OpenMLS

OSCW 2024

A guided tour

Messaging Layer Security: An Overview

Overview

- Group messaging and group key agreement protocol
- Asynchronous (requires pre-published key material)
- Dynamic groups (add/remove/update)
- Proposal/Commit paradigm
- Requires total order of (Commit) messages
- Abstract over KEMs and Signatures

Pre-published Key Material

- Clients pre-publish *KeyPackages*
- Allows asynchronous group additions
- KeyPackages advertise a client's capabilities (supported versions, ciphersuites, etc.)
- Contains the owner's *Credential* and Signature Public Key

MLS Message Format

- MLS messages are wrapped in the `MLSMessage` struct
- An `MLSMessage` can contain
  - PublicMessage (Commit or Proposal)
  - PrivateMessage (Commit, Proposal or ApplicationMessage)
  - Welcome
  - GroupInfo
  - KeyPackage
- Proposals and Commits can be sent encrypted or in the clear

MLS Group Operations

- Both group members and non-members (e.g. a server) can propose group operations (e.g. add/remove) via (cheap) *Proposals*
- Group members can enact a set of Proposals by creating a (more expensive) *Commit*
- Each Commit introduces a new *Epoch*

Motivation for Proposal/Commit paradigm:

- Allows involvement of non-members
- Computationally weaker group members can avoid more expensive commit operations

Agreement on order of Commits

- The (abstract) *Delivery Service* (DS) needs to order commit messages
- A DS can be *strongly consistent* or *eventually consistent*
  - Strongly consistent: E.g. a central server choosing a commit per epoch. Prevents forks.
  - Eventually consistent: E.g. a distributed system, where clients agree on message order via a separate algorithm
  - MLS can be made more fork-resilient (an extension is under way)

Adding and Joining Group Members

- **Invitation flow:** Group Members can be added asynchronously (via KeyPackages)
- **Join flow:** Group Members can join a group instantly (if they have the required GroupInfo)[1]
  - Applications can disallow such instant joins

[1]: Also called “external commit”, “external init” or “external join”

(TLS-like) Exporter and Pre-shared keys

- An MLS group can export key material on a per-epoch basis
- Exporters can be used in other applications/protocols
- Key material can be injected into the group
  - e.g. for channel binding, authentication purposes or improved entropy

Group State Agreement

- MLS groups can be used to ensure that members agree on arbitrary data
- Custom Proposals can be used to coordinate non-MLS operations
- Strict message ordering helps solve general problems with distributed state (group-wide mutex)

Questions?

OpenMLS

General State

- Fully implements RFC 9420
- Version 0.5 on crates.io
- State serialization currently under development
- Working on stabilizing public API
- A number of pending changes on `main`

Pluggable Providers via Traits

- RNG
- Crypto provider
  - Covers cryptographic operations
  - Ciphersuite support depends on provider
  - Currently available: libcrux, RustCrypto
- Storage provider
  - Can store various OpenMLS data types
  - Group state
  - KeyPackages
  - Private key material

Type-based Verification

- Input types with -In suffix (e.g. `MlsMessageIn`)
- Suffix removed after verification
- Output types with -Out suffix (e.g. `MlsMessageOut`)
- Only input types can be de-serialized
- Enforces verification within OpenMLS
- This can lead to duplicated code (we’re working on it)

Main API: `MlsGroup`

- Represents an MLS group
- Can be serialized and store through storage provider
- Contains a Proposal store
- Follows a stage-and-merge flow for commits
  - Commits are first staged (by processing them)
  - After inspection, staged commits can be merged
- Outputs `MLSMessage` structs (can be serialized)

Playground

- Repo with playground app: https://github.com/openmls/oscw24
- Functional DS at https://ds.openmls.tech
- Let us know if you have any questions or run into problems
- Issues and PRs welcome!