Workgroup: MIMI BoF
Internet-Draft: draft-mahy-mimi-identity-00
Published: 11 July 2022
Intended Status: Informational
Expires: 12 January 2023
Authors: R. Mahy
Wire

More Instant Messaging Interoperability (MIMI) Identity Concepts

Abstract

This document discusses concepts in instant messaging identity interoperability when using end-to-end encryption, for example with the MLS (Message Layer Security) Protocol. The goal is to explore the problem space in preparation for framework and requirements documents.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 12 January 2023.

Table of Contents

1. Introduction

The IETF began standardization work on interoperable Instant Messaging in the late 1990s, but since that period, the typical feature set of these systems has expanded widely and was largely driven by the industry without much standardization or interoperability. The MIMI (More Instant Messaging Interop) problem outline [I-D.mahy-mimi-problem-outline] identifies areas where more work is needed to build interoperable IM systems.

The largest and most widely deployed Instant Messaging (IM) systems support end-to-end message encryption using a variant of the Double Ratchet protocol [DoubleRatchet] popularized by Signal and the companion X3DH [X3DH] key agreement protocol. Many vendors are also keen to support the Message Layer Security (MLS) protocol [I-D.ietf-mls-protocol] and architecture [I-D.ietf-mls-architecture]. These protocols provide confidentiality of sessions (with Double Ratchet) and groups (with MLS) once the participants in a conversation have been identified. However, the current state of most systems require the end user to manually verify key fingerprints or blindly trust their instant messaging service not to add and remove participants from their conversations. This problem is exacerbated when these systems federate or try to interoperate.

While some single vendor solutions exist, clearly an interoperable mechanism for IM identity is needed. First this document attempts to

articulate a clear description and semantics of different identifiers used in IM systems. Next the document provides an example of how to represent those identifiers in a common way. Then the document discusses different trust approaches. Finally the document surveys various cryptographic methods of making and verifying assertions about these identifiers.

Arguably, as with email, the success of XMPP [RFC6120] was partially due to the ease of communicating among XMPP users in different domains with different XMPP servers, and a single standardized address format for all XMPP users.

The goal of this document is to explore the problem space, so that the IETF community can write a consensus requirements document and framework.

2. Types of Identifiers

IM systems have a number of types of identifiers. Few (or perhaps no) systems use every type of identifier described here. Not every configuration of the same application necessarily use the same list of identifiers.

Domain identifier: A bare domain name is often used for discovery of a specific IM service such as example.com or im.example.com. Many proprietary IM systems operate in a single domain and have no concept of domains or federation.

Handle identifier: A handle is an identifier which represents a user or service. A handle is usually intended for external sharing (for example it could appear on or in a paper or electronic business card). IM systems could have handles which are unscoped (don't contain a domain) or scoped (contain a domain). Unscoped handles are often prefixed with a commercial at-sign ("@"). Handles in some services are mutable. For example, @alice_smith could become @alice_jones or @alex_smith after change of marital status or gender transition.

Protocol	Identifier Address	Example
Jabber/XMPP	Bare JID	juliet@example.com
SIP	Address of Record (AOR)	sip:juliet@example.com
IRC	nick	@juliet
Generic example	"unscoped handle"	@juliet
Generic example	"scoped handle"	@juliet@example.com
Email style	Mailbox address	juliet@example.com

Table 1: some Handle identifier styles

User or account identifier: Many systems have an internal representation of a user, service, or account separate from the handle. This is especially useful when the handle is allowed to change. Unlike the handle, this identifier typically cannot

change. For example the user identifier could be a UUID or a similar construction. In IRC, a user identifier is prefixed with a "!" character (example: !jcapulet1583@example.com for the "nick" @juliet).

Client or Device identifier: Most commercial instant messaging systems allow a single user to have multiple devices at the same time, for example a desktop computer and a phone. Usually, each client instance of the user is represented with a separate identifier with separate keys. Typically these identifiers are internal and not visible to the end-user (XMPP fully qualified JIDs are a rare exception). The client or device identifier is often based on a UUID, a persistent long-term unique identifier like an IMEI or MAC address, a sequence number assigned by the IM service domain, or a combination. In some cases the identifier may contain the internal user identifier. These identifiers look quite different across protocols and vendors.

Protocol	Identifier Address	Example
Jabber/ XMPP	Fully-qualified JID	juliet/balcony@example.com
SIP	Contact Address	sip:juliet@[2001:db8::225:96ff:fe12:3456]
Wire	Qualified client ID	0fd3e0dc-a2ff-4965-8873-509f0af0a75c: 072b@example.com

Table 2: some Client/Device identifier styles.

Group Chat or Channel identifier (external): All or nearly all instant messaging systems have the concept of named groups or channels which support more than 2 members and whose membership can change over time. Many IM systems support an external identifier for these groups and allows them to be addressed. In IRC and many other systems, they are identified with a "#" (hash-mark) prefix. The proliferation of hashtags on social media makes this convention less common on newer systems.

Group, Conversation, or Session identifiers (internal): Most IM protocols use an internal representation for a group or 1:1 chat. In MLS this is called the group_id. The Wire protocol uses the term qualified conversation ID to refer to a group internally across domains. Among implementations of the Double Ratchet family of protocols a unidirectional sequence of messages from one client to another is referred to as a session, and often has an associated session identifier.

Team or Workspace identifier: A less common type of identifier among IM systems is used to describe a set of users or accounts. This is described variously as a team, workspace, or tenant.

One user often has multiple clients (for example a mobile and a desktop client). A handle usually refers to a single user or rarely

it may redirect to multiple users. In some systems, the user identifier is a handle. In other systems the user identifier is an internal representation, for example a UUID. Handles may be changed/renamed, but hopefully internal user identifiers do not. Likewise, group conversation identifiers could be internal or external representations, whereas group names or channel names are often external friendly representations.

It is easy to imagine a loose hierarchy between these identifiers (domain to user to device), but hard to agree on a specific fixed structure. In some systems, the group chat or session itself has a position in the hierarchy underneath the domain, the user, or the device.

As described in the next section, the author proposes using URIs as a container for interoperable IM identifiers. All the examples use the im: URI scheme (defined in [RFC3862]), but any instant messaging scheme should be acceptable as long as the comparison and validation rules are clear.

3. Representation of identifiers using URIs

Most if not all of the identifiers described in the previous section could be represented as URIs. While individual instant messaging protocol-specific URI schemes may not have been specified with this use of URIs in mind, the im: URI scheme should be flexible enough to represent all of or any needed subset of the previously discussed identifiers.

For example, the XMPP protocol can represent a domain, a handle (bare JID), or a device (fully qualified JID). Unfortunately its xmpp: URI scheme was only designed to represent handles and domains, but the im: URI scheme can represent all XMPP identifiers:

* im:xmpp=example.com (domain only)
* im:xmpp=juliet@example.com (bare JID - handle)
* im:xmpp=juliet/balcony@example.com (fully qualified JID - client/device)

Likewise the IRC protocol can represent domain, handle (nick), user (account), and channel. The examples below represent a domain, a nick, a user, a local channel, abd three ways to specify the projectX channel.

* im:irc=irc.example.com
* im:irc=irc.example.com/juliet,isuser
* im:irc=irc.example.com/juliet%21jcapulet1583%40example.com,isuser
* im:irc=irc.example.com/%26local_announcements_channel
* im:irc=irc.example.com/#projectX
* im:irc=irc.example.com/%23projectX

```
*im:irc=irc.example.com/%23projectX
*im:irc=irc.example.com/%23projectX,ischannel
```

Imagine a hypothetical WXYZ IM protocol with support for all our identifiers. These could be represented unambiguously using the conventions below, or with an explicit parameter (ex: ;id-type=):

id type	unscoped form	domain scoped form
domain	-	example.com
handle	@alice	@alice@example.com
user	BFuVxW5BfJc8R7Qw	BFuVxW5BfJc8R7Qw@example.com
device	BFuVxW5BfJc8R7Qw/072b	BFuVxW5BfJc8R7Qw/072b@example.com
channel	#projectX	#projectX@example.com
team	##engineering	##engineering@example.com
channel	##engineering/projX	##engineering/projX@example.com
group id	$TII9t5viBrXiXc	$TII9t5viBrXiXc@example.com

Table 3: examples of all identifier types in the fictional WXYZ IM protocol

Now imagine that WXYZ reserved the wxyz: URI scheme. The example below shows how almost any reasonable protocol-specific identifier scheme can be represented as an im: URI.

```
wxyz:example.com
wxyz:%40alice@example.com
wxyz:BFuVxW5BfqaMEfJDc8R7Qw/072b@example.com
wxyz:#projectX@example.com
wxyz:##engineering@example.com
wxyz:$TII9t5viBrXiX@example.com

im:wxyz=example.com
im:wxyz=%40alice@example.com
im:wxyz=BFuVxW5BfqaMEfJDc8R7Qw/072b@example.com
im:wxyz=#projectX@example.com
im:wxyz=##engineering@example.com
im:wxyz=$TII9t5viBrXiX@example.com
```

Figure 1: mapping the identifiers in the fictional WXYZ format into an im: URI

Note that if there is no domain, an im: URI, or another scheme, could use local.invalid in place of a resolvable domain name.

```
im:wxyz=%40alice@local.invalid
```

4. Different Root of Trust Approaches

Different IM applications and different users of these applications may have different trust needs. The following subsections describe three specific trust models for example purposes. Note that the descriptions in this section use certificates in their examples, but nothing in this section should preclude using a different technology which provides similar assertions.

4.1. Centralized credential hierarchy

In this environment, end-user devices trust a centralized authority operating on behalf of their domain (for example, a Certificate Authority), that is trusted by all the other clients in that domain (and can be trusted by federated domains). The centralized authority could easily be associated with a traditional Identity Provider (IdP). This is a popular trust model for companies running services for their own employees and contractors. This is also popular with governments providing services to their employees and contractors or to residents or citizens for whom they provide services.

For example XYZ Corporation could make an assertion that "I represent XYZ Corporation and this user demonstrated she is Alice Smith of the Engineering department of XYZ Corporation."

In this model, a Certificate Authority (CA) run by or on behalf of the domain generates certificates for one or more of the identifier types described previously. The specifics of the assertions are very important for interoperability. Even within this centralized credential hierarchy model, there are at least three ways to make assertions about different types of IM identifiers with certificates:

Example 1 (Separate Certs): The CA generates one certificate for a user Alice which is used to sign Alice's profile. The CA also generates a separate certificate for Alice's desktop client and a third for her phone client. The private key in each client certificate is used to sign MLS KeyPackages or Double Ratchet-style prekeys.

Example 2 (Single Combined Cert): The CA generates a single certificate per client which covers both Alice's handle and her client identifier in the same certificate. The private key in each of these certificates is used to sign MLS KeyPackages or Double Ratchet-style prekeys. Note that there is no separate key pair used to refer to the user distinct from a device. All the legitimate device key pairs would be able to sign on behalf of the user.

Example 3 (Cascading Certs): The CA generates a single user certificate for Alice's handle and indicates that the user

certificate can issue its own certificates. The user certificate then generates one certificate for Alice's desktop client and another certificate for Alice's phone client. The private key in each client certificate is used to sign MLS KeyPackages or Double Ratchet-style prekeys.

What is important in all these examples is that other clients involved in a session or group chat can validate the relevant credentials of the other participants in the session or group chat. Clients would need to be able to configure the relevant trust roots and walk any hierarchy unambiguously.

When using certificates, this could include associating an Issuer URI in the issuerAltName with one of the URIs in the subjectAltName of another cert. Other mechanisms have analogous concepts.

Regardless of the specific implementation, this model features a strong hierarchy.

The advantage of this approach is to take advantage of a strong hierarchy which is already in use at an organization, especially if the organization is using an Identity Provider (IdP) for most of its services. Even if the IM system is compromised, the presence of client without the correct end-to-end identity would be detected immediately.

The disadvantage of this approach is that if the CA colludes with a malicious IM system or both are compromised, an attacker or malicious IM system can easily insert a rogue client which would be as trusted as a legitimate client.

4.2. Web of Trust

In some communities, it may be appropriate to make assertions about IM identity by relying on a web of trust. The following specific example of this general method is used by the OMEMO community presented by [Schaub] and proposed in [Matrix1756]. This document does not take any position on the specifics of the proposal, but uses it to illustrate a concrete implementation of a web of trust involving IM identifiers.

The example uses a web of trust with cross signing as follows:

* Each user (Alice and Bob) has a master key.
* Alice's master key signs exactly two keys:
  - Alice's device-signing key (which then signs her own device keys), and
  - Alice's user-signing key (which can sign the master key of other users).

The advantage of this approach is that if Alice's and Bob's keys, implementations, and devices are not compromised, there is no way the infrastructure can forge a key for Alice or Bob and insert an eavesdropper or active attacker. The disadvantages of this approach are that this requires Alice's device-signing key to be available any time Alice wants to add a new device, and Alice's user-signing key to be available anytime she wants to add a new user to her web of trust. This could either make those operations inconvenient and/or unnecessarily expose either or both of those keys.

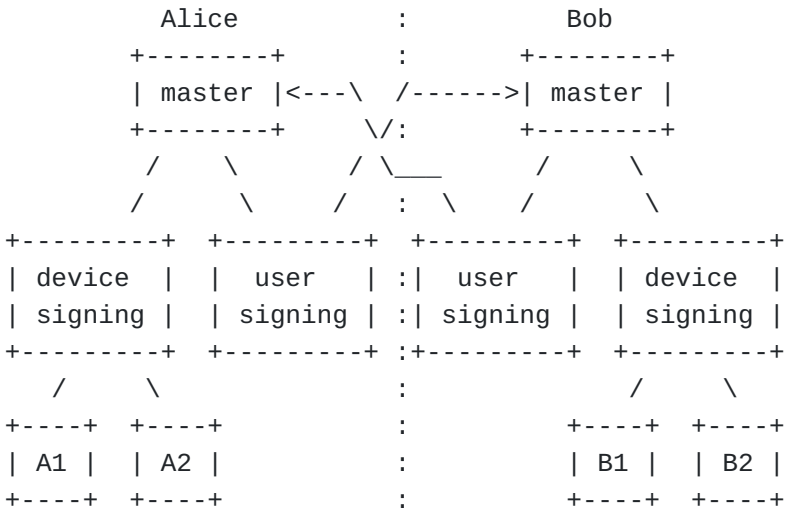

Figure 2: Alice and Bob cross sign each other's master keys

A detailed architecture for Web of Trust key infrastructure which is not specific to Instant Messaging systems is the Mathematical Mesh [I-D.hallambaker-mesh-architecture].

4.3. Well-known service cross signing

In this trust model, a user with several services places a cross signature for all their services at a well known location on each of those services (for example a personal web site .well-known page, an IM profile, the profile page on an open source code repository, a social media About page, a picture sharing service profile page, a professional interpersonal-networking site contact page, and a dating application profile). This concept was perhaps first implemented for non-technical users by Keybase. The user of this scheme likely expects that at any given moment there is a risk that one of these services is compromised or controlled by a malicious entity, but expects the likelihood of all or most of their services being compromised simultaneously is very low.

The advantage of this approach is that it does not rely on anyone but the user herself. This disadvantage is that if an attacker is

able to delete or forge cross signatures on a substantial number of the services, the forged assertions would looks as legitimate as the authentic assertions (or more convincing).

4.4. Combining approaches

These different trust approaches could be combined, however the verification rules become more complicated. Among other problems, implementers need to decide what happens if two different trust methods come to incompatible conclusions. For example, what should the application do if web of trust certificates indicate that a client or user should be trusted, but a centralized hierarchy indicates a client should not be, or vice versa.

5. Cryptographic mechanisms to make assertions about IM identifiers

5.1. X.509 Certificates

X.509 certificates are a mature technology for making assertions about identifiers. The supported assertions and identifier formats used in certificates are somewhat archaic, inflexible, and pedantic, but well understood. The semantics are always that an Issuer asserts that a Subject has control of a specific public key key pair. A handful of additional attributes can be added as X.509 certificate extensions, although adding new extensions is laborious and time consuming. In practice new extensions are only added to facilitate the internals of managing the lifetime, validity, and applicability of certificates. X.509 extensions are not appropriate for arbitrary assertions or claims about the Subject.

The Subject field contains a Distinguished Name, whose Common Name (CN) field can contain free form text. The subjectAltName can contain multiple other identifiers for the Subject with types such as a URI, email address, DNS domain names, or Distinguished Name. The rules about which combinations of extensions are valid are defined in the Internet certificate profile described in [RFC5280]. As noted in a previous section of this document, URIs are a natural container for holding instant messaging identifiers. Implementations need to be careful to insure that the correct semantics are applied to a URI, as they may be referring to different objects (ex: a handle versus a client identifier). There is a corresponding issuerAltName field as well.

Certificates are already supported in MLS as a standard credential type which can be included in MLS LeafNodes and KeyPackages. [In the X3DH key agreement protocol (used with Double Ratchet), the first message in a session between a pair of clients can contain an optional certificate, but this is not standardized.] Arguably the biggest drawback to using X.509 certificates is that

administratively it can be difficult to obtain certificates for
entities that can also generate certificates---specifically to issue
a certificate with the standard extension basicContraints=CA:TRUE.]

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            04:dc:7a:4b:89:22:98:32:35:1f:91:84:f7:e9:4e:5d:24:c4
        Signature Algorithm: ED25519
        Issuer: O = example.com, CN = acme.example.com
        Validity
            Not Before: Jul  6 06:41:50 2022 GMT
            Not After : Oct  4 06:41:49 2022 GMT
        Subject: O = example.com, CN = Alice M. Smith
        Subject Public Key Info:
            Public Key Algorithm: ED25519
                ED25519 Public-Key:
                pub:
                    a0:6b:14:1e:a8:04:2a:09:6b:62:89:48:7c:da:5c:
                    68:73:b9:2a:8e:65:50:f9:15:70:bd:91:d7:86:52:
                    1e:4f
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                4C:EA:12:32:79:03:F6:4F:47:29:37:5F:96:BB:E1:91:5E:FC
            X509v3 Authority Key Identifier:
                14:2E:B3:17:B7:58:56:CB:AE:50:09:40:E6:1F:AF:9D:8B:14
            Authority Information Access:
                OCSP - URI:http://oscp.acme.example.com
                CA Issuers - URI:http://acme.example.com/
            X509v3 Subject Alternative Name: critical
                URI:im:SvPfLlwBQi-6oddVRrkqpw/04c7@example.com,
                URI:im:%40alice.smith@example.com
            X509v3 Certificate Policies:
                [etc....]

    Signature Algorithm: ED25519
    Signature Value:
        da:21:49:cc:7a:ac:ed:7b:27:59:30:81:d9:94:c0:d7:86:e7:
        db:b2:c9:ed:72:47:19:01:aa:2a:7f:24:d6:ce:2f:4f:9d:fe:
        ab:8b:e2:0e:43:1b:62:b1:1d:12:3f:78:a2:bf:cc:7b:52:ef:
        df:c1:94:5a:3f:ca:a1:f6:88:02
```

Figure 3: mocked up IM client certificate with both client id and handle

If implementing cascading certificates, the Issuer might be a expressed as a URI in the issuerAltName extension.

TBC

Figure 4: mocked up IM client certificate issued by the domain for the handle URI as Subject. Then another certificate issued by the handle URI for the device URI as its Subject.

5.2. JSON Web Tokens (JWT) with Distributed Proof of Presence (DPoP)

JSON Web Signing (JWS) [RFC7515] and JSON Web Tokens (JWT) [RFC7519] are toolkits for making a variety of cryptographic claims. (CBOR Web Tokens [RFC8392] are semantically equivalent to JSON Web Tokens.) JWT is an appealing option for carrying IM identifiers and assertions, as the container type if flexible and the format is easy to implement. Unfortunately the semantics for validating identifiers are not as rigorously specified as for certificates at the time of this writing, and require additional specification work.

The JWT Distributed Proof of Possession (DPoP) specification [I-D.ietf-oauth-dpop] adds the ability to make claims which involve proof of possession of a (typically private) key, and to share those claims with third parties. The owner of a the key generates a proof which is used to fetch an access token which can then be verified by a third party. JWT DPoP was actually created as an improvement over Bearer tokens used for authentication, so its use as a certificate-like assertion may require substantial clarification and possibly additional profile work.

While there is support for token introspection, in general access tokens need online verification between resources and the token issuer.

While JWTs can include list of arbitrary claims, there is no native support for multiple subjects in the same JWT. There is a proposal to address this limitation with nested JWTs [I-D.yusef-oauth-nested-jwt].

```
{
    "typ": "dpop+jwt",
    "alg": "EdDSA",
    "jwk": {
         "typ": "OKP",
         "crv": "Ed25519",
         "x": "9kaYCj...3lnwW"
    }
}
.
{
    "jti": "7535d380-673e-4219-8410-b8df679c306e",
    "iat": 1653455836315,
    "htm": "POST",
    "htu": "https://example.com/client/token",
    "nonce": "WE88EvOBzbqGerznM-2P_AadVf7374y0cH19sDSZA2A",
    "sub": "im:SvPfLlwBQi-6oddVRrkqpw/04c7@example.com",
    "exp": 1661231836315
}
```

Figure 5: JOSE header and claims sections of a JWT DPoP proof referring to an IM URI

5.3. Verifiable Credentials

Verifiable Credentials (VC) is a framework for exchanging machine-readable credentials [W3C.REC-vc-data-model-20191119]. The framework is well specified and has a very flexbile assertion structure, which in addition to or in place of basic names and identifiers, can optionally include arbitrary attributes (ex: security clearance, age, nationality) up to and including Zero Knowledge Proofs depending on the profile being used. For example, a verifiable credential could be used to assert that an IM client belongs to a Customer Support agent of Sirius Cybernetic Corp, who speaks English and Vogon, and is qualified to give support for their Ident-I-Eeze product, without revealing the name of the agent.

The VC specification describes both Verifiable Credentials and Verifiable Presentations. A Verifiable Credential contains assertions made by an issuer. Holders assemble credentials into a Verifiable Presentation. Verifiers can validate the Verifiable Credentials in the Verifiable Presentation. Specific credential types are defined by referencing ontologies. The example at the end of this section uses the VCard ontology [W3C.WD-vcard-rdf-20130924].

Most of the examples for Verifiable Credentials use Decentralized Identifiers (DIDs), but there is no requirement to use DID or the associated esoteric cryptography in a specific VC profile. (Indeed the VC profile for COVID-19 for vaccination does not use DIDs). The

most significant problem with VCs are that there is no off-the-shelf mechanism for proof of possession of a private key, and no consensus to use VCs for straightforward identity assertions.


```
{
  "sub": "im:SvPfLlwBQi-6oddVRrkqpw/04c7@example.com",
  "jti": "http://im.example.com/@alice_smith/devices/04c7",
  "iss": "https://im.example.com/keys/issuer.jwk",
  "nbf": 1653455836315,
  "iat": 1653455836315,
  "exp": 1661231836315,
  "nonce": "WE88EvOBzbqGerznM-2P_AadVf7374y0cH19sDSZA2A",
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "http://www.w3.org/2006/vcard/ns"
    ],
    "type": ["VerifiableCredential", "ImDeviceCredential"],
    "credentialSubject": {
      "fn": "Alice M. Smith",
      "hasOrganizationName": "Example Corp",
      "hasOrganizationalUnit": "Engineering",
      "hasInstantMessage": "im:%40alice_smith@example.com",
      "hasInstantMessage": "im:SvPfLlwBQi-6oddVRrkqpw/04c7@example.com"
    }
  }
}
```


Figure 6: fragment of example claims payload of JWT-based VC proof referencing the VCard ontology (WIP)

5.4. Other possible mechanisms

Below are other mechanisms which were not investigated due to a lack of time.

- *Anonymous credential schemes which can present attributes without the long-term identity (ex: travel agent for specific team)
- *Zero-knowledge proofs
- *Deniable credentials

6. IANA Considerations

This document requires no action by IANA.

7. Security Considerations

TBC. (The threat model for interoperable IM systems depends on many subtle details).

8. Normative References

[I-D.ietf-oauth-dpop] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP)", Work in Progress, Internet-Draft, draft-ietf-oauth-dpop-09, 2 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-dpop-09>.

[RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, DOI 10.17487/RFC3862, August 2004, <https://www.rfc-editor.org/info/rfc3862>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[W3C.REC-vc-data-model-20191119] Sporny, M., Noble, G., Longley, D., Burnett, D., and B. Zundel, "Verifiable Credentials Data Model 1.0", World Wide Web Consortium Recommendation REC-vc-data-model-20191119, 19 November 2019, <https://www.w3.org/TR/2019/REC-vc-data-model-20191119>.

[W3C.WD-vcard-rdf-20130924] Iannella, R. and J. McKinney, "vCard Ontology", World Wide Web Consortium WD WD-vcard-rdf-20130924, 24 September 2013, <http://www.w3.org/TR/2013/WD-vcard-rdf-20130924>.

9. Informative References

[DoubleRatchet] Perrin, T. and M. Marlinspike, "The Double Ratchet Algorithm", 20 November 2016, <https://signal.org/docs/specifications/doubleratchet/>.

[I-D.hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part I: Architecture Guide", Work in Progress, Internet-Draft, draft-hallambaker-mesh-architecture-20, 20 April 2022,

<https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-architecture-20>.

[I-D.ietf-mls-architecture]
Beurdouche, B., Rescorla, E., Omara, E., Inguva, S., Kwon, A., and A. Duric, "The Messaging Layer Security (MLS) Architecture", Work in Progress, Internet-Draft, draft-ietf-mls-architecture-08, 16 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-architecture-08>.

[I-D.ietf-mls-protocol]
Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", Work in Progress, Internet-Draft, draft-ietf-mls-protocol-16, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-16>.

[I-D.mahy-mimi-problem-outline]
Mahy, R., "More Instant Messaging Interoperability (MIMI) problem outline", Work in Progress, Internet-Draft, draft-mahy-mimi-problem-outline-00, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-mahy-mimi-problem-outline-00>.

[I-D.yusef-oauth-nested-jwt]
Shekh-Yusef, R., "Multi-Subject JSON Web Token (JWT)", Work in Progress, Internet-Draft, draft-yusef-oauth-nested-jwt-05, 14 June 2022, <https://datatracker.ietf.org/doc/html/draft-yusef-oauth-nested-jwt-05>.

[Matrix1756] Chathi, H., "Cross-signing devices with device signing keys", 13 December 2018, <https://github.com/matrix-org/matrix-doc/blob/master/proposals/1756-cross-signing.md>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[Schaub] Schaub, P., "Cryptographic Identity: Conquering the Fingerprint Chaos (video)", 6 April 2021, <https://www.youtube.com/watch?v=oc5844dyrsc>.

[X3DH] Marlinspike, M. and T. Perrin, "The X3DH Key Agreement Protocol", 4 November 2016, <https://signal.org/docs/specifications/x3dh/>.

Appendix A. Acknowledgments

The author wishes to thank Richard Barnes, Tom Leavy, Joel Alwen, Marta Mularczyk, and Rifaat Shekh-Yusef for discussions about this topic.

Author's Address

Rohan Mahy
Wire

Email: rohan.mahy@wire.com