PRIVACY-AWARE DECENTRALIZED ARCHITECTURES FOR SOCIALLY NETWORKED SYSTEMS

Shirin Nilizadeh

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the School of Informatics and Computing,

Indiana University

September 2014

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Apu Kapadia, Ph.D.

Yong-Yeol Ahn, Ph.D.

Nikita Borisov, Ph.D.

Minaxi Gupta, Ph.D.

XiaoFeng Wang, Ph.D.

17 June 2014

To my parents, I would not be who am today without them

To my husband, Majid Deldar, for his endless love and support all the way through my PhD

To my daughter, Auvin, who gives me courage to continue my professional journey

Acknowledgments

I owe my gratitude primarily to my supervisor, Professor Apu Kapadia. I am deeply grateful for so much guidance, encouragement, and his consistent support during my five-year PhD study at Indiana University Bloomington (IUB). I appreciate all the time he has spent discussing research ideas, editing my papers, offering constructive critiques, and patiently mentoring me. It was an honor for me to work with him.

I am truly grateful for the two-year Fellowship that was granted to me by IU's School of Informatics and Computing (SOIC). I would like to give my sincere thanks to my research committee, Professors Yong-Yeol Ahn, Nikita Borisov, Minaxi Gupta, and Xiaofeng Wang. Their insight and advice proved very useful during the preparation of this thesis.

Further, I would like to thank all my co-authors on my very first project related to Pythia, my advisor, Professor Apu Kapadia, Naveed Alam, and Dr. Nathaniel Husted. The experience greatly inspired and motivated me to advance through this research direction. The work on DECENT and Cachet would not have been possible without the efforts of Dr. Sonia Jahid, Professor Prateek Mittel, and Professor Nikita Borisov. For our recent work on community-enhanced de-anonymization of social networks, I would like to thank Professor Yong-Yeol Ahn for his inspiring ideas and helpful suggestions.

I give my thanks to Professor Steven Myers for mentoring me when I just started my PhD. He has been a reliable resource afterwards and always has given me instructive advice. I want to thank Professor Minaxi Gupta, with whom I worked on a project very early in my PhD; though the work is outside the scope of the thesis, the collaborative experience was very much appreciated. Finally, I am very grateful to Professor Jean Camp, who not only provide me with intellectual support, but also guided me in finding a community of researchers to collaborate with inside and outside of IUB. I also value my experience in the Deter group at University of Southern California's Information

Science Institute (ISI). It was enjoyable to work with my mentor and collaborator Dr. James Blythe.

I want to thank the Center for Complex Networks and Systems Research (CNetS) and Dr. Lilian Weng for providing a dataset from Twitter. I am grateful for the technical support team at the SOIC, where Bruce Shei in particular has provided help running extensive experiments on the School's servers and clusters. I also thank John McCurley who patiently edited several of our papers and my dissertation.

All my dear friends and colleagues have offered me a great amount of joy, happiness, and courage along the way. I enjoyed scrum meetings, privacy lab lunches and weekly security reading group meetings with Tousif Ahmed, Zheng Dong, Dr. Vaibhav Garg, Roberto Hoyle, Dr. Nathaniel Husted, Qatrunnada Ismail, Greg Norcie, Dr. Sameer Patil, Zahid Rahman, Dr. Roman Schlegel, Robert Templeman, Dr. Rui Wang, Xiaoyong Zhou, and all other members of the security group. I also thank all my Iranian friends in Bloomington for their kindness and support. You helped alleviate some of the stress of living far away from home.

I would like to express my gratitude to Professor Babak Sadeghiyan, who has been my mentor in all years of my MSc and PhD studies. His advice has helped me to find my way in life.

I am certainly indebted to my parents the most. Their endless love, encouragement, and guidance supported me throughout my education, enduring the physical distance between us. I am grateful beyond words for all that they have given me. I thank my kind siblings, Amir Farhad, Farnoush, and Amir Farzad, who were always ready to talk with me and soothe my homesickness. I would like to remember my grandfather Ali Mohammad Nilipour Tabatabaei, who passed during my PhD research and who was the true supporter of my graduate studies.

I do not even know how to thank my dear husband, Majid Deldar. He trusted me and left all his achievements in Iran to accompany me here in USA so that I could live my dream. He has been through this PhD with me. I felt his love and wholehearted support every day, and I would not have been able to survive without it. I hope we will remember these years forever, together.

Shirin Nilizadeh

PRIVACY-AWARE DECENTRALIZED ARCHITECTURES FOR SOCIALLY NETWORKED SYSTEMS

Online social networks (OSNs) and socially networked applications have become a part of lives of billions of people around the world. The centralized architecture of these networks poses a threat to the privacy of their users who share their personal information with the providers of these services. Building a decentralized, peer-to-peer (P2P) network is an alternative approach to building a social network. P2P networks are inherently resistant to censorship and centralized control. However, providing the same functionalities as OSNs is challenging in P2P networks, because not everyone in these networks is trustworthy, and without having a central authority enforcing users' privacy polices is more challenging.

In order to reveal the unreliability of centralized systems, this dissertation proposes a framework for de-anonymizing users in anonymized social networks by using the community structure of these networks. We show that even a trusted centralized service provider may not be able to provide privacy for its users in this context. Then, by creating three distributed systems– Pythia, DECENT and Cachet – we demonstrate that it is possible to provide a P2P architecture for centralized OSNs, which 1) provides adequate privacy-preserving mechanisms, 2) is resistant to censorship and centralized control, 3) gives users fine-grained control of their data, 4) is scalable, and, 5) is efficient for important applications such as 'newsfeeds' and 'trending topics'. Pythia is the first privacy-aware P2P network for social search where askers anonymously send their questions and experts anonymously answer the questions. DECENT is a more generalized P2P architecture for OSNs that implements basic OSN functionalities and protects the confidentiality, integrity and availability of user content. Finally, Cachet uses a hybrid structured-unstructured overlay paradigm in which a conventional distributed hash table is augmented with social links between users enabling efficient dissemination and retrieval of data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online social networks (OSNs) such as Facebook [5], Google+ [11], Twitter [30], LinkedIn [17], and MySpace [20] have emerged as significant social and technical phenomena over the last several years. Facebook has grown beyond 1.28 billion monthly active users worldwide [105], and Google+ reached the mark of 10 million users in only 2 weeks after going public [116]. Moreover, many applications such as recommendation systems (e.g. Netflix [22], Spotify [29], Hulu [14], Rdio [25], Amazon [1], goodreads [10]), Q&A systems (e.g. Quora [24]), and health applications (e.g. PatientsLikeMe [23], Sermo [27], dailymile [4], Runkeeper [26] leverage these networks to provide their services in novel ways. This extensive use of OSNs and social-based applications have revolutionized the way our society communicates: now users employ these networks to keep in touch with friends and family, discuss current events, find answers for their questions [24], play online games (e.g. Farmvile [6], Mafia Wars [18]), make new connections, get motivation for weight loss (e.g. SparkPeople [28]) or doing exercises (e.g. dailymile [4], Runkeeper [26]), and manage and share their photos(e.g. Flickr [7], Instagram [15]), publications (e.g. Mendeley [19]), and even their health records (e.g. PatientsLikeMe [23]).

However, all these networking sites are centrally managed, and therefore they risk the privacy of users' information. Users are not in control of their data; OSN providers often claim full ownership of all user data [103, 104] while gathering an extensive amount of information about their users and storing it permanently in their databases. This information includes user identity, social relationships, age, location, photos, interests, expertise, group associations, in addition to all of his/her daily activities and expressions of emotions. This information is highly valuable to marketers, governments, researchers and other third-party organizations. For example, marketers would be able to

increase their commercial revenue through targeted advertisement when users' preferences are extracted by analyzing their social contacts, actions, gender, age, and location. This data has also been used by governments and researchers for predicting social and political [80, 177, 205] events and even stock markets [65]. OSN providers also allow program developers to access their platforms in order to create third-party applications such as games, online polls and quizzes. To make these applications useful, developers may be also allowed access to the information of users [103, 104].

Although, there are a few limits (by OSNs' privacy terms and policies) on the ways OSN providers can share and use the information they gather, they may anonymize the data before selling it by removing users' identifiers or adding noise to the data. However, it has been shown [48, 190, 238] that by using some external or prior knowledge, it is possible to re-identify some users and learn about their sensitive attributes. We first demonstrate that even studies underestimate the privacy concerns from de-anonymization by proposing a novel de-anonymization technique that leverages the community structure in social networks that highly decreases users' anonymity even in the existence of large amount of noise and small amount of prior knowledge. This piece of work illustrates that even 'trusted' OSN providers may not be able to provide privacy and anonymity for their users and existing anonymization techniques cannot preserve privacy of users' data.

Overall, potentially an OSN provider is a 'Big Brother' capable of exploiting its users' data in many ways that can violate the privacy of individual users or groups of users. Additionally, as a single point of failure, any vulnerability in the centralized system (or even accidental leaks) could be exploited by a malicious adversary to easily obtain access to the users' data at once.

Building a decentralized, peer-to-peer (P2P) network is an alternative approach that may allow secure and private social networking. In a P2P decentralized network, all computers have equivalent capabilities and responsibilities, in contrast to client/server architectures where some computers are dedicated to serving the others. P2P systems have been used for different purposes: direct and real-time communication (e.g. Chat/IRC and instant messaging, Skype), distributed computation (e.g. SETI@Home [43] and Genome@home [164]), Internet service support (e.g. P2P multicast sys-

tems [232] and Internet indirection infrastructures [225]), distributed databases (e.g. PIER [133]), and, P2P content distribution. P2P content distribution systems such as Napster [21],Kazaa [16], eDonkey/Overnet [125], Gnutella [9], BitTorrent [3], and Freenet [8] were among the most popular applications in this decade. Although some of them including Napster, Kaza and Gnutella had to stop working because of legal concerns, other P2P content distribution systems have been estimated to collectively account for approximately 43% to 70% of all Internet traffic (depending on geographical location) as of February 2009 [216].

P2P networks have been addressed as the future of Internet networking [32, 71, 72, 81, 224]. Two factors have fostered the recent explosive growth of such systems: first, the low cost and high availability of large numbers of computing and storage resources, and second, increased network connectivity. As these trends continue, the P2P paradigm is expected to become more popular. Moreover, P2P networks are inherently resistant to censorship and centralized control which makes them highly desirable for privacy preserving systems. Albeit these advantages, however, providing the same functionalities as OSNs is challenging in P2P networks and raises new privacy concerns that need to be addressed carefully [115, 135, 184]. In existing P2P networks not everyone is trustworthy and untrusted peers can violate data privacy by using data for malicious purposes (e.g. fraud, profiling) [140]. Privacy-preserving P2P systems must provide some mechanisms to protect personal information and files that are stored and shared within the network and guarantee their confidentiality and integrity. Moreover, although no central authority is collecting data in P2P networks, some peers may collude to collect and aggregate data about other peers or even try to identify individual users.

On the other hand, one approach to mediate security and privacy concerns in P2P networks is to leverage trusted social links between users [40, 86, 179, 201]. In this thesis we demonstrate that by using these trusted social links, it is possible not only to provide better privacy, but also more efficient functionalities. Thus, the P2P paradigm and social networks mutually stand to improve one another's efficiency, security, and privacy. Using a P2P architecture for social networks increases

privacy and anonymity, and using "social networks concepts" to construct P2P networks creates more trust between users. The ultimate goal is *to build a privacy-aware P2P social network that is also cost-effective and scalable*. The distributed P2P social network and its algorithms provide an infrastructure for developing as many applications as exist for both social networks and P2P networks.

In this dissertation, we will show the feasibility of using P2P networks for building privacy-preserving systems. In Chapter 2 the prior work related to topics presented in this thesis will be explained. In Chapter 3 the problem and thesis statement are described in detail. In Chapter 4 we illustrate that even having 'anonymized' social network data, the attacker is able to re-identify users and gain access to their private information. For that, we propose a community-enhanced de-anonymization technique that uses community structure in social network to re-identify users.

In Chapter 5 we propose a privacy-aware P2P network for social search. This network is designed to implement a specific social-based application where 'askers' are able to send their questions anonymously to their communities where some experts anonymously answer their questions. We identify the specific security and privacy requirements of such a privacy-preserving P2P social search system including 'users' anonymity,' 'message unobservability,' and 'expertise unlinkability.' The system guarantees that the expertise and interests of users are not identifiable and that users retain plausible deniability, i.e. the anonymity set for users when they ask of answer questions remains large enough so they can deny their actions.

In Chapter 6 we propose a general decentralized architecture for privacy preserving social networking that implements some basic functionalities such as creating and posting status updates, retrieving and viewing walls, and a 'newsfeed.' We also demonstrate that extensive use of cryptographic algorithms decreases performance of some of the functionalities. In Chapter 7 we proposed a social caching and presence protocol that greatly increases the performance of the functionalities. These novel social-based gossiping protocols leverage social trust between users and employ social neighbors as trusted caches to provide content more efficiently. These platforms will facilitate future

work in which we can investigate deploying new multi-purpose applications with different security and privacy requirements.

Chapter 2

Related Work

In this chapter we describe prior work related to the topics presented in this thesis. The related work has been grouped into sections where we 1) cover the work that examines privacy and security in OSNs; 2) talk about topics related to anonymizing and de-anonymizing techniques employed on social network data sets; 3) overview both traditional question and answering systems and social search systems that rely on social relationships to provide more relevant and reliable services; 4) give a summary on P2P networks and their applications; 5) give an overview of existing distributed designs for OSNs and discuss their benefits and limitations; 6) give an brief overview of techniques centralized OSN providers use for providing data confidentiality; and 7) explain information dissemination protocols in P2P networks and how they can be employed on distributed OSNs.

2.1 Privacy and Security Problems in OSNs

Privacy and security are of particular interest specifically in the context of OSNs given their access to sensitive, personally identifiable data involved [67, 119]. The convergence of this data can be dangerous as we discuss next.

Online social networking services include social-interaction-centered sites such as Facebook and MySpace, information-dissemination-centric services such as Twitter, and sites and services to which social interaction features are added such as Flickr and Amazon. Each of these services has different characteristics of social interaction and different vulnerabilities to attacks.

Extensive research has been done on the privacy threats and security exposures of OSNs [108, 119, 128, 142, 207], varying from information leakage [64, 85, 132, 139, 172, 175], identity theft [63, 74, 157], private data collection [132], social phishing [136, 198], spam [203], cross-site scripting,

viruses [79, 106], worms [69, 240], face recognition [119], network inference [64, 85], and link prediction [83, 244, 249]. In Chapter 4 we also show that there is no guarantee that OSN providers can preserve the privacy of their users by using current anonymization techniques.

Due to the centralized architecture of current OSNs, where all data is collected and stored by a service provider, these threats usually have higher impact, because a single attack can violate the privacy of all or many of the users. However, our proposed decentralized P2P architecture for OSNs mitigates the impact of some of attacks such as identity theft, private data collection and network inference. Our design though may be vulnerable to some other attacks such as spam, social phishing or viruses. These attacks are not in the scope of this dissertation and more study is required to evaluate these threats and propose privacy mechanisms that can prevent or mitigate them.

2.2 Anonymizing and de-anonymizing of social networks

Social network data is usually provided to researchers and third-party applications so they can analyze and study various trends using analytical tools like data mining. The explicit and implicit information contained in social graphs is extremely sensitive. To avoid negative consequences for the users involved, OSN providers usually apply anonymization techniques to these data sets before publishing them in such a way that the authorized party can analyze the data without any security breach. The practice of reversing this process where users in an anonymized data set are re-identified is called "de-anonymization". In this section we discuss relevant research related to graph anonymization techniques, de-anonymization attacks based on structure alone, and those based on attributes as well. Finally we discuss applications of network alignment to other fields.

2.2.1 Social Network anonymization

Anonymizing social graph data has attracted significant study recently. Graph data is more complex than relational data; while relational data represents information about entities, social graph data contains information about not only individual users but also sensitive associations between them.

In social network data, node existence, node properties (such as degree), sensitive node labels, link relationships, link weight, and sensitive edge labels [253] can be considered to represent the private information of individuals. Anonymization may have serve different objectives based on the privacy information under attack. Different privacy concerns may lead to different problem definitions and different privacy preservation methods accordingly.

The main technologies for the privacy protection of released social network data have been pseudonymization (use of false names) and anonymization (removing identifiable attributes such as names). However, it has been shown that these techniques are not enough for providing privacy [169, 250]. We will not discuss basic technologies such as ciphers.

Anonymization techniques can be classified into following four approaches [219]: 1) clustering, 2) clustering with constraints, 3) modification of graphs, and 4) hybrid. The *clustering-based* method applies generalization techniques [73, 255] to aggregate edges or node information so that there are many possible mappings from the clustering back to the graphs, which will always include the original. For example, one generalization technique can be the forming of nodes into groups and revealing only the number of edges between pairs of groups [84]. However, this method may not be suitable for applications in which more fine-grained data are required. *Clustering with constraints* [248] merges all nodes of each cluster to a single node. The anonymized graph includes those edges that their equivalence class nodes have the same constraints as any two nodes in the original data. *The modification of graph approach* aims to defeat attacks that try to benefit from known structures in the graph by adding, removing and/or swapping some nodes and edges in a social network. In Chapter 4 we focus on this approach. The *Hybrid approach* includes combinations of any of the above [231, 252, 256].

Many techniques for the anonymization of social networks are based on k-anonymity and randomization. k-candidate anonymity ensures that any individual cannot be re-identified from k other individuals in that graph [123]. k-degree anonymity ensures that every node has at least $k-1$ other nodes in the graph having same degree. This prevents re-identification of individuals by adversaries

who have prior knowledge of the degree of certain nodes [169]. k-neighborhood anonymity ensures that all the nodes are k-anonymous as there are at least $k-1$ other nodes $v_1, ..., v_{k-1} \in V$ such that the sub graphs constructed by the neighbors of each node $v_1, ..., v_{k-1}$ are all isomorphic [168, 251, 252]. This type of anonymization attempts to avoid an adversary who has some knowledge about the neighbors of the victim and the relationship among the neighbors, to re-identify a target victim.

Although these techniques provide protection from structural attacks, they introduce structural changes resulting in information loss. In light of this, Ying and Wu [243] proposed two randomization techniques for privacy protection in a social network graph, 1) In the *random add/deletion technique* randomly false edges are added to the network graph followed by the same number of true edge deletions such that the number of edges remains unchanged. Ying et al. [244] showed that this random add/deletion technique provides protection from both identity and link disclosure whereas k-anonymity can only protect against identity disclosure. 2) In *random switch edges* pairs of edges are swapped such that the nodes degrees remain unchanged. In Chapter 4 we show that applying our community-enhanced de-anonymization approach, the attacker is able to de-anonymize the anonymized graph if 20% of edges of a graph are randomly added/deleted.

To prevent attribute disclosure, Machanavajjhala et al. [171] proposed a stronger privacy technique called "l-diversity", where the main idea is that the values of the sensitive attributes are well represented in each group. Zhou and Pei [252] proposed use of k-anonymity and l-diversity where vertices have to be partitioned into equivalence groups, such that in every equivalence group of vertices, at most $1/l$ of the vertices are associated with the most frequent sensitive labels. The p-sensitive k-anonymity model [227] is also an extension of the k-anonymity method which requires that there be at least p distinct values for each sensitive attribute within the records that share a combination of key attributes.

The OSN providers release users' personal data not only through data sets, but also through their APIs. Beach et al. [54] proposed social-k as a personal social networking API that provides

privacy guarantees for the data that it releases. Social-k is a new extension to k-anonymity. It avoids distorting the social graph structure and instead selectively withholds user profile data from the other applications. Data is partially released wherein all data is quasi-identifiable, i.e. the released data must map to at least k distinct sets of individuals within the dataset.

Mittal et. al. [181] proposed an anonymization algorithm to provide link privacy, where relationships are sensitive while node identities may be known. This algorithm is based on random walks and perturbs the structure of a graph by deleting real edges and adding fake edges based on the structure of the original graph as opposed to random selection so that the local structures in the original social graph are preserved. The goal is to have privacy at the cost of a slight reduction in the utility of applications such as anonymous routing that leverages trust between social relationships while hiding individual relationships. The authors showed that their technique preserves the community structure of the social graph while introducing a significant amount of noise to the network. They also proved that the expected degree of each node after the perturbation algorithm is the same as in the original graph. While their approach is not focused on vertex privacy, our results in Chapter 4 suggest that this technique may also be vulnerable to graph de-anonymization, because the community structures (local structures) are preserved, which makes matching communities possible. Further, the degree distribution of the graph remains the same, which means the seed enrichment phase continue to be applicable. In the future, we will investigate the resistance of this random walk based perturbation algorithm against our community-enhanced de-anonymization approach.

2.2.2 De-anonymization attacks based on structure

De-anonymization attacks based on structure leverage patterns of connectivity in the social network to de-anonymize users. They can be classified into either 'active' or 'passive'. In active attacks, as suggested by Backstrom et al. [49], the adversary, prior to the release of the network, chooses its victims that need to be identified. Then, they create a small number of new user accounts ('Sybils'),

and try to form connections to the victims. Sybil nodes tend to have a distinguishable graph structure, making them identifiable in the whole anonymized graph. It has been shown that it is possible to search for a pattern in the graph and identify both Sybil accounts and the victims when the anonymized network is released. However, as Narayanan and Shmatikov [190] point out, active attacks are not scalable because creating thousands of fake user accounts is expensive and this attack may not be as effective in directed graphs when legitimate users do not link back to the sybil nodes.

On the other hand, in passive attacks, the attacker does not actively modify the network. Backstrom et al. [49] have also shown a passive attack where a small coalition of attackers identifies its location in the released network, and tries to discover the existence of edges among users to whom they are linked. This passive attack is less effective and works at a small-scale, because the attackers do not choose any user as a victim and they can only compromise the privacy of their social contacts. Narayanan and Shmatikov [190] demonstrated the feasibility of large-scale, passive de-anonymization of social networks. As it was explained in Section 4.2, the adversary uses the network structure to re-identify users. They show that about 30% of the verifiable members of Twitter and Flickr could be recognized with 12% error rate. Their work demonstrated the feasibility of successful re-identification based solely on the network topology. Our results show that our community-based approach can boost the performance of their, and in principle any, algorithm under high levels of noise, larger number of nodes, or fewer known seeds.

2.2.3 De-anonymization attacks based on other attributes

Several attacks have been proposed that leverage additional information beyond the structure of the networks. For example, it has been shown [35, 51, 118, 124, 249] that one can exploit private information of users by using public, non-sensitive data about individuals. Wondracek et al. [238] introduced a technique that uses social-network group membership information as well as stolen browsing history to identity users.

Users who are members of multiple social networks may have a public profile in one website

and be more cautious about their information in another one. Identifying users from different websites and aggregating their information reveals more information about them. This technique has been used for providing better recommendation systems and targeted advertisements. However, identifying users even in two public networks is not an easy problem. Lofciu et al. [134] propose strategies based on tagging behavior for the identification of users across systems. Korayem and Crandall [155] used users' activity patterns such as textual, temporal, and geographic properties of users' content as well as their local social connections across different sites to uniquely identify users. In other work, Narayan and Shmatikov have also shown the possibility of de-anonymizing users by aligning multiple datasets leveraging movie preferences [189]. Diaz et al. [96] have also shown that social network members can be identified just by observing their communication patterns between each other.

2.2.4 Network alignment

Network alignment has been of interest in other fields including Biology [159, 221]. In biological contexts, this technique is used to map two protein interaction networks to infer the function of many unknown proteins in human body. The problem is formulated as a quadratic program and solving it is NP-hard. Different approaches [53, 149, 152] have been proposed to relax the constraints or find proper heuristic functions. These approaches have been applied successfully in some applications such as finding common path-ways in biological networks [221, 222] and ontology alignment between Citeseer papers and DBLP papers [131]. However, these networks are of smaller scale compared to social networks and more investigation is needed in the context of large-scale social networks.

2.3 Question and Answering Systems

Locating a person with some specific expertise 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 (`http://makezine.com/`) and eHow (`http://www.ehow.com/`). Torrey et al. [230] study how information is searched for and learned in the traditional web. Adamic et al. [37] 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. Popular Q&A 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.

Danezis et al. show how search preferences can be shared with a select group of friends of the user in a social network [89]. 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. While it is not clear if it can be used to send questions and locate experts, one possibility for future work is to leverage these groups in Pythia. Our protocol is tailored to the orthogonal problem of communicating questions and answers privately.

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].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 [101] 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 [62,127,129]. Aardvark [130] 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. [88] 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. [167] 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 [239]. 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. [82]. We are considering a modification of Sixearch.org software as a Pythia implementation.

Finally, Kacimi et al. [143] present a protocol for 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 5.4.3.

2.4 Peer-to-Peer Networks

Peer-to-peer applications have been classified [44] as:

Communication and Collaboration applications such as Chat and instant messaging applications (AOL, Yahoo, and MSN), employ P2P architecture to provide the infrastructure for direct and real-time communication and collaboration between peer computers.

Distributed Computation applications such as SETI@Home [43], and, Genome@home [164] make use of the available peer computer processing power (CPU cycles).

Internet Service Support systems support a variety of Internet services. For example, P2P multicast systems [232], and Internet indirection infrastructures [225].

Distributed Database Systems such as PIER [133] are built on top of a peer-to-peer overlay network topology.

P2P Content Distribution Systems are designed for the sharing of digital media and other data between users. Most of the current and popular P2P systems are within this category. These applications typically allow individual computers to function as a distributed storage medium by contributing, searching, and obtaining digital content. Napster [21],Publius [235], Gnutella [38], Kazaa [16], Freenet [82], PAST [213], FreeHaven [98], Oceanstore [158], Chord [226], Groove [13], and, Mnemosyne [122].

In practice, many peer-to-peer applications are not fully decentralized. There are many examples of systems that are P2P at the core and yet have some semi-centralized organization in application, such as Usenet [31] and Napster [21]. There are also some systems that employ the notion of "supernodes" (nodes that function as dynamically assigned localized mini-servers) such as Kazaa [165] and also Tor [99] using centralized infrastructure-based guard nodes.

2.5 Decentralized OSNs

Researchers have designed and proposed several decentralized OSNs such as Diaspora [94], PeerSon [70], Safebook [87], LotusNet [40], SCOPE [173], and Persona [50]. These works do not focus on caching or leveraging social links for fast and efficient data retrieval, but address privacy either through cryptography, architectural modifications, or decentralization of the provider. We show that in the absence of social caching, the performance overhead due to cryptography and decentralization is high.

Diaspora is a social network that users install on their own personal web servers without support for encryption. We note that Diaspora is a deployed system with several hundreds of thousands of users [245] and demonstrates the feasibility of large scale decentralized approaches to social networking. Backes et al. [47] present a core API for social networking, which can also constitute a plug-in for distributed OSNs. Their primary focus is on an API that supports anonymous data access in a distributed OSN. However, they assume that the server is trusted with the data for access control. PeerSon, LotusNet, Safebook, and SCOPE benefit from DHTs in their architecture. PeerSon and

Safebook suggest access control through encryption, but they fall short in providing fine-grained policies compared to ABE-based access control in Cachet. Moreover, in all of these schemes, the overhead of key revocation affects performance whereas revocation in Cachet is efficient through the use of a semi-trusted proxy.

In LotusNet, which is based on Likir [41], the authors consider the distributed storage to be trusted and do not perform encryption. Likir uses signed grants to specify permissions and provide access control. Safebook is based on a peer-to-peer overlay network named "Matryoshka". The end-to-end privacy in Matryoshka is provided by leveraging existing hop-by-hop trust of the links. In contrast to using hop-by-hop trust for data lookup and privacy, we leverage trust relationships to improve performance and ensure privacy using cryptographic techniques.

SCOPE is a distributed data management system for specialized P2P social networks. Clients connect to and store data on a group of super-nodes with higher computation and storage capacity. However, clients do not participate in the DHT; only the super-nodes run the DHT code. Clients connect to super-nodes and rely on them for sharing and access control on their data.

Persona [50] combines ABE with a decentralized OSN architecture to ensure data confidentiality. However, Persona does not support fine-grained policies and lacks suitable revocation mechanisms [61]. Cachet uses an extended version of EASiER [137] with support for access delegation while providing extremely fine-grained access control. In addition, Cachet provides user-verifiable data integrity using digital signatures. Persona is not built upon a DHT; users and applications use a storage service hosted on a dedicated storage server or a user's own storage server. The storage service authenticates write operations through the requester's public key and hence can learn the user's social contacts. Unlike Persona, Cachet separates the policy used to encrypt the data from the ciphertext itself, thus preventing the storage nodes or a third party from inferring a user's privacy policies by getting access to a ciphertext. Persona uses a multi-reader-writer service for the wall, but lacks a protocol for commenting on wall posts. Cachet provides a full design and implementation to read, write, and comment on walls or any data that appears on a wall.

2.6 Access Control in Centralized OSNs

Some techniques such as Scramble [55], FlyByNight [170], NOYB [120], xBook [220] and Frientegrity [107] leverage a centralized provider for maintaining the overall functionality of an OSN but encrypt messages to keep them confidential from the provider. These approaches, however, allow the OSN provider to monitor the interactions of users, censor or remove content by users, and even regulate who can be part of the network. Cachet and the other decentralized solutions attempt to democratize such systems by getting rid of such a powerful centralized provider.

Scramble [55], for example, is a Firefox extension that allows centalized OSNs' users to enforce access control over their data. Although using Scramble, OSN providers do not have access to users' private information and can still infer their social activities and relationships. They also have the power to censor (remove) user generated content from the OSN. On the other hand Cachet is completely decentralized and no information is obtained by an OSN provider.

2.7 Information Dissemination in P2P Networks

One approach for disseminating information in a network is based on gossiping techniques. Mostly, this approach has been applied for disseminating information through wireless networks [58, 111, 112,182] and P2P networks [145,146,254]. However, very few have done research on dissemination of information through decentralized social networks.

Datta and Sharma propose GoDisco [92], a gossip-based decentralized mechanism in which information can be disseminated by using social links and exploiting semantic context. This mechanism is targeted at probabilistic publish/subscribe systems where a vector of interest categories is attached to each message, and information is broadcast to receivers who may be interested in the message. This mechanism is thus orthogonal to our work — while GoDisco disseminates *public* information to *interested* parties, Cachet focuses on disseminating *private* information to *authorized* parties.

Abbas et al. [33] propose a basic gossip-based protocol for establishing friendship links in a

distributed social network considering network dynamics. However, many requirements of social networks, such as dissemination of updates, availability of data, and privacy were not in the scope of their work.

Mega et al. [178] show that applying gossiping algorithms for disseminating users' updates through a P2P social network is feasible. They focus their analysis on the coverage of disseminated updates to the social network, average latency for an update to reach a destination, and the average load in terms of messages sent and received. They do not consider privacy or access control in their design and updates are pushed to all friends and FoFs. In addition, their system relies purely on gossiping protocols for disseminating updates through the social network, which has several drawbacks; for example, there is no guarantee that all updates will be available over time, and a large amount of redundant information is passed around and stored in the network. In Cachet though, updates are stored in the DHT so available over time and friends cache updates for a short time.

Carrasco et al. [75] address the problem of loading newsfeeds efficiently in *centralized* social networks where data is stored in distributed databases (e.g., at data centers). They propose partitioning social network based on users' activities over time so that data belonging to users in a partition can be stored in a way that improves locality. The proposed solution implicitly assumes that a centralized management system keeps track of all users over time to facilitate on data partitioning, but such information is not available in decentralized social networks. Nevertheless, distributed algorithms to improve locality of information in the context of decentralized social networks could improve newsfeed performance, and we leave such an exploration to future work.

Chapter 3

Problem and Thesis Statement

3.1 Problem Formulation

Online Social Networks (OSNs) such as Facebook [5], Google+ [11], Twitter [30],LinkedIn [17]and MySpace [20] have revolutionized the way people communicate socially. Each of these OSNs has billions of users and recent studies indicate that users actively spend large amounts of time on these networks [228]. In addition, several socially networked applications including recommendation systems (e.g. social search on Bing [2], Netflix [22], Hulu [14], and Rdio [25]), Q&A systems (e.g. Ask Question on Facebook, and Quara [24]), and fitness applications (e.g. Runkeeper [26] and dailymile [4]) are gaining popularity. People employ these networks extensively for various purposes, from being in touch with their friends and family, to sharing their activities, photos, and publications, and even getting motivation for their fitness activities.

Current OSNs are run on logically centralized infrastructures where a central repository stores user and application data. These centralized OSNs have several drawbacks including scalability, dependence on the caprices of a single provider, and lack of adequate user privacy. Privacy problems in OSNs have recently received much attention [119, 156, 229, 249]. OSN operators have the sole authority to control all users' data while gathering an extensive amount of data about them. They also often claim full ownership of all user data. In addition, users usually have little control over how and what information about them are presented to the world, because it mostly depends on the design of these services. Therefore, abetter alternative to centralized OSNs is desirable.

Problem statement. *Online social networks and socially networked applications have become part of billions of people's lives all around the world. The centralized architecture of these networks*

poses a threat to the privacy of their users who share their personal information not only with a limited number of their friends but also with the providers of these services. A comprehensive alternative architecture is needed that 1) preserves user privacy, 2) avoids censorship and centralized control, and 3) gives users control over their data.

3.2 Thesis

We cannot just assume that OSN providers are trustworthy because they use and share users' data not just with limited number of friends. Actually, because of their business model, they share data with other parties such as marketers who use it for targeted advertising. Even if they attempt to provide some kind of privacy by, for example, anonymizing the data and removing identifiers, the social network itself and users' attributes reveal too much information which may be employed for de-anonymizing them.

Building a decentralized, peer-to-peer (P2P) network is an alternative approach to building a social network. Additionally, P2P networks are highly desirable for privacy-preserving systems, because they are inherently resistant to censorship and centralized control. P2P content distribution systems such as Napster [21], Kazaa [16], eDonkey/Overnet, [125] BitTorrent [3], and Gnutella [9] have been among the most popular applications of this decade. Two factors have fostered the recent explosive growth of these networks: first, the low cost and high availability of large numbers of computing and storage resources provided by peers; and second, increased network connectivity.

Beyond these advantages, however, providing the same functionalities as OSNs is challenging in P2P networks and raises entirely new privacy concerns that need to be addressed carefully. In existing P2P networks, not everyone is trustworthy and network traffic is sometimes interpreted as hostile. In addition, without having a central authority, enforcing users' privacy polices is more challenging.

One approach to mediate security and privacy concerns in P2P networks is to leverage trusted social links between users [40, 86, 179, 201]. Thus, the P2P paradigm and social networks mutu-

ally stand to improve one another's efficiency, security, and privacy. Using P2P architecture for social networks increases privacy and anonymity and using social networks concepts to improve P2P networks creates more trust between users.

Recently, some designs have been proposed for decentralized OSNs [40, 50, 70, 87, 94, 173], each addressing a few aspects of such a complex system. Some [94, 173] explore the organization of nodes in a distributed network and determine where the content should be stored, while a few others [40, 50, 70] focus on privacy and access controls. However, none completely satisfies all properties mentioned in the problem statement.

Thesis Statement. *As an alternative to centralized OSNs it is possible to provide a comprehensive P2P architecture that 1) provides adequate privacy-preserving mechanisms, 2) is resistant to censorship and centralized control, 3) gives users fine-grained control of their data, 4) is scalable, and, 5) is efficient for important applications such as newsfeed and trending topics.*

3.3 Dissertation Road Map

This thesis addresses the following privacy and feasibility issues:

3.3.1 Community-enhanced de-anonymization of online social networks

Almost all OSN providers offer their services to users for free and users have no choice but to trust the service provider to properly handle their data and preserve their privacy. However, in fact, OSN providers sell the information that their users provide. They may attempt to anonymize the data before selling it. However, "a malicious attacker" may try to de-anonymize the data for several purposes. For example, she can get the anonymized/non-anomymized data sets from several private/public sources and try to de-anonymize users, and enrich their profiles by aggregating their information from these different sources. This data later can be used for more powerful targeted advertising. It is clear that when an OSN user loses her anonymity, she also loses her privacy

because some of her attributes and activities that were provided by the anonymized data set will be revealed.

In Chapter 4, we will first show that the OSN providers can threaten the privacy of users by sharing what they believe to be 'anonymized' data. Using external knowledge about the users, it has been shown [49, 151, 190, 238] that users in the dataset can be de-anonymized. For example, Narayanan and Shmatikov have shown how the social structure from one site, such as Flickr, can be used to re-identify some anonymized users on another site, such as Twitter [190]. We show that using structural properties of social networks such as their community structures, the attacker does not need to have much prior external information to initiate the de-anonymization process. In addition, we show that our approach is effective even if the service provider applies more complicated anonymization techniques such as edge perturbing. Our approach leverages 'community detection' techniques to partition the networks into 'communities' i.e., dense subgraphs that capture social structure [109, 113]. First, it maps the community structure of two graphs then, applies the network mapping technique to the nodes inside each community along with a 'seed enrichment' phase and finally to the entire graph. Using our approach, even if a user cannot be re-identified, its community may be identified and thus its degree of anonymity decreases more. We define and calculate the degree of anonymity when performing our de-anonymization approach as well as Narayanan and Shamatikov (NS)'s algorithm. Then, we run extensive simulations on both synthetic and real social network datasets, and show that our approach provides a significant improvement to NS algorithm where the degree of anonymity significantly decreases using our community-enhanced de-anonymization algorithm compared to the NS algorithm.

3.3.2 A privacy-aware P2P socially networked question and answering system

Most privacy concerns of OSNs' users can be diminished if they are developed upon a privacy-aware decentralized P2P architecture that avoids centralized control and satisfies privacy requirements of OSNs' complex functionalities.

In Chapter 5 we first show the feasibility of deploying a privacy-aware P2P network for a social search system. The idea is to allow users to pose questions to their social network in real time and to obtain answers from real humans. Centralized social search [24], however, do not provide adequate privacy because they observe and collect users' profiles and their communications with others in the social network. Thus, users may avoid asking or answering questions related to sensitive topics such as health, political activism, or even innocuous questions which may make the querier seem ignorant. We also define the privacy requirements that are specific to this application including expertise and interest unlinkability, unobservable querying and responding, and sender anonymity.

We propose Pythia, a decentralized architecture for social search that utilizes the community structure of underlying P2P social network to locate experts who are willing to answer questions. These communities are also providing 'k-anonymity' for both askers and answerers because all questions and answers are flooded inside them. Each community has a representative who is responsible for flooding questions and answers on behalf of users in the community. As a building block, Pythia uses Onion routing [114] to deliver questions/answers from users to the representatives. All the messages in the system are padded to have the same size to thwart traffic analysis of messages in transit. To provide asker/answerer unobservability, all users ask and answer questions at regular intervals (including dummy questions and answers) and thus attackers cannot readily pinpoint which users are forwarding, asking or answering questions. If no answerers are found in the local community, questions are forwarded to a remote community as part of a *remote flood*. The size of communities is a system parameter, and represents a trade-off between privacy and performance. Smaller communities provide lower privacy, but ensure that the overhead of query flooding is a low constant.

We evaluate our architecture through extensive simulations while implementing statistical *intersection attacks* that attempt to identify actual asker/ answers and their expertise over time based on the topics of questions/answers and users' churn rate. We analyze the degradation of anonymity of experts as they field questions over time. Finally, we show that time-aggregation based defenses

improve the anonymity of participants.

3.3.3 Decentralized P2P architecture for privacy-preserving social networking

In Chapter 6 we examine the design of a fully decentralized (peer-to-peer) OSN, with a special focus on privacy and security. We discuss the privacy requirements of a generalized P2P OSN and explain the main architecture, object design, access policies, and cryptographic mechanisms to enforce the privacy policies. In particular, our goal is to protect the confidentiality, integrity, and availability of user content. To provide availability of data, it is replicated and stored in a distributed hash table (DHT). Each data object is also digitally signed by their owners so that any changes to the data can be detected. Use of attribute-based encryption (ABE) on user data along with the object design provide confidentiality and integrity, as well as support for flexible attribute policies and fast revocation. In other words, our design ensures that data is not visible to unauthorized users.

Then, we show that it is feasible to design a decentralized architecture for privacy-preserving social networking and implement some basic OSN functionalities such as creating and posting status updates, retrieving and viewing walls, and a 'newsfeed'. We evaluate our design through simulations as well as experiments on the PlanetLab network and show that a P2P decentralized OSN is able to replicate the main functionality of current centralized OSNs with manageable overhead.

3.3.4 Social Caching for Decentralized P2P Social Networks

The ultimate goal is building a privacy-aware, decentralized P2P social networking infrastructure that avoids any centralized control and is both cost-effective and scalable. In our preliminary design presented in Chapter 6, we showed how the confidentiality and integrity of data can be protected by a cryptographic mechanism so that they can be stored in untrusted storage nodes of a DHT. However, the results for viewing the newsfeed indicate that the design suffers from performance issues that arise due to the fetching and decryption of hundreds of small objects belonging to friends, which is required for viewing their walls or for viewing one's own newsfeed. Therefore, optimization

techniques should be employed to make the system more efficient.

In Chapter 7, we propose Cachet, a decentralized architecture for social networks that provides strong security and privacy guarantees while efficiently supporting the central functionality of OSNs. For efficient dissemination and retrieval of data, Cachet follows a hybrid structured-unstructured overlay paradigm in which a conventional distributed hash table is augmented with social links between users. We propose a gossip-based social caching algorithm along with a presence protocol that rely on trust between social contacts to act as trusted caches. Social caches immediately propagate the new updates to their online social contacts who are also allowed to view those updates. When an offline user comes back online a presence protocol is used to locate online contacts and query them directly for updates. Additionally, these contacts provide cached updates from mutual contacts who are offline as well as the list of their mutual online contacts. After contacting some online social contacts, the user usually obtains almost all updates and the newsfeed can be generated based on those. The DHT is then used to retrieve updates that may not be cached, ensuring high availability of data. Using this gossip-based social caching and presence protocol highly reduces both the number of cryptographic operations that a users should perform to view the newsfeed and communication overhead in the network.

We built a prototype implementation of Cachet in the FreePastry simulator [213]. We show that the optimizations highly increase the performance of obtaining the wall and newsfeed from hundreds of seconds in the base architecture to less than 10. Our architecture thus demonstrates how a careful combination of several distributed systems and cryptographic techniques can be used to provide a compelling privacy-preserving alternative to centralized OSNs.

Chapter 4

Community-Enhanced De-Anonymization of Online Social Netowrks

In any OSN there is a wealth of information about its users embedded in the social graph. OSN providers attempt to anonymize the valuable their data sets before selling to or sharing them with third parties. For example, they may remove people's identities and/ or add some noise to the data by modifying relationships and attributes to a certain extent. However, these techniques may not be successful, as it has been shown [49, 151, 190, 238] that using some external knowledge about the users, the attacker can 'de-anonymize' users in the anonymized social network. For example, Narayanan and Shmatikov [190] have shown using the publicly available social structure from Flickr, one can re-identify anonymized users on Twitter. If the attacker has access to additional information such as users' public attributes, de-anonymization is even easier [154, 189].

In this chapter we study the problem of de-anoymization where the two networks have a high level of overlap, and the goal is to map the nodes in the anonymized network to the nodes with identities in the reference network. This problem is also referred as "network alignment", which is closely related to the graph isomorphism problem [110]. The graph isomorphism problem assumes that two graphs are isomorphic and the goal is about finding the perfect bijection between two graphs. The network alignment problem though assumes that two graphs may not be isomorphic and the goal is about finding the best possible projection between two networks. In general the graph isomorphism problem is in complexity class NP. However, the de-anonymization techniques usually use some prior knowledge and heuristics to solve the problem efficiently.

Contributions:

1. Our approach leverages 'community detection' techniques to partition the networks into

'communities', i.e., dense subgraphs that capture social structure [109, 113]. Our proposed approach divides the problem into smaller sub-problems that can be solved by leveraging existing network mapping methods recursively on multiple levels. First, our approach maps the community structure of two graphs (which may overlap imperfectly) by considering the community structure as a coarse-grained graph. It then applies the network mapping technique to the nodes inside each community (along with a 'seed enrichment' phase) and finally to the entire graph.

2. Through extensive simulations on both synthetic and real social network datasets, we suggest that our 'community boosting' technique that is based on 'mesoscopic'[1] properties of social networks, generically provides a significant improvement to microscopic mapping algorithms, such as the one by Narayanan and Shamatikov (subsequently we refer to their algorithm as the 'NS' algorithm) [190], and enhance their performance in a way that is more robust to noise and large network sizes.
3. We also define and compute the 'degree of anonymity' of users in the graph and analyze how it is reduced by our technique. Even when the explicit mapping of *nodes* is far from complete, we show that the mapping of *communities* greatly reduces the degree of anonymity of users, since the probability distributions of potential mappings results in far less uncertainty than a uniform random mapping. For example, we show that for our Twitter dataset with 90K nodes, with 15% edge noise and 16 seeds, the NS technique reduces anonymity by 2.6 bits (with 33% nodes explicitly mapped), whereas our approach reduces anonymity by 13.17 bits (with 65% nodes mapped). For the same dataset with partially overlapping networks, with 10% edge and vertex noise and 16 seeds, the success rate is about 49% and 29% for community-aware and community-blind algorithms respectively; the anonymity is reduced by more than 7 bits from 14.77 to 7.08 bits.

This chapter is based on collaborative work with Yong-Yeol Ahn and Apu Kapadia and is published at the 21st ACM Conference on Computer and Communications Security (CCS) [194].

[1] "Mesoscopic" is a term in physics used to refer to a granularity between 'microscopic' and 'macroscopic'.

4.1 Definitions and attack models

In general, de-anonymization is defined as "a data mining strategy in which anonymous data is cross-referenced with other data sources to re-identify the anonymous data source".[2] The idea is to collect enough information about an anonymous individual and enrich his/her profile. This profile can then be linked to other public information to attach an identity to the data. "Social-network de-anonymization" usually refers to the problem of cross-referencing two or more social graphs to enrich anonymous users' profiles and re-identify them. Some attacks only use the network structures, while others exploit user attributes, such as user names and group memberships [154,190, 238]. We now formalize the models and definitions used in our work.

4.1.1 Definitions and assumptions

We interchangeably use the term "network", "node", and "link" with "graph", "vertex", and "edge", respectively. All networks we use in this paper are undirected.

Definition 1 *A graph, $G\langle V, E\rangle$, is a set of vertices V that represents the users in the network and a set of undirected edges $E \subseteq \{e = (u, v) : u, v \in V\}$ that represents links between users. In a social network, for example, edges would correspond to social relationships. We denote the degree of a node by k_v. Let $N = |V|$ be the total number of nodes in G.*

Definition 2 *A graph G's community structure (C) is a disjoint partition of vertices in G, namely $C = \{c_1, c_2, \ldots, c_k\}$, where $c_i \neq \varnothing$ and $c_i \cap c_j = \varnothing$ if $i \neq j$ for $i, j \in \{1, 2, \ldots, k\}$. While there are many alternative definition of communities [39, 109, 196], in this paper communities are defined by Infomap algorithm [212], which finds a partition that minimizes the average number of bits per step required to describe trajectories of random walkers.*

[2] http://whatis.techtarget.com/definition/de-anonymization-deanonymization

4.1.2 Attack model

Online social network providers release anonymized social networks to third-parties for various purposes, including targeted advertising, developing new applications, academic research, public competition, etc. [119, 208]. We assume the recipient of this data, if malicious, may try to de-anonymize the social network by explicitly mapping nodes in the extreme case and/or reducing the uncertainty of mappings to the greatest extent possible.

We assume the adversary has access to two networks, $G\langle V, E\rangle$ and $G'\langle V', E'\rangle$, where $V \cap V \neq \varnothing$ and $E \cap E' \neq \varnothing$. We focus on the cases where $V \approx V'$ and $E \approx E'$, i.e., where the vertices and edges are approximately the same. The difference in these sets is characterized more formally by a 'noise' parameter (see Section 4.5.2).

One of these networks is anonymized and contains sensitive private information associated with the (anonymized) nodes in the graph. The goal of an attacker is to align the anonymized network with the other 'reference' network, re-identify anonymized users, and reveal the private information obtained from the anonymized network. If both networks are anonymized, the problem changes from re-identification to profile enrichment where the attacker tries to align two networks and collect more data about the anonymous users.

4.2 Background

4.2.1 Re-identification algorithm by Narayanan and Shmatikov (NS)

Our algorithm is designed to leverage existing mapping methods. For our evaluation, our algorithm is built upon the re-identification algorithm by Narayanan and Shmatikov [190]. Their algorithm runs in two stages: 'seed detection' and 'propagation'. In the seed-detection step the algorithm maps a small number of users (seeds) between two networks by searching for unique subgraphs. The propagation step expands the set of matched users by incrementally comparing and mapping the neighbors of the previously mapped seeds.

Seed identification

Narayanan et al. [188, 190] have proposed two seed-identification algorithms. The first one randomly samples a subset of k-cliques from the reference graph and finds the corresponding cliques in the other graph. For a chosen clique, the algorithm examines the degree sequence of the k nodes in the given clique and the number of common neighbors between each of $\binom{k}{2}$ pairs of users. For each candidate clique in the other graph, the algorithm compares the two sequences and decides based on an error parameter, θ, whether they are the same people or not. We use a similar approach for identifying initial seeds, which also help in mapping communities (see Section 4.3.1).

The other seed-detection algorithm matches a small subset of high-degree nodes (hubs) in two graphs. This approach is based on the observation that the large hubs are unique and thus can reliably be matched across graphs. The algorithm first identifies a small number of hubs in two graphs, then it builds a weighted network of hubs where the weights represent the cosine similarity of their in-neighbors. The algorithm matches two weighted graphs using a simulated annealing optimization [141] technique.

Propagation

In the propagation step the algorithm expands the set of identified seeds. In each iteration, the algorithm randomly picks an already mapped node pair (u, u'), where $u \in V$ and $u' \in V'$. From the set of u's unmapped neighbors, it picks a random node v then compares it with each unmapped node (v') in the set of u''s unmapped neighbors. The similarity $\mathcal{S}$ between v and v' is defined as the number of v's neighbors that are already mapped to the neighbors of v', divided by the square root of its degree, namely

$$\mathcal{S}(v, v') = \frac{|\{(w, w') : w \in \mathcal{N}(v); w' \in \mathcal{N}(v'); \text{and } (w, w') \in M\}|}{\sqrt{k_v k_{v'}}},$$

where $\mathcal{N}(v)$ is the set of v's neighbors. After calculating $\mathcal{S}$ for all potential candidates (the unmapped neighbors of u') and creating an ordered list of the scores (L), the algorithm identifies the best (v'_1) and the second-best candidate (v'_2) that have the highest scores. v'_1 is accepted as the counterpart of v if its score is sufficiently better than that of v'_2. The uniqueness of the best candidate is measured by 'eccentricity' as defined by

$$ecc(L) = \frac{\mathcal{S}(v, v'_1) - \mathcal{S}(v, v'_2)}{\sigma(L)},$$

where L is the ordered list of the similarity scores ($\mathcal{S}$) of the unmapped neighbors of u', and $\sigma(L)$ is the standard deviation of the values in L.

4.2.2 Community Detection

Community detection (or graph partitioning) has received much attention from various fields, because community structure is a common characteristic of a wide variety of networks across domains, and communities usually correspond to important subunits of the systems. For instance, the communities in social networks correspond to social circles and those in biological networks correspond to functional modules. Originally, graph partitioning was introduced to solve the problem of optimal allocation of processes in a distributed computing context [147]. Since then, graph partitioning and community detection have been actively studied across disciplines [109, 199, 215]. Although there is no concrete definition of a community that is agreed upon, "communities" usually refers to groups of nodes (people) that are densely connected to each other while having lesser connections to nodes residing outside of the community. A large number of community detection methods have been developed and they are widely applied to many domains of science [39, 109, 210].

Although it is known that communities often overlap [39, 196], we use disjoint, non-overlapping communities to simplify the problem. Among the variety of community detection methods, we employ the Infomap algorithm [211, 212] here, because it is one of the most widely accepted disjoint community detection algorithm; it was shown to excel in tests using synthetic benchmark

networks [161, 162]. Also note that our goal is slicing the network into smaller, dense chunks, which may not correspond to meaningful social groups. In principle, myriad other community detection (or graph partitioning) approaches can be adopted to our framework, although we leave such exploration to future work.

4.2.3 Degree of Anonymity

Pfitzmann and Kohntopp [197] defined "anonymity" as the state of being not identifiable within a set of subjects, the anonymity set. Chaum [77] first defined the anonymity set as the measure of anonymity. This measurement has been used by several networks that provide anonymity for senders or receivers of messages [59, 148, 192]. Anonymity set size, however, does not take into account that different members in the set may have different probabilities in sending/receiving the messages. Based on a particular attack, also, these probabilities may differ. Serjantov and Danezis [217] and also Diaz et al. [95] used entropy to define degree of anonymity achieved by the users of a system towards a particular attacker. This measurement depends on the distribution of probabilities and not simply the size of the anonymity set. The entropy of the system after the attack is compared against the maximum entropy in which all N users are equally probable as being the originator of the message with probability $\frac{1}{N}$. The entropy of the system is defined as:

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i,$$

where $H(X)$ is the entropy of the network, N is the number of nodes in the network, and p_i is the probability associated with node i. As the maximal entropy is $H_{\max} = \log N$, the attacker's information gain is $H_{\max} - H(X)$, and thus the degree of anonymity is defined as the normalized entropy of the system:

$$A(X) := 1 - \frac{H_{\max} - H(X)}{H_{\max}} = \frac{H(X)}{\log N},$$

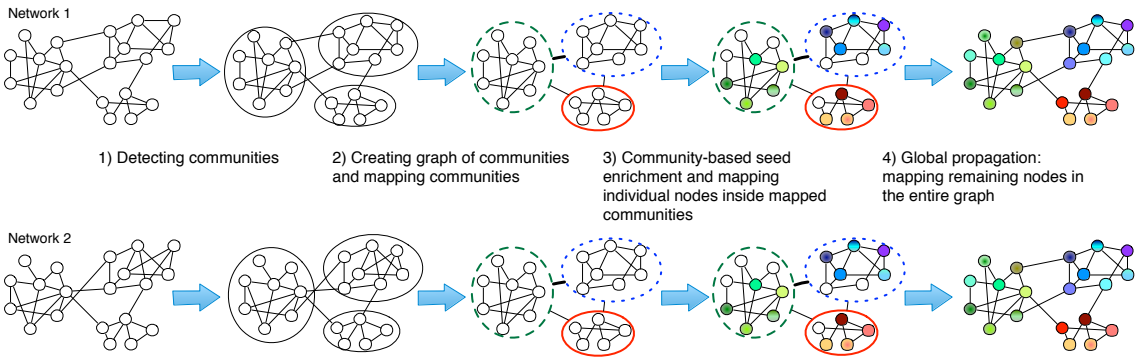


Figure 4.1: An overview of our approach where 1) each of two social graphs is divided to smaller partitions — communities; 2) communities of these two graphs are mapped; 3) nodes inside mapped communities are matched; and 4) NS propagation algorithm runs on the whole network to map remained unmapped nodes.

where $0 \leq A(X) \leq 1$. This value quantifies the amount of information the system is exposing. For instance, $A(X) = 1$ indicates that the users in the network are completely anonymous. $A(X) = 0.5$ indicates that 'half the bits' of privacy are lost compared to the uniform distribution, which corresponds to $\log N$ bits of privacy.

4.3 Our Approach: Community-enhanced De-anonymization

The key notion of our community-based de-anonymization framework is that network communities provide an effective way to divide-and-conquer the problem of de-anonymization, particularly because communities are known to capture meaningful, mesoscopic structural relationships even in the presence of noise [161]. After dividing the reference and anonymized graphs into communities, these communities can be mapped first, and then matching and re-identifying users can be performed for each corresponding community. In particular, our approach can employ an existing 'community-blind' mapping algorithm such as the NS algorithm for *community- and node-level* mapping. Our approach is thus a 'community-aware' mapping algorithm built using community-blind mapping algorithms.

Figure 4.1 illustrates our proposed network alignment framework. Our algorithm has four steps: 1) community detection, 2) community mapping, 3) seed enrichment, and 4) global propagation. These steps are explained next.

4.3.1 Community Mapping

The first step of our algorithm is detecting communities. As mentioned in Section 4.2.2, we use Infomap to slice both the reference and the anonymized networks into smaller, dense chunks; however, any community detection (or graph partitioning) approach can be adopted to our framework. The next step is to map communities that are found in the previous step. Our approach uses two strategies to map communities: (1) using already known seeds and (2) using the network of communities. Once some communities have been mapped (forming seeds at the community level), the community-blind propagation algorithm is applied to the community graph to expand the set of mapped communities.

Identifying seed communities

Before communities can be mapped, the propagation algorithm needs some pre-identified seed mappings. After detecting communities in the two networks, communities associated with seed nodes can be mapped to each other. However, conflicts are possible when two seeds with two different communities in the first network are mapped to one community in the other network. Conflicts are minimized by simply counting the number of times that two communities are mapped together. For each community in the reference network, all possible mappings are listed based on counts in descending order and this community is mapped to the community on top of the list. There are some cases where one community is mapped to two different communities with the same scores. In these cases, a mapping is picked at random.

Mapping communities by creating a network of communities

The community structure itself can be considered as a high-level, coarse-grained graph; we consider each community as a node and the connection between communities as edges. This perspective allows us to directly reuse community blind mapping algorithms such as the NS algorithm to map communities.

Given a graph G, we create a weighted undirected graph of communities, G^*, where each community is a node and a weighted edge between two communities represents the number of connections between nodes in two communities.

In our framework, a community-blind mapping algorithm is run on the network of communities and is fed with some seed communities. Since we use NS in our evaluation, we propose a slight improvement to NS propagation algorithm to exploit the weights in the graph of communities. As in the original NS algorithm, our 'weighted propagation' algorithm starts with two graphs G_1^* and G_2^*. At each iteration, the algorithm randomly picks a neighbor (μ^*) of already-mapped seeds ($\mathcal{U}^*$). We modify the similarity score function so that it includes the weight of edges in the weighted graphs. We tested different similarity functions and found the following to be more effective:

$$\tilde{\mathcal{S}}(\mu^*, \nu^*) = \frac{\sum_{(p^*,q^*)\in\mathcal{N}(\mu^*,\nu^*)} (1 - |\sqrt{w_{(\mu^*,p^*)}} - \sqrt{w_{(q^*,\nu^*)}}|)}{\sqrt{d(\mu^*)d(\nu^*)}} \tag{4.1}$$

where $\mathcal{N}(\mu^*, \nu^*)$ is the set of already mapped pairs among the neighbors of μ^* and ν^*. $w_{(\mu^*,p^*)}$ is the weight of the edge between μ^* and p^*.

Figure 4.2 illustrates an example where the mapping algorithm tries to align node A in the left graph to a node in the right graph. Three of A's neighbors are already-mapped and the algorithm starts with them and computes the similarity score for each neighbor of the mapped nodes in the right graph, i.e. $\tilde{\mathcal{S}}(A, B)$ and $\tilde{\mathcal{S}}(A, B')$. Using the score function of the original NS propagation algorithm, $\tilde{\mathcal{S}}(A, B) = \tilde{\mathcal{S}}(A, B')$ and the algorithm cannot map A to any node in the right graph. By contrast, here $\tilde{\mathcal{S}}(A, B)$ would be larger than $\tilde{\mathcal{S}}(A, B')$ and thus A is correctly mapped to B (with a suitable eccentricity threshold).

4.3.2 Seed enrichment and Local propagation

One of the major benefits from the community decomposition and mapping is that additional seeds can be identified. Seeds are usually identified based on their uniqueness at a global level; communities offer a much more narrow search space for seeds, which may have otherwise not looked unique

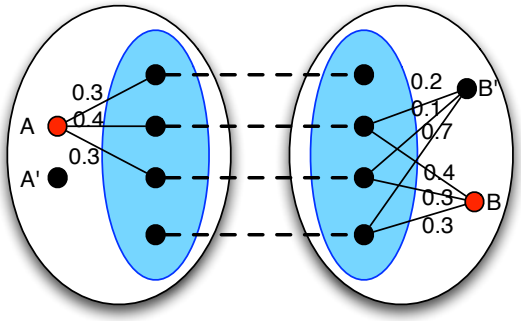


Figure 4.2: An example of mapping nodes of two weighted undirected networks using the "weighted propagation" algorithm. The numbers on the edges show the edge weights.

at the global scale. We call this step of finding more seeds leveraging community information 'seed enrichment'. Following seed enrichment, the community-blind mapping algorithm is applied to each pair of matched communities using the enriched set of seeds.

We propose the following approach to identify seeds at the community level, which is based on two distance metrics defined over nodes' degrees (d), and the clustering coefficients (cc):

$$D_d(v_i, v_j) = \frac{|d(v_i) - d(v_j)|}{max(d(v_i), d(v_j))}$$

$$D_{cc}(v_i, v_j) = \frac{|cc(v_i) - cc(v_j)|}{max(cc(v_i), cc(v_j))}$$

The clustering coefficient is a property of a node in a network and quantifies how close its neighbors are to being a clique. It can be quantified as the fraction of pairs of the node's neighbors that are connected to each other by edges. The clustering coefficient is between zero and one; if the neighborhood is fully connected, it is 1, and if there are few connections in the neighborhood, its value is close to 0.

These two metrics are computed and tested between each pair of nodes across the mapped communities. These nodes are matched and identified as seeds if either their degree or their clustering coefficients are similar enough and above a certain eccentricity threshold.

For each pair of mapped communities, the community-blind mapping algorithm is performed locally — only considering the nodes inside these communities. This algorithm, takes two subgraphs (communities) $G_{c_1}\langle V_{c_1}, E_{c_1}\rangle$ and $G_{c_2}\langle V_{c_2}, E_{c_2}\rangle$ from two networks G_1 and G_2 and the set of seeds in these communities. This algorithm can also be run in parallel on each pair of mapped communities to increase performance.

4.3.3 Global propagation

The last step in our framework, applies the community-blind mapping algorithm on the whole network using all the currently mapped nodes as seeds. This step is necessary because all communities may not be correctly mapped or mapped at all; therefore, some nodes are not chosen to be re-identified. Running the community-blind mapping algorithm globally expands these mappings. In short, community-blind mapping algorithms run only this global propagation step, while our approach adds the previous steps as intermediate steps resulting in a community-aware mapping algorithm.

4.4 Degree of Anonymity

In practice, the mapping algorithm (be it community-aware or community-blind) may still leave several nodes unmapped. For these unmapped nodes, however, the *community structure reveals information about the true mapping*. We quantitatively analyze the degree of anonymity of users in a social network towards an attacker who uses either the community-blind or community-aware de-anonymization algorithm to re-identify users.

Consider two graphs $G(V, E)$ (reference) and $G'(V', E')$ (anonymized). $u \sim u'$ means that the mapping between u and u' is *true*. The set of true mappings is:

$$M_t = \{(u, u') : u \in V; u' \in V'; \text{and } u \sim u'\}.$$

We denote the *algorithmically detected mapping* between u and u' as $u \leftrightarrow u'$. The set of mappings

is:

$$M_a = \{(u, u') : u \in V; u' \in V'; \text{and } u \leftrightarrow u'\}.$$

We denote the community mapping between c and c' as $c \leftrightarrow c'$. The set of community mappings is:

$$M_c = \{(c, c') : c \in C; c' \in C'; \text{and } c \leftrightarrow c'\}$$

where C and C' are the sets of communities in G and G'. Given M_a, we quantify the anonymity for a user $u \in V$ as the entropy over the probability distribution of potential mappings being true for user u:

$$H(u) = \sum_{u' \in V'} P(u \sim u'|M_a) \log P(u \sim u'|M_a), \tag{4.2}$$

where $P(u \sim u')$ is the marginal probability that a node $u' \in V'$ is actually the true mapping with a given node $u \in V$. If $M_a = \varnothing$ or does not provide any information, $P(u \sim u'|M_a) = \frac{1}{N}$ for every u' and $H(u)$ reaches the maximum value of $\log N$. On the other hand, if we know that the algorithm works perfectly, namely $u \sim u' \rightarrow u \leftrightarrow u'$, then $P(u \sim u'|M_a) = 1$ when $u \leftrightarrow u'$.In this case $H(u)$ becomes zero.

Likewise, given M_a and M_c, we can quantify the anonymity for a user $u \in V$ as the entropy over the probability distribution of potential mappings being true for user u:

$$H(u) = \sum_{u' \in V'} P(u \sim u'|M_a, M_c) \log P(u \sim u'|M_a, M_c). \tag{4.3}$$

We define the normalized degree of anonymity for user u as:

$$A(u) := \frac{H(u)}{H_{max}}, \tag{4.4}$$

where $H_{max} = \log N$ is the maximum entropy.

Finally, we define the degree of anonymity for the whole system by averaging the degree of

anonymity of all users in the network:

$$A(G) := \frac{\sum_{u \in V} A(u)}{N}. \tag{4.5}$$

In computing the degree of anonymity for a particular anonymized network, one can estimate the accuracy and characteristics of the algorithm through simulation-based experiments. Next, we show how the degree of anonymity can be calculated with community-blind or community enhanced de-anonymization algorithms.

4.4.1 Degree of anonymity of community-blind de-anonymization algorithm

We define V'_- (resp. V_-) as the set of nodes in V' (resp. V) that have not been mapped by the algorithm:

$$V'_- = \{u' \in V' : \nexists u \in V, \quad u \leftrightarrow u'\}.$$

If a community-blind algorithm is employed for de-anonymizing users, $P[u \sim u'|M_a]$ for a given $u \in V$ can be assigned values for all $u' \in V'$ based on the following cases.

(1) If u is mapped by the algorithm to z', the vertices $u' \in V'$ can be partitioned as:

   a) The mapped node z', i.e., $(u, z') \in M_a$. In this case we need to compute $P[u \sim z'|u \leftrightarrow z']$. This value, i.e., how often a claimed mapping is correct, can be estimated through simulation. Let this value be p_{map}.

   b) The remaining nodes that were not mapped to u, i.e., y' such that $(u, y') \notin M_a$. We need to compute $P[u \sim y'|u \leftrightarrow z', z' \neq y']$. Following a mapping of the anonymized graph, This value can be estimated as $\frac{1-p_{map}}{|V'|-1}$, which assumes that any node in this set is the correct mapping with uniform probability.

(2) If u is not mapped by the algorithm, i.e., $(u, u') \notin M_a$, we consider the correct mapping to be within the entire vertex set V' with uniform probability. That is, $P[u \sim u'|u \in V_-] = 1/|V'|$.

We note that a more fine-grained case analysis can be applied for a better estimate of the degree of anonymity. For example, Cases (1).b and (2), can be further split into nodes that were mapped to

some other node or not mapped to any node at all. We have computed the degree of anonymity under this model and observed that there is only a slight difference between the two methods. We stick to the simpler model for ease of exposition (especially as it blows up the number of cases described in Section 4.4.2), and thus in Section 4.6 we show results only for the simpler model. Again, we have computed results for both models and observed only minor differences in the degree of anonymity.

4.4.2 Degree of anonymity of community-aware de-anonymization algorithm

The community mapping reveals additional information about the true mapping, and thus several more cases need to be considered as compared to the community-blind algorithm.

We define $C'\_(resp.C\_)$ as the set of communities in $G'(resp.G)$ that have not been mapped. $c \leftrightarrow c'$ represents that the algorithm has mapped community c in G to community c' in G'.

If a community-aware algorithm is employed for de-anonymizing users $P[u \sim u'|M_a, M_c]$ can be assigned values for all $u' \in V'$ based on the following cases (again this analysis presents a simpler case analysis as described earlier):

(1) If u is mapped to some node z' by the algorithm, and the community c of u is also mapped to the community c' of z', the vertices u' can be partitioned as:

   a) The mapped node z'. In this case we need to compute $P[u \sim z'|u \leftrightarrow z', c \leftrightarrow c', u \in c, z' \in c']$, i.e., how often a claimed mapping is correct (in this circumstance). This probability can be estimated through simulation. Let this value be $p_{map,1a}$, where the second subscript '1a' refers to Case 1a.

   b) The remaining nodes y' within c' that were not mapped to u. In this case, we need to compute $P[u \sim y'|u \leftrightarrow z', c \leftrightarrow c', y' \neq z', z' \in c', y' \in c']$. The probability that a node is mapped to *any* other node in the same community estimated through simulation as $p_{map,1b}$. Thus $P[u \sim y'|u \leftrightarrow z', c \leftrightarrow c', y' \neq z', z' \in c', y' \in c'] = \frac{p_{map,1b}}{|c'|-1}$.

   c) The remaining nodes r' that are not in c' (i.e., in $G' \setminus c'$). In this case, we need to compute $P[u \sim r'|u \leftrightarrow z', c \leftrightarrow c', r' \notin c', z' \in c']$. The probability that a node is

mapped to *any* other node not in community estimated through simulation as $p_{map,1c}$. Thus $P[u \sim r' | u \leftrightarrow z', c \leftrightarrow c', r' \notin c', z' \in c'] = \frac{p_{map,1c}}{|G \setminus c'|}$.

We note that $p_{map,1a} + p_{map,1b} + p_{map,1c} = 1$.

(2) If u is mapped to some node z' by the algorithm, and the community c of u is mapped to some community c' which is different from community of z', the vertices u' can be partitioned as:

a) The mapped node z'. In this case we need to compute $P[u \sim z' | u \leftrightarrow z', c \leftrightarrow c', u \in c, z' \notin c']$, i.e., how often a claimed mapping is correct (in this circumstance). This probability can be estimated through simulation. Let this value be $p_{map,2a}$.

b) The nodes y' within c'. In this case, we need to compute $P[u \sim y' | u \leftrightarrow z', c \leftrightarrow c', z' \notin c', y' \in c']$, the probability that a node is mapped to *any* node in the mapped community. This value can be estimated through simulation as $p_{map,2b}$. Thus $P[u \sim y' | u \leftrightarrow z', c \leftrightarrow c', z' \notin c', y' \in c'] = \frac{p_{map,2b}}{|c'|}$.

c) The remaining nodes r' that are not in c' (i.e., in $G' \setminus c'$) and are not mapped to u. In this case, we need to compute $P[u \sim r' | u \leftrightarrow z', c \leftrightarrow c', r' \neq z', r' \notin c', z' \notin c']$, the probability that a node is mapped to *any* other node not in community. This value can be estimated through simulation as $p_{map,2c}$. Thus $P[u \sim r' | u \leftrightarrow z', c \leftrightarrow c', r' \neq z', r' \notin c', z' \notin c'] = \frac{p_{map,2c}}{|G \setminus c'| - 1}$.

We note that $p_{map,2a} + p_{map,2b} + p_{map,2c} = 1$.

(3) If u is mapped to some node z' by the algorithm, but the community c of u is not mapped to any community in G', the vertices u' can be partitioned as:

a) The mapped node z'. In this case we need to compute $P[u \sim z' | u \leftrightarrow z', c \in C_-, u \in c]$, i.e., how often a claimed mapping is correct (in this circumstance). This probability can be estimated through simulation. Let this value be $p_{map,3a}$.

b) The remaining nodes that were not mapped to u, i.e., r' such that $(u, r') \notin M_a$. We need to compute $P[u \sim r' | u \leftrightarrow z', z' \neq r']$. Following a mapping of the anonymized graph, This value can be estimated as $\frac{1 - p_{map,3a}}{|V'| - 1}$, which assumes that any node in this set is the correct mapping with uniform probability.

(4) If u is not mapped to any node in G' by the algorithm, and the community c of u is also not mapped to any community in G', the vertices u' can be partitioned as:

   a) We consider the correct mapping to be within the entire vertex set V' with uniform probability. That is, $P[u \sim u' | u \in V_-] = 1/|V'|$.

(5) If u is not mapped to any node in G' by the algorithm, but the community c of u is mapped to a community c', the vertices u' can be partitioned as:

   a) The nodes y' within c'. In this case, we need to compute $P[u \sim y' | u \in V_-, c \leftrightarrow c', y' \in c']$, the probability that a node is mapped to *any* node in the mapped community. This value can be estimated through simulation as $p_{map,5a}$. Thus $P[u \sim y' | u \in V_-, c \leftrightarrow c', y' \in c'] = \frac{p_{map,5a}}{|c'|}$.

   b) The remaining nodes r' that are not in c' (i.e., in $G' \backslash c'$). In this case, we need to compute $P[u \sim r' | u \in V_-, c \leftrightarrow c', r' \notin c']$, the probability that a node is mapped to *any* other node not in community. This value can be estimated as $\frac{1-p_{map,5a}}{|G \backslash c'|}$, which assumes that any node in this set is the correct mapping with uniform probability.

4.5 Evaluation

We perform simulation-based experiments using real-world and synthetic network datasets. For each experiment, we prepare a copy of the original network, partially alter its structure, and compare the network alignment performance of two approaches — community-aware and community-blind — using the networks. We also perform experiments on partially overlapping networks where some percentage of users from each network — original network and the edge-altered noisy network — are removed and thus two networks have partially different user sets.

4.5.1 Data sets

Real-world graphs

We use three real-world networks: (i) A collaboration network [191], which is a network of coauthorships between scientists who have posted preprints on the arXiv Condensed Matter E-Print Archive. In this network two authors are connected if they wrote at least one paper together. The network is constructed from all preprints posted between January 1, 1995 and March 31, 2005. This network has 36,458 nodes and 171,735 edges; (ii) A Twitter mention network [237], which captures the connections between users who mutually mentioned each other at least once between March 24th, 2012 and April 25th, 2012. We first extract the largest connected component from this graph and partition it into four graphs using the METIS graph partitioning algorithm [34] to obtain a smaller, more manageable network. We use one of the graph partitions with 90,332 nodes and 377,588 edges. (iii) Using the same Twitter mention network, we also partition it into nine graphs using the METIS graph partitioning algorithm to obtain a much smaller network with 9,745 nodes and 50,164 edges. We show the impact of size on our approach using these two networks.

Synthetic benchmark graphs

We also use LFR-Benchmark generator [161], which produces networks with realistic degree and community-size distributions while controlling the strength of communities. We generate three synthetic networks with different size — S_1 with 10,000 nodes, S_2 with 20,000 nodes, and, S_3 with 40,000 nodes. Besides the difference in size, the two networks have identical parameters: the average degree is 20, the mixing parameter μ is 0.1, and the maximum degree is 100. S_1, S_2 and S_3 have 94,097, 186,946 and 375,513 edges respectively.

4.5.2 Experimental set up

Generating noisy anonymized networks

We replicate the original network and assume that it is anonymized. First, we assume that two networks have the same set of nodes while having different but overlapping sets of edges. We prepare an array of networks with different levels of noise to investigate the impact of noise on the performance of the algorithms. We use a common edge-rewiring method [253], which we describe in Algorithm 1. Briefly, the level of noise Θ is the portion of edges that are rewired. For instance, $\Theta = 0.10$ means that 10% of the edges are rewired.

Algorithm 1 Adding noise through edge rewiring

Input: $G_1\langle V_1, E_1\rangle$ **and mixing parameter** Θ
Output: $G_2\langle V_2, E_2\rangle$**: A noisy version of** G_1 **where** $V_1 = V_2$ **but** $E_1 \neq E_2$
copy G_1 to G_2
while $num\_rewired\_edges \leq \Theta \times |E_1|$ **do**
  randomly choose an edge $e_1 \in E_1$
  find $e_2 = (u, v) \in E_2$ which is $e_1 = (u, v)$'s corresponding edge in E_2
  remove $e_2 = (u, v)$ from E_2: $E_2 \leftarrow E_2 \setminus e_2$
  randomly choose a non-existent edge $e = (u, v)$ to be added: $E_2 \leftarrow E_2 \cup e$
end while
return ($G_2\langle V_2, E_2\rangle$)

We use the following level of noises:$\{0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4\}$. While we do examine scenarios with high noise, our results suggest that noise levels greater than 20% seem to set fundamental limitation on the possibility of de-anonymization through structure alone. The range of noise in our simulations is greater than the previously observed noise in previous simulation study [189, 190].

For each noise level, we generate an ensemble of 10 networks for each of the real-world and synthetic networks. We run the InfoMap community detection algorithm [210] on every graph to detect the community structure.

To assess how reliably similar community structures can be found in the presence of noise, we compute the Normalized Mutual Information (NMI) [163] between the reference graph G_r — the graph without noise — and each of the noisy graph ensembles (G_n). $\mathrm{NMI}(G_r, G_n) = 1.0$ means

			Normalized Mutual Information						
Networks	Number of nodes	Number of edges	0.1%	5%	10%	15%	20%	30%	40%
S_1	10,000	94,098	1	1	1	1	0.99	0.97	0.90
S_2	20,000	186,946	1	1	1	1	1	0.99	0.98
S_3	40,000	375,513	1	1	1	1	0.99	0.99	0.98
Collaboration	36,458	171,735	0.85	0.70	0.60	0.51	0.43	0.30	0.22
Twitter mention	90,332	377,588	0.75	0.50	0.34	0.23	0.16	0.08	0.04

Table 4.1: The mutual information between the reference graph and each of perturbed graphs

that the community detection algorithm can detect identical communities in two networks while $\mathrm{NMI}(G_r, G_n) = 0.0$ means that the community structure is completely different. As shown in Table 4.1, the community detection algorithm identifies almost identical community structures in the synthetic networks even with a high level of noise. However, adding noise to real social networks quickly changes their community structures, although The change rate, however, is not the same for all real networks. some networks have more robust community structures than others. For example, with 10% of noise, the NMI for the collaboration network is 0.60 while it is 0.34 for the Twitter mention network. With 40% of noise, InfoMap is unable to identify communities reliably in Twitter network.

Generating noisy anonymized overlapped networks

We also conduct experiments for when the two networks are not identical to each other and may have different sets of nodes and edges. After generating noisy networks using the 'edge-rewiring method', we remove some percentage of the nodes randomly from both the original and the noisy networks so that if the noisy network has been generated by re-wiring 10% of edges, we additionally remove 5% of the nodes from each of the original and the noisy networks. Thus, the resulting networks would have different and overlapping user and edge sets.

Setup for calculating degree of anonymity

The attacker needs an estimate of the performance of our de-anonymization technique to compute the degree of anonymity. We mimic this process by performing several experiments (10 runs) on each data set with a specific level of noise and number of seeds, and obtained the overall perfor-

mance of the de-anonymization algorithm on that particular data set and settings. Then, we performed 10 new de-anonymization experiments on each data set and use the success probabilities from the previous experiments to calculate the degree of anonymity for the data set. Finally, we average the degree of anonymity values for these 10 experiments. We emphasize that the simpler version of degree of anonymity calculation requires fewer prior knowledge about the algorithm's performance and it provides an upper bound for the degree of anonymity.

Eccentricity thresholds

For the node-mapping algorithms, we set the eccentricity threshold to 0.1 for all experiments. However, for community mapping, we set this threshold to 0 because we observed that having more mapped communities always gives more correctly mapped nodes. Consequently, this threshold also results in more false positives. But, the effect of false positives in community mapping is limited.

Initial seeds

We assume the attacker has some prior knowledge about a small number of nodes. To simplify the problem, instead of running a sophisticated seed detection algorithm, we provide the same set of initial seeds for both community-blind (original NS) and community-aware algorithms. We identify cliques of four in both networks (original and perturbed) and randomly choose some of them as seeds. To investigate the sensitivity to the initial seeds, we choose 4, 8, 16, 32, 64, 128 nodes as seeds (they correspond to 1, 2, 4, 8 and 16 cliques respectively). The NS propagation algorithm uses these seeds to perform global propagation while our algorithm employs them to map communities, then performs community-based seed enrichment before running the global propagation step.

4.5.3 Measuring Performance

Although degree of anonymity is the better metric for measuring performance of de-anonymization algorithms, we also measure the number of correctly mapped nodes and incorrectly mapped nodes

(as done by Narayanan and Shmatikov [190]), normalized by the total number of nodes in the networks, which we define below.

The *success rate* P_s is defined as the percentage of correctly re-identified users in the network:

Definition 3 Success rate of de-anonymization. *Given graphs $G\langle V, E\rangle$ and $G'\langle V', E'\rangle$, the set of detected mappings M, and the true mapping M_t, the success rate P_s is*

$$P_s = \frac{|M \cap M_t|}{|V \cap V'|}.$$

Similarly, the *error rate* P_e is defined as the percentage of incorrectly mapped users:

Definition 4 Error Rate of de-anonymization. *Given the same G, G', M, and M_t as in the definition of the success rate,*

$$P_e = \frac{|M \setminus M_t|}{|V \cup V'|}.$$

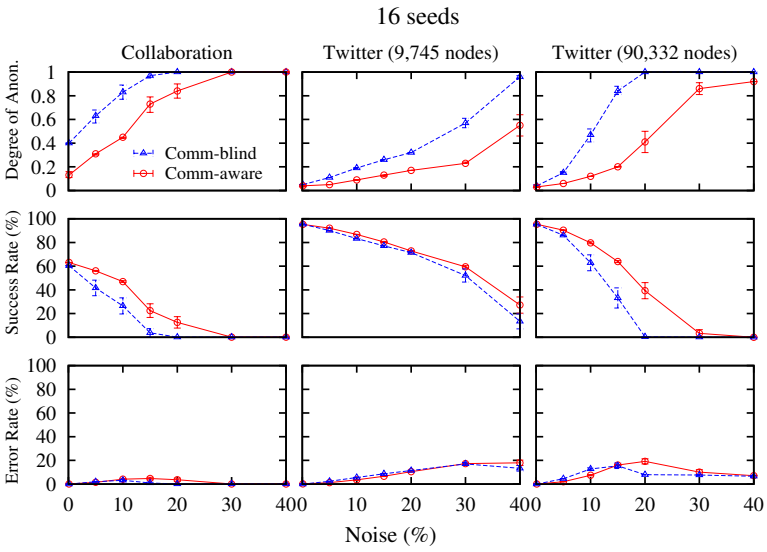
Using mapped communities, our algorithm identifies additional seeds through seed enrichment.We measure the effectiveness of this step by measuring 'true seed enrichment' and 'false seed enrichment.'

Definition 5 True seed enrichment. *Given G, G', M_t, M, and the number of seeds N_s, we consider the enriched seed mappings, M_s $(M_s \subseteq M)$ generated by community-based seed enrichment step (the mapping we have before running local propagation). The true seed enrichment is:*

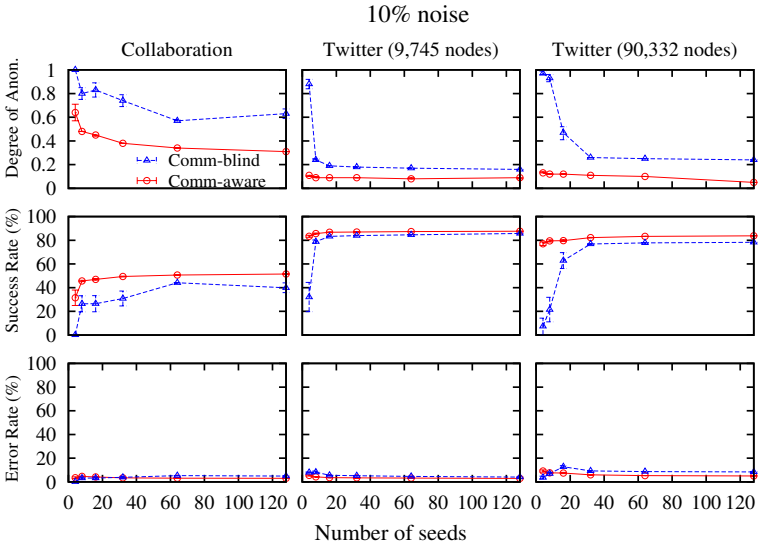
$$E_t = \frac{|M_s \cap M_t|}{N_s}$$

Definition 6 False seed enrichment. *Given G, G', M_t, M, N_s, and M_s, the false seed enrichment is:*

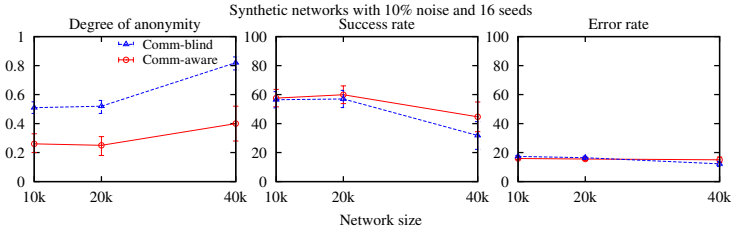
$$E_f = \frac{|M_s \setminus M_t|}{N_s}$$



(a) Number of seeds is set to 16



(b) Level of noise is set to 10%



(c) Synthetic network– noise and number of seeds are set to 10% and 16

Figure 4.3: Performance of community-aware and community-blind algorithms on Collaboration, and Twitter mention networks.

4.6 Results

We now report the performance of two algorithms (community-blind NS global propagation vs. our community-aware algorithm). Our results demonstrate that our community-based method can boost the performance of de-anonymization, particularly when (1) there are fewer number of initial seeds, (2) the system size is large, and (3) the noise level is high.

4.6.1 Impact of noise, seed size, and network size on overall performance

Impact of noise

Figure 4.3 shows the degree of anonymity $A(G)$ (top row), success rate P_s (middle row), and error rate P_e (bottom row) in terms of noise, number of seeds, and network size.[3] We can see that the community-aware algorithm is much more effective in decreasing the anonymity of users in all the networks. For example, in the collaboration network, for all the levels of noise from 0 to 20% (Figure 4.3(a), left column), the decrease in degree of anonymity when using the community-aware algorithm is about twice as much as that when using community-blind algorithm. Specifically, for 10% noise and 16 seeds, $A(G)$ is 0.45 and 0.83 (or anonymity is 6.81 and 12.57 bits) when using community-aware and community-blind algorithms, respectively. Note that for a collaboration network with 36,458 nodes, the maximum anonymity is equal to 15.15 bits. As noise increases, both algorithms are less successful in re-identifying users. However, the community-aware algorithm tolerates more noise than the community-blind algorithm. For example, in the collaboration network, with 20% noise, the community-aware algorithm is able to correctly map about 15% of users, while the community-blind algorithm can barely re-identify any user. The degree of anonymity is 0.84 and 1 (or anonymity is 12.72 and 15.15 bits) when using community-aware and community-blind algorithms, respectively. For 30% and 40% of noise, both algorithms perform poorly and the degree of anonymity is 1.

[3]The percentage of unmapped nodes is simply the difference between 100% and the sum of the success and error rates.

The same observations can be seen in the Twitter network with 90,332 nodes. In this network, the decrease is not uniform over different levels of noise (Figure 4.3(a), right column). Both algorithms are highly successful in re-identification of users when the noise is less than 5%. However, the difference between the performance of two algorithms greatly increases when the noise is above 15% and 20%. Specifically, for 15% of noise and 16 seeds, $A(G)$ is 0.2 and 0.84 (or the anonymity is 3.29 and 13.82 bits) when using community-aware and community-blind algorithms, respectively. In other words, the community-aware algorithm *reduces the anonymity by 10 additional bits* compared to the community-blind algorithm. In this case the success rate of the community-aware algorithm (65%) is almost twice as much as that of the community-blind algorithm (33%). Note that the maximum anonymity for this network is 16.46.

The results on the Twitter network with 90,332 nodes also shows the community-aware algorithm is more robust to noise, such that even with 20% of noise, it is able to correctly map about 40% of users, while the community-blind algorithm can barely re-identify any user. It also reduces the degree of anonymity about 60% (from 1 to 0.41). In other words, the anonymity is reduced by about 10 bits from 16.46 to 6.75. For 30% and 40% of noise, the community-aware algorithm also re-identifies about 3.5% and 0.04% of the users and reduces the degree of anonymity to 0.86 and 0.92, respectively.

Impact of number of seeds

Figure 4.3(b) shows the impact of the number of seeds on $A(G)$, P_s and P_e. The noise level is set to 10%, while the number of seeds changes from 4 to 128. Both algorithms are more successful when more seeds are provided to them. However, our community-aware approach is more robust to a smaller number of initial seeds. You can see that in all networks, over different numbers of initial seeds, the community-aware algorithm always reduces the anonymity (and re-identifies users) more than the community-blind algorithm. For example, in the collaboration network (Figure 4.3(b), left column), when the number of seeds is 32, $A(G)$ is 0.38 and 0.74 (or the anonymity is 5.76 and

11.21 bits) when using community-aware and community-blind algorithms, respectively.

In the Twitter network with 90,332 nodes (Figure 4.3(b), right column), a smaller number of seeds significantly affects the performance of the community-blind algorithm. However, the number of seeds only slightly affects the performance of the community-aware algorithm. For example, when the number of seeds is only four, the community-aware algorithm successfully re-identifies 77% of users while the community-blind algorithm only re-identifies about 7% of the users. Similarly, the degree of anonymity is about 0.13 and 0.97 (and anonymity is 2.14 and 15.97) when using community-aware and community-blind algorithms respectively. In other words, the community-aware algorithm *decreases the anonymity by 13.83 additional bits* compared to the community-blind algorithm.

Impact of network size

Comparing the results for Twitter network with 90,332 nodes and the Twitter network with 9,745 nodes (Figure 4.3(b), right and middle columns, respectively) illustrates the impact of size on the performance of both algorithms. Having a smaller network, both algorithms perform better in re-identifying users and tolerating the noise. For example, both algorithms are successful in re-identifying users even with 40% of noise. However, the performance difference between the community-aware and community-blind algorithms is more obvious when the network is bigger. For example, with 20% noise, $A(G)$ in the larger Twitter network is 0.41 and 1 when using the community-aware and community-blind algorithms. Thus, the diffenece is about 0.60. However, having 20% noise, $A(G)$ in the smaller Twitter network is 0.17 and 0.32 when using community-aware and community-blind algorithms, respectively and the difference is only about 0.15. Thus, our community-aware approach is more robust to the size of network.

To investigate the impact of network size in a more controlled environment, as explained in Section 6.2.5, we generated three synthetic networks that have the same network parameters but different network sizes: $\{$10,000, 20,000, 40,000$\}$. Figure 4.3(c) illustrates the performance of both

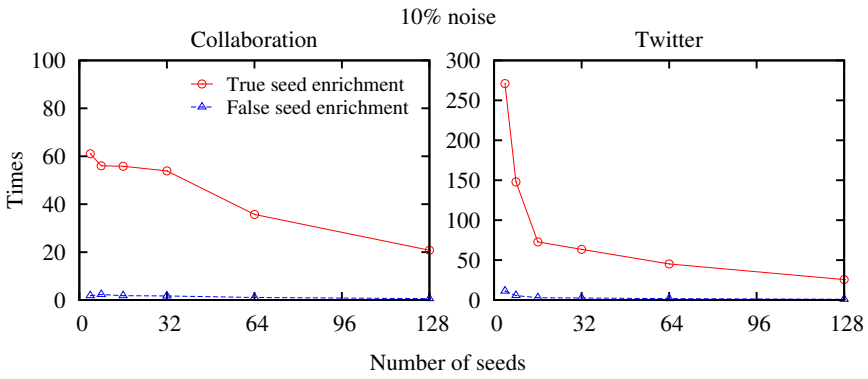


Figure 4.4: Seed enrichment performance: by running very simple seed identification algorithm inside communities, more seeds are identified

de-anonymization algorithms when the noise and the number of seeds are set to 10% and 16 and the network size varies. As can be seen, the success rates are almost the same when the network size is 10,000 and 20,000, but the community-aware algorithm performs about 17% better when the network size increases to 40,000. The degree of anonymity illustrates that the community-aware algorithm is much more effective in decreasing the anonymity of users even when the network size is as small as 10,000 (the difference between comm-aware and comm-blind is 0.25). As expected, for the network with 40,000 nodes, the difference between the degree of anonymity is even more when using the community-aware (about 0.4) and community-blind algorithms (0.82) .

The error rates of both algorithms show similar trends. Our approach exhibits slightly higher error rate in some cases but most of them occur when the community-blind approach completely fails and ours correctly identifies many more users.

In summary, if one aims to map two networks that are not identical to each other, using our community-based mapping algorithm is almost always guaranteed to reduce the anonymity more and find more successful mappings than the community-blind, global mapping algorithm. In addition, we expect a larger boost as the prior information (seeds) decreases.

4.6.2 Seed enrichment boosts the number of seeds and makes the propagation algorithm to be started with more information

After the community mapping phase, some communities already have some seeds inside them, (these are the initial seeds), however, no seed exists in many mapped communities. Here, we show that even applying a simple seed identification algorithm is effective inside mapped communities and increases the number of total seeds which eventually increases the number of mapped nodes in the network. Enriched seeds are not prior knowledge, and, the seed identification algorithm may have mapped them incorrectly, thus, they may also give wrong information to the propagation algorithm. As we explained in Section 4.3, applying more complicated approaches can improve the final results more.

Figure 4.4 shows both the rate of increase in the number of "correct" seeds and the increase rate in the number of "incorrect" seeds. For example, in the collaboration network with 16 initial seeds, and 0.1% of noise, after the seed enrichment phase, the number of correct seeds are about 56 times more (total of 890 seeds). Even having more initial seeds, the algorithm identifies more correct seeds. For example, in collaboration network, the number of correct seeds increases almost 21 times as much as 128 initial seeds. In other words, the propagation step gets about 2,649 seeds. As you can see, the increase rate for wrong seeds is comparable with the increase rate for correct seeds and based on the final error rates presented in Figure 4.3, their effects are limited. Similar results have been obtained for other networks.

4.6.3 Results for overlapping data sets

Figure 4.5 demonstrates that the community-aware algorithm also outperforms the community-blind algorithm when the original and the noisy networks do not have the exact same user base. We added noise to the user sets by randomly removing nodes (and their edges) from both the original and the noisy networks. Thus, in Figure 4.5, the noise 10% means that we first added 10% noise to the edges of original network and generated the noisy network as previously described. Then, we removed

10% of nodes (and incident edges) from both the original and the noisy networks. Since $V \neq V'$, the success rate is normalized by $|V \cap V'|$.

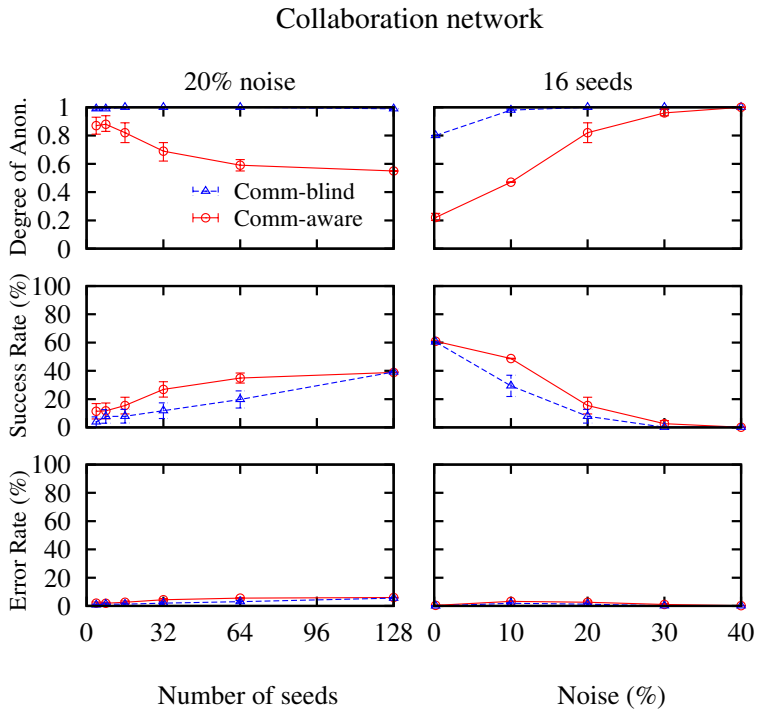


Figure 4.5: Performance on overlapped data sets

For calculating the probabilities needed for computing the degree of anonymity, we assumed that the attacker does not have information about the distribution of overlapping nodes in both networks and we obtained the degree of anonymity using the formulation provided in Section 4.4. The more precise calculation can be obtained by considering more cases where the correct mapping of a user exists/does not exist in the other network.

As we can see in Figure 4.5 (left column), the degree of anonymity decreases with more seeds. With 20% noise, the degree is reduced from 0.87 to 0.55 (and anonymity is reduced by 2 and 6.7 bits) with 4 and 128 seeds respectively.

Looking at the right column of Figure 4.5 (16 seeds with varying noise), we see that the community-blind algorithm fails completely when the noise level is more than 10% whereas the community-aware algorithm fails when the noise level is more than 30%. The community-aware

algorithm re-identifies 20% more users than the community-blind algorithm when the noise is 10% and it identifies about twice as much when the noise is 20% and 30%. Specifically, when the noise is 10%, and the number of seeds is 16, the success rate is about 49% and 29% for community-aware and community-blind algorithms respectively. In addition, the community-aware approach reduces $A(G)$ almost twice as much as the community-blind algorithm from 0.98 to 0.47. In other words, the anonymity is reduced by more than 7 bits from 14.77 to 7.08 bits. Note that the maximum entropy for this network is 15.08.

4.6.4 Time complexity

The community detection step is not a bottleneck as Infomap's time complexity is O(m). For instance, the Louvain method, another popular community detection method has been applied to large graphs with more than 20M nodes and 100M edges. As community mapping and local propagation use the same de-anonymization algorithm (on a much smaller graph), the time complexity of these steps is smaller than the final global propagation step. Therefore, the time-limiting step in the whole process is the final, global propagation step, which is determined by the algorithm being boosted. If the original algorithm can be applied to a given graph, our approach can also be applied to the graph without increasing the overall time complexity.

4.7 Summary

We showed how 'mesoscopic' properties of a social network can be leveraged to improve the degree of de-anonymization of anonymized social network datasets. In particular, decomposing the network into 'communities' allows for de-anonymization at a coarser granularity first, and then at the node level. This approach is more robust against added noise to the anonymized data set, and can perform well with fewer known seeds as well as larger networks.

This work focused on demonstrating the utility of community detection to de-anonymization, and thus focused on the structural properties of the network. Other methods can be applied for

mapping the communities in two networks, e.g., leveraging other attributes such as location, language, time, messages/posts, and so on. Future work can explore ways to combine such techniques in novel ways. For example, these attributes can be studied at the community level before drilling down into individual communities. Finally, we believe our approach is general enough to 'plug in' stock community-blind mapping algorithms. Although we have tested only the NS algorithm any other community-blind approach could be used based on our framework.

Chapter 5

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

In our first attempt for building privacy preserving P2P networks for socially networked applications, we developed a distributed architecture for a social based question and answering system, social search.

Social search systems such as Aardvark (`http://www.vark.com/`) [130] and Facebook Questions 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.[1] Horowitz and Kamvar [130] draw the distinction between the *library* model of search (e.g., web searches), where askers search for the *right document* to answer a question vs. the *village* model (e.g., Aardvark), where askers seek to get connected with *right human* because it is unlikely their complicated question can be satisfied by an existing document.

Centralized systems such as Aardvark and Facebook, unfortunately, 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 information immediately become available to the experts when the user posts a query or answers a question. Note that the asker and the answerer do not also have any control on the information shown to the others. Even if the social search service does not reveal identities of 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

[1] Aardvark claims "The vast majority of questions are answered within 10 minutes."

breaches (a central point of failure). Connecting to such services over anonymizing networks such as Tor [99] to hide the identities of askers and answerers break the utility of the system because the structure of the social network is used to match askers and answers based on their proximity in the social network. Furthermore, centralized services can manipulate or censor queries and answers and restrict communication between users in the system, anonymous or not. Thus, existing routing-layer anonymity solutions do not suffice.

An alternative is to develop **decentralized solutions**. Furthermore, we need to rethink the design of current anonymity systems to expose the structure of the social network just enough to make relevant matches between askers and answers. Within the context of such systems, we need to understand what the new privacy requirements are. Traditional notions of *sender or receiver anonymity*, for example, which aim to keep the identify of the sender or receiver secret do not capture properties such as *expertise unlinkability* for answerers or *interest unlinkability* for askers. For example, even if individual messages cannot be tied to Alice, one may be able to determine Alice is a pro-choice expert. Expertise areas may be leaked based on how such systems are structured (are similar experts connected to each other?), or how expertise is advertised in the system in order to attract queries. Thus providing expertise/topic unlinkability goes beyond anonymity of messages.

Towards building such systems we characterize the system model and security goals for decentralized live social search systems and propose Pythia. The central idea in Pythia is to partition the social network into *communities* or *flood zones* and 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 5.2), yet limits the amount of flooding to maintain scalability (see our analysis in Section 5.2.2). 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 network, and 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.

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.

This chapter is based on collaborative work with Naveed Alam, Nathaniel Husted, and Apu Kapadia and was published at Workshop on Privacy in the Electronic Society (WPES) [192].

5.1 System Model and Security Goals

We describe the high-level system model we assume for P2P social search systems, our security goals and the adversary model.

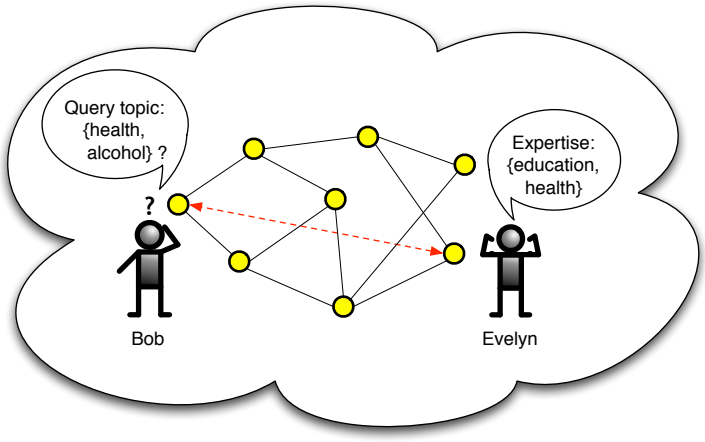


Figure 5.1: High-level system model. The nodes and links represent part of the larger social network. Bob issues a query related to alcohol abuse as indicated with the query tags. The query is routed (routing mechanism depends on specific architecture) to expert Evelyn who responds to Bob.

5.1.1 P2P Social Network

Several systems [90, 135, 214] 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* [202]. 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 this work, and discuss the system model and architecture assuming such an underlying P2P social network.

5.1.2 System model

We assume that a user is connected to and has knowledge of his/her list of friends in the social network (e.g., by bootstrapping off established social networks such as Facebook or instant messaging systems). Every user has a list of self-declared *expertise areas*, i.e., topics for which he/she can answer questions. For example, Alice's set of expertise areas could be {recipes, military intelligence analysis, 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 user prefers to 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.

Nodes (users) may be *online* or *offline* at any given time. Their online and idle status is visible to their friends, as with instant messaging applications. Offline nodes cannot assist in routing messages. 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 these query tags, the *query routing* protocol attempts to find experts for the specified tags close to the asker in the social network. Once the query is routed to an available expert, the expert can respond. We assume that not all available experts will answer questions, and whether they do will depend on how *responsive* the expert is.

5.1.3 Privacy and Security goals

The following three properties are unique to P2P live social search systems:

Expertise unlinkability: A peer's private expertise areas should not be attributable to her identity beyond a certain threshold probability. For example, Bob may be a pro-choice advocate who doesn't want to advertise his association widely with the pro-choice movement. We assume a threshold that is sufficient to provide *plausible deniability*. For example, some authors consider a probability of 0.5 to be sufficient [206]. We assume more conservative probabilities on the order of 0.1 to be sufficient for plausible deniability. However, in cases where the prior probability is higher than 0.1, we say plausible deniability is attained if the threshold is below the prior probability. For example, if 33% of nodes in a community are experts on a particular topic, then if the attacker cannot infer Alice is an expert in that topic with probability more than 0.33, we say plausible deniability is attained.

Interest unlinkability: An asker's private query tags should not be attributable to her identity beyond a certain 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. We assume the same probability threshold as discussed for expertise unlinkability.

Unobservable querying and responding: Anonymous (but observable) queries and 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 nodes related to a particular answer or question. We seek to prevent this attack and hide whether nodes are asking questions or providing answers at all. Note that although nodes in the system may observe queries and answers being exchanged, they do not know whether individual nodes are issuing queries or answers.

Sender anonymity: One important note is that we assume that the primary goal of an attacker is to uncover the expertise or interest area of a user. For such a requirement sender anonymity is necessary but not sufficient. Clearly if sender anonymity is broken for a message, the expertise area of the sender is also broken. Sender anonymity, however, does not imply expertise unlinkability. As already mentioned, expertise areas may be leaked based on how such systems are structured (are similar experts connected to each other in clusters?), or how expertise is advertised in the system in order to attract queries. Thus, providing expertise or topic unlinkability goes beyond message anonymity, and the system must be carefully designed to preserve querier/expertise unlinkability in addition to sender anonymity. As a result our proposed system Pythia uses underlying sender anonymity techniques as just one building block to provide the requisite privacy.

5.1.4 Attack model

We evaluate two classes of adversaries: *global attackers* can view all messages exchanged in the system and infer the online/offline and idle status of all the nodes at any time. *Colluding attackers* have only partial knowledge of this type—we assume some fraction of nodes c ($0 < c < 1$) are compromised and these colluding attackers can infer the online and idle status of their neighbors only, and view messages exchanged with their neighbors.

Attackers can have two different capabilities related to what they can infer about messages: the messages are *linkable* if adversaries can tell the answers (or questions) are authored by the same answerer (or asker). For example, perhaps the writing style is unique enough to link answers by

the same expert. Otherwise, the messages are *unlinkable*. Using and linking observed information, attackers try to determine the asker of a particular topic or the answerers of a particular topic. We assume all adversaries have full knowledge of the structure of the social network, and are *honest but curious*, i.e., they participate in the protocol correctly, but try to infer what they can based on their observations.

Thus we have four adversaries: *Global-Linkable*, *Global-Unlinkable*, *Colluding-Linkable*, and *Colluding-Unlinkable*.

5.2 Architecture

Figure 5.2 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 forwarding, asking or answering questions. If no answerers are found in the local community, questions are forwarded to a remote community as part of a *remote flood*.

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 plausible deniability as explained in Section 5.1.3. 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. 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 query flooding 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 Pythia, time is divided into intervals $\{t_1, ..., t_n\}$, where questions and answers are exchanged and distributed at the end of each time interval as coordinated by a *representative* who is reelected periodically. 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 note the representative can be adversarial (and pose as one of the four types of attackers outlined in Section 5.1.4). In many P2P networks (e.g., KaZaA, Gnutella and Skype), some leaders (or coordinators or super nodes) play a specialized role in the application that requires frequent communication with the other members of the set. These systems use super nodes to strike a balance between the efficiency of centralized search, and the autonomy and load balancing that provides better performance and scalability. These systems have been subject of recent studies [121, 126, 174]. Also, we assume that if these representatives go offline, a new representative is elected. The role of the representative will be made clear below.

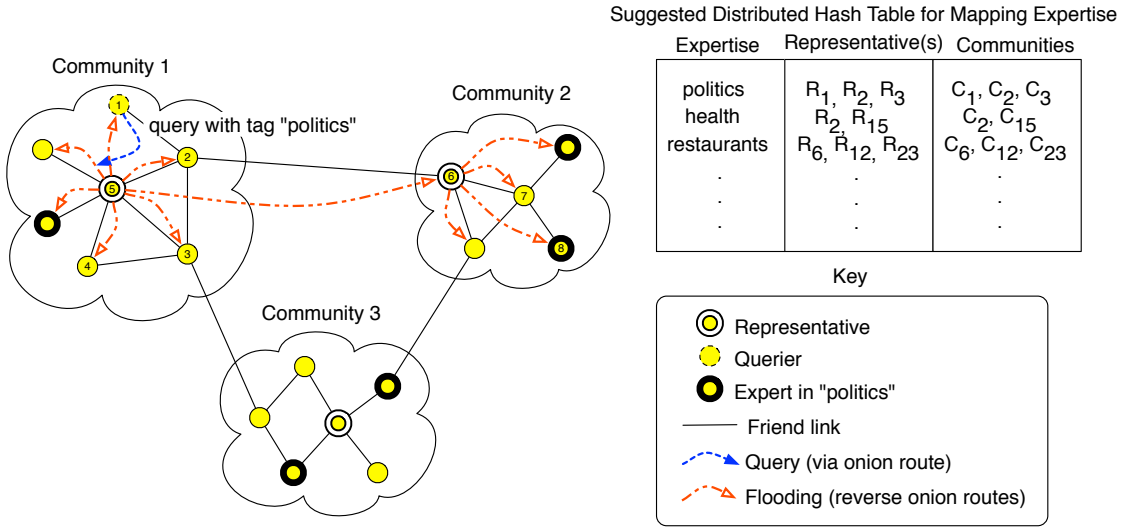


Figure 5.2: Node 1 in Community 1 initiates a query with the tag *politics*. The query via an onion route is sent to the *representative* node 5. The representative then floods the query to the local community (along with other received queries). The answerer in the community is unresponsive. The representative, having received no responses, may choose a random community or ask 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 whose response is sent to representative node 6 via an onion route (like a query message), then relayed to representative 5, and finally received by node 1 in the next flood by node 5.

5.2.1 Creating social communities

In Pythia, all nodes in the social network are grouped into self-organizing clusters called *communities*. We assume a distributed community-forming algorithm such as the one proposed by Ramaswamy et al. [204], which results in a *representative* for that community. In this algorithm, each community is initiated by a node and established based on nodes' connectivity requiring only local knowledge about neighboring nodes. In this scheme, nodes are clustered based on social relationships and each node belongs to a single community and all community members are known to each other. 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 5.2.2. Additionally, from the perspective of other communities the representative of a community is a conduit for communicating with nodes inside the community. Communities may have more than one representative, although, for simplicity, we assume that communities have only one representative.

5.2.2 Routing questions and answerers

To simulate a distributed social search, nodes in Pythia participate in three phases: asking, showing their intent to answer, and answering. In all phases the same protocol is used with different message types. All the messages in a community are forwarded to the representative of the community, and are padded to have the same size to thwart traffic analysis of messages in transit.

As a building block, Pythia uses Onion routing [114] 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 from the community and progressively build a circuit through these nodes. 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 community. The particular choice for this parameter is orthogonal to this work, and it suffices to pick a value that provides adequate mixing. To create an "onion packet", the message is encrypted with the public key of the representative and each node in

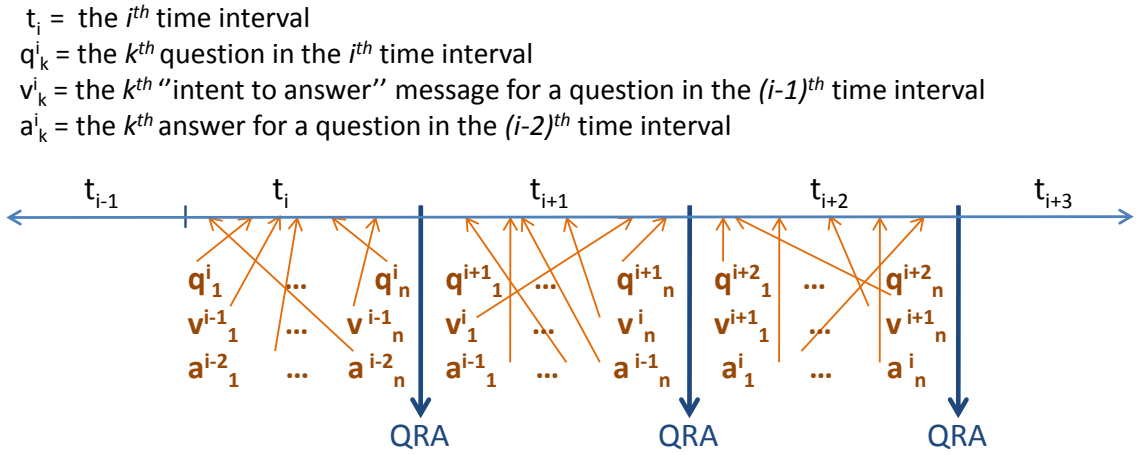


Figure 5.3: In each time period t_i, every online node sends three real or fake messages to the representative. The "intent to answer" messages are in response to the questions which were asked in the previous time period t_{i-1} and the answers are in response to the questions which were asked two time 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 circuit. 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 unless all nodes along the path are compromised. As with mix networks [91,183] we assume nodes along the route add delays and reorder messages to thwart timing attacks. Since messages are exchanged periodically, tens of seconds of delay at each hop should be sufficient. Note that to initiate a two-way conversation, an asker generates a "reply onion" [56, 78] to send the question. The reply onion is transmitted to the representative who is the last node of the chain. Then representative uses the reply onion to initiate the return chain for sending answers.

To obtain the public keys of nodes for setting up onion routes, Pythia can benefit from existing centralized or distributed key management approaches applied in peer-to-peer systems [97, 135, 153,241]. However, to avoid revealing any information about the circuit, the sender should be given a list of public keys of all the nodes within the community instead of the nodes in the circuit. An approach could be using a DHT to store several replicas of the communities' lists of public keys, and to obtain and use keys based on agreement. The nodes on the DHT responsible for storing the lists

would verify if the authorized IP address is updating or retrieving the list. To defend against global passive adversaries an extra step is needed to provide asker/answerer unobservability. As illustrated in Figure 5.3, 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. Due to the time delays imposed by nodes along the onion route, messages are received by the representative at random times in the interval.[2]

For 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 t_{i-1}. If questions with topic tags for which there exists no expertise in the local community are found, then they are forwarded to a remote community. In Pythia the remote community is chosen randomly. However, this selection can be done more efficiently and purposefully if users can advertise their topics of interests anonymously and a DHT mechanism provides the correlations between communities and topics so that questions can be directed to appropriate communities with potential answerers in that area.

Although many experts may be volunteered to answer a particular question so as not to overburden experts who are willing to answer only a couple of times a day, 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}. Answerers that have been picked to answer, find their one-time pseudonym in the respond block.

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. For all these messages, if the online node does not have a legitimate question, intent to answer, or answer, then the node sends a dummy message through the onion routing. Thus, attackers cannot directly observe who is asking or answering questions. Moreover, encrypted messages have fixed sizes so the adversary cannot infer the content

[2]The route length and average delays at each hop should be tuned so that messages are likely to be received by the representative within the time period.

from the message size. We note that the representative selects the α answers after discarding the dummy messages.

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 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. If the user is busy or idle, a dummy message is sent with the "volunteer to answer" bit set to 0 instead.

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 false, the answer string contains the real answer for the identified question.

Actions taken by representative

Although representatives are just like any other node with respect to asking and answering questions, they are responsible for distributing their communities' questions and answers. 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 shall now describe the flooding mechanism as well as the QRA message.

QRA message The QRA message is flooded by the representative at the end of each time period through reverse onion routes. As mentioned earlier, when a sender generates an onion to send any of the three messages to the representative, a reply onion is included with the message which can be used by the representative to initiate the return chain and send the QRA message. The QRA message contains three blocks as we shall now describe, along with the current time period.

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, then they are forwarded to a remote community. In Pythia the remote community is chosen randomly. However, this selection can be done more efficiently and purposefully if users can advertise their topics of interests anonymously and a DHT mechanism provides the correlations between communities and topics so that questions can be directed to appropriate communities with potential answerers in that area. We discuss advertising in Section 5.2.4.

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. We assume the representative is adversarial and thus all adversaries know the order in which messages were received by the representative. Pythia relies on onion routing to provide anonymity to the sender of a message.

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 expect such delays would be acceptable in practice. Our protocol, however, can be easily modified to send a remote query in parallel, in which case two communities are posed the question simultaneously.

5.2.3 Messaging overheads

Our controlled flooding approach does not induce high 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. Based on the format of messages and Pythia's protocol, even with a large community size of 10,000, every time period only 17MB of traffic is received by the representative and it sends out only 7MB to the community. 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,[3] 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. With smaller community sizes of hundreds to a thousand nodes, the traffic is on the order of a few hundred kilobytes to a megabyte *total* per time period. Thus the overhead of dummy messages is low and the representative is not overwhelmed with traffic.

[3]The median active users in Aardvark [130], issued 3.1 queries per month.

5.2.4 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.

Furthermore, we assume that nodes can advertise their expertise along with reputation information for that expertise. For example, users can accumulate anonymous digital cash [46] 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.

5.3 Attacks and Defenses

Our architecture ensures that questions and answers are exchanged within and between communities anonymously, yet allow for experts in the social vicinity of the asker to field queries. We designed the system to provide such functionality while maintaining expertise unlinkability through the use of such communities. Pythia uses two-way onion routing with mixing and dummy traffic to provide unobservability of question asking and answering, sender unobservability and resistance against traffic analysis attacks, especially for the Global adversaries who can observe all traffic in the system. Moreover, the use of two-way onion routing with mixing ensures that even a malicious representative cannot actively modify questions and answers to great benefit because the representative does not know who the recipients of the messages are. While this architecture also provides resistance against attacks to interest/expertise unlinkability, attackers nevertheless can attempt to correlate information about who is online and not idle and when to deduce the expertise of a victim. We briefly describe such attacks, and defenses, and evaluate our system under these attacks and defenses in the next section.

In all the attacks, attackers pick a 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. We note that representatives are the only nodes who know which messages are *local* (initiated by nodes in their community) or *remote* (coming from other communities), but an adversarial representative can share this information with other adversaries. Attackers with linking capability can narrow down the set of experts using an *intersection attack* by intersecting the sets of those present while questions on a topic were answered. In the unlinkable case, adversaries maintain counts of users who are present in the same time period when answers are received for sensitive topics. These counts are used to (try to) implicate users who are present more often when such answers are received. We call this the *counter attack*. More details on this attack are presented in Section 5.4.2. As a defense, if sending questions and answers are delayed probabilistically, then attackers have to observe online and offline nodes over a longer time interval

spanning multiple time periods. Thus attackers need to consider nodes that were not idle at any time period within the *time aggregation* interval.

5.4 Evaluation

Pythia uses two-way onion routing with mixing and dummy traffic to provide unobservability of question asking and answering, sender unobservability and resistance against traffic analysis attacks, especially for the global adversaries who can observe all traffic in the system. Moreover, the use of two-way onion routing with mixing ensures that even a malicious representative cannot actively modify questions and answers to great benefit because the representative does not know who the recipients of the messages are. While this architecture also provides resistance against attacks to interest/expertise unlinkability, attackers can attempt to correlate information about who is online and not idle to deduce the expertise of a victim. We now evaluate our system under such attacks.

To demonstrate our hypothesis that privacy and utility can indeed be balanced for privacy-aware social search, we simulated a P2P system of 60,000 nodes partitioned into social communities of around size 100. To test Pythia with various potential topologies of a social network, we created 5 randomly generated scale-free graphs with 60,000 nodes using the Network Work Bench[4] (NWB) tool and the Barabasi-Albert (BA) model. Each graph was used for 5 different simulation experiments. Communities were created using the process discussed in Section 5.2.1. A subset of communities was taken due to the computational complexity of the simulation. The subset communities were sized between 85 nodes and 115 nodes ($\mu = 95.526, \sigma = 8.6702$).

5.4.1 Usage models

Unlike searching for files in P2P networks, social search involves more social interactions [246]. 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 [130,

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

200, 209].

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 [200, 209], 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 [130]. Table 5.1 shows the topic distribution used in our simulations.

Table 5.1: 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

Users were randomly chosen for each percentage group (based on the probabilities in the first column of Table 5.1). Depending on the percentage group, the number of topics 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 [130] 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 [130] 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 [130], 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

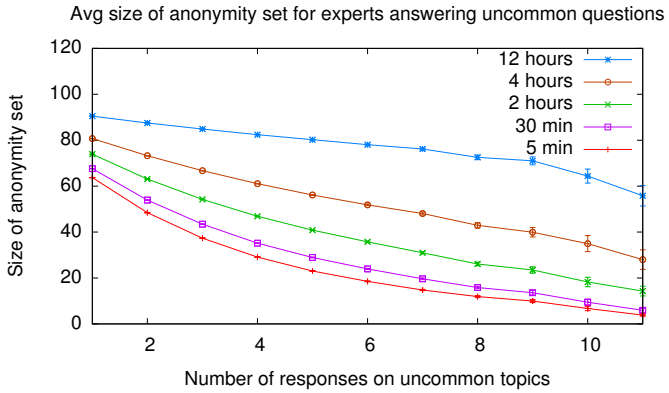


Figure 5.4: Anonymity set vs. number of questions answered for Uncommon topics against a Global-Linkable attacker after 4 weeks for various time aggregations.

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.

5.4.2 Security Evaluation

Global-Linkable adversaries

As depicted in Figure 5.5, we studied the degradation of anonymity after four weeks of system operation under the *Global-Linkable* model defined in Section 5.1.4. Anonymity sets for experts were measured depending on how many questions they answered in the four-week period. Adversaries were able to observe the online/offline/idle status of all users in the network and correlate questions and answers with those observations. As a private expert answers more and more questions, adversaries are able to narrow down the set of potential users that may be that expert.

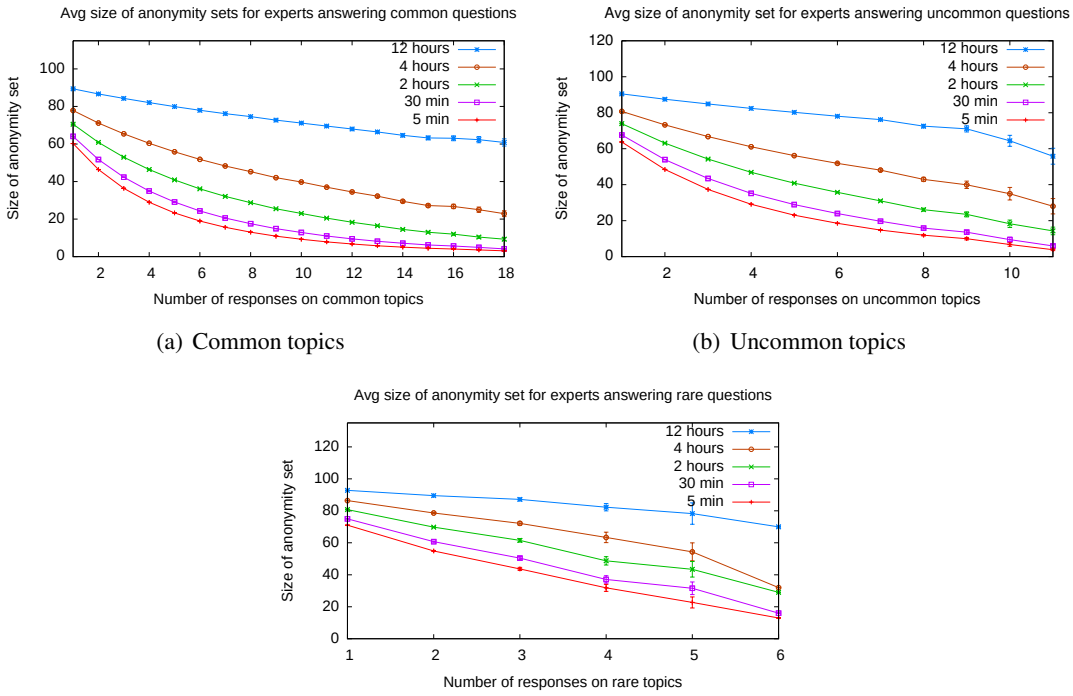
It can be seen in Figure 5.5 that the number of questions that an expert in common topics has

answered is much more than the number of questions that he/she has answered in uncommon and rare topics. This difference is due to the distribution of questions (Section 5.4.1).

Figure 5.5(c) shows that people answering rare questions have greater anonymity than answerers of uncommon and common questions over similar time spans. The anonymity as it relates to the number of questions is similar. For example, after answering 4 questions on a particular rare topic, the average size for the anonymity set is about 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 the anonymity set is about 20. After answering 18 questions after 4 weeks for a common topic, the anonymity set is still on average more than 2, which is better than the approximately 70% prior probability of the node being an expert in that topic (as 70% of nodes in a community are expert in common topics). For uncommon topics, after answering 11 questions over 4 weeks the anonymity set contains 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. Thus, after 4 weeks of asking and answering questions, a peer's private expertise or interest areas are not attributable to her identity beyond the threshold probability of 50%, 25% or 6% for common, uncommon, and rare areas.

The anonymity set for answerers in uncommon and rare topics degrades faster than for answerers in common topics. This is because common questions are asked and answered more frequently. As a result, for subsequent answers the intersection of online nodes does not change much. Since rare and uncommon questions are asked more infrequently, there is a longer time gap between answered questions on the average. Thus the sets of users change more between answers, and the intersection of online users for subsequent answers results in fewer nodes.

Figure 5.5 illustrates that if the time-aggregation defense is used in sending questions and answers, then the average size of the anonymity set increases for a particular number of questions. For example, in the case of answering 4 questions on a particular common topic, the average size of



(a) Common topics

(b) Uncommon topics

(c) Rare topics

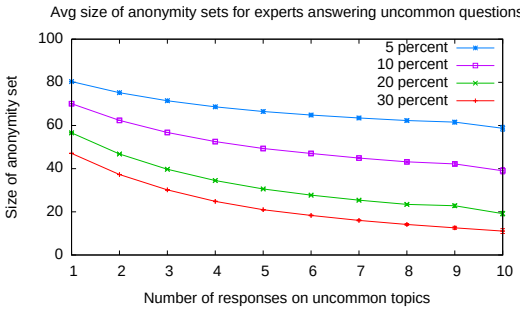
Figure 5.5: Anonymity set vs. number of questions answered for Common, Uncommon, and Rare topics against a Global-Linkable attacker after 4 weeks for various time aggregations.

the anonymity set for time intervals of *5-minutes*, *30-minutes*, *2-hours*, *4-hours* and *12-hours* are, respectively, 28.95, 34.95, 46.40, 60.39, and 82.05. This attack degradation is because the attackers have to observe online and offline nodes over a longer time interval.

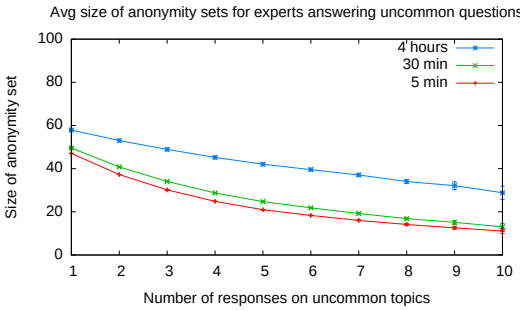
In general this adversary is very strong, since we assume the adversary can link responses from the same expert together. Based on the design of our system, we assume such linking is possible through *stylometry* (i.e., linking texts to a particular author, or recognizing a set as coming from the same author). More work is needed to see how feasible this attack is in practice and how easily it can be defeated [68].

Colluding-Linkable adversaries

As in the previous attack, Figure 5.6 shows the decline in anonymity as more answers are observed and linked by the colluding attackers. Additional graphs for Common and Rare topics are available in Figure 5.7.

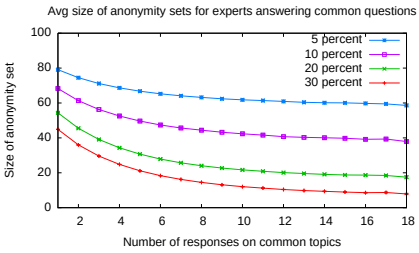


(a) Uncommon Topics with different percentage of colluding nodes

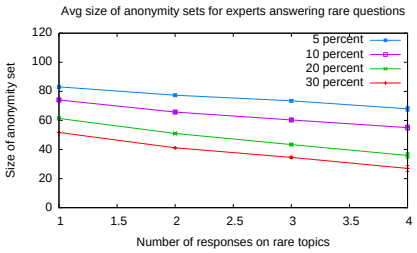


(b) Uncommon Topics With Time Aggregation for 30% colluding nodes

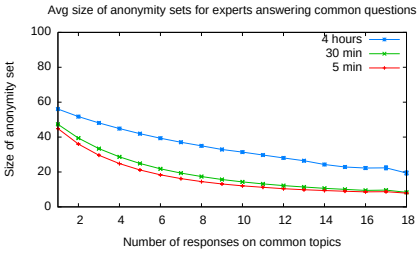
Figure 5.6: Anonymity for Uncommon topics against a Colluding-Linkable attacker after 4 weeks.



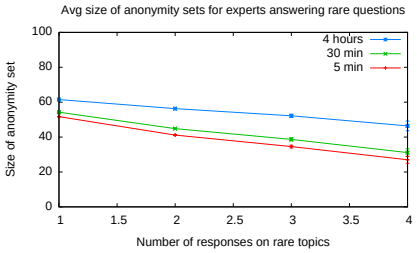
(a) Common Topics



(b) Rare Topics



(c) Common Topics With TA for 30% colluding nodes



(d) Rare Topics With TA for 30% colluding nodes

Figure 5.7: Anonymity for Common and Rare topics against a Colluding-Linkable attacker after 4 weeks.

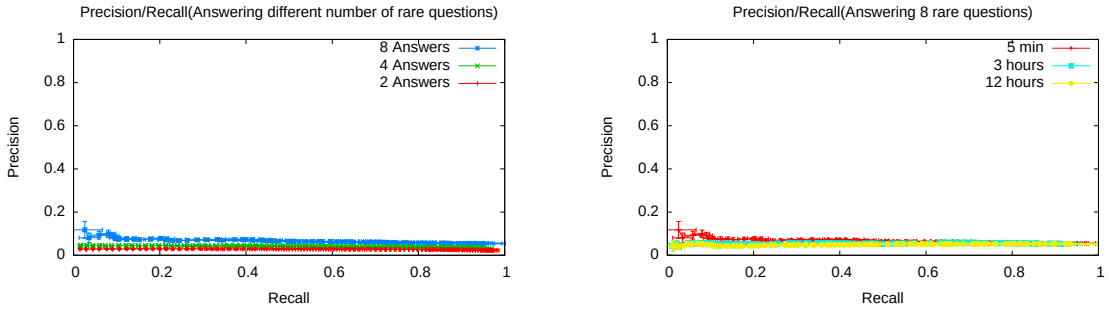
As the percentage of colluding attackers increases, the size of the anonymity set decreases. For example, Figure 5.6(a) indicates that at the end of 4 weeks, the anonymity set for an answerer who answers one question in a common topic with 5% of colluding attackers is about 80 and declines to about 70 and 55 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. We find that for about 10% of 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 5.6(b), shows that time aggregation improves anonymity as expected. For example, the anonymity set after 4 weeks and for 10 answers for a time interval of 5 minutes is 11.1 and increases to 13.1 and 28.84 when the time interval is increased to 30 minutes, and then 4 hours, respectively.

Global-Unlinkable adversaries

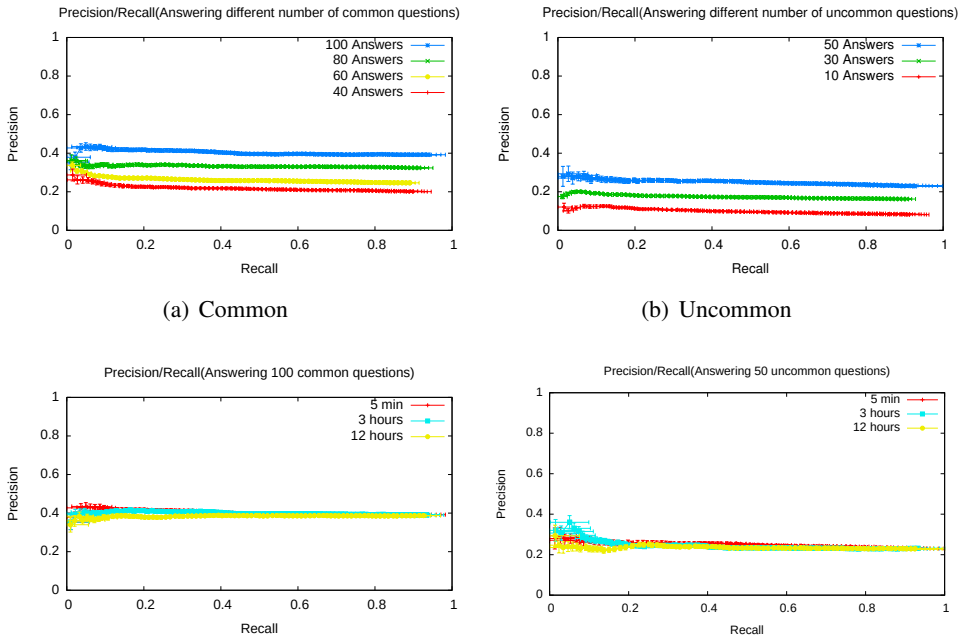
Figure 5.8 shows how Global-Unlinkable attackers can degrade the anonymity of answerers by applying the *counter attack*. Adversaries maintain counts of users present while answers are received for various topics. These counts are used to (try to) implicate users who are present more often when such answers are received. 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. After 4 weeks they sort the list of answerers in descending order of counts. The attackers then "draw a line," e.g., after the top 10 counts to see what amount of precision and recall can be obtained to implicate answerers. For analysis, we iterate over the 1st–85th answerers in drawing these lines to obtain precision vs. recall curves.

Figure 5.8 plots the precision vs. recall curves for the attacker, where each point in the graph corresponds to a particular position of the line in the sorted list of counts. The graphs for Common and Uncommon topics are available in Figure 5.9.



(a) Rare (b) Rare with Time Aggregation

Figure 5.8: Precision vs. Recall for Rare topics against a Global-Unlinkable attacker after 4 weeks.



(a) Common (b) Uncommon

(c) Common with Time Aggregation (d) Uncommon with Time Aggregation

Figure 5.9: Precision vs. Recall against a Global-Unlinkable attacker after 4 weeks.

Our simulation shows that the precision curve is flat for a particular number of answers while the recall increased from 0 to 1. The flat precision shows that this attack is not successful in finding the answerers at the top of the list and that answerers are uniformly distributed in the list. Uniform

increase in recall shows that more answerers are found as you approach the end of the list. 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 with much certainty. Even after 4 weeks the precision is at best 0.1 for rare topics, which provides plausible deniability. We see similar results for common and uncommon questions. 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 who are 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.

Our results show that when people answer more questions on a topic, precision increases and the attack is more effective in de-anonymizing the experts because the counter value for answerers who have mostly answered questions in this topic is higher.

Figure 5.8(b) shows that the time-aggregation defense (for rare topics) makes very little difference for this attacker, as the attack is already quite weak as seen in the previous three graphs.

Since the attack is quite ineffective with Global-Unlinkable adversaries, 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, the anonymity of users degrades over time (albeit less so for Colluding-Linkable adversaries). *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 [68].

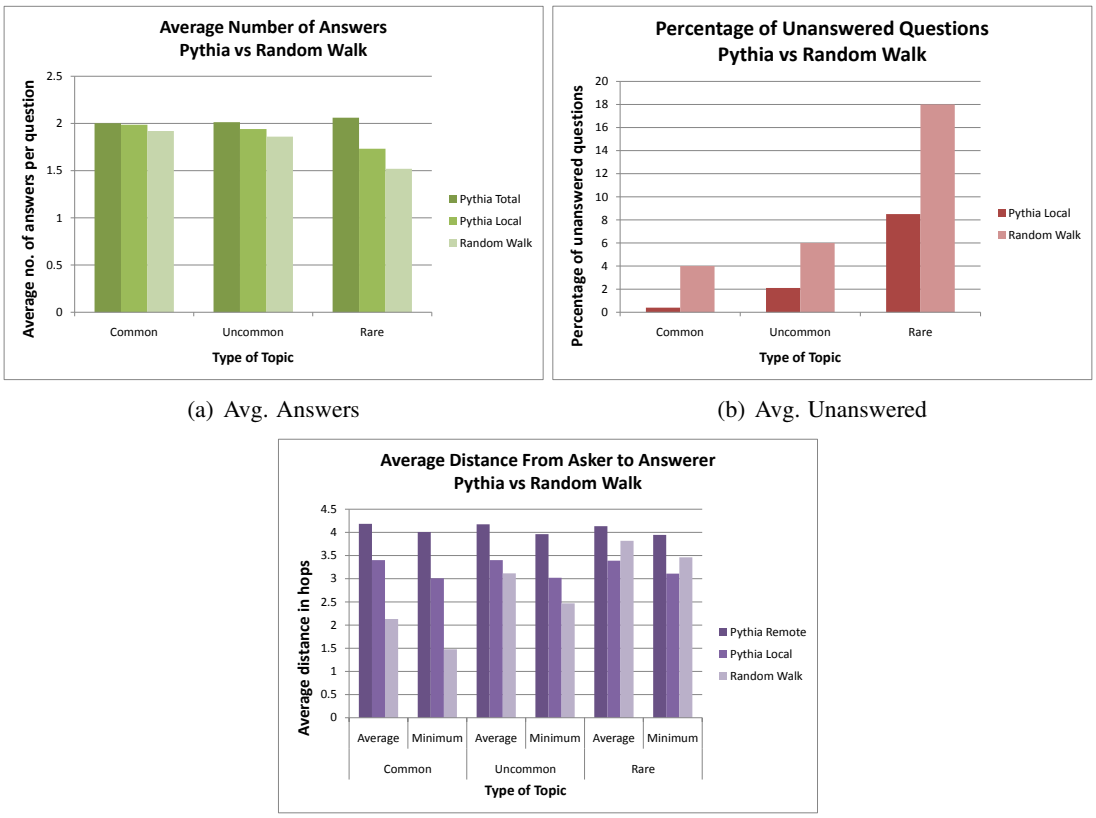
5.4.3 Comparison with Random Walk

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. [143], 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. [143], 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



(a) Avg. Answers

(b) Avg. Unanswered

(c) Avg. Distance

Figure 5.10: Performance Comparison of Pythia with Random Walk

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 5.10(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 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 5.10(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 5.10(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.

5.5 Summary and Discussions

We presented *Pythia*, a privacy-aware P2P system for live social search. We have made the first significant 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, incentivizing use of such systems for sensitive queries, and assigning reputation to anonymous experts.

Shielding. We propose the use of *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.

Metrics. For expertise unlinkability, it would be better to characterize the probability of a person being an expert using the prior probability to then calculate the posterior probability of being an expert based on the size of the anonymity set. This information could be combined with other information relating to the estimate of how many experts are suspected to exist within the anonymity set.

Reputation. Nodes could advertise their expertise along with reputation information for that expertise. For example, users could accumulate anonymous digital cash [46] for answering questions, and then prove their "wealth" as an indicator of reputation.

Incentives. Our current model largely relies on altruism. In addition to reputation mechanisms, simple policies can help control freeriding; e.g., the 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 has answered. Incentive mechanisms could take into account the quality of answers provided, which could then be combined with a reputation system.

In the next chapters, we propose a distributed architecture for online social networks so that provides general functionalities of these networks such as posting and commenting on friends' walls and viewing friends' updates on their news feeds.

Chapter 6

Decentralized P2P Architecture for Privacy Preserving Social Network

In Chapter 3, we argued that in current centralized OSNs, users are not in control of their own data. A scalable alternative architecture for OSNs is distributed P2P architecture where no central entity is in control of all data. In this chapter, we propose a general decentralized architecture that provides strong security and privacy guarantees while preserving the main functionality of online social networks. The main requirements for a privacy preserving social network include confidentiality, integrity and availability of user content, as well as the privacy of user relationships.

Unfortunately, alternative decentralized designs [40, 70, 87, 94] remove reliance on a central entity, but have not adequately tackled confidentiality and access control issues.

In this chapter, we explain how Cachet benefits from a combination of techniques to satisfy these requirements where users' data, including their status updates, posts, comments, and photos are stored as encrypted objects in a distributed hash table (DHT). Applying attribute-based encryption (ABE) to encrypt data guarantees fine-grained access control, while use of the DHT guarantees availability of the data.

Contributions:

1) We propose a decentralized OSN architecture that: i) provides flexibility in data management through OOD; ii) uses an appropriate and advanced cryptographic scheme that supports efficient access revocation and fine-grained policies on each piece of data; and iii) combines confidentiality, integrity, and availability by using the functionalities of a DHT—all the existing designs focus on one or two, but not all of these aspects. The novelty of our architecture lies in integration of existing primitives that are tailored to enhance the security and privacy

of OSNs.

2) We develop a prototype of Cachet—the wall and newsfeed functionalities, to be specific—and evaluate its performance through simulation and experiments on PlanetLab. We evaluate Cachet using a FreePastry simulator and a Kademlia implementation on PlanetLab [176,213]. Our preliminary analysis shows that the performance overhead of Cachet is acceptable, and that our architecture for privacy-preserving decentralized OSNs is feasible.

This chapter is based on collaborative work with Sonia Jahid, Prateek Mittal, Nikita Borisov and Apu Kapadia and was published at the 4th IEEE International Workshop on SEcurity and SOCial Networking (SESOC) [138]. Although the system was originally called DECENT, was renamed Cachet because of the revised architecture with enhanced social caching.

6.1 Requirements and Properties

6.1.1 Functional Model

The main OSN functionality is sharing content with social contacts, who may view, share, or comment on them. 'Wall' (main/profile page) and 'Newsfeed' are the key applications in most OSNs where the former lists all activities of a user based on the time and the latter aggregates and displays recent updates from a user's social contacts. Implementing these applications efficiently is a key challenge in a decentralized OSN.

6.1.2 Privacy and Security Requirements

The primary security requirements are confidentiality and integrity of user data, stored in distributed and untrusted storage nodes, and availability of the correct and latest version of the data. Users should be able to have complete control over the permissions to content they create and no user should be able to access content unless explicitly authorized by the owner. Finally, user relationships should remain hidden from third parties, such as the storage nodes. These can be listed as the following:

Confidentiality: Preserving the confidentiality of user content is a key requirement for a decentralized OSN. Content should be accessible to only those who are explicitly authorized by the content owner. Furthermore, nodes hosting such data may themselves not be authorized to read the data.

Integrity: We must also ensure the integrity of the data so that OSN users can be certain that content posted by their friends is authentic. This property is important in a peer-to-peer network since storage nodes are untrusted and may try to perform unauthorized updates to the stored data.

Availability: User content should remain available until it is explicitly deleted by its owner, even if the owner is offline, and despite potential malicious attempts to destroy the data. Readers should also be able to retrieve the most recent version of a content object rather than past ones.

Explicit Owner-specified Access Control: Policies controlling who may view, modify, or comment on content are defined by its owner and cannot be changed without the owner's authorization.

Relationship Privacy: Relationships between users should remain hidden from third parties that may have no relationship with the object owner and are therefore untrusted, such as storage nodes.

6.1.3 Threat Model

We assume that the participants in the decentralized OSN may be malicious (or compromised), Byzantine, and capable of launching both active and passive attacks. Distributed systems are vulnerable to the problem of Sybil attacks [100]. However, existing mechanisms are available to defend against them [76, 166]. We consider that up to 25% of the nodes in the system can be malicious, since, beyond that, existing mechanisms [76] are not able to securely route in distributed hash tables, which is a necessary prerequisite to provide both integrity and availability guarantees. In peer-to-peer networks, malicious nodes may attempt to launch denial of service (DoS) attacks against honest peers, by overwhelming their network, computational, or storage resources. We assume existence of defenses against such DoS attacks [102, 195]. Basically, routing-based attacks or DoS attacks,

which can be addressed by existing mechanisms, are not in the scope of this work, and, therefore not discuss further.

6.2 System Architecture

We propose a distributed P2P architecture for social networking that preserves privacy through a combination of design features provided by: data objects, cryptographic mechanisms, DHT, presence algorithm, and, social-based caching algorithm. The modular design allows us to separate the various cryptographic implementations (ABE, signatures) from the basic object model (container objects, permissions, etc.) and use any type of DHT.

In this decentralized architecture, data is represented in form of objects, which are encrypted and signed to provide confidentiality and integrity. Access control policies are enforced by attribute-based cryptography. A Distributed Hash Table (DHT) is also deployed to store and retrieve data objects created by their owners.

The basic prototype supports user profile and wall features including status updates, wall posts from social contacts, commenting on posts, and a basic newsfeed algorithm. Then, we extend this prototype to support newsfeed application which aggregates the most updates of social contacts.

6.2.1 Data objects

The content can be in form of status, post, comment, and albums that may contain phrases, web URLs, videos, or photos. We define a *container object* that consists of a main content object and a list of annotations. The main content takes any of mentioned types while annotations are references to other container objects. In our design, access control is enforced by defining different set of permissions for a container, internally referenced objects, and each individual annotation while access policies can be defined over social contacts and their attributes, as well as social network distance (e.g., 'friend-of-friend').

6.2.2 Access Policies

When users create objects, they may define policies on them at the same time. An object is protected with three types of policies:

- *Read Policy*: identifies those who can view/read the content of the object.
- *Write Policy*: identifies those who can delete or overwrite the object; by default, it is set to the owner of the object.
- *Append Policy*: identifies those who can append to (comment on) an object.

Policies can be defined based on user identities (e.g. *Alice AND Bob*), attributes (e.g. *friends AND family OR colleagues*) or both. The *Read Policy* is enforced through the use of attribute-based cryptography while the *Write* and *Append Policies* are enforced through a combination of cryptography and specialized DHT functionality.

6.2.3 Cryptographic Protection

Storing data objects on untrusted DHT nodes, it is necessary to use of cryptographic algorithms to protect confidentiality and integrity.

Confidentiality: Attribute-Based Encryption (ABE) [61] is appropriate for enforcing access policies in a system that attributes can be assigned to a group, list or circle of social contacts [50]. In ABE, an object is encrypted with an attribute-based policy and some public parameters. Social contacts who possess a decryption key for any attribute that satisfies the policy, can decrypt the object. In our system, a user issues different encryption/decryption keys to social contacts based on their attributes.

However, ABE solely does not provide confidentiality because it attaches the policy to the object, thus, storage nodes get information about the policy used for encrypting the object. To provide policy privacy, the *Read Policy* is enforced through the object reference rather than the object itself where the object is encrypted with a symmetric key, and the symmetric key, placed as a part of the reference, is encrypted with ABE. Therefore, confidentiality is ensured through a hybrid approach.

Integrity: Integrity of objects is guaranteed when object owner digitally signs the content excluding the list of comments; each comment reference is signed individually using a different signature key. For signing an object, the owner generates a different pair of signature keys and place the public part of the key (WPK) within the object. Thus, anyone, including the storage nodes, can retrieve the key and verify the integrity of the object. However, the storage nodes are also able to remove or change the public part of the key if it is not encrypted or signed. To avoid this, the public part of the key is also placed in the reference in the parent object. This is how the *Write policy* is enforced.

Append Policy is also enforced by controlling the signature keys. The *Append Policy* secret key (ASK) is encrypted with attribute-based policy. Comment references are signed by commenters using *Append Policy keys*, thus ensuring the integrity of the comment. The APK is stored in the object, and, it is authenticated by the *Write Policy* signature. Because APK is readable by the storage node, it can verify if the append carries the correct signature from the append-policy key.

An object reference looks as follows:

$$objRef \stackrel{\text{def}}{=} (objID, \text{ABE}(K, P), WPK)$$

where *objID* is a random object identifier used to locate it in the DHT, K is the symmetric key used to encrypt the referenced object, P is the attribute-based read policy, and *WPK* is the Write-Policy signature public key.

Data Storage

Participants in the OSN are organized into a distributed hash table (DHT), such as Pastry [213] or Kademlia [176]. The DHT creates a scalable key-value store with an efficient lookup mechanism to locate nodes that store a given object. DHT lookups can be made secure against attacks [76], ensuring that a lookup will find the correct copy of an object if it exists. (It may additionally find incorrect copies provided by malicious nodes; cryptographic integrity protection described above

can be used to identify the correct one.)

Data is stored as an object in the DHT using *objID* as the DHT key. To ensure availability despite node churn and malicious attacks, several replicas of an object are maintained. DHTs typically use the neighbor set of the node responsible for the object key to maintain replicas; the number of replicas needs to be tuned based on the churn patterns of the network (malicious nodes can also be modeled as churn in this case), which we will study in our future evaluation. To guarantee freshness, each object has a version number as part of its metadata. The version number is authenticated by the write-policy signature, thus a user can query all of the replicas and use the freshest object returned.

Malicious users may try to modify or delete an existing object. Note that the write policy prevents them from creating modifications that will be accepted by the readers, as they cannot produce a correct signature, but they may overwrite and destroy legitimate data. To address this, the write-policy public key is stored unencrypted as part of the object metadata. The storage node will refuse to overwrite the stored object unless the new data is properly signed by the write-policy key; deletions must likewise be authenticated with a signature. Thus, as long as there is always at least one honest (but curious) replica for an object, it will persist despite any malicious attacks. The write-policy public key *should not* be a user's permanent public key, as otherwise a storage node could use it to link an object to its owner. Instead, as it was explained in previous section, a separate *WPK* is generated for each object, ensuring unlinkability of objects. We use the Digital Signature Algorithm (DSA) for write-policy signatures because it allows new keys to be generated very quickly.

In addition to the standard *get* (read) and *put* (write) requests, our proposed DHT supports an *append* request, which is used to add a comment reference to an existing object. As with write requests, the storage node verifies that the append carries the correct signature from the append-policy key, using APK that is stored in the clear as part of the object.

Thus, many security and privacy issues are taken care of through existing mechanisms: lookups can be secured against attacks [45, 76, 144, 187]; availability is ensured through replication; and

malicious data overwrites are prevented through write-policy verification. As long as at least one replicating node is honest, vandalism by the storage nodes themselves can be prevented as well.

6.2.4 Example

Join: Alice generates her ABE public and master secret keys and signature key pair. Then, she creates a root object that will contain references to other objects such as profile and wall objects. The root object can be thought of a user's landing page on Facebook. She encrypts it with a symmetric key, and signs it with her write-policy signature key SPK_{Alice}. She generates a random ID, saves the root object in the DHT using this ID, ABEncrypts the symmetric key with Read-policy. Similarly, Alice sets up her profile and wall objects. She also creates references to the profile and the wall objects and updates the root object by adding these references to it.

Establish Contacts: For each social contact, Alice generates an ABE secret key with the appropriate attributes *friend, family, colleague, etc*. Keys and root object references are exchanged out of band.

Write an Status or a Post: Figure 6.1 shows an example object structure. Alice writes a status update on her wall by creating the status update object, completing with version number, contents, and public and secret keys for the write and append policies (WPK_1, WSK_1, APK_1, ASK_1). She also generates a signature over the write-policy signature key (WSK_1). She then picks a random symmetric encryption key K_1 and encrypts the object (except for WPK_1 and APK_1 and the signature); she also chooses a random id ID_1 and uses this to insert the object into the DHT. Finally, she creates a reference to the status update, including ID_1, K_1 and her write-policy public key (WPK_1) and adds it to her wall. K_1 is encrypted with an attribute-based policy P_1, which governs who can read the status update. Note that Alice's wall will also be encrypted with a symmetric key K_0, which is part of the reference to Alice's profile that she gives to her friends.

View a Status, Post, or, Comment: To read Alice's update, Bob finds the reference on Alice's wall. If Alice has intended for Bob to read the update, Bob's attributes should satisfy the attribute-

based secret key. Using this key, he decrypts the reference and gets K_1. Note that Bob has to satisfy the Read Policy associated with the wall object itself to get access to the reference. He then retrieves the object from the DHT with the key ID_1 and decrypts the encrypted fields using K_1. Finally, he verifies the signature to ensure the authenticity of the object.

Comment on a Status, or a Post: To comment, Bob, first, creates a comment object and uses the *append* operation to insert a reference to the new object into Alice's update. Assuming he satisfies the Append policy AP_1, Bob decrypts ASK_1 and generates a signature on the reference. The encryption key K_2 is further ABEncrypted using Bob's policy P_2; thus, only users who satisfy both P_1 and P_2 will be able to read the comment.

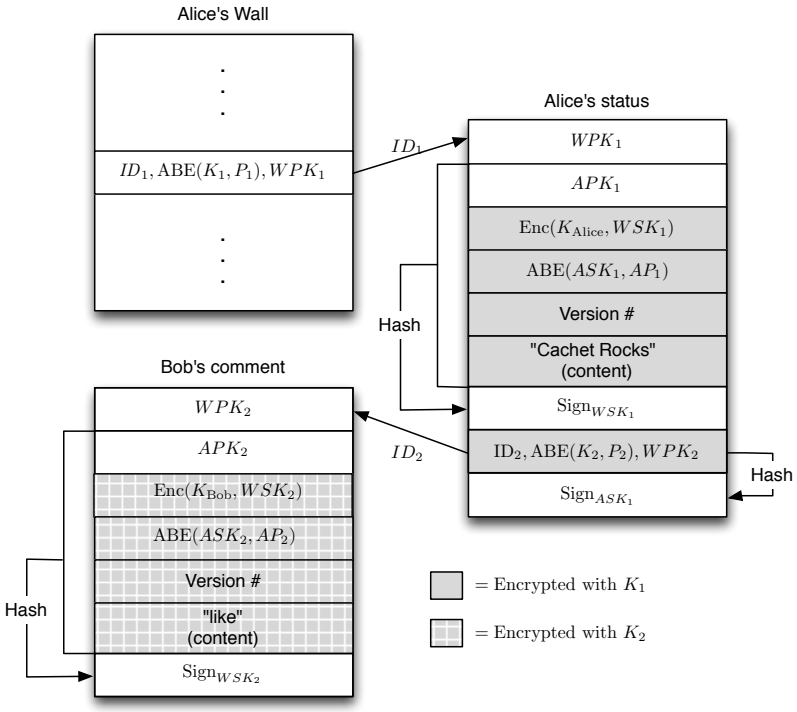
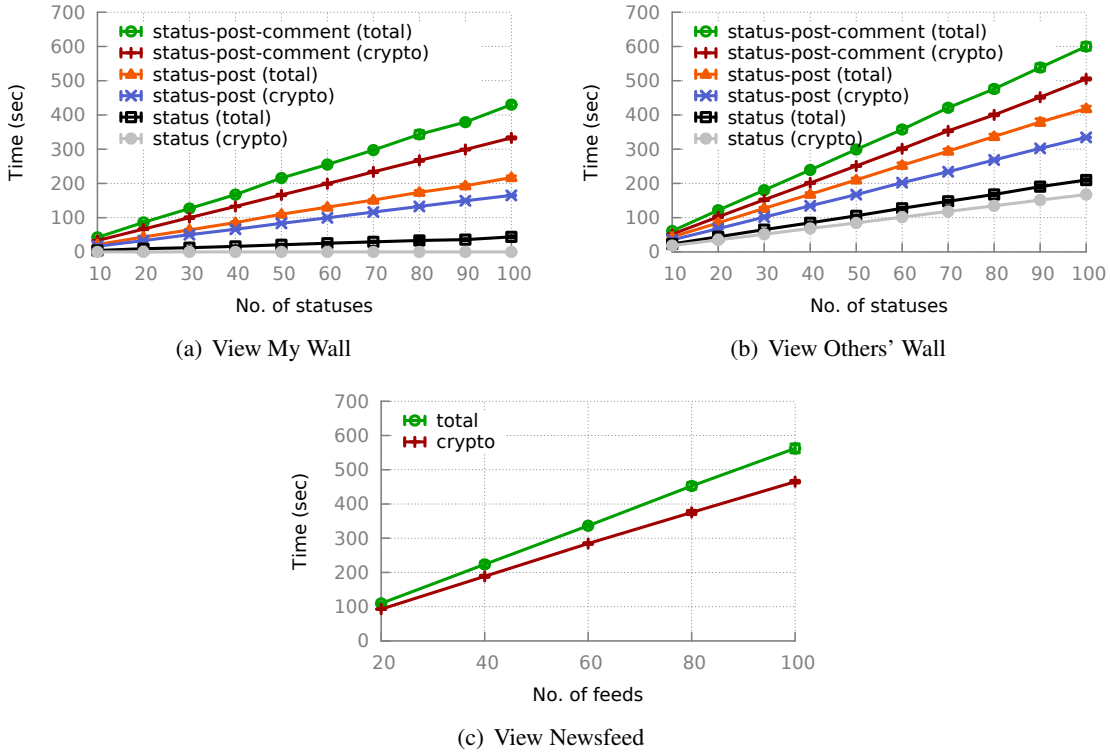


Figure 6.1: Example: data objects in our proposed P2P decentralized OSN

6.2.5 Implementation and Evaluation

We implemented a preliminary prototype of Cachet, which provides functionality similar to the Facebook wall. It also provides a basic newsfeed feature. A user's newsfeed is a collection of the latest *status update objects* from each of her social contacts. We use four different types of crypto-

graphic schemes in Cachet: EASiER for ABE, AES for symmetric encryption, DSA for signatures, and RSA to encrypt the write policy signature key. The key sizes are chosen as recommended by NIST [52] for maximum security. We use FreePastry with Euclidean network topology for simulation, and Kademlia [41, 42] for the experiments on PlanetLab as the underlying DHT. Our proxy was run on a standard server for simulation and PlanetLab experiments.



(a) View My Wall

(b) View Others' Wall

(c) View Newsfeed

Figure 6.2: Simulation Results for 10,000 nodes: The average time to view another user's wall with 10 and 20 status/post/comments is about 60 s and 120 s respectively. The average time to view a newsfeed with 20 peers is 109 s.

6.2.6 Simulation

We performed experiments to measure the performance of viewing a user's wall with varying numbers of status messages, posts, and comments. The simulation was run on a peer-to-peer network of 10 000 nodes. Figure 6.2 shows our simulation results.

View Wall: To view a wall, a user uses the wall reference to fetch the wall object, ABDecrypts the wall reference to get the AES key to decrypt the wall object, and gets the wall metadata and the

references to statuses and posts. Each reference is used to fetch the corresponding status or post. The user ABDecrypts the reference to view the content of the object.

We performed tests to view wall with: 1) only statuses, 2) statuses and same number of posts from friends, and 3) statuses, posts, and one comment on each status from friends. Figures 6.2(a) and 6.2(b) show results for viewing one's own wall and friends' walls respectively. A data point such as x statuses means that when a user views a wall that contains statuses, posts, and comments, she views x of each, i.e., $3x$ objects.

We allow users to cache the AES encryption keys for the objects they create and thus avoid ABDecryption of the references to their own objects. Therefore, viewing one's own wall with only statuses is much faster than viewing a friend's wall with only statuses. Viewing one's own wall with statuses and posts involves ABDecryption for the posts from friends and only AESDecryption for the statuses. The same applies to comments from friends. When a user has not posted anything herself to her friends' walls, then viewing friends' walls involves ABDecryption for each item on the wall, and so represents the worst case scenario.

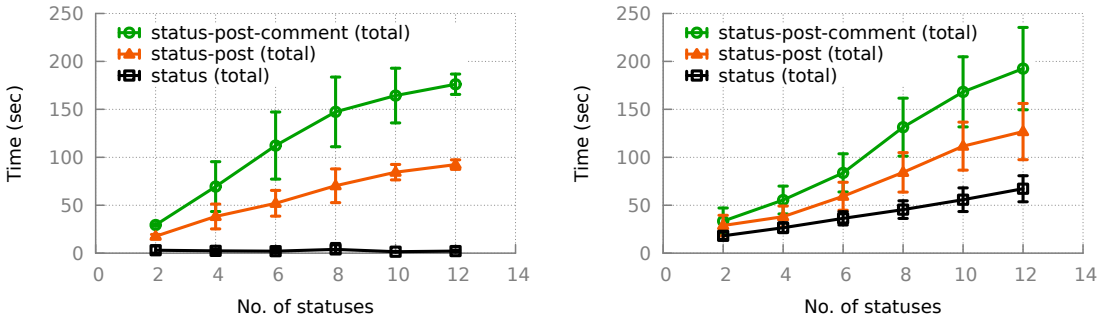
The current view time (e.g., $90\,\mathrm{s}$ to view a user's own wall with 20 statuses, 20 posts, and 20 comments) may appear large; however, content can be displayed progressively, and thus older messages can be fetched while the user is reading the most recent messages, which are loaded within the first few seconds.

View Newsfeed: We also tested our prototype to evaluate the basic newsfeed functionality. This approach fetches the latest status from each of a user's friends. Figure 6.2(c) shows the results. An example newsfeed with 40 feeds takes around $215\,\mathrm{s}$ to construct and view. This simple algorithm is inefficient because the latest status update objects are fetched sequentially, and each of them is decrypted individually, which makes viewing the newsfeed non-practical. This means that if a user has hundreds of social contacts, then she has to wait until all of her contacts' latest status update objects are fetched and decrypted.

Decryption is time-consuming. Besides, we do not perform any type of caching or utilize social

links to expedite the loading of a user's newsfeed. Furthermore, in practice a user may not be interested in viewing all of her contacts' latest statuses, but subscribe to a few selected ones instead. We explore some techniques to improve the performance in Sections 7.

Post and Comment: To post/comment on another user's wall, a user signs the reference to the post or comment with the append-policy signature key of the parent object, which she ABDecrypts from the parent object. The average time to post or comment is 3.94 s. The results are reasonable since a user can continue her OSN activities while the update is performed.



(a) View My Wall (b) View Others' Wall

Figure 6.3: PlanetLab Results: The average time to view another user's wall with 10 status/post/comments is about 168 s.

6.2.7 Experiments on PlanetLab

We performed the same experiments on 15 PlanetLab nodes to get an idea of Cachet's performance in a real deployment. The DHT was implemented on PlanetLab using a Kademlia prototype extracted from the Likir implementation [42]. Figure 6.3 shows the results of our preliminary PlanetLab experiments. As expected, the time to view walls in PlanetLab machines takes slightly longer because of network delays, such as the communication between peers and the proxy. In addition, because of node failures, a few of the users' walls could not be viewed, and in some experiments, walls were retrieved only partially. We also test the time to construct and view the newsfeed. A newsfeed with 11 feeds takes 37.3 s (95% confidence interval is $[34.4, 40.1]$ s), which closely resembles our simulation results. For improved performance and resilience, we investigate the use of

caching and replication parameters.

6.3 Summary

In this chapter we proposed a design for decentralized social networks with an emphasis on security and privacy. Using a collection of mechanisms including the object design, the traditional and advanced cryptographic mechanisms, and the use of a DHT, we showed that it is a possible to have a privacy-preserving P2P OSN that provides confidentiality, integrity and availability of data. Simulation and experiments on PlanetLab showed that such a network is scalable and the latency is manageable. In the next chapter, we propose optimization mechanisms to improve the performance of overall network.

Chapter 7

Social Caching in Privacy Preserving Decentralized P2P Social Network

In Chapter 6, we show that it is feasible to design a decentralized architecture for privacy preserving social networking and implement some basic OSN functionalities such as creating and posting status updates, retrieving and viewing walls, and a newsfeed. This preliminary analysis showed that the performance overhead of Cachet is acceptable; however, the results for viewing newsfeed indicate that the design suffers from performance issues that arise due to the fetching and decryption of hundreds of small objects belonging to friends, which is required for viewing their walls or for viewing one's own newsfeed.

We therefore propose Cachet, a decentralized architecture for social networks that provides strong security and privacy guarantees while *efficiently* supporting the central functionality of OSNs. For reducing the latency of retrieving an object, we propose the use of trust between social links who act as caches to store unencrypted objects recently seen in the social network. This technique eliminates the need for retrieving the objects from the DHT as well as decrypting the objects. To implement this technique, we proposed a gossiping-based social caching algorithm so that each user coming online identifies online social neighbors and pulls the data objects that he/she is allowed to access according to the ABE policy. When a user identifies online social contacts, by seeing a new data object, he/she pushes it to all online social contacts who satisfy the ABE policy. We also propose a presence protocol to identify online nodes in a P2P network. Similar to data objects, presence objects are disseminated and cached through the network.

Contributions:

1) We proposed a hybrid structured-unstructured overlay in which a conventional distributed

hash table is augmented with social links between users. We use the distributed hash table as a base storage layer, but add a gossip-based social caching algorithm that dramatically increases performance. New updates are immediately propagated to online social contacts. When an offline user comes back online a presence protocol is used to locate online contacts and query them directly for updates. Additionally, these contacts are used to retrieve cached updates from mutual contacts who are offline as well as speed the discovery of other online contacts. The DHT is then used to retrieve updates that may not be cached, ensuring high availability of data.

2) We implemented a prototype of the newsfeed application by using the FreePastry simulator to underlie the DHT network. Our evaluation demonstrates that (a) decentralized architectures for privacy preserving social networking are feasible, and (b) use of social contacts for object caching results in significant performance improvements.

This chapter is based on collaborative work with Sonia Jahid, Prateek Mittal, Nikita Borisov and Apu Kapadia and was published at the 8th International Conference on emerging Networking EXperiments and Technologies (CoNEXT) [193].

7.1 Social caching

Given the use of decentralization and cryptography in our base architecture, downloading and reconstructing a social contact's wall or an aggregated newsfeed is a lengthy process requiring the following steps: 1) decrypting update objects, which are ABEncrypted, to yield metadata such as an update's DHT key and symmetric decryption key; 2) accessing multiple small objects located in different storage nodes using DHT keys provided in the previous step; and 3) decrypting the retrieved update objects with their corresponding symmetric keys. Our preliminary analysis in Chapter 6 indicates that these operations can take hundreds of seconds, and thus the design needs to be improved for practical deployments.

We propose a gossip-based social caching algorithm that, in combination with an underlying

DHT, leverages social trust relationships for dramatically increased performance and reliability. Nodes maintain continuous secure (SSL) connections with online contacts to receive updates directly as soon as they are produced. We describe a presence protocol, which itself uses social caching for finding online contacts. Since ABDecryption of objects is a time-consuming bottleneck, online social contacts who satisfy the ABE policy are leveraged to provide cached, decrypted objects to other contacts who also satisfy the policy for objects related to offline contacts. We emphasize that a data object is not cached by a social contact unless he/she satisfies the ABE policy. Thus, the original ABE policies provided by our base architecture are also preserved by Cachet. The basic object structure in Cachet is extended to include a list of users' IDs that are authorized to decrypt and read the object. Therefore, an object can be forwarded to/cached by the intersection set of one's social contacts and the users in the attached list, and the ABE policies are honored as before.

Our algorithm also seeks to minimize the number of such decryptions (corresponding to DHT lookups for the objects) by dynamically learning which peers yield the most cached objects. Yet, this approach provides reliability by treating the DHT as a persistent store for objects that may not be cached. Moreover, social caching improves data locality because the social contacts of a user are usually geographically co-located, which minimizes the needed bandwidth for downloading an object.

Gossip-based protocols are reliable and robust tools for data dissemination especially when used in P2P and wireless networks [58, 60, 92, 112, 178]. However, relying purely on gossip protocols for disseminating updates through the social network has some drawbacks: 1) redundant information is passed around and stored in the network, even at nodes that do not desire this information; and 2) social circles have correlated patterns of online presence, making it challenging to ensure availability when large parts of a circle are offline.

7.1.1 Presence Protocol

Usually a centralized server keeps track of users' presence information (e.g., their current IP address) in P2P networks [40, 223]. In Cachet a distributed approach is applied where every peer stores a presence object in the DHT so that social contacts can obtain a peer's presence information at any time. The presence object has the same structure as other objects, and is ABEncrypted so that the storage node cannot learn the contents of the presence object.[1] It contains the peer's current IP address, and port. With this IP address, peers can connect to their social contacts directly and maintain live connections. The object is signed so that the storage node only allows the owner to update or rewrite it. Whenever a peer joins or leaves the Cachet network, it updates its presence information.

Note that retrieving and decrypting presence objects for all of one's social contacts will have overhead similar to constructing a newsfeed directly from the DHT. To speed up this process presence objects are cached using gossip-based social caching along with content updates. As such, once a few social contacts have been located, discovery of other contacts can proceed at an accelerated pace. Once links to online contacts have been established, subsequent updates are pushed to online contacts directly using the caching protocol described next.

7.1.2 Gossip-based Social Caching

When a node comes online and joins the Cachet network, it does not have the presence or newsfeed information for any of its social contacts. We now describe the caching algorithm used to progressively retrieve cached, unencrypted versions of these objects to greatly speed up the process of loading the newsfeed. The basic idea of the caching algorithm is for the user to perform a few DHT lookups to get presence objects of some social contacts. Then, she identifies those who are online and contacts them to 1) inform them that she is online, and 2) pull both the cached presence objects and the cached recent updates of their mutual social contacts. The user then uses the new

[1] It could be a separate object or simply embedded in the profile or root object.

unencrypted presence objects to recursively repeat the two steps above until no new contacts are obtained. At this point another DHT lookup is made for a social contact whose status is unknown and the process is repeated until the presence and status objects of all contacts have been obtained. If a user is contacted by a social contact Q who was offline before, she pushes her all cached objects that Q is authorized to read.This algorithm is thus a pull-push based gossiping algorithm because it involves both operations; a joining node pulls information from online nodes, and a node generating an update pushes them to other online nodes.

Algorithms 2 and 3 are employed by a user P when she joins the social network and the algorithm includes the following steps:

1. *Creating the Presence Table*: User P maintains a presence table that lists all social contacts along with their presence statuses. The social contacts are listed in descending order based on the number of mutual social contacts in common with P — a social contact who has the most mutual social contacts in common with P appears at top of the list. Listing social contacts in descending order of mutual contacts is a greedy approach that attempts to minimize the number of DHT lookups and communications that are needed to retrieve data objects. Thus, by contacting the social contacts on top of the table, more data objects can be potentially obtained. Initially, the presence status for all contacts is *undefined*.
2. *Selecting a Contact*: P chooses an unvisited contact Q from the presence list as follows. P chooses the first contact in the presence table whose status is known to be online. If none exist, then it chooses the top contact with an undefined status. If all contacts are visited or known to be offline, P proceeds to step 7;
3. *DHT Lookup and Connection*: If the presence status of Q is undefined, then P retrieves Q's presence object from the DHT and decrypts it. If Q is offline, then it returns to step 2 to select another contact;
4. *Pulling Information*: Since Q is online, P marks Q as visited and creates a secure connection to Q. P uses this connection to pull presence and update objects for mutual social contacts

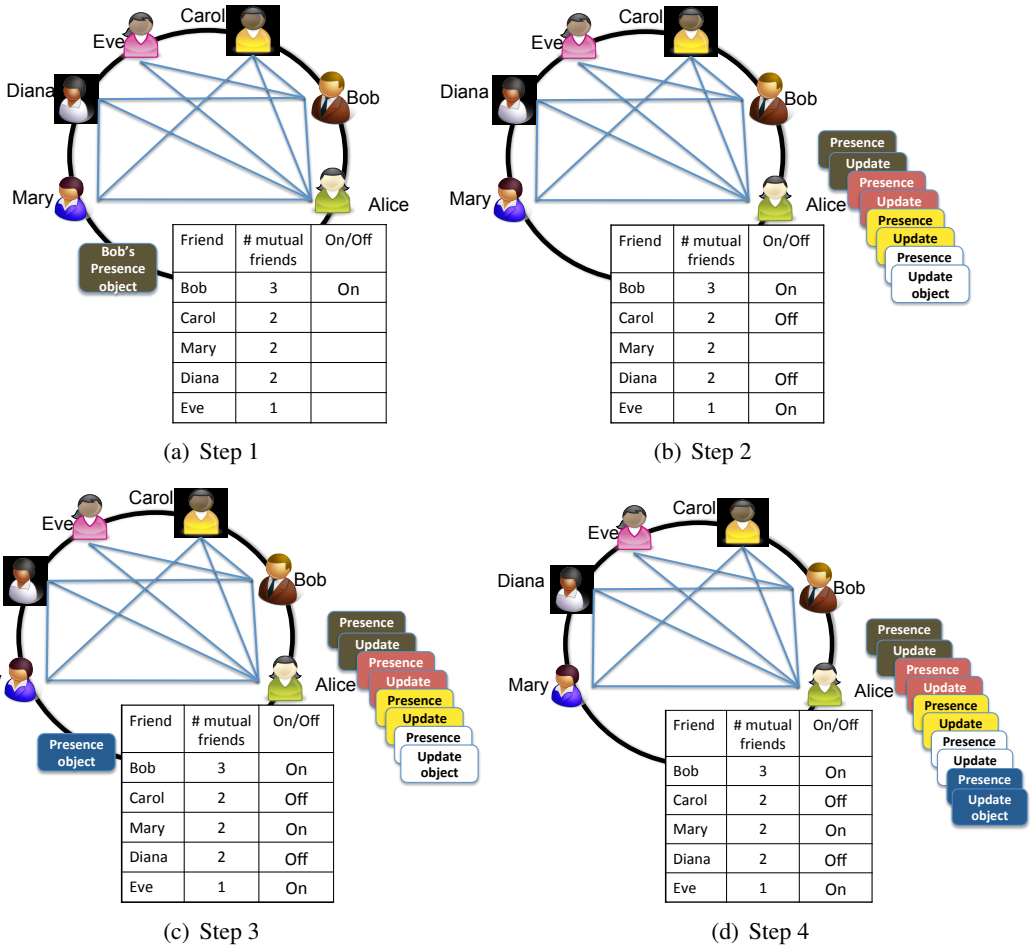
that P and Q have in common;

5. *Caching Information*: P caches the pulled objects (in unencrypted form). We assume that the object cache is unlimited and can store all social contacts' updates during a session. As argued by Mega et al. [178] most of the objects such as status, posts, comments, links, and pictures are small enough; large objects such as videos can be retrieved from the DHT or online services (e.g., YouTube) on demand only;
6. *Updating Presence Table*: P updates the presence table with the online status of social contacts based on information learned from Q. Then it returns to Step 2 to locate the next social contact to connect to;
7. *Performing DHT Lookups for Offline social contacts with No Mutual Social Contacts*: If the recent updates of some social contacts are missing, then this shows that they do not have any online mutual social contacts with P, and P must obtain these objects from the DHT. Thus, to retrieve the newsfeed, P needs to 1) derive the key for their updates by ABDecryption of their reference embedded in the parent/containing object; 2) perform DHT lookups for them; and 3) decrypt the updates by their corresponding symmetric key.

By exchanging presence status and recent updates between online social contacts, the presence table and the cache are always up-to-date. Thus, for viewing the newsfeed, peer P just retrieves recent updates from the cache.

Figure 7.1 shows an example of the Cachet P2P network. The black circle indicates that users are part of a DHT and the blue inner lines in the circle indicate the social and thus P2P connections. As you can see, Alice is connected to Bob, Carol, Eve, Diana and Mary. When she becomes online and joins the Cachet network, it does not have the presence or newsfeed information for any of its social contacts but she progressively retrieves cached, unencrypted versions of these objects to greatly speed up the process of loading the newsfeed. As you can see, Alice's presence table has listed her social contacts based on the number of mutual friends between she and her friends. In the first step 7.1(a), Alice gets and decrypts Bob's presence object from the DHT and updates her

presence table. Since Bob is online, in step 2 7.1(b), she directly contacts him through P2P network and pulls all update of data objects that he has cached for all mutual friends. Then, she updates the table. As you can see, She has received almost all update objects associated with her friends. Bob is not connected to Mary, thus, he does not have any information about her. In step 3 7.1(c), Alice performs a DHT lookup operation for Mary's presence object and updates her table. Since Mary is online, in step 4 7.1(d), Alice directly asks Mary to send her update objects. Thus, the newsfeed with 5 updates is constructed by ABDecryoting only 2 presence objects.



Friend	# mutual friends	On/Off
Bob	3	On
Carol	2	
Mary	2	
Diana	2	
Eve	1	

(a) Step 1

Friend	# mutual friends	On/Off
Bob	3	On
Carol	2	Off
Mary	2	
Diana	2	Off
Eve	1	On

(b) Step 2

Friend	# mutual friends	On/Off
Bob	3	On
Carol	2	Off
Mary	2	On
Diana	2	Off
Eve	1	On

(c) Step 3

Friend	# mutual friends	On/Off
Bob	3	On
Carol	2	Off
Mary	2	On
Diana	2	Off
Eve	1	On

(d) Step 4

Figure 7.1: An example that shows Alice's newsfeed with 5 update objecs is constructed by ABDecryoting only 2 presence objects where all other presence and update objects are provided by social contacts using gossip-based social caching.

7.1.3 Identifying Mutual Contacts and Authorized Users

Many of the benefits of social caching come from being able to identify mutual social contacts. Although relationships between users are privacy sensitive, in practice many users are comfortable sharing this information with at least their immediate social circle. For further privacy protection it is possible to use a social contact discovery protocol that reveals only mutual social contact relationships and nothing else [93].

Since cached content is stored unencrypted, it is also important to verify that a contact satisfies the access policy associated with the object. It should be possible to extend the private contact discovery protocol to learn the attributes shared by P and Q and thus make an authorization decision based on that.[2] For simplicity, however, in our current implementation we instead include an explicit list of authorized users in each container that can be used to mediate sharing.

We note that users who wish to conceal their social relationships, or reveal only a selected subset, may do so, trading off privacy for the efficiency of social caching.

7.1.4 Communicating Not Only with social contacts but Also with FoFs

As illustrated in Algorithm 1, after contacting online social contacts and getting their cached updates, recent updates for some social contacts may still be missing and user P must perform DHT lookups to retrieve them. This situation occurs for a social contact Q who is offline and does not have any mutual social contacts with P who are online (it is possible that P and Q do not have any mutual social contacts at all). To further improve caching performance, a user can easily expand her presence table and not only contact online social contacts but also social contacts-of-social contacts. However, caching presence statuses of FoFs and contacting all of them would be overwhelming. To reduce the state, Cachet maintains information of only a fixed number of FoFs, which is currently set to be equal to the number of social contacts, which grows the list by only a constant factor. FoFs are selected based on the number of mutual social contacts and ordered in the presence table as before.

[2] Briefly, instead of a contact certificate as in [93] one would use a (contact,attribute) certificate for each attribute.

Thus, it is possible that some FoFs are ranked above some social contacts by having more mutual social contacts with the user. Therefore, adding FoFs to the presence table helps eliminate more DHT lookups and decryptions — not only because it eliminates getting cached updates related to offline social contacts with no online mutual social contacts, but because an FoF may have cached more information than a social contact. This extension is possible because the list of authorized users is attached to an object and a FoF can easily verify whether someone satisfies the read policy.

Algorithm 2 User P joins the network

```
//User P joins the network
generatePresenceTable(table);
socialCachingAlg(table, cache);
for social contact Q : table.keySet() do
  if !cache.contains(Q.update) then
    getDHTKeyFor(Q.update);
    encUpdate = dhtLookUp(Q, Q.updateObj);
    update = decrypt(encUpdate);
    cache.put(Q, update);
  end if
end for
```

Algorithm 3 Social caching algorithm(presenceTable table, Cache cache)

```
for social contact Q : table.keySet() do
  Q.visited = TRUE;
  dhtLookUp(Q, Q.presenceObj);
  if Q.presence.status then
    sendTo(Q, Q.presenceObj);
    receiveMessageFrom(Q, bufr);
    if bufr.contains(presenceObj) then
      updateTable(table, bufr);
    end if
    if bufr.contains(UpdateObj) then
      selectUpdatesToKeep(cache, bufr);
    end if
  end if
  SocialContact R = selectSocialContact(&table);
  socialCachingAlg(R, table);
end for
```

Deletion and Revocation

When a user deletes an object or modifies the access policy to an object (including changes to a social contact's attributes) these changes are reflected immediately for data that are ABEncrypted and fetched from the DHT. Affected data in the caches must be updated or invalidated. While we do not specify the protocol here, in short: users can send *object invalidation requests* to remove deleted objects from caches, and *revocation requests* to update the access policies for cached objects, i.e., the list of names to be removed from the access lists for various objects. We leave the evaluation of such deletion and revocation to future work.

7.2 Evaluation

In this section, we evaluate the performance of our presence and social-caching algorithms. We do not compare Cachet's performance with other caching mechanisms [66, 150, 178, 236, 247] since they are not specifically designed for providing security and privacy as in our setting.

7.2.1 Implementation and Simulation Setup

We built a simulator for Cachet based on the FreePastry simulator [213], which implements the underlying DHT. We simulate the cryptographic operations for EASiER [137] with 1 attribute policy and 100 revocations run on a standard machine with 2.40GHz Intel Core 2 Duo, 4GB memory, and running Ubuntu 10.04. With this setting, the ABDecryption takes 422ms. The symmetric key decryption (openssl aes-128-enc) takes 0.04ms on a file of size 2500 bytes, the average size of a status update object. We simulated the communication overhead between peers by setting the average communication latency to be 180 ms.[3] To simulate the social graph in Cachet, we used the Facebook friendship graph from the New Orleans regional network [233]. This data set contains a list of all of the user-to-user links from the Facebook New Orleans network and consists of 63,732 nodes and 1.54 million edges. This data set has been used for simulating social graphs in other

[3] `http://pdos.csail.mit.edu/p2psim/kingdata/`

published work [180, 192, 234].

We evaluated the performance of Cachet by averaging results over the following unit experiment: we used FreePastry to setup a DHT amongst all nodes in the social network, except a particular random user P. We then generate updates for the entire social circle of node P, and simulate Cachet's algorithms. Although our system could be used to cache comments on objects as well, for evaluation we considered a model where newsfeeds include status updates only; we assume a usage model where users click on a particular item to fetch specific comments for that item.

Next we introduce churn in the network, and consider different percentages of nodes amongst P's social contacts and FoFs that remain online — 10%, 30% and 50%. We focus on an online/offline model where different percentages of online friends will affect the caching performance. We note that we are not attempting to evaluate the impact of churn at the DHT layer. In this work we are assessing the effect of nodes joining/leaving and impacting the performance at the caching layer based on how many social, trusted contacts are available. Due to the lack of pertinent data about online/offline patterns in OSNs, we picked various percentages. The 10–30% range is perhaps more pertinent because, for example, Skype has about 45M concurrent users online and 200M active users per month.[4]

We then simulate the node join process for node P, and measure the performance of the newsfeed application.

Performance metrics. We measure performance using the following metrics.

- *Hit Rate*: the percentage of the newsfeed or the presence objects that has been provided by social contacts. To measure worst-case performance, we assume that the number of updates on a user's newsfeed is equal to the number of her social contacts.

  Let e_m be a single unit experiment for a user u with m social contacts. Let d be the number of DHT lookups (involving ABDecryptions) that have been performed for obtaining either u's newsfeed or u's social contacts' presence objects, then:

[4] Skype Reaches A 45M Concurrent User Peak, And What Looks Like A New Stage Of Momentum, TechCrunch, Oct. 14, 2012.

$$hitRate(e_m) = \frac{m-d}{m}$$

This metric measures what fraction of objects were found in the cache.

- *Progressive Hit Rate*: the percentage of the newsfeed or the presence status objects that have been obtained after d DHT lookups and pulling social contacts' cached objects. Let e_m be a single unit experiment for a user, u, who should retrieved presence information of m social contacts and let $\sigma(d)$ be the number of obtained social contacts' presence or status objects after d DHT lookups, then:

  $$hitRate(e_m) = \frac{\sigma(d)}{m}$$

  This metric gives an indication of what percentage of the total objects have been obtained after some number of lookups.

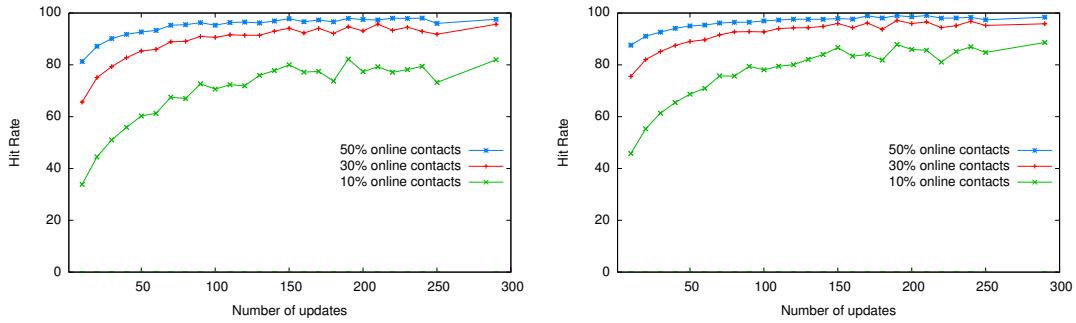
7.2.2 Results

- *Social caching provides most of the update objects for viewing the newsfeed*

  Figure 7.2(a) depicts the average *Newsfeed Hit Rate* as a function of number of updates (equal to the number of social contacts of a person) and the fraction of online social contacts. As it is expected, the Newsfeed Hit Rate increases with a larger percentage of online social contacts. However, interestingly, retrieving a larger newsfeed where each of its objects corresponds to a social contact does not decrease the Newsfeed Hit Rate, because more online social contacts are available to push the cached data to the user.

  We repeated a slightly different experiment where online FoFs are also be contacted by a user to get cached objects belong to their mutual offline social contacts by adding them to the presence table. Although Figure 7.2(b) shows that leveraging FoFs increases the hit rate slightly, the difference is not very high. Thus, we decided to not include FoFs in our algorithm given the additional overhead of caching FoF objects.

  These figures are dependent on the social network graph, and they show what fraction of updates



(a) Only social contacts are contacted

(b) Social Contacts and if needed FoFs are contacted

Figure 7.2: These figures depict the average *Newsfeed Hit Rate* as a function of the number of updates and the fraction of online social contacts. It can be seen that social caching provides most of the update objects needed for viewing the newsfeed. By comparing the two figures it can be seen that leveraging FoFs increases the hit rate slightly, but the difference is not great.

can be provided by social contacts using the social caching algorithm. The results indicate that by using social caching one can rely on her social network to provide most of her newsfeed objects. However, social caching alone is insufficient to ensure availability of the complete newsfeed, necessitating the DHT storage layer.

- *Social caching decreases the latency for retrieving the newsfeed*

  The results illustrated in Figure 7.2 imply that with social caching, the latency for viewing the newsfeed would be much lower than loading all objects from the DHT and decrypting them. To investigate, we examined the latency for retrieving the newsfeed in Cachet both with and without social caching enabled.

  To calculate the latency, in each single experiment, we considered the simulation time for 1) the communication latency between peers, 2) ABDecrypting the references to both presence and update objects that are not provided by social contacts, 3) performing DHT lookups for retrieving these objects, and 4) decrypting the objects.

  Figure 7.3 shows that using just the base architecture, the latency for obtaining newsfeed is very high and is also highly dependent on the number of updates; it is also dominated by the AB-Decryption latency time. In contrast, applying social caching, the latency decreases by up to an order of magnitude. The time needed to view a newsfeed decreases as the number of online so-

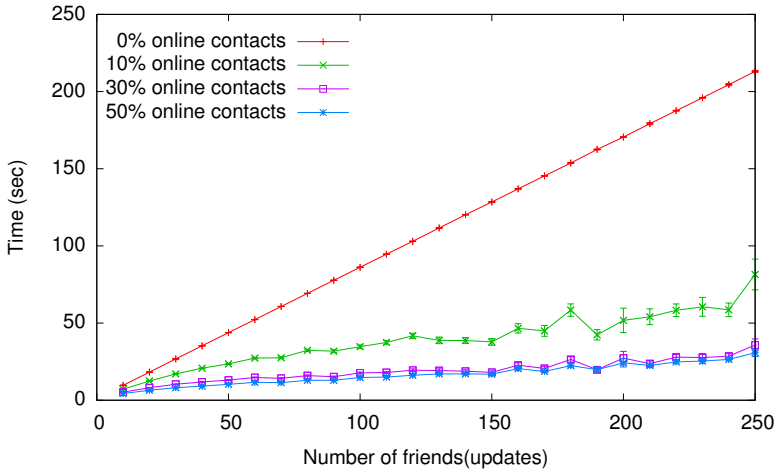


Figure 7.3: This Figure shows the latency for retrieving the newsfeed in Cachet both with and without social caching enabled. It can be seen that even with only 10% of social contacts online, social caching provides a dramatic performance improvement.

cial contacts grows, but even with only 10% of social contacts online, social caching provides a dramatic performance improvement.

- *Most of the presence objects would be available after a few DHT lookups and decryptions*

  We measured the number of presence (or newsfeed update) objects that is provided by online social contacts after contacting them. Figure 7.4 plots the *Average Progressive Hit Rate* after d DHT lookups and ABDecryptions. For this experiment, we plotted the *Average Progressive Hit Rate* for users with $100 \leq m \leq 200$ where m is one's number of social contacts. As it can be seen, having 50%, 30%, and 10% online social contacts, one can receive about 70% of all presence objects by performing about 2, 5, and 18 DHT lookups (and ABDecryptions) respectively.

  This figure shows that users do not need to wait until all presence (or update) objects are retrieved. Furthermore, they can identify most online social contacts (and their updates) by looking up only a few presence (or update) objects and contacting the social contacts who are identified to be online. Thus, user perceived latency can be reduced by rendering the newsfeed when a threshold number of objects are retrieved, and then the feed is updated as more objects are fetched. For example, with 30% online social contacts, 70% of the newsfeed can be loaded after only a tenth

of the total lookups, corresponding to load times of the majority of the newsfeed in under a second.

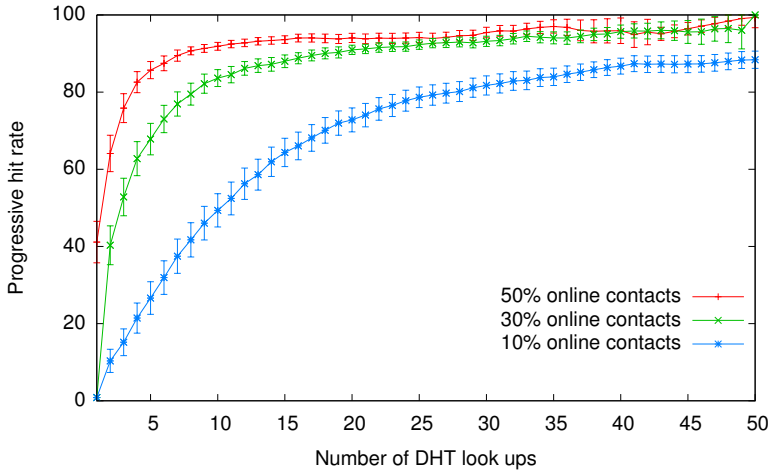


Figure 7.4: This Figure shows the *Average Progressive Hit Rate* for users who have 100-200 of social contacts. It can be seen that after a few DHT lookups, users learn about online/offline status of most of their social contacts. Similarly, most of newsfeed updates can be retrieved after performing a few DHT lookups and contacting the identified online social contacts.

7.3 Summary

By leveraging social trust between friends in a social network, we proposed gossip-based social caching and presence protocol that limit the number of DHT lookups and ABDecryptions, thus reduce the time latency for viewing a wall or newsfeed. Social contacts in Cachet not only provide information about their own updates, but also about updates from other mutual contacts. This use of social contacts makes the architecture practical. Therefore, Cachet is a decentralized architecture for social networks that not only provides strong security and privacy guarantees but also efficiently supports the central functionality of OSNs.

Chapter 8

Conclusions

Online social networks have become an integral part of people's daily lives. These social networking sites offer various opportunities for users to interact with each other and the world. Despite the differences between the services provided by different OSNs, they all collect an extensive amount of information about their users. Users are required to agree to the Terms of Use policies of these sites, which often contain clauses permitting social networking operators to store data on users or even share it with third parties. Public sharing of so much personal information via these sites has raised concern over privacy. We believe that collecting and storing all public and private personal information via these sites raises a great privacy concern.

In Chapter 4, we showed that anonymizing social network data sets before sharing them with others is not enough for maintaining users' anonymity and privacy. We demonstrated that utilizing the 'mesoscopic' properties of a social network, such as its community structure, improves the degree of de-anonymization of an anonymized social network dataset. Our approach leverages 'community detection' techniques to partition the networks into communities. First, it maps the community structure of two graphs and then applies the network-mapping technique to the users inside each community along with a 'seed-enrichment' phase and then to the entire graph. We referred to our algorithm as a "community-aware de-anonymization" algorithm as opposed to community-blind algorithms, which do not use community structure. We defined and calculated the degree of anonymity for both our community-aware algorithm and Narayanan and Shamatikov's algorithm (NS), which is a community-blind algorithm. Then, we run extensive simulations on both synthetic and real social network datasets, and show that our approach provides a significant improvement to the NS algorithm where the degree of anonymity significantly decreases using our

community-enhanced de-anonymization algorithm compared to the NS algorithm. We conclude that the centralized OSN providers may not be able to preserve the privacy of their users.

As an alternative to centralized OSNs, we propose scalable and practical privacy-aware decentralized architectures for socially networked systems by (1) identifying the system's privacy and security requirements, (2) defining proper data structures, (3) utilizing a scalable data-management system for storing and retrieving data, (4) applying appropriate cryptographic algorithms for enforcing access control policies and integrity, (5) defining efficient protocols for disseminating (meta-) data, and (6) employing some optimization algorithms to enhance performance. We demonstrated that our proposed privacy-aware, decentralized P2P architectures avoid centralized control, both giving users control over their data and preserving user-privacy requirements that are specific to a OSN.

In this thesis, we presented three privacy-aware decentralized architectures for socially networked systems:

- First, we presented Pythia: a privacy-aware P2P decentralized social search system. We defined some additional privacy requirements for such a social-based question and answering system, including: *expertise unlinkability*, *interest unlinkability*, *unobservable querying and responding*, and *sender anonymity*. Pythia employs several privacy mechanisms to satisfy these specific privacy requirements.

  In Pythia, users in the social network are grouped into *anonymizing communities* or *flood zones*. Each community has at least one representative who is assumed to be honest but curious. In Pythia askers and answerers both employ onion routing to send their questions/answers to the representative of their community; thus, *sender anonymity* is provided. Representatives forward questions/answers to all users in the community by means of a *local flood*. The local flood allows anonymous answerers to receive questions without having to reveal their areas of expertise. These two techniques together provide *expertise and interest unlinkability*. To provide *asker/answerer unobservability*, all users ask and answer questions

at regular intervals (including dummy questions and answers) and thus attackers cannot readily pinpoint which users are forwarding, asking, or answering questions. In addition, all messages are padded to have the same size.

We evaluated Pythia through simulations against two classes of adversaries: *global attackers* can view all messages exchanged in the system and infer the online/offline and idle status of all the users at any time. *Colluding attackers* have only partial knowledge of this type. Attackers may also have this capability to link messages that are authored by the same answerer (or asker). All of these adversaries are able to perform intersection attacks using this information. Clearly, the global-linkable adversary can perform most powerful intersection attack because of the amount of information that it obtains. Intersection attacks degrade the anonymity of users over time and this degrade is more for global-linkable adversary. However, 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 [68].

- Next, we proposed a more general decentralized architecture for socially networked systems that provides strong security and privacy guarantees while preserving the main functionality of online social networks. The main requirements for a privacy-aware social network include confidentiality, integrity and availability of user content. Our basic architecture employs a combination of techniques to satisfy these requirements: users' data, including their status updates, posts, comments, and photos are managed through object oriented design (OOD). An object contains not only the encrypted data but also meta-data that help enforcement of privacy policies for reading and appending to the object. Attribute-Based Encryption (ABE) is used for enforcing access policies. However, ABE alone does not provide confidentiality, because it attaches the policy to the object; thus, storage nodes get information about the policy used for encrypting the object. Therefore, confidentiality is ensured through a hybrid approach by use of both ABE encryption and symmetric encryption. Integrity of objects is guaranteed by digitally signing the object content. The Append Policy is also enforced by

controlling the signature keys.

Participants in our decentralized architecture for OSNs are organized into a distributed hash table (DHT). Data is stored as an object in the DHT. To ensure availability despite node churn and malicious attacks, several replicas of an object are maintained. Combination of all these mechanisms insures confidentiality, integrity, and availability.

We developed a prototype of this system—the wall and newsfeed functionalities, to be specific—and evaluated its performance through simulation and experiments on PlanetLab. Our analysis showed that the performance overhead of the system is acceptable, but not excellent, and that our architecture for privacy-preserving decentralized OSNs is feasible.

- Finally, we demonstrated that our basic decentralized architecture for privacy preserving social networking is practical if a few optimizations are employed. We proposed Cachet, a decentralized architecture for social networks that provides strong security and privacy guarantees while efficiently supporting the central functionality of OSNs. In particular, the bottleneck in the basic architecture is the huge number of ABE decryptions required to view a newsfeed or a wall. For reducing the latency of retrieving an object, we proposed the use of trust between social links who act as caches to store unencrypted objects recently seen in the social network. This technique eliminates the need for retrieving the objects from the DHT as well as decrypting the objects. To implement this technique, we proposed a gossiping-based, social-caching algorithm so that each user coming online identifies online social neighbors and pulls the data objects that he/she is allowed to access according to the ABE policy. When a user identifies online social contacts, by seeing a new data object, he/she pushes it to all online social contacts who satisfy the ABE policy. We also propose a presence protocol to identify online nodes in a P2P network. Similar to data objects, presence objects are disseminated and cached through the network.

  We implemented a prototype of the newsfeed application by using the FreePastry simulator to underlie the DHT network. Our evaluation demonstrates that Cachet 1) provides adequate

privacy-preserving mechanisms, 2) is resistant to censorship and centralized control, 3) gives users fine-grained control of their data, 4) is scalable, and 5) is efficient for important applications, such as newsfeed.

8.1 Future Research

We now indicate directions for future work in centralized and decentralized architectures for socially networked systems.

- *Need for more sophisticated anonymization techniques.* It is not enough just to remove the identities and sensitive information from the network and release the data sets. Our experiments also showed that even adding more than 20% noise to the network does not prevent attackers from identifying users. These results show that more advanced privacy-preserving anonymization techniques are required that resist against more complicated attacks such as community-aware de-anonymization. Most existing anonymization techniques assume that the adversary only has some background knowledge; however, the attacker may utilize different background knowledge to launch an attack. Although providing a general anonymization technique that preserves the privacy of users against various attackers with different capabilities may not be possible, based on the usage of the data, more powerful anonymization techniques are required. The anonymization problem can be seen as an optimization problem that attempts to maintain a balance between privacy and utility. Modeling and measuring utility of the published social network data is also a problem. Effective metrics need to be defined to quantify the information loss incurred by the changes of nodes and edges when anonymizing data.
- *Possible improvements on community-enhanced de-anonymization of social networks.*

  Use of additional attributes for re-identifying communities and users. Other methods can be applied for mapping the communities in two networks, e.g. leveraging other attributes such as location, language, time, messages/posts, and so on. Future work can explore ways to

combine such techniques in novel ways. For example, these attributes can be studied at the community level before drilling down into individual communities.

Evaluating performance of community-enhanced de-anonymization technique against other anonymization techniques. Based on utility and application of social network data, various anonymization techniques have been proposed. Our de-anonymization approach can be tested against these techniques. For example, Mittal et. al. [181] proposed an anonymization technique that perturbs edges of the social graph while preserves the local structure of the social network. They showed that their technique preserves the community structure of the social graph while introducing a significant amount of noise to the network. They also prove that the expected degree of each node after the perturbation algorithm is the same as in the original graph. Our results in Chapter 4 suggest that this technique may also be vulnerable to community-enhanced de-anonymization approaches, because the community structures are preserved which makes matching communities possible and also the degree distribution of the graph remains the same which makes the seed enrichment phase be still applicable. However, as future work we can evaluate this random walk based anonymization technique using our community-aware de-anonymization approach.

- *Possible improvements on Pythia.*

  Shielding. We propose the use of *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.

  Metrics. For expertise unlinkability, it would be better to characterize the probability of a person being an expert, using the prior probability to then calculate the posterior probability of being an expert based on the size of the anonymity set. This information could be combined with other information relating to the estimate of how many experts are suspected to exist

within the anonymity set.

Advertising. When users join the network, they may advertise their expertise using a DHT-based approach. A representative can then select remote communities more efficiently and purposefully and direct questions to communities with potential answerers for a topic. However, a sophisticated method is needed to resist linkage attacks where joins/leaves of nodes in communities cannot be easily correlated with entries added/removed from the DHT.

Reputation. Users could advertise their expertise along with reputation information for that expertise. For example, users could accumulate anonymous digital cash [46] for answering questions and then prove their 'wealth' as an indicator of reputation.

Incentives. Our current model largely relies on altruism. In addition to reputation mechanisms, simple policies can help control freeriding; e.g., the 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 has answered. Incentive mechanisms could take into account the quality of answers provided, which could then be combined with a reputation system.

- *Possible improvements on Cachet.*

  Privacy issues. While we believe that Cachet's privacy guarantees surpass existing systems, there is still room for further improvement. First, users that do not satisfy the access control policy of a particular object will be aware that they are being excluded from accessing the object, as opposed to being oblivious to its existence (as in current OSNs). Second, our social caching algorithm leaks information about the identities of users who satisfy a particular policy to all of those identities. Finally, our newsfeed algorithm also reveals information about when a user comes online or offline. However, we emphasize that "online" does not necessarily mean that users are logged in and available. It could be that the person's laptop/desktop is connected to the Internet and participating in the DHT and caching protocol, although the person is not at the computer. In future work we will investigate techniques to limit such sources of information leakage.

Deployment challenges. In contrast to the deployment model of today's popular OSNs, users in Cachet face the burden of (a) spending additional computational resources, and (b) volunteering data storage and bandwidth. However, we note that our social gossiping and caching algorithms make the computational overhead of decrypting ABEncrypted objects practical. Furthermore, online social networks are mostly used to store small objects such as status posts and comments, minimizing the burden for users to volunteer excessive storage and bandwidth.

The decentralized architecture of Cachet brings with it several challenges from a networking viewpoint. First, resilience against node churn becomes an important consideration. The underlying DHT should have enough replication to handle temporary instabilities. We note that data objects are also available through our caching mechanism to alleviate instability issues due to churn. In future work we will study the use of less structured overlay topologies, e.g., the use of more stable 'super peers' [36, 173, 200, 242] to further improve stability. Second, users behind NAT make it difficult to realize peer-to-peer connections with other users. Our architecture requires NAT hole-punching mechanisms, such as that of Evans et al. [185]

Social and economic challenges. Existing OSNs such as Facebook and Google+ already have several hundred million users. Thus a significant challenge for new social network architectures is to be able to attract enough users to achieve a critical mass. We believe that enhanced privacy properties of decentralized architectures such as Cachet would give an incentive to users to switch. Furthermore, government regulations or standards encouraging inter-operable OSN architectures can also help to offset the economic challenges for deployment.

Scalability issues. DHTs are designed to be scalable, but as the network becomes very large, e.g., with one billion nodes, scalability concerns are valid — nodes will be involved in more overhead for maintaining the DHT structure, and the amount of cached objects may be larger. We leave such evaluations to future work but comment that our caching algorithm scales with

the number of friends, and thus we do not expect the large network size to overly affect the performance of our caching algorithm.

- *Social-based DHT.*

  It is also possible to investigate how the DHT can leverage social network properties to improve reliability against various attacks and to improve replication and placement strategies. To this end we need to define some attack models that are based on storage and access patterns and then propose defenses against them.

- *Implementing new applications on top of Cachet.*

  It is not clear how to implement applications such as chat rooms, friend search, and friend recommendation system, and identifying trending topics on top of a decentralized P2P social networking system. Each of these applications has special functional, privacy, and security requirements. For example, Identifying trending topics in real time even in Twitter as a centralized OSN is a challenging problem due to the heterogeneity and immense scale of the data [160, 186]. In addition, not all information on Twitter is informational; a study [117] has shown that about 8% of 25 million URLs posted to the Twitter point to phishing, malware, and scams listed on popular blacklists. Twitter itself has a blog, "State of Twitter Spam" [1]: "At Twitter, we see spamming as a variety of different behaviors that range from insidious to annoying. Posting harmful links to phishing or malware sites, repeatedly posting duplicate tweets, and aggressively following and un-following accounts to attract attention are just a few examples of spam on Twitter. Like it or not, as the system becomes more popular, more and more spammers will try to do their thing. We are constantly battling against spam to improve the Twitter experience and we are happy to report that it is working." Spam can impose bias on the results of trending topics algorithms. Some algorithms [57, 218] have also been proposed for distinguishing between messages about real-world events and non-event messages. In our privacy-preserving P2P social network, where there is no central control and

[1] https://blog.twitter.com/2010/state-twitter-spam

users have control over their data, it is more challenging to develop algorithms for identifying trending topics. This is because data objects are distributed through the P2P network, not all objects are public, and spam can be more prevalent.

Bibliography

[1] Amazon. https://www.amazon.com.

[2] Bing social. http://www.bing.com/social.

[3] Bittorrent. www.bittorrent.com.

[4] dailymile. http://www.dailymile.com.

[5] Facebook. www.facebook.com.

[6] Farmville. https://company.zynga.com/games/farmville.

[7] Flickr. https://www.flickr.com.

[8] Freenet. https://freenetproject.org.

[9] Gnutella. http://rfc-gnutella.sourceforge.net/.

[10] goodreads. https://www.goodreads.com.

[11] Google+. https://plus.google.com.

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

[13] Groove. http://www.groove.com.

[14] Hulu. https://www.hulu.com.

[15] Instagram. http://instagram.com.

[16] Kazaa. http://www.kazaa.com.

[17] Linkedin. www.linkedin.com.

[18] Mafiawars. https://www.mafiawars.zynga.com/fbconnect?

[19] Mendeley. http://www.mendeley.com.

[20] Myspace. www.myspace.com.

[21] Napster. http://www.napster.com.

[22] Netflix. https://www.netflix.com.

[23] Patientslikeme. http://www.patientslikeme.com.

[24] Quora. https://www.quora.com.

[25] Rdio. https://www.rdio.com.

[26] Runkeeper. http://runkeeper.com.

[27] Sermo. http://www.sermo.com.

[28] Sparkpeople. http://www.sparkpeople.com.

[29] Spotify. https://www.spotify.com.

[30] Twitter. twitter.com.

[31] Usenet. http://www.usenet.com.

[32] In pursuit of the future internet, 2013. http://www.redorbit.com/news/technology/1112989504/future-of-the-internet-protocol-pursuit-peer-to-ip-network-103013/.

[33] S. M. A. Abbas, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips. A gossip-based distributed social networking system. In *Proceedings of the 2009 18th IEEE International Workshops on*

Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE '09, pages 93–98, Washington, DC, USA, 2009.

[34] Amine Abou-Rjeili and George Karypis. Multilevel algorithms for partitioning power-law graphs. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 10–pp. IEEE, 2006.

[35] Alessandro Acquisti and Ralph Gross. Predicting social security numbers from public data. *Proceedings of the National academy of sciences*, 106(27):10975–10980, 2009.

[36] D. Adami, C. Callegari, S. Giordano, M. Pagano, and T. Pepe. A real-time algorithm for skype traffic detection and classification. In Sergey Balandin, Dmitri Moltchanov, and Yevgeni Koucheryavy, editors, *Smart Spaces and Next Generation Wired/Wireless Networking*, volume 5764 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009.

[37] L.A. Adamic, J. Zhang, E. Bakshy, and M.S. 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.

[38] Eytan Adar and Bernardo A Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000.

[39] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[40] L.M. Aiello and G. Ruffo. LotusNet: tunable privacy for distributed online social network services. *Computer Communications*, 35(1):75–88, 2012.

[41] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. Tempering Kademlia with a robust identity based system. In *P2P*, 2008.

[42] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. An identity-based approach to secure P2P applications with Likir. *Peer-to-Peer Networking and Applications*, 4:420–438, 2011.

[43] David P Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[44] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. In *Proceedings of (ACM) Computing Surveys (CSUR)*, pages 335–371, 2004.

[45] Marc Sanchez Artigas, Pedro Garcia Lopez, Jordi Pujol Ahullo, and Antonio F. Gomez Skarmeta. Cyclone: A novel design schema for hierarchical DHTs. In *Proc. IEEE Int. Conf. on Peer-to-Peer Computing*, pages 49–56, Washington, DC, USA, 2005. IEEE Computer Society.

[46] Man Ho Au, Sherman S. M. Chow, and Willy Susilo. Short e-cash. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 332–346. Springer, 2005.

[47] Michael Backes, Matteo Maffei, and Kim Pecina. A security API for distributed social networks. In *NDSS*, 2011.

[48] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 181–190, New York, NY, USA, 2007. ACM.

[49] Lars Backstrom, Cynthia Dwork, and Jon M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. *Commun. ACM*, 54(12):133–141, 2011.

[50] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM*, 2009.

[51] Marco Balduzzi, Christian Platzer, Thorsten Holz, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. Abusing social networks for automated user profiling. In *Recent Advances in Intrusion Detection*, pages 422–441. Springer, 2010.

[52] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management—part 1: General (revised). Technical Report SP800–57, NISsT, 2007.

[53] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. Algorithms for large, sparse network alignment problems. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 705–710. IEEE, 2009.

[54] Aaron Beach, Mike Gartrell, and Richard Han. Social-k: Real-time k-anonymity guarantees for social network applications. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 600–606. IEEE, 2010.

[55] Filipe Beato, Markulf Kohlweiss, and Karel Wouters. Scramble! your social network data. In *Proceedings of the 11th international conference on Privacy enhancing technologies*, PETS'11, pages 211–225, Berlin, Heidelberg, 2011. Springer-Verlag.

[56] Cheryl Beaver, Richard Schroeppel, and Lillian Snyder. A design for anonymous, authenticated information sharing. In *Proceedings of the 2001 IEEE, Workshop on Information Assurance and Security*, 2001.

[57] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *ICWSM*, 2011.

[58] Sonia Ben Mokhtar, Alessio Pace, and Vivien Quema. FireSpam: Spam Resilient Gossiping in the BAR Model. In *29th IEEE Symposium on Reliable Distributed Systems (SRDS 2010)*, November 2010.

[59] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free mix routes and how to overcome them. In *Designing Privacy Enhancing Technologies*, pages 30–45. Springer, 2001.

[60] Marin Bertier, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. The GOSSPLE anonymous social network. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, Middleware '10, pages 191–211. Springer-Verlag, 2010.

[61] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Security & Privacy*, 2007.

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

[63] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW*. ACM, 2009.

[64] HAN Bo and Paul COOK1 Timothy BALDWIN. Geolocation prediction in social media data by finding location indicative words. *Proceedings of COLING 2012: Technical Papers*, pages 1045–1062, 2012.

[65] Johan Bollen and Huina Mao. Twitter mood as a stock market predictor. *Computer*, pages 91–94, 2011.

[66] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10. IEEE Press, 2010.

[67] Danah Boyd. Facebook\'s privacy trainwreck. *Convergence: The International Journal of Research into New Media Technologies*, 14(1):13–20, 2008.

[68] Michael Brennan and Rachel Greenstadt. Practical attacks against authorship recognition techniques. *Proceedings of the Twenty-First Innovative Applications of Artificial Intelligence Conference*, 2009.

[69] Garrett Brown, Travis Howe, Micheal Ihbe, Atul Prakash, and Kevin Borders. Social networks and context-aware spam. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 403–412. ACM, 2008.

[70] Sonja Buchegger, Doris Schiöberg, Le Hung Vu, and Anwitaman Datta. PeerSoN: P2P social networking — early experiences and insights. In *SNS*, 2009.

[71] John Buford, Heather Yu, and Eng Keong Lua. *P2P networking and applications*. Morgan Kaufmann, 2009.

[72] John F Buford and Heather Yu. Peer-to-peer networking and applications: Synopsis and research directions. In *Handbook of Peer-to-Peer Networking*, pages 3–45. Springer, 2010.

[73] Alina Campan and Traian Truta. Data and structural k-anonymity in social networks. *Privacy, Security, and Trust in KDD*, pages 33–54, 2009.

[74] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, pages 197–210, 2012.

[75] Berenice Carrasco, Yi Lu, and Joana M. F. da Trindade. Partitioning social networks for time-dependent queries. In *Proceedings of the 4th Workshop on Social Network Systems*, SNS '11. ACM, 2011.

[76] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI*, 2002.

[77] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[78] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[79] Shin-Ming Cheng, Weng Chon Ao, Pin-Yu Chen, and Kwang-Cheng Chen. On modeling malware propagation in generalized social networks. *Communications Letters, IEEE*, 15(1):25–27, 2011.

[80] Fabio Ciulla, Delia Mocanu, Andrea Baronchelli, Bruno Gonçalves, Nicola Perra, and Alessandro Vespignani. Beating the news using social media: the case study of american idol. *EPJ Data Science*, 1(1):1–11, 2012.

[81] Dave Clark, Bill Lehr, Steve Bauer, Peyman Faratin, Rahul Sami, and John Wroclawski. Overlay networks and the future of the internet. *Communications and Strategies*, 63:109, 2006.

[82] 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.

[83] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[84] Graham Cormode, Divesh Srivastava, Ting Yu, and Qing Zhang. Anonymizing bipartite graph data using safe groupings. *Proceedings of the VLDB Endowment*, 1(1):833–844, 2008.

[85] Denzil Correa, Ashish Sureka, and Raghav Sethi. Whacky!-what anyone could know about you from twitter. In *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*, pages 43–50. IEEE, 2012.

[86] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Leveraging social links for trust and privacy in networks. In *INetSec 2009, Open Research Problems in Network Security. April 23-24, 2009, Zurich, Switzerland*, Zurich, SUISSE, 04 2009.

[87] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network. In *WOWMOM*, 2009.

[88] Leudo Antonio Cutillo, Refik Molva, and Thorsten 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.

[89] 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.

[90] George Danezis, Claudia Díaz, Carmela Troncoso, and Ben Laurie. Drac: An architecture for anonymous low-volume communications. In *Privacy Enhancing Technologies*, 2010.

[91] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. *Security and Privacy, IEEE Symposium on*, 0:2, 2003.

[92] Anwitaman Datta and Rajesh Sharma. GoDisco: Selective gossip based dissemination of information in social community based overlays. In *ICDCN'11*, pages 227–238, 2011.

[93] Emiliano De Cristofaro, Mark Manulis, and Bertram Poettering. Private Discovery of Common Social Contacts. In *9th International Conference on Applied Cryptography and Network Security (ACNS)*, volume 6715 of *LNCS*, pages 147–165. Springer, 2011.

[94] Diaspora*. `https://joindiaspora.com/`.

[95] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, pages 54–68. Springer, 2003.

[96] Claudia Diaz, Carmela Troncoso, and Andrei Serjantov. On the impact of social network profiling on anonymity. In *Privacy Enhancing Technologies*, pages 44–62. Springer, 2008.

[97] Manal El Dick and Esther Pacitti. Leveraging p2p overlays for large-scale and highly robust content distribution and search. In *VLDB PhD Workshop*, 2009.

[98] Roger Dingledine, Michael J Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer, 2001.

[99] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[100] John Douceur. The Sybil Attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *International Workshop on Peer-to-Peer Systems (IPTPS)*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, March 2002.

[101] Geoffrey B. Duggan and Stephen 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.

[102] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. In *ACM SIGMETRICS*, 2005.

[103] Inc. Facebook. Google privacy policy. `http://www.google.com/policies/privacy/`.

[104] Inc. Facebook. Information we receive and how it is used. `https://www.facebook.com/about/privacy/your-info`.

[105] Inc. Facebook. Facebook reports first quarter 2014 results, 2014. `http://investor.fb.com/releasedetail.cfmReleaseID=842071`.

[106] Mohammad Reza Faghani and Hossein Saidi. Malware propagation in online social networks. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 8–14. IEEE, 2009.

[107] Ariel J. Feldman, Aaron Blankstein, Michael J. Freedman, and Edward W. Felten. Social networking with Frientegrity: privacy and integrity with an untrusted provider. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, Berkeley, CA, USA, 2012. USENIX Association.

[108] Philip W. L. Fong, Mohd Anwar, and Zhen Zhao. A privacy preservation model for facebook-style social network systems. In *ESORICS*, 2009.

[109] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[110] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[111] Daniela Gavidia. A gossip-based distributed news service for wireless mesh networks. In *In Proc. 3rd IEEE Conf. on Wireless On demand Network Syst. and Services (WONS '06*, pages 59–67. IEEE Computer Society, 2006.

[112] Daniela Gavidia, Gian Paolo Jesi, Chandana Gamage, and Maarten van Steen. Canning spam in gossip wireless networks. In *Procedings of the* 4^{th} *IEEE Conference on Wireless On demand Network Systems and Services (WONS)*, January 2007.

[113] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[114] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.

[115] Nathaniel S. Good and Aaron Krekelberg. Usability and privacy: a study of kazaa p2p file-sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 137–144. ACM, 2003.

[116] DINO GRANDONI. Chart: Google+ hit 10 million users 50 times faster than facebook, 2011. http://www.thewire.com/technology/2011/07/chart-google-reached-10million-users-50-times-faster-facebook-or-twitter40295/.

[117] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37. ACM, 2010.

[118] Virgil Griffith and Markus Jakobsson. Messin' with texas deriving mother's maiden names using public records. In *Applied Cryptography and Network Security*, pages 91–103. Springer, 2005.

[119] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 71–80, New York, NY, USA, 2005. ACM.

[120] Saikat Guha, Kevin Tang, and Paul Francis. NOYB: privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 49–54. ACM, 2008.

[121] Seung Chul Han and Ye 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.

[122] Steven Hand and Timothy Roscoe. Mnemosyne: Peer-to-peer steganographic storage. In *Peer-to-Peer Systems*, pages 130–140. Springer, 2002.

[123] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer Science Department Faculty Publication Series*, page 180, 2007.

[124] Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. Preventing private information inference attacks on social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8):1849–1862, 2013.

[125] Oliver Heckmann, Axel Bock, Andreas Mauthe, and Ralf Steinmetz. The edonkey file-sharing network. *GI Jahrestagung (2)*, 51:224–228, 2004.

[126] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 9, 2007.

[127] Helene A. Hembrooke, Laura A. Granka, Geraldine K. Gay, and Elizabeth 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.

[128] G Hogben. Security issues and recommendations for online social networks. *Position Paper ENISA*, 80211(1), 2007.

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

[130] Damon Horowitz and Sepandar D. Kamvar. The anatomy of a large-scale social search engine. In *WWW '10: Proceedings of the 19th international conference on World wide web*, 2010.

[131] Wei Hu, Yuzhong Qu, and Gong Cheng. Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, 67(1):140–160, 2008.

[132] Jean-Pierre Hubaux, Matthias Grossglauser, Théophile Studer, and Mathias Humbert. Nowhere to hide: Navigating around privacy in online social networks. In *The 18th European*

Symposium on Research in Computer Security (ESORICS), number EPFL-CONF-187143, 2013.

[133] Ryan Huebsch, Joseph M Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the internet with pier. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 321–332. VLDB Endowment, 2003.

[134] Tereza Iofciu, Peter Fankhauser, Fabian Abel, and Kerstin Bischoff. Identifying users across social tagging systems. In *ICWSM*, 2011.

[135] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving p2p data sharing with oneswarm. In *ACM SIGCOMM*, 2010.

[136] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 50, 2007.

[137] Sonia Jahid, Prateek Mittal, and Nikita Borisov. EASiER: Encryption-based access control in social networks with efficient revocation. In *ASIACCS*, 2011.

[138] Sonia Jahid, Shirin Nilizadeh, Prateek Mittal, Nikita Borisov, and Apu Kapadia. DECENT: A decentralized architecture for enforcing privacy in online social networks. In *Proceedings of the 4th IEEE International Workshop on Security and Social Networking (SESOC)*, pages 326–332, March 2012.

[139] Prachi Jain, Paridhi Jain, and Ponnurangam Kumaraguru. Call me maybe: understanding nature and risks of sharing mobile numbers on online social networks. In *Proceedings of the first ACM conference on Online social networks*, pages 101–106. ACM, 2013.

[140] Mohamed Jawad, Patricia Serrano-Alvarado, and Patrick Valduriez. Protecting data privacy in structured p2p networks. In *Data Management in Grid and Peer-to-Peer Systems*, pages 85–98. Springer, 2009.

[141] David S Johnson, Cecilia R Aragon, Lyle A McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, 37(6):865–892, 1989.

[142] H Jones and J H Soltren. Facebook: Threats to privacy. *Project MAC MIT Project on Mathematics and Computing*, 14, 2005.

[143] Mouna Kacimi, Stefano Ortolani, and Bruno 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, 2009.

[144] Apu Kapadia and Nikos Triandopoulos. Halo: High-Assurance Locate for Distributed Hash Tables. In *NDSS*, 2008.

[145] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03. IEEE Computer Society, 2003.

[146] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *SIGOPS Oper. Syst. Rev.*, 41(5):2–7, October 2007.

[147] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[148] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go-mixes providing probabilistic anonymity in an open system. In *Information Hiding*, pages 83–98. Springer, 1998.

[149] Gunnar W Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(Suppl 1):S59, 2009.

[150] Bernhard Klein and Helmut Hlavacs. A socially aware caching mechanism for encounter networks. *Telecommunication Systems*, 2011.

[151] Jon Kleinberg. Anonymized social networks, hidden patterns, and privacy breaches. In *International Workshop and Conference on Network Science (NetSci07)*, May 2007.

[152] Giorgos Kollias, Shahin Mohammadi, and Ananth Grama. Network similarity decomposition (nsd): A fast and scalable approach to network alignment. *IEEE Trans. on Knowl. and Data Eng.*, 24(12):2232–2243, December 2012.

[153] Masaki Kondo, Shoichi Saito, Kiyohisa Ishiguro, Hiroyuki Tanaka, and Hiroshi Matsuo. Bifrost : A novel anonymous communication system with dht. *Parallel and Distributed Computing Applications and Technologies, International Conference on*, 0, 2009.

[154] Mohammed Korayem and David Crandall. De-anonymizing users across heterogeneous social computing platforms. In *AAAI International Conference on Weblogs and Social Media (ICWSM)*, 2013.

[155] Mohammed Korayem and David J. Crandall. De-anonymizing users across heterogeneous social computing platforms. In *ICWSM*, 2013.

[156] Aleksandra Korolova, Rajeev Motwani, Shubha U Nabar, and Ying Xu. Link privacy in social networks. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 289–298. ACM, 2008.

[157] Katharina Krombholz, Dieter Merkl, and Edgar Weippl. Fake identities in social media: a case study on the sustainability of the facebook business model. *Journal of Service Science Research*, 4(2):175–212, 2012.

[158] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM Sigplan Notices*, 35(11):190–201, 2000.

[159] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010.

[160] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[161] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[162] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[163] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[164] Stefan M Larson, Christopher D Snow, Michael Shirts, and Vijay S Pande. Folding@ home and genome@ home: Using distributed computing to tackle previously intractable problems in computational biology. *arXiv preprint arXiv:0901.0866*, 2009.

[165] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the kazaa network. In *Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on*, pages 112–120. IEEE, 2003.

[166] Chris Lesniewski-Laas and M. Frans Kaashoek. Whanau: a Sybil-proof distributed hash table. In *NSDI*, 2010.

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

[168] Kun Liu, Kamalika Das, Tyrone Grandison, and Hillol Kargupta. Privacy-preserving data analysis on graphs and social networks, 2008.

[169] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 93–106. ACM, 2008.

[170] Matthew M. Lucas and Nikita Borisov. flyByNight: Mitigating the privacy risks of social networking. In *WPES*, 2008.

[171] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[172] Delfina Malandrino, Andrea Petta, Vittorio Scarano, Luigi Serra, Raffaele Spinelli, and Balachander Krishnamurthy. Privacy awareness about information leakage: Who knows what about me? In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 279–284. ACM, 2013.

[173] Mehdi Mani, Anh-Minh Nguyen, and Noël Crespi. SCOPE: A prototype for spontaneous P2P social networking. In *PerCom Workshops*, 2010.

[174] Mehdi Mani, Winston Seah, and Noël 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.

[175] Huina Mao, Xin Shuai, and Apu Kapadia. Loose tweets: An analysis of privacy leaks on Twitter. In *Proceedings of The 2011 ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 1–11, October 2011.

[176] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS*, 2002.

[177] Diana Maynard, Kalina Bontcheva, and Dominic Rout. Challenges in developing opinion mining tools for social media. *Proceedings of@ NLP can u tag# user_generated_content*, 2012.

[178] Giuliano Mega, Alberto Montresor, and Gian Pietro Picco. Efficient dissemination in decentralized social networks. In *Peer-to-Peer Computing*, pages 338–347, 2011.

[179] Prateek Mittal, Matthew Caesar, and Nikita Borisov. X-Vine: Secure and pseudonymous routing in DHTs using social networks. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium*, 2012.

[180] Prateek Mittal, Matthew Caesar, and Nikita Borisov. X-vine: Secure and pseudonymous routing using social networks. In *NDSS*, 2012.

[181] Prateek Mittal, Charalampos Papamanthou, and Dawn Song. Preserving link privacy in social network based systems. 2013.

[182] Eytan Modiano, Devavrat Shah, and Gil Zussman. Maximizing throughput in wireless networks via gossiping. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '06/Performance '06. ACM, 2006.

[183] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol — version 3, December 2004.

[184] Anirban Mondal and Masaru Kitsuregawa. Privacy, security and trust in p2p environments: A perspective. In *Database and Expert Systems Applications, 2006. DEXA'06. 17th International Workshop on*, pages 682–686. IEEE, 2006.

[185] Andreas Müller, Nathan Evans, Christian Grothoff, and Samy Kamkar. Autonomous nat traversal. In *10th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P 2010)*, pages 61–64. IEEE, 2010.

[186] Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. Is it really about me?: message content in social awareness streams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 189–192. ACM, 2010.

[187] Arjun Nambiar and Matthew Wright. Salsa: a structured approach to large-scale anonymity. In *ACM CCS*, 2006.

[188] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *IJCNN*, 2011.

[189] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, IEEE Symposium on Security and Privacy, pages 111–125, 2008.

[190] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 173–187. IEEE Computer Society, 2009.

[191] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[192] Shirin Nilizadeh, Naveed Alam, Nathaniel Husted, and Apu Kapadia. Pythia: A privacy aware, peer-to-peer network for social search. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11. ACM, 2011.

[193] Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia. Cachet: A decentralized architecture for privacy preserving social networking with caching. In *Proceedings of The 8th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 337–348, December 2012.

[194] Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *The 21st ACM Conference on Computer and Communications Security (CCS '14), To appear*, November 2014.

[195] Ivan Osipkov, Peng Wang, and Nicholas Hopper. Robust accounting in decentralized P2P storage systems. In *ICDCS*, 2006.

[196] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[197] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymityÑa proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

[198] Facebook hit by phishing attacks for a second day. `http://edition.cnn.com/2009/TECH/04/30/facebook.phishing.attacks/index.html`.

[199] Alex Pothen. Graph partitioning algorithms with applications to scientific computing. In *Parallel Numerical Algorithms*, pages 323–368. Springer, 1997.

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

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

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

[203] Md Sazzadur Rahman, Ting-Kai Huang, Harsha V Madhyastha, and Michalis Faloutsos. Efficient and scalable socware detection in online social networks. In *USENIX Security Symposium*, pages 663–678, 2012.

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

[205] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. Truthy: mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 249–252. ACM, 2011.

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

[207] David Rosenblum. What anyone can know: The privacy risks of social networking sites. *IEEE Security and Privacy*, 2007.

[208] David Rosenblum. What anyone can know: The privacy risks of social networking sites. *Security & Privacy, IEEE*, 5(3):40–49, 2007.

[209] Dario Rossi, Marco Mellia, and Michela Meo. Understanding Skype signaling. *Comput. Netw.*, 53(2):130–140, 2009.

[210] M. Rosvall and C.T. Bergstrom. Mapping change in large networks. *PloS one*, 5(1):e8694, 2010.

[211] Martin Rosvall and Carl T. Bergstrom. Infomap clustering tool. `http://www.tp.umu.se/~rosvall/code.html`.

[212] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[213] Antony Rowstron and Peter Druschel. Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[214] Daniel R. Sandler and Dan 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.

[215] Kirk Schloegel, George Karypis, and Vipin Kumar. *Graph partitioning for high performance scientific simulations*. Army High Performance Computing Research Center, 2000.

[216] Hendrik Schulze and Klaus Mochalski. Internet study 2008/2009, 2009. `http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf`.

[217] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, pages 41–53. Springer, 2003.

[218] Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 685–688. Association for Computational Linguistics, 2010.

[219] Sanur Sharma, Preeti Gupta, and Vishal Bhatnagar. Anonymisation in social network: A literature survey and classification. *International Journal of Social Network Mining*, 1(1):51–66, 2012.

[220] Kapil Singh, Sumeer Bhola, and Wenke Lee. xBook: Redesigning privacy control in social networking platforms. In *USENIX Security Symposium*, 2009.

[221] Rohit Singh, Jinbo Xu, and Bonnie Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Research in computational molecular biology*, pages 16–31. Springer, 2007.

[222] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, 2008.

[223] Skype. `http://www.skype.com/`.

[224] Ralf Steinmetz and Klaus Wehrle. *2. What Is This "Peer-to-Peer" About?* Springer, 2005.

[225] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 73–86. ACM, 2002.

[226] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 149–160. ACM, 2001.

[227] Xiaoxun Sun, Lili Sun, and Hua Wang. Extended k-anonymity models against sensitive attribute disclosure. *Computer Communications*, 34(4):526–535, 2011.

[228] Matt Tatham. Experian marketing services reveals 27 percent of time spent online is on social networking, 2013. `http://press.experian.com/United-States/Press-Release/experian-marketing-services-reveals-27-percent-of-time-spent-online-is-c` `aspx`.

[229] Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: better privacy for social networks. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 169–180. ACM, 2009.

[230] Cristen Torrey, Elizabeth F. Churchill, and David 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.

[231] BK Tripathy and GK Panda. A new approach to manage security against neighborhood attacks in social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 264–269. IEEE, 2010.

[232] Robbert VANRENESSE, Ken Birman, Adrian Bozdog, D DIM-ITRIU, Manpreet Singh, and Werner Vogels. Heterogeneity-aware peer-to-peer multicast. In *Proceedings of the 17th International Symposium on Distributed Computing (DISC2003)*, 2003.

[233] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.

[234] Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, pages 363–374, New York, NY, USA, 2010. ACM.

[235] Marc Waldman, Aviel D Rubin, and Lorrie Faith Cranor. Publius: A robust, tamper-evident censorship-resistant web publishing system. In *9th USENIX Security Symposium*, pages 59–72, 2000.

[236] Chen Wang, Li Xiao, Yunhao Liu, and Pei Zheng. Distributed caching and adaptive search in multilayer P2P networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, ICDCS '04. IEEE Computer Society, 2004.

[237] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. Virality Prediction and Community Structure in Social Networks. *Scientific Reports*, 3, August 2013.

[238] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 223–238. IEEE Computer Society, 2010.

[239] Le-Shin Wu, Ruj Akavipat, and Filippo 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, 2005.

[240] Wei Xu, Fangfang Zhang, and Sencun Zhu. Toward worm detection in online social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 11–20. ACM, 2010.

[241] Zhiqian Xu and Hai Jiang. A framework of decentralized pki key management based on dynamic trust. In *Proceedings of the International Conference on Security & Management*, 2008.

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

[243] Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, volume 8, pages 739–750. SIAM, 2008.

[244] Xiaowei Ying and Xintao Wu. On link privacy in randomizing social networks. *Knowledge and information systems*, 28(3):645–663, 2011.

[245] Honggang Zhang and Sudarshan Vasudevan. Design and analysis of a choking strategy for coalitions in data swarming systems. In *INFOCOM*, 2012.

[246] Jun Zhang and Mark 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.

[247] Jing Zhao, Ping Zhang, and Guohong Cao. On cooperative caching in wireless P2P networks. In *Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems*, ICDCS '08. IEEE Computer Society, 2008.

[248] Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *Privacy, security, and trust in KDD*, pages 153–171. Springer, 2008.

[249] Elena Zheleva and Lise Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540. ACM, 2009.

[250] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 506–515, Washington, DC, USA, 2008. IEEE Computer Society.

[251] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515, 2008.

[252] Bin Zhou and Jian Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems*, 28(1):47–77, 2011.

[253] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations Newsletter*, 10(2):12–22, 2008.

[254] Runfang Zhou and Kai Hwang. Gossip-based reputation aggregation for unstructured peer-to-peer networks. *Parallel and Distributed Processing Symposium, International*, 0, 2007.

[255] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.

[256] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.

Shirin Nilizadeh: Curriculum Vitae

School of Informatics and Computing
Indiana University Bloomington
Homepage: http://private.soic.indiana.edu/people/shirin-nilizadeh/
Email: shirnili@indiana.edu

Education

Ph.D. in Informatics- Security Informatics, September 2014
School of Informatics and Computing, Indiana University at Bloomington
Advisor: Prof. Apu Kapadia
Dissertation: *Privacy-Aware Decentralized Architectures for Socially Networked Systems*
GPA: 3.958/4

M.Sc. in Information Technology- Information Security, February 2007
Computer Engineering and Information Technology, Amirkabir University of Technology
Thesis: *Representing Linear approximations of Block Ciphers Using Graph Modeling*
Advisor: Prof. Babak Sadeghiyan
GPA: 17.10/20

B.Sc. in Computer Engineering- Software Engineering, June 2004
Department of Computer Engineering Islamic Azad University at Najafabad
GPA: 17.64/20

Fellowships and Awards

Awards
Winner of InWIC Award for the Best Graduate Research Poster, Received a full scholarship to attend the 2012 Grace Hopper Celebration of Women in Computing in Baltimore, MD
Runner up for InWIC Award for the Best Graduate Research Poster, 2010
Awarded by Iran Telecommunication Research Center (ITRC) for my M.Sc. Thesis

Travel Grants
Microsoft scholarship to attend Indiana Celebration of Women in Computing (InWIC), 2014
Travel grant to attend International Workshop on Cyber Crime (IWCC), 2013
Travel grant to attend 21st USENIX Security Symposium, 2012
Microsoft scholarship to attend Indiana Celebration of Women in Computing (InWIC), 2012

Travel grant for ACM Conference on Computer and Communications Security (CCS), 2011
Travel Grant to attend Workshop on Privacy in the Electronic Society (WPES), 2011
Travel Grant to attend CRA-W Grad Cohort Workshop, 2010
Travel Grant to attend Indiana Celebration of Women in Computing (InWIC), 2010

Fellowships
NSF-TRUST fellowship for attending Women's Institute in Summer Enrichment (WISE) program at CMU, 2011
Two Year Research Fellowship, School of Informatics and Computing, Indiana University, 2009-2011

Publications

Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn, "Community-Enhanced De-anonymization of Online Social Networks," To appear in the 21st ACM Conference on Computer and Communications Security (CCS ?14), Scottsdale, Arizona, USA, November 3?7, 2014.

Vaibhav Garg and Shirin Nilizadeh, "Craigslist Scams and Community Composition: Investigating Online Fraud Victimization", To be appeared at International Workshop on Cyber Crime (IWCC 2013), San Francisco, May, 2013.

Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia, "Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching," In Proceedings of The 8th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12), Nice, France, Dec, 2012.

Sonia Jahid, Shirin Nilizadeh, Prateek Mittal, Nikita Borisov, and Apu Kapadia, "DECENT: A Decentralized Architecture for Enforcing Privacy in Online Social Networks," In Proceedings of 4th IEEE International Workshop on Security and Social Networking (SESOC '12), Lugano, Switzerland, Mar 19, 2012.

Shirin Nilizadeh, Naveed Alam, Nathaniel Husted, and Apu Kapadia, "Pythia: A Privacy Aware, Peer-to-Peer Network for Social Search", In Proceedings of 2011 ACM Workshop on Privacy in the Electronic Society (WPES '11), pp. 43–48, Chicago, Illinois, October 17, 2011.

Shirin Nilizadeh, Naveed Alam, Nathaniel Husted, and Apu Kapadia. "Pythia: A Privacy Aware, Peer-to-Peer Network for Social Search," Technical Report TR687, Indiana University Bloomington, Oct.2010.
https://www.cs.indiana.edu/cgi-bin/techreports/TRNNN.cgi?trnum=TR687.

Shirin Nilizadeh, and Babak Sadeghiyan, "Linear Approximations Representation of Moamagar Block Cipher", In Proceedings of the 12th International Conference on Computer Science (CSICC), Iran, 2007. (in Farsi)

Academic Activities

Teaching Experience

Guest Lecture, CSCI-P 538: Computer Networks, Fall 2013
Guest Lecture, I430/520, CSCI-B 649: Security For Networked Systems, Spring 2013
School of Informatics and Computing, Indiana University Bloomington

Assistant Instructor, I-433/533: Systems & Protocol Security & Information Assurance, Fall 2012
School of Informatics and Computing, Indiana Univeristy Bloomington

Lecturer, Network Security, Spring 2009
Lecturer, Design and Implementation of Programming Languages, Spring 2009
Lecturer, A Web-based Programming Language (ASP), Spring 2009
Lecturer, Computer Networks, Fall 2008
Lecturer, Data Structures and Algorithms, Fall 2008
Lecturer, A Web-based Programming Language (ASP), Fall 2008
Lecturer, Data Structures and Algorithms, Summer 2008
Department of Computer Engineering, Islamic Azad University at Najafabad

Advising and Mentoring

Graduate Student Mentor for the research experience program for undergraduate CS majors, School of Informatics and Computing, Spring 2012
Advisor and Mentor for 20 undergraduate students, Fall 2008 and Spring 2009, Department of Computer Engineering, Islamic Azad University at Najafabad

Talks at Conferences and Workshops

"Privacy-aware Decentralized Architectures for Socially Networked Systems", 'Student Knowledge Exchange on Technical Aspects of Privacy 2013', University of Notre Dame, September 2013.
"Pythia: A Privacy Aware, Peer-to-Peer Network for Social Search", 2011 ACM Workshop on Privacy in the Electronic Society (WPES '11), Chicago, Illinois, October 2011.
"Linear Approximations Representation of Moamagar Block Cipher", 12th International Conference on Computer Science (CSICC), Shahid Beheshti University (SBU), Tehran, Iran, February 2007.

Posters

"Community-Enhanced De-anonymization of Online Social Networks", InWIC 2014.
"Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching", GHC 2012.
Winner of Best Graduate Poster Award, "DECENT: A Decentralized Architecture for Enforcing

Privacy in Online Social Networks", InWIC 2012.
Runner up for the Best Graduate Poster Award, "Peer-to-Peer Aardvark", Indiana Celebration of Women in Computing (InWIC), 2010.

Technical Skills

Languages: Experienced in Java, Perl, C, Visual Basic, familiar with .Net Framework, R, Pascal and Bash scripting
Web Design: Experienced in HTML, Front page, ASP, ASP.Net, familiar with JavaScript and VBScript
Databases: MS SQL Server, MySQL, SQLite and MS Access
Graph Networks: Network Workbench, and JUNG Library in Java, NetworkX in Python
Machine Learning: Experienced in using Weka, and SVM-light for data mining
Miscellaneous: Experienced in using SVN, LaTeX, Microsoft Office, and UML, familiar with gnuplot

Volunteering

Peer-reviewed and sub-reviewed papers submitted to Human-centric Computing and Information Sciences Journal, PETS 2014, IEEE S&P 2014, NDSS 2013, TRUST 2013, ACSAC 2013, WPES 2013, WPES 2012, ACISP 2012, and CCS 2011.
Student volunteer at PETS 2013, Bloomington, Indiana
Have been facilitating the weekly Security Reading Group, open to the research community in School of Informatics and Computing since Fall 2011.
Acted as a student scribe for 21st USENIX Security Symposium.
Reviewed applications for the NCWIT Award for Indiana Aspirations for Women in Computing 2011-12.
Member of executive committee, The Third Robotic and Artificial Mice Competition and the 4th Iranian student Conference on Electrical and Computer Engineering, Islamic Azad University at Najafabad, Summer 2003.