Messaging Layer Security
Internet-Draft
Intended status: Informational
Expires: 4 January 2026

R. Robert
Phoenix R&D GmbH

3 July 2025

MLS Targeted Messages
draft-robert-mls-targeted-messages-00

Abstract

MLS targeted messages allow sending encrypted messages to a specific member of an MLS group.

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-robert-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

Table of Contents

1. Introduction

MLS application messages make sending encrypted messages to all group members easy and efficient. Sometimes application protocols mandate that messages are only sent to specific group members, 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, targeted messages define 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 that provides confidentiality and authentication, reusing mechanisms from [RFC9420], in particular [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 extension defines the mls_targeted_message WireFormat, where the content is a TargetedMessage.

```
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 {
  opaque group_id<V>;
  uint64 epoch;
  uint32 recipient_leaf_index;
  opaque authenticated_data<V>;
  uint32 sender_leaf_index;
  opaque kem_output<V>;
  opaque ciphertext<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 the its LeafNode. The signature scheme used is the signature scheme specified in 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.verify(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 SenderAuthDataAAD.epoch:

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

The targeted_message_psk is used as the psk parameter to the 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 TargetedMessagesTBM struct.

5. Encryption

Targeted messages uses HPKE to encrypt the message content between two leaves.

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. Sender data encryption

In addition, TargetedMessageSenderAuthData is encrypted similarly to MLSSenderData as described in Section 6.3.2 of [RFC9420]. The TargetedMessageSenderAuthData.sender_leaf_index field is the leaf index of the sender. The TargetedMessageSenderAuthData.signature field is the signature of the TargetedMessageTBS structure. The TargetedMessageSenderAuthData.kem_output field is the KEM output of the HPKE encryption.

The key and nonce provided to the AEAD are computed as the KDF of the first KDF.Nh bytes of the ciphertext generated in the following section. 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_data_key = ExpandWithLabel(sender_auth_data_secret, "key",
                      ciphertext_sample, AEAD.Nk)
sender_data_nonce = ExpandWithLabel(sender_auth_data_secret, "nonce",
                      ciphertext_sample, AEAD.Nn)
```

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

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

5.3. Application data encryption

The TargetedMessageContent struct contains the application data to be sent to the recipient. The application_data field contains the application data to be sent, and the padding field contains padding bytes to ensure that the ciphertext is of a length that is a multiple of the AEAD tag length.

The TargetedMessageContent struct is serialized and then 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 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, an is used as the aad parameter for the HPKE encryption.

The sender computes TargetedMessageSenderAuthData.kem_output andTargetedMessage.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,
                  receiver_node_private_key,
                  hpke_context,
                  targeted_message_tbm,
                  ciphertext,
                  targeted_message_psk,
                  psk_id)
```

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

6. Security Considerations

In addition to the sender authentication, Targeted Messages are authenticated by using a pre-shared key (PSK) between the sender and the recipient. The PSK is exported from the group key schedule using the label "targeted message psk". This ensures that the PSK is only valid for a specific group and epoch, and the Forward Secrecy and Post-Compromise Security guarantees of the group key schedule apply to the targeted messages as well. The PSK also ensures that an attacker needs access to the private group state in addition to the HPKE/signature’s private keys. This improves confidentiality guarantees against passive attackers and authentication guarantees against active attackers.‘

7. IANA Considerations

7.1. 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

7.2. MLS Signature Labels

7.2.1. TargetedMessageTBS

* Label: "TargetedMessageTBS"
* Recommended: Y

* Reference: RFC XXXX

8. 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

TODO acknowledge.

Author’s Address

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