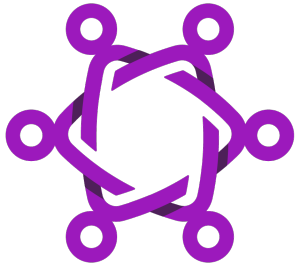


Old individual drafts

draft-mahy-mls-sd-cwt-credentials

draft-mahy-mls-ext-commit-pp

SD-CWTs and SD-JWTs as MLS Credentials

- This is relatively straightforward.
- Currently uses the SD-KBT for SD-CWT. Might be able be to use the SD-CWT directly instead in MLS.
- Works great with pseudonyms.

```
struct {
    CredentialType credential_type;
    select (Credential.credential_type) {
        case basic:
            opaque identity<V>;
        case x509:
            Certificate certificates<V>;
        ...
        case sd_cwt:
            opaque sd_kbt<V>;
        case sd_jwt:
            SdJwt sd_jwt;
    };
} Credential;
```

SD-JWT as Credential

- the canonical format of an SD-JWT looks like this:
- Can convey it "as is", or decomposed and unbase64ed to make it smaller
- Can convey a regular JWT


```
struct {
    Bool compacted;
    select (compacted) {
        case true:
            opaque protected<V>;
            opaque payload<V>;
            opaque signature<V>;
            SdJwtDisclosure disclosures<V>;
            opaque sd_jwt_key_binding<V>;
        case false:
            opaque sd_jwt_kb<V>;
    };
} SdJwt;
```

Encrypted disclosures in SD-CWT

- Allows a disclosure to be ephemerally AEAD encrypted
- In the MLS context, allows a disclosure to be hidden from DS but shared with all (or a subset) of members.

```
<<[
    /salt/   h'bae611067bb823486797da1ebbb52f83',
    /value/  "ABCD-123456",
    /claim/  501    / inspector_license_number /
]>>,
```

```
/ sd_aead_encrypted_claims / 19 : [ / AEAD encrypted disclosures /
    [
        / nonce /      h'95d0040fe650e5baf51c907c31be15dc',
        / ciphertext / h'208cda279ca86444681503830469b705
                         89654084156c9e65ca02f9ac40cd62b5
                         a2470d',
        / tag /        h'1c6e732977453ab2cacbfd578bd238c0'
    ],
    ...
]
```

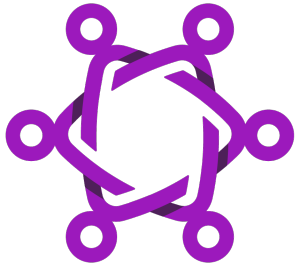
Include Pending Proposals in External Commits

- 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; makes some operations more atomic
- (ex: Removes and AppDataUpdate)



New individual drafts

draft-mahy-mls-private-external

draft-mahy-mls-new-framed-content-tbs

draft-mahy-mls-new-content-types

draft-mahy-private-external—Problem

- MLS deployments split into Public and Private handshakes
  - PublicMessage handshakes (smart DS, GroupContext/ratchet tree known by DS) — Cisco, Amazon, Apple, Google, Wire
  - PrivateMessage handshakes (DS only does ordering and fanout, GC/tree private to members) — XMTP, Germ, Mozilla
- External Proposals (and External Commits) are still PublicMessage. For example External Add proposal **still reveals LeafNode** of pending joiner to DS.

draft-mahy-private-external—Wrap External Proposals

- Defines new WireFormat: PrivateExternalMessage
- New Joiner HPKE Encrypts its PublicMessage containing an External Add Proposal to the Group
- The DS cannot see the KeyPackage or LeafNode inside
- Members can decrypt the PublicMessage, then authenticate and authorize the External Add Proposal

draft-mahy-private-external—Advertise HPKE Public Key for group

- Advertise ExternalEncryptionInfo (of ***future*** epoch) in AAD of each Commit, containing
  - CipherSuite
  - HPKE (encryption) Public Key for group
  - Signature Public Key for the group
- Members can verify the contents of the AAD
- Members or DS can share this safely with potential new joiners

draft-mahy-private-external—Preventing attacks

- Attacker could flood the DS with ExternalProposals that members need to decrypt in order to validate.
- Mitigate using the application protocol:
  - for example: require a privacy pass token or include an Argon 2di memory-hard hash to discourage abuse.
- To prevent active attacks, HMAC the authentication_secret with public keys

draft-mahy-mls-new-framed-content-tbs

- Making application messages (and member proposals) more efficient
- Current FramedContentTBD structure hashes the full GroupContext in every request
    - GroupContext only changes on transition to a new epoch
- New proposed FramedContentTBD structure includes the hash of the GroupContext
- Results in faster signing and verification
- New GroupContext extension indicates support (in LeafNode), and mandatory use (in GroupContext)
- We should have done this ages ago

draft-mahy-mls-new-content-types — Mechanism

- Currently MLS has three content types: application, proposal, and commit
- The secret tree creates ratchets for each sender for application messages and handshake messages
- Extend the secret tree with additional content types: status and ephemeral. These are treated like application messages except that
  - they use a different ratchet
  - the specific content type is specified in PrivateMessage
  - app can more aggressively fast forward through status and ephemeral rathets

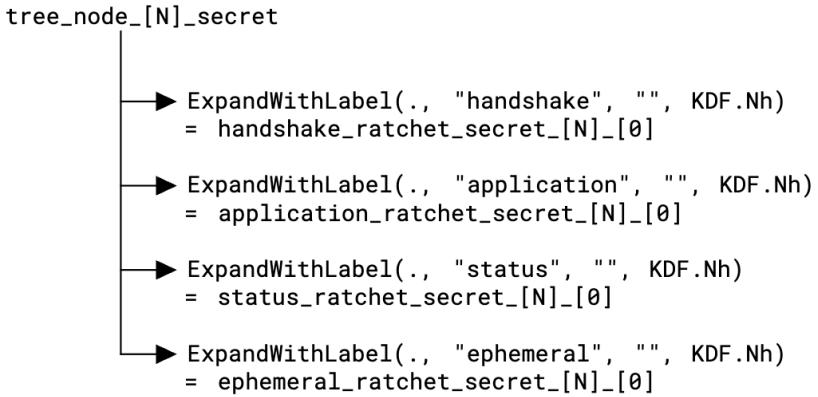
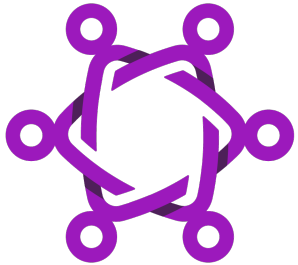


Figure 1: Initialization of the Hash Ratchets from the Leaves of a Secret Tree

draft-mahy-mls-new-content-types — Uses

- Status ratchet is a natural fit for (user) presence.
  - Only decrypt the last message from each member.
  - Can aggressively fast forward through the ratchet up to the max number of steps
- In the Ephemeral ratchet can completely discard offline messages. (ex: isTyping)



Drafts used by MIMI

Extensions used by MIMI*

- MLS Extensions (draft-ietf-mls-extensions)
  - Safe Extensions
    - Component IDs
    - AppDataUpdate
    - Safe AAD
    - required_wire_formats/supported_wire_formats
  - Content negotiation
  - Self Remove proposals
- Ratchet Tree Options (draft-mahy-mls-ratchet-tree-options)
- SemiPrivateMessage (draft-mahy-mls-semiprivatemessage)

* MLS credential formats for MIMI have not been decided

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)

SemiPrivateMessage (draft-mahy-mls-semiprivatemessage)

- Like we have valid `external_senders`, have a list of `external_receivers` who get handshake messages.
- Optionally 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(keys/crypto stays in the MLS stack)

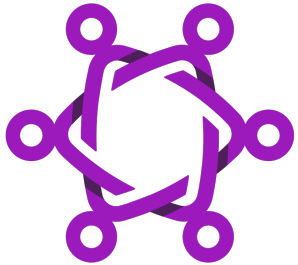
How does it work?

- New 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

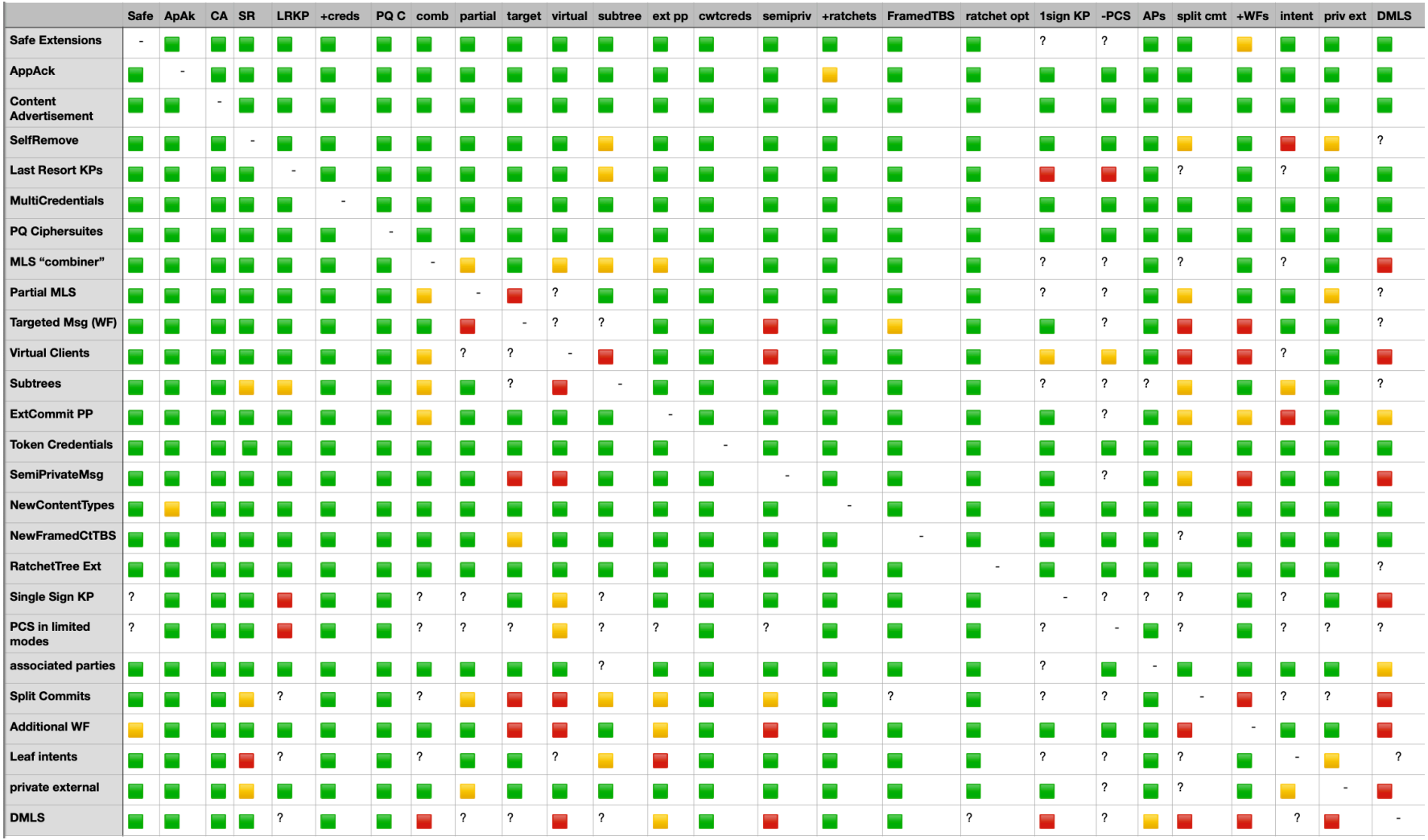
```
struct {
    opaque group_id<V>;
    uint64 epoch;
    ContentType content_type;
    opaque authenticated_data<V>;
    KeyForExternalReceiver keys_for_external_receivers<V>;
    opaque encrypted_sender_data<V>;
    opaque ciphertext<V>;
} SemiPrivateMessage;
```

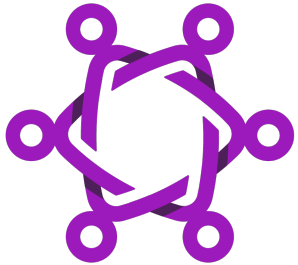
```
struct {
    opaque group_id<V>;
    uint64 epoch;
    ContentType content_type;
    opaque authenticated_data<V>;
    KeyForExternalReceiver keys_for_external_receivers<V>;
} SemiPrivateContentAAD;
```

```
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)
```



Compatibility of MLS extensions


Safe
ApAk
CA
SR
LRKP
+creds
PQ C
comb
partial
target
virtual
subtree
ext pp
cwtcreds
semipriv
+ratchets
FramedTBS
ratchet opt
1sign KP
-PCS
APs
split cmt
+WFs
intent
priv ext
DMLS
Safe Extensions
AppAck
Content Advertisement
SelfRemove
Last Resort KPs
MultiCredentials
PQ Ciphersuites
MLS “combiner”
Partial MLS
Targeted Msg (WF)
Virtual Clients
Subtrees
ExtCommit PP
Token Credentials
SemiPrivateMsg
NewContentTypes
NewFramedCtTBS
RatchetTree Ext
Single Sign KP
PCS in limited modes
associated parties
Split Commits
Additional WF
Leaf intents
private external
DMLS



Questions