Messaging Layer Security
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

R. Mahy
M. Chenani
Ephemera

20 October 2025

# Private External Message extensions for Messaging Layer Security (MLS)
draft-mahy-mls-private-external-00

## Abstract

MLS groups that use private handshakes lose member privacy when sending external proposals. This document addresses this shortcoming by encrypting external proposals using an HPKE public key derived from the epoch secret. It also provides a mechanism to share this key and protect it from tampering by a malicious intermediary.

## 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-private-external/draft-mahy-mls-private-external.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mahy-mls-private-external/.

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-private-external.

## Status of This Memo

## Copyright Notice



## Table of Contents

## 1. Introduction

The MLS protocol [RFC9420] was designed to support both a model where the Distribution Service (DS) sees the contents of MLS handshake messages and often assumes a policy enforcement role, and a model where the DS is merely responsible for forwarding handshake messages and possibly enforcing ordering of messages. In the first model clients send every handshake as a PublicMessage (or a SemiPrivateMessage [I-D.mahy-mls-semiprivatemessage]), whereas in the second model the clients send in-group handshakes as a PrivateMessage. As of this writing there are non-trivial commercial deployments using both the PublicMessage model (ex: Cisco, Amazon, Ring Central, Wire) and the PrivateMessage model (ex: Ephemera, Germ).

In the PrivateMessage model, group members enjoy substantially more privacy from the DS. In the PublicMessage model, the DS usually can provide (authorized) non-members with enough information that they can join a group via an external commit. Even in the PublicMessage model, some (usually large) groups use external proposals to join. In the PrivateMessage model, (authorized) non-members can also join using external proposals (or rarely using external commits if the GroupInfo is shared by an existing member), however the joiner is currently forced to send the proposal (or commit) as a PublicMessage and therefore reveal potentially private information such as their credential and capabilities to the DS.

This extension allows groups using PrivateMessage to maintain the privacy of external handshake messages by encrypting them to a public key derived from the group’s epoch secret. It also provides a way to convey that public key safely to prevent active attacks.

## 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. Mechanism

### 3.1. External Encryption Key Derivation

Groups using this extension derive a dedicated HPKE [RFC9180] key pair from the epoch secret for encrypting external messages. This key pair is derived independently from the ratchet tree structure.

The external encryption key pair is derived as follows:

```
external_encryption_secret =
    ExpandWithLabel(epoch_secret, "external encryption", "", KDF.Nh)

(external_encryption_private_key, external_encryption_public_key) =
    DeriveKeyPair(external_encryption_secret)
```

Where:

* epoch_secret is the epoch secret from [RFC9420]

* ExpandWithLabel is from [RFC9420]

* DeriveKeyPair is from [RFC9180]

* KDF.Nh is the output size of the hash function for the cipher suite

All group members in the current epoch can derive the same key pair from their shared epoch secret. The public key is made available to external senders via the ExternalEncryptionInfo structure (Section 3.2).

## 3.2. Additional information shared in every commit

Groups participating in this mechanism include a root_private_signature_key component (see Section 4.6 of [I-D.ietf-mls-extensions]) in the GroupContext of type RootPrivateSignature, containing a unique random private signature key corresponding to the group's cipher suite. Whenever a commit removes a member from a group, this component MUST be replaced with a new unique random private signature key.

Members sending a commit need to calculate the future epoch_secret, external_encryption_secret, and external_encryption_public_key for the new epoch that would result if the commit is accepted. The commit sender includes one additional Additional Authentication Data (AAD) component (see Section 4.9 of [I-D.ietf-mls-extensions]) of type ExternalEncryptionInfo in every commit (including commits sent in a PrivateExternalMessage). The ExternalEncryptionInfo includes the external_encryption_public_key for the future epoch.

> Note: SafeSignWithLabel is not used, because there are two different component IDs represented.

```
struct {
    opaque root_private_signature_key<V>;
} RootPrivateSignature;

struct {
    ProtocolVersion version = mls10;
    opaque group_id<V>;
    uint64 epoch;
    CipherSuite ciphersuite;
    HPKEPublicKey external_encryption_public_key;
    SignaturePublicKey root_public_signature_key;
} ExternalEncryptionInfoTBS;

struct {
    CipherSuite ciphersuite;
    HPKEPublicKey external_encryption_public_key;
    SignaturePublicKey root_public_signature_key;
    /* SignWithLabel(root_private_signature_key, */
    /*    "ExternalEncryptionInfoTBS", ExternalEncryptionInfoTBS) */
    opaque external_encryption_signature<V>;
} ExternalEncryptionInfo;
```

### 3.3. Sending an external proposal or external commit to the group

A non-member client that wishes to send a message to the group, first constructs a PublicMessage called external_message_plaintext. The PrivateExternalMessage wire format wraps that external_message_plaintext by encrypting it to the external_encryption_public_key.

```
/*  PublicMessage.content.sender.sender_type != member  */
PublicMessage external_message_plaintext;

encrypted_public_message = EncryptWithLabel(external_encryption_public_key,
    "PrivateExternalMessageContent", PrivateExternalMessageContext,
    external_message_plaintext)

struct {
    /* PublicMessage (the plaintext) is pretty self contained */
} PrivateExternalMessageContext;

struct {
    opaque group_id<V>;
    uint64 epoch;
    ContentType content_type;
    opaque authenticated_data<V>;
    HPKECiphertext encrypted_public_message<V>;
} PrivateExternalMessage;

PrivateExternalMessage.authenticated_data =
   external_message_plaintext.content.authenticated_data

struct {
    ProtocolVersion version = mls10;
    WireFormat wire_format;
    select (MLSMessage.wire_format) {
        case mls_public_message:
            PublicMessage public_message;
        case mls_private_message:
            PrivateMessage private_message;
        ...
        case mls_private_external_message;
            PrivateExternalMessage private_external_message
    };
} MLSMessage;
```

### 3.4. Decryption and verification by members

Members receiving a PrivateExternalMessage check that the group_id matches a known group and that the epoch is the current epoch.

To decrypt the message, members first derive the external encryption key pair from their current epoch secret:

```
/* Derive the external encryption key pair from epoch_secret */
external_encryption_secret =
    ExpandWithLabel(epoch_secret, "external encryption", "", KDF.Nh)

(external_encryption_private_key, external_encryption_public_key) =
    DeriveKeyPair(external_encryption_secret)

/* Decrypt the external message */
external_message_plaintext = DecryptWithLabel(
    external_encryption_private_key,
    "PrivateExternalMessageContent", PrivateExternalMessageContext,
    encrypted_public_message.kem_output,
    encrypted_public_message.ciphertext)
```

They then verify the following values in the PrivateExternalMessage match their corresponding field in the external_message_plaintext.content:

* group_id,

* epoch,

* content_type, and

* authenticated_data

Finally, they process the external_message_plaintext as if it were a regular PublicMessage.

## 4. Security Considerations

An established MLS group which only exchanges handshakes using MLS PrivateMessage enjoys a high level of privacy for its members. The GroupContext and the ratchet tree, including the contents of the credentials in MLS leaf nodes is not visible to outsiders nor to the DS. However, during the process of joining, private information is often leaked to the DS. This mechanism focuses on improving the privacy for the external joining mechanisms.

There are three mechanisms for potential new members to join an MLS group: an existing member gets a KeyPackage (KP) for the new member and commits an Add proposal with the KP; the joiner sends an external proposal asking to join the group that needs to be committed by an existing member; or the joiner fetches the GroupInfo of the group (usually from the DS) and sends an external commit. In the base MLS protocol [RFC9420], an external join or external commit needs to be sent as an MLS PublicMessage, which greatly reduces the privacy of the group.

### 4.1. Security of External Proposals

External Add proposals in [RFC9420] are sent using an MLS PublicMessage, which is integrity protected but reveals the public signature key, MLS capabilities, MLS credential to the DS, and KeyPackageRef (used to correlate Welcome messages). If a public key representing the entire target MLS group is available, the external proposer can encrypt this information to all group members without revealing it to the DS. The external proposer needs a way to get this public key and not the key of an active attacker, and the DS and members need a reasonable authorization and rate limiting mechanisms to prevent from being overwhelmed by such encrypted requests.

The ExternalEncryptionInfo defined in Section 3.2 contains a per-group, per-epoch signature key shared by all members of the group The ExternalEncryptionInfo could be posted in transparency ledger, shared as gossip, or additionally signed by a specific member. The specific mechanism can be tailored to a specific application as needed.

Application protocols above the MLS layer would also need to provide authorization. For example, in the MIMI protocol [I-D.ietf-mimi-protocol] this could be a join code. Other techniques such as using single or limited use pseudonymous tokens, privacy pass [RFC9576], or anonymous credit tokens [I-D.schlesinger-cfrg-act] are all reasonable options. The privacy of some of these techniques could also be reinforced by using Oblivious HTTP [RFC9458].

### 4.2. Security of External Commits

TODO

### 4.3. Security of KeyPackages and Welcomes

In the classical usage of MLS, a member of a group fetches a KeyPackage, commits an Add proposal containing that KeyPackage, the sends a Welcome to the new member. Both the returned KeyPackage and the query for it could reveal a lot of private information. In order to forward a Welcome message to the correct recipient, the DS needs to be able to associate the KeyPackageRef with some resource that eventually delivers to the appropriate client.

TODO add more.

## 5. IANA Considerations

TODO IANA

## 6. References

### 6.1. Normative References

[I-D.ietf-mls-extensions]
Robert, R., "The Messaging Layer Security (MLS) Extensions", Work in Progress, Internet-Draft, draft-ietf-mls-extensions-08, 21 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-extensions-08>.

[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>.

### 6.2. Informative References

[I-D.ietf-mimi-protocol]
Barnes, R., Hodgson, M., Kohbrok, K., Mahy, R., Ralston, T., and R. Robert, "More Instant Messaging Interoperability (MIMI) using HTTPS and MLS", Work in Progress, Internet-Draft, draft-ietf-mimi-protocol-04, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-protocol-04>.

[I-D.mahy-mls-semiprivatemessage]
Mahy, R., "Semi-Private Messages in the Messaging Layer Security (MLS) Protocol", Work in Progress, Internet-Draft, draft-mahy-mls-semiprivatemessage-06, 16 October 2025, <https://datatracker.ietf.org/doc/html/draft-mahy-mls-semiprivatemessage-06>.

[I-D.schlesinger-cfrg-act]
Schlesinger, S. and J. Katz, "Anonymous Credit Tokens", Work in Progress, Internet-Draft, draft-schlesinger-cfrg-act-00, 18 August 2025, <https://datatracker.ietf.org/doc/html/draft-schlesinger-cfrg-act-00>.

[RFC9458] Thomson, M. and C. A. Wood, "Oblivious HTTP", RFC 9458, DOI 10.17487/RFC9458, January 2024, <https://www.rfc-editor.org/rfc/rfc9458>.

[RFC9576] Davidson, A., Iyengar, J., and C. A. Wood, "The Privacy Pass Architecture", RFC 9576, DOI 10.17487/RFC9576, June 2024, <https://www.rfc-editor.org/rfc/rfc9576>.

## Authors' Addresses

Rohan Mahy
Email: rohan.ietf@gmail.com

Mojtaba Chenani
Ephemera
Email: chenani@outlook.com