'connection_id=empty'.

Client		Server
-----		-----
ClientHello (connection_id=empty)	----->	
	<-----	HelloVerifyRequest (cookie)
ClientHello (connection_id=empty) (cookie)	----->	
		ServerHello (connection_id=100) Certificate ServerKeyExchange CertificateRequest ServerHelloDone
	<-----	
Certificate ClientKeyExchange CertificateVerify [ChangeCipherSpec] Finished <CID=100>	----->	
		[ChangeCipherSpec] Finished
	<-----	
Application Data <CID=100>	=====>	
	<=====	Application Data

Legend:

<...> indicates that a connection id is used in the record layer
 (...) indicates an extension
 [...] indicates a payload other than a handshake message

Figure 4: Example DTLS 1.2 Exchange with CID

Note: In the example exchange the CID is included in the record layer once encryption is enabled. In DTLS 1.2 only one handshake message is encrypted, namely the Finished message. Since the example shows how to use the CID for payloads sent from the client to the

server, only the record layer payloads containing the Finished message or application data include a CID.

8. Privacy Considerations

The CID replaces the previously used 5-tuple and, as such, introduces an identifier that remains persistent during the lifetime of a DTLS connection. Every identifier introduces the risk of linkability, as explained in [[RFC6973](#)].

An on-path adversary observing the DTLS protocol exchanges between the DTLS client and the DTLS server is able to link the observed payloads to all subsequent payloads carrying the same ID pair (for bi-directional communication). Without multi-homing or mobility, the use of the CID exposes the same information as the 5-tuple.

With multi-homing, a passive attacker is able to correlate the communication interaction over the two paths. The lack of a CID update mechanism in DTLS 1.2 makes this extension unsuitable for mobility scenarios where correlation must be considered. Deployments that use DTLS in multi-homing environments and are concerned about these aspects SHOULD refuse to use CIDs in DTLS 1.2 and switch to DTLS 1.3 where a CID update mechanism is provided and sequence number encryption is available.

The specification introduces record padding for the CID-enhanced record layer, which is a privacy feature not available with the original DTLS 1.2 specification. Padding allows to inflate the size of the ciphertext making traffic analysis more difficult. More details about record padding can be found in Section 5.4 and Appendix E.3 of RFC 8446.

Finally, endpoints can use the CID to attach arbitrary per-connection metadata to each record they receive on a given connection. This may be used as a mechanism to communicate per-connection information to on-path observers. There is no straightforward way to address this concern with CIDs that contain arbitrary values. Implementations concerned about this aspect SHOULD refuse to use CIDs.

9. Security Considerations

An on-path adversary can create reflection attacks against third parties because a DTLS peer has no means to distinguish a genuine address update event (for example, due to a NAT rebinding) from one that is malicious. This attack is of particular concern when the request is small and the response large. See [Section 6](#) for more on address updates.

Additionally, an attacker able to observe the data traffic exchanged between two DTLS peers is able to replay datagrams with modified IP address/port numbers.

The topic of peer address updates is discussed in [Section 6](#).

10. IANA Considerations

This document requests three actions from IANA.

10.1. Extra Column to TLS ExtensionType Values Registry

IANA is requested to add an extra column named "DTLS-Only" to the "TLS ExtensionType Values" registry to indicate whether an extension is only applicable to DTLS and to include this document as an additional reference for the registry.

10.2. Entry to the TLS ExtensionType Values Registry

IANA is requested to allocate an entry to the existing "TLS ExtensionType Values" registry, for connection_id(TBD1) as described in the table below. Although the value 53 has been allocated by early allocation for a previous version of this document, it is incompatible with this document. Once this document is approved for publication, the early allocation will be deprecated in favor of this assignment.

Value	Extension Name	TLS 1.3	DTLS-Only	Recommended	Reference
TBD1	connection_id	CH, SH	Y	N	[[This doc]]

A new column "DTLS-Only" is added to the registry. The valid entries are "Y" if the extension is only applicable to DTLS, "N" otherwise. All the pre-existing entries are given the value "N".

Note: The value "N" in the Recommended column is set because this extension is intended only for specific use cases. This document describes the behavior of this extension for DTLS 1.2 only; it is not applicable to TLS, and its usage for DTLS 1.3 is described in [\[I-D.ietf-tls-dtls13\]](#).

10.3. Entry to the TLS ContentType Registry

IANA is requested to allocate tls12_cid(TBD2) in the "TLS ContentType" registry. The tls12_cid ContentType is only applicable to DTLS 1.2.

11. References

11.1. 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>>.

[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>>.

[RFC7366]

Gutmann, P., "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7366, DOI 10.17487/RFC7366, September 2014, <<https://www.rfc-editor.org/info/rfc7366>>.

[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>>.

[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>>.

11.2. Informative References

[I-D.ietf-tls-dtls-rrc]

Tschafenig, H. and T. Fossati, "Return Routability Check for DTLS 1.2 and DTLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls-rrc-00, 9 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-dtls-rrc-00.txt>>.

[I-D.ietf-tls-dtls13]

Rescorla, E., Tschafenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-dtls13-43.txt>>.

[RFC6973]

Cooper, A., Tschafenig, 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>>.

Appendix A. History

RFC EDITOR: PLEASE REMOVE THE THIS SECTION

draft-ietf-tls-dtls-connection-id-12

*Improved peer address update text

- *Editorial improvements

- *Clarification regarding the use of the TLS ExtensionType Values Registry

draft-ietf-tls-dtls-connection-id-11

- *Enhanced IANA considerations section

- *Clarifications regarding CID negotiation and zero-length CIDs

draft-ietf-tls-dtls-connection-id-10

- *Clarify privacy impact.

- *Have security considerations point to [Section 6](#).

draft-ietf-tls-dtls-connection-id-09

- *Changed MAC/additional data calculation.

- *Disallow sending MAC failure fatal alerts to non-validated peers.

- *Incorporated editorial review comments by Ben Kaduk.

draft-ietf-tls-dtls-connection-id-08

- *RRC draft moved from normative to informative.

draft-ietf-tls-dtls-connection-id-07

- *Wording changes in the security and privacy consideration and the peer address update sections.

draft-ietf-tls-dtls-connection-id-06

- *Updated IANA considerations

- *Enhanced security consideration section to describe a potential man-in-the-middle attack concerning address validation.

draft-ietf-tls-dtls-connection-id-05

- *Restructured Section 5 "Record Payload Protection"

draft-ietf-tls-dtls-connection-id-04

- *Editorial simplifications to the 'Record Layer Extensions' and the 'Record Payload Protection' sections.

- *Added MAC calculations for block ciphers with and without Encrypt-then-MAC processing.

draft-ietf-tls-dtls-connection-id-03

- *Updated list of contributors
- *Updated list of contributors and acknowledgements
- *Updated example
- *Changed record layer design
- *Changed record payload protection
- *Updated introduction and security consideration section
- *Author- and affiliation changes

draft-ietf-tls-dtls-connection-id-02

- *Move to internal content types a la DTLS 1.3.

draft-ietf-tls-dtls-connection-id-01

- *Remove 1.3 based on the WG consensus at IETF 101

draft-ietf-tls-dtls-connection-id-00

- *Initial working group version (containing a solution for DTLS 1.2 and 1.3)

draft-rescorla-tls-dtls-connection-id-00

- *Initial version

Appendix B. Working Group Information

RFC EDITOR: PLEASE REMOVE THE THIS SECTION

The discussion list for the IETF TLS working group is located at the e-mail address tls@ietf.org. Information on the group and information on how to subscribe to the list is at <https://www1.ietf.org/mailman/listinfo/tls>

Archives of the list can be found at: <https://www.ietf.org/mail-archive/web/tls/current/index.html>

Appendix C. Contributors

Many people have contributed to this specification, and we would like to thank the following individuals for their contributions:

- * Yin Xinxing
Huawei
yinxinxing@huawei.com
- * Nikos Mavrogiannopoulos
RedHat
nmav@redhat.com
- * Tobias Gondrom
tobias.gondrom@gondrom.org

Additionally, we would like to thank the Connection ID task force team members:

- *Martin Thomson (Mozilla)
- *Christian Huitema (Private Octopus Inc.)
- *Jana Iyengar (Google)
- *Daniel Kahn Gillmor (ACLU)
- *Patrick McManus (Mozilla)
- *Ian Swett (Google)
- *Mark Nottingham (Fastly)

The task force team discussed various design ideas, including cryptographically generated session ids using hash chains and public key encryption, but dismissed them due to their inefficiency. The approach described in this specification is the simplest possible design that works given the limitations of DTLS 1.2. DTLS 1.3 provides better privacy features and developers are encouraged to switch to the new version of DTLS.

Appendix D. Acknowledgements

We would like to thank Hanno Becker, Martin Duke, Lars Eggert, Ben Kaduk, Warren Kumari, Francesca Palombini, Tom Petch, John Scudder, Sean Turner, Eric Vyncke, and Robert Wilton for their review comments.

Finally, we want to thank the IETF TLS working group chairs, Chris Wood, Joseph Salowey, and Sean Turner, for their patience, support and feedback.

Authors' Addresses

Eric Rescorla (editor)
RTFM, Inc.

Email: ekr@rtfm.com

Hannes Tschofenig (editor)
Arm Limited

Email: hannes.tschofenig@arm.com

Thomas Fossati
Arm Limited

Email: thomas.fossati@arm.com

Achim Kraus
Bosch.IO GmbH

Email: achim.kraus@bosch.io