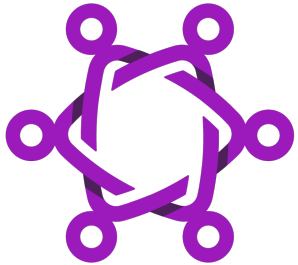

Extension Negotiation

Related to Section 2 of draft-ietf-mls-extensions

Types of MLS Extensions

- Cipher Suites
- Wire Formats
- “Core Struct” Extensions: GroupContext, GroupInfo, KeyPackage, LeafNode (confusingly called “*Extension Types*” in RFC 9420
- Proposals Types
- Credential Types
- Signature Labels
- Public Key Encryption Labels
- Exporter Labels
- Pre-Shared Keys
- Additional Authenticated Data

How do we signal support and mandatory use today?

- RFC 9420 has:
  - Capabilities in a LeafNode

```
struct {
    ProtocolVersion versions<V>;
    CipherSuite cipher_suites<V>;
    ExtensionType extensions<V>;
    ProposalType proposals<V>;
    CredentialType credentials<V>;
} Capabilities;
```

  - Required Capabilities in GroupContext

```
struct {
    ExtensionType extension_types<V>;
    ProposalType proposal_types<V>;
    CredentialType credential_types<V>;
} RequiredCapabilities;
```

- No Capabilities for Wire Formats (Issue #33) and no way to signal labels, pre-shared keys, or AAD

Signal individual extensions or Features?

- Do we signal support for individual extensions or “Features” (which can have multiple extensions) or both?
- How does this interact with the contract for “Safe Extensions” and domain separation?
  - How does the MLS stack know which extensions belong to a specific feature (shared domain)?
  - How does the MLS stack distinguish two proposals belonging to the same safe “feature”?
  - How does the MLS stack know if an UpdatePath is required and if External Proposals are allowed for a particular safe proposal?
- Example: the Associated Parties draft (a “feature”) has 1 GroupContext extension, 1 GroupInfo extension, 3 proposal types, and at least 1 exporter label.
  - according to Safe Extensions, they need to be part of the same safe extension to access the same data. This really needs to be the same feature (the three proposals modify the data in the GroupContext extensions—one of which does not require an UpdatePath)
  - The 3 proposal types *on the wire* still need to be distinguished one from the other

Interaction with Safe Extensions

Safe Extensions register 3 “generic” safe proposal types

- extension_proposal
- extension_path_proposal
- extension_external_proposal (UpdatePath is not required)

But if one safe “feature” has multiple proposals there is no way to distinguish them.

If instead we register safe proposals individually, the stack needs to be separately configured to know if an UpdatePath is required / if external proposals are allowed

Concrete Proposal

- Reserve a new range in the “Extension Types” registry for “features” for clusters of related extensions. These can be for safe “features” or non-safe “features”. These appear in Capabilities.extensions and Required Capabilities.extension_types.
- New Safe Proposal Types and Credential Types do NOT appear individually in the proposals and credentials sections of Capabilities / RequiredCapabilities. Non-safe ones have to appear individually. For safe proposals, the used generic types appear in Capabilities / RequiredCapabilities.

- Safe Proposal Types, Safe Wire Formats, etc. use the “generic” safe extension proposal types wire format type and use the ExtensionContent which references the **feature** identifier.
- If any Safe Feature has multiple items of the same type, they add a discriminator as the first octet after the ExtensionContent struct.

- Add a new SupportedWireFormats LeafNode extension and a new RequiredWireFormats GroupContext extension