draft-barnes-mls-sframe*

Richard Barnes
Raphael Robert

* https://github.com/bifurcation/draft-barnes-mls-sframe
The hard part is always key management

SFrame defines how you encrypt a media payload

What security properties you get from that encryption depends on how the encryption keys are managed

Traditional RTC key-management (SDES, DTLS-SRTP) has addressed 1:1

Lots of use cases nowadays are N:N ~ conferencing
MLS provides **continuous group authenticated key exchange** with **FS / PCS**

- Authenticated key agreement - Makes a key known only to identified parties
- Group - Arbitrary number of parties in the group
- Continuous - Members can join and leave the group
- Forward Security - Recovery
The Shape of MLS [out of scope for SFrame]
Mapping MLS outputs to SFrame inputs

SFrame needs: lookup_key(KID) -> Key

MLS produces a new key per batch of adds/removes/updates (“epoch”)

… from which we need to derive a key per member in the group (“sender”)

So we need:

   Scheme for creating sender keys from MLS epochs

   Encoding of (epoch, sender ID) tuple into KID
KID = Epoch + sender ID

- Epoch N-1
  - Export
  - sframe_epoch_secret_[N-1]

- Epoch N
  - Export
  - sframe_epoch_secret_[N]
  - KDF
  - sender_base_key

- Epoch N+2
  - Export
  - sframe_epoch_secret_[N+2]
Lossy compact encoding

Epochs in MLS are identified by an **8-byte** counter. Heavy!

For compactness: Truncate the epoch to E bits (value of E agreed by members)

E = 4-8 probably sufficient for most cases, esp. with batched key rotation

\[
\text{KID} = (\text{sender\_index} \ll E) + (\text{epoch} \mod (1 \ll E)) \\
\text{sender\_index} = \text{KID} \gg E \\
\text{truncated\_epoch} = \text{KID} \mod (1 \ll E)
\]
The draft in three equations

\[ KID = (\text{sender\_index} \ll E) + (\text{epoch} \mod (1 \ll E)) \]

\[ \text{sframe\_epoch\_secret} = \text{MLS-Exporter}(`SFrame 10 MLS`, ``, \text{AEAD.Nk}) \]

\[ \text{sender\_base\_key}[\text{index}] = \text{HKDF-Expand} (\text{sframe\_epoch\_secret}, \text{encode\_big\_Endian}(\text{index}, 4), \text{AEAD.Nk}) \]
class MLSContext : public SFrame
{
public:
    using EpochID = uint64_t;
    using SenderID = uint32_t;

    MLSContext(CipherSuite suite_in, size_t epoch_bits_in);

    void add_epoch(EpochID epoch_id, const bytes& sframe_epoch_secret);

    output_bytes protect(EpochID epoch_id,
        SenderID sender_id,
        output_bytes ciphertext,
        input_bytes plaintext);

    output_bytes unprotect(output_bytes plaintext, input_bytes ciphertext);
Questions for the WG

Does this approach seem generally correct?

An MLS extn could be used to negotiate parameters (cipher, E). Should we?

Should we adopt a draft that defines this approach?