Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2019

E. Omara
Google
R. Robert
Wire
March 11, 2019

The Messaging Layer Security (MLS) Federation draft-omara-mls-federation-00

Abstract

This document describes how the Messaging Layer Security (MLS) can be used in a federated environment where different MLS implementations can interoperate by defining the message format for user key retrieval. The document also describes some use cases where federation could be useful.

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 September 12, 2019.

Copyright Notice

Table of Contents

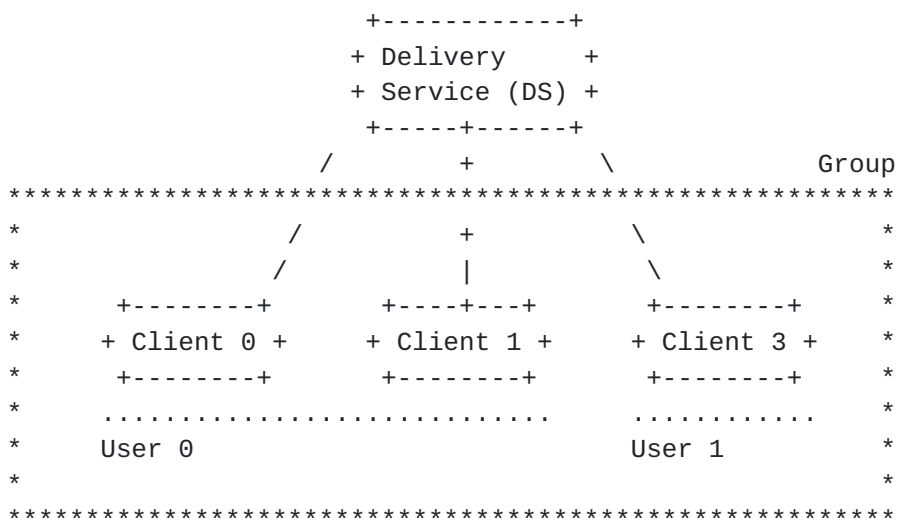
1. Introduction

MLS Architecture draft [MLSARCH] describes the overall MLS system architecture assuming the client and servers (Delivery Service and Authentication Service) are operated by the same entity. This document describes the minimum changes needed to allow different MLS clients operated by the same or different entities to communicate with each and explaining The use cases where federation could be useful.

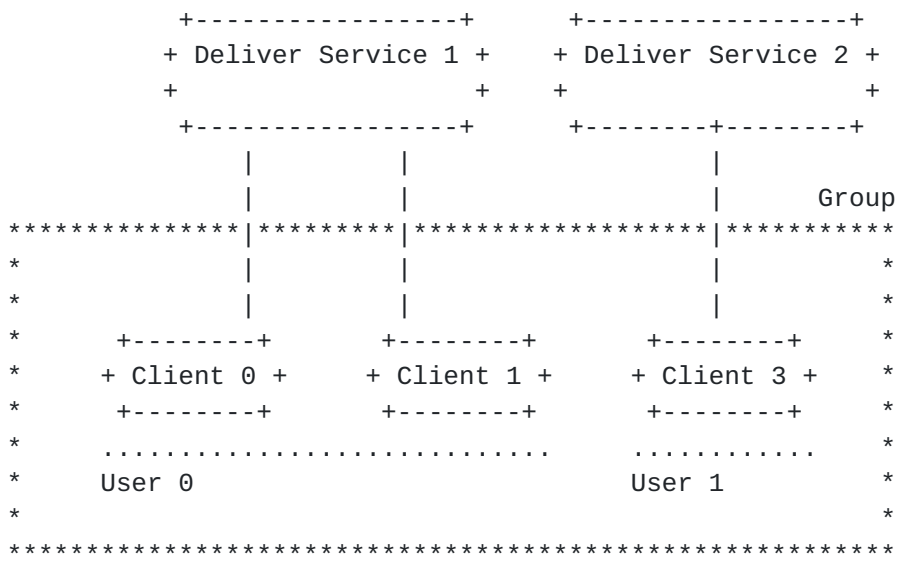
The focus of this document will be the interaction between the client and the Delivery Service, specifically how the client retrieves the identityKey and InitKeys for another client. There is no changes needed for the Authentication Service.

Discovering which Delivery service the client communicates with is out of the scope of this document.

The below diagram shows an MLS group where all clients are operated under the same deliver service:



one possible environment is to have different client implementations operated by the same delivery service, which will look like the diagram above, another environment is to have different or same clients operated By different delivery services:



2. Terminology

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.

Client: An agent that uses this protocol to establish shared cryptographic state with other clients. A client is defined by the cryptographic keys it holds. An application or user may use one client per device (keeping keys local to each device) or sync keys among a user's devices so that each user appears as a single client.

User Init Key: A short-lived HPKE key pair used to introduce a new client to a group. Initialization keys are published for each client (UserInitKey).

Identity Key: A long-lived signing key pair used to authenticate the sender of a message.

We use the TLS presentation language [RFC8446] to describe the structure of protocol messages.

3. Use cases

3.1. Different Delivery Servers

Different applications operated by different entities can use MLS to exchange E2EE messages. For example in email applications, clients of email1.com can encrypt and decrypt E2EE email messages from email2.com.

3.2. Different client applications

Different client applications operated by the same server can use MLS to exchange E2EE handshake and application messages. For example different browsers can implement the MLS protocol, and web developers write web applications that use the MLS implementation in the browser to encrypt and decrypt the messages. This will require a new standard Web API to allow the client applications to set the address of the delivery service in the browser. A more concrete example is using MLS in the browser to exchange SRTP keys for multi-party conference call.

4. Functional Requirements

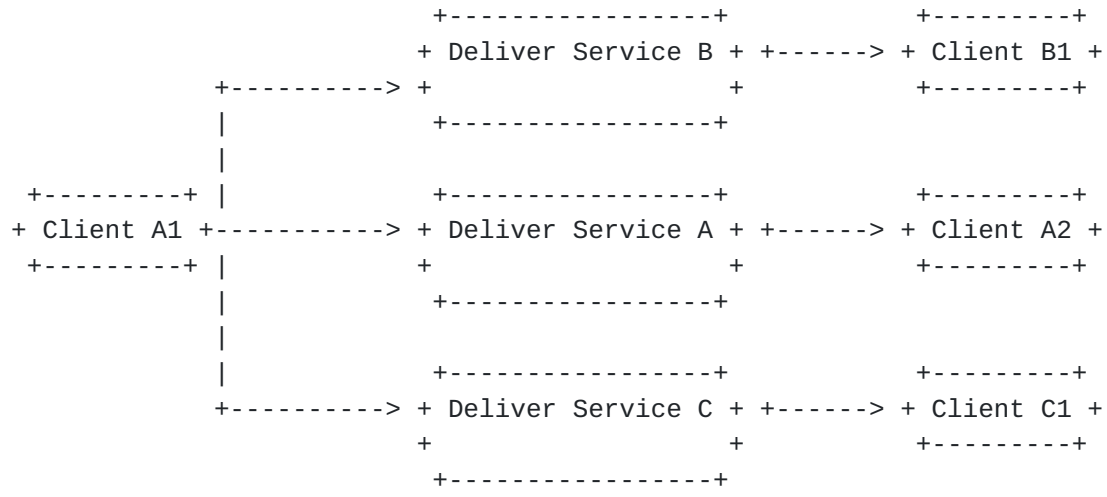
4.1. Delivery service

In MLS environment the messages can either be delivered using client fanout or server fanout, each will have different requirements.

In a federated environment the client may communicate with one or more delivery services. Discovering the delivery service and syncing between different delivery services are out of scope of this document.

4.1.1. Client fanout

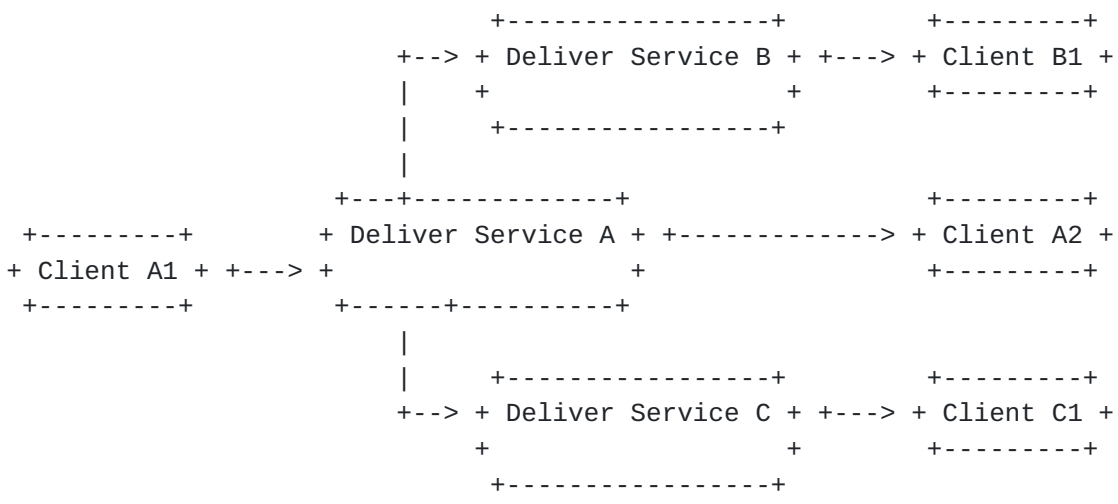
In this mode, the client SHOULD support communicating with multiple delivery services. Discovering the delivery service is out of scope of this document.



In this mode, the delivery service SHPULD be stateless, and it the clients responsibility to maintain the group state. OPEN QUESTION: How ordering could be enforced in this mode?

4.1.2. Server fanout

Multiple delivery services can be avoided, with server side fan out, and all keys requests can be proxied through a single delivery service. The protocol between different delivery services is out of the scope of this document.



OPEN QUESTION: How server assist could be used with multiple servers? how the server state is shared and synced ?

4.2. Authentication service

There is no change needed for the authentication service, however the authentication in a federated environment becomes more important. The ideal solution would be using a shared transparency log like [KeyTransparency].

5. Message format

The encrypted message payload is defined in the MLS protocol document [MLSPROTO], in order to get federation between different systems, the identity key and user init key retrieval MUST be defined as well. The identity key can always be included in the user init key response.

```
enum {
        P256_SHA256_AES128GCM(0x0000),
        X25519_SHA256_AES128GCM(0x0001),
        (0xFFFF)
} CipherSuite;

struct {
   opaque identity<0..2^16-1>;
   CipherSuite supported_suites<0..255>;
}  GetUserInitKeyRequest;

struct {
        opaque user_init_key_id<0..255>;
        CipherSuite cipher_suites<0..255>;
        HPKEPublicKey init_keys<1..2^16-1>;
        Credential credential;
        opaque signature<0..2^16-1>;
} UserInitKey;

struct {
        opaque identity<0..2^16-1>;
        UserInitKey user_init_key;
} UserInitKeyBundle;
```

The delivery service will return one or more user init key bundles, one for each member.

```
struct {
   UserInitKeyBundle user_init_keys<0..2^32-1>;
}  GetUserInitKeyResponse;
```

OPEN QUESTION: What if different clients have different cipher suites?

6. Security Considerations

6.1. Version negotiation

In a federated environment, version negotiation is more critical, to avoid forcing a downgrade attack by malicious 3rd party delivery services. The negotiation could either be done in the UserInitKeyBundle or in a separate handshake message.

7. IANA Considerations

This document makes no requests of IANA.

8. References

8.1. 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/info/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/info/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/info/rfc8446>.

8.2. Informative References

[KeyTransparency] Google, ., "Key Transparency", n.d., <https://KeyTransparency.org>.

[MLSARCH] Omara, E., Barnes, R., Rescorla, E., Inguva, S., Kwon, A., and A. Duric, "Messaging Layer Security Architecture", 2018.

[MLSPROTO] Barnes, R., Millican, J., Omara, E., Cohn-Gordon, K., and R. Robert, "Messaging Layer Security Protocol", 2018.

Authors' Addresses

Emad Omara
Google

Email: emadomara@google.com

Raphael Robert
Wire

Email: raphael@wire.com