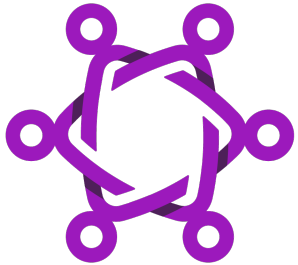


Rohan’s MLS drafts @ IETF120

Rohan Mahy: rohan.ietf@gmail.com

Summary

Safe Extensions framework (in draft-ietf-mls-extensions):

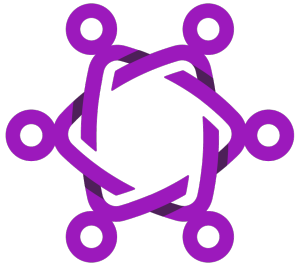
- Clarify how Safe Extension extensions types are registered with IANA for Proposals, Credentials, and Wire Formats and where these tokens are placed in MLS (ex: GroupContext, LeafNode Capabilities) - 10 minutes
- Add support for SafeAAD - 5 minutes

Extensions related to MIMI:

- draft-barnes-mls-appsync (me or Richard) - 15 minutes. (Needs RequiredAppIDs in GC and SupportedAppIDs in LN)
- draft-mahy-mls-semiprivatemessage - 10 minutes
- draft-mahy-mls-member-proof - abandoning
- draft-mahy-mls-kp-context - no time requested
- draft-mahy-mls-room-policy-ext - no time requested -> would become a safe extension

Extensions not related to MIMI: - 2 minute status update

- SelfRemove (in draft-ietf-mls-extensions)
- draft-mahy-mls-xwing - waiting on CFRG KEM combiner design team for likely replacement
- draft-mahy-mls-ratchet-tree-options - useful way to talk about changes to ratchet trees and GroupInfo



Safe Extensions

Section 2 of draft-ietf-mls-extensions

SafeExtensions is Great! What's missing?

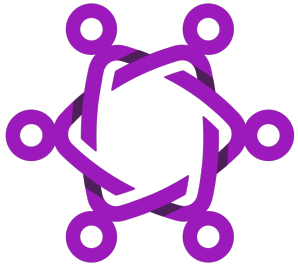
What is missing from the SafeExtensions:

- Way to send AAD safely with multiple extensions (SafeAAD) PR#29

SafeExtensions is Great! What's missing?

What is missing from the SafeExtensions:

- Way to send AAD safely with multiple extensions (SafeAAD) PR#29
- Better explanation of how you register Proposals, Credentials, WireFormats (, and AAD) extensions. Issue #30
- Better explanation of how you use extensions in GroupContext, and in LeafNode Capabilities. Overloading of term Extension as used for SafeExtensions vs. LN, KP, GC, and GI extensions.
    - ex: If I define a safe Proposal, do I register an Extension and not a Proposal?
    - Do I put the proposal in the Proposals Capabilities? Using the same number?



Application State Sync

draft-barnes-mls-appsync

Group State Agreement is Powerful

MLS confirms that the group agrees on the state of the group

Applications frequently want all clients to agree on application state

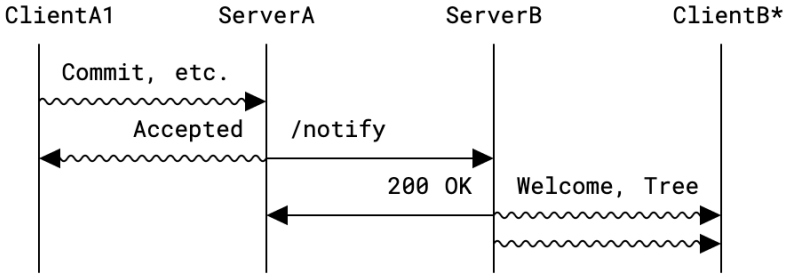
Let’s import application state into the state of the group!

Envisioned Use in MIMI

Participant list (roster) is included in the MLS key schedule via AppState extension

(Non-participant) Room policy could be a separate AppState *application*

Changes to the PList are communicated in AppSync proposals



The Commit includes regular Adds *and* an AppSync Proposal updates app state

```
ClientA1: Prepare Commit over AppSync(+Bob), Add*
ClientA1->ServerA: [[ Commit, Welcome, GroupInfo?, RatchetTree? ]]
ServerA: Verify that AppSync, Adds are allowed by policy
ServerA: Identifies Welcome domains based on KP hash in Welcome
ServerA->ServerB: POST /notify/a.example/r/clubhouse Intro{ Welcome, RatchetTree? }
ServerB: Recognizes that Welcome is adding Bob to room clubhouse
ServerB->ClientB*: [[ Welcome, RatchetTree? ]]
```

AppSync Extension + Proposal

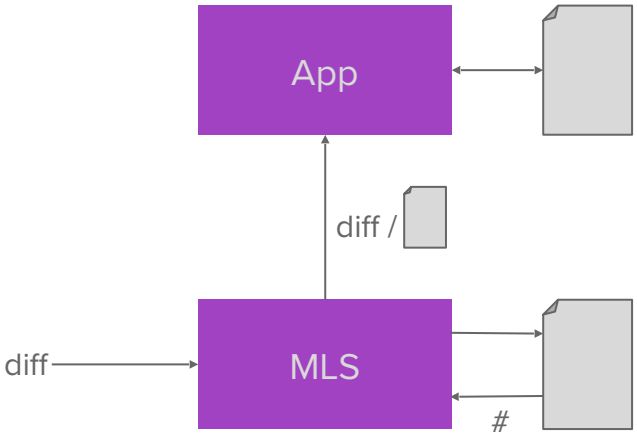
```
struct {
  uint32 applicationId;
  StateType stateType;
  select (stateType) {
    case irreducible:
      OpaqueElement state;
    case map:
      OpaqueMapElement mapEntries<V>;
    case unorderedList:
      OpaqueElement unorderedEntries<V>;
    case orderedArray:
      OpaqueElement orderedEntries<V>;
  };
} ApplicationState;
```

GroupContext Extension

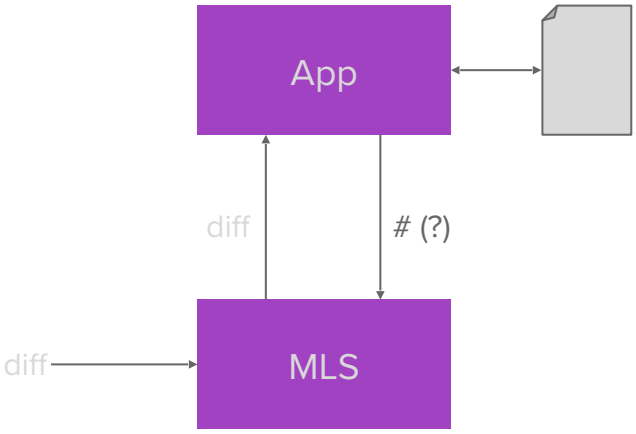
```
struct {
  uint32 applicationId;
  StateType stateType;
  select (stateType) {
    case irreducible:
      OpaqueElement newState;
    case map:
      OpaqueElement removedKeys<V>;
      OpaqueMapElement newOrUpdatedElements<V>;
    case unorderedList:
      uint32 removedIndices<V>;
      OpaqueElement addedEntries<V>;
    case orderedArray:
      ElementWithIndex replacedElements<V>;
      uint32 removedIndices<V>;
      ElementWithIndex insertedElements<V>;
      OpaqueElement appenededEntries<V>;
  };
} AppSync;
```

Proposal

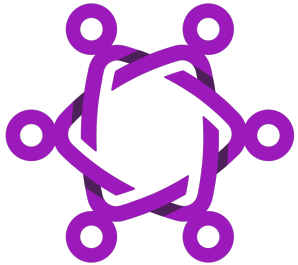
APIs and Design Choices



State intelligible to MLS (current)



State opaque to MLS almost PR #1



SemiPrivateMessage

draft-mahy-mls-semiprivatemessage

Why do we need a SemiPrivateMessage?

- Like we have valid `external_senders`, have a list of `external_receivers` who get handshake messages.
- Could be used in MIMI to share handshakes with the Hub, without other providers seeing group metadata.
- Safer than sharing the encrypted PrivateMessage content in a roll-your-own manner as done in MIMI PR#79 (keys/crypto stays in the MLS stack)

How does it work?

- SafeExtension (wire format) that builds off PrivateMessage (adds one element)
- PrivateMessage today is encrypted with 2 keys and 2 nonces (1 pair for `sender_data` and 1 pair for `ciphertext`).
- Encrypt those keys and nonces for each `external_receiver`
- Include those `keys_for_external_receivers` in message structure

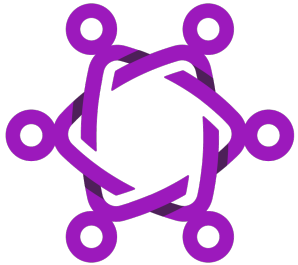
```
struct {
    opaque group_id<V>;
    uint64 epoch;
    ContentType content_type;
    opaque authenticated_data<V>;
    KeyForExternalReceiver keys_for_external_receivers<V>;
    opaque encrypted_sender_data<V>;
    opaque ciphertext<V>;
} SemiPrivateMessage;
```

```
struct {
    opaque group_id<V>;
    uint64 epoch;
    ContentType content_type;
    opaque authenticated_data<V>;
    KeyForExternalReceiver keys_for_external_receivers<V>;
} SemiPrivateContentAAD;
```

```
struct {
  opaque sender_data_key<V>;
  opaque sender_data_nonce<V>;
  opaque key<V>;
  opaque nonce<V>;
} PerMessageKeysAndNonces;

PerMessageKeysAndNonces key_and_nonce;

encrypted_key_and_nonce = EncryptWithLabel(
  external_receiver_pulic_key,
  "SemiPrivateMessageReceiver",
  private_message, key_and_nonce)
```



Other stuff

Summary

Safe Extensions framework (in draft-ietf-mls-extensions):

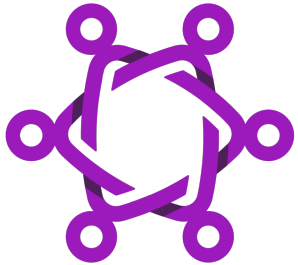
- Clarify how Safe Extension extensions types are registered with IANA for Proposals, Credentials, and Wire Formats and where these tokens are placed in MLS (ex: GroupContext, LeafNode Capabilities) - 10 minutes
- Add support for SafeAAD - 5 minutes

Extensions related to MIMI:

- draft-barnes-mls-appsync (me or Richard) - 15 minutes. (Needs RequiredAppIDs in GC and SupportedAppIDs in LN)
- draft-mahy-mls-semiprivatemessage - 10 minutes
- draft-mahy-mls-member-proof - abandoning
- draft-mahy-mls-kp-context - no time requested
- draft-mahy-mls-room-policy-ext - no time requested -> would become a safe extension

Extensions not related to MIMI: - 2 minute status update

- SelfRemove (in draft-ietf-mls-extensions)
- draft-mahy-mls-xwing - waiting on CFRG KEM combiner design team for likely replacement
- draft-mahy-mls-ratchet-tree-options - useful way to talk about changes to ratchet trees and GroupInfo



Backup slides

Extensions for MIMI

- MIMI is currently planning to have room state shared as GroupContextExtension to get **group agreement**.
- Richard Barnes proposed a generic application state extension that enables this.
    - MIMI would use this for
        - the **room policy document** (ex: This room is a members-only room. You have to have the “admin” role to add and remove users); and
        - the **participation list** which contains users and their roles (ex: Alice is an admin, Bob is a regular participant)
    - The participation list is updated (patched?) via a new proposal type which does not require an *UpdatePath*
    - The actual policy will likely be defined in MIMI.
- What else do we need in MLS?
    - KeyPackage Context?
    - Pending Proposals in External Commits?
    - Improvements to Conveying ratchet tree?

draft-mahy-mls-kp-context

- KeyPackage extension which restricts the use of the KeyPackage to a specific context
    - Only use this KeyPackage to join a specific MLS group
    - This KeyPackage is only meant to be used if the Adder has the following “user” identity
    - This KeyPackage is only meant to be used if the Adder is in the following domain
    - This KeyPackage is only meant to be used if the Adder has the following public key
- Might add
    - Only use this KeyPackage to join a specific “room”
- Any other contexts we might be missing?
- Next steps? Add to extensions draft?
- Could be incorporated into **AppState general solution**

SelfRemove proposal in MLS Extensions

- Open Issue: User still cannot ensure that removing oneself is atomic
  - Option 1: User's client sends a commit with Remove proposals for other clients, then sends a SelfRemove Proposal. Takes 2 epochs to remove, with one client present
  - Option 2: User's client sends Remove proposals for other clients and SelfRemove proposal at the same time. External Commit still is obliged to ignore the Remove proposals.
- Solutions:
  - Option A: Add list of other client indexes to delete (must be clients of the same user) to the SelfRemove
  - Option B: Change the behavior of External Commit in presence of SelfRemove to commit valid Remove proposals. Makes the extension no longer a safe extension?
  - **Option C**: Generically support sending all valid pending proposals in an external commit.

Include Pending Proposals in External Commits

- Currently external commits don’t include otherwise valid pending proposals

- Extension to require external commits to include all valid pending proposals (from wherever new joiner got the GroupInfo)
    - DS is responsible to provide pending proposals too if it provides GroupInfo
    - Clients need to accept External Commits which include the pending proposals by reference
    - Clients sending External Commit need to fetch and include valid pending proposals
- Not a safe extension. Its an ordinary GroupContext extension.

Why?

- Solves consistency problem

Options sending Ratchet Tree and GroupInfo

- Conveying the ratchet tree
    - RFC9420 only describes how to convey entire ratchet tree as an extension in Welcome and/or GroupInfo.
    - Already describes that ratchet tree for Welcome can be out-of-band but not *how*.
        - Some MLS DS's reconstruct the ratchet tree from Commits/Proposals
        - Could be provided via an HTTPS URL or stapled to an MLS message
- Conveying the GroupInfo
    - Client needs to provide at least the GroupInfo signature and any GroupInfo extensions (external_pub). Otherwise DS can reconstruct a GroupInfo.
    - Provide either full GroupInfo or a PartialGroupInfo (signature + GroupInfo ext)

~~draft-mahy-mls-x25519kyber768draft00~~ →
draft-mahy-mls-xwing-00

- Desire for handful of complimentary post-quantum (PQ) security extensions for MLS:
    - Straightforward MLS cipher suite: replace classical KEM with a hybrid PQ/traditional KEM. Drop-in replacement in many MLS libraries without changes to any other part of the MLS stack. Single KEM which is performant and works for the vast majority of implementations. Address harvest-now / decrypt-later using the simplest, most practicable solution available. ⬅ **You Are Here**
    - Versions of existing cipher suites that use PQ signatures; and specific guidelines on the construction, use, and validation of hybrid signatures.
    - One or more mechanisms which reduce bandwidth and/or storage requirements; or improve performance (ex: by updating post-quantum keys less frequently than classical keys, or by sharing portions of PQ keys across a large number of clients or groups.)
- NIST announced the ML-KEM standard based on Kyber. Need to use slightly more complicated combiner for ML-KEM vs plain concatenation with Kyber.
- Poll for adoption?