Messaging Layer Security
Internet-Draft
Intended status: Informational
Expires: 5 January 2026

K. Kohbrok
Phoenix R&D

4 July 2025

Leaf Operation Intents
draft-kohbrok-mls-leaf-operation-intents-00

Abstract

The Messaging Layer Security (MLS) protocol defined in [RFC9420] is an asynchronous secure group messaging protocol, which allows group members to propose their own removal from a group.

However, in some cases MLS clients can’t reliably use regular Remove or SelfRemove proposals to leave a group because they don’t have an up-to-date group state.

This document specifies a LeafOperationIntent, which does not need an up-to-date group state but which retains sufficient binding to the client’s current state to avoid replay attacks.

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://kkohbrok.github.io/draft-kohbrok-mls-leaf-operation-intents/draft-kohbrok-mls-leaf-operation-intents.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-kohbrok-mls-leaf-operation-intents/.

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/kkohbrok/draft-kohbrok-mls-leaf-operation-intents.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 5 January 2026.

Table of Contents

1. Introduction

To leave an MLS group, a member cannot create a commit, but rather has to propose its own removal. This can create difficulties, some of which have been solved by the introduction of the SelfRemove proposal, which may be included in external commits.

One drawback of Remove and SelfRemove proposals is that they (like other proposals) are bound to a specific group epoch. This means that authors of such proposals must have an up-to-date group state to send such a proposal and continue to keep track of that group state both to re-send the proposal if necessary.

This can be a problem if an application wants to cleanly leave a group and immediately delete the associated group state, e.g., to erase the assocaited metadata. The deletion of the group state makes it impossible to for the client to re-send the proposal in case it’s not covered by the next commit. Similarly, the client might be offline at the time and the group state might not be up-to-date.

The LeafOperationIntent specified in this document allow a client to bind an intent to leave a group or update its own leaf to the leaf’s current state. That intent can then be proposed by any party (e.g. an external sender) in an arbitrary epoch as long as the leaf doesn’t change its state due to an update.

2. LeafOperationIntent

A LeafOperationIntent can be created by clients and distributed either to other group members or to one or more external senders.

```
HashReference LeafNodeRef;

MakeLeafNodeRef(value)
  = RefHash("MLS 1.0 LeafNode Reference", value)

enum {
  reserved(0),
  update(1)
  remove(2),
  (255)
} IntentType

struct {
  IntentType intent_type;
  select (Intent.intent_type) {
    case remove:
      {}
    case update:
      LeafNode leaf_node;
  }
} Intent

struct {
  opaque group_id<V>;
  uint32 sender_index;
  LeafNodeRef leaf_ref;
  Intent intent;
} LeafOperationIntentTBS

struct {
  opaque group_id<V>;
  uint32 sender_index;
  LeafNodeRef leaf_ref;
  Intent intent;
  /* SignWithLabel(., "LeafOperationIntentTBS", LeafOperationIntentTBS) */
  opaque signature<V>;
} LeafOperationIntent

struct LeafOperationProposal {
  LeafOperationIntent intent;
}
```

LeafNode, RefHash and SignWithLabel are as defined in [RFC9420].

* group_id: The ID of the group in which context the LeafOperationIntent was sent

* sender_index: The index of the sender’s leaf in the group

* leaf_ref: A hash computed over the LeafNode of the sender using the MakeLeafNodeRef function

* intent: The intent and a potential payload

* signature: A signature over all fields except the signature itself using the sender’s leaf signature key

2.1. Creating and proposing a LeafOperationIntent

A group member creates a LeafOperationIntent by populating the group_id, sender_index and leaf_ref according to the current state of the group and the sender’s leaf.

The intent indicates the operation the client would like to have proposed. A proposed and committed intent causes either the removal or the update of the sender’s leaf in the same way as a remove or update proposal would.

Finally the sender creates the signature by calling SignWithLabel on the LeafOperationIntentTBS populated as described above with "LeafOperationIntentTBS" as label.

Recipients of a LeafOperationIntent can include it in a LeafOperationProposal.

2.2. Processing a LeafOperationProposal

Recipients of a LeafOperationProposal MUST perform the following steps on the intent contained in the proposal.

* Verify that the group_id matches the group in which the proposal was sent

* Verify that the leaf_ref is the LeafRef of the leaf at the sender_index

* Verify the signature over the intent using the signature public key in the leaf at the sender_index

After that, the proposal MUST be validated and processed as if it were a Remove or Update proposal (depending on the type of the intent) originating from the sender of the intent (not the sender of the LeafOperationProposal).

External commits may include one or more LeafOperationProposals.

Open questions:

* Do we want to have an MLS wire format for LeafOperationIntent?

* Do we need an extension for this (like we do for SelfRemove proposal)?

3. Security Considerations

In contrast to proposals, LeafOperationIntents are not bound to an epoch and thus remain valid as long as the creator’s leaf doesn’t change its state.

Each LeafOperationIntent can thus be proposed and committed to until the sender is either removed from the group or updates its own leaf.

This allows scenarios, where, for example, members get added to or removed from a group in the time between the creation and the proposal of the intent.

If a tighter bound to the epoch, i.e. the current group state is required, clients should use regular Update, Remove or SelfRemove proposals instead.

4. IANA Considerations

This document requests the addition of a new Proposal Type under the heading of "Messaging Layer Security".

The leaf_operation_intent MLS Proposal Type is used to allow members or external sender to convey the intent of a leaf owner to perform an operation on their leaf.

* Value: 0x000c (suggested)

* Name: leaf_operation_intent

* Recommended: Y

* External: Y

* Path Required: Y

5. Normative References

[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.

Contributors

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

Author’s Address

Konrad Kohbrok
Phoenix R&D
Email: konrad@ratchet.ing