MIMI Protocol abuse reporting

draft-ietf-mimi-protocol PR#83

Rohan Mahy — rohan.ietf@gmail.com

MIMI Interim 02-Oct–2024

Abuse-report requirements

- Receiver of a message can report it as allegedly abusive to the Hub provider.
    - Why the hub?
        - Because it knows the policy of the room
        - Because all messages go through the hub (see below)
    - Sends 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
- Follower servers do not learn sender/content of any message.
- Hub servers need to know the sender of a message in order to enforce policies related to which participants can send messages.
- Receiving client does not need to store ciphertext indefinitely to report a message (could decrypt/store in history, then report later)
- Hubs don’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.
- Issue: Receiver sending an abuse-report directly to the Hub (not through their local provider) sort of violates the principle that we are not defining client to server protocol. If it goes through the receiver’s local server, this leaks both the sender’s identity and the message content to the receiver’s local server, unless we further the encrypt the the report for the Hub (ex: using the HPKE external_receiver public key of the hub).
- Desirable that after backing up and restoring history (ex: I upgrade my phone), the new client can still report an abusive message
- * There may be additional requirements (Hub reporting its findings to sender’s provider) that can be implemented as additional endpoints that do not have any client implementation implications. Probably handled with SLAs

Mechanism described in PR

- Incoming messages are ***Franked***
- Sender
  - Creates a unique per-message secret
  - Encodes the key, 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
  - Forwards to receiver
- Receiver
  - Verifies the tag is correct or throws message away (sender didn’t lie)

Franking of incoming messages (more)

- To reference Figure 3 of "Message Franking via Committing Authenticated Encryption", the changes to the protocol are:
- - Kf is included inside M,
- - C1 is the result of AEAD encryption of plaintext M with AAD Kf.
- - md includes the room ID as the recipient instead of a specific user (Bob)
- - in addition to C1, C2, and a, the server sends the hash of md (since Alice is not visible to the receiver)

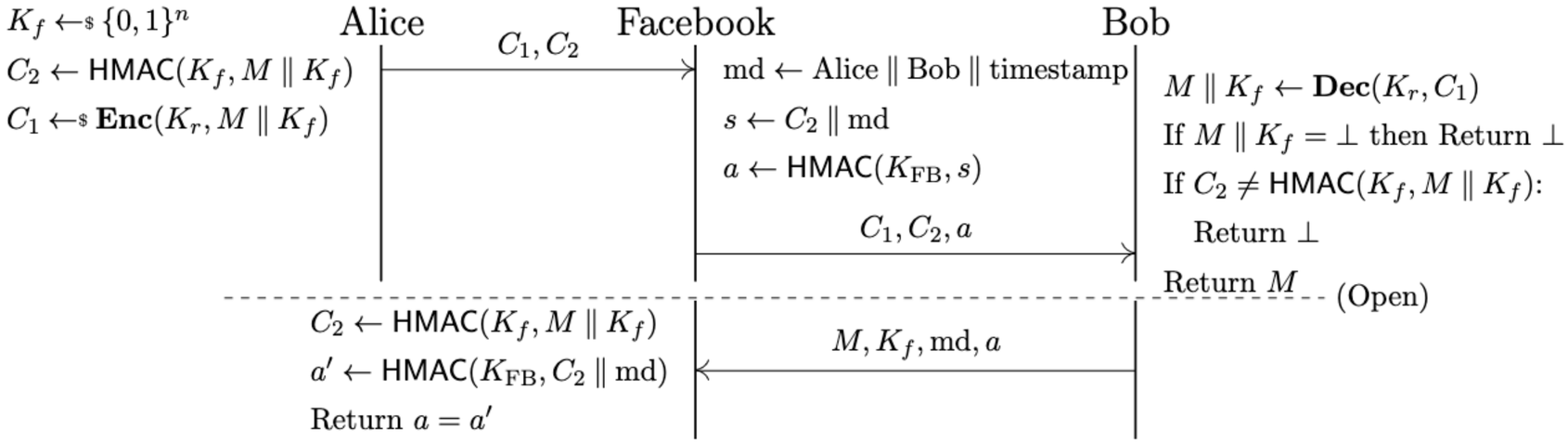

Figure 3: Facebook's message franking protocol [51]. The key K_r is a one-time-use symmetric key derived as part of the record layer protocol. The top portion is the sending of an encrypted message to the recipient. The bottom portion is the abuse reporting protocol.

Mechanism described in PR

- Report is sent by receiver
  - Contains decrypted message and frank
  - Hub can verify that it franked THIS MESSAGE from the same sender
  - Hub can verify the message presented by the reporter matches a message sent by the Sender.

Issue: Sender privacy too weak

- The main issue is that a follower server can brute force guess the sender of the original message from the franking context. No bueno.
- If we had a key shared between the members and the hub, we could mix this with the context and that attack would go away.
  - Requires a semi-standard way to do this in MLS.