Async Add

Current problem

The architecture document says:

“No operation in MLS requires two distinct clients or members to be online simultaneously.”

This is true strictly speaking, but the reality looks different

How do we add today?

Handshake message	From member	From external
AddProposal	✓	✓
Commit	✓	✗

Dependency: even if all members agree the AddProposal is valid, the new member has to wait for another member to come online to be able to send messages.

Not really a new problem

The problem has been known for a while and has been partially alleviated by the proposal-commit scheme.

The consensus was that if the new member really needs to send to the group right away, we could use mechanisms like “send from outside”, or some other proprietary mechanism.

What is new?

“Send from outside” is not part of the protocol and therefore vendors would have to come up with a proprietary concept.

Moreover, such a mechanism only works if exactly **one** new member wants join. If another new member wants to join, they might both be able to send to the existing group, but would need an additional mechanisms to also communicate between each other.

Worst case scenario: None of the existing members ever comes online. That leaves the new members without a functioning group (legacy group with dangling AddProposals).

What can we do?

There are two options:

- The WG deems this is still only an edge case and vendors should solve this at the application layer

- The WG looks at proposals to improve things

Reminder: Access control

How does validation of new members work exactly?

- There is a notion of a policy that clients enforce
- All clients MUST have the same policy in order to prevent forks of a group
- Clients MUST reject handshake messages that are semantically invalid w.r.t. the policy

Reminder: Validation

- A member or an external party creates an AddProposal and sends it to the group. The AddProposal SHOULD be valid.
- A member MUST check the validity of the AddProposal before including it in a Commit.
- Other members MUST also check the validity before applying the Commit or otherwise reject the Commit (and subsequent Commits in future epochs).
- In case members rejected a Commit, they MUST wait for a valid Commit for the same epoch instead or issue a Commit themselves

Solution proposal

External Commit

Future members can create a Commit message and send it to the group, thus inserting themselves in the tree.

External Commit

External Commits work like regular Commits, with a few differences:

- External Commits MUST reference the Add Proposal that adds the issuing new member to the group
- External Commits MUST contain a \`path\` field (and is therefore a "full" Commit)
- External Commits MUST be signed by the new member
- When processing a Commit, both existing and new members MUST use an empty init secret (all-zero vector of size Hash.length).

Steps

1. An AddProposal is created by either:
    a. A member
    b. An external party
    c. A future member

Steps

2. The future member has access to the following information:
    a. The group ID
    b. The current epoch ID
    c. The ciphersuite
    d. The public tree
    e. The confirmed transcript hash
    f. The group extensions

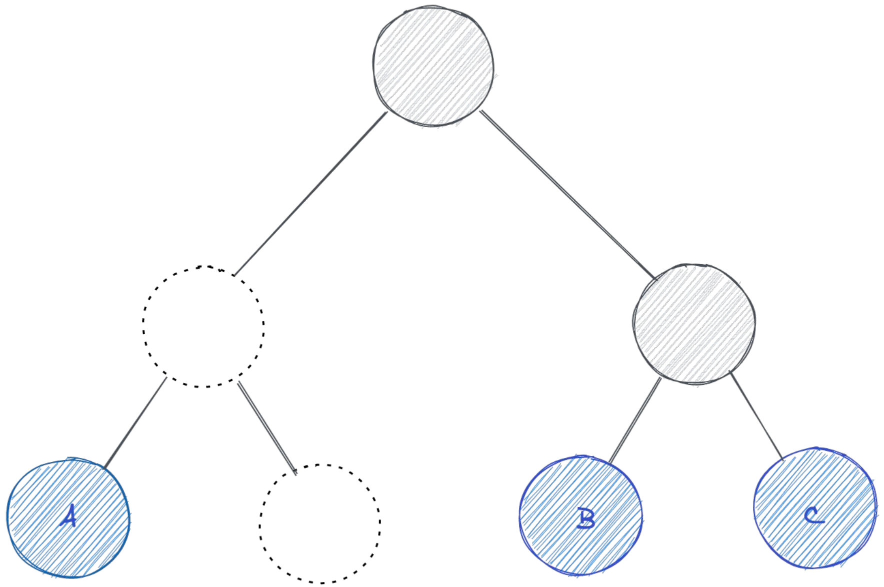
Steps

3. The future member creates a regular “full” Commit by
    a. Creating a KeyPackage (with a ParentHash extension)
    b. Inserting the KeyPackage in the left-most free leaf
    c. Calculating an UpdatePath object
    d. KEMing the secret values to its copath
    e. Using an empty init secret to advance the key schedule
    f. Calculating the confirmation tag

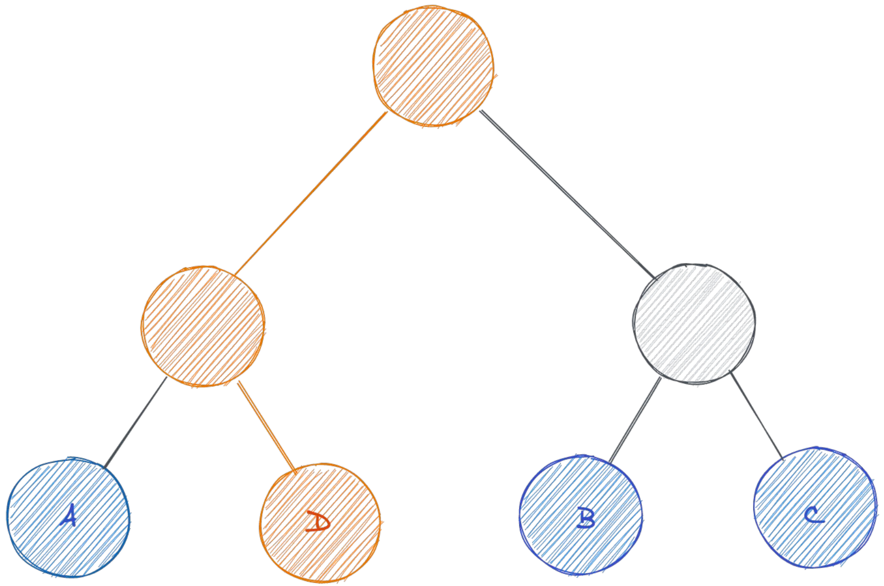
Steps

4. The future member applies the Commit with an empty init secret
5. Existing members apply the Commit with an empty init secret

Sample tree before External Commit



Sample tree after External Commit



Welcome Key Schedule

$$\begin{tikzcd}
 & & & \text{PSK} \arrow[d] & \text{GroupContext[n]} \arrow[d] & \\
\text{New member} & & \text{joiner\_secret} \arrow[r] & \text{member\_secret} \arrow[r] & \text{epoch\_secret} \arrow[r] & \text{init\_secret\_[n]} \\
 & & & \text{PSK} \arrow[d] & \text{GroupContext[n]} \arrow[d] & \\
\text{Existing member} & \text{init\_secret\_[n-1]} \arrow[r] & \text{joiner\_secret} \arrow[r] \arrow[uu, "\text{Welcome}"'] & \text{member\_secret} \arrow[r] & \text{epoch\_secret} \arrow[r] & \text{init\_secret\_[n]} \\
 & & \text{commit\_secret} \arrow[u] & & &
\end{tikzcd}$$

External Commit Key Schedule

$$\begin{tikzcd}
 & & & \text{PSK} \arrow[d] & \text{GroupContext[n]} \arrow[d] & \\
\text{New member} & 0 \arrow[r] & \text{joiner\_secret} \arrow[r] & \text{member\_secret} \arrow[r] & \text{epoch\_secret} \arrow[r] & \text{init\_secret\_[n]} \\
 & & \text{External Commit (commit\_secret)} \arrow[u] \arrow[dd] & & & \\
 & & & \text{PSK} \arrow[d] & \text{GroupContext[n]} \arrow[d] & \\
\text{Existing member} & 0 \arrow[r] & \text{joiner\_secret} \arrow[r] & \text{member\_secret} \arrow[r] & \text{epoch\_secret} \arrow[r] & \text{init\_secret\_[n]} \\
 & \cancel{\text{init\_secret\_[n-1]}} & & & &
\end{tikzcd}$$

Validation of External Commit

- A member or an external party creates an AddProposal and sends it to the group. The AddProposal SHOULD be valid.
- The new member issues an External Commit which references the AddProposal
- Existing members MUST check the validity of both the AddProposal and the External Commit before applying the Commit or otherwise reject the Commit (and subsequent Commits in future epochs).
- In case members rejected a Commit, they MUST wait for a valid Commit for the same epoch instead or issue a Commit themselves

Group agreement

From the PoV of an existing member

The group agreement for existing members doesn’t change.

Since the new member enters the group with a Commit, this gives strong guarantees that the new member agrees with the current group state.

Group agreement

From the PoV of the new member

The new member has similar guarantees about the group agreement compared to the situation where it was added through the usual Welcome mechanism.

Existing members need to do Updates in order to increase guarantees for the new member.

Entropy

The overall entropy of the group potentially decreases with an External Commit, since the init secret from the previous epoch is not re-used. It is comparable to the situation where a new group is created, and entropy increases over time through (internal) Commits.

Improving entropy

- More updates from existing members
- Re-inject old init secret via PSK
- Re-use old node secrets (see email from Joel Alwen on MLS ML about hardening randomness)
- More?

Pros

- Makes MLS much more suitable for applications where onboarding new clients is done without the help of existing ones
- High security guarantees are still in place (Confidentiality, Authentication, FS & PCS)
- Caters for scenarios where the AS can still provide strong authentication (e.g. through HSMs)

Cons

- Entropy temporarily decreases

Open questions

- Optional? Right now it is de facto optional since it requires the public state to be accessible to the new member. In high security settings External Commits can be forbidden.
- Can External Commits reference more than just one AddProposal? If so, what are the implications? If not, what do we do with pending proposals?
- Should we derive something from the Key Schedule to use as a public init secret for External Commits?