Transport Layer Security
Internet-Draft
Intended status: Standards Track
Expires: 9 October 2026

J. Preu Mattsson
Ericsson
H. Tschofenig
UniBw M.
M. Txen
Mnster Univ. of Applied Sciences

7 April 2026

Large Record Sizes for TLS and DTLS with Reduced Overhead
draft-ietf-tls-super-jumbo-record-limit-03

Abstract

TLS 1.3 records limit the inner plaintext (TLSInnerPlaintext) size to 2^14 + 1 bytes, which includes one byte for the content type. DTLS 1.3 uses the same plaintext size limit. This document defines a TLS extension that allows endpoints to advertise larger per-direction maximum inner plaintext sizes, up to 2^30 - 256 bytes, while reducing overhead in TLS 1.3 and DTLS 1.3 record headers.

About This Document

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

The latest revision of this draft can be found at https://tlswg.github.io/super-jumbo-record-limit/draft-ietf-tls-super-jumbo-record-limit.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-tls-super-jumbo-record-limit/.

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

Source for this draft and an issue tracker can be found at https://github.com/tlswg/super-jumbo-record-limit.

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/.

Copyright Notice

Table of Contents

1. Introduction

TLS 1.3 records limit the inner plaintext (TLSInnerPlaintext) size to 2^14 + 1 bytes, which includes one byte for the content type. Records also have a 3-byte overhead due to the fixed opaque_type and legacy_record_version fields. TLS-based protocols are increasingly used to secure long-lived interfaces in critical infrastructure, such as telecommunication networks. In some infrastructure use cases, the upper layer of DTLS expects a message oriented service and uses message sizes much larger than 2^14-bytes. In these cases, the 2^14-byte limit in TLS necessitates an additional protocol layer for fragmentation, resulting in increased CPU and memory consumption and

additional complexity. Allowing 2^30-byte records would eliminate additional fragmentation in almost all use cases. In [RFC6083] (DTLS over SCTP), the 2^14-byte limit is a severe restriction.

This document defines a "large_record_size_limit" extension. The extension is negotiated during the handshake, and each endpoint independently advertises the maximum inner plaintext (TLSInnerPlaintext) size it is willing to receive. Therefore, the two traffic directions can use different limits. This extension is valid in TLS 1.3 and DTLS 1.3. The extension works similarly to the "record_size_limit" extension defined in [RFC8449]. Additionally, this document defines new TLS 1.3 TLSLargeCiphertext and DTLS 1.3 unified_hdr structures to enable inner plaintexts up to 2^30 - 256 bytes with reduced overhead. For example, ciphertexts up to 64 bytes can be supported with 4 bytes less overhead and ciphertexts up to 2^14 bytes can be supported with 3 bytes less overhead, which is useful in constrained IoT environments. The "large_record_size_limit" extension is incompatible with middleboxes expecting TLS 1.2 records.

2. 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.

3. The "large_record_size_limit" Extension

The ExtensionData of the "large_record_size_limit" extension is LargeRecordSizeLimit:

```
uint32 LargeRecordSizeLimit;
```

LargeRecordSizeLimit denotes the maximum size, in bytes, of inner plaintexts that the endpoint is willing to receive. It includes the content type and padding (i.e., the complete length of TLSInnerPlaintext). AEAD expansion is not included. This is the same value as RecordSizeLimit negotiated in the "record_size_limit" extension [RFC8449].

The large record size limit only applies to records sent toward the endpoint that advertises the limit. An endpoint can send records that are larger than the limit it advertises as its own limit. A TLS endpoint that receives a record larger than its advertised limit MUST generate a fatal "record_overflow" alert; a DTLS endpoint that receives a record larger than its advertised limit SHOULD discard the

record and SHOULD NOT generate a fatal "record_overflow" alert. An endpoint MUST NOT add padding to records that would cause the length of TLSInnerPlaintext to exceed the limit advertised by the other endpoint.

Endpoints MUST NOT send a "large_record_size_limit" extension with a value smaller than 64 or larger than 2^30 - 256. An endpoint MUST treat receipt of a smaller or larger value as a fatal error and generate an "illegal_parameter" alert.

The server sends the "large_record_size_limit" extension in the EncryptedExtensions message. During resumption, the limit is renegotiated. Records are subject to the limits that were set in the handshake that produces the keys that are used to protect those records. This admits the possibility that the extension might not be negotiated during resumption. If the extension is not negotiated in a subsequent handshake, records protected with keys from that handshake are not subject to "large_record_size_limit" and are instead subject to the record size limits and record formats defined by TLS 1.3 [RFC8446bis] and DTLS 1.3 [RFC9147], unless another negotiated extension specifies otherwise.

Unprotected messages and records protected with early_traffic_secret or handshake_traffic_secret are not subject to the large record size limit and remain subject to the record size limits and record formats defined by TLS 1.3 [RFC8446bis] and DTLS 1.3 [RFC9147]. In particular, these records MUST use TLSCiphertext (TLS) or the DTLSCiphertext unified_hdr length encoding from [RFC9147] (DTLS), rather than TLSLargeCiphertext or the varuint length encoding defined in this document.

When the "large_record_size_limit" extension is negotiated:

* All TLS 1.3 records protected with application traffic keys (derived from application_traffic_secret_N) MUST use the TLSLargeCiphertext structure instead of the TLSCiphertext structure.

  Instead of using a fixed-length field, this specification defines a variable-length unsigned integer type, referred to as varuint, as specified in Section 2.1.2 of [RFC9420]. The varuint encoding is similar to the variable-length integer encoding defined in Section 16 of [RFC9000], but requires minimum-size encoding. As defined in Section 2.1.2 of [RFC9420], the two most significant bits of the first byte indicate the base 2 logarithm of the integer encoding length in bytes. The remaining bits encode the integer value in network byte order. The encoded representation MUST use the smallest number of octets necessary to represent the

integer value to ensure a unique wire representation. When decoding, any value that uses more octets than necessary is an invalid length encoding and MUST be treated as if the record exceeded the peer's advertised record size limit. This means that integers are encoded in 1, 2, or 4 bytes and can encode 6-, 14-, or 30-bit values, respectively. The varuint type is defined only for the record length fields specified in this document. Table 1 summarizes the encoding properties from Section 2.1.2 of [RFC9420].

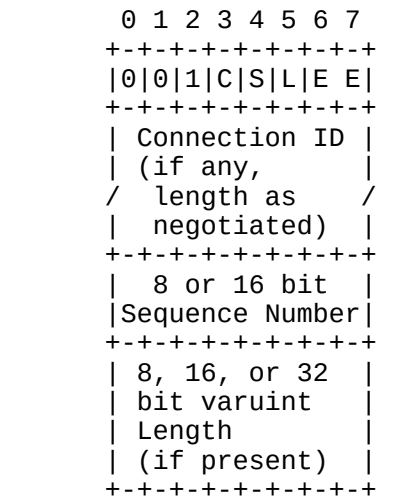
```
struct {
    varuint length;
    opaque encrypted_record[TLSLargeCiphertext.length];
} TLSLargeCiphertext;
```

Prefix	Length	Usable Bits	Min	Max
00	1	6	0	63
01	2	14	64	16383
10	4	30	16384	1073741823
11	invalid	-	-	-

Table 1: Summary of varuint Encodings.

If the first two bits of the length field are 11, the encoded length is invalid and MUST be treated as if the record exceeded the peer's advertised record size limit.

* All DTLS 1.3 records protected with application traffic keys (derived from application_traffic_secret_N) and with length present MUST use a unified_hdr structure with a varuint length equal to the TLS 1.3 length field defined above.



* An endpoint MAY generate records protected with application traffic keys (derived from application_traffic_secret_N) with inner plaintext length that is equal to or smaller than the LargeRecordSizeLimit value it receives from its peer. An endpoint MUST NOT generate a protected record with inner plaintext length that is larger than the LargeRecordSizeLimit value it receives from its peer.

The "large_record_size_limit" extension is not compatible with middleboxes expecting TLS 1.2 records and SHOULD NOT be negotiated where such middleboxes are expected. Endpoints that support this specification SHOULD prefer "large_record_size_limit" over "record_size_limit" and "max_fragment_length", and a client SHOULD offer at most one of these extensions. A server MUST NOT send extension responses to more than one of "large_record_size_limit", "record_size_limit", and "max_fragment_length". A client MUST treat receipt of more than one of "large_record_size_limit", "record_size_limit", and "max_fragment_length" as a fatal error, and it SHOULD generate an "illegal_parameter" alert.

The Path Maximum Transmission Unit (PMTU) in DTLS also limits the size of records. The record size limit does not affect PMTU discovery and SHOULD be set independently. The record size limit is fixed during the handshake and so should be set based on constraints at the endpoint and not based on the current network environment. In comparison, the PMTU is determined by the network path and can change dynamically over time.

4. Limits on Key Usage

TLS 1.3 [RFC8446bis] and DTLS 1.3 [RFC9147] limit the number of full-size records that may be encrypted under a given set of keys. Increasing the maximum inner plaintext size to more than 2^14 bytes while keeping the same confidentiality and integrity advantage per write key therefore requires lower AEAD limits. When "large_record_size_limit" has been negotiated with a record size limit larger than 2^14 + 1 bytes, existing AEAD limits SHALL be decreased by a factor of (LargeRecordSizeLimit) / (2^14). For AES-GCM, usage against the confidentiality limit is block-based: each protected record consumes ceil(record_plaintext_length / 16) * 16 bytes. For example, when AES-GCM is used in TLS 1.3 [RFC8446bis] with a 64 kB record limit, only around 2^22.5 full-size records (about 6 million) may be encrypted under a given set of keys. For ChaCha20/Poly1305, the record sequence number would still wrap before the safety limit is reached.

5. Security Considerations

Large record sizes might require more memory allocation for senders and receivers. Additionally, larger record sizes also means that more processing is done before verification of non-authentic records fails. TLS implementations MUST NOT provide access to the decrypted message content until after its integrity is confirmed.

The use of larger record sizes can either simplify or complicate traffic analysis, depending on the application. The LargeRecordSizeLimit is just an upper limit and it is still the sender that decides the size of the inner plaintexts up to that limit.

6. IANA Considerations

IANA is requested to assign a new value in the TLS ExtensionType Values registry defined by [RFC8447]:

* The Extension Name should be large_record_size_limit
* The TLS 1.3 value should be CH, EE
* The DTLS-Only value should be N
* The Recommended value should be Y
* The Reference should be this document

7. References

7.1. 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/rfc/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/rfc/rfc8174>.

[RFC8446bis] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-14, 13 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-rfc8446bis-14>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/rfc/rfc8447>.

[RFC8449] Thomson, M., "Record Size Limit Extension for TLS", RFC 8449, DOI 10.17487/RFC8449, August 2018, <https://www.rfc-editor.org/rfc/rfc8449>.

[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/rfc/rfc9147>.

[RFC9420] Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", RFC 9420, DOI 10.17487/RFC9420, July 2023, <https://www.rfc-editor.org/rfc/rfc9420>.

7.2. Informative References

[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/rfc/rfc6083>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

Change Log

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

Changes from -02 to -03:

* Addressed WGLC comments from Magnus Westerlund, Ilari Liusvaara,Valery Smyslov, Eric Rescorla and Martin Thomson.

Changes from -01 to -02:

* Variable length field equal to the one defined in MLS

* Clarification that the extension value is equal to RFC8449

* Clarification and corrections on AEAD limits

Changes from -00 to -01:

* Keep alive

Changes from -05 to -00:

* WG adoption

Changes from -04 to -05:

* Grammar and comprehension tweaks.

* Added change log

Changes from -03 to -04:

* Corrected uint24 to uint32.

Changes from -02 to -03:

* Major rewrite based on discussions at IETF 119

* New independent extension instead of flag extension used together with record_size_limit

* New record format without opaque_type and legacy_record_version fields. This reduces overhead

* Support inner plaintext size up to 2^32 - 256 bytes

Acknowledgments

The authors would like to thank Richard Barnes, Stephen Farrell, Benjamin Kaduk, Colm MacCrthaigh, Eric Rescorla, Benjamin Schwartz, Ira McDonald, Magnus Westerlund, Ilari Liusvaara, Valery Smyslov and Martin Thomson for their valuable comments and feedback. Some of the text were inspired by and borrowed from [RFC8449].

We would also like to thank our TLS working group chairs for their support.

Authors' Addresses

John Preu Mattsson
Ericsson
Email: john.mattsson@ericsson.com

Hannes Tschofenig
University of the Bundeswehr Munich
85577 Neubiberg
Germany
Email: hannes.tschofenig@gmx.net

Michael Txen
Mnster Univ. of Applied Sciences
Email: tuexen@fh-muenster.de