Workgroup: Network Working Group
Internet-Draft: draft-ietf-mls-architecture-08
Published: 16 June 2022
Intended Status: Informational
Expires: 18 December 2022
Authors: B. Beurdouche (Inria & Mozilla)
E. Rescorla (Mozilla)
E. Omara (Google)
S. Inguva (Twitter)
A. Kwon (MIT)
A. Duric (Wire)

The Messaging Layer Security (MLS) Architecture

Abstract

The Messaging Layer Security (MLS) protocol [I-D.ietf-mls-protocol] specification has the role of defining a Group Key Agreement protocol, including all the cryptographic operations and serialization/deserialization functions necessary for scalable and secure group messaging. The MLS protocol is meant to protect against eavesdropping, tampering, message forgery, and provide further properties such as Forward Secrecy (FS) and Post-Compromise Security (PCS) in the case of past or future device compromises.

This document describes a general secure group messaging infrastructure and its security goals. It provides guidance on building a group messaging system and discusses security and privacy tradeoffs offered by multiple security mechanisms that are part of the MLS protocol (e.g., frequency of public encryption key rotation).

The document also provides guidance for parts of the infrastructure that are not standardized by the MLS Protocol document and left to the application or the infrastructure architects to design.

While the recommendations of this document are not mandatory to follow in order to interoperate at the protocol level, they affect the overall security guarantees that are achieved by a messaging application. This is especially true in case of active adversaries that are able to compromise clients, the delivery service, or the authentication service.

Discussion Venues

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

Discussion of this document takes place on the MLS Working Group mailing list (mls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mls/.

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

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 18 December 2022.

Copyright Notice

Table of Contents

1. Introduction

RFC EDITOR: PLEASE REMOVE THE FOLLOWING PARAGRAPH

The source for this draft is maintained in GitHub. Suggested changes should be submitted as pull requests at https://github.com/mlswg/mls-architecture. Instructions are on that page as well. Editorial

changes can be managed in GitHub, but any substantive change should be discussed on the MLS mailing list.

End-to-end security is a requirement for instant messaging systems and is commonly deployed in many such systems. In this context, "end-to-end" captures the notion that users of the system enjoy some level of security -- with the precise level depending on the system design -- even in the face of malicious actions by the operator of the messaging system.

Messaging Layer Security (MLS) specifies an architecture (this document) and a protocol [I-D.ietf-mls-protocol] for providing end-to-end security in this setting. MLS is not intended as a full instant messaging protocol but rather is intended to be embedded in concrete protocols, such as XMPP [RFC6120]. Implementations of the MLS protocol will interoperate at the cryptographic level, though they may have incompatibilities in terms of how protected messages are delivered, contents of protected messages, and identity/authentication infrastructures. The MLS protocol has been designed to provide the same security guarantees to all users, for all group sizes, even when it reduces to only two users.

2. General Setting

Informally, a group is a set of users who possibly use multiple endpoint devices to interact with the Service Provider (SP). A group may be as small as two members (the simple case of person to person messaging) or as large as thousands.

In order to communicate securely, users initially interact with services at their disposal to establish the necessary values and credentials required for encryption and authentication.

The Service Provider presents two abstract functionalities that allow clients to prepare for sending and receiving messages securely:

* An Authentication Service (AS) functionality which is responsible for attesting to bindings between application-meaningful identifiers and the public key material used for authentication in the MLS protocol. This functionality must also be able to generate credentials that encode these bindings and validate credentials provided by MLS clients.

* A Delivery Service (DS) functionality which can receive and distribute messages between group members. In the case of group messaging, the delivery service may also be responsible for acting as a "broadcaster" where the sender sends a single message which is then forwarded to each recipient in the group by the DS. The DS is also responsible for storing and delivering initial

public key material required by MLS clients in order to proceed with the group secret key establishment that is part of the MLS protocol.

For convenience, this document adopts the representation of these services being standalone servers, however the MLS protocol design is made so that this is not necessarily the case. These services may reside on the same server or different servers; they may be distributed between server and client components; and they may even involve some action by users. For example:

- Several secure messaging services today provide a centralized DS, and rely on manual comparison of clients' public keys as the AS.

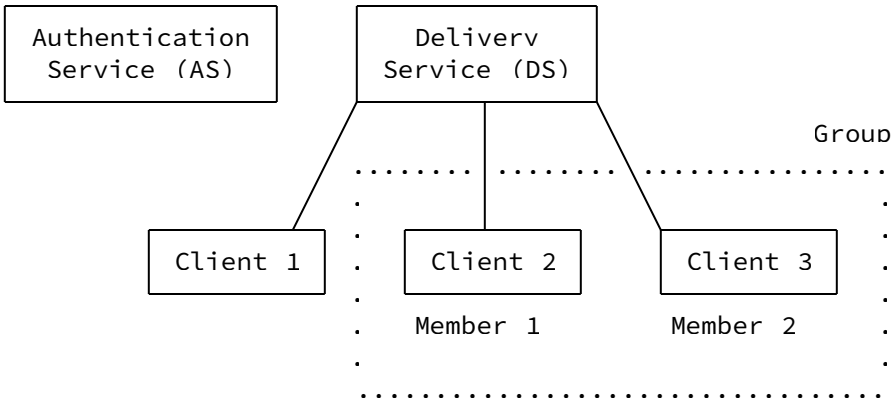
- MLS clients connected to a peer-to-peer network could instantiate a decentralized DS by transmitting MLS messages over that network.

- In an MLS group using a PKI for authentication, the AS would comprise the certificate issuance and validation processes, both of which involve logic inside MLS clients as well as various servers.

It is important to note that the Authentication Service functionality can be completely abstract in the case of a Service Provider which allows MLS clients to generate, redistribute and validate their credentials themselves.

Similarly to the AS, the Delivery Service can be completely abstract if users are able to distribute credentials and messages without relying on a central Delivery Service. Note, though, that the MLS protocol requires group operation messages to be processed in-order by all MLS clients.

In some sense, a set of MLS clients which can achieve the AS and DS functionalities without relying on an external party do not need a Service Provider.



In many systems, the AS and the DS are actually operated by the same entity and may even be the same server. However, they are logically distinct and, in other systems, may be operated by different entities. Other partitions are also possible, such as having a separate directory functionality or service.

According to this architecture design, a typical group messaging scenario might look like this:

1. Alice, Bob and Charlie create accounts with a service provider and obtain credentials from the AS.

2. Alice, Bob and Charlie authenticate to the DS and store some initial keying material which can be used to send encrypted messages to them for the first time. This keying material is authenticated with their long-term credentials.

3. When Alice wants to send a message to Bob and Charlie, she contacts the DS and looks up their initial keying material. She uses these keys to establish a new set of keys which she can use to send encrypted messages to Bob and Charlie. She then sends the encrypted message(s) to the DS, which forwards them to the recipients.

4. Bob and/or Charlie respond to Alice's message. In addition, they might choose to update their key material which provides post-compromise security Section 7.2.2. As a consequence of that change, the group secrets are updated.

Clients may wish to do the following:

*create a group by inviting a set of other clients;

*add one or more clients to an existing group;

*remove one or more members from an existing group;

*update their own key material

*join an existing group;

*leave a group;

*send a message to everyone in the group;

*receive a message from someone in the group.

At the cryptographic level, clients (and by extension members in groups) have equal permissions. For instance, any member can add or remove another client in a group. This is in contrast to some designs in which there is a single group controller who can modify the group. MLS is compatible with having group administration restricted to certain users, but we assume that those restrictions are enforced by authentication and access control at the application layer.

Thus, for instance, while the MLS protocol allows for any existing member of a group to add a new client, applications which use MLS might enforce additional restrictions for which only a subset of members can qualify, and thus will handle enforcing group policies (such as determining if a user is allowed to add new users to the group) at the application level.

2.1. Group Members and Clients

While informally, a group can be considered to be a set of users possibly using multiple endpoint devices to interact with the Service Provider, this definition is too simplistic.

Formally, a client is a set of cryptographic objects composed of public values such as a name (an identity), a public encryption key and a public signature key. Ownership of a client by a user is determined by the fact that the user has knowledge of the associated secret values. When a client is part of a Group, it is called a Member. In some messaging systems, clients belonging to the same user must all share the same signature key pair, but MLS does not assume this.

Users will often use multiple devices, e.g., a phone as well as a laptop. Different devices may be represented as different clients, with independent cryptographic state, or they may share cryptographic state, relying on some application-provided mechanism to sync across devices.

The formal definition of a Group in MLS is the set of clients that have knowledge of the shared group secret established in the group key establishment phase of the protocol and have contributed to it. Until a Member has been added to the group and contributed to the group secret in a manner verifiable by other members of the group, other members cannot assume that the Member is a member of the group.

3. Authentication Service

The Authentication Service (AS) has to provide two functionalities:

1. Issue credentials to clients that attest to bindings between identities and signature key pairs

2. Enable a group member to verify that a credential presented by another member is valid

A member with a valid credential authenticates its MLS messages by signing them with the private key corresponding to the public key in its credential.

The AS is considered an abstract layer by the MLS specification, part of this service could be, for instance, running on the members' devices, while another part is a separate entity entirely. The following examples illustrate the breadth of this concept:

* A PKI could be used as an AS [RFC5280]. The issuance function would be provided by the certificate authorities in the PKI, and the verification function would correspond to certificate verification by clients.

* Several current messaging applications rely on users verifying each others' key fingerprints for authentication. In this scenario, the issuance function is simply the generation of a key pair (i.e., credential is just an identifier and public key, with no information to assist in verification). The verification function is the application functionality that enables users to verify keys.

* In a system based on Key Transparency (KT) [KeyTransparency], the issuance function would correspond to the insertion of a key in a KT log under a user's identity. The verification function would correspond to verifying a key's inclusion in the log for a claimed identity, together with the KT log's mechanisms for a user to monitor and control which keys are associated to their identity.

By the nature of its roles in MLS authentication, the AS is invested with a large amount of trust and the compromise of one of its

functionalities could allow an adversary to, among other things, impersonate group members. We discuss security considerations regarding the compromise of the different AS functionalities in detail in Section 7.4.3.

The association between members' identities and signature keys is fairly flexible in MLS. As noted above, there is no requirement that all clients belonging to a given user use the same key pair (in fact, such key reuse is forbidden to ensure clients have independent cryptographic state). A member can also rotate the signature key they use within a group. These mechanisms allow clients to use different signature keys in different contexts and at different points in time, providing unlinkability and post-compromise security benefits. Some security trade-offs related to this flexibility are discussed in the security considerations.

In many applications, there are multiple MLS clients that represent a single entity, for example a human user with a mobile and desktop version of an application. Often the same set of clients is represented in exactly the same list of groups. In applications where this is the intended situation, other clients can check that a user is consistently represented by the same set of clients. This would make it more difficult for a malicious AS to issue fake credentials for a particular user because clients would expect the credential to appear in all groups of which the user is a member. If a client credential does not appear in all groups after some relatively short period of time, clients have an indication that the credential might have been created without the user's knowledge. Due to the asynchronous nature of MLS, however, there may be transient inconsistencies in a user's client set, so correlating users' clients across groups is more of a detection mechanism than a prevention mechanism.

4. Delivery Service

The Delivery Service (DS) is expected to play multiple roles in the Service Provider architecture:

* Acting as a directory service providing the initial keying material for clients to use. This allows a client to establish a shared key and send encrypted messages to other clients even if the other client is offline.

* Routing MLS messages among clients.

Depending on the level of trust given by the group to the Delivery Service, the functional and privacy guarantees provided by MLS may differ but the authentication and confidentiality guarantees remain the same.

Unlike the Authentication Service which is trusted for authentication and secrecy, the Delivery Service is completely untrusted regarding this property. While privacy of group membership might be a problem in the case of a Delivery Service server fanout, the Delivery Service can be considered as an active, adaptive network attacker from the point of view of the security analysis.

4.1. Key Storage

Upon joining the system, each client stores its initial cryptographic key material with the Delivery Service. This key material, called a KeyPackage, advertises the functional abilities of the client such as supported protocol versions, supported extensions, and the following cryptographic information:

* A credential from the Authentication Service attesting to the binding between the identity and the client's signature key.

* The client's asymmetric encryption public key material.

All the parameters in the KeyPackage are signed with the signature private key corresponding to the credential.

As noted above, users may own multiple clients, each with their own keying material, and thus there may be multiple entries stored by each user.

The Delivery Service is also responsible for allowing users to add, remove or update their initial key material, and for ensuring that the identifier for these keys are unique across all keys stored on the Delivery Service.

4.2. Key Retrieval

When a client wishes to establish a group, it first contacts the Delivery Service to request a KeyPackage for each other client, authenticates the KeyPackages using the signature keys, and then can use those to form the group.

4.3. Delivery of Messages

The main responsibility of the Delivery Service is to ensure delivery of messages. Some MLS messages need only be delivered to some members of a group (e.g., the message initializing a new member's state), while others need to be delivered to all members. The Delivery Service may enable these delivery patterns via unicast channels (sometimes known as "client fanout"), broadcast channels ("server fanout"), or a mix of both.

For the most part, MLS does not require the Delivery Service to deliver messages in any particular order. The one requirement is that because an MLS group has a linear history, the members of the group must agree on the order in which changes are applied. Concretely, the group must agree on which MLS Commit messages to apply. There are a variety of ways to achieve this agreement, but most of them rely on some help from the Delivery Service. For example, if a Delivery Service provides delivery in the same order to all group members, then the members can simply apply Commits in the order in which they appear.

Each Commit is premised on a given state or "epoch" of the group. The Delivery Service must transmit to the group exactly one Commit message per epoch.

Much like the Authentication Service, the Delivery Service can be split between server and client components. Achieving the required uniqueness property will typically require a combination of client and server behaviors. For example, all of the following examples provide a unique Commit per epoch:

- A "filtering server" Delivery Service where a server rejects all but the first Commit for an epoch and clients apply each Commit they receive.

- An "ordering server" Delivery Service where a server forwards all messages but assures that all clients see Commits in the same order, and clients.

- A "passive server" Delivery Service where a server forwards all messages without ordering or reliability guarantees, and clients execute some secondary consensus protocol to choose among the Commits received in a window.

The MLS protocol provides three important pieces of information within an MLSCiphertext message in order to provide ordering:

- The Group Identifier (group ID) to allow for distinguishing the group for which the message has been sent;

- The Epoch number, which represents the number of changes (version) of the group associated with a specific group ID, and allows for lexicographical ordering of messages from different epochs within the same group;

- The Content Type of the message, which allows the Delivery Service to determine the ordering requirement on the message, in particular distinguishing Commit messages from other messages.

The MLS protocol itself can verify these properties. For instance, if the Delivery Service reorders messages from a client or provides different clients with inconsistent orderings, then clients can put messages back in their proper order. The asynchronous nature of MLS means that within an epoch, messages are only ordered per-sender, not globally.

Note that some forms of Delivery Service misbehavior are still possible and difficult to detect. For instance, a Delivery Service can simply refuse to relay messages to and from a given client. Without some sort of side information, other clients cannot generally distinguish this form of Denial of Service (DoS) attack.

4.4. Membership knowledge

Group membership is itself sensitive information and MLS is designed to limit the amount of persistent metadata. However, large groups often require an infrastructure which provides server fanout. In the case of client fanout, the destination of a message is known by all clients, hence the server usually does not need this information. However, they may learn this information through traffic analysis. Unfortunately, in a server-side fanout model, the Delivery Service can learn that a given client is sending the same message to a set of other clients. In addition, there may be applications of MLS in which the group membership list is stored on some server associated with the Delivery Service.

While this knowledge is not a breach of the protocol's authentication or confidentiality guarantees, it is a serious issue for privacy. In the case where metadata has to be persisted for functionality, it SHOULD be stored encrypted at rest. Applications should also consider anonymous systems for server fanout such as Loopix [Loopix].

4.5. Membership and offline members

Because Forward Secrecy (FS) and Post-Compromise Security (PCS) rely on the active deletion and replacement of keying material, any client which is persistently offline may still be holding old keying material and thus be a threat to both FS and PCS if it is later compromised.

MLS cannot inherently defend against this problem, especially in the case where the client has not processed messages, but MLS-using systems can enforce some mechanism to try to retain these properties. Typically this will consist of evicting clients which are idle for too long, or mandating a key update from clients that are not otherwise sending messages. The precise details of such

mechanisms are a matter of local policy and beyond the scope of this document.

5. Functional Requirements

MLS is designed as a large-scale group messaging protocol and hence aims to provide both performance and safety to its users. Messaging systems that implement MLS provide support for conversations involving two or more members, and aim to scale to groups with tens of thousands of members, typically including many users using multiple devices.

5.1. Membership Changes

MLS aims to provide agreement on group membership, meaning that all group members have agreed on the list of current group members.

Some applications may wish to enforce ACLs to limit addition or removal of group members, to privileged clients or users. Others may wish to require authorization from the current group members or a subset thereof. Such policies can be implemented at the application layer, on top of MLS. Regardless, MLS does not allow for or support addition or removal of group members without informing all other members.

Membership of an MLS group is managed at the level of individual clients. In most cases, a client corresponds to a specific device used by a user. If a user has multiple devices, the user will be represented in a group by multiple clients. If an application wishes to implement operations at the level of users, it is up to the application to track which clients belong to a given user and ensure that they are added / removed consistently.

MLS provides two mechanisms for changing the membership of a group. The primary mechanism is for an authorized member of the group to send a Commit that adds or removes other members. The second mechanism is an "external join": A member of the group publishes certain information about the group, which a new member can use to construct an "external" Commit message that adds the new member to the group. (There is no similarly unilateral way for a member to leave the group; they must be removed by a remaining member.)

With both mechanisms, changes to the membership are initiated from inside the group. When members perform changes directly, this is clearly the case. External joins are authorized indirectly, in the sense that a member publishing a GroupInfo object authorizes anyone to join who has access to the GroupInfo object. External joins do not allow for more granular authorization checks to be done before the new member is added to the group, so if an application wishes to both allow external joins and enforce such checks, then the

application will need to do such checks when a member joins, and remove them if checks fail.

Application setup may also determine other criteria for membership validity. For example, per-device signature keys can be signed by an identity key recognized by other participants. If a certificate chain is used to sign off on device signature keys, then revocation by the owner adds an alternative flag to prompt membership removal.

An MLS group's secrets change on every change of membership, so each client only has access to the secrets used by the group while they are a member. Messages sent before a client joins or after they are removed are protected with keys that are not accessible to the client. Compromise of a member removed from a group does not affect the security of messages sent after their removal. Messages sent during the client's membership are also secure as long as the client has properly implemented the MLS deletion schedule.

5.2. Parallel Groups

Any user or client may have membership in several groups simultaneously. The set of members of any group may or may not form a subset of the members of another group. MLS guarantees that the FS and PCS goals within a given group are maintained and not weakened by user membership in multiple groups. However, actions in other groups likewise do not strengthen the FS and PCS guarantees within a given group, e.g. key updates within a given group following a device compromise does not provide PCS healing in other groups; each group must be updated separately to achieve internal goals. This also applies to future groups that a member has yet to join, that are likewise unaffected by updates performed in current groups.

Applications may strengthen connectivity among parallel groups by requiring periodic key updates from a user across all groups in which they have membership.

Applications may use the PSK mechanism to link healing properties among parallel groups. For example, suppose a common member M of two groups A and B has performed a key update in group A but not in group B. The key update provides PCS with regard to M in group A. If a PSK is exported from group A and injected into group B, then some of these PCS properties carry over to group B, since the PSK and secrets derived from it are only known to the new, updated version of M, not to the old, possibly compromised version of M.

5.3. Asynchronous Usage

No operation in MLS requires two distinct clients or members to be online simultaneously. In particular, members participating in conversations protected using MLS can update the group's keys, add

or remove new members, and send messages without waiting for another user's reply.

Messaging systems that implement MLS have to provide a transport layer for delivering messages asynchronously and reliably.

5.4. Access Control

The MLS protocol allows each member of the messaging group to perform operations equally. This is because all clients within a group (members) have access to the shared cryptographic material. However every service/infrastructure has control over policies applied to its own clients. Applications managing MLS clients can be configured to allow for specific group operations. On the one hand, an application could decide that a group administrator will be the only member to perform add and remove operations. On the other hand, in many settings such as open discussion forums, joining can be allowed for anyone.

The MLS protocol can, in certain modes, exchange unencrypted group operation messages. This flexibility is to allow services to perform access control tasks on behalf of the group.

While the Application messages will always be encrypted, having the handshake messages in plaintext has inconveniences in terms of privacy as someone could collect the signatures on the handshake messages and use them for tracking.

> **RECOMMENDATION:** Prefer using encrypted group operation messages to avoid privacy issues related to non-encrypted signatures.

Note that in the default case of encrypted handshake messages, any access control policies will be applied at the client, so the application must ensure that the access control policies are consistent across all clients to make sure that they remain in sync. If two different policies were applied, the clients might not accept or reject a group operation and end-up in different cryptographic states, breaking their ability to communicate.

> **RECOMMENDATION:** Avoid using inconsistent access control policies in the case of encrypted group operations.

MLS allows actors outside the group to influence the group in two ways: External signers can submit proposals for changes to the group, and new joiners can use an external join to add themselves to the group. The external_senders extension ensures that all members

agree on which signers are allowed to send proposals, but any other policies must be assured to be consistent as above.

** RECOMMENDATION:** Have an explicit group policy setting the conditions under which external joins are allowed.

5.5. Recovery After State Loss

Group members whose local MLS state is lost or corrupted can reinitialize their state by re-joining the group as a new member and removing the member representing their earlier state. An application can require that a client performing such a reinitialization prove its prior membership with a PSK.

There are a few practical challenges to this approach. For example, the application will need to ensure that all members have the required PSK, including any new members that have joined the group since the epoch in which the PSK was issued.

Reinitializing in this way does not provide the member with access to group messages from during the state loss window, but enables proof of prior membership in the group. Applications may choose various configurations for providing lost messages to valid group members that are able to prove prior membership.

5.6. Support for Multiple Devices

It is typically expected for users within a group to own various devices. A new device can be added to a group and be considered as a new client by the protocol. This client will not gain access to the history even if it is owned by someone who owns another member of the group. Restoring history is typically not allowed at the protocol level but applications can elect to provide such a mechanism outside of MLS. Such mechanisms, if used, may reduce the FS and PCS guarantees provided by MLS.

5.7. Extensibility

The MLS protocol provides several extension points where additional information can be provided. Extensions to KeyPackages allow clients to disclose additional information about their capabilities. Groups can also have extension data associated with them, and the group agreement properties of MLS will confirm that all members of the group agree on the content of these extensions.

5.8. Application Data Framing and Negotiation

Application messages carried by MLS are opaque to the protocol; they can contain arbitrary data. Each application which uses MLS needs to define the format of its application_data and any mechanism

necessary to negotiate the format of that content over the lifetime of an MLS group. In many applications this means managing format migrations for groups with multiple members who may each be offline at unpredictable times.

> **RECOMMENDATION:** Use the default content mechanism defined in [I-D.mahy-mls-content-neg], unless the specific application defines another mechanism which more appropriately addresses the same requirements for that application of MLS.

The MLS framing for application messages also provides a field where clients can send information that is authenticated but not encrypted. Such information can be used by servers that handle the message, but group members are assured that it has not been tampered with.

5.9. Federation

The protocol aims to be compatible with federated environments. While this document does not specify all necessary mechanisms required for federation, multiple MLS implementations can interoperate to form federated systems if they use compatible authentication mechanisms, ciphersuites, and infrastructure functionalities.

5.10. Compatibility with Future Versions of MLS

It is important that multiple versions of MLS be able to coexist in the future. Thus, MLS offers a version negotiation mechanism; this mechanism prevents version downgrade attacks where an attacker would actively rewrite messages with a lower protocol version than the ones originally offered by the endpoints. When multiple versions of MLS are available, the negotiation protocol guarantees that the version agreed upon will be the highest version supported in common by the group.

In MLS 1.0, the creator of the group is responsible for selecting the best ciphersuite supported across clients. Each client is able to verify availability of protocol version, ciphersuites and extensions at all times once he has at least received the first group operation message.

Each member of an MLS group advertises the protocol functionality they support. These capability advertisements can be updated over time, e.g., if client software is updated while the client is a member of a group. Thus, in addition to preventing downgrade attacks, the members of a group can also observe when it is safe to upgrade to a new ciphersuite or protocol version.

6. Operational Requirements

MLS is a security layer that needs to be integrated with an application. A fully-functional deployment of MLS will have to make a number of decisions about how MLS is configured and operated. Deployments that wish to interoperate will need to make compatible decisions. This section lists all of the dependencies of an MLS deployment that are external to the protocol specification, but would still need to be aligned within a given MLS deployment, or for two deployments to potentially interoperate.

The protocol has a built-in ability to negotiate protocol versions, ciphersuites, extensions, credential types, and additional proposal types. For two deployments to interoperate, they must have overlapping support in each of these categories. A required_capabilities extension can help maintain interoperability with a wider set of clients by ensuring that certain functionality continues to be supported by a group, even if the clients in the group aren't currently relying on it.

MLS relies on the following network services. These network services would need to be compatible in order for two different deployments based on them to interoperate.

* An **Authentication Service**, described fully in Section 3, defines the types of credentials which may be used in a deployment and provides methods for:

    1. Issuing new credentials,
    2. Validating a credential against a reference identifier, and
    3. Validating whether or not two credentials represent the same user.

* A **Delivery Service**, described fully in Section 4, provides methods for:

    1. Delivering messages sent to a group to all members in the group.
    2. Delivering Welcome messages to new members of a group.
    3. Downloading KeyPackages for specific clients, and uploading new KeyPackages for a user's own clients.

* Additional services may or may not be required depending on the application design:

  - If assisted joining is desired (meaning that the ratchet tree is not provided in Welcome messages), there must be a method to download the ratchet tree corresponding to a group.

  - If assisted joining is desired and the Delivery Service is not able to compute the ratchet tree itself (because some proposals or commits are sent encrypted), there must be a method for group members to publish the updated ratchet tree after each commit.

  - If external joiners are allowed, there must be a method to publish a serialized GroupInfo object (with an external_pub extension) that corresponds to a specific group and epoch, and keep that object in sync with the state of the group.

  - If an application chooses not to allow assisted or external joining, it may instead provide a method for external users to solicit group members (or a designated service) to add them to a group.

  - If the application uses external PSKs, or uses resumption PSKs that all members of a group may not have access to, there must be a method for distributing these PSKs to group members.

  - If an application wishes to detect and possibly discipline members that send malformed commits with the intention of corrupting a group's state, there must be a method for reporting and validating malformed commits.

MLS requires the following parameters to be defined, which must be the same for two implementations to interoperate:

* The maximum total lifetime that is acceptable for a KeyPackage.

* How long to store the resumption secret for past epochs of a group.

* The degree of tolerance that's allowed for out-of-order message delivery:

  - How long to keep unused nonce and key pairs for a sender

  - A maximum number of unused key pairs to keep.

  - A maximum number of steps that clients will move a secret tree ratchet forward in response to a single message before rejecting it.

MLS provides the following locations where an application may store arbitrary data. The format and intention of any data in these locations must align for two deployments to interoperate:

- Application data, sent as the payload of an encrypted message.
- Additional authenticated data, sent unencrypted in an otherwise encrypted message.
- Group IDs, as decided by group creators and used to uniquely identify a group.
- The application_id extension of a LeafNode.

MLS requires the following policies to be defined, which restrict the set of acceptable behavior in a group. These policies must be consistent between deployments for them to interoperate:

- A policy on when to send proposals and commits in plaintext instead of encrypted.
- A policy for which proposals are valid to have in a commit, including but not limited to:
  - When a member is allowed to add or remove other members of the group.
  - When, and under what circumstances, a reinitialization proposal is allowed.
  - When proposals from external senders are allowed.
  - When external joiners are allowed.
- A policy for when two credentials represent the same client. Note that many credentials may be issued authenticating the same identity but for different signature keys, because each credential corresponds to a different device (client) owned by the same application user. However, one device may control many signature keys but should still only be considered a single client.
- A policy on how long to allow a member to stay in a group without updating its leaf keys before removing them.

Finally, there are some additional application-defined behaviors that are partially an individual application's decision but may overlap with interoperability:

- If there's any policy on how or when to pad messages.

*If there is any policy for when to send a reinitialization proposal.

*How often clients should update their leaf keys.

*Whether to prefer sending full commits or partial/empty commits.

*Whether there should be a required_capabilities extension in groups.

7. Security and Privacy Considerations

MLS adopts the Internet threat model [RFC3552] and therefore assumes that the attacker has complete control of the network. It is intended to provide the security services described in the face of such attackers.

*The attacker can monitor the entire network.

*The attacker can read unprotected messages.

*The attacker can generate, inject and delete any message in the unprotected transport layer.

In addition, these guarantees are intended to degrade gracefully in the presence of compromise of the transport security links as well as of both clients and elements of the messaging system, as described in the remainder of this section.

Generally, MLS is designed under the assumption that the transport layer is present to protect metadata and privacy in general, while the MLS protocol is providing stronger guarantees such as confidentiality, integrity and authentication guarantees. Stronger properties such as deniability can also be achieved in specific architecture designs.

7.1. Assumptions on Transport Security Links

Any secure channel can be used as a transport layer to protect MLS messages such as QUIC, TLS, WireGuard or TOR. However, the MLS protocol is designed to consider the following threat-model:

*The attacker can read, write, and delete arbitrary messages inside the secure transport channel.

This departs from most threat models where we consider that the secure channel used for transport always provides secrecy. The reason for this consideration is that in the group setting, active malicious insiders or adversarial services are to be considered.

7.1.1. Metadata Protection for Unencrypted Group Operations

The main use of the secure transport layer for MLS is to protect the already limited amount of metadata. Very little information is contained in the unencrypted header of the MLS protocol message format for group operation messages, and application messages are always encrypted in MLS.

MLS avoids needing to send the full list of recipients to the server for dispatching messages because that list is potentially extremely large in MLS. Therefore, the metadata typically consists of a pseudo-random Group Identifier (GID), a numerical value to determine the epoch of the group (the number of changes that have been made to the group), and another numerical value referring to the specific key needed to decrypt the ciphertext content.

The MLS protocol provides an authenticated "Additional Authenticated Data" field for applications to make data available outside the MLSCiphertext.

> **RECOMMENDATION:** Use the "Additional Authenticated Data" field of the MLSCiphertext message instead of using other unauthenticated means of sending metadata throughout the infrastructure. If the data is private, the infrastructure should use encrypted Application messages instead.

Even though some of this metadata information does not consist of secret payloads, in correlation with other data a network observer might be able to reconstruct sensitive information. Using a secure channel to transfer this information will prevent a network attacker from accessing this MLS protocol metadata if it cannot compromise the secure channel.

More importantly, there is one specific case where having no secure channel to exchange the MLS messages can have a serious impact on privacy. In the case of unencrypted group operation messages, observing the signatures of the group operation messages may lead an adversary to extract information about the group memberships.

> **RECOMMENDATION:** Never use the unencrypted mode for group operations without using a secure channel for the transport layer.

7.1.2. DoS protection

In general we do not consider Denial of Service (DoS) resistance to be the responsibility of the protocol. However, it should not be possible for anyone aside from the Delivery Service to perform a trivial DoS attack from which it is hard to recover. This can be achieved through the secure transport layer.

In the centralized setting, DoS protection can typically be performed by using tickets or cookies which identify users to a service for a certain number of connections. Such a system helps in preventing anonymous clients from sending arbitrary numbers of group operation messages to the Delivery Service or the MLS clients.

> **RECOMMENDATION:** Anonymous credentials can be used in order to help DoS attacks prevention, in a privacy preserving manner. Note that the privacy of these mechanisms has to be adjusted in accordance with the privacy expected from the secure transport links. (See more discussion further down.)

7.1.3. Message Suppression and Error Correction

As noted above, MLS is designed to provide some robustness in the face of tampering within the secure transport, i.e., tampering by the Delivery Service. The confidentiality and authenticity properties of MLS prevent the DS reading or writing messages. MLS also provides a few tools for detecting message suppression, with the caveat that message suppression cannot always be distinguished from transport failure.

Each encrypted MLS message carries a "generation" number which is a per-sender incrementing counter. If a group member observes a gap in the generation sequence for a sender, then they know that they have missed a message from that sender. MLS also provides a facility for group members to send authenticated acknowledgements of application messages received within a group.

As discussed in Section 4, the Delivery Service is trusted to select the single Commit message that is applied in each epoch from among the ones sent by group members. Since only one Commit per epoch is meaningful, it's not useful for the DS to transmit multiple Commits to clients. The risk remains that the DS will use the ability maliciously.

While it is difficult or impossible to prevent a network adversary from suppressing payloads in transit, in certain infrastructures such as banks or governments settings, unidirectional transports can be used and be enforced via electronic or physical devices such as diodes. This can lead to payload corruption which does not affect the security or privacy properties of the MLS protocol but does affect the reliability of the service. In that case specific measures can be taken to ensure the appropriate level of redundancy and quality of service for MLS.

> **RECOMMENDATION:** If unidirectional transport is used for the secure transport channel, prefer using a protocol which provides Forward Error Correction.

7.2. Intended Security Guarantees

MLS aims to provide a number of security guarantees, covering authentication, as well as confidentiality guarantees to different degrees in different scenarios.

7.2.1. Message Secrecy and Authentication

MLS enforces the encryption of application messages and thus generally guarantees authentication and confidentiality of application messages sent in a group.

In particular, this means that only other members of a given group can decrypt the payload of a given application message, which includes information about the sender of the message.

Similarly, group members receiving a message from another group member can authenticate that group member as the sender of the message and verify the message's integrity.

Message content can be deniable if the signature keys are exchanged over a deniable channel prior to signing messages.

Depending on the group settings, handshake messages can be encrypted as well. If that is the case, the same security guarantees apply.

MLS optionally allows the addition of padding to messages, mitigating the amount of information leaked about the length of the plaintext to an observer on the network.

7.2.2. Forward and Post-Compromise Security

MLS provides additional protection regarding secrecy of past messages and future messages. These cryptographic security properties are Forward Secrecy (FS) and Post-Compromise Security (PCS).

FS means that access to all encrypted traffic history combined with an access to all current keying material on clients will not defeat the secrecy properties of messages older than the oldest key of the compromised client. Note that this means that clients have the extremely important role of deleting appropriate keys as soon as they have been used with the expected message, otherwise the secrecy of the messages and the security for MLS is considerably weakened.

PCS means that if a group member's state is compromised at some time t but the group member subsequently performs an update at some time t', then all MLS guarantees apply to messages sent by the member after time t', and by other members after they have processed the update. For example, if an attacker learns all secrets known to

Alice at time t, including both Alice's long-term secret keys and all shared group keys, but Alice performs a key update at time t', then the attacker is unable to violate any of the MLS security properties after the updates have been processed.

Both of these properties are satisfied even against compromised DSs and ASs.

7.2.3. Non-Repudiation vs Deniability

MLS provides strong authentication within a group, such that a group member cannot send a message that appears to be from another group member. Additionally, some services require that a recipient be able to prove to the service provider that a message was sent by a given client, in order to report abuse. MLS supports both of these use cases. In some deployments, these services are provided by mechanisms which allow the receiver to prove a message's origin to a third party. This is often called "non-repudiation".

Roughly speaking, "deniability" is the opposite of "non-repudiation", i.e., the property that it is impossible to prove to a third party that a message was sent by a given sender. MLS does not make any claims with regard to deniability. It may be possible to operate MLS in ways that provide certain deniability properties, but defining the specific requirements and resulting notions of deniability requires further analysis.

7.3. Endpoint Compromise

The MLS protocol adopts a threat model which includes multiple forms of endpoint/client compromise. While adversaries are in a very strong position if they have compromised an MLS client, there are still situations where security guarantees can be recovered thanks to the PCS properties achieved by the MLS protocol.

In this section we will explore the consequences and recommendations regarding the following compromise scenarios:

- The attacker has access to a specific symmetric encryption key
- The attacker has access to the group secrets for one group
- The attacker has access to a signature oracle for any group
- The attacker has access to the signature key for one group
- The attacker has access to all secrets of a user for all groups (full state compromise)

Recall that the MLS protocol provides chains of AEAD keys, per sender that are generated from Group Secrets. These keys are used to protect MLS Plaintext messages which can be Group Operation or Application messages. The Group Operation messages offer an additional protection as the secret exchanged within the TreeKEM group key agreement are public-key encrypted to subgroups with HPKE.

7.3.1. Compromise of AEAD key material

In some circumstances, adversaries may have access to specific AEAD keys and nonces which protect an Application or a Group Operation message. While this is a very weak kind of compromise, it can be realistic in cases of implementation vulnerabilities where only part of the memory leaks to the adversary.

When an AEAD key is compromised, the adversary has access to a set of AEAD keys for the same chain and the same epoch, hence can decrypt messages sent using keys of this chain. An adversary cannot send a message to a group which appears to be from any valid client since they cannot forge the signature.

The MLS protocol will ensure that an adversary cannot compute any previous AEAD keys for the same epoch, or any other epochs. Because of its Forward Secrecy guarantees, MLS will also retain secrecy of all other AEAD keys generated for *other* MLS clients, outside this dedicated chain of AEAD keys and nonces, even within the epoch of the compromise. However the MLS protocol does not provide Post Compromise Secrecy for AEAD encryption within an epoch. This means that if the AEAD key of a chain is compromised, the adversary can compute an arbitrary number of subsequent AEAD keys for that chain.

These guarantees are ensured by the structure of the MLS key schedule which provides Forward Secrecy for these AEAD encryptions, across the messages within the epoch and also across previous epochs. Those chains are completely disjoint and compromising keys across the chains would mean that some Group Secrets have been compromised, which is not the case in this attack scenario (we explore stronger compromise scenarios as part of the following sections).

MLS provides Post-Compromise Secrecy against an active adaptive attacker across epochs for AEAD encryption, which means that as soon as the epoch is changed, if the attacker does not have access to more secret material they won't be able to access any protected messages from future epochs.

In the case of an Application message, an AEAD key compromise means that the encrypted application message will be leaked as well as the signature over that message. This means that the compromise has both

confidentiality and privacy implications on the future AEAD encryptions of that chain. In the case of a Group Operation message, only the privacy is affected, as the signature is revealed, because the secrets themselves are protected by HPKE encryption.

Note that under that compromise scenario, authentication is not affected in neither of these cases. As every member of the group can compute the AEAD keys for all the chains (they have access to the Group Secrets) in order to send and receive messages, the authentication provided by the AEAD encryption layer of the common framing mechanism is very weak. Successful decryption of an AEAD encrypted message only guarantees that a member of the group sent the message.

7.3.2. Compromise of the Group Secrets of a single group for one or more group epochs

The attack scenario considering an adversary gaining access to a set of Group secrets is significantly stronger. This can typically be the case when a member of the group is compromised. For this scenario, we consider that the signature keys are not compromised. This can be the case for instance if the adversary has access to part of the memory containing the group secrets but not to the signature keys which might be stored in a secure enclave.

In this scenario, the adversary gains the ability to compute any number of AEAD encryption keys for any AEAD chains and can encrypt and decrypt all messages for the compromised epochs.

If the adversary is passive, it is expected from the PCS properties of the MLS protocol that, as soon as an honest Commit message is sent by the compromised party, the next epochs will provide message secrecy.

If the adversary is active, the adversary can follow the protocol and perform updates on behalf of the compromised party with no ability to an honest group to recover message secrecy. However, MLS provides PCS against active adaptive attackers through its Remove group operation. This means that, as long as other members of the group are honest, the protocol will guarantee message secrecy for all messages exchanged in the epochs after the compromised party has been removed.

7.3.3. Compromise by an active adversary with the ability to sign messages

Under such a scenario, where an active adversary has compromised an MLS client, two different settings emerge. In the strongest compromise scenario, the attacker has access to the signing key and can forge authenticated messages. In a weaker, yet realistic

scenario, the attacker has compromised a client but the client signature keys are protected with dedicated hardware features which do not allow direct access to the value of the private key and instead provide a signature API.

When considering an active adaptive attacker with access to a signature oracle, the compromise scenario implies a significant impact on both the secrecy and authentication guarantees of the protocol, especially if the attacker also has access to the group secrets. In that case both secrecy and authentication are broken. The attacker can generate any message, for the current and future epochs until an honest update from the compromised client happens.

Note that under this compromise scenario, the attacker can perform all operations which are available to an legitimate client even without access to the actual value of the signature key.

Without access to the group secrets, the adversary will not have the ability to generate messages which look valid to other members of the group and to the infrastructure as they need to have access to group secrets to compute the encryption keys or the membership tag.

7.3.4. Compromise of the authentication with access to a signature key

The difference between having access to the value of the signature key and only having access to a signing oracle is not about the ability of an active adaptive network attacker to perform different operations during the time of the compromise, the attacker can perform every operation available to a legitimate client in both cases.

There is a significant difference, however in terms of recovery after a compromise.

Because of the PCS guarantees provided by the MLS protocol, when a previously compromised client performs an honest Commit which is not under the control of the adversary, both secrecy and authentication of messages can be recovered in the case where the attacker didn't get access to the key. Because the adversary doesn't have the key and has lost the ability to sign messages, they cannot authenticate messages on behalf of the compromised party, even if they still have control over some group keys by colluding with other members of the group.

This is in contrast with the case where the signature key is leaked. In that case PCS of the MLS protocol will eventually allow recovery of the authentication of messages for future epochs but only after compromised parties refresh their credentials securely.

Beware that in both oracle and private key access, an active adaptive attacker, can follow the protocol and request to update its own credential. This in turn induces a signature key rotation which could provide the attacker with part or the full value of the private key depending on the architecture of the service provider.

> **RECOMMENDATION:** Signature private keys should be compartmentalized from other secrets and preferably protected by an HSM or dedicated hardware features to allow recovery of the authentication for future messages after a compromise.

7.3.5. Security consideration in the context of a full state compromise

In real-world compromise scenarios, it is often the case that adversaries target specific devices to obtain parts of the memory or even the ability to execute arbitrary code in the targeted device.

Also, recall that in this setting, the application will often retain the unencrypted messages. If so, the adversary does not have to break encryption at all to access sent and received messages. Messages may also be sent by using the application to instruct the protocol implementation.

> **RECOMMENDATION:** If messages are stored on the device, they should be protected using encryption at rest, and the keys used should be stored securely using dedicated mechanisms on the device.

> **RECOMMENDATION:** If the threat model of the system is against an adversary which can access the messages on the device without even needing to attack MLS, the application should delete plaintext messages and ciphertexts immediately after encryption or decryption.

Even though, from the strict point of view of the security formalization, a ciphertext is always public and will forever be, there is no loss in trying to erase ciphertexts as much as possible.

Note that this document makes a clear distinction between the way signature keys and other group shared secrets must be handled. In particular, a large set of group secrets cannot necessarily be assumed to be protected by an HSM or secure enclave features. This is especially true because these keys are extremely frequently used and changed with each message received by a client.

However, the signature private keys are mostly used by clients to send a message. They also are providing the strong authentication guarantees to other clients, hence we consider that their protection by additional security mechanism should be a priority.

Overall there is no way to detect or prevent these compromise, as discussed in the previous sections, performing separation of the application secret states can help recovery after compromise, this is the case for signature keys but similar concern exists for the encryption private key used in the TreeKEM Group Key Agreement.

> **RECOMMENDATION:** The secret keys used for public key encryption should be stored similarly to the way the signature keys are stored, as keys can be used to decrypt the group operation messages and contain the secret material used to compute all the group secrets.

Even if secure enclaves are not perfectly secure, or even completely broken, adopting additional protections for these keys can ease recovery of the secrecy and authentication guarantees after a compromise where, for instance, an attacker can sign messages without having access to the key. In certain contexts, the rotation of credentials might only be triggered by the AS through ACLs, hence be outside of the capabilities of the attacker.

7.4. Service Node Compromise

7.4.1. General considerations

7.4.1.1. Privacy of the network connections

There are many scenarios leading to communication between the application on a device and the Delivery Service or the Authentication Service. In particular when:

* The application connects to the Authentication Service to generate or validate a new credential before distributing it.
* The application fetches credentials at the Delivery Service prior to creating a messaging group (one-to-one or more than two clients).
* The application fetches service provider information or messages on the Delivery Service.
* The application sends service provider information or messages to the Delivery Service.

In all these cases, the application will often connect to the device via a secure transport which leaks information about the origin of the request such as the IP address and depending on the protocol the MAC address of the device.

Similar concerns exist in the peer-to-peer use cases of MLS.

> **RECOMMENDATION:** In the case where privacy or anonymity is important, using adequate protection such as TOR or a VPN can improve metadata protection.

More generally, using anonymous credentials in an MLS based architecture might not be enough to provide strong privacy or anonymity properties.

7.4.2. Delivery Service Compromise

MLS is intended to provide strong guarantees in the face of compromise of the DS. Even a totally compromised DS should not be able to read messages or inject messages that will be acceptable to legitimate clients. It should also not be able to undetectably remove, reorder or replay messages.

However, a DS can mount a variety of DoS attacks on the system, including total DoS attacks (where it simply refuses to forward any messages) and partial DoS attacks (where it refuses to forward messages to and from specific clients). As noted in Section 4.3, these attacks are only partially detectable by clients without an out-of-band channel. Ultimately, failure of the DS to provide reasonable service must be dealt with as a customer service matter, not via technology.

Because the DS is responsible for providing the initial keying material to clients, it can provide stale keys. This does not inherently lead to compromise of the message stream, but does allow it to attack forward security to a limited extent. This threat can be mitigated by having initial keys expire.

7.4.2.1. Privacy of delivery and push notifications

An important mechanism that is often ignored from the privacy considerations are the push-tokens. In many modern messaging architectures, applications are using push notification mechanisms typically provided by OS vendors. This is to make sure that when messages are available at the Delivery Service (or by other mechanisms if the DS is not a central server), the recipient application on a device knows about it. Sometimes the push notification can contain the application message itself which saves a round trip with the DS.

To "push" this information to the device, the service provider and the OS infrastructures use unique per-device, per-application identifiers called push-tokens. This means that the push notification provider and the service provider have information on which devices receive information and at which point in time.

Even though they can't necessarily access the content, which is typically encrypted MLS messages, the service provider and the push notification provider have to be trusted to avoid making correlation on which devices are recipients of the same message.

For secure messaging systems, push notification are often sent real-time as it is not acceptable to create artificial delays for message retrieval.

> **RECOMMENDATION:** If real time notifications are not necessary and that specific steps must be taken to improve privacy, one can delay notifications randomly across recipient devices using a mixnet or other techniques.

Note that it is quite easy for legal requests to ask the service provider for the push-token associated to an identifier and perform a second request to the company operating the push-notification system to get information about the device, which is often linked with a real identity via a cloud account, a credit card or other information.

> **RECOMMENDATION:** If stronger privacy guarantees are needed vis-a-vis the push notification provider, the client can choose to periodically connect to the Delivery Service without the need of a dedicated push notification infrastructure.

7.4.3. Authentication Service Compromise

The Authentication Service design is left to the infrastructure designers. In most designs, a compromised AS is a serious matter, as the AS can serve incorrect or attacker-provided identities to clients.

*The attacker can link an identity to a credential

*The attacker can generate new credentials

*The attacker can sign new credentials

*The attacker can publish or distribute credentials

Infrastructures that provide cryptographic material or credentials in place of the MLS client (which is under the control of the user) have often the ability to use the associated secrets to perform operations on behalf of the user, which is unacceptable in many situations. Other mechanisms can be used to prevent this issue, such

as the service blessing cryptographic material used by an MLS client.

> **RECOMMENDATION:** Make clients submit signature public keys to the AS, this is usually better than the AS generating public key pairs because the AS cannot sign on behalf of the client. This is a benefit of a Public Key Infrastructure in the style of the Internet PKI.

An attacker that can generate or sign new credentials may or may not have access to the underlying cryptographic material necessary to perform such operations. In that last case, it results in windows of time for which all emitted credentials might be compromised.

> **RECOMMENDATION:** Using HSMs to store the root signature keys to limit the ability of an adversary with no physical access to extract the top-level signature key.

7.4.3.1. Authentication compromise: Ghost users and impersonations

One thing for which the MLS Protocol is designed for is to make sure that all clients know who is in the group at all times. This means that - if all Members of the group and the Authentication Service are honest - no other parties than the members of the current group can read and write messages protected by the protocol for that Group.

Beware though, the link between the cryptographic identity of the Client and the real identity of the User is important. With some Authentication Service designs, a private or centralized authority can be trusted to generate or validate signature keypairs used in the MLS protocol. This is typically the case in some of the biggest messaging infrastructures.

While this service is often very well protected from external attackers, it might be the case that this service is compromised. In such infrastructure, the AS could generate or validate a signature keypair for an identity which is not the expected one. Because a user can have many MLS clients running the MLS protocol, it possibly has many signature keypairs for multiple devices.

In the case where an adversarial keypair is generated for a specific identity, an infrastructure without any transparency mechanism or out-of-band authentication mechanism could inject a malicious client into a group by impersonating a user. This is especially the case in large groups where the UI might not reflect all the changes back to the users.

> **RECOMMENDATION:** Make sure that MLS clients reflect all the membership changes to the users as they happen. If a choice has

to be made because the number of notifications is too high, a public log should be maintained in the state of the device so that the user can examine it.

While the ways to handle MLS credentials are not defined by the protocol or the architecture documents, the MLS protocol has been designed with a mechanism that can be used to provide out-of-band authentication to users. The "authentication_secret" generated for each user at each epoch of the group is a one-time, per client, authentication secret which can be exchanged between users to prove their identity to each other. This can be done for instance using a QR code that can be scanned by the other parties.

Another way to improve the security for the users is to provide a transparency mechanism which allows each user to check if credentials used in groups have been published in the transparency log. Another benefit of this mechanism is for revocation. The users of a group could check for revoked keys (in case of compromise detection) using a mechanism such as CRLite or some more advanced privacy preserving technology.

RECOMMENDATION: Provide a Key Transparency and Out-of-Band authentication mechanisms to limit the impact of an Authentication Service compromise.

We note, again, that as described prior to that section, the Authentication Service is facultative to design a working infrastructure and can be replaced by many mechanisms such as establishing prior one-to-one deniable channels, gossiping, or using TOFU for credentials used by the MLS Protocol.

Another important consideration is the ease of redistributing new keys on client compromise, which helps recovering security faster in various cases.

7.4.3.2. Privacy of the Group Membership

Often, expectation from users is that the infrastructure will not retain the ability to constantly map the user identity to signature public keys of the MLS protocol. Some infrastructures will keep a mapping between signature public keys of clients and user identities. This can benefit an adversary that has compromised the AS (or required access according to regulation) the ability of monitoring unencrypted traffic and correlating the messages exchanged within the same group.

RECOMMENDATION: Always use encrypted group operation messages to reduce issues related to privacy.

In certain cases, the adversary can access specific bindings between public keys and identities. If the signature keys are reused across groups, the adversary can get more information about the targeted user.

> **RECOMMENDATION:** Do not use the same signature keypair across groups.

> **RECOMMENDATION:** Separate the service binding the identities and the public keys from the service which generates or validates the credentials or cryptographic material of the Clients.

7.5. Considerations for attacks outside of the threat model

Physical attacks on devices storing and executing MLS principals are not considered in depth in the threat model of the MLS protocol. While non-permanent, non-invasive attacks can sometimes be equivalent to software attacks, physical attacks are considered outside of the MLS threat model.

Compromise scenarios typically consist in a software adversary, which can maintain active adaptive compromise and arbitrarily change the behavior of the client or service.

On the other hand, security goals consider that honest clients will always run the protocol according to its specification. This relies on implementations of the protocol to securely implement the specification, which remains non-trivial.

> **RECOMMENDATION:** Additional steps should be taken to protect the device and the MLS clients from physical compromise. In such settings, HSMs and secure enclaves can be used to protect signature keys.

7.6. Cryptographic Analysis of the MLS Protocol

Various academic works have analyzed MLS and the different security guarantees it aims to provide. The security of large parts of the protocol has been analyzed by [BBN19] (draft 7), [ACDT21] (draft 11) and [AJM20] (draft 12).

Individual components of various drafts of the MLS protocol have been analyzed in isolation and with differing adversarial models, for example, [BBR18], [ACDT19], [ACCKKMPPWY19], [AJM20] and [ACJM20] analyze the ratcheting tree as the sub-protocol of MLS that facilitates key agreement, while [BCK21] analyzes the key derivation paths in the ratchet tree and key schedule. Finally, [CHK19] analyzes the authentication and cross-group healing guarantees provided by MLS.

8. Informative References

*ACDT19: https://eprint.iacr.org/2019/1189

*ACCKKMPPWY19: https://eprint.iacr.org/2019/1489

*ACJM20: https://eprint.iacr.org/2020/752

*AJM20: https://eprint.iacr.org/2020/1327

*ACDT21: https://eprint.iacr.org/2021/1083

*AHKM21: https://eprint.iacr.org/2021/1456

*CHK19: https://eprint.iacr.org/2021/137

*BCK21: https://eprint.iacr.org/2021/137

*BBR18: https://hal.inria.fr/hal-02425247

*BBN19: https://hal.laas.fr/INRIA/hal-02425229

9. IANA Considerations

This document makes no requests of IANA.

10. References

10.1. Normative References

[I-D.ietf-mls-protocol]
Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", Work in Progress, Internet-Draft, draft-ietf-mls-protocol-14, 3 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-14>.

10.2. Informative References

[I-D.mahy-mls-content-neg]
Mahy, R., "Content Negotiation for Message Layer Security (MLS)", Work in Progress, Internet-Draft, draft-mahy-mls-content-neg-00, 31 March 2022, <https://datatracker.ietf.org/doc/html/draft-mahy-mls-content-neg-00>.

[KeyTransparency] Google, "Key Transparency", 2017, <https://KeyTransparency.org>.

[Loopix] Piotrowska, A. M., Hayes, J., Elahi, T., Meiser, S., and G. Danezis, "The Loopix Anonymity System", 2017.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/rfc/rfc3552>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/rfc/rfc6120>.

Contributors

Richard Barnes
Cisco

Email: rlb@ipv.sx

Katriel Cohn-Gordon
Meta Platforms

Email: me@katriel.co.uk

Cas Cremers
CISPA Helmholtz Center for Information Security

Email: cremers@cispa.de

Britta Hale
Naval Postgraduate School

Email: britta.hale@nps.edu

Konrad Kohbrok

Email: konrad.kohbrok@datashrine.de

Brendan McMillion

Email: brendanmcmillion@gmail.com

Thyla van der Merwe

Email: tjvdmerwe@gmail.com

Jon Millican
Meta Platforms

Email: jmillican@fb.com

Raphael Robert

Email: ietf@raphaelrobert.com

Authors' Addresses

Benjamin Beurdouche
Inria & Mozilla

Email: ietf@beurdouche.com

Eric Rescorla
Mozilla

Email: ekr@rtfm.com

Emad Omara
Google

Email: emadomara@google.com

Srinivas Inguva
Twitter

Email: singuva@twitter.com

Albert Kwon
MIT

Email: kwonal@mit.edu

Alan Duric
Wire

Email: alan@wire.com