Workgroup: Network Working Group
Internet-Draft: draft-ietf-mls-architecture-05
Published: 26 July 2020
Intended Status: Informational
Expires: 27 January 2021
Authors: E. Omara B. Beurdouche E. Rescorla S. Inguva
Google INRIA Mozilla Twitter
A. Kwon A. Duric
MIT Wire

The Messaging Layer Security (MLS) Architecture

Abstract

This document describes the reference architecture, functional and security requirements for the Messaging Layer Security (MLS) protocol. MLS provides a security layer for group messaging applications, where the number of clients ranges from two to many. It is meant to protect against eavesdropping, tampering, and message forgery.

Discussion Venues

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

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

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 a concrete protocol 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.

This document is intended to describe the overall messaging system architecture which the MLS protocol fits into, including the operational requirements needed to achieve a functional system, and to describe the security goals it is intended to fulfill.

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 services that allow clients to prepare for sending and receiving messages securely:

* An Authentication Service (AS) which is responsible for maintaining user long term identities, issuing credentials which allow them to authenticate each other, and potentially allowing users to discover each other's long-term identity keys.

* A Delivery Service (DS) which is responsible for receiving 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 to a group 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 clients in order to proceed with the group secret key establishment process.

```
   ----------------      --------------
  | 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, hence we show them as being separate here. Other partitions are also possible, such as having a separate directory server.

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 addtion, they might choose to update their key material which provides post-compromise security Section 3.2.2.1. 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 a 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 contributed to the group secret, other members cannot assume she is a member of the group.

2.2. Authentication Service

The basic function of the Authentication Service (AS) is to provide a trusted mapping from user identities (usernames, phone numbers, etc.), to long-term identity keys, which may either be one per Client or may be shared amongst the clients attached to a user.

The Authentication Service (AS) is expected to play multiple roles in the architecture:

- A certification authority or similar service which signs some sort of portable credential binding an identity to a signature key.

- A directory server which provides the key for a given identity (presumably this connection is secured via some form of transport security such as TLS).

The MLS protocol assumes a signature keypair for authentication of messages. It is important to note that this signature keypair might be the identity keypair directly, or a different signature keypair for which the public key has been for example signed by the identity private key. This flexibility 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.

Ultimately, the only requirement is for the applications to be able to check the credential containing the protocol signing key and the identity against the Authentication Service at any time.

By definition, the Authentication Service is invested with a large amount of trust. A malicious AS can impersonate - or allow an attacker to impersonate - any user of the system. As a corollary, by impersonating identities authorized to be members of a group, an AS can break confidentiality.

This risk can be mitigated by publishing the binding between identities and keys in a public log such as Key Transparency (KT) [KeyTransparency]. It is possible to build a functional MLS system without any kind of public key logging, but such a system will necessarily be somewhat vulnerable to attack by a malicious or untrusted AS.

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.

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 DS server fanout, the Delivery Service can be considered as an active adaptative 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 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 key material signed with the signature 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 keying material and to ensure that the identifier for these keys are unique across all keys stored on the DS.

2.3.2. Key Retrieval

When a client wishes to establish a group, it first contacts the DS to request a KeyPackage for each other client, authenticate it 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 DSs provide:

* Reliable delivery: when a message is provided to the DS, 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 DS needs some latitude in enforcing ordering across clients.

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

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

* The Group Identifier (GID) to allow to distinguish the group for which the message has been sent;

* The Epoch number, which represent the number of changes (version) of the group associated with a specific GID, and allows for lexicographical ordering of two messages from the same group;

* The Content Type of the message, which allows the DS to determine the ordering requirement on the message.

The MLS protocol itself can verify these properties. For instance, if the DS 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 faned-out to clients per Epoch, to provide Clients with a consistent view of the evolving Group State.

Note that some forms of DS misbehavior are still possible and difficult to detect. For instance, a DS 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 persisted 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 DS 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 DS.

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 hasn't processed messages but MLS-using systems can enforce some mechanism to try retaining these properties. Typically this will consist of evicting clients which are idle for too long, 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.

3. System Requirements

3.1. 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.

3.1.1. 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.

3.1.2. Recovery After State Loss

Conversation participants whose local MLS state is lost or corrupted can reinitialize their state and continue participating in the conversation.

[[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."]]

3.1.3. Support for Multiple Devices

It is typically expected for users within a Group to own different 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.

3.1.4. 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.

3.1.5. Privacy

The protocol is designed in a way that limits the server-side (AS and DS) metadata footprint. The DS only persists data required for the delivery of messages and avoids Personally Identifiable Information (PII) or other sensitive metadata wherever possible. A Service Provider that has control over both the AS and the DS, will not be able to correlate encrypted messages forwarded by the DS, with the initial public keys signed by the AS.

[[OPEN ISSUE: These privacy statements seem very strong. BB. I would be willing to keep them as requirements since we have example solutions in the Server-Assist draft.]]

3.1.6. 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 and infrastructure functionalities.

3.1.7. 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 proposed 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.

3.2. Security Requirements

3.2.1. Connections between Clients and Servers (one-to-one)

We assume that all transport connections are secured via some transport layer security mechanism such as TLS [RFC8446]. However, as noted above, the security of MLS will generally survive compromise of the transport layer, so long as identity keys provided by the AS are authenticated at a minimum. However, MLS ciphertext contains the Group Identifier, Epoch number and Content Type that may be used to improve attacks on the privacy of the group.

3.2.2. Message Secrecy and Authentication

The trust establishment step of the MLS protocol is followed by a conversation protection step where encryption is used by clients to transmit authenticated messages to other clients through the DS. This ensures that the DS does not have access to the group's private content.

MLS aims to provide secrecy, integrity and authentication for all messages.

Message Secrecy in the context of MLS means that only intended recipients (current group members), can read any message sent to the group, even in the context of an active attacker as described in the threat model.

Message Integrity and Authentication mean that an honest Client can only accept a message if it was sent by a group member and that no Client can send a message which other Clients accept as being from another Client.

A corollary to this statement is that the AS and the DS cannot read the content of messages sent between Members as they are not Members of the Group. MLS optionally provides additional protections regarding traffic analysis so as to reduce the ability of attackers, or a compromised member of the messaging system, to deduce the content of the messages depending on (for example) their size. One of these protections includes padding messages in order to produce ciphertexts of standard length. While this protection is highly recommended it is not mandatory as it can be costly in terms of performance for clients and the SP.

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

3.2.2.1. 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.

3.2.2.2. 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 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 other 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.

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.

3.2.2.3. 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 are maintained and not weakened by user membership in multiple groups.

3.2.2.4. 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.

3.2.2.5. Denial of Service

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 DS to perform a trivial DoS attack from which it is hard to recover.

3.2.2.6. Non-Repudiation vs Deniability

As described in Section 4.4, 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. [[OPEN ISSUE: Exactly how to supply this is still a protocol question.]]

4. Security 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. 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.

4.1. Transport Security Links

[TODO: Mostly DoS, message suppression, and leakage of group membership.]

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.

4.3. Authentication Service Compromise

A compromised AS is a serious matter, as the AS can provide incorrect or attacker-provided identities to clients. As noted in Section 2.2, detecting this form of attack requires some sort of transparency/ logging mechanism. Without such a mechanism, MLS cannot detect a compromised AS.

4.4. Client Compromise

MLS provides a limited form of protection against compromised Clients through PCS. When the Client is fully compromised, then the attacker will be able to decrypt any messages for groups in which the Client is a member, and will be able to send messages impersonating the compromised Client. However, if the Client afterwards updates its keying material (see Section 3.2.2.1) (using fresh randomness that the attacker does not know) then the PCS property enables the Client to recover.

In addition, a client cannot send a message to a group which appears to be from another client with a different identity. Note that if devices from the same user share keying material, then one will be able to impersonate another.

Finally, clients should not be able to perform DoS attacks Section 3.2.2.5.

5. IANA Considerations

This document makes no requests of IANA.

6. Contributors

*Katriel Cohn-Gordon

University of Oxford

me@katriel.co.uk

*Cas Cremers

University of Oxford

cas.cremers@cs.ox.ac.uk

*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

7. 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

Emad Omara
Google

Email: emadomara@google.com

Benjamin Beurdouche
INRIA

Email: benjamin.beurdouche@inria.fr

Eric Rescorla
Mozilla

Email: ekr@rtfm.com

Srinivas Inguva
Twitter

Email: singuva@twitter.com

Albert Kwon
MIT

Email: kwonal@mit.edu

Alan Duric
Wire

Email: alan@wire.com