Workgroup: Network Working Group
Internet-Draft: draft-robert-mls-extensions-00
Published: 27 May 2022
Intended Status: Informational
Expires: 28 November 2022
Authors: R. Robert

The Messaging Layer Security (MLS) Extensions

Abstract

This document describes extensions to the Messaging Layer Security (MLS) protocol.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/mlswg/mls-extensions.

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 28 November 2022.

Copyright Notice

Table of Contents

1. Introduction
2. Extensions
  2.1. AppAck
    2.1.1. Description
3. IANA Considerations
  3.1. Extended MLS Extension types
  3.2. Extended MLS Proposal types
4. Normative References
Author's Address

1. Introduction

This document describes extensions to the Messaging Layer Security (MLS) protocol that are not part of the main protocol specification. The protocol specification includes a set of core extensions that are likely to be useful to many applications. The extensions described in this document are intended to be used by applications that need to extend the MLS protocol.

2. Extensions

2.1. AppAck

Type: Proposal

2.1.1. Description

An AppAck proposal is used to acknowledge receipt of application messages. Though this information implies no change to the group, it is structured as a Proposal message so that it is included in the group's transcript by being included in Commit messages.

```
struct {
    uint32 sender;
    uint32 first_generation;
    uint32 last_generation;
} MessageRange;

struct {
    MessageRange received_ranges<V>;
} AppAck;
```

An AppAck proposal represents a set of messages received by the sender in the current epoch. Messages are represented by the sender and generation values in the MLSCiphertext for the message. Each MessageRange represents receipt of a span of messages whose generation values form a continuous range from first_generation to last_generation, inclusive.

AppAck proposals are sent as a guard against the Delivery Service dropping application messages. The sequential nature of the generation field provides a degree of loss detection, since gaps in the generation sequence indicate dropped messages. AppAck completes this story by addressing the scenario where the Delivery Service drops all messages after a certain point, so that a later generation is never observed. Obviously, there is a risk that AppAck messages could be suppressed as well, but their inclusion in the transcript means that if they are suppressed then the group cannot advance at all.

The schedule on which sending AppAck proposals are sent is up to the application, and determines which cases of loss/suppression are detected. For example:

* The application might have the committer include an AppAck proposal whenever a Commit is sent, so that other members could know when one of their messages did not reach the committer.

* The application could have a client send an AppAck whenever an application message is sent, covering all messages received since its last AppAck. This would provide a complete view of any losses experienced by active members.

* The application could simply have clients send AppAck proposals on a timer, so that all participants' state would be known.

An application using AppAck proposals to guard against loss/suppression of application messages also needs to ensure that AppAck messages and the Commits that reference them are not dropped. One way to do this is to always encrypt Proposal and Commit messages, to make it more difficult for the Delivery Service to recognize which messages contain AppAcks. The application can also have clients enforce an AppAck schedule, reporting loss if an AppAck is not received at the expected time.

3. IANA Considerations

This document requests the creation of the following new IANA registries:

* MLS Extension Types (Section 3.1)

*MLS Proposal Types (Section 3.2)

All of these registries should be under a heading of "Messaging Layer Security", and assignments are made via the Specification Required policy [RFC8126].

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

3.1. Extended MLS Extension types

This registry lists identifiers for extensions to the MLS protocol. The extension type field is two bytes wide, so valid extension type values are in the range 0x0000 to 0xffff.

Template:

*Value: The numeric value of the extension type. Extended MLS extension types start with the value 0x0100.

*Name: The name of the extension type

*Message(s): The messages in which the extension may appear, drawn from the following list:

-KP: KeyPackage objects

-LN: LeafNode objects

-GC: GroupContext objects (and the group_context_extensions field of GroupInfo objects)

-GI: The other_extensions field of GroupInfo objects

*Recommended: Whether support for this extension is recommended by the IETF MLS WG. Valid values are "Y" and "N". The "Recommended" column is assigned a value of "N" unless explicitly requested, and adding a value with a "Recommended" value of "Y" requires Standards Action [RFC8126]. IESG Approval is REQUIRED for a Y->N transition.

*Reference: The document where this extension is defined

Initial contents:

Value	Name	Message(s)	Recommended	Reference
N/A	N/A	N/A	N/A	RFC XXXX

Table 1

3.2. Extended MLS Proposal types

This registry lists identifiers for types of proposals that can be made for changes to an MLS group. The extension type field is two bytes wide, so valid extension type values are in the range 0x0000 to 0xffff.

Template:

- *Value: The numeric value of the proposal type. Extended MLS proposal types start with the value 0x0100.

- *Name: The name of the proposal type

- *Recommended: Whether support for this extension is recommended by the IETF MLS WG. Valid values are "Y" and "N". The "Recommended" column is assigned a value of "N" unless explicitly requested, and adding a value with a "Recommended" value of "Y" requires Standards Action [RFC8126]. IESG Approval is REQUIRED for a Y->N transition.

- *Path Required: Whether a Commit covering a proposal of this type is required to have its path field populated.

- *Reference: The document where this extension is defined

Initial contents:

Value	Name	Recommended	Path Required	Reference
0x0100	app_ack	Y	Y	RFC XXXX

Table 2

4. Normative References

[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>.

Author's Address

Raphael Robert

Email: ietf@raphaelrobert.com