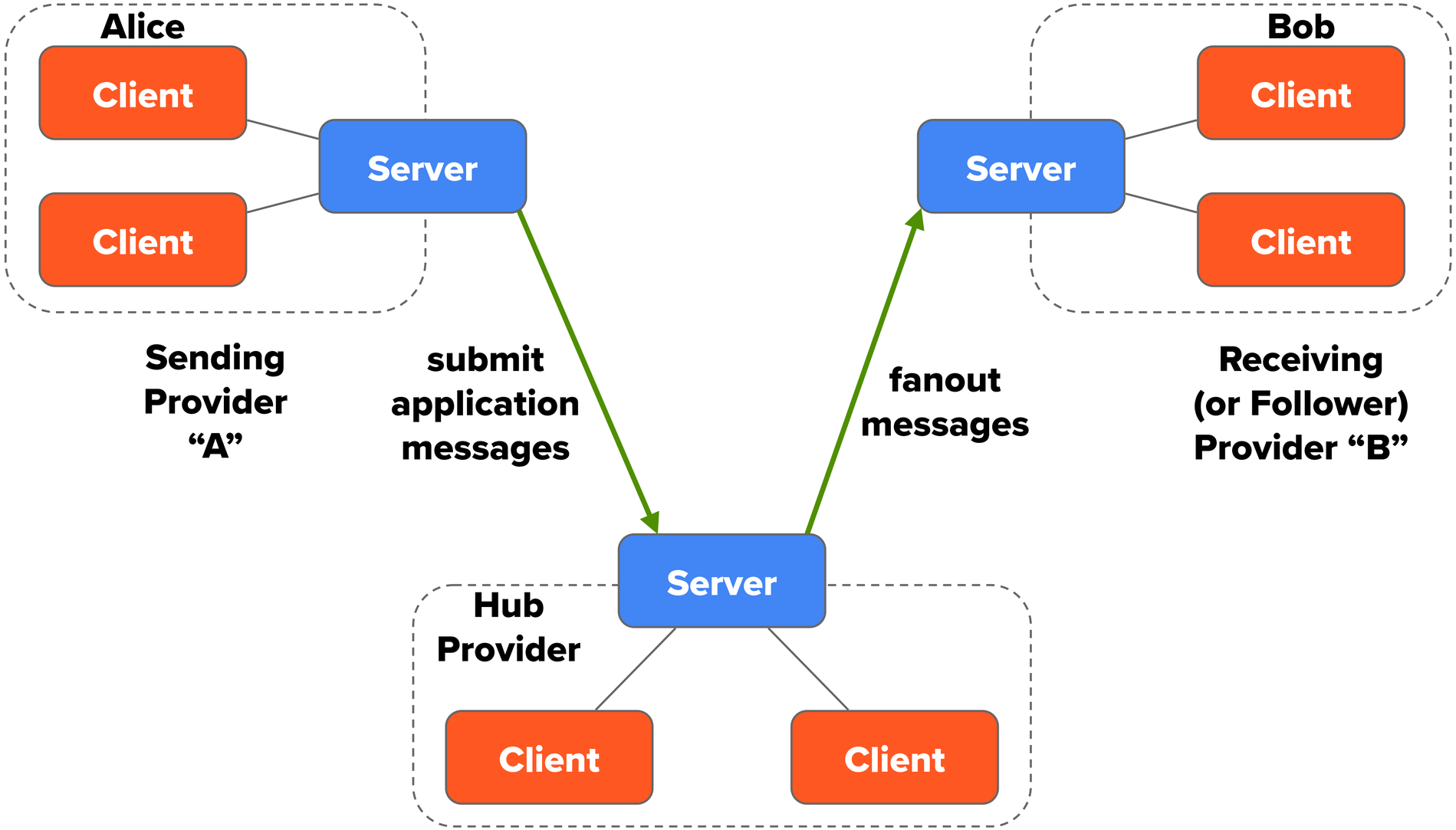
Franking in MIMI
(More Instant Messaging Interopability)

Crypto Forum Research Group (CFRG)
IETF122 Bangkok — 17 March 2025

Rohan Mahy — rohan.ietf@gmail.com

MIMI is Client - Server - Server - Server - Client



MIMI Architecture (not related to franking)

- (sending) Client to (sender's) Server to (Hub) Server to (follower) Server to (receiving) Client
- Hub is responsible for ordering and fanout of all messages.
  - Hub sees end-to-end encrypted application messages (instant messages), but not the content; it attaches its received timestamp.
  - Hub sees the contents of the MLS handshake messages; knows the participants in the room, and their roles (admin, moderator, etc.).
- Local provider can know about which groups its clients belong to, and when one of its clients sends a message. (Out of scope of MIMI.) Doesn't know who or how many participants are in the room from other providers.
- Follower servers don't learn anything about the sender of a message; do not know the members of the group.

MIMI abuse-reporting requirements

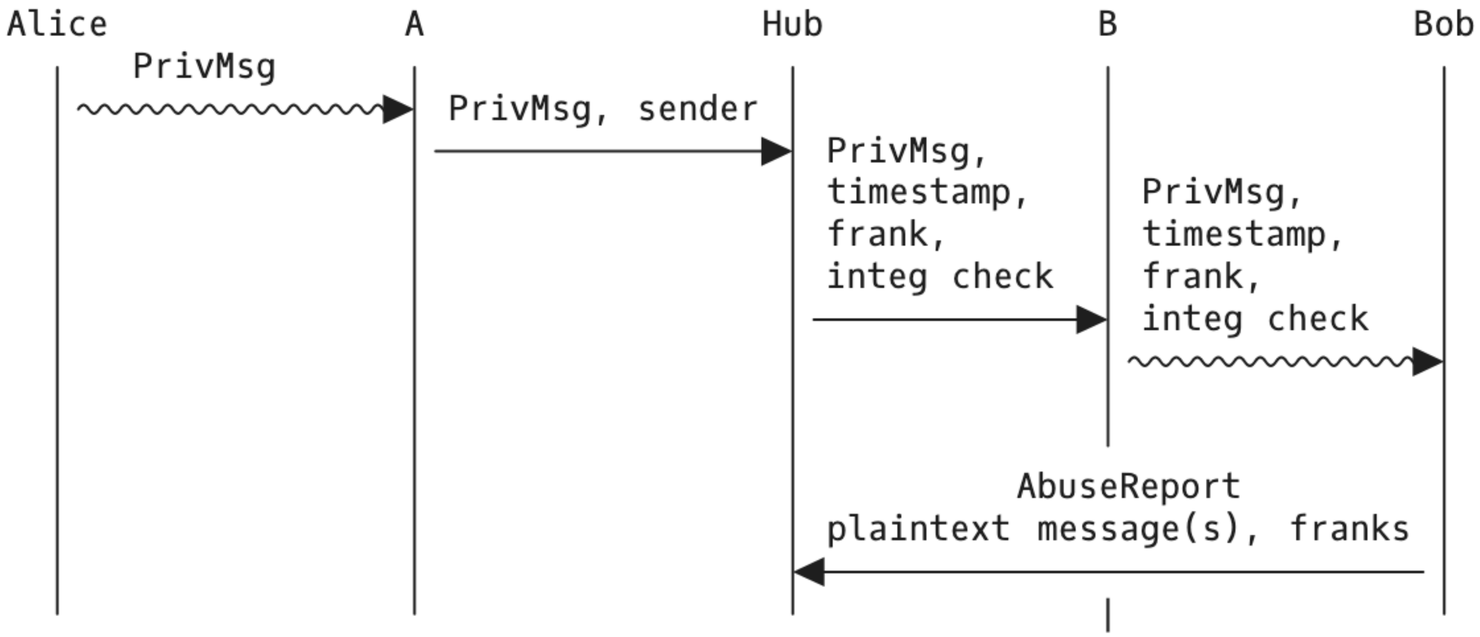
- Receiver of a message can report it as allegedly abusive to Hub, including sending the plaintext of the allegedly abusive message(s).
- Hub can’t read the content of messages unless they are reported
- Receiver can’t attribute a bogus abusive message to a sender
- Sender can’t deny a message they sent
- Hub needs to know the sender of a message in order to enforce policies related to which participants can send messages. Only the Hub knows the policy of the room.
- Receiving client does not need to store ciphertext indefinitely to report a message
- Hub doesn’t need to store ciphertext to verify they received a particular message
- Not especially heavy to implement for client or server.
    - Adding extra roundtrips before sending a message would be especially painful for clients.
- Desirable that after backing up and restoring history (ex: I upgrade my phone), the new client can still report an abusive message received on and backed up from its old client.

Franking of incoming messages

- Incoming messages are ***Franked*** using a scheme similar to Facebook/WhatsApp
- Sending Client
  - Hash a unique per-message salt
  - Includes the salt, its identity, the room URI inside the message
  - Generates a franking tag covering the message and puts it in the AAD
- Hub
  - Encrypts a hash of the tag and the sender URI + room URI + timestamp. (Needs to keep track of which secrets used for which range of timestamps.)
  - Adds an integrity check over the frank, timestamp, and sender URI to make sure other providers did not tamper
  - Forwards to receiver
- Receiving Client
  - Verifies the tag and integrity check are correct or throws message away (sender didn’t lie, follower didn’t tamper)

MIMI Franking proposal: What data goes where?

- M = f(salt, content, sender, room)
- franking_tag = HMAC(salt, M) *[AAD]*
- context = sender || room || timestamp
- C1 = Enc(K_{app_ratch}, M, C2)
- C3 = Enc(K_{send_ratch}, sender)
- PrivMsg = g(C1, franking_tag, C3)
- frank = Enc(K_{hub}, franking_tag || context)
- integrity_check = HMAC($K_{hub+member}$, frank || context)
- frank_sig = Sig($PrivKey_{hub}$, hash(frank || context)) ← *(possible option)*



Members + Hub secret

- Currently the document assumes that the current members of the MLS group and the Hub share a (per-epoch) secret established using draft-kohbrok-mls-associated-parties.
- Integrity of the hub timestamp and frank (plus a unique tag provided by the sender to prevent replay) are verified by members using an HMAC with a secret derived from the secret above.
- Note: Integrity could instead be provided by a signature by the hub over the same fields, but would be somewhat slower. It would probably use a key derived from or related to an the existing key for the Hub known to the group members (in the MLS `external_senders` or `external_receivers` extension)

Reporting

- Report is sent by receiver ***directly to the Hub*** (not through reporting client's provider)
  - Contains decrypted message(s) and frank(s)
  - Hub can verify that it franked a reported message from the same sender, in the relevant room.
  - Hub can verify the message plaintext presented by the reporter matches a message sent by the Sender (by calculating the franking_tag).

Questions?

Who is willing to review?

Thanks on behalf of the authors and the MIMI WG!

Table of who gets what fields?

field	property	where is field	Alice	A	Hub	B	Bob	Notes
salt	confidentiality via MLS app data	content	✅ originator	❌	❌	❌	✅	
sender	partial confidentiality	1. MIMI submit 2. MLS sender 3. content	✅ originator	yes	✅	❌	✅	confidential to B 1. receivers discard if not correct 3. to prevent false attribution
content	confidentiality via MLS app data	MLS app data	✅ originator	❌	❌	❌	✅	
franking_tag	integrity	MLS AAD	✅ originator	yes	✅	yes	✅	clients discard if tag does not match content
room ID	corresponds	MIMI messages	✅	✅	✅ originator	✅	✅	“key field” in MIMI; each group ID has one room ID
MLS group ID	integrity	MLS “header”	✅	yes	✅ originator	yes	✅	
MLS epoch	integrity	MLS “header”	✅	yes	✅	yes	✅	reveals how busy the room is; set by last committer
hub timestamp	partial integrity	1. MIMI submit resp 2. MIMI fanout	✅	yes	✅ originator	yes	✅	clients discard if not correct; only protected during franking
frank	partial integrity	1. MIMI submit resp 2. MIMI fanout	✅	yes	✅ originator	yes	✅	clients discard if not correct

Franking of incoming messages (more)

- To reference Figure 3 of "Message Franking via Committing Authenticated Encryption", the changes to the protocol are:
  - K_f (salt) is included inside M,
  - C1 is the result of AEAD encryption of plaintext M with AAD K_f.
  - K_r is the MLS app ratchet key; K_{FB} is the hub's symmetric franking key; C2 is the franking_tag; a is the frank
  - md (context) includes the room ID as the recipient instead of a specific user (Bob); **sender** (Alice) **does not appear on the wire** from Hub to B
- Additions
  - In addition to C1, C2, and a, the server sends a franking integrity check hash (since Alice is not visible to the receiver)
  - C3 is the encryption of the sender

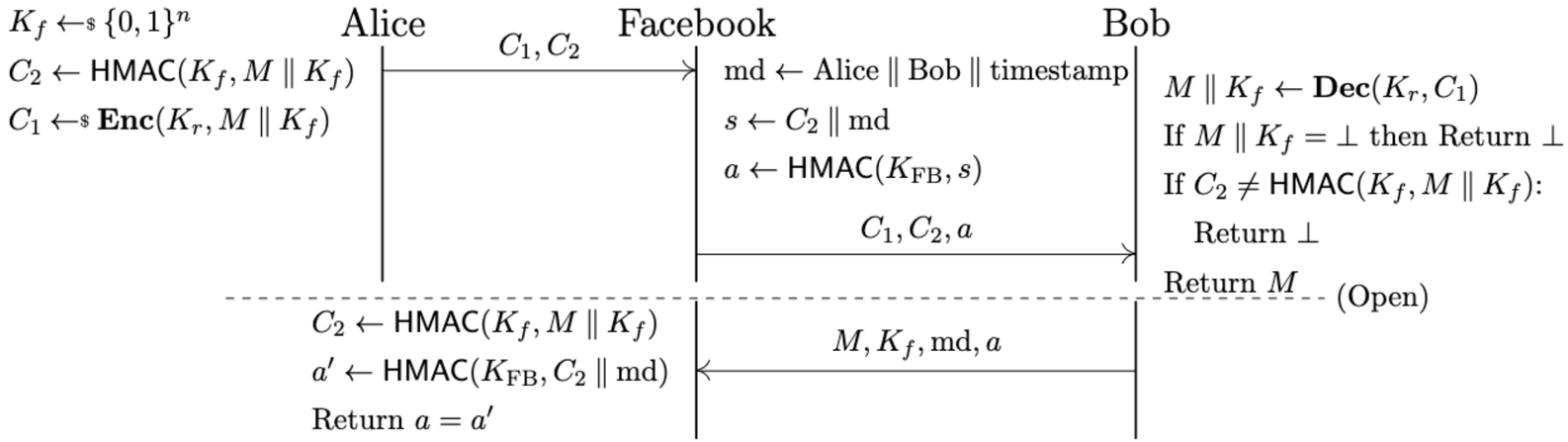


Figure 3: Facebook's message franking protocol [51]. The key K_r is a one-time-use symmetric key derived as part of the record layer protocol. The top portion is the sending of an encrypted message to the recipient. The bottom portion is the abuse reporting protocol.