More Instant Messaging Interoperability
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

R. L. Barnes
Cisco

20 October 2025

An Architecture for More Instant Messaging Interoperability (MIMI) draft-ietf-mimi-arch-02

Abstract

The More Instant Messaging Interoperability (MIMI) working group is defining a suite of protocols that allow messaging providers to interoperate with one another. This document lays out an overall architecture enumerating the MIMI protocols and how they work together to enable an overall messaging experience.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-mimi-arch/.

Discussion of this document takes place on the More Instant Messaging Interoperability Working Group mailing list (mailto:mimi@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mimi/. Subscribe at https://www.ietf.org/mailman/listinfo/mimi/.

Source for this draft and an issue tracker can be found at https://github.com/bifurcation/mimi-arch.

Status of This Memo

Table of Contents

1. Introduction

Today, there are many providers of messaging functionality. A provider typically provides the client software (e.g., a mobile app) and the servers that facilitate communications among clients. The core function of MIMI is enabling users to have messaging interactions across message providers.

This overall goal breaks down into several sub-goals:

* Message formats that enable the user-level features of a messaging system

* Tracking of state across multiple providers

* End-to-end security of user messages

* Transport of protocol messages among providers

In this document, we describe the high-level functions of these protocols, and how they work toegether to enable an overall messaging application.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used by this document and the MIMI working group for a shared understanding of the overall system:

Messaging Provider or _Provider_: A service offering instant messaging to users. Each provider has a logical server to route events between users (or clients, more specifically).

User: A (normally) human operator of a client. Users have a distinct _User ID_ to canonically identify them.

Client: A user interface for messaging, performing encryption as needed. Presents chats to the user to interact with. Synonymous with _MLS Client_. Clients have a _Client ID_ to canonically represent them among the user’s other clients. Clients MAY also be called _Devices_ to differentiate them from a named application.

Server: A logical location operated by a messaging provider which ensures message and information delivery. A server may be realized by multiple physical computers. Users have accounts on a specific server. Servers are considered to be "participating" in a room if they have at least one joined user participant.

Hub: The specific server in a room with operational responsibility for delivery between all servers in the room. This includes messages and, where applicable, information about the room or underlying cryptographic state.

Follower: All non-hub servers in a room. Followers are required to interact with the hub server to send messages, and are responsible for "last mile" delivery of a message to its local users.

Room: The virtual space where users communicate. This is semantically different from an _MLS Group_: an MLS Group is responsible for handling client keys while a room is simply the user-facing construct for communications. Rooms have a cryptographic state component as well. MLS uses a Group to represent that state. Rooms have a _Room ID_ to canonically identify them. Rooms may additionally be called _Chats_, _Conversations_, or _Channels_.

State: The room’s user participation information, cryptographic state, and other metadata as required, collectively.

User Participation: The set of users which can engage in conversation within a given room, or could engage if they complete further actions. For example, users may be "invited" to converse, and can accept (join) or reject (leave). Users are not considered to have "membership". Instead, users are _participants_ in the room. A list of these users is called the _Participant List_.

Client Membership: The set of clients belonging to participating users within a given room’s cryptographic state. Clients are not considered to have "participation". Instead, clients are _members_ of the room. A list of these clients is called the _Membership_ for a room.

Active Participant: A participating user with at least one client member in the room’s cryptographic state.

Inactive Participant: A participating user with zero client members in the room’s cryptographic state. Users in this state may be unable to decrypt messages sent while no clients are members.

Add (Operation): Places a client or user into a joined state, able to converse with other clients/users also in the joined state. When adding a user, all of their clients are implicitly added as well.

Remove (Operation): Kicks a client or user from a room, preventing further conversation being received from that entity, and preventing that entity from seeing future conversation. When a user is removed, all of their clients are explicitly removed as well. Removal may be voluntary or non-voluntary.

Policy: The authorization structure within a room. Policy governs whether an action is possible, such as whether User A can add User B to the room. Policies are changed over time by users and servers.

Policy Envelope: Set by the hub server during room creation, the set of policies which can be changed in the room.

Event: A structure used by servers to relay changes to the room and messages from clients.

State Event: An event which mutates the _state_ of the room. These may partially be visible to the servers of the room for authentication and authorization.

Message Event: An event containing a message from a client. Contents are not visible to servers in the room.

3. Overall Scope

Figure 1 shows the critical entities in the overall MIMI system and their interactions. Each human _user_ is represented in the system by one or more _clients_, where each client is a specific software or hardware system belonging to a single user. Each provider is represented by a _server_ (logically a single server, but possibly realized by multiple physical devices).

Messaging interactions are organized around _rooms_. All messaging interactions take place in the context of a room. (Some non-messaging interactions may take place outside of a room, such as operations to fetch information required to set up a room.) Rooms have a notion of _user participation_ as well as _client membership_, both tracked as lists. Rooms additionally have policies about things like how the room may be joined and what capabilities each member/participant has.

The protocol interactions that drive a room unfold among the servers whose users are participants in the room. There is exactly one _hub_ server for the room, which is in primary control of the room. All other servers are known as _followers_. Follower servers interact directly with the hub server. Interactions between clients occur indirectly, via the servers for the clients' providers.

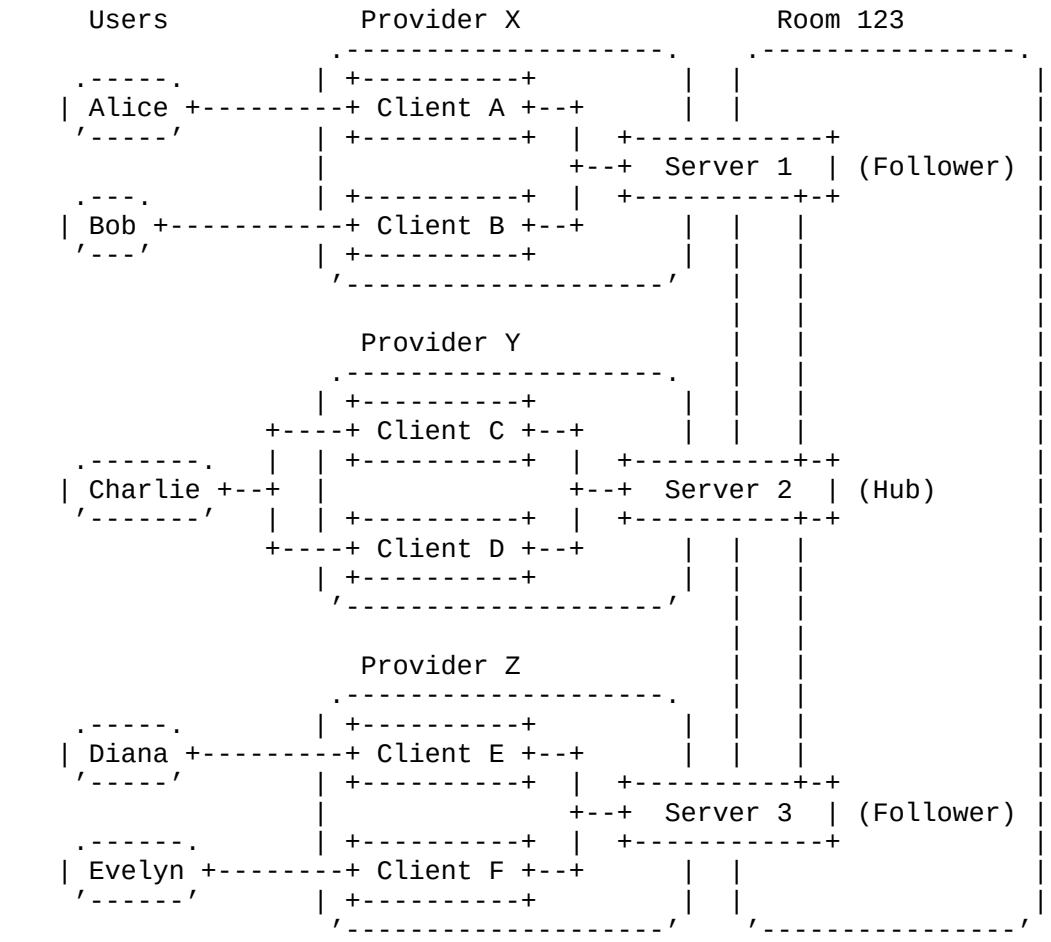

Figure 1: MIMI Entities and Interactions

4. Room State

A room represnts a messaging interaction among a specific set of clients, with a single _state_. A major goal of the MIMI protocols is to syncrhonize the state of a room across all of the servers and clients participating in the room. Changes to the room's state can be proposed by either clients or servers, though as dicussed in Section 4.4, one important aspect of the room's state is an authorization policy that determines which actors are allowed to make which changes.

The creation of a room is a local operation on the hub server, and thus outside the scope of MIMI. The hub server establishes the initial state of the room.

The state of the room includes a few types of information, most importantly:

* The end-to-end security state of the room

* The user-level participation state of the room

* The authorization policy for the room

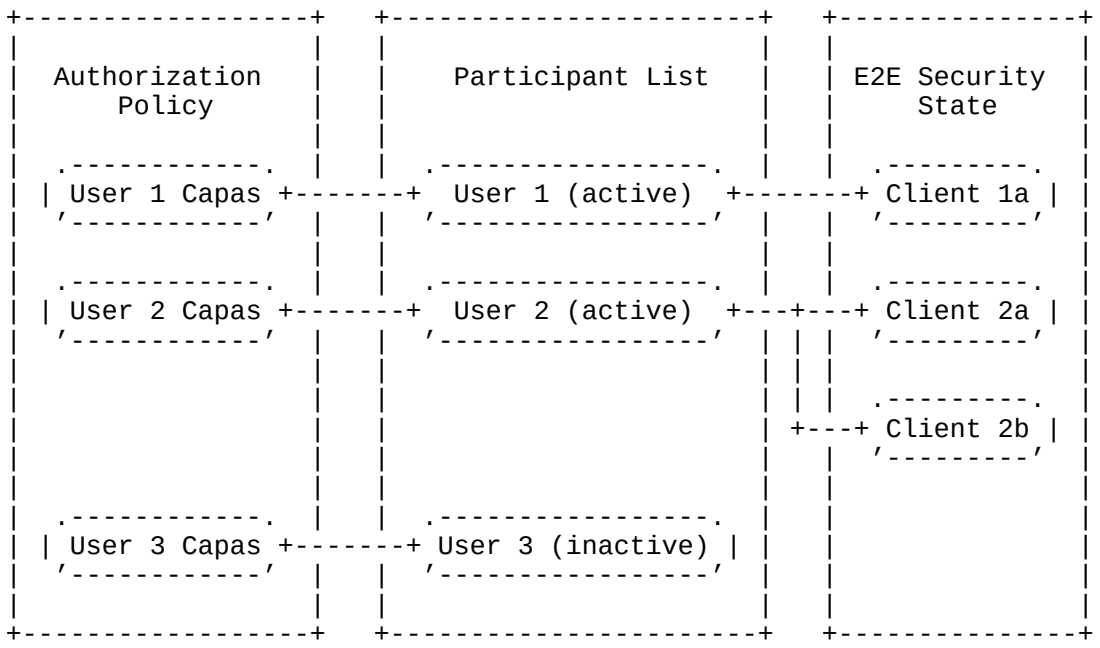

Figure 2: Elements of the Room State

4.1. End-to-End Security State

Messages sent within a room are protected by an end-to-end security protocol to ensure that the servers handling messages cannot inspect or tamper with messages. This means that the required cryptographic keys need to be provisioned to any client from which a user can interact with the room. The state of this end-to-end security protocol thus represents the precise set of clients that can send and receive messages in the room, the most precise notion of membership for a room. A client that has the required keys for end-to-end

security is said to be a member of the end-to-end security state of the room.

The end-to-end security state of a room has public and private aspects. Servers may store the public aspects of the end-to-end security state, such as identities and credentials presented by the clients in the room. The private aspects of the group, such as the symmetric encryption keys, are known only to the clients.

4.2. Participants and Members

The _participant list_ for a room is the set of users who are allowed to interact with the room in some way. The specific list of ways in which a user may participate is defined by authorization policy, as discussed in Section 4.4.

Note the parallel terminology with regard to inclusion of clients or users in the room:

* A _client_ is a _member_ of the _end-to-end security state_ of the room

* A _user_ is a _participant_ in the room

The user-level _participant list_ and the client-level _membership_ of the room are distinct entities managed by separate protocols, but they must be consistent with each other. A client may be a member of the E2EE state of a room only if its user is a participant in the room. However, a user may be a participant in a room without any client belonging to the user being part of the end-to-end security state of the room. (Such a user will not be able to read or send messages, but may be able to take other actions. It is up to client implementations how this state is represented.)

A user with at least one client joined to the end-to-end security state of the room is known as an _active user_, since such a user can fully participate in the room.

4.3. Membership Changes

The participant list and client membership of a group can change over time, via _add_ and _remove_ operations at both the user level and the client level. These operations are independent at the protocol level: For example, a user may be added to a room before any of its clients are available to join, or a user may begin using a new device (adding the device without changing the user-level participation).

As discussed above, user-level participation and client-level membership must be kept in sync. When a user is added, some set of their clients should be added as well; when a user leaves or is evicted, any clients joined to the room should be removed. The cryptographic constraints of end-to-end security protocols mean that servers cannot perform this synchronization; it is up to clients to keep these two types of state in sync.

4.4. Policy

Each room has an associated _policy_ that governs which protocol actions are authorized for the room while the policy is in effect. The policy defines several aspects of the room's behavior, for example:

* Admission policy: Do new members need to be explicitly added by a current member of the room, or can some set of users join unilaterally?

* Capabilities per user: Is a given user allowed to ...

  - Send messages in the room?

  - Add or remove other users?

  - Grant or deny capabilities to other users?

* Capabilities per server: Is a given server participating in the room allowed to...

  - Add or remove users?

  - Grant or deny capabilities to users?

The hub server for a room defines the _policy envelope_ for the room, the set of acceptable policies for the room. The hub also sets the initial policy for the room when it is created. Pursuant to that initial policy, the clients and servers participating in the room may then make further changes to the policy.

At any given time, all of the clients and servers have the same view of the room's policy. A client or server that receives an event that is not compliant with the room's policy may thus safely discard it, since all of the other participating clients/servers should also reject the event.

5. Protocol Interactions

As shown in Figure 3, MIMI protocols define server-to-server interactions and client-to-client interactions. Each client interacts with the overall system by means of its provider's server (whether hub or follower). Client-to-client interactions are done by means of these servers.

The messages sent within a room are forwarded among participating clients by servers. However, messages are protected by an end-to-end security protocol so that their content is only accessible to the clients participating in the room.

In addition to forwarding messages, servers participate in control protocols that coordinate the state of the room across the participating providers. Both message forwarding and control protocols leverage a common framework for sharing _events_ among servers. Events are protected with the same end-to-end security protocol as clients' messages, so that the actors updating a room are authenticated and the clients participating in a room can confirm that they agree on the state of the room.

Note that some parts of the overall system are explicitly out of scope for MIMI. Namely, client-server interactions internal to a provider (indicated by "(Provider)" in Figure 3) can be arranged however the provider likes.

The MIMI protocol implemented by servers thus incorporates a few sub-protocols:

* A transport protocol for sending room events among servers
* A state synchronization protocol for coordinating updates to the room state
* A message forwarding protocol

A common end-to-end security layer provide common security services to all of these functions.

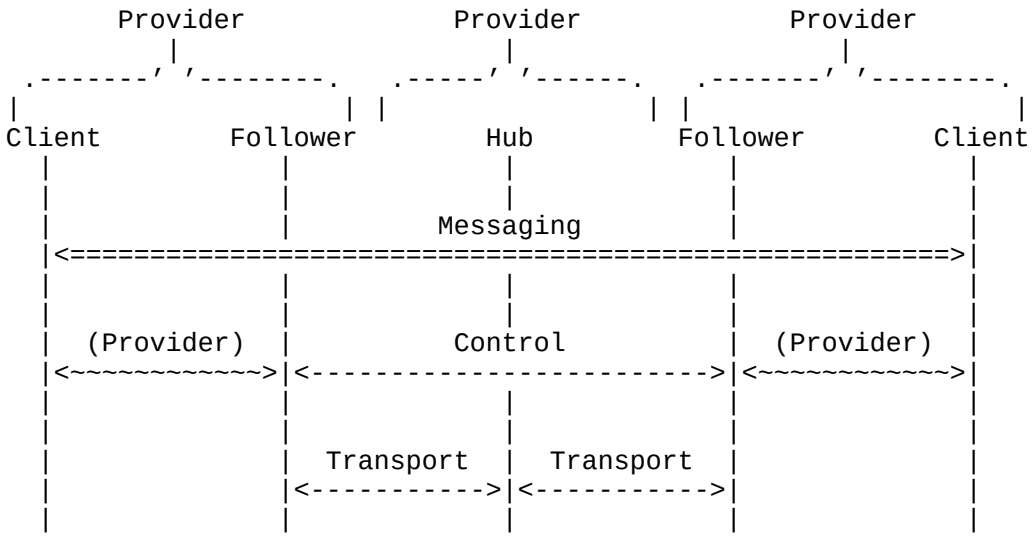

Figure 3: MIMI Protocols

5.1. End-to-End Security

As noted above, all of the clients participating in a room are part of the same end-to-end security context. This allows them to protect their messages so that they are secure from inspection or tampering as they transit MIMI servers.

In addition to the message protection noted above, the end-to-end security layer of the protocol provides a few additional functions to the remainder of the protocol:

* Authentication of the actors making changes to a room

* Confirmation that the clients in a room agree on the state of the room

The authentication function allows MIMI servers to verify the identity of a client making a change to the room, as an input to a policy evaluation to check whether the change is authorized. MIMI servers can make changes to a room, within the bounds of the room's authorization policy. Thus, MIMI servers also need to be represented in the end-to-end security state of the room, but as actors who can only authenticate, and are not given access to confidential end-to-end security state. In MLS terms, they are added as external senders, not as members of the group.

The MIMI protocol includes end-to-end security components to keep the end-to-end security state of the room aligned with the room's participant list, and to ensure that all clients participating in the room have the proper configuration (e.g., trusting the appropriate set of servers).

5.2. Events and Transport

A room's activities are realized by servers exchanging _events_. Events come in two types:

* *State events*, which make changes to the room state

* *Message events*, which describe actual messaging activity in the room

Each event originates at one of the servers participating in the room (possibly as a result of some interaction with a client). The originating server sends the event to the hub server for the room, who distributes it to the other follower servers.

Each event is authenticated by its originating server so that all other participating servers can verify its origin, even those to whom the event has been distributed by the hub. If an event was ultimately created by a client, it is also authenticated by the client that created it.

The overall MIMI protocol defines this event framework, including its authentication scheme, as well as the mechanics of how events are delivered from one server to another.

5.3. Room State Synchronization

The servers involved in a room use an application state synchronization protocol to coordinate changes to a room's state, particularly those listed in Section 4. A few types of room state are synchronized, in what can be viewed as independent control sub-protocols:

A *policy control protocol* distributes information about the policy envelope of a room, and allows participants in a room to propose changes to the policy within that envelope.

A *participation control protocol* manages the user-level membership of the room, including the various ways that members might join or leave a room (or be added/removed by other users).

As discussed above, the *end-to-end security control protocol* manages the end-to-end security state of the room. This protocol also allows servers to distribute cryptographic information that clients have pre-registered, which allows clients to be asynchronously added to rooms.

5.4. Messages

Mesage events are end-to-end secure objects that carry application messages, often in the standard MIMI content format. The end-to-end encapsuation ensures that the message content is only accessible to the clients participating in the room, not the servers that help to distribute it.

The MIMI message format [I-D.ietf-mimi-content] defines how clients achieve the various features of a messaging application, for example:

* Text messaging
* File attachements
* Replies
* Reactions
* Initiation of real-time sessions

Messages transit MIMI servers by means of a *message forwarding protocol*, which carries an opaque, encrypted message payload together with enough metadata to facilitate delivery to the clients participating in a room.

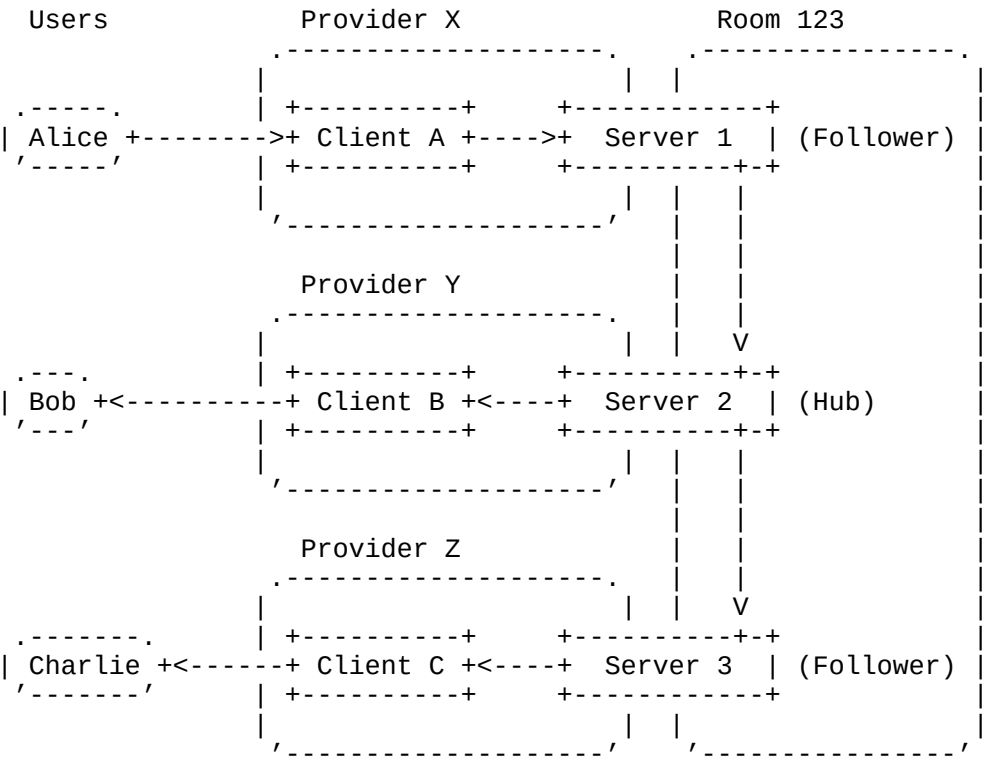

Figure 4: The hub fans out messages to participating servers; servers deliver messages to users' clients.

When a client sends a message, the message is delivered to its provider's server using some provider-internal mechanism. If the provider is not the hub, then the server forwards the message to the hub for delivery. In either case, the hub distributes the message to all of the servers participating in the room. Each provider's server then forwards the message to clients of users who are participating in the room.

6. Actors, Identifiers, and Authentication

There are several types of entity to be identified in the MIMI system, including:

* Rooms,

* Servers,

* Users, and

* Clients.

A server's identity is effectively the identity of the provider it represents. A room is hosted by a single hub server at a given time, so its identity is within the scope of the hub server's identity.

To facilitate the application of policies based on these identifiers to protocol actions, each actor presents one or more credentials that associate a signature key pair to their identifiers. Protocol messages are then signed by their senders to authenticate the origin of the message.

For a deeper discussion of identity, see [I-D.mahy-mimi-identity].

7. Security Considerations

TODO

* Authorization policy attached to a room

* E2E security for messages provided by message delivery protocol

* E2E/E2M/M2E/M2M security for events provided by transport protocol

* HbH security provided by TLS

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

9.2. Informative References

[I-D.ietf-mimi-content] Mahy, R., "More Instant Messaging Interoperability (MIMI) message content", Work in Progress, Internet-Draft, draft-

ietf-mimi-content-07, 7 July 2025,
<https://datatracker.ietf.org/doc/html/draft-ietf-mimi-content-07>.

[I-D.mahy-mimi-identity]
Mahy, R., "More Instant Messaging Interoperability (MIMI) Identity Concepts", Work in Progress, Internet-Draft, draft-mahy-mimi-identity-03, 3 March 2025,
<https://datatracker.ietf.org/doc/html/draft-mahy-mimi-identity-03>.

Acknowledgments

TODO acknowledge.

Author's Address

Richard L. Barnes
Cisco
Email: rlb@ipv.sx