Messaging Layer Security
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

R. Robert
Phoenix R&D GmbH

2 March 2026

Messaging Layer Security (MLS) Targeted Messages
draft-ietf-mls-targeted-messages-00

Abstract

This document defines targeted messages for the Messaging Layer Security (MLS) protocol. A targeted message allows a member of an MLS group to send an encrypted and authenticated message to another member of the same group without creating a new group. The mechanism reuses Hybrid Public Key Encryption (HPKE) and the MLS key schedule to provide confidentiality, authentication, and binding to the group state.

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://raphaelrobert.github.io/mls-targeted-messages/draft-robert-mls-targeted-messages.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-mls-targeted-messages/.

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/raphaelrobert/mls-targeted-messages.

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 3 September 2026.

Copyright Notice

Copyright (c) 2026 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

MLS application messages make sending encrypted messages to all group members easy and efficient. Sometimes application protocols require that a group member sends a message only to specific members of the same group, either for privacy or for efficiency reasons.

Targeted messages are a way to achieve this without having to create a new group with the sender and the specific recipients, which might not be possible or desired. Instead, this document defines the format and encryption of a message that is sent from a member of an existing group to another member of that group.

The goal is to provide a one-shot messaging mechanism offering confidentiality and authentication, reusing mechanisms from [RFC9420] and [RFC9180]. Targeted messages can be used as a building block for more complex messaging protocols.

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. Format

This document defines the mls_targeted_message WireFormat, where the content is a TargetedMessage.

A TargetedMessage is carried in the MLSMessage envelope defined in Section 6 of [RFC9420]:

```
case mls_targeted_message:
    TargetedMessage targeted_message;
```

```
struct {
  opaque group_id<V>;
  uint64 epoch;
  uint32 recipient_leaf_index;
  opaque authenticated_data<V>;
  opaque encrypted_sender_auth_data<V>;
  opaque ciphertext<V>;
} TargetedMessage;

struct {
  uint32 sender_leaf_index;
  opaque signature<V>;
  opaque kem_output<V>;
} TargetedMessageSenderAuthData;

struct {
  opaque group_id<V>;
  uint64 epoch;
  uint32 recipient_leaf_index;
  opaque authenticated_data<V>;
  TargetedMessageSenderAuthData sender_auth_data;
} TargetedMessageTBM;

struct {
  ProtocolVersion version = mls10;
  WireFormat wire_format = mls_targeted_message;
  opaque group_id<V>;
  uint64 epoch;
  uint32 recipient_leaf_index;
  opaque authenticated_data<V>;
  uint32 sender_leaf_index;
  opaque kem_output<V>;
} TargetedMessageTBS;

struct {
  opaque group_id<V>;
  uint64 epoch;
  opaque label<V> = "MLS 1.0 targeted message psk";
} PSKId;

struct {
  opaque application_data<V>;
  opaque padding[length_of_padding];
} TargetedMessageContent;
```

4. Authentication

A targeted message is authenticated by the sender's signature. The sender uses the signature key of its LeafNode. The signature scheme is determined by the cipher suite of the MLS group. The signature is computed over the serialized TargetedMessageTBS struct and is included in the TargetedMessageSenderAuthData.signature field:

```
signature = SignWithLabel(sender_leaf_node_signature_private_key,
              "TargetedMessageTBS", targeted_message_tbs)
```

The recipient MUST verify the signature:

```
VerifyWithLabel(sender_leaf_node.signature_key,
                "TargetedMessageTBS",
                targeted_message_tbs,
                signature)
```

In addition, targeted messages are authenticated using a pre-shared key (PSK), exported through the MLS exporter for the epoch specified in the TargetedMessage:

```
targeted_message_psk =
  MLS-Exporter("targeted message", "psk", KDF.Nh)
```

The targeted_message_psk is used as the psk parameter in the Hybrid Public Key Encryption (HPKE) encryption. The corresponding psk_id parameter is the serialized PSKId struct.

4.1. Additional Authenticated Data (AAD)

Targeted messages can include additional authenticated data (AAD) in the TargetedMessage.authenticated_data field. This field is used to carry application-specific data that is authenticated but not encrypted. The AAD is included in the TargetedMessageTBM struct.

5. Encryption

Targeted messages use HPKE to encrypt the message content to a specific group member.

Unlike the HPKE Base mode used in [RFC9420], targeted messages use HPKE PSK mode (Section 5.1.3 of [RFC9180]). The PSK is derived from the MLS group key schedule, binding the encryption to the group state and providing authentication that the sender holds the group's PSK.

5.1. Padding

The TargetedMessageContent.padding field is set by the sender, by first encoding the application data and then appending the chosen number of zero bytes. A receiver identifies the padding field in a plaintext decoded from TargetedMessage.ciphertext by first decoding the application data; then the padding field comprises any remaining octets of plaintext. The padding field MUST be filled with all zero bytes. A receiver MUST verify that there are no non-zero bytes in the padding field, and if this check fails, the enclosing TargetedMessage MUST be rejected as malformed. This check ensures that the padding process is deterministic, so that, for example, padding cannot be used as a covert channel.

5.2. Application Data Encryption

The TargetedMessageContent struct is serialized and encrypted using HPKE.

The HPKE context is a TargetedMessageContext struct with the following content, where group_context is the serialized context of the MLS group:

```
struct {
  opaque label<V>;
  opaque context<V>;
} TargetedMessageContext;

label = "MLS 1.0 TargetedMessageData"
context = group_context
```

The TargetedMessageContext struct follows the same structure as EncryptContext in Section 5.1.3 of [RFC9420], but uses PSK mode rather than Base mode.

The TargetedMessageContext struct is serialized as hpke_context and is used by both the sender and the recipient. The recipient’s leaf node HPKE encryption key from the MLS group is used as the recipient’s public key recipient_node_public_key for the HPKE encryption.

The TargetedMessageTBM struct is serialized as targeted_message_tbm, and is used as the aad parameter for the HPKE encryption.

The sender computes TargetedMessageSenderAuthData.kem_output and TargetedMessage.ciphertext:

```
(kem_output, ciphertext) = SealPSK(
                                        /* pkR */
                                        recipient_node_public_key,
                                        /* info */
                                        hpke_context,
                                        /* aad */
                                        targeted_message_tbm,
                                        /* pt */
                                        targeted_message_content,
                                        /* psk */
                                        targeted_message_psk,
                                        /* psk_id */
                                        psk_id)
```

The recipient decrypts the content as follows:

```
targeted_message_content = OpenPSK(kem_output,
                  recipient_node_private_key,
                  hpke_context,
                  targeted_message_tbm,
                  ciphertext,
                  targeted_message_psk,
                  psk_id)
```

The functions SealPSK and OpenPSK are defined in [RFC9180].

5.3. Sender Data Encryption

TargetedMessageSenderAuthData is encrypted similarly to MLSSenderData as described in Section 6.3.2 of [RFC9420]. It contains the sender's leaf index, the signature over TargetedMessageTBS, and the Key Encapsulation Mechanism (KEM) output of the HPKE encryption.

The key and nonce provided to the Authenticated Encryption with Associated Data (AEAD) are computed as the Key Derivation Function (KDF) of the first KDF.Nh bytes of the ciphertext generated in Section 5.2. If the length of the ciphertext is less than KDF.Nh, the whole ciphertext is used. In pseudocode, the key and nonce are derived as:

```
sender_auth_data_secret =
  MLS-Exporter("targeted message", "sender auth data secret", KDF.Nh)

ciphertext_sample = ciphertext[0..KDF.Nh-1]

sender_auth_data_key = ExpandWithLabel(sender_auth_data_secret,
                           "key", ciphertext_sample, AEAD.Nk)
sender_auth_data_nonce = ExpandWithLabel(sender_auth_data_secret,
                             "nonce", ciphertext_sample, AEAD.Nn)
```

The Additional Authenticated Data (AAD) for the encrypted_sender_auth_data ciphertext is the first three fields of TargetedMessage:

```
struct {
  opaque group_id<V>;
  uint64 epoch;
  uint32 recipient_leaf_index;
} SenderAuthDataAAD;
```

6. Recipient Validation

Upon receiving a TargetedMessage, the recipient MUST perform the following validation steps:

* Verify that group_id matches a group the recipient is a member of.

* Verify that epoch corresponds to the current epoch or a past epoch for which the recipient still has the necessary key material.

* Verify that recipient_leaf_index matches the recipient's own leaf index in the specified epoch.

* After decrypting TargetedMessageSenderAuthData, verify that sender_leaf_index refers to a non-blank leaf in the ratchet tree of the specified epoch.

* Verify the signature as described in Section 4.

The recipient MUST NOT process or act on the decrypted TargetedMessageContent until all of the above validation steps have completed successfully. In particular, the content MUST NOT be passed to the application before the sender's signature has been verified.

If any of these checks fail, the TargetedMessage MUST be rejected.

7. Security Considerations

This section describes the security properties of targeted messages and their limitations relative to MLS application messages [RFC9420].

7.1. Authentication

Targeted messages use two complementary authentication mechanisms. The sender's signature (Section 4) binds the message to the sender's identity: the recipient verifies the signature against the sender's LeafNode signature key, confirming that the holder of that key produced the message. The PSK exported from the group key schedule provides a second layer that proves group membership. Per Section 9.1 of [RFC9180], HPKE PSK mode provides outsider authentication, ensuring that an entity that does not know the PSK cannot forge a valid ciphertext. It does not, however, authenticate which PSK holder produced the message. Sender identity relies entirely on the signature.

Because the PSK is derived from the MLS key schedule, it is only valid for a specific group and epoch. The Forward Secrecy and Post-Compromise Security guarantees of the group key schedule therefore extend to targeted messages. The PSK also ensures that an attacker needs access to the private group state in addition to the HPKE and signature private keys, improving confidentiality guarantees against passive attackers and authentication guarantees against active attackers.

7.2. Signature Verification Before Processing

The decryption flow necessarily produces plaintext before the signature can be verified: the recipient first decrypts the sender authentication data (Section 5.3) to obtain the sender's leaf index, KEM output, and signature, then decrypts the HPKE ciphertext to obtain the TargetedMessageContent, and finally verifies the signature over the TargetedMessageTBS structure.

Because the PSK is shared among all group members and each member's HPKE public key is available in the ratchet tree, any group member can construct HPKE ciphertext that decrypts successfully while claiming a different sender identity. The signature is the sole mechanism that binds the message to the claimed sender. Section 6 requires that the recipient not act on the decrypted content until signature verification has succeeded.

7.3. Forward Secrecy

Targeted messages encrypt directly to the recipient's leaf node HPKE encryption key. Unlike application messages in [RFC9420], which derive per-message keys from a secret tree, targeted messages have no per-message key derivation. Compromising the recipient's leaf private key therefore exposes all targeted messages encrypted to that key within the epoch. Forward secrecy is at epoch granularity only: it depends on the key schedule advancing to a new epoch and on the recipient deleting the previous epoch's leaf private key. Applications that require stronger forward secrecy guarantees SHOULD advance the epoch frequently.

7.4. Sender Identity Confidentiality

The sender_auth_data_secret used to encrypt the TargetedMessageSenderAuthData is derived from the MLS exporter (Section 5.3) and is available to all group members. Sender identity is therefore protected from the Delivery Service and from entities outside the group, but not from other group members who obtain the encrypted message.

7.5. Replay Protection

Targeted messages do not include a generation counter or nonce at the protocol level. A captured targeted message can therefore be replayed within the same epoch and will pass all validation checks. However, targeted messages are a stateless one-shot mechanism: replaying a message causes the recipient to see duplicate content but does not change any group state. Applications that require replay detection SHOULD include a unique nonce in the authenticated_data field and track previously seen values.

8. IANA Considerations

8.1. MLS Wire Formats

The mls_targeted_message MLS Wire Format is used to send a message to a subset of members of an MLS group.

* Value: 0x0006 (suggested)
* Name: mls_targeted_message
* Recommended: Y
* Reference: RFC XXXX

8.2. MLS Signature Labels

8.2.1. TargetedMessageTBS

* Label: "TargetedMessageTBS"
* Recommended: Y
* Reference: RFC XXXX

8.3. MLS Exporter Labels

8.3.1. targeted message

* Label: "targeted message"
* Recommended: Y
* Reference: RFC XXXX

9. 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>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

[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>.

Acknowledgments

Author’s Address

Raphael Robert
Phoenix R&D GmbH
Email: ietf@raphaelrobert.com