MIMI Transport Requirements

Eric Rescorla
ekr@rtfm.com

2022-11-10
Abstract Architecture

Instant Messaging

Voice/Video

Accounts/Identity

Traffic Keys

Traffic Keys

User Credentials

Encrypted Messages

Key Establishment

Message Transport

You are Here
Question: How much are we defining?

- A full system obviously needs a client-to-server protocol
  - Message protection and content need to be E2E
  - ... but message transport is not

- Most existing systems (XMPP, SIMPLE, etc. do it all)

- Is client $\leftrightarrow$ server in scope?
Naming and discovery

- Two main kinds of existing identifiers
  - *System Specific (SSI)*. e.g., “1.650.555.1000 on WhatsApp” (or maybe mimi:16505551000@whatsapp.com)
  - *System Independent (SII)*: e.g., 1.650.555.1000 or ekr

- In general, an SII isn’t enough to automatically contact someone
  - You don’t know what system they are on
  - The same SII may appear on multiple systems (e.g., phone numbers on WhatsApp + iMessage)

- *Discovery* is the process of determining which system(s) an SII appears on
Question: Do we need to support discovery?

1. Only solve for SSIs
2. Solve for SSIs now and build discovery separately
3. Integrate discovery and consent (SPIN, draft-rosenberg)
   - These designs assume that Alice has some out of band channel to contact Bob
   - What about systems that just use handles?

Eric Rescorla ekr@rtfm.com
MIMI Transport Requirements 2022-11-10
Consent?

- Alice can just send messages to Bob if she has his identifier
  - This is a spam vector
- Or does she need to get consent first?
  - Typically this consists of sending an *invite*
  - ... Bob has to accept before seeing Alice’s messages
KeyPackage Availability

• Sending encrypted messages requires the KeyPackage\(^1\)
• This leaks whether the recipient exists
• Potential risk of KeyPackage exhaustion

\(^1\)Recall: the KeyPackage contains the public key of the recipient.
Question: which modes do we support?

1. Alice can send messages to Bob immediately
2. Alice can send messages to Bob but they’re quarantined until Bob accepts
   - Potential concerns about excess data on Bob’s side
3. Alice can’t do anything until Bob consents
Messages and Channels

• (At least) three modalities
  • 1-1 messages
  • ad-hoc groups messages with \( \geq 2 \) people
  • named groups (Channels/rooms)

• Some overlap between group messages and channels
  • **Group** messages are defined by the members
  • Can’t add new members (unlike channels!)

• What about multiple group messages (or 1-1 messages) with the same membership?
  • This is handled inconsistently by existing messaging services
Question: What models do we support?

1. Everything’s an ad hoc group
   - Is this rich enough? What about moderation, etc.?

2. Channels are fundamentally different (XMPP, Slack, etc.)
   - And maybe we don’t need ad hoc group messages?
   - Or do channels first and then ad hoc groups later.

- MLS can support any of these modes
Channel/room Management

- XMPP (MUC) and Matrix have fairly complicated room management
  - Ownership
  - Moderation
  - Kick/ban etc
  - Ask to join chats
- A lot of systems don’t
- This is out of charter scope. Assumption is that this only works on the owning service.
Question: room/channel portability?

- General assumption seems to be a room/channel lives on one system
  - Except for Matrix
- Is it possible to move channels between owners?
  - For instance, if the last member from the owner leaves
  - Linearized matrix allows this
  - XMPP, MTP, etc. don’t seem to allow this
State exists at both levels
  • Transport: which people are in the room?
  • MLS: which keys are in the room?

How tightly in sync are these?

And do they have to be cryptographically bound
Warmup: Bob gets a new phone

Bob gets a new phone. Sends Commit

Alice:K1
Bob:K2
MLS Membership

Alice:K1
Bob:K2, K3
MLS Membership

Room Membership

Room Membership
Charlie added to chat by service

Room Membership

Alice
Bob

MLS Membership

Alice:K1
Bob:K2

Room Membership

Alice
Bob
Charlie

MLS Membership

Alice:K1
Bob:K2

Charlie added to group
Charlie adds himself to MLS group

Room Membership

Alice
Bob
Charlie

Room Membership

Alice
Bob
Charlie

Alice:K1
Bob:K2

MLS Membership

Alice:K1
Bob:K2
Charlie:K4

MLS Membership

Charlie sends Commit
Question: Do we need MLS-level access control?

• MLS just controls which keys are in the group
  • But who decides which Commits are accepted by the group members?
  • For instance, can the owning provider add members
  • Or do group members need to approve it?

• General idea: there is some policy/ACL
  • That clients have enforce

• But how is that policy authenticated?
  1. Just trust the service
  2. Specify MLS-level policy authentication
  3. Require MLS-level policy authentication
Question: Privacy for metadata?

- MLS mostly protects the content of messages
- But what about metadata?
  - Who is messaging who
  - Channel membership
  - Contact lists
- Are we going to try to do anything here?