MLS interim

NYC January 2020

Raphael Robert, Benjamin Beurdouche

Summary

- Ciphersuites
- Server assist / DS specification
- Deniability

Ciphersuites

Open questions:

- Is the signature scheme fixed for the group (vs per client signature scheme)?
- If so, is the signature scheme part of the ciphersuite?
- What ciphersuites do we want?
- Should there be an MTI? (which one)

Ciphersuites

Current list:

`MLS10_128_HPKEX25519_AES128GCM_SHA256_Ed25519`	`{0x00,0x01}`
`MLS10_128_HPKEP256_AES128GCM_SHA256_P256`	`{0x00,0x02}`
`MLS10_128_HPKEX25519_CHACHA20POLY1305_SHA256_Ed25519`	`{0x00,0x03}`
`MLS10_256_HPKEX448_AES256GCM_SHA384_Ed448`	`{0x00,0x04}`
`MLS10_256_HPKEP521_AES256GCM_SHA384_P521`	`{0x00,0x05}`
`MLS10_256_HPKEX448_CHACHA20POLY1305_SHA256_Ed448`	`{0x00,0x06}`

Server assist

- New I-D specifying aspects of the DS
- Emphasis on metadata protection
- 3 modes for group state/metadata distribution

Server assist

3 modes:

- P2P transfer of group state in the Welcome message (as currently defined in protocol draft)
- Ratcheting tree is stored on the DS in plaintext (including hashes, signatures and credentials in leaf nodes)
- Encrypted tree is stored on the DS

Encrypted server assist

Naive approach:

- The tree is stored on the DS symmetrically encrypted
- The key changes with every epoch
- The DS receives the old & new key, decrypts the tree, fans out the message according to the roster, re-encrypts the tree under the new key before persisting on disk (beware of privacy after remove)
- The encryption keys are exported from the key schedule

Encrypted server assist

Advantages:

- Gives us “encryption at rest”
- The roster is only known temporarily to the DS (same as Signal)
- Relatively lightweight computationally
- Scales well for large groups

Encrypted server assist

Protection of metadata in queues:

- Metadata in message headers allow for correlation of messages (guess group membership)
- Messages could be blinded by DS and reconstructed by clients
- Messages could be encrypted by DS and decrypted by clients (KEMed to clients)

Encrypted server assist

Other aspects:

- Clients should authenticate to the DS to prove membership of a group (deniably)
- Clients can retrieve only parts of the tree (like direct path and copath), integrity checks? Probably.

Encrypted server assist

Fancier concepts:

- Use proxy re-encryption (PRE) to re-encrypt tree between epochs (more load on the server, “hipster crypto”, what about the roster?)
- Other technologies?
- The nice thing is that specific mechanisms to protect privacy are mainly independent from the protocol

Deniability

Basic idea:

- Send signature keys over a deniable channel to clients (similar to “sender keys” concept)
- Re-use ClientInitKeys to establish the deniable channel
- Concrete proposal of a mechanism based on HPKE

A volunteer for an I-D ?

Deniability

Signatures that can be deniable:

- Proposals
- Commits
- Application messages

Signatures that cannot be deniable:

- CIKs
- Tree signatures*

(*) Discussion needed

Deniability

Deniable HPKE:

- 1. Use unauthenticated HPKE to send a deniable content such as a completely fresh CIK and an authenticated CIK from the initiator (This CIK is completely unauthenticated)
- 2. Do a static - static between the initiator and the responder CIKs which are strongly authenticated by the AS such that the deniable content is now implicitly authenticated
- 3. Create or use a deniable CIK in the protocol. Beware that the the guarantees given depend on the ContentType. (This step can be dangerous)