A Protocol for Anonymous and Private Federated Search (DRAFT)

Lee Pike
Galois, Inc.
leepike@galois.com

ABSTRACT

A *federated search* is a data query over segregated data sources. We present a novel but simple protocol for anonymous and private federated search. We also provide a mathematical model for reasoning about the protocol's anonymity guarantees and prove they hold. This is the first treatment of anonymity in federated search.

1. INTRODUCTION

A *federated search*[1] is a data query over segregated data sources [7]. This paper presents a solution to the problem of protecting the anonymity of a client making a federated search query while ensuring the client also has the access credentials necessary to make her searches. The protocol we propose is data-agnostic and only addresses the anonymity and privacy aspects of carrying out a data query; we do not address issues such as how to aggregating or rank the relevance of data from disparate sources, for example. In our model, we assume there is a set of clients making queries and a set of servers containing data (e.g., databases, webpages, etc.) that respond to the queries.

Some potential use-cases for a protocol solving this problem include

- *Humanitarian aid-workers*: an aid organization may maintain records online in various databases to which only aid-workers should have access. However, due to dangers an aid-worker might incur by a repressive regime, the databases should contain no record of who makes what query.
- *Whistle-blowing*: a large corporation may wish to encourage individuals to report abuses or corruption without fear of retribution. Therefore, the corporation might want to allow anonymous searches of its intranet, but in addition, ensure that an individual cannot access information to which she is not privileged.
- *Sensitive research*: a library or news aggregator might wish to ensure that only those who subscribe to services have access to those services but otherwise allow anonymous search.
- *Corporate liability*: the protocol prevents the loss of privacy and anonymity even if databases or servers are compromised, or if a single trusted third party in the protocol is compromised (see Section 4.3).

[1]Also known as *meta-search*.

Outline. We describe this paper's contributions and related work in Section 2. In Section 3, we list the properties the protocol should satisfy. In Section 4, we informally describe the protocol. We present a mathematical model of the protocol to reason about the protocol's anonymity guarantees in Section 5, and we prove the anonymity properties hold in Section 6. Finally, we provide concluding remarks in Section 7.

2. CONTRIBUTIONS AND RELATED WORK

To our knowledge, this is the first protocol to address anonymity and privacy in federated search. Our purpose in this paper is to formulate the problem and rigorously analyze one solution. Thus, the principal contributions of this paper are (1) the formulation of anonymity and privacy properties for federated search and (2) a simple protocol that satisfies these properties. Once we have formulated the problem, our protocol is a straightforward but novel solution. We also present a simple mathematical model in which to analyze the correctness of the protocol.

We know of only one brief paper (summarizing a poster session presentation) specifically addressing privacy in federated search [8]. This paper presents a cryptographic protocol for protecting the privacy of responses in a federated search. It does not address the anonymity of clients and servers.

We hypothesize that cryptographic protocols that have been developed for privately and anonymously negotiating shared hidden credentials [6] could be used to solve the problem we pose here (but no publications exist extending those results to federated search). In brief, the hidden credentials problem is for agents to anonymously provide information

to one another only if each has the set of credentials demanded by the other. The protocol succeeds by having a trusted third-party assign a pseudonym to Alice and to tie credentials to Alice's pseudonym. Alice is provided with secret keys corresponding to her credentials. Bob can encrypt to these credentials, so that only someone in possession of Alice's private key and her credential private keys can decrypt the message. A more sophisticated protocol has been developed for cases such as anonymous and private "matchmaking" between individuals [14]. The protocol generalizes other privacy and anonymity related problems such as secret handshakes and trust negotiation.

However, these protocols depend on two assumptions which may be too strong in our context. First, they assume, of course, that clients and servers implement the novel cryptographic protocols described. While these protocols might be standardized at some time, our solution works for "legacy" clients and servers and requires only standard Transport Layer Security [16] to ensure confidentiality and integrity in point-to-point communications.

Second, the protocols assume an underlying anonymizing network for communications. (Protocols to guarantee sender anonymity for internet traffic include the Onion Router [5], Crowds [12], and Hordes [9]. The $\mathcal{P}^5$ protocol is designed to provide both sender and receiver anonymity over standard internet protocols [13].) An anonymizing network is required by those protocols because there is direct point-to-point communication between clients and servers. While anonymizing networks exist [4], they may not be available to a client (e.g., she cannot install locally the software required to join an anonymizing network).

We do not need a cryptographic protocol to solve the anonymous and private federated search problem. In our protocol, we have trusted third-parties broker all communications, forwarding search requests and routing responses. We describe assumptions about these trusted third-parties to provide sender anonymity (see Section 4). Assuming the trusted third-parties are not compromised, a novel aspect of our protocol is that for a client's identity to be revealed to a server or vice versa, *both* a client and a server must be compromised.

Related work from the database communities includes a 2004 position paper in which directions in maintaining anonymity and privacy are highlighted, with a focus on allowing individuals to maintain control over their personal data, even after it has been released. The authors draw on examples of deployed anonymizing technology, such as temporary email addresses or temporary credit card numbers [1]. A few simple protocols are also given for transferring private data. Indeed, our proposed protocol can be thought of as being inspired by the ideas of an anonymous database server architecture [2], combined with anonymous guarantees for clients and servers.

Aggarwal et al. also describe a distributed architecture to maintain privacy in database queries that does not intrinsically depend on encrypted communication [2]. (Their motivation for not relying on encryption is its performance overhead.) The motivational example given in the paper is that one database on one server contains a credit card number `XOR`ed with a random number, and another database on another server contains that random number. To recover the credit card number, both servers are queried and the results are combined. The architecture addresses privacy of the data but not the anonymity of the client or database servers.

Clayton, Danezis, and Kuhn describe some practical attacks on mechanisms that provide user anonymity, with a focus on web-based applications [3]. For example, they describe UNIX-based attacks for discovering the association of anonymous email addresses with user identities. A practical implementation of our proposed protocol must avoid these pitfalls.

3. CORRECTNESS PROPERTIES

Here we state the properties we wish for the protocol to satisfy. Subsequently in the paper, we refer to the properties of the protocol by their names and item number. In Section 6, we prove the anonymity properties, Properties 1 – 4, in our mathematical model. In Section 7, we discuss the remaining properties.

Technically, we prove three anonymity properties and one pseudonymity property. *Anonymity* is the concept of completely hiding one's identity in a transaction, whereas *pseudonymity* is the concept of associating one's identity with a pseudonym such that the pseudonym is associated with the same individual, but which individual that is remains secret. [10]. For convenience, we shall refer to the four properties collectively as the "anonymity properties". As we will describe shortly, only clients have pseudonyms in our architecture, so they may be pseudonymous but not anonymous. Servers have no pseudonyms. As noted, a key characteristic of our protocol is that for a client's identity to be revealed to a server or vice versa, *both* a client and server must be compromised.

Below, we state our properties informally. In Section 6, we state the first four properties more rigorously in our mathematical model.

Property 1 *Client-server pseudonymity*: a client remains pseudonymous to servers, unless the client disseminates its own identity to a server.

Property 2 *Client-client anonymity*: This property is the conjunction of the following two properties:

- *Pseudonymity*: a client c remains pseudonymous to another client c', unless both (1) c disseminates its own identity to some server, and (2) some server disseminates the identity of either c or c' to a client;
- *Anonymity*: a client remains anonymous to other clients unless pseudonymity is violated and a server disseminates the client's pseudonym to another client.

Property 3 *Server-client anonymity*: a server remains anonymous to clients, unless the server disseminates its own identity to a client.

Property 4 *Server-server anonymity*: a server s remains anonymous to another server s', unless both (1) s disseminates its own identity, and (2) some client disseminates the identity of either s or s' to a server.

Of course, nominally, a client or server will protect its own identity. Our qualifications regarding a client or server disseminating its own identity is to reason about cases in which an agent has been compromised.

We have two privacy properties:

Property 5 *Privacy*: for any client query and corresponding server response, no agent other than the client and server ever possess both the query and the response. In addition, *forward privacy* is preserved: a compromised agent does not cause queries and responses previously issued to be revealed.

Property 6 *Accessibility*: the protocol prevents a client from querying a server if the client does not possess an access control privilege to query that server.

Because the protocol we propose contains trusted intermediaries, Property 5 ensures that no intermediary ever contains both a query and responses to that query. In our protocol, Property 6 is enforced by a mandatory access control maintained by a trusted intermediary.

Finally, we state a liveness property:

Property 7 *Delivery*: if a client makes a query, and a server responds to it, the response is eventually provided to the client.

The purpose of this liveness property is to prevent the other properties from being satisfied by a "no-op" protocol that sends no messages.

4. THE PROTOCOL: ASSUMPTIONS AND DESCRIPTION

In this section, we begin by describing environmental assumptions about the protocol and then we describe the protocol's agents and execution. We finish by describing the trust model for the protocol and then by informally remarking on our design choices for the protocol.

4.1 Communication Model

Our model of point-to-point communication is as follows. Associated with each agent is an *address* or *identity*—we associate an agent's identity with its address and use the terms synonymously—such that any agent possessing another agent's address can send messages to that agent. Furthermore, we assume messages cannot be delivered to an unintended recipient (thus, we assume there are no eavesdropping attacks), as provided by, for example, Transport Layer Security protocols [16]. We also assume communication is non-lossy, so any message sent is received—that is, there are no denial-of-service attacks. However, lossy communication only affects the correctness of the protocol (Property 7); it does not affect any of the security properties.

We do *not* assume assured sender anonymity by the network as provided by anonymizing networks. Because of this, we depend on the trusted third-parties to take steps ensuring sender anonymity.

4.2 Trusted Third-Parties

We posit two trusted third-parties (TTPs)—an *advertiser* and a *router*.

The advertiser. The advertiser's job is to take queries from clients and depending on the access privileges of a client, forward the query to the appropriate servers. We assume the following about the advertiser:

1. The advertiser's address is universally known.
2. Each server registers its address with the advertiser so it can deliver messages to servers.

The router. The router's job is to take the responses of servers and forward them to the appropriate clients. The router has these characteristics:

1. The router's address is universally known.
2. Clients register their addresses with the router. For each address registered, the router returns *pseudonym* such that there is a one-to-one function from pseudonyms to addresses. The function is computable only by the router. For example, one implementation is to simply generate random numbers and store the random number and address pair in a lookup table kept secret by the router.

   Furthermore, we assume the router verifies the identity of clients that it registers and assigns them to *server access groups*. A server group is the set of servers a client is permitted to query.

   In an implementation of the protocol, the router may also be responsible for aggregating the responses for a particular query, or the job could be left to individual clients; our description of the protocol is agnostic in regards to where aggregation takes place (although the decision may have security ramifications).

4.3 Architecture Remarks

Let us briefly motivate the architecture before describing the protocol itself. Our architecture assumes a fairly static set of servers (rather than a federated search over, say, the World Wide Web). One purpose of the trusted third-parties (TTPs) is to ensure anonymity between clients and servers as well as preventing unauthorized access by clients to servers. The extent to which the advertiser and router provide sufficient anonymity is implementation-dependent. To a great extent, anonymity depends on the TTPs' ability to prevent

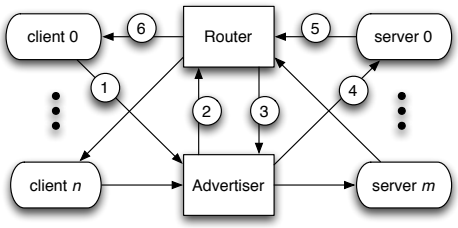

Figure 1: The Architecture

traffic analysis [11] by a malicious eavesdropper. For example, the advertiser might randomize the duration between receiving queries from clients and distributing them to servers to make timing attacks more difficult. In the case that the TTPs are compromised, note that the advertiser is essentially stateless, and the router's state is comprised of just its access control table and its mapping from identities to pseudonyms. Being (mostly) stateless can help the TTPs from violating *forward privacy* if compromised; that is, a compromised TTP should not cause the privacy of previous federated searches to be lost.

Why do we distinguish the advertiser and router? By separating them, no agent other than the client that issues a query and the server responding to it know both the query and response (Property 5). Thus, our architecture requires *both* the advertiser and router to be compromised to violate the privacy of the client-server transaction, eliminating a single point of failure regarding the loss of privacy.

4.4 The Protocol

The protocol itself is simple. For each query a client makes, the following steps are taken, as illustrated in Figure 1. There, clients are labeled 0 through n, servers 0 through m, and the edge labels $(1-6)$ correspond to the six steps of the protocol, as follows:

1. The client (in the figure, client 0) sends the pair `(id, query)` to the advertiser, where `id` is the pseudonym of client 0 generated by the router and `query` is the client's query.

2. The advertiser sends `id` to the router.

3. The router looks up the client associated with the pseudonym `id` in its secret table. The router replies to the advertiser with the client's server group, `S`, where `S` is a set of servers.

4. The advertiser broadcasts `(id, query)` to each server in `S`. In the figure, only server 0 is in `S`.

5. If a server responds to a received query, the server sends a response `(id, resp)` to the router, where `resp` is its response to the query.

6. The router computes the address associated with `id` and sends `resp` to client 0.

4.5 Threat Model

Clients and servers may behave maliciously. We presume the advertiser and router to be non-malicious, but they may inadvertently pass on messages generated by malicious clients or servers. Any number of clients or servers may exhibit any of the following malicious behaviors and the protocol properties in Section 3 must still hold:

1. A client or server sends any address it possesses to another agent (i.e., a client, a server, the advertiser, or the router).

2. A client or server sends any pseudonym it possesses to another agent.

3. A client or server sends a query to the advertiser using some other client's pseudonym it possesses.

4. A client sends a response to the router, or a server sends a response `(id, resp)` to the router such that the message `resp` contains another agent's address or a client's pseudonym.

These behaviors are described in more detail in Section 5.

5. MATHEMATICAL MODEL

The purpose of our mathematical model is to reason about the anonymity properties of the protocol. Because our protocol is not fundamentally a cryptographic protocol, our model is based on reasoning over a transition system rather than about cryptographic strength. We first describe the state components below, then we describe the initial state and state transitions, including both nominal and malicious transitions. We then give examples of protocol executions before proving the anonymity properties hold of the protocol in Section 6.

In our mathematical model, we abstract the state of the agents to just those elements required to reason about the correctness of the protocol with respect to anonymity—that is, those state elements that may lead two agents to introduce a new communication channel by passing pseudonyms or addresses.

State components. The *agents* are the advertiser A, router R, the clients, and the servers, all of which are disjoint. Let C be the nonempty finite set of clients, and let S be the nonempty finite set of servers.

Each agent maintains a *address set* denoting the set of other agents for which that agent has a *address* to send messages. For an agent x, we denote its address set as ADR_x. An agent x can send messages to an agent y only if $y \in ADR_x$. The recipient of a message does not necessarily possess the address of the sender.

Associated with each client is exactly one pseudonym denoted pn_c for client c. Additionally, we have a mapping from pseudonyms to addresses, $f(pn_c) = c$, computable only by the router. Each agent x also maintains an *pseudonym set*, PN_x, of pseudonyms it possesses.

Now we can define pseudonymity and anonymity in our model:

- **Pseudonymity** A client c is *pseudonymous* to agent y if and only if $x \notin ADR_y$.
- **Anonymity** An agent x is *anonymous* to agent y if and only if both x is pseudonymous to y, and if x has a pseudonym, then $pn_x \notin PN_y$.

The advertiser maintains a set *Queries* containing pairs of the form (pn_c, q) representing the queries it has received from clients. The queries include both the client c's pseudonym pn_c and c's query q.

The router's state is a set *Responses* of pairs of the form $(pn_c, resp)$, where pn_c is a client's pseudonym returned by the server, and *resp* is the server's response. Additionally, the router maintains a client's *server groups* as a relation $G_c(s)$ that is true if server s is in client c's server group. The server group designates the set of servers a client is permitted to query.

Thus, a system *state* of the system is the collection of each agent's respective address set, each agent's respective pseudonym set, the advertiser's set of queries, and router's set of responses, and the router's server groups.

Initial state. Initially, the router possesses the addresses of the clients, and the advertiser possesses the addresses of the servers. Furthermore, we allow that the addresses of the advertiser and router are universally known. That is,

- The advertiser's address set includes itself, the set of servers, and the router: $ADR_A = S \cup \{A, R\}$.
- The router's address set includes itself, the set of clients and the advertiser: $ADR_R = C \cup \{A, R\}$.
- For all clients c, its address set includes itself, the advertiser and router: $ADR_c = \{c, A, R\}$.
- For all servers s, its address set includes itself, the advertiser, and router: $ADR_s = \{s, A, R\}$.
- For each client c, its pseudonym set initially contains its own pseudonym: $PN_c = \{pn_c\}$. The pseudonym set of the Router is the set of all clients: $PN_R = \{pn_c \mid c \in C\}$. For every other agent x, its pseudonym set is initially empty: $PN_x = \emptyset$.
- The advertiser's set of clients' queries is initially empty: $Queries = \emptyset$.
- The router's set of responses is initially empty: $Responses = \emptyset$.
- The router's server groups $G_{-}(-)$ is initialized to be some relation on clients and servers.

State transitions. State transitions are events in which one agents sends a message to another agent, causing the receiving agent's state to update. We denote a *transition* by a pair (x, S_{i+1}), where x is the agent sending a message and S_{i+1} is the state updated from state S_i resulting from the message being sent. Our notation for a state update is

$$S_{i+1} = S_i \text{ with } exp,$$

where *exp* is the set of updates carried out. For example, from state S_i, if the next transition is (c, S_{i+1}), representing client c sending x its own address and its own pseudonym, we write

$$S_{i+1} = S_i \text{ with } c \in ADR_x \text{ and } pn_c \in PN_x$$

to show that c's address is now contained in x's address set and c's pseudonym is now contained in x's pseudonym set. For another example, if c sends its own address and the address of y to x, we write

$$S_{i+1} = S_i \text{ with } \{c, y\} \subseteq ADR_x$$

All transitions must satisfy the invariant that if state S_{i+1} is the result of agent x sending a message to agent y, then in state S_i, x must have y's address: $y \in ADR_x$. Transitions (and hence communication) are modeled asynchronously; e.g., if x sends a message before y sends a message, x's message may arrive before or after y's message. State transitions occur when messages are received, not when it is sent.

Below, we first model nominal transitions, then malicious transitions.

Nominal transitions. In Section 4, we described six transitions: (1) a client's query, (2) the advertiser requesting a client's server group, (3) the router responding with the server group, (4) the advertiser forwarding queries to servers, (5) the servers responding to queries, and (6) the router forwarding responses to clients. Of these, we only need to model transitions (1), (4), and (5) to reason about the pseudonymity properties: these transitions are the only ones in which a pseudonym or an address is transmitted from one agent to another not already in possession of the pseudonym or address as part of the protocol. (Below, we cover transitions in which a pseudonym or address is maliciously sent.)

Transition 1 *Client Query*: a client c sends a query q to the advertiser, so the advertiser updates its set of client queries and pseudonym set: $S_{i+1} = S_i$ with $(pn_c, q) \in Queries$ and $pn_c \in PN_A$.

Transition 2 *Advertiser Relay*: the advertiser forwards a query to the appropriate servers, causing them to update their set of address pseudonyms: if $(pn_c, q) \in Queries$ in state S_i, then for each server s such that $G_c(s)$, $S_{i+1} = S_i$ with $pn_c \in PN_s$.

Transition 3 *Server Response*: for a client c and server s, if c's pseudonym is possessed by s, then s can send a response to the router, including c's pseudonym: if in state S_i $pn_c \in PN_s$, then $S_{i+1} = S_i$ with $(pn_c, resp) \in Responses$.

Malicious transitions. We consider clients and servers to be untrusted, so they can send messages with malicious intent. We assume the advertiser and router to be trusted, so they do not generate malicious content, but they may inadvertently pass on malicious content sent to them by clients or servers—for example, if a server encodes[2] the address of another agent in its response. Inadvertently malicious messages sent by the advertiser or router are captured by the transitions. Furthermore, we do not address issues such as a server maliciously sending incorrect responses to a query—we only address transitions that may result in the loss of privacy or anonymity.

Again, any agent x can send any message to another agent y (including the advertiser and router) if $y \in ADR_x$.

Transition 4 *Malicious Address Transfer*: We have three kinds of malicious address transfers: one to cover malicious clients and servers sending addresses to each other, one to cover the advertiser inadvertently forwarding addresses to servers, and one to cover the router inadvertently forwarding addresses to clients.

1. If the sender x is a malicious client or server and the receiver y is also either a client or server, then the receiver y's address set is updated with any addresses in the sender x's address set: if in state S_i, $M \subseteq ADR_x$, then $S_{i+1} = S_i$ with $M \subseteq ADR_y$.
2. If the sender is the advertiser, then it executes a malicious address transfer of address z to a server s only if there exists some query $(pn_c, q) \in Queries$ such that z is encoded in q.
3. If the sender is the router, then it executes a malicious address transfer of address z to a client c only if there exists some response $(pn_s, resp) \in Responses$ such that z's address is encoded in $resp$.

Transition 5 *Malicious Pseudonym Transfer*: These are analogous to the malicious address transfers. Again, we have three cases:

1. If the sender x is a malicious client or server and the receiver y is also either a client or server, then the receiver y's pseudonym set is updated with any pseudonyms in the sender x's pseudonym set: if in state S_i, $I \subseteq PN_x$, then $S_{i+1} = S_i$ with $I \subseteq PN_y$.
2. If the sender is the advertiser, then it executes a malicious pseudonym transfer of pn_z to a server s only if there exists some query $(pn_c, q) \in Queries$ such that pn_z is encoded in q.
3. If the sender is the router, then it executes a malicious pseudonym transfer of pn_z to a client c only if there exists some response $(pn_s, resp) \in Responses$ such that pn_z is encoded in $resp$.

[2]We leave the notion of *encoding* an address or pseudonym uninterpreted. We do not model what constitutes an encoding but only the effects of an encoding.

Transition 6 *Malicious Query*: a client c updates the advertiser's query set with a query (pn_x, q) such that at least one of the following hold:

1. the sender uses a pseudonym that is not its own: $pn_x \neq pn_c$ (and $pn_x \in PN_c$), or
2. the sender encodes either pseudonyms or addresses in its query: $I \subseteq PN_c$ and $I \subseteq q$, or $M \subseteq ADR_c$ and $M \subseteq q$.

Transition 7 *Malicious Response*: A server s issues a response $(pn_c, resp)$ to the router such that at least one of the following hold:

1. c made no query to s: in the state, there exists no $(pn_c, q) \in Queries$, or
2. s encodes pseudonyms or addresses in the response: $I \subseteq PN_s$ and $I \subseteq resp$, or $M \subseteq ADR_s$ and $M \subseteq resp$.

Executions. An *execution path* of the system is an initial state S_0 followed by a finite sequence of transitions of the form

$$(x_1, S_1), (x_2, S_2), \ldots (x_n, S_n)$$

(Recall that in a transition (x, S), x is the agent that sent a message and S is the updated state based on the message sent.)

In our model, executions are monotonic insofar as no transition reduces the number of elements in an agent's pseudonym set or address set. Because executions are monotonic, executing the same transition multiple times does not affect the state; we therefore assume each transition in an execution path is unique. Let's consider a brief example of executing the transition system we have defined.

Example. This example demonstrates an execution path enabling one malicious client to transfer its address to another client, with the help of a malicious server. Assume c_0 and c_1 are clients, s is a server such that $G_{c_0}(s)$, and $G_{c_1}(s)$. The client c encodes its address in its query, providing it to the server. Then c_1 makes a query, and in s's response to c_1's query, s encodes c_0's address.

1. S_0 (initial state)
2. (c, S_1), where $S_1 = S_0$ with $(pn_{c_0}, q) \in Queries$, $c_0 \in q$, and $pn_{c_0} \in PN_A$ (malicious query)
3. (A, S_2), where $S_2 = S_1$ with $pn_{c_0} \in PN_s$ and $c_0 \in ADR_s$ (advertiser relay)
4. (s, S_3), where $S_3 = S_2$ with $(pn_c, resp) \in Responses$ (server response)
5. (c_1, S_4), where $S_4 = S_3$ with $(pn_{c_1}, q') \in Queries$ and $pn_{c_1} \in PN_A$ (client query)
6. (A, S_5), where $S_5 = S_4$ with $pn_{c_1} \in PN_s$ (advertiser relay)

7. (s, S_6), where $S_6 = S_5$ with $(pn_{c_1}, resp') \in Responses$ and $c_0 \in resp'$ (malicious response)

8. (R, S_7), where $S_7 = S_6$ with $c_0 \in ADR_{c_1}$ (malicious address transfer)

6. ANONYMITY PROOFS

Below, we formalize and prove the four anonymity properties stated in Section 3. Our proofs are by induction over the transitions of our transition system.

We begin by proving anonymity properties about the clients, Properties 1 and 2. Lemma 1 is a slight generalization of Property 1, which we will also use in the proof of Property 2. Property 2 requires two additional lemmas, since it contains both anonymity and pseudonymity components.

Lemma 1 describes a precondition that must hold in order for a client to lose its pseudonymity: the client must have disseminated its own address by encoding it in a query.

LEMMA 1. *For all clients c and client or server x, if $c \in ADR_x$ and $c \neq x$, then c executed a malicious query (pn_y, q), for some pn_y, such that $c \in q$.*

PROOF. *More precisely, we show that if $c \in ADR_x$ and $c \neq x$ in state S_n, then in state S_i such that $i < n$, the advertiser possesses a query $(pn_y, q) \in Queries$, sent by c such that $c \in q$.*

The proof is by induction.

- Initial state*: in the initial state, $c \notin ADR_x$ for $c \neq x$.*
- Induction step*: only the malicious address transfer transition (Transition 4) updates ADR_x, for all clients and servers x. Suppose transition (y, S_{n+1}) is a malicious address transfer, and $S_{n+1} = S_n$ with $c \in ADR_x$ for $c \neq x$. For our induction hypothesis, suppose that in all states S_i where $i \leq n$, for all clients c and all clients or servers x, $c \notin ADR_x$ if $c \neq x$. Consider the cases of the sender y in transition (y, S_{n+1}):*
  - *If y is a client, then c, $x \in ADR_y$ in state S_n, so the result holds trivially by the induction hypothesis.*
  - *If y is a server, then $c \in ADR_y$ and the result holds trivially by the induction hypothesis.*
  - *If y is the advertiser, then in state S_n, there exists a query $(pn_y, q) \in Queries$ such that c is encoded in q. Thus, in state S_n, c is in the address set of some client c'. If $c \neq c'$, then the result holds trivially by the induction hypothesis. Otherwise, $c = c'$, implying our lemma.*
  - *If y is the router, then $c \in ADR_s$ for some server in state S_n, and the result holds trivially by the induction hypothesis.*

Our result follows immediately from the cases. □

Property 1 follows directly from Lemma 1:

THEOREM 1 (CLIENT-SERVER PSEUDONYMITY). *For all clients c and servers s, $c \notin ADR_s$, unless c executed a malicious query in which it sent its own address.*

PROOF. *The proof is immediate from Lemma 1.* □

Lemma 2 is similar to Lemma 1 in that it describes another precondition that must hold for a client to lose its pseudonymity. In particular, we show that for some client to obtain another client's address, some server must have behaved maliciously.

LEMMA 2. *For all clients c and c', if $c \in ADR_{c'}$ and $c \neq c'$, then there exists a server s that executed either (1) a malicious response $(pn_{c'}, resp)$ such that $c \in resp$, or (2) a malicious address transfer providing c's address to c'.*

PROOF. *More precisely, we show that if $c \in ADR_{c'}$ and $c \neq c'$ in state S_n, then in state S_i such that $i < n$, there exists a malicious server s such that either (1) the router possesses a response $(pn_{c'}, resp) \in Responses$ sent by s such that $c \in resp$, or (2) s executes a malicious address transfer $S_{i+1} = S_i$ with $c \in ADR_{c'}$.*

The proof is by induction.

- Initial state*: in the initial state, $c \notin ADR_{c'}$ for $c \neq c'$.*
- Induction step*: only the malicious address transfer transition (Transition 4) updates $ADR_{c'}$, for all clients c'. Suppose transition (x, S_{n+1}) is a malicious address transfer, and $S_{n+1} = S_n$ with $c \in ADR_{c'}$ for $c \neq c'$. For our induction hypothesis, suppose that in all states S_i where $i \leq n$, for all clients c and c', $c \notin ADR_{c'}$ if $c \neq c'$. Consider the cases of the sender x in transition (x, S_{n+1}):*
  - *If x is a client, then c, $c' \in ADR_x$ in state S_n, so the result holds trivially by the induction hypothesis.*
  - *If x is a server, then the second disjunct of the lemma's consequent holds.*
  - *There are no channels from the advertiser to a client, so x cannot be a client.*
  - *If x is the router, then the router contains a response $(pn_{c'}, resp)$ such that $c \in resp$, and the first disjunct of the lemma's consequent holds.*

Our result follows immediately from the cases. □

We must prove one more lemma before proving the client-client anonymity property (Property 2). So far, we have only reasoned about client-client pseudonymity—that is, the conditions under which a client's address is revealed. We need to prove Lemma 3, which states the conditions under which a client's pseudonym is revealed to another client.

LEMMA 3. *For all clients c and c', if $c \in PN_{c'}$ and $c \neq c'$, then either (1) some server s executed a malicious response $(pn_{c'}, resp)$ such that $pn_c \in resp$, or (2) c executed a malicious query (pn_y, q), for some pn_y, such that $c \in q$.*

PROOF. *More precisely, we show that if $c \in PN_{c'}$ and $c \neq c'$ in state S_n, then in state S_i such that $i < n$, either the router possesses a response $(pn_{c'}, resp) \in Responses$, such that $pn_c \in resp$, or the advertiser possesses a query $(pn_y, q) \in Queries$, sent by c, such that $c \in q$.*

The proof is by induction.

- Initial state*: in the initial state, $c \notin PN_{c'}$ for $c \neq c'$.*
- Induction step*: only the malicious pseudonym transfer transition (Transition 5) updates $PN_{c'}$. Suppose transition (x, S_{n+1}) is a malicious pseudonym transfer, and $S_{n+1} = S_n$ with $c \in PN_{c'}$. For our induction hypothesis, suppose that in all states S_i where $i \leq n$, for all clients c and c', $c \notin PN_{c'}$, if $c \neq c'$. Consider the cases of the sender x in transition (x, S_{n+1}):*
  - *If x is a client, then $c' \in ADR_x$ in state S_n, so by Lemma 1, the second disjunct of our lemma's consequent holds.*
  - *If x is a server, then $c' \in ADR_x$ in state S_n, so also by Lemma 1, the second disjunct of our lemma's consequent holds.*
  - *Because there are no channels from the advertiser to clients, x cannot be the advertiser.*
  - *If x is the router, it possesses a malicious response $(pn_{c'}, resp) \in Responses$, such that $pn_c \in resp$, so the first disjunct of our lemma's consequent holds.*

Our result follows immediately from the cases. □

Now we can prove the client's anonymity property (Property 2). We emphasize again that the protocol ensures a client c's address is not revealed to another client c' unless both c disseminates its own address *and* a server disseminates either c's address or c''s address.

THEOREM 2 (CLIENT-CLIENT ANONYMITY). *For all clients c and c', $c \neq c'$ implies both*

- *$c \notin ADR_{c'}$, unless both (1) c executed a malicious query (pn_y, q), for some pn_y, such that $c \in q$, and (2) there exists a server s that executed either (a) a malicious response $(pn_y, resp)$, for some pn_y, such that either $c \in resp$ or $c' \in resp$ or (b) a malicious address transfer containing either c or c'.*
- *$c \notin PN_{c'}$, unless either (1) some server s executed a malicious response $(pn_{c'}, resp)$ such that $pn_c \in resp$, or (2) c executed a malicious query (pn_y, q), for some pn_y, such that $c \in q$.*

PROOF. *The proof is immediate from Lemmas 1, 2, and 3.* □

Now we just have left the anonymity properties for the servers, Properties 3 and 4. Their proofs are analogous to those for the clients. We begin by proving Lemma 4, which states a necessary condition that must hold for a server's address to be revealed to a client. The lemma is a slight generalization of Property 3.

LEMMA 4. *For all servers s and client or server x, if $s \in ADR_x$ and $s \neq x$, then s executed a malicious response $(pn_c, resp)$, for some client c, such that $s \in resp$.*

PROOF. *More precisely, we show that if $s \in ADR_x$ and $s \neq x$ in state S_n, then in state S_i such that $i < n$, the router possesses a response $(pn_c, resp) \in Responses$, sent by s, such that $s \in resp$.*

The proof is by induction.

- Initial state*: in the initial state, $s \notin ADR_x$ for $s \neq x$.*
- Induction step*: only the malicious address transfer transition (Transition 4) updates ADR_x, for all clients and servers x. Suppose transition (y, S_{n+1}) is a malicious address transfer, and $S_{n+1} = S_n$ with $s \in ADR_x$ for $s \neq x$. For our induction hypothesis, suppose that in all states S_i where $i \leq n$, for all servers s and all clients or servers x, $s \notin ADR_x$ if $s \neq x$. Consider the cases of the sender y in transition (y, S_{n+1}):*
  - *If y is a client, then s, $x \in ADR_y$ in state S_n, so the result holds trivially by the induction hypothesis.*
  - *If y is a server, then $s \in ADR_y$ and the result holds trivially by the induction hypothesis.*
  - *If y is the advertiser, then in state S_n, there exists some query $(pn_z, q) \in Queries$ such that s is encoded in q. Thus, in state S_n, s is in the address set of some client, and the result holds trivially by the induction hypothesis.*
  - *If y is the router, then there exists some response $(pn_c, resp) \in Responses$, for some client c, sent by some seller s', such that $s \in resp$. If $s \neq s'$, then our result holds trivially by the induction hypothesis. Otherwise, the consequent of our lemma follows.*

Our result follows immediately from the cases. □

We can now use the lemma to conclude the proof of Property 3:

THEOREM 3 (SERVER-CLIENT PSEUDONYMITY). *For all servers s and clients c and c', if $s \in ADR_c$, then s executed a malicious response $(pn_{c'}, resp)$, such that $s \in resp$.*

PROOF. *The proof is immediate from Lemma 4.* □

Our final theorem proves Property 4, that servers remain anonymous to other servers. To prove this property, we must prove Lemma 5 stating that for a server's address to be disseminated requires both some client and some server to act maliciously.

LEMMA 5. *For all servers s and s', if $s \in ADR_{s'}$ and $s \neq s'$, then there exists a client c that executed either (1) a malicious query (pn_x, q), for some pn_x, such that either $s \in q$, or (2) a malicious address transfer providing s's address to s'.*

PROOF. *More precisely, we show that if $s \in ADR_{s'}$ and $s \neq s'$ in state S_n, then in state S_i such that $i < n$, there exists a malicious client c such that either (1) the advertiser possesses a query $(pn_x, resp) \in Queries$, for some pn_x, sent by c, such that $s \in resp$, or (2) c executes a malicious address transfer $S_{i+1} = S_i$ with $s \in ADR_{s'}$.*

The proof is by induction.

- Initial state*: in the initial state, $s \notin ADR_{s'}$ for $s \neq s'$.*
- Induction step*: only the malicious address transfer transition (Transition 4) updates $ADR_{s'}$, for all clients s'. Suppose transition (x, S_{n+1}) is a malicious address transfer, and $S_{n+1} = S_n$ with $s \in ADR_{s'}$ for $s \neq s'$. For our induction hypothesis, suppose that in all states S_i where $i \leq n$, for all servers s and s', $s \notin ADR_{s'}$ if $s \neq s'$. Consider the cases of the sender x in transition (x, S_{n+1}):*
  - *If x is a client, then the second disjunct of our lemma's consequent holds.*
  - *If x is a server, then the result follows trivially from the induction hypothesis.*
  - *If x is the advertiser, then the advertiser contains a query (pn_y, q), for some pn_y, such that $s \in q$, so the first disjunct of our lemma holds.*
  - *There are no channels from the router to servers, so x cannot be the router.*

Our result follows immediately from the cases. □

Our final anonymity theorem proving Property 4 shows that for a server's address to be revealed to another server, both a client and a server must behave maliciously.

THEOREM 4 (SERVER-SERVER ANONYMITY). *For all servers s and s', $s \neq s'$ implies both $s \notin ADR_{s'}$, unless both (1) s executed a malicious response $(pn_c, resp)$, for some client c, such that $s \in resp$, and (2) there exists a client c that executed either (a) a malicious query (pn_x, q), for some pn_x, such that either $s \in q$, or (b) a malicious address transfer providing s's address to s'.*

PROOF. *The theorem follows immediately from Lemmas 4 and 5.* □

7. CONCLUSION

In this conclusion, we address the remaining properties from Section 3 and then discuss potential extensions and modifications to the protocol.

Remaining properties. We have proved the anonymity properties hold in our mathematical model. This leaves the two privacy properties and the liveness property stated in Section 3. Our model does not address the privacy properties or the liveness property. As discussed in Section 4.3, the Privacy Property (Property 5), stating that no agent other than a client and a responding server possesses both a query and response, holds from the architectural design. Of course, if either the client or server is compromised or malicious, they could broadcast query-response pairs to other agents. The Accessibility Property (Property 6), stating that a client cannot query a server outside its server access group, also holds provided the advertiser performs an access check on each query, and the access control table in the router has not been modified. Finally, the Delivery Property (Property 7) stating that queries are eventually answered. This property depends on the implementation; as mentioned earlier, its purpose is to prevent the "no-op protocol" (i.e., a protocol that sends no messages) from vacuously satisfying the other properties. An implementation must also take measures to prevent denial-of-service attacks by malicious clients and servers.

Protocol extensions and modifications. One extension to the protocol would be for the advertiser or router to have filtering capabilities. The filters could remove identifying information that a client or sender includes, either by mistake or by malicious intent.

Clients are assigned pseudonyms once, allowing a server to determine its history of transactions with a client. The history can be used to gain trust without sacrificing pseudonymity [15]. Alternatively, a new pseudonym could be generated for each client query to maintain stricter anonymity guarantees.

Finally, the protocol does not address implementation details concerning how to achieve setup conditions for the initial state of the protocol. This includes the advertiser obtaining the addresses of the servers, the router obtaining the addresses of the clients, the clients obtaining their own pseudonyms, and the router mapping clients to server groups. Our mathematical model does not address issues regarding dynamic behavior—e.g., clients arriving or disappearing. However, if every pseudonym assigned by the router is unique, we hypothesize the correctness of the protocol should not be affected.

In conclusion, we have presented a novel protocol to address anonymity and privacy for federated search. This is the first such work of which these authors are aware. We believe federated search to be widespread as data sources become more diverse, more transient, and users require more personalized search. We hope this work inspires additional research in the security of federated search.

Acknowledgments

Adam Wick originally suggested applying this protocol the author developed to the problem of federated search. Eric Mertens and Iavor Diatchki provided valuable feedback on the protocol. Dylan McNamee and Levent Erkök provided useful feedback on an early draft of this paper.

8. REFERENCES

[1] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-molina, K. Kenthapadi, N. Mishra, R. Motwani, U. Srivastava, D. Thomas, J. Widom, and Y. Xu. Vision paper: Enabling privacy for the paranoids. In *Proceedings of VLDB 2004*, pages 708–719, 2004.

[2] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In *Conference on Innovative Data Systems Research (CIDR)*, pages 186–199, 2005.

[3] R. Clayton, G. Danezis, and M. G. Kuhn. Real world patterns of failure in anonymity systems. In *In Proceedings of the Workshop on Information Hiding*, pages 230–245, 2001.

[4] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[5] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.

[6] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, pages 1–8. ACM, 2003.

[7] P. Jacsó. Internet insights - thoughts about federated searching. In *Information Today*, page 17. October 2004.

[8] W. Jiang, L. Si, and J. Li. Protecting source privacy in federated search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (Poster Session)*, pages 761–762. ACM, 2007.

[9] B. Neil and L. C. Shields. Hordes: A protocol for anonymous communication over the internet. *ACM Journal of Computer Security*, 2002.

[10] J. R. Rao and P. Rohatgi. Can pseudonymity really guarantee privacy? In *Proceedings of the 9th conference on USENIX Security Symposium*. USENIX Association, 2000.

[11] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[12] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1:66–92, 1998.

[13] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *In IEEE Symposium on Security and Privacy*, pages 58–70, 2002.

[14] J. S. Shin and V. D. Gligor. A new privacy-enhanced matchmaking protocol. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

[15] A. Singh. TrustMe: Anonymous management of trust relationships in decentralized P2P systems. In *In IEEE International. Conference on Peer-to-Peer Computing*, pages 142–149, 2003.

[16] S. A. Thomas. *SSL and TLS essentials securing the Web*. Wiley, 2000.