More Instant Messaging Interoperability (mimi)
Internet-Draft
Intended status: Informational
Expires: 7 May 2024
4 November 2023

G. Hogben
F. Olumofin
Google

Interoperable Private Identity Discovery for E2EE Messaging

draft-party-mimi-user-private-discovery-03

Abstract

This document specifies how users can privately discover each other's Service Specific Identifiers (SSIs) when using end-to-end encrypted messaging services across multiple providers. Users can retrieve SSIs without revealing their social graphs to service providers they are not delivering messages through, using their phone numbers, email, user IDs, or other Service Independent Identifiers (SIIs). Our specification can be based on private information retrieval or associative private sets membership schemes, both of which provide reasonable tradeoffs between privacy and cost.

About This Document

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

The latest revision of this draft can be found at https://datatracker.ietf.org/doc/giles-interop-user-private-discovery/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-party-mimi-user-private-discovery/.

Discussion of this document takes place on the mimi 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/femigolu/giles-interop-user-private-discovery.

Status of This Memo

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 7 May 2024.

Copyright Notice

Table of Contents

1. 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.

A *service specific identifier* (SSI) is a unique identifier for a user within a single service provider's service, and encodes the service provider in the identifier. For example, a user's account handle and provider identifier is an SSI.

A *service independent identifier* (SII) is a unique identifier for a user that is independent of any specific service provider. For example, a user's E.164 phone number or email address are SIIs, since they can be used to identify the user across multiple different services.

2. Problem statement

The *discovery problem* is resolving a user's SII into one SSI for that user, while preserving user privacy in the process.

3. Threat actors

* Alice, Bob, and Carol: Three users within the interoperable E2EE messaging ecosystem.

* Sender Messaging Platform: A messaging service provider platform where a registered user has an account and has established a mapping of SII to SSI. Examples from Fig. is Platform 1 for Alice and Carol, and Platform 2 for Bob.

* Potential Recipient Messaging Platform: A messaging service provider platform where a discovered SSI is registered. An example from Fig. 1 is the role of Platform 2 when Alice resolves Bob's SSI using Bob's SII. This has three variants in the threat model:

    1. Recipient platform with SSI - the sender sends a message (so this platform will learn the sender identity).

    2. Non-recipient platform with SSI that the recipient SII has an account with but does not send a message to.

3. Non-recipient platform without SSI - potential recipient does not have an SSI registered with this platform.

```
,-----.  ,-----.   ,---------.          ,-------------------.          ,--------
-.
|Alice|  |Carol|   |Front End|          |Discovery Provider|          |Front En
d|
|-----|  |-----|   |---------|          |-------------------|          |--------
-|
|-----|--|-----|   |---------|          |-------------------|          |--------
-|
`-----'  `-----'   `---------'          `-------------------'          `--------
-'
   |                    |                          |                          |
   |                    |                          |                          |
 ,---.          ,-------------------.   ,-------------------------.   ,-------------
-----.
 |Bob|          |Discovery Provider|    |Key Distribution Service|   |Discovery Pro
vider|
 |---|          |-------------------|   |-------------------------|   |-------------
-----|
 |---|          |-------------------|   |-------------------------|   |-------------
-----|
 `---'          `-------------------'   `-------------------------'   `-------------
-----'
                        |                          |                          |
            ,-------------------------.   ,--------------------.  ,-----------------
--------.
            |Key Distribution Service|    |Mappings DB Bob,...|   |Key Distribution
 Service|
            |-------------------------|   |--------------------|  |-----------------
--------|
            |-------------------------|   |--------------------|  |-----------------
--------|
            `-------------------------'   `--------------------'  `-----------------
--------'
                        |                                                     |

          ,----------------------------.                          ,-------------
------.
          |Mappings DB Alice,Carol,...|                           |Mappings DB B
ob,...|
          |----------------------------|                          |-------------
------|
          |----------------------------|                          |-------------
------|
          `----------------------------'                          `-------------
------'
```

Figure 1: Threat actors and systems

* Third Party Platform: A platform that provides discovery services but is not a messaging service provider. Bob might register with such a service directly, or such a service may act as a proxy for Messaging Platform 2 through contractual business agreement.

* Front End: A service within a platform that receives users' requests and collaborates with other services to process them.

* Discovery Provider: Works to resolve SII to SSI.

* Key Distribution Service: Manages public key material of registered users.

4. Privacy requirements

1. *Social graph*: Discovery service providers should not learn the SII or SSI a user is querying for unless they are sending or receiving a message on to that user.

2. *Querying user identity*: A discovery service provider should not share the querying user identity with other discovery services when it requires their help for discovery.

3. *Metadata*: Discovery service should not learn the exact timing of when a message is sent (after discovery).

4.1. Requirements by threat actor

The following table describes the requirements to protect the privacy of an intended recipient’s SSI during discovery broken down by the various threat actors. The possible list of services that may resolve a discovery request based on their knowledge of the SSI is shown in the first column. The second and third columns are the minimum and possible privacy requirements. The optimal privacy requirements assume that the two devices in E2EE messaging endpoints are on different messaging service platforms.

Note that current messaging systems segment a user’s social graph across their contacts’ messaging services. Without proper privacy mitigations, a discovery process for the new interoperable ecosystem can enable an attacker to aggregate these fragments of the user’s social graph across different services, violating their privacy. Performing the discovery process for contacts that are never used is common so that it is very likely that most clients will perform discovery for SIIs that they never send a message to. This is why we propose hiding the SII from the sender platform unless a message is sent. We believe this is possible technically because:

1. Spam prevention requirements only apply to sent messages (standard IP based techniques can be used to prevent DDoS of the discovery service itself).

2. Client costs for SII hiding mechanisms scale well enough with database size + number of services.

Service	Minimum privacy requirements	Optimal privacy requirements
Sender Platform	Do not hide SSI	Hide SSI
Recipient Platform with SSI	Do not hide SSI	Do not hide SSI
Non-recipient Platform with SSI	Hide SSI	Hide SSI
Non-recipient Platform without SSI	Hide SSI	Hide SSI
Third party service	Hide SSI	Hide SSI

Table 1

Table 1: Discovery privacy requirements by threat actors

5. Privacy non-requirements

1. *Hiding SII <> service mapping*: Hiding service reachability or the existence of a mapping between an SII and SSI for a service provider is an explicit non-goal. All major E2EE messaging services already publish unACLd reachability information without opt-out i.e. +16501234567, reachable on Messages, Whatsapp, Telegram (not including name or any other info). Therefore this should not be a privacy goal (and would not be feasible to implement). *However it may be a business goal to prevent scraping of the full list of account-holders.*

2. *Contact lookup by name* or anything except an SII.

6. Other Non-functional Requirements

1. No single entity should be financially responsible for resolving all discovery queries (e.g. even within a geographical region).

2. Costs for each participating entity of storing and resolving SII should be proportional to their number of participating users.

3. Performance should support each client device resolving users' contact SIIs at least once every 24 hours.

7. SSI Discovery

SSI discovery means retrieving the SSI that an SII maps to. There are two alternative cryptographic techniques to achieve the privacy properties for the retrieval:

1. Private Information Retrieval (PIR)

2. Private Set Membership (PSM)

The discovery process is illustrated in Figure 2. Optionally, Alices client may encrypt the SSI of interest using PIR or PSM before forwarding the SII query to the Discovery Provider of the Sender Messaging Platform.

The DP for the Sender Messaging Platform may either look up or compute an encrypted response directly, or it may forward the request to the Potential Recipient or Third Party Discovery Provider indicated by the provider identifier included in the request. Regardless of which party processes the request, a DP will compute an encrypted response and forward it back to Alice. Alice can then decrypt the encrypted response (if applicable) to obtain the SSI.

Alices client may also optionally send the discovery request directly to a potential recipient or 3p DPs.

We assume a fixed list of DPs for each SMP so that the client does not have to specify in the query request which DPs to use.

```
                         Sender Messaging Platform         Potential Recipient o
r Third
     Alice                    Discovery Provider                Party Discovery
Provider

             0. resolve SII

        <

              1. SII | Encrypted(SII)

        >

                                 1b. SII | Encrypted(SII)

                                             >


                                                 2. Lookup | Compute Response

                                            <

                                                   3. SII | Encrypted(SII)

                                                               >




     | 4. Lookup | Compute Response

 <

                                                   5. SSI | Encrypted(SSI)

                                            <

              6. SSI | Encrypted(SSI)

        <


             7. SSI | Decrypt Response =>SSI

        <

     Alice                    Sender Messaging Platform         Potential Recipi
ent or Third
```

                         Discovery Provider   Party Discovery Provi
der

Figure 2: Discovery with Sender Messaging Platform

Note: * Note that the DPs should not learn that Alice is the author of the request. * Alice is not required to hide discovery requests when the processor DP is within the Sender Messaging Platform. * Alices client may, but is not required to hide discovery requests from Potential Recipient DPs. Both of these requests can be sent in the clear.

7.1. Private Information Retrieval (PIR)

A PIR protocol enables a client holding an index (or keyword) to retrieve the database record corresponding to that index from a remote server. PIR schemes have communication complexities sublinear in the database size and they provide access privacy for clients which precludes the server from being able to learn any information about either the query index or the record retrieved. A standard single-server PIR scheme provides clients with algorithms to generate a query and decode a response from the server. It also provides an algorithm for the server to compute a response.

We proposed a lattice-based PIR framework by Patel et al[PIRFramework] with sharded databases. This framework is applicable with any standard PIR scheme such as the open source implementation here (https://github.com/google/private-retrieval). Cost estimates suggest this is feasible even for a very large database with 10 billion records/mappings.

7.1.1. Cost estimates

Use database shards each of ~1 million mappings. For 1.28 TB (10 billion records), breaking this down into 10,000 shards each of size 1 million records gives a cost estimate for each query as below:

Parameter/Metric	Cost estimate
Server Storage Per Device	14 MB
Client Device Storage (for 10 billion records)	5 MB
Upload Bandwidth Per Query	14 KB
Download Bandwidth Per Query	21 KB
Client Time Per Query	0.1s
Server Time Per Query (Single Thread)	0.8-1s

Table 2

7.2. Private Set Membership (PSM)

The discovery provider holds a set of SIIs that maps to an associated set of SSI. A PSM protocol enables a client with an SII to lean the associated SSI held by the server with the following privacy guarantees:

1. The discovery provider does not learn the SII held by the client.

2. The discovery provider does not learn whether a matching SII was found or not.

3. The client does not learn any information about the other SIIs and associated SSIs held by the discovery provider.

An open source implementation is available here (https://github.com/google/private-membership).

7.2.1. Cost estimates

For a database with 1.28 TB (10 billion associated records of SSI), using 1,000 shards each of size 10 million records, the cost estimate for each query is:

Parameter/Metric	Cost estimate
Communication	2.8 MB
Client Time Per Query	0.1s
Server Time Per Query (Single Thread)	1-2s

Table 3

7.3. Cross-service identity spoofing

Today, a messaging service may support one or more ways of identifying a user including email address, phone number, or service specific user name.

Messaging interoperability introduces a new problem that traditionally has been resolvable at the service level: cross-service identity spoofing, where a user on a given E2EE may or may not be addressable at the same ID on another service due to a lack of global uniqueness constraints across providers.

As a result, a user may be registered at multiple services with the same handles, e.g. if Bob’s email is bob@example.com (mailto:bob@example.com) and his phone number is 555-111-2222 and he is registered with Signal and iMessage, he would be addressable at bob@example.com (mailto:bob@example.com):iMessage, 555-111-2222:iMessage, and 555-111-2222:Signal. In this case, the same userId on iMessage and Signal is acceptable as the phone number can map to only one individual who proves their identity by validating ownership of the SIM card.

On services where a user can log in with a username _alone_, however e.g. Threema and FooService, the challenge becomes:

* Alice messages Bob at Bob’s preferred service (bob@Threema)

* Eve messages Alice impersonating Bob using bob@FooService

* Alice needs some indicator or UI to know that bob@Threema isn’t bob@FooSercice and that when bob@FooService messages, it should not be assumed that bob@FooService is bob@Threema.

Options for solving this are: 1. Storing the supported services for a contact in Contacts and if a recipient receives a message from an unknown sender, to treat it as spam or otherwise untrusted from the start. 2. Requiring the fully qualified username for services that rely on usernames only - e.g. bob@threema.com vs bob.

8. Thoughts on open questions from 10/10/2023 Interim Meeting[MIMI20231010]

8.1. Trusted Authorities for Mapping SIIs to SSIs

Which actors should be trusted authorities for mapping SIIs to SSIs?

In general, this should be considered out of scope for this proposal, however we expect that by default, Messaging Service Providers (MSP) should be trusted authorities for creating these mapping. Users may "own" their SIIs, but messaging service providers own SSIs. MSP should verify ownership of SIIs (one time password code to phone via text or call, or to email).

An MSP may share established mapping data with 3P discovery providers to facilitate lookups, or may delegate establishing new mappings to these providers under contractual agreements between them. Preferably, delegate discovery providers should be lookup providers only and should not create or update existing mappings unless the delegate is a reputable/trusted certification authority.

If a 3p discovery service is used, it may also authenticate the mapping independently or it may act as a pass-through for a signed mapping by an MSP or another identity provider.

SSL is sufficient to authenticate the mapping assertion.

8.2. Discovery Scaling

Does discovery need to scale to accommodate 10s, 100s, or 1000s of service?

A discovery request should be sent to a specific MSP or 3P discovery provider. It is up to those providers if they want to fan out the discovery to other providers or answer the discovery request from its own mapping only. It will be costly to fork out discovery requests to a large number of discovery providers while completely hiding the SSI from these providers. We do not want forking to fit DDoS patterns on these services.

However the protocols should be feasible (in terms of computation and communication cost) for 1000s of services.

8.3. Acceptable leakage for discovery

What is it acceptable for queries to reveal about the social graph, and to whom?

A query should not reveal the SII in a user's query to discovery providers unless the discovery provider is also within the Sender's platform or the Recipient's platform with the SSI mapping. For an encrypted query and *since discovery precedes E2EE messaging*, a discovery provider won't be able to tell if the SSI maps to an SSI in its service. It is okay to take the no-leakage approach for all providers.

Alice may use the different provider owning each SSI that her phone maps to. Bob may use different email addresses to map to multiple SSI with the same provider.

Returning an SSI set of different cardinalities leaks information to a discovery provider about the likely sets of SSIs that are of interest for a query. A one-to-one mapping of SII to SSI does not leak such information. A discovery provider cannot tell when a privacy-preserving discovery returns an empty result or a single SII. However, it will be able to tell when a large number of SSIs are returned.

8.4. Rate Limiting

Is rate limiting useful to prevent scraping?

It is up to a discovery provider to rate-limit given the potential computational cost of responding to batch queries from a single user. Nonetheless, we should require that a user should be able to look up no less than 50 SII per discovery provider for each messaging provider in a given 24 hours period. Third party discovery providers are under obligation to messaging service providers and are excluded from the minimum discovery load per user.

8.5. SII Mappings

An SII may map to multiple SSIs. Should the requestor learn all of them, and if so, how?

* _One service that returns all SSIs for an SII?_

* _Query each service provider independently?_

* _User figures out out-of-band what service provider to query?_

SII mapping to multiple SSIs within a single provider

1. This is a choice that MSPs will have to make, if they want to allow it.

2. Having multiple SSIs per SII makes preserving the privacy of discovery more challenging because of the side channel leakage of response size. The tradeoff is acceptable if on the average users have multiple SSI with a MSP.

3. For privacy reasons (i.e., protecting the association of multiple SSIs), the user may not want to group multiple SSIs together.

4. We may devise a scheme where an SII could be suffixed with an index during registration and discovery of the SSI to retrieve from the set. For example, given an SII +1234567890, a user may map +12345678900 to the first Whatsapp SSI, and +1234567891 to the second Whatsapp SSI and so on.

The user should figure out out-of-band what discovery provider to query, and discovery providers should not be required to fork out discovery requests to other providers given the computational cost impact.

8.6. Notes

9. IANA Considerations

This document has no IANA actions.

9.1. Appendix

10. Normative References

[MIMI20231010]
Geoghegan, T., "Discovery requirements", MIMI Virtual interim October 10, 2023 , n.d..

[PIRFramework]
Patel, S., Seo, J. Y., and K. Yeo, "Don’t be Dense: Efficient Keyword PIR for Sparse Databases", 32nd USENIX Security Symposium, USENIX Security 2023 , n.d..

[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>.

Acknowledgments

The technical description of the private information retrieval framework is based on Sarvar Patel, Joon Young Seo and Kevin Yeo’s USENIX Security ’23 paper titled "Don’t be Dense: Efficient Keyword PIR for Sparse Databases " (https://www.usenix.org/conference/usenixsecurity23/presentation/patel).

Authors’ Addresses

Giles Hogben
Google
Email: gih@google.com

Femi Olumofin
Google
Email: fgolu@google.com