MLS Interim 2019-10

Agenda

PRs and Issues

Current PRs

- #213 Change ciphertext content to be standard TLS format.
- #209 Proposals and Commits
- #208 Add authenticated_data to the mls message.
- #200 Non-destructive add
- #198 Initial design for the key schedule Exporter
- #194 Clarified and Changed WelcomeInfo Secrets
- #192 Made it clear that Init messages MUST be sent unencrypted

Current issues [Protocol]

- #160 Advertise a global app generation for a sender
- #142 Prevent suppression of Handshake messages
- #105 Version neServer-initiatedgotiation
- #104 removal
- #100 Allow grouping of GroupOperations within a Handshake
- #95 Decouple curves from symmetric+hash identifiers
- #93 State resync
- #92 ACK / NACK
- #91 User-Initiated Add
- #88 "Pre-warm" trees with some double-joined nodes
- #87 Allow server to cache the roster / tree
- #21 Trivial DoS by a malicious client

Current Issues [Advisory, Editorial]

- #168 Include obligations regarding clients refusing to update more clearly
- #118 Discuss DH cofactor issues and Update DH and EC parameters text.
- #97 UIK rotation
- #76 Retry considerations

- #110 Syntax unification

New issues

- AStree for handshake encryption as well (split at leaves)
- (proposal* commit?)
- Extension points?
- Log-scale clients?

PR Triage

tl;dr

- #209 Proposals and Commits — Let's do it (?)
- #194 [WelcomeInfo Secrets] — Obsoleted by #209
- #192 [Init messages unencrypted] — Obsoleted by #209
- #200 Non-destructive add — Rebase on #209 and land
- #198 [key schedule Exporter] — [[BEURDOUCHE?]]
- #208 [authenticated_data] — Meh. Discuss?
- #213 [Standard TLS format] — Overhead?

Proposals

- Handshake msgs => Proposals
- Each epoch starts with a Commit
  - Proposals identified by sender and truncated hash
  - Applied in order specified
- Transcript is over MLSPlaintext messages carrying Commits
  - Proposals sort of included, transitively via truncated hash
  - **Result** of proposals confirmed in confirmation hash

```
struct {
    ProposalType msg_type;
    select (Proposal.msg_type) {...};
} Proposal;

struct {
    ClientInitKey init_key;
} Add;

struct {
    HPKEPublicKey leaf_key;
} Update;

struct {
    uint32 removed;
} Remove;

struct {
    uint32 sender;
    opaque hash[4];
} ProposalID;

struct {
    ProposalID updates<0..2^16-1>;
    ProposalID removes<0..2^16-1>;
    ProposalID adds<0..2^16-1>;
    DirectPath path;
} Commit;
```

WelcomeInfo & Init=Plaintext Obsoleted

- Sender of Commit also sends Welcome to new members
  - => No need for new members to decrypt Add
  - => No need for secrets in the Add aside from init secret
- Refactor also made clear the taxonomy of high-level messages:
  - MLSPlaintext used for messages within a group
  - Welcome and Init introduce clients to a group
  - So we don’t even have a way to encrypt Init messages

Non-Destructive Add (reminder + rebase)

- Same changes to tree structure
- Add message processing => Add proposal processing

```
struct {
  HPKEPublicKey public_key;
  uint32_t unmerged_leaves<0..2^32-1>;
  optional<Credential> credential;
} RatchetNode;

...


struct {
    HPKEPublicKey public_key;
    Credential credential;
} LeafNodeInfo;

struct {
  HPKEPublicKey public_key;
  uint32_t unmerged_leaves<0..2^32-1>;
} ParentNodeInfo;
```

Exporter

```
epoch_secret
     |
     +--> Derive-Secret(., "exporter", GroupContext_[n])
        = exporter_secret
```

- TLS lets you export keys
- Why not MLS too? Just copy/paste the construction*
  - *Current PR does not just copy/paste the construction)
- Split off from group (epoch) secret? at leaves? both?

AEAD on MLSCiphertext

- Such information would be:
  - Authenticated end-to-end
  - Visible to the DS, but not authenticated (unless over the top)
- Variant: Alternatively/also covered by message signature
- Pros:
  - Might address some new use cases for this (as with SRTP)
- Cons:
  - More complexity
  - Risk of information leakage

MLSCiphertext content encoding

```
// TLS
struct {
  opaque content[TLSPlaintext.length];
  ContentType type;
  uint8 zeros[length_of_padding];
} TLSInnerPlaintext;
```

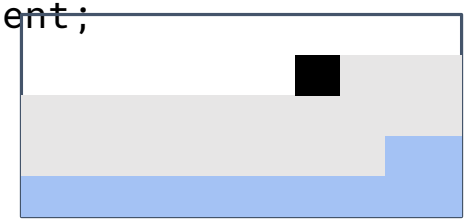
```
// Brendan PR
struct {
  select (content_type) {...}
  opaque signature<0..2^16-1>;
  opaque padding<0..2^16-1>;
} MLSCiphertextContent;
```

```
// Current MLS
struct {
  opaque content[length_of_content];
  uint8 signature[sig_len];
  uint16 sig_len;
  uint8  marker = 1;
  uint8  zero_padding[length_of_padding];
} MLSCiphertextContent;
```

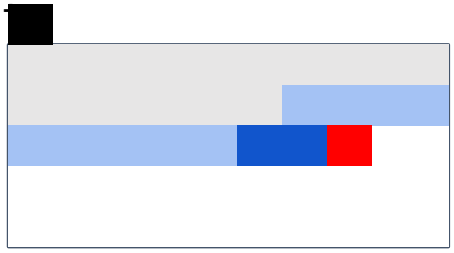
```
// Richard counter-offer
struct {
  uint8 zero_padding[length_of_padding];
  ContentType content_type;
  select(content_type) {...}
  opaque signature[*];
} MLSCiphertextContent;
```

```
// TLS
struct {
  opaque content[TLSPlaintext.length];
  ContentType type;
  uint8 zeros[length_of_padding];
} TLSInnerPlaintext;
```

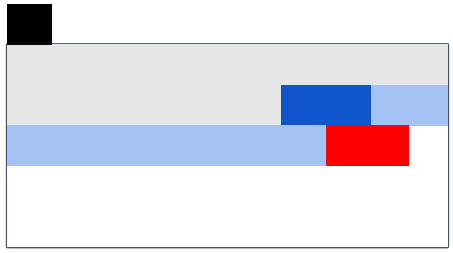
```
// Brendan PR
struct {
  select (content_type) {...}
  opaque signature<0..2^16-1>;
  opaque padding<0..2^16-1>;
} MLSCiphertextContent;
```



```
// Current MLS
struct {
  opaque content[length_of_content];
  uint8 signature[sig_len];
  uint16 sig_len;
  uint8  marker = 1;
  uint8  zero_padding[length_of_padding];
} MLSCiphertextConten
```



```
// Richard counter-offer
struct {
  uint8 zero_padding[length_of_padding];
  ContentType content_type;
  select(content_type) {...}
  opaque signature[*];
} MLSCiphertextContent;
```



Issue Triage