A Private Discovery Protocol for Anonymity Systems (Version 2)

Ceren Kocaoğullar

February 2021

1 Introduction

This document proposes an updated private discovery protocol for anonymity systems using non-colluding servers, which obliviously perform lookups. Our new approach differs vastly from the previous idea, which used an approach similar to Oblivious DNS (ODNS) [1], required the ID to be unique to our system, and limited the updates to user information. The new approach (1) eliminates the need for query servers, and (2) *requires* the user to use an ID valid outside of the system, such as an email address or phone number. The first change is important, because it strengthens our threat model. In the previous design, an attacker having access to the query server and any discovery server would be able to link the searcher and searchee. This is not the case in the current design. The second change improves usability, since users can discover each other through their already-known IDs. In addition, it allows us to have two-factor authentication.

We still use the secret sharing idea with a (k, n) threshold scheme where network locations and public keys of the users are the secrets. Moreover, we again propose to use dummy queries to create cover traffic as an additional privacy measure against passive de-anonymisation attacks.

The main changes to the system are: (1) We use discovery messages similar to Mixminion *forward messages* [2], where the sender remains anonymous by determining the whole path of a message. (2) We use IDs to perform two-factor authentication for both the system to authenticate registering users, and the searchees to authenticate searchers. (3) Discovery servers respond with fake information even though they do not have any record of searchee, so that if a malicious user queries the system with a large list of users, they cannot effectively figure out if they exist in the network or not.

2 Summary of Notation & Terms

ID: User ID

- This is the ID that the users use to look others up.
- Has to be unique to each user.
- Preferably easy to say, read, and remember.
- Does not have to be kept secret.
- *Must* be an identifier that allows one to reach the owner of the ID through other means, such as an email address or a phone number.

Loc: Network Location

- Can include different types of information depending on a network's architecture, e.g. IP address or any other type of client ID that is uses to identify clients within the network, additional identifier of intermediary nodes (such as provider IP in Loopix), etc.
- Does not need to be easy to say, read, or remember.

D**: Discovery Node**

- These nodes are connected to each other and the users.
- Their network locations and public keys are provided to all users as they join the network.
- Each discovery node holds a collection of key - value pairs, where keys are IDs, and values are pieces of (Loc, pk) data of the users, which make no sense unless k or more of them are combined.
- These nodes also function as authentication servers. They are capable of carrying out two-factor authentication to verify IDs. Moreover, the users register to the system by providing a password, which allows them to update their ID and Loc when necessary.

Searcher : The user who is looking up another user through our scheme.

Searchee : The user who is being looked up through our scheme.

3 Security Goals

Relationship unobservability: We aim to achieve unlinkability in terms of who is searching for whom. To specify our goal, we use the term *relationship unobservability* defined by Pfitzmann and Hansen as follows: "Relationship unobservability ... means that it is sufficiently undetectable whether anything is sent out of a set of could-be senders to a set of could-be recipients. In other words, it is sufficiently undetectable whether within the relationship unobservability set of all possible sender-recipient(s)-pairs, a message is sent in any relationship" [3].

Network location (Loc) privacy: Under specific non-collusion assumptions described in Section 4, an adversary having control over corrupted servers involved in lookups should not be able to learn user network locations.

Sender online unobservability [4]: Whether or not a user is searching for someone through our scheme should be indistinguishable to unauthorised third parties.

Authentication :When a user joins the network and registers to the lookup system, we should verify their identity. Moreover, as searcher obtains a searchee's network location and private key, we should make sure that this information is correct. Also, after the lookup, once the searcher initiates communication, the searchee should be able to authenticate them.

4 Assumptions & Threat Model

Assumptions

To begin with, as our protocol assumes that it runs on top of an anonymity system such as Tor [5] or Loopix [4], we assume that all communications are encrypted. Some messages in the mechanism require a special packet structure described in Section 5.1. The rest use messages regular to the system.

Another important assumption is a ***non-collusion assumption***. We assume that n or more discovery nodes do not collude. This ensures *network location privacy* since we employ a (k, n) secret sharing scheme among discovery nodes.

Threat Model

For our threat model, we assume adversaries interested in linking users with lookup queries as well as learning network locations. These adversaries have two different capabilities: (1) They can have the capacity of a global passive adversary (GPA), i.e. they can observe all traffic between users, discovery nodes, and query

nodes. (2) They can own corrupt discovery or query nodes, as long as this meets our aforementioned non-collusion assumptions. This capability includes observing and interfering with the inside state, operations, and communications of the corrupted node.

We are aware that as long as the underlying network does not promise immunity against GPAs, e.g. Tor, our system's promise is not sufficient for protecting the whole communication against passive attacks from attackers with such capabilities. However, even in that case, providing such protection in only discovery stage might still be useful. For instance, it prevents the attackers from determining how many people are in a user's address book.

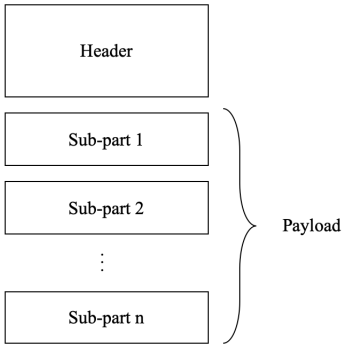
We do not assume that all users are benign. This might raise a question about why network locations are kept secret from the discovery servers, as users are allowed to search for any user's location. However, disabling access to massive amounts of user data at once seems like a reasonable goal, especially given that the searchees are able to authenticate searcher identities and can choose not to respond/drop packets. Moreover, blocking strategies at discovery nodes can be employed for suspicious queries.

5 System Design

5.1 Packet Structure

The proposed system requires a special type of data packet, consisting of a header and a payload (Figure

Figure 1: The discovery message packet structure querying n nodes.



1). The header is similar to that of Mixminion's [2], specifically its *forward messages*': Structurally, the header is not very different from classical onion routing headers: It contains layers of encryption, each containing information about the next hop. The main difference from the regular onion routing headers is that *the sender decides on the whole message path.* In addition, hashes are used for integrity checking, but we will not go into the details of that at this stage.

The payload is made up of individual sub-parts, each of which is encyrpted with a different key. Padding is used for making sure that all sub-parts are of equal length. It is important to note that the number of payloads remain unchanged through a packet's journey, so that a malicious node or an eavesdropper cannot figure out at which stage of its journey the packet is.

5.2 Registration to the System

This section of our scheme prepares some steps required for achieving network location privacy and authentication. Regarding the first one, the user divides the secret (network locations and public keys of the users are the secrets in our case) into pieces before registering to the discovery servers.

In terms of authentication, the registration step uses two methods: Two-factor authentication using ID, and password authentication between the discovery server and user for enabling authenticated updates to user data. The registration steps are described below.

1. The user performs cryptographic operations (e.g. as in [6]) to divide their (Loc, pk) information into n pieces.

2. The user picks a discovery node at random. They then send the selected node a *registration initiation message*, which is a regular message, i.e. it does not have the structure described in Section 5.1, but the existing message structure of the underlying network. This packet includes only the ID of the user.

3. As the selected discovery node receives the *registration initiation message*, it performs two-factor authentication to verify ID.

4. Once the authentication is complete, the user picks n random discovery nodes and a password. They then send a *registration message* to the initially selected discovery node. This message *has* the packet structure described in Section 5.1. Each of the n sub-parts of the payload consist of (1) a piece of the secret, (2) ID, (3) the password that the user selected. Every sub-part is encrypted with a different D's pk.

$$\{secret_1, ID, password\}_{pk_{D_1}}, \{secret_2, ID, password\}_{pk_{D_2}}, \ldots, \{secret_n, ID, password\}_{pk_{D_n}}$$

5. Upon receiving the *registration message*, the initially selected discovery node decrypts its sub-part, registers the user in its database, peels a layer of encryption from the header to reveal the next discovery node and relays the message to them.

5.3 Searching for a User

In this section, we will describe the steps for looking up a user through our scheme. We first explain the main lookup mechanism, followed by explanations of responding in the absence of an answer, searcher authentication by searchee, and cover traffic.

5.3.1 The Lookup Mechanism

Steps of the lookup protocol in a scenario where Alice is the searcher and Bob is the searchee are described below.

1. Alice randomly picks k discovery servers. She also produces k symmetric keys sym_i for each discovery server D_i. Alice then determines the message path for the *discovery message*, including the path for reaching the initial discovery server, the path among the n selected servers, and the path returning to her.

2. She creates and sends out a discovery message, structured as in Section 5.1. Each sub-part of the payload consists of Bob's ID and sym_i, encrypted with pk_{D_i}.

$$\{ID_B, sym_1\}_{pk_{D_1}}, \{ID_B, sym_2\}_{pk_{D_2}}, \ldots, \{ID_B, sym_k\}_{pk_{D_k}}$$

3. Upon receiving the message, each discovery server decrypts the sub-part encrypted with their pk, and respond with $secret_i$, their own piece of secret, encrypted with the symmetric key they have received. After leaving the last discovery node on the path, the payload looks as below:

$$\{ID_B, secret_1\}_{sym_1}, \{ID_B, secret_2\}_{sym_2}, \ldots, \{ID_B, secret_n\}_{sym_n}$$

   If a discovery server does not have a record matching ID_B for any reason, it responds with a fake piece of secret. The details about this are explained in Section 5.3.2.

4. Alice receives the message, decrypts all pieces of information, and combines them to reveal Loc_B and pk_B.

5.3.2 Responding In the Absence of an Answer

Our system requires the discovery nodes to respond with fake information even though they do not have a record of the searchee. As we employ a threshold scheme, this requirement introduces the problem that when the pieces are put together, they must yield a meaningful (Loc, pk) pair. Leaving the responsibility of creating fake secret pieces to one discovery server would make our job easy. However, each secret piece in the discovery response message must be encrypted with a symmetric key, each of which can only be revealed by the intended discovery server. Another difficulty is that the fake responses for an ID should be consistent, so that a user cannot figure out if the information is fake or real.

To solve this problem, we need a function which takes ID as an input and yield a realistic dummy (Loc, pk) pair. All discovery servers will know this function. The function can be updated at random periods to imitate users updating their information. Even if malicious discovery servers disclose the function to accomplice user nodes, this does not pose an additional privacy risk, since the discovery servers already know which users are existent in the system.

5.3.3 Searcher Authentication by Searchee

We want to allow searchees verify searcher identity upon initial contact. This can simply be achieved by searchee performing authentication by sending a message encrypted with searcher's pk. This message will be sent to the searcher through communication channels outside of our network, where ID can be used to reach out to them, e.g. email, SMS, etc.

5.3.4 Cover Traffic

Using dummy messages to provide a layer of protection against de-anonymisation attacks have been explored in Loopix by using *cover messages* [4] and in Talek by using *FakeWrites* [7]. In our scheme, we adopt the same idea to protect against violation of relationship unobservability through passive attacks such as traffic analysis attacks.

The dummy messages in our scheme look like regular discovery messages. They may ask for existing or non-existing users, sent with a fixed or dynamic frequency. Advantages and disadvantages of different approaches in terms of network communication costs, computational costs, and privacy should be investigated in the implementation and evaluation phase of our project.

References

[1] Paul Schmitt et al. "Oblivious DNS: Practical Privacy for DNS Queries". In: *arXiv* (2018), pp. 17–19. DOI: 10.2478/popets-2019-0028. arXiv: 1806.00276.

[2] G. Danezis, R. Dingledine, and N. Mathewson. "Mixminion: Design of a type III anonymous remailer protocol". In: *Proceedings - IEEE Symposium on Security and Privacy* 2003-January (2003), pp. 2–15. ISSN: 10816011. DOI: 10.1109/SECPRI.2003.1199323.

[3] Andreas Pfitzmann and Marit Hansen. "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management". In: (2010).

[4] Ania M Piotrowska et al. "The Loopix Anonymity System This paper is included in the Proceedings of the The Loopix Anonymity System". In: *USENIX Security* (2017).

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004.

[6] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[7] Raymond Cheng et al. *Talek: Private Group Messaging with Hidden Access Patterns*. Vol. 1. 1. Association for Computing Machinery, 2020, pp. 84–99. ISBN: 9781450388580. DOI: 10.1145/3427228.3427231. arXiv: 2001.08250.