The road to RFC

draft-ietf-mls-protocol

Recent Work

Since draft-09...

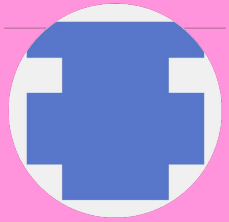
8 virtual interims

32 pull requests merged

6 new contributors

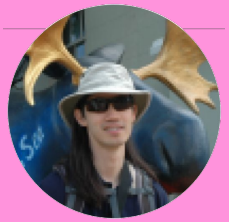

@arianvp


@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

```
+ These ciphersuites map to HPKE primitives and TLS signature schemes as follows
+ {{I-D.irtf-cfrg-hpke}} {{RFC8446}}:
+
+ | Value  | KEM    | KDF    | AEAD   | Signature              |
+ |:-------|:-------|:-------|:-------|:-----------------------|
+ | 0x0001 | 0x0020 | 0x0001 | 0x0001 | ed25519                |
+ | 0x0002 | 0x0010 | 0x0001 | 0x0001 | ecdsa_secp256r1_sha256 |
+ | 0x0003 | 0x0020 | 0x0001 | 0x0003 | ed25519                |
+ | 0x0004 | 0x0021 | 0x0003 | 0x0002 | ed448                  |
+ | 0x0005 | 0x0012 | 0x0003 | 0x0002 | ecdsa_secp521r1_sha512 |
+ | 0x0006 | 0x0021 | 0x0003 | 0x0003 | ed448                  |
```

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)

```
1975     struct {
1976         ProposalID updates<0..2^16-1>;
1977         ProposalID removes<0..2^16-1>;
1978         ProposalID adds<0..2^16-1>;
1979
1980 -       KeyPackage key_package;
1981 -       DirectPath path;
1982     } Commit;
```

```
1980     struct {
1981         ProposalID updates<0..2^16-1>;
1982         ProposalID removes<0..2^16-1>;
1983         ProposalID adds<0..2^16-1>;
1984
1985 +       optional<DirectPath> path;
1986     } Commit;
```

Make the tree optional in GroupInfo

```
2105    struct {
2106      opaque group_id<0..255>;
2107      uint64 epoch;
2108  -   optional<Node> tree<1..2^32-1>;
2109      opaque confirmed_transcript_hash<0..255>;
2110      opaque interim_transcript_hash<0..255>;
2111      Extension extensions<0..2^16-1>;
2112      opaque confirmation<0..255>;
2113      uint32 signer_index;
2114      opaque signature<0..2^16-1>;
2115    } GroupInfo;
```

```
2091    struct {
2092      opaque group_id<0..255>;
2093      uint64 epoch;
2094  +   opaque tree_hash<0..255>;
2095      opaque confirmed_transcript_hash<0..255>;
2096      opaque interim_transcript_hash<0..255>;
2097      Extension extensions<0..2^16-1>;
2098      opaque confirmation<0..255>;
2099      uint32 signer_index;
2100      opaque signature<0..2^16-1>;
2101    } GroupInfo;
```

New joiners to the group need to know the tree

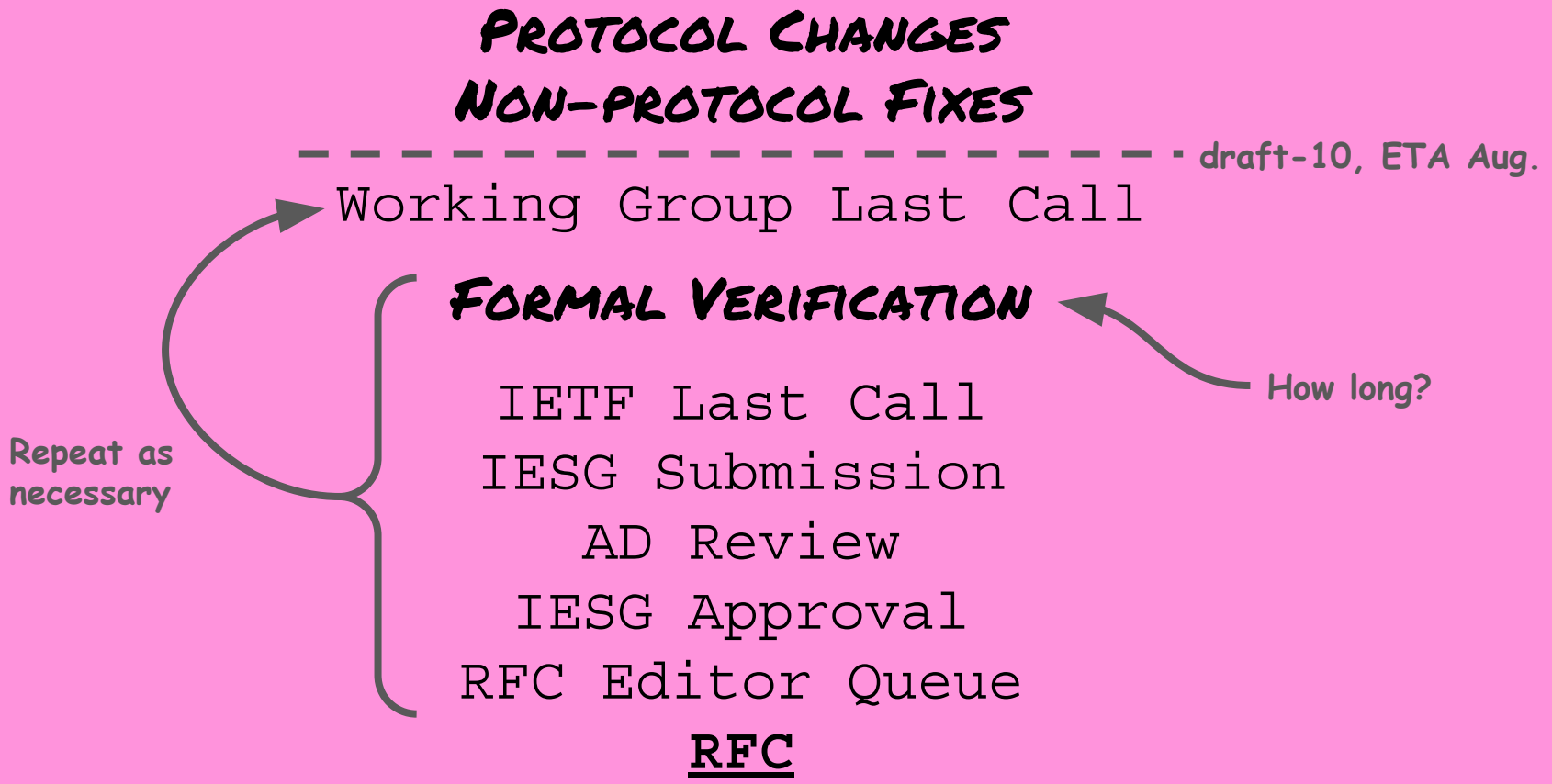
But the tree is (a) big to upload and (b) cacheable; send a **commitment** instead

Joiner needs to get the tree **before processing the Welcome**

The Road to RFC

Pace of major changes has slowed

Time to start wrapping up...


Protocol Changes
Non-protocol Fixes
draft-10, ETA Aug.
Working Group Last Call
Formal Verification
How long?
IETF Last Call
IESG Submission
AD Review
IESG Approval
RFC Editor Queue
RFC
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?

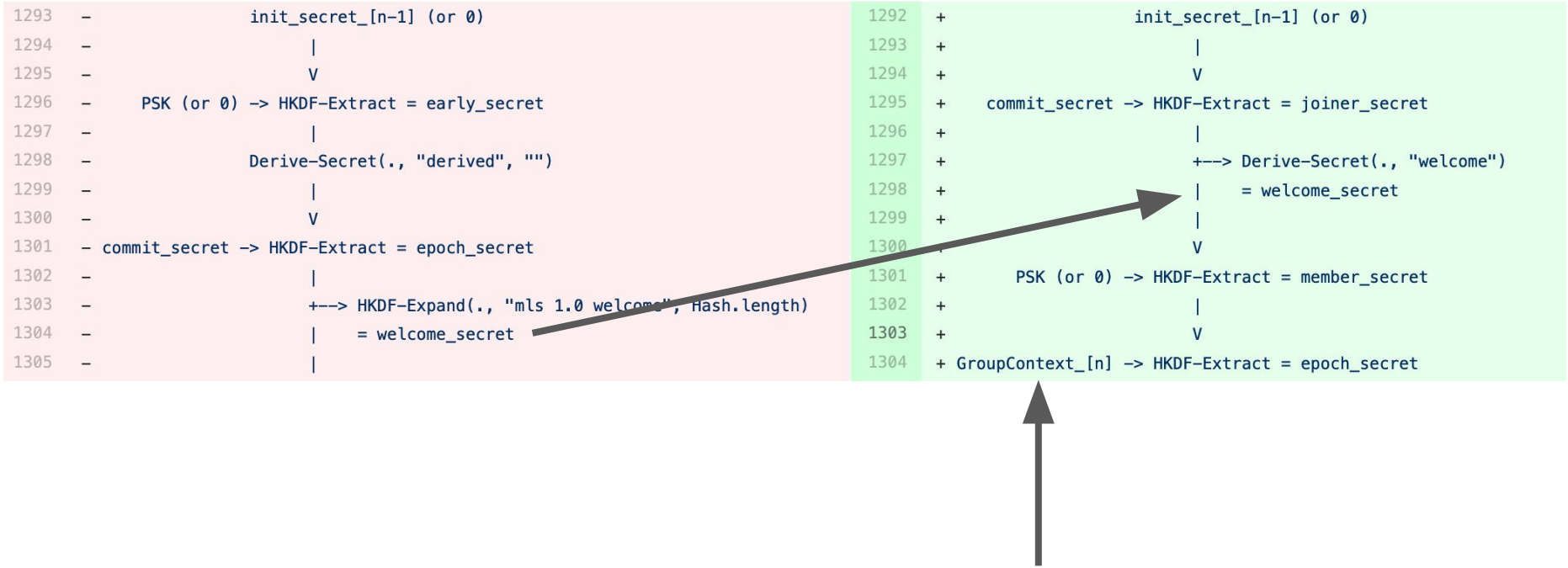
Reflecting reality in the key schedule

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


1293 - init_secret_[n-1] (or 0)
1294 - |
1295 - V
1296 - PSK (or 0) -> HKDF-Extract = early_secret
1297 - |
1298 - Derive-Secret(., "derived", "")
1299 - |
1300 - V
1301 - commit_secret -> HKDF-Extract = epoch_secret
1302 - |
1303 - +--> HKDF-Expand(., "mls 1.0 welcome", Hash.length)
1304 - | = welcome_secret
1305 - |
1292 + init_secret_[n-1] (or 0)
1293 + |
1294 + V
1295 + commit_secret -> HKDF-Extract = joiner_secret
1296 + |
1297 + +--> Derive-Secret(., "welcome")
1298 + | = welcome_secret
1299 + |
1300 + V
1301 + PSK (or 0) -> HKDF-Extract = member_secret
1302 + |
1303 + V
1304 + GroupContext_[n] -> HKDF-Extract = epoch_secret

Simplifying sender data encryption

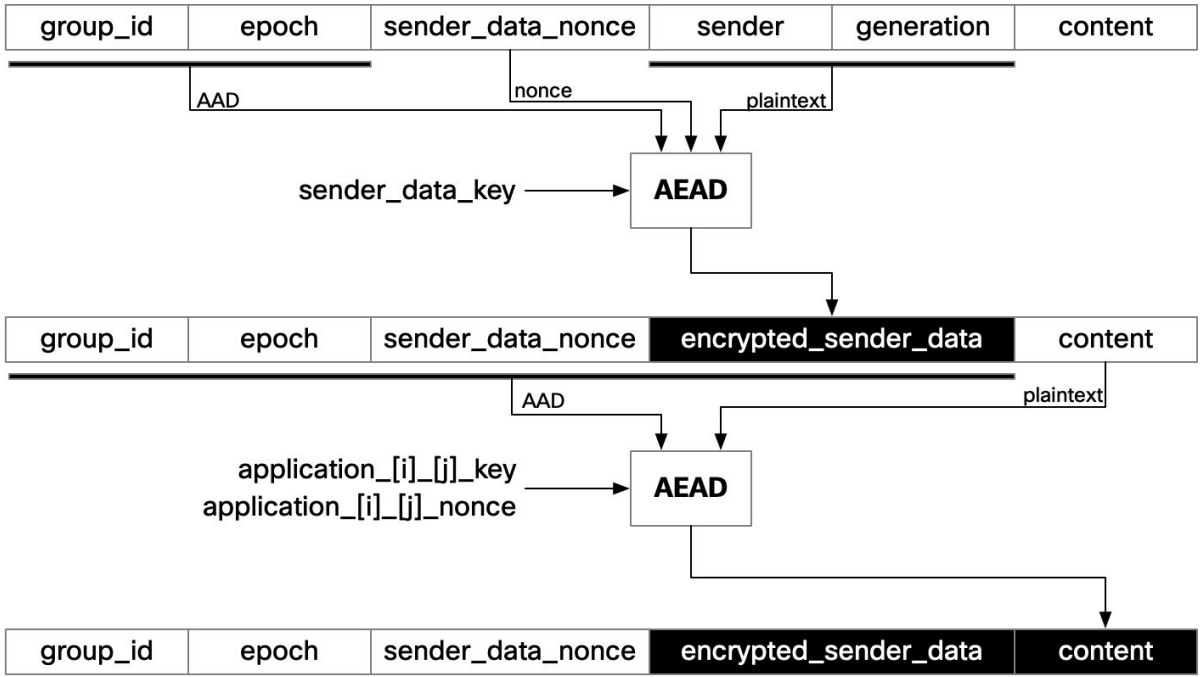
Goal: Prevent DS from seeing sender and generation

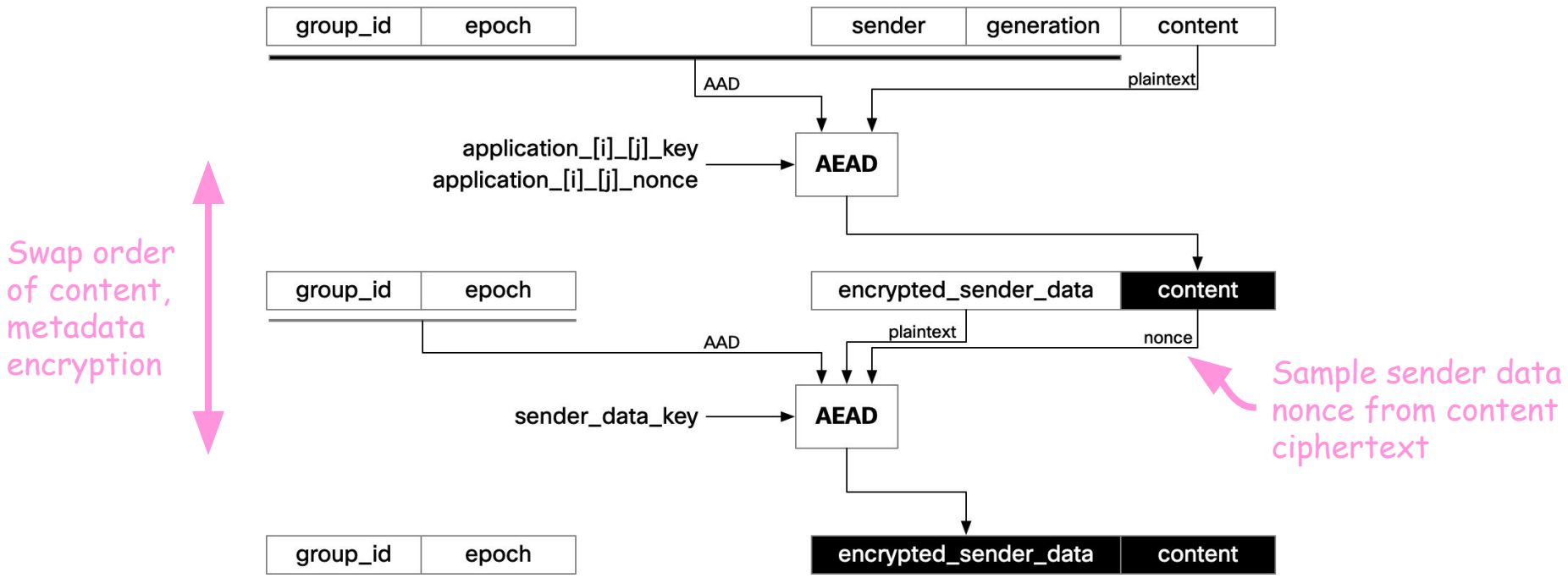
First attempt: "Masking" à la QUIC

- sample ciphertext => KDF => XOR
- Concerns about lack of authn

Second attempt: Sample AEAD nonce from ciphertext

- Saves explicit sender_data_nonce, still AEAD


group_id
epoch
sender_data_nonce
sender
generation
content
AAD
nonce
plaintext
sender_data_key
AEAD
group_id
epoch
sender_data_nonce
encrypted_sender_data
content
AAD
plaintext
application_[i]_[j]_key
application_[i]_[j]_nonce
AEAD
group_id
epoch
sender_data_nonce
encrypted_sender_data
content


group_id
epoch
sender
generation
content
AAD
plaintext
application_[i]_[j]_key
application_[i]_[j]_nonce
AEAD
Swap order of content, metadata encryption
group_id
epoch
encrypted_sender_data
content
AAD
plaintext
nonce
sender_data_key
AEAD
Sample sender data nonce from content ciphertext
group_id
epoch
encrypted_sender_data
content

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

> (group ID, epoch) -> HKDF(epoch_secret, "epoch ID", epoch_id_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