

Messaging Layer Security
Internet-Draft
Intended status: Informational
Expires: 17 April 2026

R. Mahy
14 October 2025

New Content Types for Messaging Layer Security (MLS)
draft-mahy-mls-new-content-types-00

Abstract

This Messaging Layer Security (MLS) extensions adds two new variations of the application content type, each with a separate key ratchet. It also creates an MLS capability to negotiate use of the new types, and an IANA registry to register additional content types.

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://rohanmahy.github.io/mls-new-content-types/draft-mahy-mls-new-content-types.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-mahy-mls-new-content-types/>.

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

Source for this draft and an issue tracker can be found at <https://github.com/rohanmahy/mls-new-content-types>.

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 17 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Conventions and Definitions	3
3. Negotiating Support	3
4. Sending and Receiving	4
5. Security Considerations	6
6. IANA Considerations	6
6.1. MLS Extension Types	6
6.1.1. supported_content_types MLS Extension	6
6.1.2. required_content_types MLS Extension	7
6.2. MLS Content Types	7
7. References	8
7.1. Normative References	8
7.2. Informative References	9
Acknowledgments	9
Author's Address	9

1. Introduction

Some messaging protocols (ex: XMPP [RFC6120]) make a distinction between regular messages--where each message is relevant, and status or "presence" messages--where only the most recent update per sender is relevant. In addition, some messages may have a sufficiently short relevance (for example, typing notifications) that they can be discarded if the receiver is offline. In large messaging systems with lots of updates, optimizing decryption of such messages, and optionally suppressing delivery of irrelevant message can result in improved performance.

This document defines two new MLS [RFC9420] content types: status and ephemeral. These largely act like the application content type, but the new content types each maintain distinct key ratchets in the secret tree. Only the most recent status message from each sender needs to be decrypted. Only ephemeral messages received within a small amount of time (ex: 10 seconds) are relevant, and of those only the most recent from each sender.

This allows an application to fast-forward over generations that contain irrelevant messages.

2. Conventions and Definitions

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. Negotiating Support

If a client supports the mechanism in this document, it adds a `supported_content_types` extension to its `LeafNode.Capabilities.ExtensionTypes` with the specific non-default content types it supports (for example, status and/or ephemeral in this specification).

If an MLS group `GroupContext.RequiredCapabilities.extension_types` contains a `required_content_types` extension, every member of the MLS group MUST be prepared to receive messages with any of the (non-default) content types listed.

It also redefines the `ContentType` enum as shown below.

```

enum {
    reserved(0),
    application(1),
    proposal(2),
    commit(3),
    status(4),
    ephemeral(5),
    (255)
} ContentType;

struct {
    ContentType content_types<V>;
} ContentTypes;

ContentTypes supported_content_types;
ContentTypes required_content_types;

```

4. Sending and Receiving

If a group lists a specific content type in its `required_content_types` as described in the previous section, a member MAY send an MLS `PrivateMessage` with that content type.

The construction of the `PrivateMessage` is the same as for sending an application message, except that the per-sender ratchet is used derived from the relevant content type, as shown in the figure below, which replaces Figure 26 of [RFC9420], and the new version of three structs defined later in this section (`FramedContent`, `FramedContentAuthData`, and `PrivateMessgeContent`) replace those defined in [RFC9420]:

```

tree_node_[N]_secret
|
|
+--> ExpandWithLabel(., "handshake", "", KDF.Nh)
|   = handshake_ratchet_secret_[N]_[0]
|
+--> ExpandWithLabel(., "application", "", KDF.Nh)
|   = application_ratchet_secret_[N]_[0]
|
+--> ExpandWithLabel(., "status", "", KDF.Nh)
|   = status_ratchet_secret_[N]_[0]
|
+--> ExpandWithLabel(., "ephemeral", "", KDF.Nh)
|   = ephemeral_ratchet_secret_[N]_[0]

```

Figure 1: Initialization of the Hash Ratchets from the Leaves of a Secret Tree

```
struct {
    opaque group_id<V>;
    uint64 epoch;
    Sender sender;
    opaque authenticated_data<V>;

    ContentType content_type;
    select (FramedContent.content_type) {
        case application:
        case status:
        case ephemeral:
            opaque application_data<V>;
        case proposal:
            Proposal proposal;
        case commit:
            Commit commit;
    };
} FramedContent;

struct {
    /* SignWithLabel(.., "FramedContentTBS", FramedContentTBS) */
    opaque signature<V>;
    select (FramedContent.content_type) {
        case commit:
            /*
                MAC(confirmation_key,
                    GroupContext.confirmed_transcript_hash)
            */
            MAC confirmation_tag;
        case application:
        case status:
        case ephemeral:
        case proposal:
            struct{};
    };
} FramedContentAuthData;

struct {
    select (PrivateMessage.content_type) {
        case application:
        case status:
        case ephemeral:
            opaque application_data<V>;

        case proposal:
            Proposal proposal;

        case commit:
```

```
        Commit commit;
    };

    FramedContentAuthData auth;
    opaque padding[length_of_padding];
} PrivateMessageContent;
```

All clients in a group need to agree on the "maximum number of steps that clients will move a secret tree ratchet forward in response to a single message before rejecting it" as described in Section 7 of [RFC9750]. If a client is about to exhaust that number of steps for its own status or ephemeral ratchet, it **MUST** send a new commit.

On receipt of a PrivateMessage with a supported, non-default content type, the receiver likewise decrypts the message using the relevant ratchet.

5. Security Considerations

TODO Security

6. IANA Considerations

This document requests the addition of various new values under the heading of "Messaging Layer Security". Each registration is organized under the relevant registry Type.

This document also requests the creation of a new MLS Content Types registry as described in Section 6.2.

RFC EDITOR: Please replace XXXX throughout with the RFC number assigned to this document.

6.1. MLS Extension Types

6.1.1. supported_content_types MLS Extension

The supported_content_types MLS Extension Type is used inside LeafNode objects. It contains a list of non-default ContentTypes supported by the client node.

Value: 0x0009 (suggested)

Name: supported_content_types

Message(s): LN: This extension may appear in LeafNode objects

Recommended: Y

Reference: RFC XXXX

6.1.2. required_content_types MLS Extension

The required_content_types MLS Extension Type is used inside GroupContext objects. It contains a list of non-default ContentTypes that are mandatory for all MLS members of the group to support.

Value: 0x000a (suggested)

Name: required_content_types

Message(s): GC: This extension may appear in GroupContext objects

Recommended: Y

Reference: RFC XXXX

6.2. MLS Content Types

This document requests the creation of a new IANA "MLS Content Types" registry under the "Messaging Layer Security" group registry heading. Assignments are via the Specification Required policy [RFC8126] using the MLS Designated Experts.

Template:

- * Value: The numeric value of the component ID
- * Name: The name of the component
- * Recommended: Same as in Section 17.1 of [RFC9420]
- * Reference: The document where this content type is defined

Initial Contents:

Value	Name	R	Ref
0x00	RESERVED	-	RFC9420
0x01	application	Y	RFC9420
0x02	proposal	Y	RFC9420
0x03	commit	Y	RFC9420
0x04	status	Y	RFCXXXX
0x05	ephemeral	Y	RFCXXXX
0x06-			
0xff	UNASSIGNED	-	RFC9420

Table 1

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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/rfc/rfc8126>>.
- [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>>.
- [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>>.

- [RFC9750] Beurdouche, B., Rescorla, E., Omara, E., Inguva, S., and A. Duric, "The Messaging Layer Security (MLS) Architecture", RFC 9750, DOI 10.17487/RFC9750, April 2025, <<https://www.rfc-editor.org/rfc/rfc9750>>.

7.2. Informative References

- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <<https://www.rfc-editor.org/rfc/rfc6120>>.

Acknowledgments

TODO acknowledge.

Author's Address

Rohan Mahy
Email: rohan.ietf@gmail.com