QUIC
Internet-Draft
Intended status: Informational
Expires: 24 August 2026

B. Dowling
Kings College London
B. Hale
Naval Postgraduate School
K. Kohbrok
R. Robert
Phoenix R&D GmbH
X. Tian
Naval Postgraduate School
B. Wimalasiri
University of Sheffield

20 February 2026

Securing QUIC with MLS
draft-tian-quic-quicmls-01

Abstract

This document describes how Messaging Layer Security (MLS) can be used in place of Transport Layer Security (TLS) to secure QUIC.

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://xisentian.github.io/ietf-quic-mls/draft-tian-quic-quicmls.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tian-quic-quicmls/.

Discussion of this document takes place on the QUIC Working Group mailing list (mailto:quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

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

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/.

Copyright Notice

Table of Contents

1. Introduction

Recent advances in key exchange protocol design for communications under network delays, disruption, and supporting low latency has offered new alternative key establishment techniques in the form of continuous key agreement. One such continuous key agreement protocol has been standardized by the IETF, namely Messaging Layer Security (MLS) (RFC9420). The MLS key agreement handshake can be generalized for dynamic or degraded communications settings that do not support duplex links, have highly mobile devices, and/or require asynchronous communications. QUIC's use of TLS for key agreement was primarily designed for relatively short-lived client-server connections with synchronous key initialization over reliable network availablity. MLS offers an alternative to users for cases where network reliability, attentuation, or disruption are concerns through asychronous key updates which also provide post-compromise security, enabling long-lived sessions with controllable key refresh periodicity. The MLS key agreement handshake can be slotted into QUIC in a fairly straightforward manner, preserving other needed QUIC functionality. The combination of QUIC and MLS thus addresses a need for a robust security protocol in certain evolving communication environments.

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. Notation

We use terms from MLS [RFC9420], TLS [RFC8446], and QUIC [RFC9000]. Below, we have restated relevant terms and define new ones:

Application Message: The private application data payload transported and encrypted by QUIC.

Authentication Service (AS): An abstract architectural component of MLS that provides mechanisms for verifying the authenticity of group members, typically by issuing credentials (e.g., certificates) that bind identities to cryptographic keys

Delivery Service (DS): An abstract architectural component of MLS for reliably delivering MLS messages (e.g., handshake or application messages) between group members while ensuring proper ordering.

Control Message: An MLS Proposal or Commit message to change the cryptographic state, as opposed to application data.

Key Derivation Function (KDF): A Hashed Message Authentication Code (HMAC)-based expand-and-extract key derivation function (HKDF) as described in RFC5869.

Key Encapsulation Mechanism (KEM): A key transport protocol that allows two parties to obtain a shared secret based on the receiver’s public key.

4. Scope

While MLS is designed for group settings, we limit discussions in this document to the two-party communications case. This choice is deliberate to avoid significant changes to QUIC and MLS.

5. Requirements for Integration

5.1. QUIC-HS Requirements

As an integrated secure transport protocol QUIC can be broken into two major components: a handshake layer (HS) responsible for key agreement and management and a record layer (RL) which provides secure (via HS keys) and reliable transport. Specifically, the HS layer requires the following from TLS as specified in RFC9001:

1. The handshake protocol must function as an authenticated key exchange (AKE) that produces distinct and unrelated keys for every connection where the server is always authenticated and the client is optionally authenticated.

2. Transport parameter authentication for both endpoints with additional confidentiality protection for the server transport parameters.

3. Provide means for authenticated negotiation of an application protocol.

With respect to (1) MLS provides authenticated key agreement to achieve the same outcome of AKE without exchanging key material. Using existing PKI certificates, communicating partners can generate MLS Key Packages to begin MLS sessions that produce indistinguishably random keys. Mutual authentication is enabled through asymmetric

verification of credentials within MLS Key Packages. External commits which are used for "open" MLS groups can be used to allow an unauthenticated client to be added (i.e. optional client authentication). With respect to transport parameter negotiations in (2), MLS satisfies this with publication of a signed GroupInfo object inside Welcome messages that contains all the necessary information to join a group and a public key to encrypt a secret to the existing group members. For (3) Authenticated negotiation of application protocols can be achieved using the MLS Content Advertisement Extension in [I-D.ietf-mls-extensions] which would be included in the GroupInfo object.

5.2. MLS Requirements

The prerequisits for MLS are the Delivery Service (DS) and Authentication Service (AS) functionalities as specified in RFC9750. A DS ensures MLS messages are delivered to all participants and enforces ordering of commits. For example, the DS can be a centralized server or a reliable transport protocol like TCP. An AS provides the issuance and verification of user credentials. For example, the AS can be any existing web Public Key Infrastructure (PKI) (e.g. certificate authority based scheme).

For most cases using QUIC, the DS functionality is satisfied by QUIC RL through guarantees for reliable ordered delivery of messages just as it provides TLS. The AS functionality is provided through existing PKI certificates already used by TLS. In non-traditional settings (e.g. dynamic topologies, delay/disruption prone environments) where the DS functionality cannot be met by QUIC RL alone, other mechanisms to ensure (ordered) delivery MUST be employed. Likewise, the AS functionality must also exist. Details and recommendations for alternative strategies relating to either functionalities are outside the scope of this document.

6. Protocol Overview

At a high level, MLS provides the traffic keys used to encrypt and authenticate QUIC’s secure channel. In turn, QUIC’s transport logic which handles ordering and reliable delivery of packets helps to maintain the MLS state. Note, additional mechanisms may be used to maintain MLS state in the event that QUIC transport is insufficient.

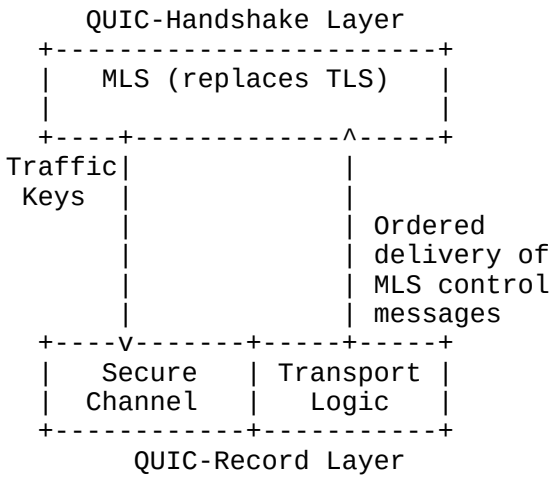


Figure 1: QUIC-MLS layer interactions

6.1. MLS Functionality

As a stateful protocol, MLS leverages a cryptographic structure called a ratchet tree to establish and maintain a shared cryptographic group state. The tree's root is the group's shared secret and each MLS group member is represented by a unique leaf node containing their HPKE encryption public key and signature public key from the AS. Intermediate nodes contain the HPKE encryption public key pairs calculated by members of the subgroup they cover. This property enables efficiency in updates to the ratchet tree. MLS group operations (e.g. add, remove, update) trigger modifications from each leaf along the path of intermediate nodes from the leaf to the root, thereby ratchetting the group state forward into a new epoch. Group operations are solicited through the DS via MLS Proposal messages and are processed (aka committed) by MLS Commit messages. Therefore, a QUIC-MLS session with respect to the HS functionality, is the creation and modification of the MLS ratchet state (the shared state).

Each epoch of an MLS session has a distinct asymmetric ratchet tree as previously described that seeds the MLS Key Schedule used to derive shared secrets used for key generation (e.g. via a Key Derivation Function). Values from the asymmetric ratchet tree and key schedule are used to derive a symmetric ratchet tree called the Secret Tree which is used to generate keys used for encrypting and authenticating application messages. QUIC-MLS provides keys from this symmetric ratchet tree to the QUIC record layer.

6.2. QUIC-MLS Execution

An initiator may unilaterally start a QUIC-MLS session with a partner. They then update the keys throughout their communications by sending MLS updates with each message until either parties terminates the session by issuinng a (self) removal proposal and commit. An important distinction from QUIC-TLS is that key updates here are ratcheted in accordance with the MLS Key Schedule (see Section 8 of [RFC9420]) as to provide forward secrecy and post-compromise security rather than using QUIC's native key update mechanism (see Section 6.1 of [RFC9001]) which only provides forward secrecy. Furthermore, by attaching a configurable series of previous MLS Commit messages to each sending stream, we ensure that the protocol is both robust enough to handle truly asynchronous settings under eventually consistent DS but also flexible enough to adapt to synchronous settings with strongly constistent DS assumptions.

6.2.1. Initialization

An initiating client can create MLS Key Packages based off of preloaded credentials obtained from the AS (i.e. via digital certificates) or through direct communications. In the absence of an AS, initators may use preinstalled long term signature keys from itself and the intended partner(s) to create the Key Packages necessary to start a session. The initiator uses the Key Packages to create an MLS Add proposal, which once committed to, starts the MLS session with the corresponding partner. In order for the other member to join and initialize their own cryptographic state, they will need to recieve their personalized MLS Welcome message. The initiator can wait to receive a MLS Commit from the partner or unilaterally commit their own Add proposal and thereby asynchronously intiate a session. If they unilaterally commmit their own Add proposal, then 0-RTT data may be sent along with these first batch of packets using the encryption key derived from the MLS group secret in accordance with the MLS Key Schedule (see Section 8 RFC9420). After the recipient verifies the initiator's Key Package and uses the Welcome message to construct the shared MLS state, the shared key is established and can be used by the record layer to commence secure communications via 1-RTT packets.

6.2.2. Updates

When a member chooses to update the group key, they send an Update proposal and the other member commits to that proposal to ratchet the group state into the next epoch. Alternatively, one party may serve as the session admin with commit authority or may unilaterally update (via an empty commit): in either case, when an eventually consistent DS is used, they MUST attach the antecedent series of commits leading

the the current epoch in order for the other member(s) to reconcile state agreement. In the two party case the series of commits reduces to simply the last commit.

To tighten the FS/PCS compromise windows, senders MAY attach an update proposal with every data message sent.

6.2.3. Termination

Session termination occurs when any party sends a (self) removal proposal and commit it. This MAY be sent multiplexed with the last data message or as an independent stream with no data message. For self removals in a two-party group, the proposer may terminate the session without waiting for commitment depending on application needs. The remove proposal shall be sent as the payload of a Crypto frame inside of a Handshake packet.

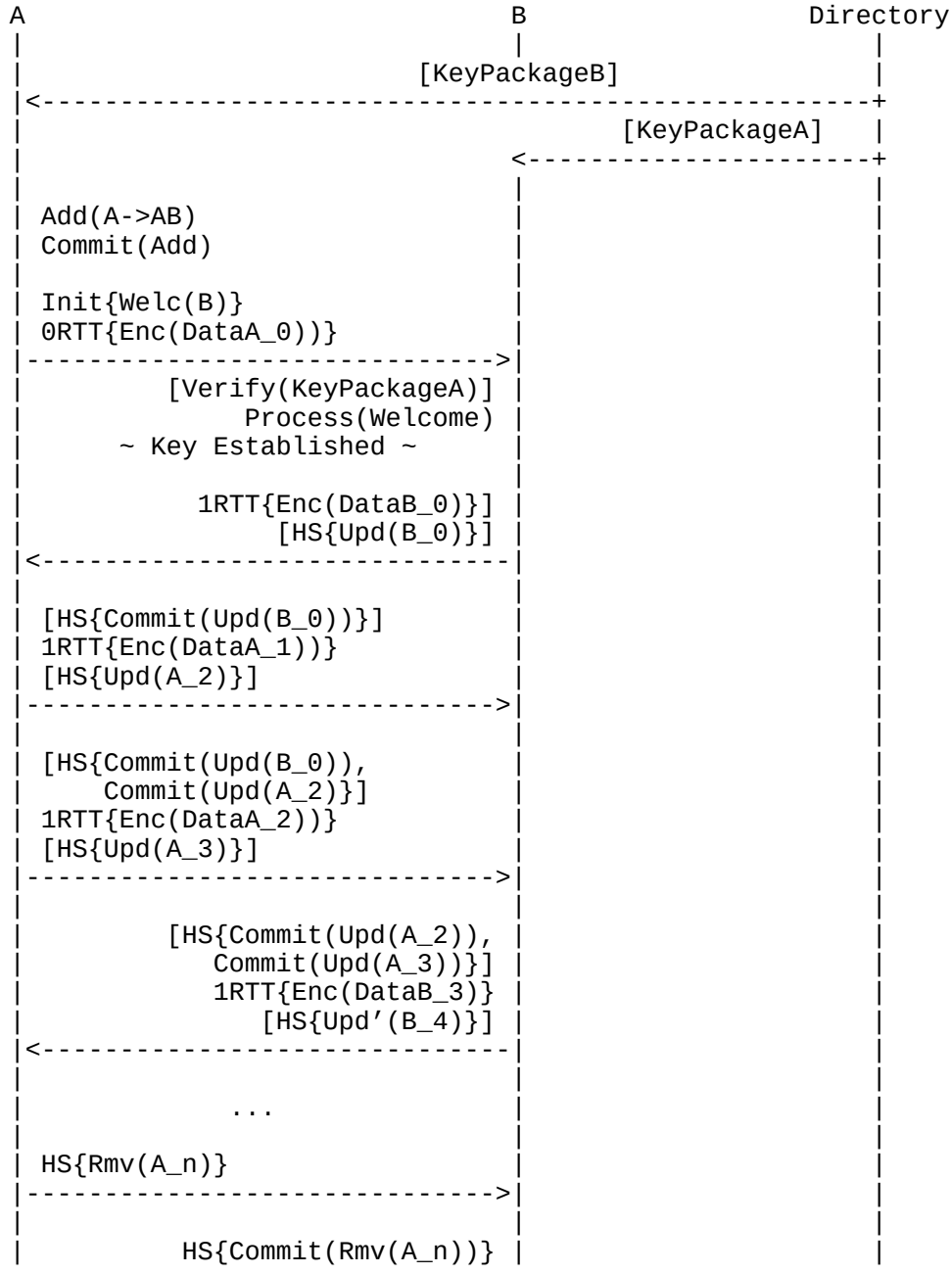
6.2.4. Resumption

Similar to TLS where 0RTT keys are used to resume a session and quickly send encrypted application messages, MLS can make use of resumption pre-shared key (PSK) from historical epochs to send 0RTT encrypted data. With TLS, data sent encrypted using 0RTT key alone is not forward secret and suffers from replay attacks. With MLS, a member uses a Reinit proposal which describes the historical group state they belonged to to rejoin the group with their resumption PSK. Once the Reinit proposal is committed it cannot be duplicated (i.e. conflicting commit) making QUIC-MLS immune from the types of replay attacks that 0RRT TLS keys are vulnerable to. The Reinit proposal for QUIC-MLS is sent in in the Crypto frame of a Initial Packet, behaving similar to an External Join. With an encryption key derived from the new epoch secret (e.g. via HKDF Expand) of the new state, the 0-RTT data can be sent along with the Reinit proposal and commits.

6.2.5. Example Execution

The following two-party QUIC-MLS execution illustrates initialization, updates, termination, and reinitialization of a session with per-message updates for the strictest security and commit transcripts for the an eventually consistent DS. What is shown can be relaxed to accommodate longer update windows and/or streamlined by reducing or eliminating commit transcripts for highly available and reliable networks which support a strongly consistent DS. Specifically, the protocol can be adapted to the synchronous and strongly consistent DS setting by removing the positive acknowledgement of latest commit HS packet (e.g. removing A’s second Commit(Upd(B_0) message and B’s HS{Commit(Upd(A_2)} message). The

bracketed (e.g. optional) per-message ratchetting updates which follow each 1RTT data message shown may also be removed to allow more longer security windows (see Security Considerations (Section 9.2)).



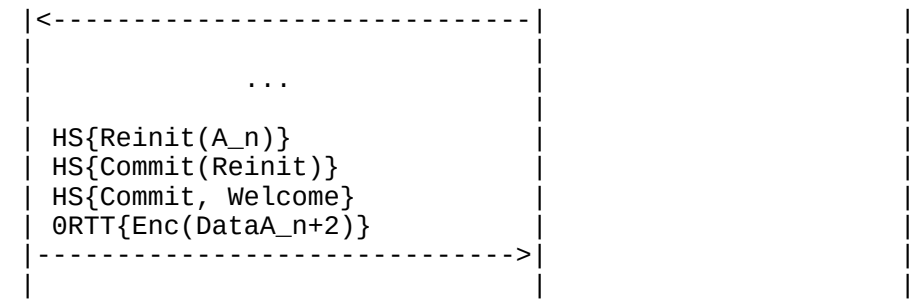


Figure 2: Client A creates a two-party QUIC-MLS session with Client B, and then (optionally) updates the group state until termination (via self- removal). Each MLS message sent is accompanied by a configurable series of commits to proposals. Brackets ('[]') indicate implicit/ optional communications steps. Handshake (HS), Init, and 0RTT packets wrap MLS messages indicated by braces ('{}'). Values following underscores ('_') indicate the associated epoch number.

TODO: Double check the reinit process -- is it better to just restart a new session? Seems like a new MLS session would have fewer steps in the two party case.

7. Packet Protection

As with QUIC with TLS, QUIC-MLS protects packets with keys derived from the MLS state, using an AEAD algorithm selected common to the communicating partners from their KeyPackage objects. Some key changes to how QUIC with TLS does packet protection (see Section 5 of [RFC9001]) are as follows:

* Initial Packets, which should be used to send MLS Welcome messages that are already encrypted with the joiner's public key retain their AEAD encryption using the secret derived from initial salt and connection ID as specified in Section 5.2 of [RFC9001]. Initial packets, which previously did not provide confidentiality or authentication to the payload, now carries a payload that DOES have confidentiality and authentication guarantees from MLS. NOTE: It may be redundant to keep the AEAD protection but removing it has downstream effects on Retry Packet usage/abuse. *TODO:* Decide if we want to even keep the Retry mechanism since we can form a MLS session unilaterally. ''

* Handshake Packets are proctected by keys derived from the MLS epoch secret.

* 0-RTT Packet Protection is provided by the traffic key computed from with the MLS Welcome message sent concurrently.

7.1. Key Derivation

For QUIC-MLS, traffic keys are derived the same as they are for QUIC with TLS except that the MLS Epoch Secret is used in place of the TLS Master Secret. Furthermore, the Early Secret, an artifact of the TLS key schedule, is never derived as it is unused (see Section 7.1 of [RFC8446]). All other keys used to encrypt or sign public and private MLS group control messages are derived according to the MLS Key Schedule (see Section 8 of [RFC9420]) using the appropriate lables. For example the following shows how traffic secrets used to encrypt 1RTT packets are derived from the MLS Epoch Secret:

```
0 -> HKDF-Extract = MLS Epoch Secret
          |
          +-----> Derive-Secret(., "A ap 0")
          |                     = client_application_traffic_secret_0
          |
          +-----> Derive-Secret(., "B ap 0")
          |                     = server_application_traffic_secret_0
```

Figure 3: QUIC traffic key derivation from MLS Epoch Secret

TODO: Technically MLS doesn’t expose access to the epoch secret, if we want to abide by safe extensions then we ought to use exporter secret here right?

7.2. Key Deletion

In general, keys should be deleted immediately after they are consumed. The key deletion schedule of MLS aligns with QUIC’s policies for discarding keys in that members may keep unconsumed values for handling out-of-order message delivery. Therefore, keys should be deleted in accordance first with the QUIC policies for discarding keys and then the MLS Key Deletion Schedule (Section 9.2 of [RFC9420]) for MLS specific values.

8. Message Framing

The main interface from QUIC used by MLS are the CRYPTO and NEW_TOKEN frames. As payloads of Initial, Handshake, and 0RTT packets, the CRYPTO frame will carry a majority of MLS Handshake messages that facilitate group control. Initial packets only carry the MLS Welcome and External Join messages to add new users to a group. Previous group members who wish to rejoin a group (using their MLS Resumption Preshared Keys) may use the NEW_TOKEN frame inside of Initial or

Retry packets to resume membership in the session with their PreSharedKey proposal as tokens. Handshake packets carry CRYPTO frames that have as payloads all other MLS proposals and commits in accordance with RFC9420.

```
Initial_Packet{
  [...]
  Packet Payload = CRYPTO_FRAME{}
}
Handshake_Packet{
  [...]
  Packet Payload = CRYPTO_FRAME{}
}
Retry_Packet{
  [...]
  Retry_Token = NEW_TOKEN_FRAME{}
}
CRYPTO_FRAME{
  Type (i) = tbd,
  Offset (i),
  Length (i),
  Crypto Data = <MLS_Message>
}
NEW_TOKEN_FRAME{
  Type (i) = 0x07,
  Token Length (i),
  Token = MLS_PreSharedKey_Proposal
}
```

Figure 4: QUIC-MLS packet and frame structures for carrying MLS messages (e.g. proposals, commits, welcome, etc.)

9. Security Considerations

9.1. Attacks on Authentication

While message integrity is provided by the symmetric key used in AEAD, insider attacks on non-repudiation (e.g., source forgery) on application messages may still be possible because QUIC headers and payloads aren’t signed. While the content (including sender information) is protected by AEAD which uses the symmetric group key, a malicious insider may still be able decrypt, modify, and re-encrypt the content using the same symmetric group key that they can compute. Thus, application messages sent by one member may be reordered or modified by another. However, in terms of group control, non-repudiation is unaffected because handshake messages are protected as signed MLS private messages.

9.2. Ratchet Window

The frequency of updates to the MLS group state can be adjusted as desired based on either time or number of messages. Following the maximum 604800 seconds (7 day) limit of the ticket_lifetime from TLS that forces a new DH handshake to establish fresh secrets, QUIC-MLS epochs must similarly not exceed 7 days in duration without an update.

9.3. Use of 0RTT

QUIC-MLS use of 0-RTT differs from QUIC-TLS in that a fresh key generated by the MLS state is used instead of using a pre-shared key generated from a previous TLS session between the communicating parties. QUIC-TLS uses 0-RTT and 1-RTT to explicitly differentiate the security levels but in QUIC-MLS their security levels are the same since they both are based on fresh randomness (akin to how 1-RTTs are protected by ephemeral Diffie Hellman keys). Unlike the caveats to using 0-RTTs for QUIC-TLS to thwart against certain types of replay attacks, QUIC-MLS has none.

10. IANA Considerations

TODO

11. References

11.1. 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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[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>.

11.2. Informative 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>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Benjamin Dowling
Kings College London
Email: benjamin.dowling@kcl.ac.uk

Britta Hale
Naval Postgraduate School
Email: britta.hale@nps.edu

Konrad Kohbrok
Phoenix R&D GmbH
Email: konrad.kohbrok@datashrine.de

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

Xisen Tian
Naval Postgraduate School
Email: xisen.tian1@nps.edu

Bhagya Wimalasiri
University of Sheffield
Email: bhagya.wimalasiri@kcl.ac.uk