Private Social Network Analysis: How to Assemble Pieces of a Graph Privately

Keith B. Frikken
Miami University
230 Kreger Hall
Oxford, OH 45056
frikkekb@muohio.edu

Philippe Golle
Palo Alto Research Center
3333 Coyote Hill Rd
Palo Alto, CA, 94304
pgolle@parc.com

ABSTRACT

Connections in distributed systems, such as social networks, online communities or peer-to-peer networks, form complex graphs. These graphs are of interest to scientists in fields as varied as marketing, epidemiology and psychology. However, knowledge of the graph is typically distributed among a large number of subjects, each of whom knows only a small piece of the graph. Efforts to assemble these pieces often fail because of privacy concerns: subjects refuse to share their local knowledge of the graph.

To assuage these privacy concerns, we propose reconstructing the whole graph privately, i.e., in a way that hides the correspondence between the nodes and edges in the graph and the real-life entities and relationships that they represent. We first model the privacy threats posed by the private reconstruction of a distributed graph. Our model takes into account the possibility that malicious nodes may report incorrect information about the graph in order to facilitate later attempts to de-anonymize the reconstructed graph. We then propose protocols to privately assemble the pieces of a graph in ways that mitigate these threats. These protocols severely restrict the ability of adversaries to compromise the privacy of honest subjects.

Categories and Subject Descriptors: E.3 [Data]: Data Encryption; K.4.1 [Computers and Society]: Public Policy Issues—*Privacy*

General Terms: Security

Keywords: Privacy, Anonymity, Social networks, Graph

1. INTRODUCTION

Connections in distributed systems, such as social networks, online communities or peer-to-peer networks, form complex graphs. For example, a computer network consists of routers, hosts and the physical links between them. A peer-to-peer network consists of nodes and overlay connections between these nodes. A social network consists of a population of users with connections defined, for example, by who exchanges email with whom. Regardless of their nature, all networks can be modeled by a graph, whose vertices represent network elements, and whose edges represent connections between these elements.

Distributed network graphs are extremely valuable and tremendous resources have been expanded to study them [6, 22, 25, 26]. The structure of computer network graphs is helpful in understanding the impact of link failures, the problem of congestion, etc. Studies of peer-to-peer network graphs have helped identify formal and informal online communities, leaders, outliers, etc [16, 24]. Finally, the structure of social network graphs is important to fields as varied as marketing [7], epidemiology, sociology [2, 12] and even counter-terrorism [13, 23].

However, knowledge of these graphs is typically distributed among a large number of subjects, each of whom knows only a small piece of the graph. Efforts to assemble these pieces often fail because of privacy concerns: subjects refuse to share their local knowledge of the graph, because that information is sensitive, for commercial, security or privacy reasons [1, 15]. For example, network operators may be reluctant to reveal the network links they own to their competitors. Nodes in a peer-to-peer network or individuals in a social network may refuse to reveal their connections to other nodes or individuals out of privacy concerns. For this reason, network graphs are notoriously difficult to obtain: those with knowledge of the graph (or pieces of it) may refuse to participate in data collection efforts, or they may even submit intentionally incorrect information [6].

This tension between privacy on the one hand and the benefits of learning the graph on the other hand can often be resolved. Indeed, the most valuable information about a graph is often the least privacy sensitive, while conversely the most private information is of little value. In typical graphs of non-trivial size, valuable information comes from the overall, global structure of the graph, whereas privacy-sensitive information, consisting of the local neighborhood around a node and the correspondence between this neighborhood and the real-life entities it represents, is often less valuable. To allay privacy concerns, we propose collecting the graph *privately*, i.e. in a way that does not expose the correspondence between the nodes and edges in the graph and the real-life entities and relationships that these nodes and edges represent.

Example. Suppose researchers would like to study friendship relationships among a group of students. The goal is simply to learn the graph of who is friends with whom. Note that researchers are interested only in "global" features of the graph, such as the number and size of cliques or the number of loners (i.e. friendless or near-friendless students), but not in the particular friendships of a particular student. Some students would almost certainly refuse to report who are their friends if there is a risk that this information can later be attributed to them. But almost all students may volunteer this information if it is collected privately, with guarantees that graph elements cannot be tied to the real-life entities or the relationships that they represent, beyond inferences that are unavoidable (a student who reports three friends knows she is represented in the graph by a node of out-degree three).

Privacy from whom? This paper makes the common assumption that data is collected collaboratively by a number of entities whom we call "authorities". Some authorities may be corrupted by an adversary, but we assume an honest majority of authorities at all times (if authorities are "honest-but-curious", we need only assume a single honest authority at all times). In scenarios where authorities collect data about a subject population (e.g., votes, census data or the like), privacy usually means privacy from the authorities. It is typically assumed that subjects will not compromise their own privacy, or at least that there is no harm in anyone damaging their own privacy (e.g., a citizen revealing her own census information). However, the threat to privacy coming from subjects is more serious in the reconstruction of private graphs. The existence of edges between subjects means that a malicious subject can compromise not just her own privacy, but also the privacy of other subjects which she is connected to. The following example illustrates this new threat, and the difficulties of designing protocols for reconstructing graphs privately.

Pseudonyms: a simple solution and its limitations. Assume in our earlier example that each student chooses a pseudonym and lets her friends know the pseudonym that she has chosen, so that all friendship links can be reported to the authorities pseudonymously. The authorities reconstruct the friendship graph on vertices labelled with pseudonyms, then strip the vertices of their pseudonymous labels and publish the resulting anonymized graph. This approach would seem to offer good privacy, as long as the authorities do not learn the relationship between pseudonyms and real identities. But a closer look reveals a problem: a malicious student, in collusion with one or more malicious authorities, can learn which node in the anonymized graph corresponds to his pseudonym, and thus to him. This attack would not be of much concern if it allowed the malicious student to damage only his own privacy. But the potential breach of privacy extends further: with knowledge of the node that represents him, the malicious student can learn which nodes represent his friends, the friends of his friends, etc. For example, the malicious student can learn how many friends each of his friends has. Thus, privacy protection for private graph reconstruction must consider threats to privacy coming both from the *authorities* that collect the data as well as from the *subjects* themselves.

Overview. This paper proposes a model of privacy for reconstructing a graph privately (Section 2). This threat model accounts not only for malicious authorities, but also for malicious subjects and coalitions of malicious subjects and authorities. In Section 3, we show that there is a fundamental trade-off between the privacy and accuracy of the graph reconstructed: exact private reconstruction of a graph is not always possible. In Section 4, we introduce cryptographic building blocks used in the rest of the paper. In Section 5, we propose several protocols to recover the graph in a private manner. Finally, we summarize our results in Section 6.

1.1 Related Work

The importance of preserving the privacy of subjects is well recognized in the literature on social network analysis (see for example [6]). A well-entrenched and near-universal approach to protecting privacy is to assign pseudonyms to individuals and collect and store data pseudonymously [20]. In a famous study of email relationships within an organization [24], individuals were "randomly assigned an identification number". Pseudonyms offer adequate privacy protection as long as all the authorities collecting data are trusted not to collude with any of the subjects submitting data. But as explained in the introduction, collusion between a single authority and a single subject has the potential to undermine the privacy of a large number of honest subjects. This paper proposes protocols that offer much stronger guarantees of privacy.

The problem of reconstructing a graph privately is an instance of a secure multiparty computation. In theory, generic secure multiparty computation techniques [27, 8] could allow the subjects to compute privately the graph that represents the relationships between them. These generic solutions would, however, incur impossibly large communication and computation costs. More efficient privacy-preserving protocols are needed to solve specific problems.

Recent work [3] allows two parties, each in possession of a graph, to compute some algorithms (e.g., the shortest distance between two vertices) on their joint graph in a privacy-preserving manner. However [3] does not consider the problem of assembling a joint graph, nor does it immediately generalize to multiple parties owning multiple pieces of a graph.

2. MODEL

We consider a set S of subjects, denoted $S = \{S_1, \ldots, S_n\}$, where each subject has a set of relationships with other subjects. Each S_i has a set R_i which contains all subjects to which S_i has a relationship. These R-sets imply a graph $G = (V, E)$ where $V = \{S_1, \ldots, S_n\}$ and $(S_i, S_j) \in E$ if and only if $S_j \in R_i$. When we refer to the node representing a subject in the graph, we call it "node S_i", and when we refer to the corresponding entity we call it "subject S_i". We assume that every subject S_i knows a subset E_i of the edges. The set E_i includes at least all the edges whose origin is S_i (it may also include additional edges, since an individual may be aware of a relationship between two other individuals). We also assume that $|E_i| << |E|$ for all i. In other words, no subject knows more than a small fraction of the edges in E. This is a reasonable assumption in our setting: there is no way to reconstruct privately a graph that is already known.

Assumptions about the graph. Depending on the application considered, the graph G may be directed or undirected. We assume that the graph is directed, but it is easy to modify our protocols to the case where the graph is undirected.

We offer here justifications for some of our main assumptions:

- **Distributed graph.** We assume that the graph must be reconstructed from partial knowledge of edges distributed among the nodes. But in some settings there is a trusted authority (e.g., a cell-phone company or the mail server of a corporation) with knowledge of the complete graph (of calls or emails exchanged), who could effortlessly publish an anonymized version of the graph. Even in those settings, however, the trusted authority may not be willing or allowed to release even sanitized versions of the data it owns. Our model is thus useful not only when the graph data exists only in distributed form, but also when access to centralized data is restricted or forbidden.
- **Importance of reconstructing the graph.** If the goal is to learn properties of the graph G that are locally computable by subjects (such as the average degree of a node), then a simpler solution would be for each subject to report (anonymously) only those properties without revealing anything else about the structure of the graph G. This ensures excellent privacy and can be implemented with a simple mix network [4, 17]. In what follows, we assume that this simple approach does not work, either because the property of interest is not locally computable (e.g. computing the diameter of G, or finding its centers, cliques, etc.), or because the property of interest is not yet known at the time that G is constructed.

Our protocols assume unlabelled nodes, but it is easy to modify them to support node labelling. Labels in a privately reconstructed graph should not be identifying and must thus belong to a set that is very small compared to the total number of nodes. For example, a node label might identify a subject as "male" or "female".

Goal. Ideally, our goal is to learn the graph G privately, i.e., without exposing the identity of the nodes in the graph. More precisely, the ideal goal is to learn a graph AG (i.e., anonymized graph) that is isomorphic to G without revealing the isomorphism that relates AG to G. We wish to do so without relying on a single trusted party. We assume therefore that the isomorphic graph AG is constructed by a number of authorities, none of whom (below a quorum) should be able to learn the isomorphism.

Unfortunately, we show in Section 3 that this ideal goal is unattainable in many environments. Our protocols achieve weaker goals:

- Our protocols output a graph AG that is not strictly isomorphic to G, but rather is isomorphic to a "perturbed" version of G. The perturbations consist of the addition and/or deletion of a small number of edges and/or nodes.
- Our protocols allow the adversary to learn some information about the isomorphism that maps G to AG, but ensure that the adversary learns only minimal information about the isomorphism restricted to honest nodes in G. In particular, our protocols leak *much less* information about honest nodes than the simple approach based on pseudonyms described in the introduction.

Adversarial model. We assume that a strict majority of authorities are honest (if authorities are honest-but-curious, then this assumption can be weakened to the existence of a single honest authority). We assume that the authorities communicate with one another and with subjects via a *bulletin board*, i.e., a shared piece of memory with authenticated, appendive write access. The bulletin board abstraction can be implemented with standard distributed cryptographic protocols. We assume that some subjects may be malicious, and that malicious subjects may collude with one another as well as with corrupt authorities.

3. TRADE-OFF BETWEEN PRIVACY AND ACCURACY

In this section, we show that there is a fundamental trade-off between the privacy and accuracy of the reconstructed graph: exact private reconstruction of a graph is not always possible. This trade-off is a fundamental consequence of our adversarial assumptions and applies not only to the protocols proposed in this paper, but to any protocol that assumes the same threat model.

Our threat model allows the adversary to control a subset of the subjects. If these subjects report particular patterns of relationships that "stand out" (i.e., footprints) in the graph AG, the adversary can recognize these footprints and learn part of the isomorphism that maps G to AG. The adversary can do so with knowledge only of the output of the protocol (the graph AG) and of the inputs of subjects under its control (pieces of the graph G). Note that this attack does not necessarily require the subjects controlled by the adversary to lie about their relationships: their true pattern of relationships may be easily distinguishable from the patterns of relationships of honest subjects. (If not, lying is always an option!)

Example. There are innumerable patterns that an adversary may recognize in AG. A simple example would be a vertex with an unusually large in- or out-degree. Suppose that the adversary controls m subjects and instructs all of them to report a link to a targeted honest subject (whether or not this link exists in reality). If m is large enough, the anonymized graph AG will have only one vertex with m or more incoming edges, and the adversary knows that this vertex represents the targeted honest subject. Not only has the adversary partially broken the anonymity of AG, it has broken the anonymity of a honest subject outside of its control.

Counter-measures. We propose several types of counter-measures that are complementary and can be used in combination. These counter-measures, described and analyzed in more detail in section 5, dramatically reduce the ability of an adversary to undermine the privacy of honest subjects.

1. **Subject-authority collusion resilience:** In section 5.1, we introduce a protocol that is more resilient against adversaries that corrupt both subjects and authorities than the nym-based protocol described earlier. This

first protocol will serve as a starting point for our other protocols. The crucial insight is that the nyms are revealed neither to the subjects nor to the authorities.

2. **Mitigating footprint attacks:** To mitigate adversaries that introduce patterns into the graph, we introduce several protocols that prevent certain footprints. These patterns include: disallowing edge values (section 5.2), disallowing multi-edges (section 5.3), disallowing self-edges (section 5.4), bounding the outdegree of a node (section 5.5), and bounding the indegree of a node (section 5.6). Of course, which set of these countermeasures is used depends on the environment and the purpose of the collector. For example, multi-edges should not be disallowed if they are crucial to the purpose of the data collector. For these reasons all of the above counter-measures are optional, but we recommend the usage of as many of these techniques as possible to prevent de-anonymization.

3. **Consent:** In some environments a subject will know all out-going edges as well as in-coming edges. In this case, a powerful counter-measure is to require that the subject agree to every in-coming edge to its node in the graph. We introduce a protocol that supports privately reporting consent in section 5.7.

4. **Noisy graph reconstruction:** In the process of reconstructing a private graph, the authorities add to the graph a number of random edges, or delete from the graph some nodes or edges chosen at random (see section 5.8). The protocol ensures that edges and nodes added or deleted are indistinguishable from other nodes and edges. There is a trade-off between the quality and privacy of the reconstructed graph: more noise lowers the similarity between G and AG but makes it harder to learn the mapping from G to AG.

4. CRYPTOGRAPHIC BUILDING BLOCKS

Our protocols use cryptographic building blocks which we now describe briefly. These building blocks are all standard distributed cryptographic algorithms run jointly by a quorum of authorities.

Threshold ElGamal encryption. ElGamal is a randomized public-key encryption scheme that allows for semantically secure re-encryption of ciphertexts. The ElGamal encryption scheme allows for threshold key generation and encryption [19]. In what follows, all ciphertexts are encrypted with a threshold version of ElGamal. The authorities hold shares of the corresponding decryption key, such that a quorum consisting of all parties can decrypt.

Robust re-encryption mix network. A re-encryption mix network [17] re-encrypts and permutes a number of input ElGamal ciphertexts. In our application, the authorities play the role of mix servers. If we allow active adversaries, we must use robust re-encryption mixnets such as [10] or [14]. When we say the authorities "mix" a set of inputs according to a permutation π, we mean that they run the set of inputs through a mix network and we let π denote the global (secret) permutation that matches the outputs to the inputs. This permutation π is not known to the authorities, unless they all collude (which our adversarial model does not allow). Note also that for notational simplicity, we define π as the permutation that matches the *outputs* of the mix network to its *inputs*. Thus, input $\pi(i)$ is matched to output i.

Mixing ciphertext tuples. A regular mixnet takes as input a list of single ElGamal ciphertexts. In this paper, we will also consider mixing operations where every input is a vector of ElGamal ciphertexts (pairs, triplets, etc.) All vectors mixed within a batch will always have the same number of elements. Mix servers must re-encrypt individually every ciphertext component of the vector, then mix the vectors. Note that the vectors are mixed, but the order of the ciphertexts *within* any given vector is fixed. When active adversaries are allowed, techniques such as those described in [9] can be used to ensure the correctness of tuple mixing (essentially, a checksum field is added to vectors to verify that the order of the vector elements is not modified). The computational cost of mixing tuples is proportional to the number of elements in the vector.

Oblivious test of plaintext equality. Let $E(m_1)$ and $E(m_2)$ be two ElGamal ciphertexts. A protocol of Jakobsson and Schnorr [11] lets the joint holders of the decryption key determine whether $m_1 = m_2$ without revealing any other information (hence the name oblivious). Using ElGamal's multiplicative homomorphism, the problem is equivalent to testing whether an ElGamal ciphertext (a, b) is an encryption of 1, i.e. whether $\log_g(y) = \log_a(b)$. This protocol takes as input $E(m_1), E(m_2)$ and outputs either $m_1 = m_2$ or $m_1 \neq m_2$.

Proof of plaintext knowledge. Let $E(m) = (g^a, my^a)$ be an encryption generated by Alice. She can prove that she knows the plaintext m by proving knowledge of $\log_g(g^a)$. This can be done with Schnorr's protocol [21], which can be made non-interactive with the Fiat-Shamir heuristic [5].

5. PROTOCOLS

In this section we introduce protocols for constructing the anonymized graph. The protocol of section 5.1 is a preliminary solution and future sections extend this protocol.

5.1 Base Protocol

In this section we introduce a protocol that improves upon the simple nym-based solution that was described in the introduction. In essence, our new protocol allows subjects to report relationships (edges in the graph) pseudonymously, while hiding from both subjects and authorities the mapping between subjects and pseudonyms. The details of the protocol are as follows.

Setup. The authorities create a list of n ciphertexts $E(1), \ldots, E(n)$. They mix this list to obtain a permuted list:

$$E(\pi(1)), \ldots, E(\pi(n))$$

They then append n copies of the value $E(-1)$ to the end of this list to form a list L of $2n$ ciphertext elements. The authorities then post L to the bulletin board. Intuitively, the value $\pi(i)$ is the pseudonym associated with subject S_i. The n additional ciphertexts $E(-1)$ are dummies used to hide the number of edges that a subject reports.

Data Collection. Subject S_i does the following steps:

1. S_i chooses a random permutation σ_i (over $2n$ items) such that for $1 \leq j \leq n$:
   - If $S_j \in R_i$, then $\sigma_i^{-1}(j) \in [1, n]$.
   - If $S_j \notin R_i$, then $\sigma_i^{-1}(j) \in [n+1, 2n]$

   Note that such a permutation always exists. The permutation σ_i maps to the range $[1, n]$ the encrypted nyms of all subjects to whom S_i wants to report a relationship. When S_i has fewer than n relationships, σ_i also maps "dummy" entries (i.e., encryptions of -1) to the range $[1, n]$. The encrypted nyms of subjects to whom S_i does not want to report a relationship are mapped to the range $[n+1, 2n]$, as are left-over dummy entries (if any).

2. S_i mixes L with σ_i and submits this to the authorities along with a proof of proper mixing. We denote the list submitted by S_i as $L_i = L[\sigma_i(1)], \ldots, L[\sigma_i(2n)]$.

Graph Reconstruction. To reconstruct the graph the authorities create a set T and initialize it to the empty set. They then do the following:

1. For each subject S_i:
   (a) The authorities verify the proof of correct mixing. If the proof is invalid, the authorities discard L_i and continue without S_i's values.
   (b) They create n tuples: $\Big(E(\pi(i))\ ;\ L[\sigma_i(j)]\Big)$ for $1 \leq j \leq n$, and they add all of these tuples to T.

2. The authorities mix the list of tuples in T and prove correct mixing to obtain a new list of tuples T'.

3. For each tuple $\Big(E(x)\ ;\ E(y)\Big)$ in T', the authorities jointly decrypt $E(y)$. If the value is not in $[1, n]$, they discard the tuple. Otherwise, they jointly decrypt $E(x)$ and record an edge from x to y in AG.

Security Properties.

1. The number of edges that a subject reports is hidden from an adversary (even if the adversary corrupts some authorities).

2. Subjects do not learn their own nym or other participants' nyms as part of the setup phase. This is significantly different from the nym-based approach described earlier. Thus, if the adversary corrupts some authorities, it will not be able to unlink the nyms even with the help of corrupted subjects (unless unlinking can be done from knowledge of the adversary's input and AG alone).

3. A subject can report only edges from itself to other subjects, but not between other subjects, because the authorities prepend the reporting subject's nym to the tuples. This differs from the nym-based scheme where an adversary could report edges between any two subjects whose nyms the adversary has learned during the data collection phase (e.g., subjects who reported an edge to one of the corrupted subjects controlled by the adversary).

4. Multi-edges are disallowed in this protocol; during data-collection the subjects report a permuted list, and each subject can appear at most once in this list.

5. The protocol does not require any form of consent to report an edge to a subject. In this regard, the nym-based protocol is stronger since an edge to a subject can only be reported with knowledge of the pseudonym assigned to that subject. We address consent-issues in Section 5.7.

5.2 Edge Values

In some cases, the edges may be labelled with values. If these values are not necessary to perform the functions of the data collector, then they should be omitted (as they clearly make de-anonymization easier). But if the values are necessary for the data collector, then the protocol needs to allow subjects to report edge values. To reduce the effectiveness of de-anonymization attacks, edge values should be restricted to a particular range. Next we propose a protocol that allows the reporting of edge values. We state only the differences between this protocol and that of Section 5.1.

Setup. No differences

Data Collection. Subject S_i creates n values, $v_{i,1}, \ldots, v_{i,n}$ where $v_{i,j}$ is the edge value associated with subject $\sigma_i(j)$) (note that if the jth value is a dummy entry (i.e., if it is E(-1)), then $v_{i,j}$ can be arbitrarily chosen). Subject S_i will then submit L_i, $E(v_{i,1}), \ldots, E(v_{i,n})$, and proofs of plaintext knowledge for the values $v_{i,j}$.

Graph Reconstruction. In Step 1(a), the authorities check the proofs of plaintext knowledge and discard any input not accompanied by a valid proof. In Step 1(b), the authorities build triples $\Big(E(\pi(i))\,;\,E(\pi(j))\,;\,E(v_{i,j})\Big)$.

In Step 3, when the authorities are processing a triple $\Big(E(x)\,;\,E(y)\,;\,E(z)\Big)$, they first decrypt z. If z is not a valid edge value, then the authorities discard the entire triple. Otherwise, they proceed as before, and if all values are valid, they add an edge from x to y with edge value z in AG.

5.3 Multi-edges

The preliminary solution from Section 5.1 disallows multi-edges. In graphs where multi-edges do not occur naturally or are not needed by the data collector, this is preferred because it makes de-anonymization more difficult. However, multi-edges may be necessary, in some environments, to achieve the data collector's purpose, and thus we give a protocol that allows multi-edges.

There are two cases to consider: i) the edges do not have edge values and ii) the edges have edge values. The first case is solved by the protocol in Section 5.2, because the multiplicity of an edge can be represented by an edge value. However, the second case is more difficult since each edge is already labelled with an edge value. The main idea of this protocol is that a subject will report several edge values, with one of these values representing the multiplicity of the edge and the rest of the values being the individual edge values. To prevent leaking the multiplicity of an edge in the data collection phase, we place an upper bound m on the multiplicity of edges (subjects pad their inputs with dummy values to reach this upper bound). Another advantage of placing a bound on the multiplicity is that it mitigates de-anonymization attacks. As the protocols is very similar to that of Section 5.1, we state only the differences.

Setup. No differences.

Data Collection. In addition to the previously submitted information, n sets of $m+1$ encrypted values are submitted. Subject S_i constructs its jth set of values as follows:

1. Suppose $\sigma_i(j) = k$. If $k > n$ then the $m+1$ values are encryptions of randomly chosen values. If $k < n$, and S_i wants to report $\bar{m} \leq m$ edges with S_k, with edge values $v_1, \ldots, v_{\bar{m}}$, then the jth set consists of the values $E(\bar{m}), E(v_1), \ldots, E(v_{\bar{m}})$ followed by $m-\bar{m}$ encryptions of random values.
2. Along with the list of $m+1$ values, S_i submits a proof of plaintext knowledge for each of these values.

Graph Reconstruction. The authorities do the following:

1. The authorities verify all proofs provided by subject S_i. If any proof fails then S_i's tuples are discarded.
2. For every valid tuple $\Big(E(y)\ ;\ E(v_0)\ ;\ \ldots\ ;\ E(v_m)\Big)$ that subject S_i reports, the authorities build a tuple $\Big(E(\pi(i))\ ;\ E(y)\ ;\ E(v_0)\ ;\ \ldots\ ;\ E(v_m)\Big)$
3. The authorities mix all tuples and verify correct mixing.
4. For each tuple $\Big(E(x)\ ;\ E(y)\ ;\ E(v_0)\ ;\ \ldots\ ;\ E(v_m)\Big)$ the authorities do the following:
   (a) They decrypt v_0. If it is not in $[1, m]$, they discard the tuple.
   (b) They decrypt the values $v_1, \ldots, v_{v_0}$. If any of these values is not a valid edge value then they discard this tuple.
   (c) They decrypt y. If $y = -1$, they discard the tuple.
   (d) They decrypt x and add v_0 edges from x to y in AG with respective values $v_1, \ldots, v_{v_0}$.

5.4 Self-edges

The previous protocols allow self-edges (i.e., a subject can report an edge to himself). In some environments this does not naturally occur, and in other environments self-edges may not be necessary to the data collector's goals. Therefore, a protocol that disallows self-edges is necessary.

We will state only the differences between this protocol and the one given in Section 5.1.

Setup. No differences.

Data Collection. The subjects do the same steps as before, but subject S_i chooses the permutation σ_i such that $\sigma_i[2n] = i$.

Graph Reconstruction. Before Step 1(a), the authorities perform an oblivious test for plaintext equality with inputs $L_i[2n]$ and $E(\pi(i))$. If this test fails, then subject S_i's values are discarded. The protocol then continues as before.

Security Properties. These are the same as in Section 5.1 except that self-edges are disallowed.

5.5 Bounding the Out-degree

The previous protocols allow a subject to report up to n relationships (i.e., up to a relationship with every single subject). By reporting a large number of relationships an adversary can introduce footprints into the graph that can later be used to de-anonymize portions of the graph. One technique for mitigating this type of attack is to bound the number of relationships that each subject can report. This technique is not acceptable in all environments, but when it is acceptable to the data collector, bounding the out-degree is preferable from a privacy perspective. In this section we describe changes to the base protocol that prevent subjects from reporting more than $\lambda(\leq n)$ relationships. We state only the differences from the protocol in Section 5.1.

Setup. The authorities append only λ (instead of n) encryptions of -1.

Data Collection. Step 1 of the protocol changes to

1. S_i chooses a permutation σ_i (over $n + \lambda$ items) such that:
   - If $S_j \in R_i$, then $index_{\sigma_i^{-1}}(j) \in [1, \lambda]$.
   - If $S_j \notin R_i$, then $index_{\sigma_i^{-1}}(j) \in [\lambda + 1, n + \lambda]$

Graph Reconstruction. In Step 1(b), the authorities use only the first λ items to build the tuples.

Security Properties. These are the same as in Section 5.1 except that each subject can report at most λ edges.

5.6 Bounding the In-degree

Another defense against de-anonymization is to bound the in-degree of a node. This is more difficult than bounding the out-degree because the edges for the in-degree come from several different nodes (and thus there is not a single choke point). This extension is not always usable, but there are environments where bounding the in-degree is useful. The following is a protocol that bounds the in-degree to λ edges.

Setup. The authorities set up a quorum ElGamal encryption scheme, denoted E. The authorities create a list of n ciphertexts $E(1), \ldots, E(n)$. They mix this list and produce a permuted list $E(\pi(1)), \ldots, E(\pi(n))$. The authorities then mix the list $E(1), \ldots, E(n)$ a second time and produce another permuted list $E(\tau(1)), \ldots, E(\tau(n))$. They then combine the two permuted lists to make a list of pairs

$$\Big(E(\pi(1))\ ;\ E(\tau(1))\Big), \ldots, \Big(E(\pi(n))\ ;\ E(\tau(n))\Big).$$

Finally, they append n pairs of the form $\Big(E(-1)\ ;\ E(-1)\Big)$ and output to the bulletin board this list of $2n$ pairs, we call this list L.

Data submission. This is the same as the base protocol (in Section 5.1) except that when the subjects mix the list they mix the list of ordered pairs (and thus have to prove proper mixing of tuples).

Graph Reconstruction. To reconstruct the graph the authorities create sets T_1 and T_2 and initialize them to $\emptyset$. They then do the following:

1. For each subject S_i:

   (a) The authorities verify the proof of correct mixing. If the proof is invalid, the authorities discard L_i and continue without S_i's values.

   (b) For each value $L_i[j] = \Big(E(y)\ ;\ E(z)\Big)$ where $1 \leq j \leq n$, the authorities create n tuples of the form $\Big(E(\pi(i))\ ;\ E(y)\ ;\ E(z)\Big)$ for $1 \leq j \leq n$, and they add all of these tuples to T_1.

2. The authorities mix the list of tuples in T_1 and prove correct mixing to obtain a new list of tuples T_1'.

3. For each triple $\Big(E(x)\ ;\ E(y)\ ;\ E(z)\Big)$ in T_1', the authorities decrypt z. They then sort the triples based upon the z values (after discarding all values where $z = -1$). If there are more than λ values with the same z values, then the extra values are discarded so that only λ remain. The authorities then build ordered pairs of the form $\Big(E(x)\ ;\ E(y)\Big)$, from the remaining values and put them in the list T_2.

4. The authorities mix the list of tuples in T_2 and prove correct mixing to obtain a new list of tuples T_2'.

5. For each tuple $\Big(E(x)\ ;\ E(y)\Big)$ in T_2', the authorities jointly decrypt $E(y)$. If the value is not in $[1, n]$, then they discard the tuple. Otherwise, they jointly decrypt $E(x)$ and add an edge from x to y in AG.

Security Properties. These are the same as the base protocol except that each subject has at most λ edges pointing to it in AG.

5.7 Consent

In this section, we propose a protocol that protects against the malicious insertion of information by requiring that both parties (the source and the destination) consent to an edge in the graph. Recall that the nym-based approach required consent, but the notion we present in this section is stronger for two reasons:

- The consent is deniable: a subject can agree to an edge with another subject, but can later change his mind and prevent the relationship from being reported after all.

- The consent is non-transferable: if a subject S_i gives consent to a subject S_j, then S_j cannot transfer this consent to another subject S_k.

Consent is achieved with a relatively simple idea: the subjects agree upon a common random value and then both of them report the value along with the edge information. Furthermore, the edge is added only if two parties report the same information.

To model the information that each subject has in this protocol we modify the notion of R_i (the relationships that S_i wants to report). In this case we use two sets $R_{i,out}$ (the relationships that S_i wants to report to others) and $R_{i,in}$ (the relationships that S_i wants to report from others). If edges are symmetric then this can be modeled as a single set.

Setup. Same as in Section 5.1.

Subject Setup. Suppose subjects S_i and S_j would like to report an edge from S_i to S_j. They agree on a random value $r_{i,j}$ chosen uniformly from a range large enough that collisions between values generated by distinct pairs of nodes are highly unlikely. For example, $r_{i,j}$ could be a 160-bit integer.

Data Collection. Subject S_i does the following steps:

1. It creates permutations $\sigma_{i,out}$ and $\sigma_{i,in}$ to represent the sets $R_{i,out}$ and $R_{i,in}$ respectively using Step 1 of the protocol in section 5.1. It permutes L using $\sigma_{i,out}$ and $\sigma_{i,in}$ to obtain $L_{i,out}$ and $L_{i,in}$ respectively. It also generates proofs of proper mixing.

2. It generates n encrypted values $E(q_{i,1,out}), \ldots, E(q_{i,n,out})$ where $q_{i,j,out} = r_{i,k}$ (from the subject setup phase) if $\sigma_{i,out}(j) = k$ and $1 \leq k \leq n$ and is a random value otherwise. It submits these values to the authorities along with proofs of plaintext knowledge.

3. It generates n encrypted values $E(q_{i,1,in}), \ldots, E(q_{i,n,in})$ where $q_{i,j,in} = r_{k,i}$ (from the subject setup phase) if $\sigma_{i,in}(j) = k$ and $1 \leq k \leq n$ and is a random value otherwise. It submits these values to the authorities along with proofs of plaintext knowledge.

Graph Reconstruction. To reconstruct the graph, the authorities create two sets T_1 and T_2 and initialize them to $\emptyset$. They then do the following:

1. For each subject S_i:

   (a) The authorities check all proofs (of proper mixing and of plaintext knowledge). If any of the proofs fail for a subject S_i then all of S_i's values are discarded.

   (b) For each item in $L_{i,out}[\ell]$ $(1 \leq \ell \leq n)$, the authorities build a triple $\Big(E(q_{i,\ell,out})\ ;\ E(\pi(i))\ ;\ L_{i,out}[\ell]\Big)$ and add this triple to T_1.

   (c) For each item $L_{i,in}[\ell]$ $(1 \leq \ell \leq n)$, the authorities build a triple $\Big(E(q_{i,\ell,in})\ ;\ L_{i,in}[\ell]\ ;\ E(\pi(i))\Big)$ and add this triple to T_1.

2. They mix the tuples in T_1, and verify correct mixing (call the new list of tuples T_1').

3. They find all tuples with matching r values (i.e., tuples that were reported by two parties), and combine the tuples' information. The authorities decrypt the first element of every tuple and find all matching values. The authorities discard all tuples that do not match any other tuple or that match more than one tuple in the first element.

   The authorities are left with pairs of tuples that match on the first element:

   $$\Big(r_{m,n}\ ;\ E(\pi(m_1))\ ;\ E(\pi(n_1))\Big) \text{and}$$

   $$\Big(r_{n,m}\ ;\ E(\pi(m_2))\ ;\ E(\pi(n_2))\Big),$$

   where $r_{m,n} = r_{n,m}$. They then check that $E(\pi(m_1))$ and $E(\pi(m_2))$ decrypt to the same plaintext, using the oblivious test of plaintext equality. Similarly, the authorities check that $E(\pi(n_1))$ and $E(\pi(n_2))$ decrypt to the same plaintext. If either of these checks fail then both tuples are discarded. If both checks succeed, then the authorities build an ordered pair $\Big(E(\pi(m_1))\ ;\ E(\pi(n_1))\Big)$ and adds it the the set T_2.

4. They mix the tuples in T_2, and verify correct mixing (call the new list of tuples T_2').

5. A quorum of authorities then decrypt the pairs in T_2' of the form $\Big(E(x)\ ;\ E(y)\Big)$ and register an edge from x to y in AG.

Security properties. In the above protocol an edge will only be added from S_i to S_j (in the anonymized graph) if both subjects agree to it. The consent to add an edge to or from a subject is non-transferable (unlike the nym-based consent presented earlier). This is a strong defense against de-anonymization, because the footprint of an honest subject in the graph is a subset of what that subject reports.

5.8 Adding Noise

In this section, we propose a protocol that allows the authorities to add "noise" to the reconstructed private graph. Specifically, the authorities can add random edges to the graph or delete from the graph random nodes or edges. The protocol ensures that added or deleted edges and nodes are indistinguishable from other nodes and edges. There is a trade-off between the quality and privacy of the reconstructed graph: more noise lowers the similarity between G and AG but makes it harder for an adversary to learn the mapping from G to AG. The appropriate level of noise will depend on the application and the estimated privacy threat. We present this protocol in a self-contained manner, but all of the previous protocols can be modified to use the techniques of this protocol to add noise to the graph.

Let n denote the number of subjects, and let $\bar{n}$ be the number of nodes to be deleted. Let $\bar{e}$ denote the number of true edges to be deleted, and e' the number of fake edges to be added to the graph (in practice, the values $\bar{e}$ and e' may depend on the expected value of the number e of edges submitted).

Setup. The authorities set up a quorum ElGamal encryption scheme, denoted E, and publish the public parameters of E. The authorities create a list $L = (c_1, \ldots, c_n)$ of n ciphertexts where

- For $i = 1, \ldots, n - \bar{n}$, we define $c_i = E(i)$;
- For $i = n - \bar{n} + 1, \ldots, n$, we define $c_i = E(-1)$.

The authorities mix L and output to the bulletin board the permuted list $\pi(L) = c_{\pi(1)}, \ldots, c_{\pi(n)}$.

Data collection. To report an edge between subjects S_i and S_j, a subject (who need not necessarily be either S_i or S_j) mixes $\pi(L)$ according to a random permutation σ, proves correct mixing, then submits to the authorities the pair $(c'_{\pi(i)}, c'_{\pi(j)})$, where $c'_{\pi(i)}$ is the unique element in $\sigma(\pi(L))$ that is a re-encryption of $c_{\pi(i)}$ and similarly $c'_{\pi(j)}$ is the unique element in $\sigma(\pi(L))$ that is a re-encryption of $c_{\pi(j)}$.

Addition of noise. Let P denote the set of all pairs of ciphertexts submitted by subjects. The authorities perform the following steps to add noise to the graph:

1. **Deleting edges.** The authorities mix the set P, verify correct mixing, then delete the last $\bar{e}$ elements of the permuted set.

2. **Deleting nodes.** The authorities process all the pairs left in P one by one as follows. Let (E_1, E_2) be a pair of ciphertexts in P. The authorities mix (E_1, E_2) to obtain a new pair (E_1', E_2') and verify correct mixing. The authorities use the oblivious test of plaintext equality to check whether E_1' is an encryption of the value (-1). If it is, the pair (E_1, E_2) is dropped from P. Otherwise, the authorities use the oblivious test of plaintext equality to check whether E_2' is an encryption of the value (-1). If it is, the pair is dropped. Otherwise it stays in P.

3. **Adding an edge.** The authorities create a list $A = (a_1, \ldots, a_{n-\bar{n}})$ of $n - \bar{n}$ elements, where $a_i = E(i)$. They mix this list according to a secret permutation (let's call it σ), verify correct mixing, and add to set P the pair of ciphertexts $(a_{\sigma(1)}, a_{\sigma(2)})$.

4. **Final round of mixing.** Step 3 can be repeated to add as many edges as wanted. After that, the augmented set P is mixed one last time to ensure that added edges are indistinguishable from true edges.

The order of the operations in the protocol above has been optimized for efficiency. The protocol minimizes the leakage of information: an adversary learns only the number of edges and nodes removed and the number of edges added (i.e., the smallest possible leakage of information). As noted earlier, this protocol is most useful in combination with the protocols of the previous sections to provide additional privacy protection against de-anonymization attacks.

5.9 Composition of Protocols

In the previous sections we have introduced various protocols for collecting the graph anonymously. Other than the base protocol, each protocol is an extension that either adds one layer of security (e.g., bounding the out-degree) or allows something else to be reported (e.g., edge values). The protocol chosen to reconstruct a graph privately will depend upon the environment and the purpose of the data collector. It is generally best to utilize as many of the defenses as possible. This requires composing some (or all) of the protocols introduced in this section.

6. CONCLUSION

The complex graphs that represent connections in distributed systems, such as social networks, online communities or peer-to-peer networks are of tremendous interest and value to scientists in fields as varied as marketing, epidemiology and psychology. However, knowledge of these graphs is typically distributed among a large number of subjects, each of whom knows only a small piece of the graph. Privacy concerns make these subjects reluctant to share their local knowledge of the graph.

This paper studies the problem of assembling pieces of a graph privately. We define what it means to reconstruct a graph privately, propose a threat model and cryptographic protocols that allow a group of authorities to jointly reconstruct a private graph.

In future work, we hope to implement these protocols and demonstrate their value with simulations. We hope these protocols will help collect data that privacy concerns previously made difficult or impossible to collect.

7. REFERENCES

[1] J. Black. The Perils and Promise of Online Schmoozing. Business Week Online, February 20, 2004. `http://yahoo.businessweek.com/technology/content/feb2004/tc20040220_3260_tc073.htm`

[2] J. Boissevain. Friends of friends: Networks, manipulators, and coalitions. 1974. Oxford.

[3] J. Brickell and V. Shmatikov. Privacy preserving graph algorithms in the semi-honest model. In *Proc. of Asiacrypt '05*. To appear.

[4] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. In *Communications of the ACM*, 24(2):84-88, 1981.

[5] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of CRYPTO'86*. LNCS 263, pp. 186–194.

[6] Garton, L., Haythornthwaite, C., and Wellman, B. (1997). Studying online social networks. Journal of Computer Mediated Communication, 3(1).

[7] M. Gladwell. The Tipping Point: How Little Things Can Make a Big Difference (pp. 30-88). Back Bay Books, 2002.

[8] O. Goldreich, S. Micali and A. Widgerson. How to play any mental game. In *STOC'87*, pp. 218–229. ACM, 1987.

[9] P. Golle and M. Jakobsson. Reusable Anonymous Return Channels. In *ACM Workshop on Privacy in the Electronic Society '03,* pp. 94–100. ACM Press, 2003.

[10] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of USENIX'02*, pp. 339–353.

[11] M. Jakobsson and C. Schnorr. Efficient Oblivious Proofs of Correct Exponentiation. In *Proc. of CMS .99.*

[12] M. Kochen. The small world. 1989. Norwood, NJ: Ablex.

[13] V. E. Krebs. Uncloaking terrorist networks. In *First Monday*, Vol. 7 (4), April 2002.

[14] A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. of ACM CCS '01*, pp. 116–125.

[15] A. Newitz. Defenses lacking at social network sites. SecurityFocus, Dec 31, 2003. `http://www.securityfocus.com/news/7739`

[16] M. Newman. Who is the best connected scientist? A study of scientific coauthorship networks. In *Phys Rev*, E 64.

[17] W. Ogata, K. Kurosawa, K. Sako and K. Takatani. Fault tolerant anonymous channel. In *Proc. of ICICS '97*, pp. 440-444, 1997. LNCS 1334.

[18] `http://www.orgnet.com/`

[19] T. Pedersen. A Threshold Cryptosystem Without a Trusted Third Party. In *Proc. of Eurocrypt '91*, pp. 129–140. LNCS 547.

[20] R. Rice. Network analysis and computer-mediated communication systems. In *Advances in social network analysis*, pp. 167–203. 1994.

[21] C.P. Schnorr. Efficient signature generation by smart cards. In *Journal of Cryptology*, 4:161.174, 1991.

[22] M. A. Smith. *Communities in Cyberspace.* Routledge, London, 1999, pp. 195–219.

[23] M. K. Sparrow. The application of network analysis to criminal intelligence: an assessment of the prospects. In *Social Networks*, Vol. 13, pp. 251–274.

[24] J. Tyler, D. Wilkinson and B. A. Huberman. Email as Spectroscopy: Automated Discovery of Community Structure within Organizations. In the *proceedings of Communities & Technologies 2003*, pp. 81–96.

[25] B. Welman. *Networks in the Global Village.* Westview Press, Boulder, CO, 1999.

[26] B. Wellman. Computer networks as social networks. In *Science*, 293, 14, Sept 2001, pp. 2031–34.

[27] A. C. Yao. Protocols for secure computations. In *FOCS'82*, pp. 160–164. IEEE Computer Society, 1982.