Authors: B. Beurdouche   E. Rescorla   E. Omara   S. Inguva
         Inria & Mozilla   Mozilla   Google   Twitter
         A. Kwon   A. Duric
         MIT   Wire

The Messaging Layer Security (MLS) Architecture

Abstract

The Messaging Layer Security (MLS) protocol [MLSPROTO] document has the role of defining a Group Key Agreement, all the necessary cryptographic operations, and serialization/deserialization functions necessary to create a scalable and secure group messaging protocol. The MLS protocol is meant to protect against eavesdropping, tampering, message forgery, and provide good properties such as forward-secrecy (FS) and post-compromise security (PCS) in the case of past or future device compromises.

This document, on the other hand is intended to describe 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 mechanism that are part of the MLS protocol (ie. frequency of public encryption key rotation).

The document also extends the guidance to 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, most will vastly influence the overall security guarantees that are achieved by the overall messaging system. 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

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.

DISCLAIMER: A lot of work is still ongoing on the current version of this draft. Especially, this preliminary writing of the security considerations has not been reviewed by the working group yet and might contain errors. Please file an issue on the document's GitHub if you find errors.

[[TODO: Remove disclaimer.]]

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 when the service provider they are using performs unsatisfactorily.

Messaging Layer Security (MLS) specifies an architecture (this document) and an abstract protocol [MLSPROTO] 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]. In addition, it does not specify a complete wire encoding, but rather a set of abstract data structures which can then be mapped onto a variety of concrete encodings, such as TLS [RFC8446], CBOR [RFC7049], and JSON [RFC7159]. Implementations which adopt compatible encodings will have some degree of interoperability at the message level, though they may have incompatible 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 maintaining a binding between a unique identifier (identity)

and the public key material (credential) used for authentication
in the MLS protocol. This functionality must also be able to
generate these credentials or validate them if they are provided
by MLS clients.

*A Delivery Service (DS) functionality which can receive and
redistributing 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 it is not necessarily the case.

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.

```
    ----------------        --------------
  | Authentication |      | Delivery     |
  | Service (AS)   |      | Service (DS) |
    ----------------        --------------
                      /        |         \          Group
                     / **********************************
                    /  *       |          \           *
         ----------    *  ----------       ----------  *
        | Client 0 |   * | Member 1 |     | Member N | *
         ----------    *  ----------       ----------  *
        ............   *  ............     ............ *
         User 0        *  User 0           User 1      *
                       *                               *
                       **********************************
```
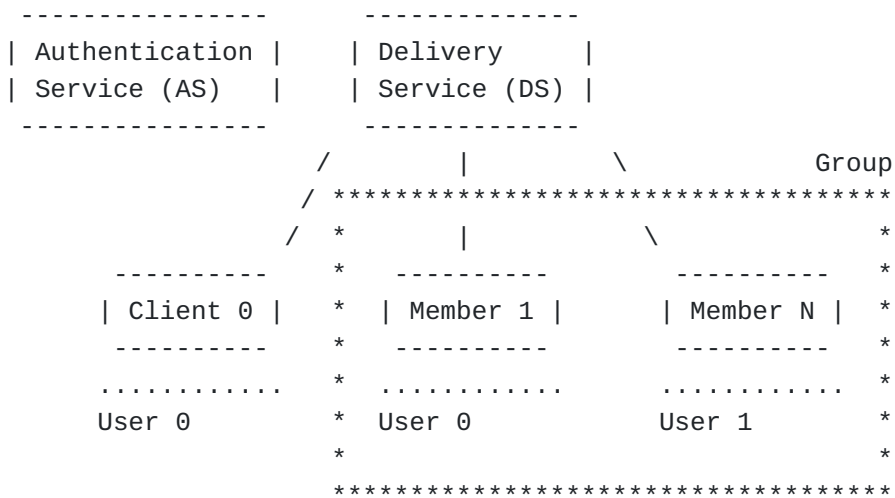
In many systems, the AS and the DS are actually operated by the same entity and may even be the same server. However, they are logically distinct and, in other systems, may be operated by different entities. Other partitions are also possible, such as having a separate directory functionality or service.

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 3.2.2](). 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 by
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 and its signature key pair uniquely defines its identity to
other clients or members in the Group. In some messaging systems,
clients belonging to the same user must all share the same identity
key pair, but MLS does not assume this.

Users will typically own multiple clients, potentially one or more
per end-user devices (phones, web clients or other devices...) and
may choose to authenticate using the same signature key across
devices, using one signature key per device or even one signature
key per group.

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.

2.2. Authentication Service

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

  1. authenticate the credentials (i.e. the identity/signature
     keypair) used in a group

2. authenticate messages sent in groups given the signature over
   the message and the sending member's 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.

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 Section 3.4.3.

2.2.1. Credential Authentication

In many cases, the first functionality might be provided by a
service which fulfills a role similar to a certification authority
in the WebPKI: it provides a binding of an identity (e.g., a user
name, phone number, email address, etc) to a signature key. The
identity/signature key pair can then either be used directly in a
group, or as an root of trust which in turn authenticates
credentials used in the group.

The flexibility afforded by the latter option allows for multiple
infrastructure considerations and has the benefit of providing ways
to use different signature keys across different groups by using
hierarchical authentication keys. This flexibility also comes at the
price of a security tradeoff, described in the security
considerations, between potential unlinkability of the signature
keys across groups and the amount of time required to reinstate
authentication and secrecy of messages after the compromise of a
device.

2.2.2. Message Authentication

MLS messages are authenticated by a signature conforming to the
signature scheme of the group's ciphersuite. To allow for message
deniability (see Section Section 3.2.3), messages are not required
to be signed by the private key corresponding to a member's
credential, but the key must be authenticated using some mechanism.
Thus, message authentication relies on the accuracy of the key's
authentication vice the credential authentication.

While credential authentication can be performed by a separate
entity, message authentication should be performed by each member
separately due to the encryption layer of the protocol which
protects the signature of the message.

2.3. Delivery Service

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

  *To act 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.

  *To route messages between clients and to act as a message
   broadcaster, taking in one message and forwarding it to multiple
   clients (also known as "server side fanout").

Because the MLS protocol provides a way for clients to send and
receive application messages asynchronously, it only provides causal
ordering of application messages from senders while it has to
enforce global ordering of group operations to provide Group
Agreement. [[TODO: Casual ordering?]]

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.

2.3.1. 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 and 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 signed
   with the signature public key associated with 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.

2.3.2. 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.

2.3.3. Delivery of messages and attachments

The main responsibility of the Delivery Service is to ensure delivery of messages. Specifically, we assume that Delivery Services provide:

  *Reliable delivery: when a message is provided to the Delivery
   Service, it is eventually delivered to all clients.

  *In-order delivery: messages are delivered to the group in the
   order they are received by the Delivery Service and in
   approximately the order in which they are sent by clients. The
   latter is an approximate guarantee because multiple clients may
   send messages at the same time and so the Delivery Service needs
   some latitude in enforcing ordering across clients.

  *Consistent ordering: the Delivery Service must ensure that all
   clients have the same view of message ordering for
   cryptographically relevant operations. This means that the
   Delivery Service MUST enforce global consistency of the ordering
   of group operation messages.

Note that the protocol provides three important pieces of information within an MLSCiphertext message in order to provide ordering:

  *The Group Identifier (GID) 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 GID, 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.

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 detect this misconduct. However, the protocol relies on the ordering, and on the fact that only one honest group operation message is fanned-out to clients per Epoch, to provide clients with a consistent view of the evolving Group State.

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.

2.3.4. Membership knowledge

Group membership is itself sensitive information and MLS is designed to drastically limit the amount of persistant metadata. However, large groups often require an infrastructure which provides server fanout. In the case of client fanout, the destinations 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 break of authentication or confidentiality, 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.

2.3.5. 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 mandate a silent key update from clients that is not attached to other messaging traffic, thus containing the threat of compromise. The precise details of such mechanisms are a matter of local policy and beyond the scope of this document.

2.4. Functional Requirements

MLS is designed as a large scale group messaging protocol and hence
aims to provide 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 as large
as 50,000 members, typically including many users using multiple
devices.

2.4.1. 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. Regardless, MLS does not allow for or support
addition or removal of group members without informing all other
members.

Once a client is part of a group, the set of devices controlled by
the user can only be altered by an authorized member of the group.
This authorization could depend on the application: some
applications might want to allow certain members of the group to add
or remove devices on behalf of another member, while other
applications might want a more strict policy and allow only the
owner of the devices to add or remove them at the potential cost of
weaker PCS guarantees. Application setup may also determine other
forms of membership validity, e.g. through an identity key alignment
to the member with separate signature keys per device. 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.

[[OPEN ISSUE: Above paragraph conflicts slightly under assumptions
about multiple device memberships vs. those described below under
"Support for Multiple Devices"]]

Members who are removed from a group do not enjoy special
privileges: compromise of a removed group member does not affect the
security of messages sent after their removal but might affect
previous messages if the group secrets have not been deleted
properly.

2.4.2. Parallel Groups

Any user 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.

Some applications may strengthen connectivity among parallel groups
by requiring periodic key updates from a user across all groups in
which they have membership, or using the PSK mechanism to link
healing properties among parallel groups. Such application choices
however are outside the scope of MLS.

2.4.3. Security of Attachments

The security properties expected for attachments in the MLS protocol
are very similar to the ones expected from messages. The distinction
between messages and attachments stems from the fact that the
typical average time between the download of a message and the one
from the attachments may be different. For many reasons (a typical
reason being the lack of high bandwidth network connectivity), the
lifetime of the cryptographic keys for attachments is usually higher
than for messages, hence slightly weakening the PCS guarantees for
attachments.

2.4.4. 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 shared keys, add or
remove new members, and send messages and attachments without
waiting for another user's reply.

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

2.4.5. 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. An application
can, for example, decide to provide specific permissions to a group
administrator that will be the one to perform add and remove
operations, but the flexibility is immense here. 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, the application level must make sure 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.

2.4.6. Recovery After State Loss

Group members whose local MLS state is lost or corrupted can reinitialize their state and continue participating in the group. This 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.

[[OPEN ISSUE: The previous statement seems too strong, establish what exact functional requirement we have regarding state recovery. Previously: "This may entail some level of message loss, but does not result in permanent exclusion from the group." -- Statement edited]]

2.4.7. 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 undermine
the FS and PCS guarantees provided by MLS.

2.4.8. Extensibility / Pluggability

Messages that do not affect the group state can carry an arbitrary
payload with the purpose of sharing that payload between group
members. No assumptions are made about the format of the payload.

2.4.9. 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.

2.4.10. 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.

3. 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 and inject 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.

3.1. 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 and write 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.

3.1.1. 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.

Contrary to popular messaging services, the full list of recipients cannot be sent 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 index referring to the key needed to decrypt the ciphertext content, and another numerical value to determine the epoch of the group (the number of group operations that have been performed).

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

   RECOMMENDATION: Use the "Authenticated Additional 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
to access 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.

3.1.2. 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.)

3.1.3. Message Suppression and Error Correction

The MLS protocol is particularly sensitive about group operation
message loss and reordering. This is because in the default setting,
MLS clients have to process those specific messages in order to have
a synchronized group state, after what the MLS protocol efficiently
generates keys for application messages. [[TODO: It is unclear from
this text whether MLS is "sensitive" in that it provides additional
constraints to prevent this, or is "sensitive" in that it is
vulnerable. Need to clarify]]

The Delivery Service can have the role of helping with reliability, but is mainly useful for reliability in the asynchronous aspect of the communication between MLS clients.

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.

3.2. 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.

[[TODO: Authentication guarantees at the moment of joining a group are interesting and I don't see a section where it would fit. I'm thinking in particular about the parent hash and tree hashes in combination with with signatures and the key schedule. I know that several groups have worked on this and results are scattered between a few papers. In particular, I think the guarantees for a member being added to a new group are interesting.]]

3.2.1. 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.

3.2.2. 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.

3.2.3. 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 if often called "non-repudiation"), but it should
also be possible to operate MLS in a "deniable" mode where such
proof is not possible.

3.3. 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)

[[TODO: Cite the research papers in the context of these compromise
models]]

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.

3.3.1. 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 adaptative 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.

3.3.2. 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.

3.3.3. 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 adaptative 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.

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

DISCLAIMER: Significant work remains in this section. [[TODO: Remove disclaimer.]]

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 adaptative network attacker to perform different operations during the time of the compromise, the attacker can perform every operations 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 adaptative attacker, can follow the protocol and request to update its own credential. This in turn induce 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 compromised.

Even if the dedicated hardware approach is used, ideally, neither the Client or the Authentication service alone should provide the signature private key. Both should contribute to the key and it should be stored securely by the client with no direct access.

3.3.5. 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 send 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 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 key 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.

[[TODO: Considerations for Signature keys being :reused or not across groups]]

3.3.6. More attack scenarios

[[TODO: Make examples for more complex attacks, cross groups, multi collusions...]]

[[TODO: Do we discuss PCFS in this document? If yes, where?]]

3.4. Service Node Compromise

3.4.1. General considerations

3.4.1.1. 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 credential in an MLS based
architecture might not be enough to provide strong privacy or
anonymity properties.

3.4.2. 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 2.3.3,
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.

3.4.2.1. 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 notification 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 of the push notification provider, the client can choose to periodically connect to the Delivery Service without the need of a dedicated push notification infrastructure.

3.4.3. 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 credential 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.

3.4.3.1. 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 the 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 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.

3.4.3.2. 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 correlate 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 to 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.

3.5. 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 sometime 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 adaptative 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
> setting, HSMs and secure enclaves can be used to protect
> signature keys.

> More information will be available in the Server-Assist draft.

[[TODO: Reference to server assist when the draft is available.]]

4. IANA Considerations

This document makes no requests of IANA.

5. Contributors

  *Katriel Cohn-Gordon

   University of Oxford

   me@katriel.co.uk

   *Cas Cremers

   CISPA Helmholtz Center for Information Security

   cremers@cispa.de

   *Britta Hale

   Naval Postgraduate School

   britta.hale@nps.edu

   *Konrad Kohbrok

   Wire

   konrad@wire.com

   *Thyla van der Merwe

   Royal Holloway, University of London

   thyla.van.der@merwe.tech

   *Jon Millican

   Facebook

   jmillican@fb.com

   *Raphael Robert

   Wire

   raphael@wire.com

6. Informative References

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

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

   [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/
                   info/rfc3552>.

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

     [RFC7049]   Bormann, C. and P. Hoffman, "Concise Binary Object
                   Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
                   October 2013, <https://www.rfc-editor.org/info/rfc7049>.

     [RFC7159]   Bray, T., Ed., "The JavaScript Object Notation (JSON)
                   Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159,
                   March 2014, <https://www.rfc-editor.org/info/rfc7159>.

     [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>.

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