draft-ietf-mls-protocol-04

Topics

Brief summary of changes from -03

Simplify Key Schedule (Issue #90)

Server initiated removal (Issue #104)

Version negotiation (Issue #105)

=== ⬆ Tuesday / Thursday ⬇ ===

Use common framing (Issue #101)

Decouple curves from symmetric+hash identifiers (Issue #95)

Changes from -03

Changes from -03

Terminology alignment

Application key schedule fixes

ECIES -> HPKE

Version negotiation fields

Hash of WelcomeInfo in the Add

Key separation within TreeKEM

Garbage Collection

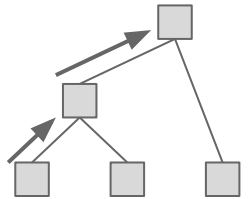
Key Separation within TreeKEM

Konrad pointed out that it would be better if node secrets were not directly related to each other.

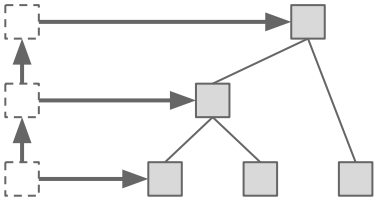
Instead of directly deriving up the tree, derive a chain of ephemeral secrets, from which nodes are derived.

The academics seemed to think this would improve analysis, so 👍

Before:



After:



Garbage Collection

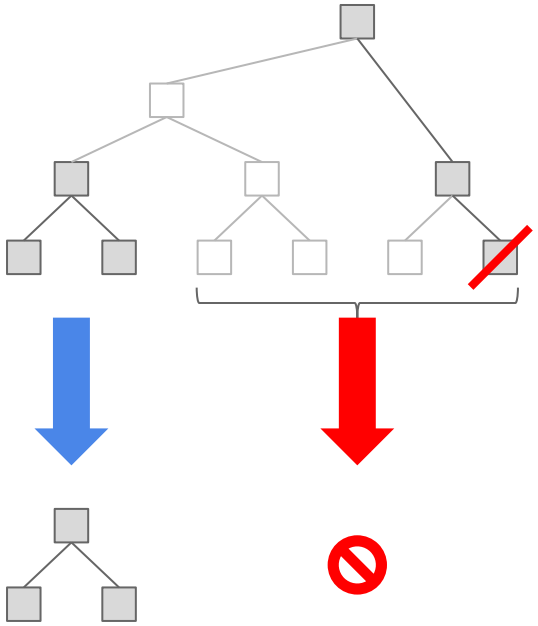
As discussed at the interim, would be good if:

1. New members can be added in blank slots
2. Removing people can shrink the tree

Resolution there was “if we can convince ourselves that the tree invariant is preserved”

$$\forall N \in T, M \in G : secret(M, N) \iff desc(M, N)$$

Q.E.D.!



Simplify Key Schedule (#90)

Problem Statement

GroupState is included in key schedule

GroupState includes roster and tree, which are linear-size

=> Every Derive-Secret operation has to hash linear-size data ☹

Potential Objectives

1. Overall simplicity

2. A smaller coefficient on the linear amount of data hashed

3(weak). Sub-linear amount of data hashed

3(strong). Sub-linear amount of state

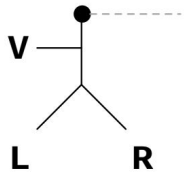
Proposed: Target 3(weak), not 3(strong)

... since 3(strong) would require adding a lot of information to Handshake messages

Tree Hash (née Ampelmann Hash)

```
struct {
  uint8 hash_type = 0;
  optional<LeafNodeInfo> info;          <-- DH key + Credential
} LeafNodeHashInput;                        nullopt <=> blank

struct {
  uint8 hash_type = 1;
  opaque public_key_hash[Hash.length];  <-- should probably be optional
  opaque left_hash[Hash.length];            b/c blank
  opaque right_hash[Hash.length];
} ParentNodeHashInput
```



Root hash is a commitment to the whole contents of the tree

GroupState.(roster, tree) -> GroupState.tree_hash

Why Not Log State?

It might be nice if an endpoint could live without linear-size state (i.e., copath)

This would increase the size of Handshake messages by 2-3x

In addition to direct path, need to send two other log-size lists:

- Merkle(-ish) inclusion proof for signer public key
- Hashes for copath nodes' other children to update node hashes

Maybe could be done as an extension?

Decouple … Identifiers (#95)

MLS-04

```
enum {
  ecdsa_secp256r1_sha256(0x0403),
  ed25519(0x0807),
  (0xFFFF)
} SignatureScheme;


enum {
  P256_SHA256_AES128GCM(0x0000),
  X25519_SHA256_AES128GCM(0x0001),
  (0xFFFF)
} CipherSuite;
```

TLS 1.3

```
enum {
  ecdsa_secp256r1_sha256(0x0403),
  ed25519(0x0807),
  ...
  (0xFFFF)
} SignatureScheme;

CipherSuite TLS_AEAD_HASH = VALUE;

enum {
  secp256r1(0x0017),
  x25519(0x001D),
  (0xFFFF)
} NamedGroup;
```

Which groupings?

(Group + AEAD + Hash), (Signature)

(Group), (AEAD + Hash), (Signature)

(Group + Hash), (AEAD + Hash), (Signature)

How much do we care about...

- Matching hash strength to DH group size?
- Overall level-matching?