Pythia: A Privacy Aware, Peer-to-Peer Network for Social Search

Shirin Nilizadeh, Naveed Alam, Nathaniel Husted, Apu Kapadia

Indiana University Bloomington
{shirnili, nalam, nhusted, kapadia}@indiana.edu

Indiana University Technical Report TR687

October 2010

Abstract

Social search systems such as Aardvark and Facebook Questions allow users to pose questions to their social network in real time. Upon receiving a question on a particular topic, Aardvark for example forwards the question to available "experts" close to the querier in the social network to facilitate immediate, relevant answers to questions that prove too complex for web searches, e.g., when "Googling it" is not likely to yield adequate answers. While such systems have tremendous potential to tap into expertise, they are monolithic and do not provide adequate privacy. For example, Aardvark and Facebook have complete knowledge of the social network's structure, and users cannot pose anonymous queries or hide their areas of expertise. Thus the success of these systems will be limited to more general categories of questions and expertise, since many users will avoid asking or answering questions related to sensitive topics such as health, political activism, sexual orientation, and even innocuous questions which may make the querier seem ignorant.

We propose Pythia, the first distributed peer-to-peer (P2P) social-network architecture that provides queriers and experts with anonymity and yet provides a mechanism to locate relevant experts. Through simulation on social network graphs, we show it is possible to strike a balance between privacy, timely satisfaction of queries, and proximity of experts to queriers, while maintaining scalability. Furthermore we explore statistical attacks specific to this domain, and show how queriers and experts can maintain their anonymity over time to defend against such attacks. We thus provide the first blueprint of a privacy-aware, P2P system for social search, and hope to spur future research into the development of such systems.

1 Introduction

We focus on a compelling application of social networking called *social search*. Social search is a broad category that exploits an *asker's* social network to find answers that are more relevant to the asker of the query. For example, search results for movie recommendations could be ranked based on movie rankings from the asker's friends or people close in the social network.

A new class of what we call *live social search* systems is now emerging—social search systems such as Aardvark[1] [20] leverage the power of social networks to connect askers with *online* experts (i.e., humans with domain expertise) who are close to the asker in the social network, thus facilitating *live* exchanges of information between real humans. Aardvark claims "The vast majority of questions are answered within 10 minutes." Horowitz and Kamvar [20] draw the distinction between the *library* model of search, where askers search for the *right document* to answer a question vs. the *village* model, where askers seek to get connected with *right human* because it is unlikely their complicated question can be satisfied by an existing document. In fact, Google recently acquired Aardvark, noting on their blog "sometimes the information just is not online in one simple place."[2] While we have used Aardvark as an exemplar of such a system, there is considerable interest in this model of social search. Facebook Questions is currently being rolled out to its users, and provides a social search service. Its tagline is "Ask the Facebook community a question on any topic and get quality answers from people you know."[3] Cha-Cha,[4] features experienced human *guides* who search the web, sift through the results, and return relevant search results to the querier. Thus we don't consider Cha-Cha to fit under the model of live social search since they provide a human-mediated "library search" as opposed to a "village search".

Unfortunately, centralized systems such as Aardvark and Facebook do not provide adequate privacy to users because they maintain full knowledge about the social network. For example, the identities of askers and experts, participants' interests and expertise areas, and their communication are all known to such systems. In addition, a user's identity and profile immediately become available to the other users when the user posts a query or answers a question. The user's location is also revealed, making it easier for an adversary to gain additional knowledge about that user.

While these systems support general queries related to restaurant recommendations, home improvement, and product recommendations, the inability to ask or field queries privately limits the scope of queries and advertised expertise. For example a pro-choice or pro-life advocate may want to field queries about the topic but not want her colleagues, or anybody (including the social search service) to know of her leanings. Similarly, experts may want to engage in political activism and keep their involvement secret. Askers, too, may ask questions on these topics and others such as health related or legal issues, only under the cover of privacy. In some cases, users may simply be embarrassed to ask "simple" questions with the fear of being perceived as ignorant. Even if the social search service kept identities private with respect to communicating askers and experts, the social service itself has knowledge of all this information. All this information is vulnerable to abuse, subpoenas, secondary use, unauthorized data aggregation, and is also a prime target for data breaches. Thus, in addition to the existing social search systems supporting non-private questions and answers, a system is needed that supports private queries and responses, where askers and experts are private to each other and no central authority can uncover private expertise or an individual's queries (we formalize the security goals in Section 2.3).

---

[1] `http://www.vark.com/`

[2] `http://googleblog.blogspot.com/2010/02/google-acquires-aardvark.html`

[3] `http://www.readwriteweb.com/archives/why\_facebook\_questions\_could\_be\_zuckerbergs\_dream.php`

[4] `http://www.chacha.com/`

We propose *Pythia*,[5] the first distributed peer-to-peer social network architecture that can support sensitive queries and answers in a privacy-aware way. Since Pythia is a distributed system, there is no central clearinghouse for queries and answers. Queries are routed anonymously within the social network to experts in a way that both askers and experts can maintain anonymity. While there exist systems to route traffic anonymously in distributed systems such as Tor [12] or Crowds [28], and even along social links [26], it is not clear how such routing-layer solutions can help find available experts in the social neighborhood of an asker while maintaining anonymity. Furthermore, Tor provides weak anonymity guarantees: it does not provide unobservability or privacy against a global passive adversary. Pythia aims to provide both these properties. Using existing anonymizing networks with a centralized server like Aardvark is also unsatisfactory because the central server is a central point of failure, and the server cannot leverage social network vicinity to route queries in a user's network. Our design of Pythia therefore addresses the trade-off of balancing privacy and quality in the system making use of requisite routing-layer protocols.

The central idea in our design of Pythia is to partition the social network into *communities* or *flood zones* and then use *local flooding* to send questions to online experts within the community. Such flooding provides a high degree of privacy within the community (as we explain in Section 3) but yet limits the amount of flooding to maintain scalability. When no nearby experts are found, we argue that beyond 2 or 3 hops in the social network, it probably doesn't matter where the expert is located in the social network, and thus any remote community with online experts can be contacted using *remote flooding* within the distant community. We show that for a low constant amount of overhead, Pythia supports private queries (with anonymity relating to the cluster size) while maintaining the quality of responses when nearby experts are available.

We evaluate our architecture through extensive simulations and analyze statistical *intersection attacks* that apply to this new domain of P2P social search systems. At a high level, attackers are able to log the presence (online vs. offline) of nodes and correlate this information with questions and answers observed in the network with the goal of learning the identities of the experts for a particular topic. We analyze the degradation of anonymity of experts as they field questions over time. We analyze various categories of questions and expertise (rare topics vs. common topics for example) as well as users with different preferences (how many questions they are willing to answer in a day or week). We use real-world data on these aspects for parameter selection in our simulations. Finally, we show that time-aggregation based defenses improve the anonymity of participants.

---

[5] In Greek mythology, Pythia was the gateway to the Delphic Oracle, "the most important oracle in Greece," and "answered questions put to her by worshippers." See `http://www.answers.com/topic/delphic-oracle`

Contributions We make the following contributions: **1)** We identify a new class of privacy-aware P2P systems for the exciting area of *live social search*, where the system must match askers and experts without any centralized knowledge, and must defend participants' privacy. **2)** We present the design of *Pythia*, the first decentralized, peer-to-peer live social search system that supports private queries and responses while maintaining the quality of answers in the system. **3)** Through extensive simulations, we show that it is indeed possible to balance privacy and quality in a P2P social search system. We show how privacy of askers and experts degrade over time, and evaluate defenses. **4)** We provide a roadmap for future research to realize the potential of P2P live social search.

2 System Model and Design

In this section we describe the high-level system model we assume for P2P social search systems, and then describe our security goals and adversary model. In the following section we present our communication architecture for Pythia to address these security goals and adversary model.

2.1 P2P Social Network

Several systems [11,21,30] have been proposed for building P2P social networks, where nodes are connected to friends' nodes to form a large, distributed peer-to-peer network connected by *social links* [26]. Creation of such a network can be bootstrapped through an existing social networking platform such as Facebook, or through instant-messaging applications such as AOL Instant Messenger, Skype, or Google Talk. The creation of this underlying network is not the focus of our paper, and discuss the system model and architecture assuming such an underlying P2P social network.

2.2 System model

We assume that a user is connected to, and has knowledge of his/her list of friends in the social network. Every user has a list of declared *expertise areas*, i.e., topics for which he/she can answer questions. In this paper we do not seek to measure and evaluate the level of expertise, and thus the expertise areas are self-declared. For example, Alice's set of expertise areas could be {restaurants, Bloomington IN, military intelligence analyst, computer networks, environmental activism}. Some of these expertise areas are *private expertise areas*, and are not known to other users in the network. In such cases, the expert prefers to answer others' questions anonymously, and not be known as an expert in that area. We focus on private expertise areas because it is straightforward to locate non-anonymous experts using a distributed hash table (DHT) that stores the IP addresses of experts for a particular topic, for example. Therefore, in the remainder of the paper we assume that each node's expertise areas are kept private from other users in the system.

Each user in the social network is represented as a node. Nodes may be *online* or *offline* at any given time. An online node may also be *idle* at any time. Sometimes online users are busy or not at the computer and therefore not available for conversation. We refer to

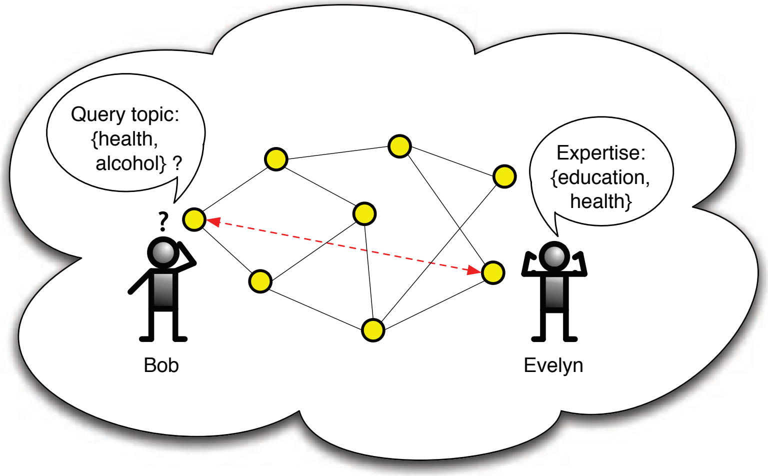


Figure 1: High-level system model. The nodes and links represent part of the larger social network. Bob issues a query related to alcohol abuse as indicated with the query tags. The query is routed (routing mechanism depends on specific architecture) to expert Evelyn who responds to Bob.

such cases as having idle status. This *online and idle status* of nodes is visible only to their friends. Offline nodes cannot assist in routing queries and responses. While idle nodes can route messages, only *available* (not idle) nodes are capable of answering questions.

A peer with a query can attach a set of *query tags* to the query, where the tags indicate topics related to the query. Based on the query tags, the *query routing* protocol attempts to find experts for the specified tags close to the asker in the social network. We expect that some queries will be best directed to nodes close to the asker in the social network, although in the absence of adequate expertise in the social vicinity, queries will be directed to more distant experts. Once the query is routed to an available expert, the expert can provide a response. We further assume that not all available experts will answer the question, and will depend on how *responsive* the expert is. To this end, we assume experts are willing to answer only a few questions per week.

2.3 Security goals

The first two properties are standard to P2P anonymizing networks, the other three are specific to Pythia:

Anonymous queries: Routed queries should not be attributable to a specific identity beyond a "reasonable" probability. We purposefully leave this probability threshold unspecified to support algorithms that strike different trade-offs between privacy and other properties. For example, it may be reasonable to offer anonymity of "1 in a 10" rather than "1 in a million." This property allows privacy-conscious askers to make queries related to health conditions for example, while hiding their identity. In our work, we deliberately evaluate settings where "1 in a 10" or "1 in a 20" anonymity is considered adequate enough for "probable deniability" [28], where there is a reasonable chance with which the user can say he/she wasn't the originator of a query. The reason for this level of anonymity will be made clearer below when we talk about expertise unlinkability. We also note that Pythia supports larger levels of anonymity even though we use “1 in 10” and “1 in 20” anonymity in our examples.

Anonymous responses: Similar to anonymous queries, the answer being routed back to the asker should not be attributable to a particular identity beyond reasonable probability. In the event the asker and expert choose to engage in a conversation, the anonymity property applies to the conversation transcript. In other words, a particular identity cannot be tied to the conversation with reasonable probability.

The following three properties are unique to P2P live social search systems. While the previous properties apply to particular instances of communication, the next three properties seek to hide interest and expertise areas of users:

Unobservable queries and responses: We also seek to hide whether nodes are asking questions or providing answerers. Note that anonymous queries/responses may allow an attacker to observe that a particular node is asking or answering a question, allowing the attacker to narrow down the set of possible users for a particular answer or question. With this property, the attacker cannot tell *which* nodes actually asked or answered a question at all.

Expertise unlinkability: A peer's private expertise areas should not be attributable to her identity beyond reasonable probability. For example, Bob may be an academic professional who is also a pro-choice advocate who doesn't want to widely advertise his association with the pro-choice movement. An attacker might notice that Bob always seems to be online when pro-choice answers are received and links this area of expertise with Bob. (see Section 5). Thus, anonymity much greater than “1 in 50” is not necessary since the prior probability of a user being an expert in a particular topic is assumed to be in that general range.

Interest unlinkability: This property states that an asker's private query tags should not be attributable to her identity beyond reasonable probability. For example, Alice may want to ask several queries about terrorist organizations she hears about on the news but may worry about being labeled a terrorist. An attacker might notice that Alice always seems to be online when terrorism related queries are sent and link this area of interest with Alice.

2.4 Attack model

Type of adversary. We assume *honest-but-curious* attackers, where nodes in the network participate in the protocol fairly but try to infer as much information as possible from passive observations. We assume a strong version of such adversaries, where all the honest-but-curious attackers collude with each other. Such nodes are hard to detect (as opposed to an active attacker who may falsify messages, drop messages, and so on) and attempt to use their observations to uncover the expertise areas of anonymous experts or topics of interest of anonymous askers. Furthermore, such adversaries include the *global passive adversary* who can observe all communications in the system. As described in Section 3, Pythia employs some special nodes called *representatives* that need to be trusted for delivery of messages, but we assume these nodes can also be adversarial. We also assume the adversary has full knowledge of the structure of the social network.

Adversary's capabilities. The attackers can pose as regular users in the system and thus receive all the queries and answers that are exchanged in the communities.[6] Even though attackers cannot tell who is asking or answering questions, the attackers can observe the online/offline and idle status of the nodes whenever a question or answer is sent in a community. We assume a strong attacker who can determine such online/idle status of nodes through network-level pings, communication attempts via instant messaging, or through simple instant message (or even Facebook) status messages. More specifically, over a period of time, the attackers observe the network and whenever a particular topic is asked or is answered by experts in the community, the attackers collect online/offline and idle status of nodes in the community. Using and linking this collected information, the attackers try to determine the asker of a particular topic or the answerers of a particular topic.

Adversary classes. We classify the attackers based on their knowledge of the network and specific capabilities:

- **Global Attacker.** This powerful attacker has the knowledge of the online/offline and idle status of all the nodes in a community at any time. While such an attack would be difficult in practice, a set of attackers can, for example, try to become friends with everybody in the system. In the extreme case, all nodes in the network are "friends" with at least one attacker, and thus their online and idle status is known to the attackers. We assume the attackers add nodes to the existing social network to attack the existing nodes, and are *outside adversaries*.
- **Colluding Attackers.** These attackers have partial knowledge of the network; i.e. they can observe the online/offline and idle status of only their friends. We assume some fraction of users c (where $0 < c < 1$) are attackers and are thus *inside adversaries*. We assume these malicious nodes are uniformly distributed across the network, although in the future we plan to examine other attack models where attackers may be clustered into communities (relating to different ways in which botnet infections spread). This scenario represents a botnet of compromised nodes inside the network, and these nodes are used to observe other nodes.

Moreover, we assume that these two types of attackers can have two different capabilities:

- **Linkable.** Attackers have the ability of linking answers generated by a particular expert in the community for example by comparing the context of answers in that topic. Answers authored by the same expert could have similarities in writing style, similar typos and punctuation, and so on. Thus we assume that attackers with this capability can link messages from the same expert.
- **Unlinkable.** Attackers do not have ability to infer any useful information from the answers in a particular topic to link them to a particular expert.

Thus we have four types of adversaries: *Global-Linkable*, *Global-Unlinkable*, *Colluding-Linkable*, and *Colluding-Unlinkable*. Next, we describe Pythia's architecture in Section 3 and evaluate these four adversaries in the context of Pythia in Sections 4 and 5.

---

[6] Since attackers can inject questions by copying questions and thereby obtain answers for those questions, we assume attackers can receive answers as well.

3 Architecture

Figure 2 shows the high-level architecture of Pythia. The central idea in Pythia is to partition the *nodes* in the social network into *anonymizing communities* or *flood zones*. Questions from a community are received by all nodes in the community by means of a *local flood*. The local flood allows anonymous answerers to receive questions without having to reveal their expertise areas. Furthermore, to provide asker/answerer unobservability, all nodes ask and answer questions at regular intervals (including dummy questions and answers) and thus attackers cannot readily pinpoint which nodes are asking or answering questions. If no answerers are found in the local community, questions are forwarded to a remote community (with known answerers) as part of a *remote flood*.

Each community has a special *representative*. In many distributed systems such as P2P networks (e.g., KaZaA, Gnutella and Skype), sensor networks and ad-hoc wireless networks, some leaders (or coordinators and super nodes) play a specialized role in the application that requires frequent communication with the other members of the set. The purpose is to strike a balance between the efficiency of centralized search, and the autonomy and load balancing which provides better performance and scalability. These super node based peer to peer networks have been subject of recent studies [5, 7, 15, 16, 18, 24, 32, 35]. Also, we assume that if these representatives go offline, a new representative is elected.

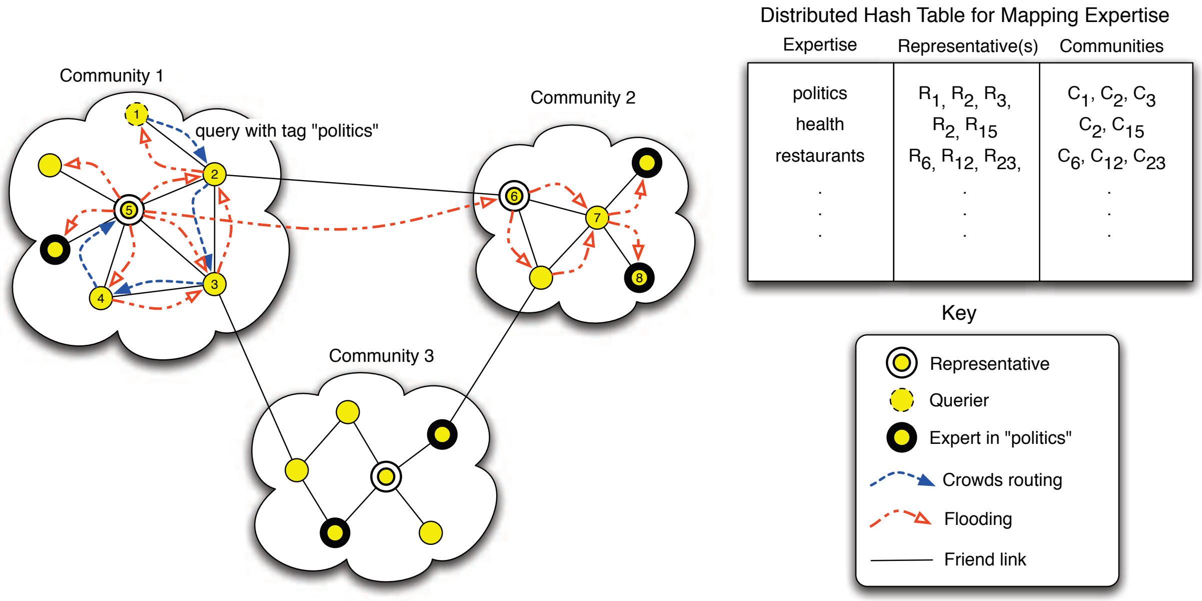
The role of the representative will be made clear below. In Pythia, time is divided into time intervals $\{t_1, ..., t_n\}$, where questions and answers are exchanged and distributed at the end of each time interval as coordinated by the representative. As we will show later, longer time intervals provide better privacy against attackers, but delay communication times. For practical purposes, one can assume time slots are "a few minutes" long. We again note the representative can be adversarial, but is trusted to perform communication tasks honestly.

Communities are small enough to limit the overhead of flooding as well as to target answerers who are close to the asker in the social network, but large enough to provide an anonymity set that has reasonable degree of anonymity. Our work does not seek to provide near-complete anonymity (i.e., where the answerers can be any of a several million nodes in the network) but attempts to strike a good trade-off between privacy and performance. In this context, we assume that being anonymous within a set of "tens to a hundred" nodes is a good balance. The size of communities, however, is a system parameter, and represents a trade-off between privacy and performance. Smaller communities provide lower privacy, but ensure that the overhead of flooding of queries is a low constant (as opposed to a full flood of the entire network, which has overhead linear in the size of the social network for example). In the following sections, we first describe Pythia's components then we describe different phases of the social search algorithm.

Pythia first establishes communities, following which questions and answers can be routed in the system. We now describe the various stages and the Pythia protocol.

3.1 Creating social communities

In Pythia, all nodes in the social network are grouped into self-organizing clusters called *communities*. In this paper we do not focus on optimizing community creation, and instead modify the distributed clustering scheme proposed by Ramaswamy et al. [27] for our pur-



Distributed Hash Table for Mapping Expertise

Expertise	Representative(s)	Communities
politics	R_1, R_2, R_3,	C_1, C_2, C_3
health	R_2, R_{15}	C_2, C_{15}
restaurants	R_6, R_{12}, R_{23},	C_6, C_{12}, C_{23}
⋮	⋮	⋮

Figure 2: Node 1 in Community 1 initiates a query with the tag *politics*. The query follows a random walk (using Onion routing) along social links and is eventually sent to the *representative* node 5. The representative then initiates a local flood of the query as indicated (along with other received queries). The answerer in the community is unresponsive. The representative, having received no responses, asks the DHT for communities with answerers in politics, and initiates a remote flood in Community 2 by contacting representative node 6. Node 8 is a responsive answerer who responds to the query, and the response is sent to representative node 6 via a random walk, then related to representative 5, and finally received by node 1 in the next flood by node 5.

poses. Each community is initiated by a node called the *initiator* and established based on nodes' connectivity requiring only local knowledge about neighboring nodes. Each community has a group ID and each node belongs to a single community. Every user knows which of his/her friends' are in his/her community. The initiator of a community, once formed, can act as the *representative* of the community. The representative is seen as the head of the community and is responsible for forwarding questions and answers on behalf of users in the community as described in Section 3.2. In addition, from the perspective of other communities, the representative of a community is a conduit for communicating with nodes inside the community. Communities can have more than one representative, but they can only have one initiator. For simplicity we assume that communities have only one representative. Since it is not the focus of our paper, we briefly summarize Ramaswamy et al.'s community creation protocol in Appendix A.

3.2 Routing messages and finding answerers

The aim of the routing algorithm is to implement a routing technique that effectively distributes the messages to the answerers in the social network while providing anonymity to

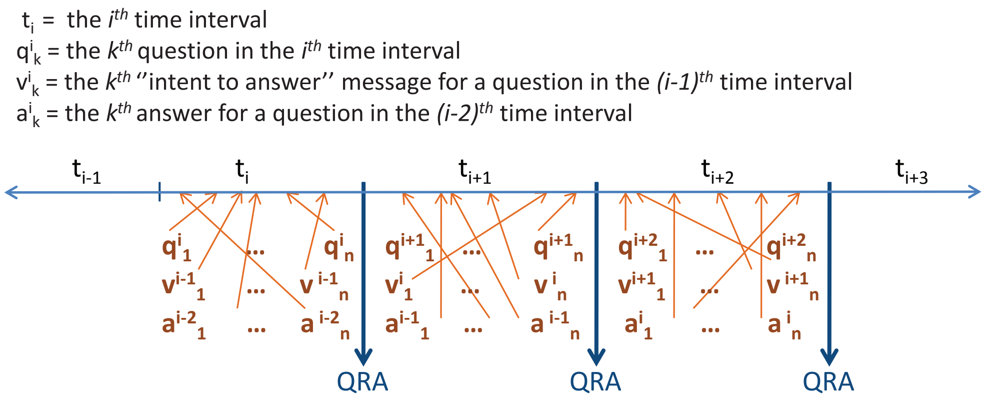

t_i = the i^th time interval
q^i_k = the k^th question in the i^th time interval
v^i_k = the k^th "intent to answer" message for a question in the (i-1)^th time interval
a^i_k = the k^th answer for a question in the (i-2)^th time interval
t_{i-1}
t_i
t_{i+1}
t_{i+2}
t_{i+3}
q^i_1 … q^i_n
v^{i-1}_1 … v^{i-1}_n
a^{i-2}_1 … a^{i-2}_n
q^{i+1}_1 … q^{i+1}_n
v^i_1 … v^i_n
a^{i-1}_1 … a^{i-1}_n
q^{i+2}_1 … q^{i+2}_n
v^{i+1}_1 … v^{i+1}_n
a^i_1 … a^i_n
QRA
QRA
QRA

Figure 3: In each time period t_i, every online node sends three real or fake messages to the representative. The "intent to answer" messages are in response to the questions which were asked in the previous time period t_{i-1} and the answers are in response to the questions which were asked two time periods previously, t_{i-2}. The representative separates, mixes and initiates a local flood of all the real questions, pseudonyms of volunteers and real answers. It initiates a remote flood of the questions which do not get enough volunteers in the community.

the asker and the answerers. To simulate a distributed social search, nodes in Pythia participate in four phases: advertising, asking, showing their intent to answer and answering. In all phases, the same protocol is used while the type of messages is different. All the messages in a community are forwarded to the representative of the community, all of them are padded to have the same size, and all are encrypted by the public key of the representative.

3.2.1 Messages sent by user nodes

As a building block, Pythia uses Onion routing [14] to deliver messages from nodes to representatives. We assume that the list of IP addresses in a community is available to all nodes after community creation. The sender of a message can pick a set of n random nodes and progressively build a circuit as done in Tor [12] (we conservatively set $n = 6$, which provides adequate "mixing time" in social networks, such that from the receiver's point of view the message could have originated anywhere in the social network. The particular choice for this parameter is orthogonal to this work, and it suffices to pick a value that provides adequate mixing). After a circuit is established, and session keys between the sender and each node in the circuit, the sender creates an "onion packet" by encrypting the message with each node's key. As a node receives a message, it decrypts the outer layer, and passes it to the next node in the circuit. This approach prevents attackers in collusion with the representative to link a particular message with a particular sender, providing asker/answerer anonymity.

For asker/answerer unobservability an extra step is needed. As illustrated in Figure 3, during every time period t_i, *every* online node sends three separate messages to the representative, along three different routes, at randomly chosen times in the interval. Every time period, a user can ask up to one question, and sends q^i in time period t_i. Each online user also sends one "intent to answer" message v^{i-1} for a question received in the previous time period t_{i-1}. Representatives pick at most α answerers (at random) to answer the question (e.g., we set $\alpha = 2$ in our simulations), and thus at most α selected answerers from the set of answerers who sent volunteer messages for the question are informed about this selection in the previous time period t_{i-1} (how nodes are informed will be made clear in Section 3.2.2). Online nodes send one answer a^{i-2} for the question they selected in t_{i-2} and askers must wait between 2 to 3 time periods to receive answers to their question. For all these messages, if the online node does not have a legitimate question, intent to answer, or answer, the node sends a dummy message. Thus attackers cannot directly observe who is asking or answering questions.

The three types of messages sent by nodes in the community are:

Question message $q^i = \langle t, rep, qid, T, fake\_bit, question \rangle$. This message consists of the time period t when the message was sent, a representative identifier *rep*, a question identifier *qid*, a set T of topic tags, a fake bit *fake_bit* and the *question* string. As we said earlier, every representative in the community has a unique identifier. For simplicity in the routing layer, we assume this field is the representative's IP address, so that the final node in the onion circuit can send the message directly to the representative. The question id is a sufficiently long (e.g., 160 or 256-bit) random identifier (and thus unique with high probability) which distinguishes questions. This identifier will be used by representatives for routing answers for appropriate questions. The topic tags shows the topic of the question which is used for finding appropriate communities that have answerers in this expertise and for asking the questions to them. The fake bit is for identifying the question as a fake or real question. If the fake bit is true, the question string is a dummy string and the representative does not need to flood the question to the community.

Intent to answer message $v^i = \langle t, rep, qid, pnym, v\_bit \rangle$. This message consists of the time period t, a representative identifier *rep*, a question identifier *qid*, a user's pseudonym *pnym* and a "volunteer to answer" bit *v_bit*. Each node picks *one* question it is willing to answer per time period, and sends a one-time pseudonym for the question it has volunteered to answer. If the user is busy or idle, a dummy message must still be sent as indicated by the "volunteer to answer" bit.

Answer message $a^i = \langle t, rep, qid, fake\_bit, answer \rangle$ consists of the time period t, representative identifier *rep*, question identifier *qid*, a *fake_bit*, and the *answer* string. Every online node must answer at most one question per time period, and if a real answer is not sent, it is signaled by the *fake_bit*. The usage of representative identifier and question identifier is similar to its use for other messages. If the fake bit is true, the answer string contains the real answer for the identified question.

All messages are sent via an Onion route to the representative and all nodes send the same types of encrypted messages. Moreover, encrypted messages have fixed sizes so the adversary cannot infer the content from the message size. Attackers therefore cannot pinpoint which nodes are asking and answering questions.

3.2.2 Actions taken by representative

At the end of each time period, after receiving the three types messages described above from online nodes in the community, the representative floods a *QRA message* to all nodes, containing **q**uestions, chosen **r**esponders, and **a**nswers. We describe the flooding mechanism as well as the QRA message next.

QRA message The QRA message flooded by the representative at the end of each time period contains three blocks as described next, the current time period, and a digital signature. This message can thus be verified for freshness (through the time slot id embedded in the message) and integrity.

Question block. At the end of time period t_i, the representative collects all questions $\langle q^i \rangle$ received in time period t_i, and creates a *question block* $\langle q^i \rangle'$ with all the real questions. If questions with topic tags for which there exists no expertise in the local community are found, they are forwarded to a remote community with answerers in that area. Advertised topics for communities are obtained through a DHT. We describe the expertise advertising process next, in Section 3.3.

Respond block. Next the representative collects all the intents to answer blocks $\langle v^i \rangle$ where users have previously indicated which of the questions in t_{i-1} they are willing to answer. Based on parameter α, the representative picks at most α intents per question, and creates a *respond block* $\langle v^i \rangle'$ containing only the selected intents. Answerers that have been picked to answer, find their one-time pseudonym in the respond block.

Answer block. The representative collects all received answers $\langle a^i \rangle$ for questions in t_{i-2}, and creates an *answer block* $\langle a^i \rangle'$ with these answers.

We note that reordering the messages within each block is not necessary: first we assume the representative is adversarial and thus all adversaries know the order in which messages were received by the representative. We rely on the Onion routing mechanism to provide anonymity to the sender of a message. While even such mechanisms are open to attack and counter defenses, this is not the point of our research. A system like Pythia opens up new avenues for attack, and we evaluate those avenues even in the face of perfectly anonymous routing.

Flooding To reduce load, the representative floods the messages through his/her social links. Flooding through the social network is done in the usual way with nodes relaying messages and ensuring that messages are not resent along a link. Sometimes *local flooding* cannot reach all nodes in the community because of online/offline pattern of nodes and it is possible that in some cases some nodes are isolated from other nodes in the community and therefore cannot receive the messages that the representative has flooded to the community. To solve this problem, nodes that do not receive the flood at the time period boundary (after a *timeout* period) initiate a direct connection to the RP requesting the message. The representative sends all messages directly to the unconnected nodes. Note that this direction communication takes place only for the messages sent by the representative and thus do not reveal any information to attackers for compromising privacy. The content to the message received from the representative is the same for all nodes, whether by direct connection or through other nodes.

Remote flooding When a local representative rep_{local} sends a question to a remote community via the remote representative rep_{remote}, rep_{remote} includes the question in its QRA block, except the identity of the representative is changed to rep_{remote}. Answers are routed back to the originating representative rep_{local} when received. In the interest of space, we omit details on this protocol, but provide results from our simulation for both local and remote floods. While users may need to wait an extra time interval to get remote answers, we note that usually we expect queries to be satisfied locally and if not, users would wait for a remote answer. Our protocol, however, can be easily modified to send a remote query in parallel, in which case two communities are posed the question simultaneously as opposed to falling back to a remote community when a local answer is not available.

Messaging overheads At each time interval, the traffic generated in each community can be analyzed by calculating the messages sent by each member and the representative. Consider N members in each community. Each member generates three messages: one question, one answer and an intent to answer message. Assuming the size of each message as 500 bytes, $N * (500 + 500 + 500) = 1500 * N$ bytes of traffic is sent to representative by users during a time interval. The representative floods a QRA message containing three blocks at the end of each time period. Assuming that users ask on average 2 questions per week,[7] the number of real questions asked in a day is $0.29 * N$ in the community. In other words, about 30% of members are asking a question per day. In an overly conservative estimate, assuming all questions are asked in one time interval in a day and assuming that the community receives half of this amount of questions from remote communities, the size of total questions would be $0.45 * N * 500$. Assuming getting 2 answers for each question, the size of total answers is $2 * 0.45 * N * 500$. The size of QRA's response block is $0.45 * N * 70$. Thus the size of one QRA message is $0.45 * N * (500 + 1000 + 70)$ which is flooded by the representative to N members. Therefore with a community size of 100 within a time interval, 146.5 KB traffic is received by the representative and it sends out 69 KB to the community.

3.3 Advertising and reputation

When nodes join the network, they need to send their list of interests to the representative. To prevent linkage attacks, users send individual messages through an Onion circuit, where each message contains only one expertise area. Furthermore, users pick a random time slot from the range of 1 to β time slots in the future to advertise each interest. We assume β is large enough (e.g., 1 week) so that there is enough churn and online/offline/idle/available activity that expertise updates cannot be linked with specific users in the community.

Upon receiving such advertisements, the representative stores the expertise areas of users to the Community Interest distributed hash table that lists interests of all nodes in a community. Every node can access the DHT (Distributed Hash Table) and verify whether its expertise has been added to the DHT. The DHT's entries are available to anyone in the social network thus everyone in the social network can look up the expertise areas that a community advertises (or which communities advertise a particular expertise area) but no one knows who in the community is an answerer in what area. Interests are removed from the DHT every 2β time slots unless the interest is advertised again. Thus users must re-advertise their interests by picking a new random time slot after the previous advertisement. We leave more sophisticated methods for advertising interests as well as the determination of lower values of β for future work.

---

[7] The median active users in Aardvark [20], issued 3.1 queries per month.

Furthermore, we assume that nodes can advertise their expertise along with reputation information for that expertise. For example, users can accumulate anonymous digital cash [3] for answering questions, and then prove their wealth (e.g., different credit thresholds can be used to establish low, medium, and high levels of expertise). A detailed reputation mechanism is outside the scope of this paper and we leave details and evaluation for such a scheme to future work. Absent such a scheme, remote communities can be picked at random, although in our evaluation we assume such a scheme to identifying suitable remote communities.

4 Attacks and Defenses

In Section 3 we describe Pythia's architecture and justified the design choices for attaining the various privacy goals; in this section we provide the details of our attacks. Honest-but-curious attackers can gain complete or partial information about online and idle status of nodes within a community for each time period. In particular, the *anonymity set* for a user associated with an anonymous query or response is the set of non-idle users for the particular time period. In a stronger attack, questions and answers exchanged in Pythia also contain public information about topic tags, and attackers can use this additional information to correlate the online/idle status of nodes *across* time periods. Topic tags can also be used to uncover the identities of answerers for particular topics, or the askers of certain topics. Since attacks against expertise/interest linkability are stronger than attacks against anonymous queries/responses, we study the degradation of privacy against expertise/interest linkability. Without loss of generality, we now describe attacks against expertise linkability, which we will analyze through simulation based on realistic parameters. We note that representatives are the only nodes who know which messages are *local* (initiating by nodes in their community) or *remote* (coming from other communities), and to give adversaries this capability we assume the representatives are also adversarial. Thus, the adversary we consider is strong and has the capability to identify which community is the originator of a question or an answer. Dummy messages between communities will not help and the goal of the adversary is to de-anonymize the identity of the expert in a community.

We now describe the attacks used by the various attackers. In all the attacks, attackers pick a sensitive topic to attack (i.e., to determine which users have that topic as an expertise area) and collect information about nodes who are online and not idle in each time slot when an answer for that topic is observed.

4.1 Intersection attack—linkable answers

This attack is used by the Global-Linkable and Colluding-Linkable attackers. The main idea is that every time an answer for a topic is received it can be linked to the previous answer, and so the attackers can eliminate idle and offline users as candidates for the answerer behind the answer(s). Global attackers can eliminate more candidates because they have a global view of status, whereas colluding attackers can eliminate only those nodes that they are able to observe. As the colluding fraction increases more nodes can be eliminated resulting in smaller anonymity sets for users, whereas lower colluding fractions will result in larger anonymity sets for the answerers. These attacks show the worst case anonymity for users, since it assumes a strong adversary that can link multiple answers from the same answerer.

4.2 Counter Attack—unlinkable answers

This attack is used by the Global-Unlinkable and Colluding-Unlinkable adversaries. In this scenario, attackers cannot link answers by the same answerer, and thus answers for a particular expertise area could come from one or more answerers. Furthermore, since these answerers may not be online at the same time, the naive intersection attack described above will not work to easily eliminate candidates.

In this attack, the attacker maintains a *counter table* to log the presence of users when questions are answered. The counter table contains a counter value for every node in the community that the attacker can observe. In the beginning, the counter value is initialized to zero. The aim of the attackers is to find the answerers with expertise e' in a community C. To do so, whenever an answer with the expertise e' is observed in the community, the attacker increments the counter for every online and non-idle node for that time interval. The attackers record the counter value across time periods (for up to 4 weeks in our experiments). Presumably, the answerers who have mostly answered questions in this particular topic have a higher counter value and are de-anonymized. To study the effectiveness of this attack, we consider the following attack strategy: the attacker sorts the list of answerers by descending order of counts, and then "draws a line" somewhere in the list with the hope that a large number of answerers are included in the section of the list above the line. For example, the nodes with the highest counts are likely to be answerers, and drawing a line higher up in the list will uncover a few answerers (at the risk of missing several answerers below the line). For studying such performance, the use of *precision* and *recall* are the most appropriate metrics. For the selected set (above the line) *Precision* is the ratio of the number of answerers in the selected set to the total number of users in the set. For example, if attackers are trying to uncover nuclear answerers and the top 3 nodes are picked; then, if 2 are nuclear answerers, the attackers have a precision of 66.67%. However, the attackers may have missed 20 other nuclear answerers in the community of 100, resulting in a *recall* of $\frac{3}{23}$, since only 3 of the 23 nuclear answerers were in the selected set. Through simulation we find the average precision and recall for various points at which the attacker may draw the line, showing the tradeoffs.

4.3 Time-aggregation defense

The attack is successful only as long as the attackers have few nodes that are online and not idle. However, if the time is delayed in sending the questions and answers then the attackers have to observe online and offline nodes over a longer time interval spanning multiple time periods. Thus attackers need to consider candidates that were not idle at any time period within the delay interval. A time delay causes the intersection set to enlarge and the and the node counter to increment. We have analyzed the attack over different time-aggregations and show how the attack degrades as the time-delay increases.

5 Evaluation

We first evaluate Pythia for its ability to maintain the privacy and anonymity of answerers in the face of adversaries as described in Section 2.4. Next, we evaluate its performance, i.e., its ability to pair askers with answers and the social-network distance of answerers from askers. We create a realistic simulation of a Pythia network that mimics current production social-search services like Aardvark.

5.1 Usage models

Unlike searching for files in P2P networks, social search involves more social interactions [36]. For simulating this complex process resembling a large-scale social search engine, we defined asking, answering, expertise and online/offline models using released data about Aardvark and Skype [20, 25, 29].

Online/Offline Model For simplicity, we assumed users are all from a country with 3 time zones. We assume that Pythia's client can be run by default similar to Skype so peer lifetime matches PC operation schedules, i.e., the client is turned-on during the day and turned-off during the night. We considered 6:00 am as the start hour of day and 12:00 am as the end of the day. According to [25, 29], in Skype 95% of peers disappear after 10 hours of activity. Therefore by having three time zones, people start their PCs some time during the day after 6:00 am and are online for 10 hours. A day is divided into 288 time-slots where each time-slot constitutes a 5 minute time-interval. The time span simulated in the experiments set to 4 weeks to analyze the degradation of anonymity over a month. The time at which users come online depends on the time zones to which they are assigned. Note that users may still be idle while they are logged into the system; we discuss the idle status of users in the *answering model* paragraph below.

Expertise Model Every user can ask questions in 36 different topics by tagging the question with its topic. In our system users are assigned a different number of expertise topics according to the rough distribution of users and topics in Aardvark as described by Horowitz and Kamvar [20]. Table 1 shows the topic distribution used in our simulations.

Users were randomly chosen for each percentage group (based on the probabilities in the first column of Table 1). Depending on the percentage group, the number of topics

Table 1: Distribution of percentage of users and number of topics

Percentage of Nodes	Minimum Topics	Maximum Topics
2%	1	2
17%	3	4
31%	5	8
27%	9	16
17%	17	32
6%	33	36

was randomly selected from the minimum and maximum range. The topics were randomly selected from the topic pool containing 36 topics and assigned to the user. We defined three types of expertise categories: Common, Uncommon and Rare. We have considered 36 expertise topics equally divided among the three categories. Further, common topics are assigned to 70% of nodes, uncommon topics are assigned to 30% and rare topics are assigned to 6% of nodes (relating to the number of standard deviations around the mean distribution of topics). For example, if Topic 4 is a common topic, approximately 70% of the nodes have Topic 4 in their list of areas. Likewise, if Topic 31 is a rare topic, only about 6% of the nodes will have Topic 31 as their expertise area. Nodes can answer a question with a tag/topic if they have the associated expertise topic listed in their interests. We note that topics such as *restaurants* and *movies* might fall into the common category where most users are interested in such topics. While many users may not consider such topics to be sensitive we assume some users want to keep their “common” interests or hobbies private.

Asking Model Although according to Horowitz and Kamvar [20] the median active user in Aardvark issued 3.1 queries per month, in our asking model every user issues on an average 2 queries per week. The reason for choosing this number is that the data presented by Horiwitz and Kamvar [20] indicate the network has grown dramatically from October 2008 to October 2010. And if these systems become more popular, people may ask questions more often. We believe that an average of about 8 queries per month is a reasonable assumption as such systems get more popular. We also point out the degradation of anonymity is *faster* with our choice of this parameter. According to the normal distribution, the distribution of common, uncommon and rare questions are respectively 70%, 28% and 2% (relating to the number of standard deviations around the mean distribution of topics).

Answering Model According to Horowitz and Kamvar [20], in Aardvark 20% of the users are more active than others and these active users are responsible for 85% of the responses. In Pythia, we model this pattern by labeling 20% of users as *active users* and the remaining as *passive users*. If an active answerer gets a query, she responds with an 85% probability while if a passive answerer gets that question, she responds with a 15% probability. Although people are online for most of the day, they may be unresponsive. For active users we randomly chose 15% of their online time slots to be idle and for passive users we chose 85% of their online time slots to be idle. Like Aardvark, Pythia users can choose the maximum number of times in a day that they can be bothered for answering a question. Simulated users answer 1 to 5, chosen uniformly at random, questions a day.

The representative attempts to obtain $\alpha = 2$ answers for any question and if not found locally, sends the question serially to the representatives of other communities until it gets at least $\alpha = 2$ answers. In our experiments we find that on average 0.25 remote communities are contacted per question.

5.2 Security Evaluation

To test Pythia with various potential topologies of a social network we created randomly generated scale-free graphs using the Network Work Bench[8] (NWB) tool and the Barabasi-Albert (BA) model. The BA model is known to have similar properties to social networks "in the wild" [2]. When generating the graphs we used a minimum links parameter of 5 to set the minimum number of outgoing links of a user to 5. This parameter is specific to NWB. We created 5 graphs with 60,102 nodes. Communities are created using the process discussed in Section. 3.1. A subset of communities is taken due to the computational complexity of the simulation. The subset communities are sized between 85 nodes and 115 nodes ($\mu = 95.526, \sigma = 8.6702$). Each graph was used for 5 simulation experiments. Therefore, each group of 5 experiments had the same social network structure, topology and number of communities but had different distributions of expertise and online/offline assignments to the nodes. The graphs represent values (as the case may be) after 4 weeks of system operation. Standard error bars are plotted for all values. Our data show that the confidence intervals for estimating the average are reasonable for our simulation parameters.

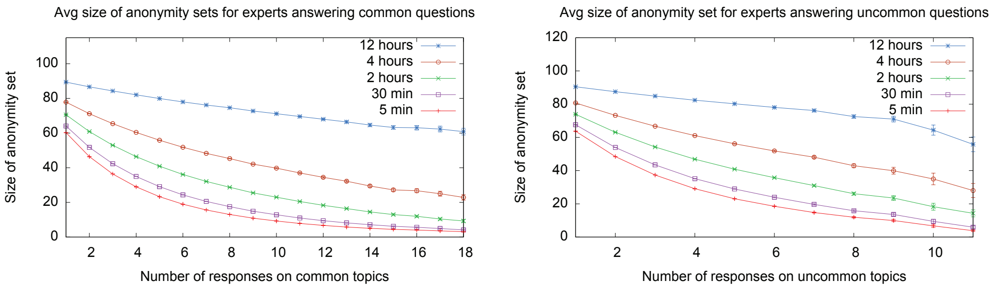
We now evaluate attacks as described in Section 4. We vary the number of colluding attackers from 5% to 30% of the nodes in the network. We evaluate defenses by aggregating over time intervals ranging from 5 minutes to 12 hours.

Intersection attack by Global-Linkable adversaries Figure 4 shows how Global-Linkable attackers can degrade the anonymity of answerers watching answers for a topic and linking them to the answerers. The X-axis indicates the number of questions that answerers have responded to after 4 weeks and the Y-axis indicates the average size of answerers' anonymity sets. We see that the number of questions that an expert in common topics has answered is much more than the number of questions that he/she has answered in uncommon and rare topics. This difference is because of the distribution of questions as explained in the asking model (Section 5.1).

Through the intersection attack, answering more questions in a topic decreases the anonymity of the answerer more because attackers can observe more answers that can be linked to the answerer and observe who was online and not idle at those times. Figure 4(c) shows that people answering rare questions have better anonymity than answerers of uncommon and common questions over similar timespans. The anonymity as it relates to the number of questions is similar. For example, after answering 4 questions in a particular rare topic, the average size for the anonymity set is 31.82 while it is 29.11 and 28.95 for answering the same number of questions in a particular uncommon and common topic. The graphs also show that even answering 5 questions in a particular common, uncommon or rare topic during 4 weeks, the average size of anonymity set is about 20. We also note that after answering

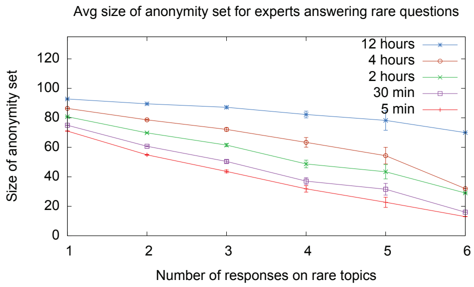
---

[8] `http://nwb.slis.indiana.edu/`



(a) Common topics

(b) Uncommon topics

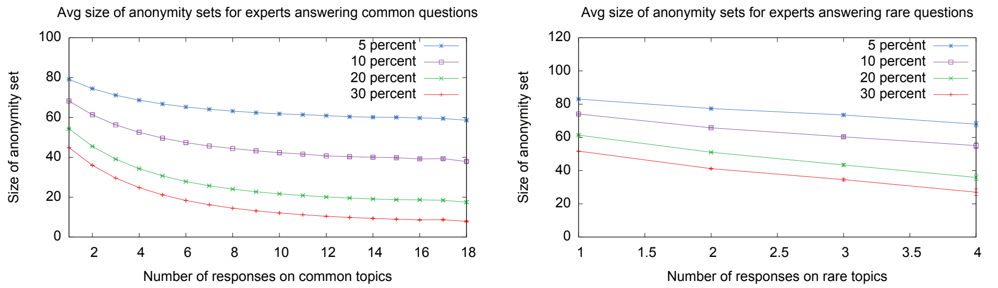


(c) Rare topics

Figure 4: Anonymity set vs. number of questions answered for Common, Uncommon, and Rare topics against a Global-Linkable attacker after 4 weeks. As the number of questions answered in a 4-week time period increases, the anonymity set decreases as expected. As a defense, questions can be sent at larger time intervals and this "aggregation" defense is shown in each graph. For 5-minute time intervals, e.g., users desiring privacy for uncommon topics can restrict themselves to answering 4 questions in 4 weeks and expect "1 in 4" anonymity, which is better than the prior probability of being an expert in that topic. Time aggregations of 4 hours will allow enough anonymity to answer 11 questions over 4 weeks.

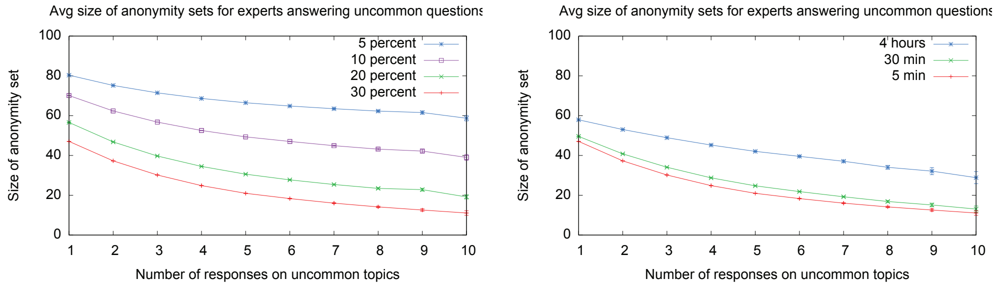
18 questions over 4 weeks for a common topic, the anonymity set is still more than size 2 on average, which is still better than the approximately 70% prior probability of the node being an expert in that topic. For uncommon topics, after answering 11 questions over 4 weeks the anonymity set is of size 4, which is better than the approximate 30% prior probability. For rare questions, the prior probability is approximately 6% and holds for 5–6 questions after 4 weeks. The anonymity set for answerers in uncommon and rare topics degrades faster than for answerers in common topics. This is because common questions are usually answered in close-by time slots in a day and the same nodes are online in those time-slots. However, rare and uncommon questions are not frequently asked and there is a long time gap from when the previous question was answered.

Figure 4 illustrates that if the time-aggregation defense is used in sending the questions and answers, the average size of anonymity set increases for a particular number of questions. For example, in the case of answering 4 questions in a particular common topic, the average size of anonymity set for time intervals of *5-minute*, *30-minute*, *2-hours*, *4-hours* and *12-hours* are respectively, 28.95, 34.95, 46.40, 60.39, and 82.05. You can see that how the attack degrades as the time-delay increases. This attack degradation is because the attackers have



(a) Common Topics

(b) Rare Topics



(c) Uncommon Topics

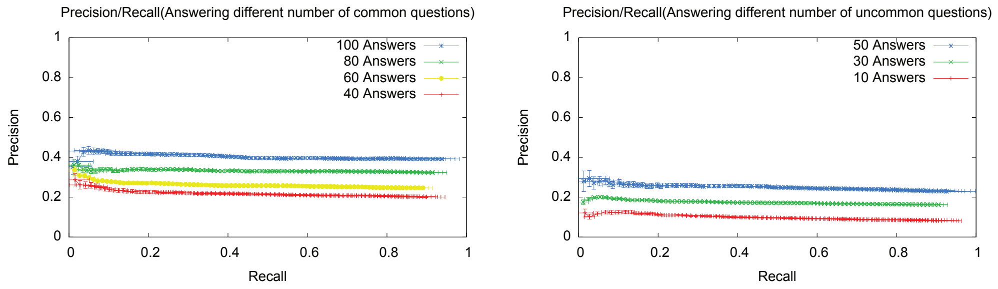
(d) Uncommon Topics With Time Aggregation for 30% colluding nodes

Figure 5: Anonymity set vs. number of questions answered for Common, Uncommon, and Rare topics against a Colluding-Linkable attacker after 4 weeks of time in all. As the number of questions answered increases in a 4-week timespan, the anonymity set decreases as expected. As a defense, questions can be sent at larger time intervals and this "aggregation" defense is shown for uncommon topics.

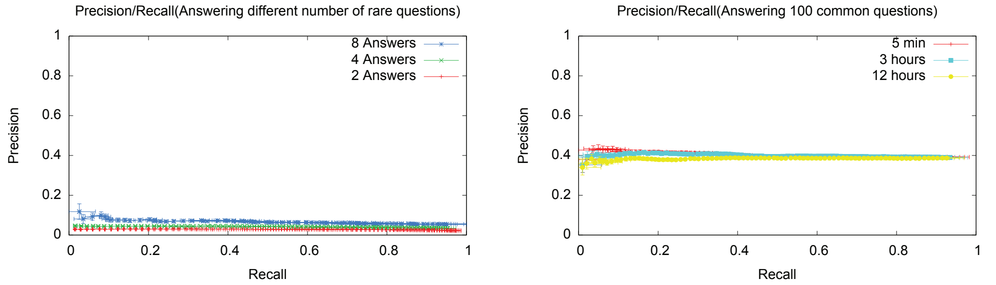
to observe online and offline nodes over a longer time interval spanning multiple time periods.

Intersection attack by Colluding-Linkable adversaries As in the previous attack, the anonymity degrades as an answerer answers more questions in a particular topic. Figure 5 shows the decline in anonymity as more answers are observed and linked by the colluding attackers. As the percentage of colluding attackers increases, the size of the anonymity set decreases and there is a greater decline in the size of the anonymity set as more answers are observed from the answerer. For example, from Figure 5(a), at the end of 4 weeks, the anonymity set for an answerer who answers one question in a common topic with 5% of colluding attackers is 79.1 and declines to 68.3 and 54.4 when the colluding percentage increases to 10 and 20 respectively. As observed in the Global-Linkable Attack, the anonymity set for answerers in rare and uncommon topics degrades faster than the anonymity set for answerers in common topics. Assuming about 10% attackers in the system, experts for common and uncommon topics can answer 18 and 10 questions respectively in 4 weeks and still have "1 in 40" anonymity, while experts for rare topics can answer 4 questions in 4 weeks and have more than "1 in 50" anonymity, which is much greater than the prior probabilities in all cases.

Figure 5(d), shows the effect of time aggregation for colluding percentage 30% on the size



(a) Common (b) Uncommon



(c) Rare (d) Common with Time Aggregation

Figure 6: Precision vs. Recall for different number of answers received for Common, Uncommon, and Rare topics against a Global-Unlinkable attacker in a 4-week timespan. The X-axis represents *recall* and the Y-axis represents *precision* in all cases. As the number of answers for a question increases, the precision increases. Each point is accompanied by standard error bars along both axes.

of the anonymity set as the number of answered questions in a uncommon topic increases. Time aggregation improves anonymity since the attackers cannot easily intersect online and idle nodes. For example, the anonymity set after 4 weeks and for 10 answers for time interval 5 minutes is 11.1 and increases to 13.1 and 28.84 when time interval is increased to 30 minutes and 4 hours respectively.

Counter attack by Global-Unlinkable adversaries Figure 6 shows how Global-Unlinkable attackers can degrade the anonymity of answerers by applying the counter attack (see Section 4.2). The counter table contains a counter value for every node in the community. To find the local answerers with a particular expertise, attackers increment the counter for every online, non-idle node whenever an answer with this expertise is observed in the community. In our experiments, the attackers record the counter value across 4 weeks. Then after 4 weeks, they sort the list of answerers by descending order of counts, and draw a line after each answerer, up to the 85th answerer, in the list with the hope that a large number of answerers are included in the section of the list above the line.

Figure 6 plots the precision vs. recall (see Section 4.2) curves for the attacker , where each point in the graph corresponds to a particular position of the line in the sorted list of counts. Looking at the graphs in Figures 6(a), 6(b), and 6(c), we see the precision is flat for a particular number of answers while the recall is increasing from 0 to 1. The flat precision shows this attack is not successful in finding the answerers at top of the list and answerers are uniformly distributed in the list. Uniform increase in recall shows that reaching to the end of the list, more answerers are found. Thus even though attackers can say that the answerers are among the nodes above the 85th node in the list, they cannot find the answerers by a high enough assurance. Even after 4 weeks, the precision is at best 0.43 for common topics, whereas the prior probability is 0.7, and thus attackers cannot perform better than what was already known. We see similar results for uncommon questions, and for rare questions even after 4 weeks, experts have at least "1 in 10" anonymity because precision is about 10%. In general attackers perform poorly (compared to the Global-Linkable) attack because there are multiple experts for the same topic answering questions on that topic but online/offline or idle/not-idle at different times. Thus it is *not* the case that experts in a particular topic "bubble" to the top of this list. While we expected this behavior intuitively, our experiments confirmed our hypothesis.

As explained in Section 4 Global-Unlinkable attackers cannot link answers by the same answerer, and thus answers for a particular expertise area could come from one or more answerers. This is the reason of seeing more number of answers for a particular common, uncommon and rare topic in this attack compared with the intersection attack during a 4-week period. Similar to intersection attack, because of the distribution of questions, we see more number of questions that have been answered by experts in common topics than the number of questions that have been answered in uncommon and rare topics. In our experiments, the maximum number of answers in a particular common, uncommon and rare topic are respectively, 140, 70 and 14 and the minimums are 20, 3 and 1.

Figures 6(a), 6(b), and 6(c) illustrate that when people answer more questions in a particular common, uncommon and rare topic, the precision increases more and the attack is more effective in de-anonymizing the experts because more data about online and not idle status of answerers is available to attackers and the counter value for answerers who have mostly answered questions in this particular topic is higher.

Figure 6(d) shows that the time-aggregation defense (for common topics) makes very little difference for this attacker, because the attack is already quite weak as seen in the previous three graphs. For example, in the case of answering 100 questions in a particular common topic, the maximum precision for time intervals of *5-minute*, *3-hours* and *12-hours* are respectively, 0.43, 0.41 and 0.38.

Note that since we show the attack is already quite ineffective with Global-Unlinkable adversaries, in the interest of space we omit results for Colluding-Unlinkable adversaries.

Implications of our results Thus, if we assume adversaries can link responses from experts by noticing similarities in text (Global-Linkable adversaries), the anonymity of users degrades over time. This is a fundamental limitation of anonymity systems that cannot control the content of messages being exchanged. On the other hand our results are promising by showing that if answers are not linkable, anonymity improves greatly. Users must therefore ensure that their messages do not contain revealing characteristics. Filtering text to break linkages is a subject of ongoing research in our group.

5.3 Comparison with Random Walk

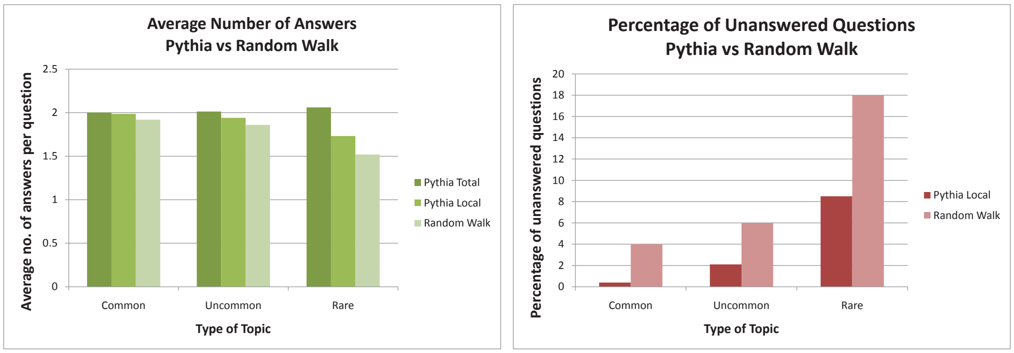
Pythia's design supports unobservability of messages and is thus more secure than solutions that use random walks to find questions in the neighborhood. Flooding however pays the price of potentially finding answerers farther away in the social network than with random walks. Random walks are also vulnerable to issues such as graph disconnectivity. In this section we evaluate Pythia against the technique proposed by Kacimi et al. [22], and show that Pythia performs reasonably well or even better than the random walk depending on the metric under evaluation.

Random Walk Experiment In the technique used by Kacimi et al. [22], a node creates a query packet consisting of her question, the expertise of the question and two dummy answers. The node then selects one active online friend and forwards the packet to her. On receiving the packet, the receiving node stores the message and the forwarding node in a local table. Every node, on receiving the packet, selects one of their active online friends, different from the sender, and forwards the question packet. The node also stores the friend to whom it forwarded the query packet in its local table. After getting a question, if the node has the expertise, she responds according to her active or passive behavior which is assigned to the node by the answering model. When the node responds to the question, it replaces one of the dummy answers in the question by her answer. If the number of real answers in the query packet is less than two then it forwards the query packet to another active friend. If the node does not have the expertise to answer the question or does not answer the question according to the answering model, then it forwards the question to another active friend with a 99/100 probability. This forwarding probability acts as a randomly chosen TTL for every query packet. If the user decides not to forward the question, if the number of answers is two, or there are no online friend to send the question to, the questions along with its answer(s) are sent back along the path it was forwarded until the asker receives it.

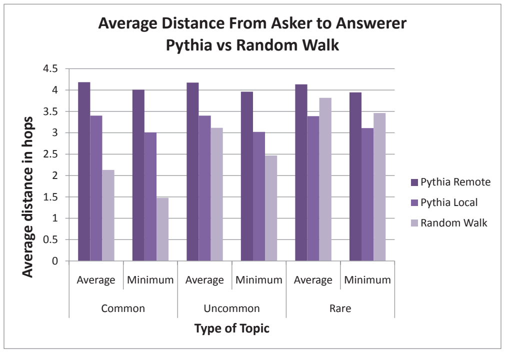
Security Comparison with Random Walk Kacimi et al.'s technique aims to provide security against the platform (e.g., Facebook). As such the idea of "replacing" dummy answers with real answers can be observed by colluding nodes on either side of an answerer, thus greatly reducing the expertise anonymity of users. Furthermore, their technique makes the asker send multiple random walks. Colluding friends noticing the same question from Alice can easily infer that Alice originated the question. In our evaluation, we fix this problem by initiating a single path from the asker. Despite the security weaknesses of Kacimi et al.'s scheme, we next evaluate the performance of Pythia in the context of their scheme.

Performance Comparison with Random Walk In our experiments the average number of nodes contacted is set to 100 for Pythia Local (where only local nodes are contacted) and for the random walk.

Average number of answers received. On an average, questions in Pythia receive 2, 2.01 and 2.06 answers for common, uncommon and rare topics respectively. As seen in Figure 7(a), for all topics, the average number of answers received for a question from a local community in Pythia is higher than the average number of answers received per question in the random walk experiment. When the required number of answers are not received from the local



(a) Avg. Answers (b) Avg. Unanswered



(c) Avg. Distance

Figure 7: Performance Comparison of Pythia with Random Walk

community, the question is forwarded to remote communities. Random walk receives fewer answers on average than Pythia because as we see in the next graph a larger fraction of questions go unanswered. In Pythia, the average number of answers received from a local community is 1.99, 1.94 and 1.73 for common, uncommon and rare topics respectively while the average number of answers received in the Random Walk experiment is 1.92, 1.86 and 1.52 for common, uncommon and rare topics respectively. Higher number of answers are observed for rare topics in Pythia because when a question does not receive the required number of answers from the local community, the representative forwards the question to other communities. For common and uncommon topics, the minimum required number of answers are usually received from the local community and therefore these questions are not forwarded to remote communities.

Average number of unanswered questions. Figure 7(b) shows the percentage of questions not receiving any answers from the local community in Pythia to the percentage of questions not receiving any answer in Random Walk. In the Random Walk experiment, the question is forwarded along a single path which may be cut short when no online nodes are present or if the node chooses not to forward the question further. Thus, not all questions get answers—0.4%, 2.1% and 8.5% of questions in common, uncommon and rare topics respectively do not get any local answers. In the random walk experiment, 4%, 6% and 18% of common, uncommon and rare questions respectively do not get any answers.

Average distance of askers to answerers. The distance of askers to answerers in the local community is greater for common and uncommon topics in Pythia than in Random Walk as can be seen in the Figure 7(b). This behavior is expected because a random walk is expected to hit an expert within a few hops for such areas. In contrast, two experts from the entire community are picked at random in Pythia, and may not be as close. However, the distance of answerers for rare topics is better in Pythia than compared to Random Walk. This is expected because a random walk would take much longer to find experts and could stray far away from the asker in the social network. Pythia on the other hand locates rare experts from the community when available. Answerers in remote communities are on an average at a distance of 4 hops from the asker. The average minimum distance of rare answerers to askers in a local community in Pythia is 3.11 while it is 3.46 in Random Walk.

6 Discussion

Community Creation. Creating communities is an important part of Pythia. A small diameter to reduce distance between all nodes in the community and low variance in sizes between all communities to offer a uniform level of anonymity across communities is desirable. Currently we use a process discussed in Section 3.1 to provide a baseline set of communities to use with Pythia, but future work can seek to improve these properties. Creating communities with more population increases the anonymity of users. On the other hand it decreases the performance. Finding the best practical size for communities is left to future work.

Supporting conversations. In this paper we consider single exchanges of one question and answer. Ideally, askers and answerers would have the ability to continue their conversation. For greatest privacy, messages can be related by the representative by following the similar exchange of messages and QRA blocks, but that will introduce a 1 time-period delay for each message. We leave such extensions and their analysis to future work.

Shielding. We have evaluated the degradation of anonymity for an expert over the course of 4 weeks. Our assumption is that daily online/offline patterns of a particular user will be stable across days, and experts would expect to see a lot of the same users at the same times across different days. We are currently exploring what we call *shielding sets*, where participants would recognize the set of users that are usually online while they are online, and only ask or answer questions when a large fraction of these users are online. Participants can thus keep track of their anonymity sets, and compute their loss of anonymity when they ask or answer questions. Such a study needs long term data about the online/offline patterns of users, and we are seeking such data for further exploration.

Reputation. One major function of Aardvark is to learn which participants are responsive and good at answering questions on what topics. By building reputation scores for users, questions can be targeted more effectively. Assigning and managing reputations for each user hinders anonymity and privacy. A reputation scheme is needed which provides reputations that can be used to target experts while at the same time protecting the anonymity of users. Pythia's community-based design is particularly amenable to assigning reputations to communities. A reputation system at the granularity of a community helps locate experts for a particular topic without actually identifying the expert. We discuss a possible approach in Section 3.3. Flooding helps to ensure that questions reach experts as well as other potential users that can answer the question. Thus certain communities might build up reputation for certain topics, while keeping the actual expert anonymous within the community. One alternative is to assign cryptographic tokens such as digital cash [3, 6] to experts, who can accumulate credit for their answers, and prove the accumulated wealth while maintaining anonymity. Such techniques will improve the targeting of questions to the right experts and thus improve the quality of answers received.

Tagging. Currently we assume a mechanism to match query tags with areas of expertise. In the simplest case, string matching will ensure that property tagged queries will be matched with areas of expertise. In practice, however, a mechanism is needed to correlate similar tags and areas. Web ontologies[9] would be a suitable way to achieve semantic mapping of tags to areas.

We leave the area of distributed ontology mapping to future work, but in general one could envision a DHT-based approach, where experts can lookup the query tags to see if there is a mapping to his or her interests. Of course, in the interest of privacy, some mechanism is needed to obtain such mappings in a privacy-preserving way. A naive solution would be for all nodes to keep the entire ontology locally, and then make semantic mappings without any loss of privacy.

Hybrid approaches. One interesting approach to balance quality and privacy would be to "hook" into existing centralized social search engines. Queriers can send a question to Aardvark through a local flood, following which the question enters Aardvark through several (say 3) peers in the community. Aardvark can then send the question to experts close to the peers in the social network. Queriers can thus make anonymous queries through hook nodes, and receive reasonable quality answers from Aardvark.

Incentives. It is important to note that freeriding is a problem in systems such as Pythia where user participation is required. Incentive mechanisms that promote user participation and discourage freeriders will be useful. We have based our model on the real-world system Aardvark, which largely relies on altruism. We have already discussed digital cash techniques to incentivize users to build reputation and thus answer truthfully and also frequently. Simple policies can help control freeriding for e.g., number of questions that users can ask may depend on how recently the user has joined or on the number of questions that a user answered. This would be an interesting area for future research as incentive mechanisms can also take into account the quality of answers provided which can be combined with a reputation system.

7 Related Work

Locating a person with some specific expertise or knowledge has been addressed in several contexts such as knowledge sharing, Q&A systems, social search and peer-to-peer systems.

Knowledge sharing is a broad topic discussing the transmission of knowledge from one individual to another. Traditional knowledge sharing networks include websites such as MAKE Magazine[10] and eHow.[11] Torrey et al. [33] study how information is searched for and learned in the traditional web. Adamic et al. [1] studied knowledge sharing and its relation to Yahoo Answers, an online question-and-answering (Q&A) service, as well as what type of questions people asked. Q&A services come in a number of different forms. Popular services include Yahoo! Answers, Amazon Askville, Wiki-answers, and Google Groups. These services allow users to post their questions and answer other questions using pseudonyms. The only privacy provided by these websites is the use of pseudonyms, and due to their centralized nature, user IPs can still be tracked. None of these are *live* social search systems as they offer only offline communication, and the service does not actively find experts for queries.

---

[9] `http://tomgruber.org/writing/ontology-definition-2007.htm`

[10] `http://makezine.com/`

[11] `http://www.ehow.com/`

Several social search applications exist that use an individual's social network to parse out their most relevant search results. For example, Google now has a social search feature as part of their web browser.[12]

Danezis et al. show how search preferences can be shared with a select group of friends of the user in a social network [10]. The end goal is to rank search engine results based on preferences of the social group. Target groups that are "cohesive" (densely linked) are used for propagating preferences using a broadcast approach. One possibility for future work is to leverage these groups in Pythia. While they provide a framework for sharing information within such groups, it is not clear if it can be used to send questions and locate experts. Our protocol is tailored to the orthogonal problem of communicating questions and answers privately as opposed to sharing search preferences privately.

Other services such as Cha-Cha set up a "human middleman" to maximize the number of relevant results returned to the user. Many of these demonstrate the power of humans to filter out inconsequential data during searches. Duggan and Payne [13] show that individuals with greater knowledge in an area show increased search performance. The issues associated with domain-knowledge and search success are also well researched topics [4,17,19]. Aardvark [20] modified the traditional form of Q&A networks to improve the quality and relevance of answers by instantaneously leveraging the user's social networks. The downside to Aardvark-style live social search systems are that they are a central clearing house of questions and answers and thus provide less privacy to their users.

Cutillo et al. [9] describe a peer-to-peer architecture implementation for social networks. Their goal is to remove centralized control and to provide privacy by leveraging trust amongst the trusted friends in the network. Li et al. [23] study the feasibility of P2P as a web search engine infrastructure. These systems however do not support live social search. Pythia can be seen as complementary to such systems as it can implement live social search on top of P2P social networks. Wu et al. present a P2P based distributed search system called Sixearch.org [34]. Sixearch (Social Web search via adaptive peers) is a P2P search engine for locating static content, e.g., document collections. This system could be considered a successor to the Freenet system proposed by Clarke et al. [8]. We are considering a modification of Sixearch.org software as a Pythia implementation as future work.

Related to the flooding-style of routing in Pythia, P5 [31] is a protocol for scalable anonymous communication over the Internet. Users of the system locally select a level of anonymity and performance and map themselves to a broadcast group, which provides requisite performance. Although it provides anonymous communication, it lacks specific features required by Q&A networks.

Last, and most related to our work, Kacimi et al. [22] present a protocol that allows anonymous opinion exchange among users connected over an untrusted social network platform. We have pointed out the shortcomings of this approach along with a comparative evaluation in Section 5.3.

---

[12] `http://www.google.com/support/websearch/bin/answer.py?hl=en\&answer=165228`

8 Conclusions

We present *Pythia*, a privacy-aware peer-to-peer system for live social search. We have made the first comprehensive attempt at designing such a distributed system with strong privacy guarantees, and show the feasibility of our approach through extensive simulations. While this work provides an important first step, we hope to spur further research in areas such as privacy-aware query routing, defenses against intersection attacks in such systems, incentivizing use of such systems for sensitive queries, and assigning reputation to anonymous experts. We are entering an age of social networking applications, where social search is bound to succeed through services such as Aardvark and Facebook Questions. Yet much work remains to be done to support private queries about sensitive issues. Without such systems, the full potential for social search will not be realized.

Acknowledgments

We would like to thank Johan Bollen, Filippo Menczer, Katy Börner, Russell Duhon, and Chris Schneider for their helpful comments.

This research was funded in part by a grant from the Indiana University Center for Applied Cybersecurity Research.

References

[1] L. Adamic, J. Zhang, E. Bakshy, and M. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *Proceeding of the 17th international conference on World Wide Web*, pages 665–674. ACM, 2008.

[2] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47–97, 2002.

[3] M. H. Au, S. S. M. Chow, and W. Susilo. Short e-cash. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 332–346. Springer, 2005.

[4] S. Bhavnani. Important cognitive components of domain-specific search knowledge. *Ann Arbor*, 1001:48109–1092, 2001.

[5] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 85–98, Berkeley, CA, USA, 2005. USENIX Association.

[6] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto*, volume 82, pages 199–203, 1983.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA, 2003. ACM.

[8] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2009.

[9] L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *WONS'09: Proceedings of the Sixth international conference on Wireless On-Demand Network Systems and Services*, pages 133–140, Piscataway, NJ, USA, 2009. IEEE Press.

[10] G. Danezis, T. Aura, S. Chen, and E. Kıcıman. How to share your favourite search results while preserving privacy and quality. In *Privacy Enhancing Technologies*, pages 273–290. Springer, 2010.

[11] G. Danezis, C. Díaz, C. Troncoso, and B. Laurie. Drac: An architecture for anonymous low-volume communications. In M. J. Atallah and N. J. Hopper, editors, *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 202–219. Springer, 2010.

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

[13] G. B. Duggan and S. J. Payne. Knowledge in the head and on the web: using topic expertise to aid search. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 39–48, New York, NY, USA, 2008. ACM.

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

[15] S. C. Han and Y. Xia. Optimal leader election scheme for peer-to-peer applications. In *ICN '07: Proceedings of the Sixth International Conference on Networking*, page 29, Washington, DC, USA, 2007. IEEE Computer Society.

[16] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 9, 2007.

[17] H. A. Hembrooke, L. A. Granka, G. K. Gay, and E. D. Liddy. The effects of expertise and feedback on search term selection and subsequent learning: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(8):861–871, 2005.

[18] D. Heutelbeck and M. Hemmje. Distributed leader election in p2p systems for dynamic sets. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, page 29, Washington, DC, USA, 2006. IEEE Computer Society.

[19] C. Holscher and G. Strube. Web search behavior of Internet experts and newbies. *Computer networks*, 33(1-6):337–346, 2000.

[20] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *WWW '10: Proceedings of the 19th international conference on World wide web*, New York, NY, USA, 2010. ACM.

[21] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving p2p data sharing with oneswarm. In *ACM SIGCOMM*, 2010.

[22] M. Kacimi, S. Ortolani, and B. Crispo. Anonymous opinion exchange over untrusted social networks. In *SNS '09: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 26–32, New York, NY, USA, 2009. ACM.

[23] J. Li and F. Dabek. F2F: Reliable storage in open networks. In *5th IPTPS*. Citeseer, 2006.

[24] M. Mani, W. Seah, and N. Crespi. Super nodes positioning for p2p ip telephony over wireless ad-hoc networks. In *MUM '07: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 84–89, New York, NY, USA, 2007. ACM.

[25] J. A. Pouwelse, P. Garbacki, J. Yang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. V. Steen, and H. J. Sips. Tribler: A social-based peer-to-peer system. In *In The 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[26] K. Puttaswamy, A. Sala, and B. Zhao. Improving anonymity using social links. In *Secure Network Protocols, 2008. NPSec 2008. 4th Workshop on*, pages 15–20, 2008.

[27] L. Ramaswamy, B. Gedik, and L. Liu. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 16:814–829, 2005.

[28] M. K. Reiter and A. D. Rubin. Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.

[29] D. Rossi, M. Mellia, and M. Meo. Understanding skype signaling. *Comput. Netw.*, 53(2):130–140, 2009.

[30] D. R. Sandler and D. S. Wallach. Birds of a fethr: Open, decentralized micropublishing. In *8th International Workshop on Peer-to-Peer Systems (IPTPS '09) April 21, 2009, Boston, MA*, 2009.

[31] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: a protocol for scalable anonymous communication. *J. Comput. Secur.*, 13(6):839–876, 2005.

[32] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using sip. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 63–68, New York, NY, USA, 2005. ACM.

[33] C. Torrey, E. F. Churchill, and D. W. McDonald. Learning how: the search for craft knowledge on the internet. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1371–1380, New York, NY, USA, 2009. ACM.

[34] L.-S. Wu, R. Akavipat, and F. Menczer. Adaptive query routing in peer web search. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1074–1075, New York, NY, USA, 2005. ACM.

[35] Z. Xu and Y. Hu. Sbarc: A supernode based peer-to-peer file sharing system. In *In Eighth IEEE International Symposium on Computers and Communications, Kemer-Antalya*, 2003.

[36] J. Zhang and M. S. Ackerman. Searching for expertise in social networks: a simulation of potential strategies. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 71–80, New York, NY, USA, 2005. ACM.

A Community Creation Protocol

When a node is looking to join a community, it checks to see if there are any initiators located within its friends-of-friends network. If there are no initiators in that range, it becomes an initiator itself. Over time, the network will reform these communities to optimize the initiators to produce clusters of equal size and remove node outliers (nodes without a cluster). This process is done by following the "Two Hop Return Probability" calculation discussed in [27] and comparing it to a Two Hop Probability threshold. The tuning of this threshold value is beyond the scope of this paper. Nodes determine the community they belong to based on attraction values received from their peers about specific initiators keyed by a random initiator ID (I). This process uses a flooding mechanism limited by a TTL. The attraction values are calculated, starting with each initiator I, using the following process:

1. Each I calculates $A_I = \frac{1}{|Edges|_I}$ and sends a message with $TTL = 2$ to all neighbors N.
2. For each $n \in N$, n records A_I. All values received for each I will be added to A_I. n then sends out a message $A_I = A_I / \frac{1}{|Edges|_n}$ with the $TTL - 1$ to all its neighbors N.
3. This process repeats until $TTL = 0$
4. A node then chooses its initiator I which has the highest A_I.

The community creation process will happen over an extended period of time. The bootstrapping process will involve each node getting a list of friends and friends-of-friends from the social networks they belong to. They can then use these lists as the full search space for community creation. An optimal algorithm for the bootstrapping is beyond the scope of this paper as we are primarily concerned with designing Pythia to provide optimal privacy during the Q&A process and assuming that communities are created in the optimal manner. We leave the study of more optimal clustering to future work, but demonstrate here that existing algorithms are viable.