The road to RFC

draft-ietf-mls-protocol
RECENT WORK
Since draft-09...

8 virtual interims

32 pull requests merged

6 new contributors

@arianvp
@chelseakomlo
@d1vyank
@ericcornelissen
@tomtau
@uhoreg
# PRs since draft-09

- #308 - Remove nonce from SenderData AAD.
- #317 - Change expiration extension to lifetime extension.
- #318 - Fix markdown formatting issue for Ciphersuite section
- #319 - Use correct type for uint32.
- #321 - Extensions -> Extension
- #322 - Minor fix
- #329 - Rename messaging service to service provider
- #330 - Minor fixes
- #331 - Make ratcheting optional for Adds
- #334 - Explicitly state the order in which proposals are applied when creating a commit
- #335 - Fix HPKE setup function name
- #338 - Rely More on HPKE
- #339 - Upper bound on group size in early phase too low
- #341 - Fix in lifetime extension
- #342 - Allow external proposals to be signed.
- #343 - Upper bound for Commit

- #348 - Make the tree in the Welcome optional
- #350 - IANA updates and their consequences
- #352 - Use node_index for both hashes
- #353 - Explain the meaning of a Commit with no proposals
- #354 - misc little fixes
- #355 - Validate external proposals from preconfigured senders
- #356 - Minor editorial changes
- #357 - Fix all compiler warnings.
- #358 - Fix build by switching to GitHub actions
- #359 - Fix bugs in tree math and cleanup docs.
- #361 - Use correct arguments to Derive-Secret
- #363 - Fix compile errors again.
- #364 - Use the KDF from HPKE
- #370 - Minor extension fixes
- #371 - Define HPKE on first use
- #372 - Commit Generation Clarifications
Relying more on HPKE

HPKE started off as just a base encrypt-to-public-key mechanism

It has grown to cover most of the primitives we need:

   KDF, AEAD, Derive-Key-Pair (Signatures still from TLS)

Less spec text

Better agility
Make ratcheting optional for Adds

“Proposal/Commit will make Adds $O(\log N)$ instead of $O(1)$, but if that’s an issue, we can always special-case Add-only Commits.”
-- R. Barnes (probably), circa Nov. 2019

It’s an issue: In large, infrequently-updating groups, its $O(N)$ … so we added special case logic for it

No PCS on Add-only commit, only FS w.r.t. new members (PCS iff path)

<table>
<thead>
<tr>
<th></th>
<th>1975</th>
<th>struct {</th>
<th>1980</th>
<th>struct {</th>
<th>1985</th>
<th>+ optional&lt;DirectPath&gt; path;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1976</td>
<td>ProposalID updates&lt;0..2^16-1&gt;;</td>
<td>1981</td>
<td>ProposalID updates&lt;0..2^16-1&gt;;</td>
<td>1986</td>
<td>} Commit;</td>
</tr>
<tr>
<td></td>
<td>1977</td>
<td>ProposalID removes&lt;0..2^16-1&gt;;</td>
<td>1982</td>
<td>ProposalID removes&lt;0..2^16-1&gt;;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1978</td>
<td>ProposalID adds&lt;0..2^16-1&gt;;</td>
<td>1983</td>
<td>ProposalID adds&lt;0..2^16-1&gt;;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1979</td>
<td></td>
<td>1984</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1980</td>
<td>- KeyPackage key_package;</td>
<td>1985</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1981</td>
<td>- DirectPath path;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1982</td>
<td>} Commit;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Make the tree optional in GroupInfo

New joiners to the group need to know the tree

But the tree is (a) big to upload and (b) cacheable; send a \textbf{commitment} instead

Joiner needs to get the tree \textbf{before processing the Welcome}
The Road to RFC
Pace of major changes has slowed

Time to start wrapping up...
Protocol Changes
Non-protocol Fixes

Working Group Last Call

Formal Verification
IETF Last Call
IESG Submission
AD Review
IESG Approval
RFC Editor Queue
RFC

draft-10, ETA Aug.

How long?
Repeat as necessary
REMAINING ISSUES + PRS
Confirmed Protocol Issues (binned, [PRs])

- Update the key schedule to reflect reality [#362, #336]
  - #325 - Simplify epoch secret derivation?
  - #326 - Authenticate that added members know the PSK
- #302 - Use masking instead of AES-GCM for sender data [#360]
- Make MLSCiphertext fully opaque [#349]
  - #142 - Prevent suppression of Handshake messages
  - #269 - Randomize values in the common framing header
- PSKs, session resumption, and authentication
  - #366 - Add extensions to the Commit message [#369]
  - #367 - Negotiate PSKs
  - #368 - Proof of prior membership in the group / Resumption
  - #374 - Derive an "authentication secret"
Uncertain and Non-Protocol Issues

- #160 - Advertize a global app generation for a sender
- #373 - Address DoS by malicious insiders
- Post-protocol-completion editorial review
  - #365 - Update security considerations
  - #273 - Editorial: structure of the document
  - #168 - Clarify obligation of clients to Update

... anything else?
Reflecting reality in the key schedule

Current key schedule has a few problems:

1. When a PSK is used, it doesn’t authenticate that new joiners know it
2. The GroupContext gets used in a bunch of individual derivations

Proposed solutions:

1. Reorder so that the joiner has to use the PSK to get the epoch secret
2. Add the GroupContext once, into the epoch_secret
init_secret_[n-1] (or Ø)

PSK (or Ø) \rightarrow HKDF-Extract = early_secret

Derive-Secret(. , "derived", """)

commit_secret \rightarrow HKDF-Extract = epoch_secret

commit_secret \rightarrow HKDF-Extract = joiner_secret

PSK (or Ø) \rightarrow HKDF-Extract = member_secret

GroupContext_[n] \rightarrow HKDF-Extract = epoch_secret
Simplifying sender data encryption

Goal: Prevent DS from seeing sender and generation

First attempt: “Masking” à la QUIC

sample ciphertext => KDF => XOR

Concerns about lack of authn

Second attempt: Sample AEAD nonce from ciphertext

Saves explicit sender_data_nonce, still AEAD
The diagram outlines the process of securing sender data using AEAD (Authenticated Encryption with Associated Data). The steps are as follows:

1. **Group ID** and **Epoch** are identified as input parameters.
2. **Sender Data Nonce** is processed with **AAD** and **nonce** to obtain **sender_data_key**.
3. **sender_data_key** is then used with **plaintext** to produce **encrypted_sender_data**.
4. The **encrypted_sender_data** is further processed with another **application_[i]_[j]_key** and **application_[i]_[j]_nonce** to secure the data further.
5. The final output includes **encrypted_sender_data** and **content**.
Swap order of content, metadata encryption

Sample sender data nonce from content ciphertext
Simplifying sender data encryption

Benefit: No explicit nonce

Nothing for adversary to tamper with

No need for more entropy

Cost: Sampling from ciphertext?

Should effectively be a random nonce …?

Proposal: Do ~this or do nothing
Make MLSCiphertext fully opaque

MLSCiphertext still exposes **group ID**, **epoch**, and **content type**

Proposal: Render these opaque to the DS

\[(\text{group ID}, \text{epoch}) \rightarrow \text{HKDF(\text{epoch}_\text{secret}, \text{"epoch ID"}, \text{epoch}_\text{id}_\text{len})}\]

- content_type moves inside encrypted content

**Pro:** Reveals minimum necessary information by default

**Con:** Adversarial collisions can cause partial DoS
PSKs, Session Resumption, and Authentication

Britta and Konrad proposed a bunch of changes in #336, addressing a few different use cases, including:

- Authentication that a member was part of the group in the past
- Verifying OOB that two members have the same view of the group

Proposal: splitting these out into more incremental chunks:

- Adding extensions to Commit
- Enabling negotiation of PSKs
- “Resumption” via PSKs generated off of the key schedule
- Deriving “authentication secret” from the epoch secret
FIN