Privacy-Preserving Relationship Path Discovery in Social Networks

Ghita Mezzour[1], Adrian Perrig[1], Virgil Gligor[1], and Panos Papadimitratos[2]

[1] Carnegie Mellon University, Pittsburgh, PA 15213
{mezzour, perrig, gligor}@cmu.edu
[2] EPFL, Lausanne, Switzerland
panos.papadimitratos@epfl.ch

Abstract. As social networks sites continue to proliferate and are being used for an increasing variety of purposes, the privacy risks raised by the full access of social networking sites over user data become uncomfortable. A decentralized social network would help alleviate this problem, but offering the functionalities of social networking sites is a distributed manner is a challenging problem. In this paper, we provide techniques to instantiate one of the core functionalities of social networks: discovery of paths between individuals. Our algorithm preserves the privacy of relationship information, and can operate offline during the path discovery phase. We simulate our algorithm on real social network topologies.

1 Introduction

Social Networks have proliferated over the past few years, offering numerous services that attract millions of subscribers. In fact, Social Networking Sites are among the most frequently visited Internet sites (e.g., Myspace, Facebook). Their novel functionality and the ways of personal interaction they offer fuel their tremendous success.

A fundamental feature of social networks is the *relationship graph* that connects users. This graph enables two individuals to find the relationship paths that connect them. These paths are useful to express trustworthy users: nearby people (with short relationship paths connecting them) often deserve a higher level of trust. The path discovery mechanism can be used as a building block for many social networking applications: (1) discovering a relationship path to a recruiter may boost the chances of a job applicant to get the position; vice-versa, discovering a relationship path to an applicant could help the recruiter get a more trusted judgment on the applicant; (2) relationship path discovery can provide a basis for access control mechanisms suitable for Social Networks, where users determine the authorized users based on their distance to themselves in the social network; (3) a path to a person submitting an online review can boost confidence in the review; and (4) ensuring the receiver of an email that the sender is nearby in her social network can help avoid falsely flagging the email as spam.

Although the relationship graph is at the core of the usefulness of social networks, personal relationships represent sensitive, private information that can also be misused. A primary concern is the unwelcome linkage among users. For example, two professionals employed by rival companies that have a connection may trigger suspicion. Or, connections of innovators and venture capitalists could alert the competition by giving leads to upcoming technological developments. Or, simply, a social relationship can correspond to a sensitive personal real-world relationship. Of greater concern is the discovery of entire relationship paths and, in the end, of the entire graph. A

significant negative consequence of this discovery is the large-scale targeting, tracking, and monitoring of multiple individuals in real life based on discovered relationship paths. Other privacy concerns can arise from graph operations; e.g., user de-anonymization through merging of relationship graphs [15].

Problem Scope. Protecting the privacy of relationship paths in a social network is a separate concern from that of protecting the privacy of (1) other content of individual users, (2) pairwise inter-user relationships, or (3) inter-user relationships based on a single intermediary contact. The stored content of a social network site can be protected using cryptographic primitives [4, 10, 13]. Pairwise-private relationship protection arise in practical settings such as instant messaging [12].

The protection of a user's pairwise private relationships has received extensive coverage outside social networks; e.g., "trust-negotiation" between a client and a server. In trust negotiations, a client's relationships-revealing credentials could not be disclosed to a server, and the server's relationship-revealing access policies could not be disclosed to the client. In other settings, private set-intersection protocols (viz., Section 3) can maintain a limited type relationship privacy, namely help discover one intermediate contact between two users privately.

In all these examples, enforcement of privacy protection is a matter of *local* user policies; e.g., client-server policies, pairwise-private policies. In contrast, protecting relationship paths among users in social networks requires (uniform) enforcement of *global* privacy policies because all users are affected by unauthorized privacy breaches. This is a significantly more challenging problem, whose solutions, nevertheless, present opportunities for novel use in a variety of other applications.

Decentralized Access. Any effective solution to relationship-path privacy would be based on decentralized access control mechanisms and policies. Decentralized access control is required by both privacy and robustness concerns. First, mobile users should be able to discover other mobile users with whom they have a relationship without having to connect to a centrally administered social network site, which could track their movement or which would be unreachable without network access. For example, two nearby users should still be able to discover their private relationship paths even in the absence of Internet connectivity.

Second, perhaps more important, a single point of privacy failure should be avoided. The centralized, full access that Social Networking Sites have over relationship data pose serious privacy threats. Some Social Networking Sites have permissive privacy policies that enable them to use such data for commercial purposes. Moreover, even sites with strict privacy policies can be subject to disclose private information disclosure. For example, sites can be subpoenaed to disclose user information indiscriminately, which would be ineffective whenever such information would be decentralized; i.e., subpoenas would have to be network-node selective, need-to-know based and would yield limited user information. Another risk are malicious insiders with access to private information that can disclose private information. Finally, administrator errors or security vulnerabilities can also lead to privacy disclosures, as we have frequently witnessed recently in the case of leaked databases containing credit card and social security numbers.

Design Challenges. Implementing decentralized access control to protect relationship-path privacy poses significant challenges, and obvious approaches do not address the problem. An approach where people make their relationships list visible to their neighbors is not viable because

that would disclose sensitive information. Another approach would be to flood relationship path discovery messages throughout the social network, but that would incur high overhead and latency, especially if some users are offline; they would delay path discovery until they are online. Furthermore, these challenges are exacerbated during relationship-path discovery. For example, if two users want to discover private relationship paths to each other, the path-discovery protocol could not rely on all intermediate users sites to be on-line and help find relationship paths. In other words, two users should be able to discover private relationship paths to each other based only on their local communication.

Contributions. In this paper we propose a system that enables users to benefit from social networking applications without the privacy exposure associated with having all their data stored at a central site. The key contribution of this paper is thus our privacy-preserving multi-hop relationship path discovery mechanism. Our design enables new access control policies based on social relationships, new trust establishment policies, and new ways to manage communication applications (e.g., e-mail spam).

2 System Model and Problem Statement

A *relationship path* between two users *u* and *v* is a sequence of users whose pairwise relationships (i.e., friendship) connect the two users. We call the two users the *end users* of the relationship path.

The *distance* (depth) *d* between two users on a relationship path is the number of edges (aka hops) from one user to the other on that relationship path.

A *bridge contact* user u_i, of user *u* to a user *v* on some relationship path is the direct relationship (i.e., edge) of user *u* to user *v* on that relationship path.

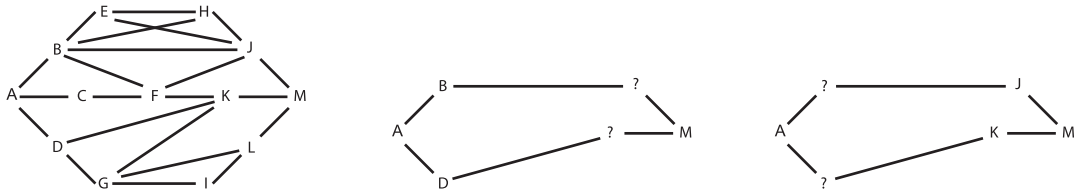
A *private relationship path* from user *u* to user *v* is a tuple (u, u_i, d, v) encoding a relationship path $(u, u_i, ..., v_j, v)$ of distance *d*. By symmetry, (v, v_i, d, u) is the private relationship path from user *v* to user *u* of distance *d*. Whenever a relationship path $(u, u_i, ..., v_j, v)$ is private, neither user (*u*, *v*) can discover any intermediate users on that path to the other user beyond its respective bridge contact (u_i, v_j). Of course, there may be multiple private relationship paths from *u* to *v* with the same bridge contact and different distances, and an user may have multiple bridge contacts each on a different private relationship path to the other user.

A bridge contact u_i has different roles: (1) it can facilitate the introduction of *u* to *v*. This can be relevant, for example, in a job search scenario. (2) It helps *u* assign a trust value to *v*, which can be useful when the private relationship paths are used to enforce access control. (3) It helps *u* track/blacklist misbehaving users. Assume that in the email whitelisting scenario, many of u_i's friends send spam messages to *u*. *u* may no longer want to whitelist a user *v* to which *u* has a private relationship path (u, u_i, d, v). Whenever there are multiple private relationship paths (u, u_i, d_i, v), then *u* would assign trust value *v* based on the shortest private relationship path.

An *obfuscated private relationship path* $(u, u_i, d, ?)$ is a private relationship path, where the identity of one end user is unknown.

The example topology of Figure 1(a) is provided to illustrate the use of our system entities defined above. In that figure, users *A* and *M* discover private relationship paths to each other at

distances $d \leq 3$. A discovers the two private relationship paths $(A,B,3,M)$ and $(A,D,3,M)$, where B and D are the bridge contacts and 3 is the distance of the private relationship paths. These private relationship paths captured by graph edges $A-B-?-M$ and $A-D-?-M$, depicted in Figure 1(b), where "?" means that user A does not know the identity of the corresponding user. User M discovers the two private relationship paths $(M,J,3,A)$ and $(M,K,3,A)$, corresponding to $M-J-?-A$ and $M-K-?-A$ depicted in Figure 1(c).



(a) Example topology (b) A(lice) (c) M(ary)

Fig. 1. Example topology and private relationship paths of distance $d \leq 3$ discovered by A(lice) and M(ary).

2.1 Assumptions

We assume that each user v is free to choose the users with whom she wishes to establish a relationship, but that relationships can only be established based on the consent of both users. We assume the existence of a secure and authenticated channel between each pair of friends. We assume that friends can communicate anonymously, for example using Tor. We assume that v has a public-private homomorphic key pair. We also assume that two users who want to discover private relationship paths can authenticate each other's homomorphic public keys.

In this work, we also assume the typical case where relationship paths decline in value with increasing length. Hence, users do not gain any benefits from increasing the length of their own relationship paths. This assumption is based on the observation that for all social network applications in use today, shorter paths are more desirable and take precedence over longer paths.

2.2 Desired Properties

Privacy. The central goal of our protocol is to enable two parties to discover private relationship paths to each other in an efficient manner while minimizing the additional information about relationships that users can learn. We will compare our approaches to an *ideal scheme*, which returns private relationship paths without disclosing any additional information. We say that our approach provides privacy if it does not leak any more information than the ideal scheme.

Path Integrity. The protocol needs to provide *path integrity* for the discovered paths, i.e., an adversary cannot alter discovered paths. An adversary may attempt to violate path integrity in two ways: alter the bridge contact, or shorten the discovered path.

Completeness. Our protocol should also be *complete* in the sense that each party discovers all the discoverable private relationship paths. On a discoverable private relationship path, all users on the corresponding relationship path do not object the discovery.
Offline discovery. Our protocol should enable two users to discover private relationship paths offline, i.e., based only on their physical interaction and without the help of intermediate users. Not requiring intermediate users to be online during the discovery phase is very appealing for most situations. The ability of two users to discover private relationship paths based exclusively on their local interaction, without being connected to the Internet, is particularly relevant for mobile users.
Low overhead. We aim for low communication and computation overhead.

2.3 Adversary Model

We distinguish between two kinds of adversaries: internal and network adversaries. An internal adversary sets up an account and creates relationships with users they can e.g., her friends in real life. The network adversary controls the communication channel between users, can eavesdrop, stop or inject messages. This adversary is, however, not part of the social network, and does not participate in our protocol.

The internal adversary is free to arbitrarily deviate from our protocols. The adversary may wish to alter the topology of the social network to bring more value to themselves. For example, an adversary may want to make other users see her nearby (i.e., at a small distance from them). The adversary M may also want to alter the bridge contact in a private relationship path discovered by some user u to M. v may trust one bridge contact more than other. Similarly, the adversary may try to deny value to honest users. The adversary may also want to break the privacy of the social network by discovering the relationships of other users. A misbehavior is not considered an attack if it only results in an outcome permitted in social networks as described in Section 2.1. For example, a user may not want an extremely sensitive relationship to be leaked. This user suppresses all private relationship paths corresponding to a relationship path containing that relationship.

The network adversary can observe the network traffic of users in order to learn about their relationships. It should be noted that people who use regular e-mails or encrypted e-mails are already vulnerable to having this kind of adversary learn their relationships. The network adversary can observe the source and destination of their e-mails, as well as the frequency at which they send and receive e-mails from each person. This combined information enables the network adversary to infer the friends of a user as the people with whom the user has many e-mail exchanges. People concerned about this kind of adversary usually use some form of anonymous communication. We do not discuss this kind of adversary further in the paper.

3 Background on Privacy-Preserving Cryptographic Techniques

In this section, we present background on privacy-preserving cryptographic techniques that we will build on. Private set intersection protocols [3, 8, 11] enable two or more parties that each hold a set of inputs drawn from a large domain to jointly calculate the intersection of their inputs, without

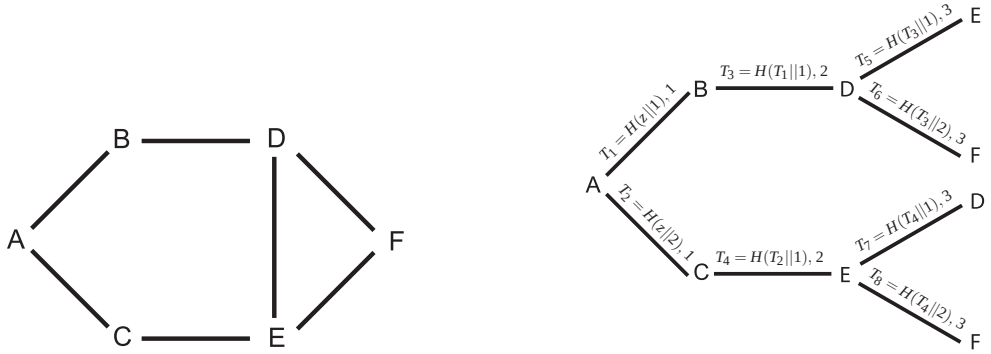
leaking additional information. The private set intersection proposed by Freedman et al. [8][1] is a two-party protocol between a client C and a server S. C's input is a set of size k_C, drawn from some domain of size N; S's input is a set of size k_S drawn from the same domain. At the conclusion of the protocol, C learns which specific inputs are shared by both C and S. That is, if C inputs $X = x_1, \ldots, x_{k_C}$ and S inputs $Y = y_1, \ldots, y_{k_S}$, C learns $X \cap Y : \{x_i | \exists j, x_i = y_j\}$. The protocol is based on the presentation of sets as roots of a polynomial and on the use of homomorphic cryptosystems. The protocol follows the basic structure. C defines a polynomial P whose roots are her inputs: $P(y) = (x_1 - y)(x_2 - y)\cdots(x_{k_C} - y) = \sum_{w=0}^{k_C} \alpha_w y^w$. C sends to S homomorphic encryptions of the coefficients of this polynomial. S uses the homomorphic properties of the encryption system to evaluate the polynomial at each of her inputs. She then multiplies each result by a fresh random number n to get an intermediate result, and she adds to it an encryption of the value of her input. That is, S computes $Enc(n.P(y_j) + y_j)$ for each of her inputs $y_j \in Y$. S randomly permutes this set and returns it to C. C decrypts each element of this set. For each element in the intersection of the two parties' inputs, the result of this computation is the value of the corresponding element, whereas for all others the result is random. The computation overhead for C mainly consists of k_C homomorphic encryptions and k_S homomorphic decryptions. The k_C homomorphic encryptions only need to be computed once per input set. The computation overhead for S is mainly due to the evaluation of each of her inputs on a degree-k_C polynomial. The use of multiple-low degree polynomials and Horner's Rule make the asymptotic computation overhead $O(k_C + k_S \ln\ln k_C)$ exponentiations.

A slight variation of the above protocol is provably secure in the random oracle model against malicious adversaries, where a malicious adversary may behave arbitrarily. The security definition of private set intersection protocols is based on a comparison between an ideal and a real implementation. In an ideal implementation, a trusted third party receives the inputs of the two parties and outputs the result of the intersection, whereas in the real implementation there is no trusted third party. The security model requires that in the real implementation of the protocol, each party does not learn more information than the ideal implementation. Such a security model does not deal with attacks that apply to the ideal model. For example, a party may lie about her input set.

4 Protocol Overview

The goal of our protocol is to enable two users to discover private relationship paths to each other offline, without disclosing superfluous relationships. This requires users to store topology information that enables the discovery, without revealing relationships. Our protocol operates in two phases: a *token flooding phase* and a *path discovery phase*. During the infrequent token flooding phase, we assume that users are online so they can exchange obfuscated topology information. The path discovery phase runs when two users want to discover private relationship paths to each other, offline based on the collected obfuscated topology information; it returns to the two users private relationship paths.

[1] The paper refers to the protocol both by private set intersection and private matching. We only refer to the protocol as private set intersection in order to avoid confusion with matchmaking protocols.



(a) Extended Hexagonal topology

(b) Tokens propagated

Fig. 2. Basic scheme. Extended hexagonal topology and obfuscated tokens propagated for originator *A*.

During the token flooding phase, users disseminate cryptographic tokens that become associated with obfuscated relationship paths. Each user issues tokens to her neighbors to explore relationship paths starting from herself. Other users will cryptographically obfuscate these tokens and continue the flooding process up to a pre-determined number of hops. The obfuscation operation is deterministic and independent of the identity of the user. This enables the originator to compute the value of all possible tokens at any distance. The originator will in fact simulate the flooding and compute the obfuscated token values at all distances up to the maximum flooding depth. The originator associates each computed token with an obfuscated relationship path. The path discovery phase runs when two users want to discover private relationship paths to each other. The two users perform a private set intersection protocol: one user plays the role of a client and enters the tokens she has computed as originator, and the other plays the role of a server and enters the tokens received. The first user learns the existing private relationship paths based on the common tokens. A second run of the private set intersection protocol with the roles of the two users inverted enables the second user to learn existing private relationship paths. The second private set intersection protocol does not run if *v* is not interested in discovering private relationship paths.

We now explain some more details based on the example in Figure 2(a). We consider the token flooding phase where node *A* is the originator and floods her tokens through the network. To limit the flood up to a maximum distance, each token is accompanied with the distance from the originator. Figure 2(b) depicts the hash tree that is created during the flooding process for *A*'s tokens. To initiate the token flooding process, *A* picks a random number *z*. To prevent nodes from inferring relationship topology information during the flooding process, each token is obfuscated through a one-way hash function H and a counter value – through this approach each token represents a unique sequence of users. As Figure 2(b) shows, *A* sends token $T_1 = H(z \,||\, 1)$ to user *B*, and token $T_2 = H(z \,||\, 2)$ to user *C*, along with the distance 1. *B* and *C* obfuscate the token through another application of the one-way hash function and a counter value, and forwards the unique token values

to their neighbors along with the distance value 2. Users store the received tokens in their list of received tokens.

Figure 2(b) only shows the tokens that are distributed for user A; in practice each user distributes her own set of tokens. Due to the deterministic operation of the token obfuscation process, the token originator can simulate the token flooding phase and compute all tokens depicted in the hash tree in Figure 2(b).

To demonstrate the discovery of a private relationship path, consider the case where users A and F meet. One user will use the computed list of obfuscated tokens, and the other user will use the list of all received tokens. In our example in Figure 2(b), A will compute tokens $T_1, T_2, \ldots, T_8$, and F will use all received tokens, which include tokens T_6 and T_8. After performing private set intersection of the two lists, they find a match with tokens T_6 and T_8. Since A knows that token T_6 was derived from token T_1 which was handed to user B, B is the bridge contact for that private relationship path. A further knows that T_6 is at distance 3 hops. Analogously, A knows that C is the bridge contact for the 3-hop long path represented by T_8.

5 Protocol Description

We describe next our protocol in more detail, assuming all users follow the protocol. Sections 6.1, and 6.2 discuss protocol deviations. Table 1 describes our notation.

Table 1. Notation.

o	originator
r	relay node
v, u	users in the social network
x_i	i^{th} friend of x
$\|x\|$	number of friends of user x
d	distance
d_{max}	maximum distance of private relationship paths
deg_{max}	maximum number of neighbors of any node

5.1 Basic Scheme

In this section, we first describe a basic version of our protocol and extend it in Section 5.2.

Token Flooding Phase. As we describe in Section 4, the originator floods tokens that represent obfuscated paths and that can later be used to discover private relationship paths. Algorithm 1 is used by the originator o to construct and send tokens to her friends. To start, the originator o generates a random number z and uses it as a seed to compute nodes at depth 1 of a hash tree. o computes $T_i = H(z||i), i = 1, ..., |o|$ and sends $(d = 1, T_i)$ to each friend o_i.

Algorithm 1 Basic scheme. Token propagation. Originator o.

Generate a random number z
Send $(d = 1, T_i = H(z||i))$ to o_i, $i = 1, \ldots, |o|$

Algorithm 2 Basic scheme. Token propagation. Relay r.

for reception of (d, T) from r_i **do**
  Insert (r_i, d, T) into list of received tokens

  if $d < d_{max}$ **then**
    Send $(d+1, T_j = H(T||j))$ to r_j, $j = 1, \ldots, |r|$, $j \neq i$
  end if
end for

Algorithm 3 Basic scheme. Hash tree construction. Originator o.

Insert$(o_i, 1, T_i = H(z||i))$, $i = 1, \ldots, |o|$ into hash tree
while Unmarked (o_i, d, T) **do**
  Mark (o_i, d, T)

  if $d < d_{max}$ **then**
    Insert $(o_i, d+1, T_j = H(T||j))$, $j = 1, \ldots, deg_{max}$ in hash tree
  end if
end while

Algorithm 2 is used by the friends of the originator to construct and send tokens to their friends. This propagation continues hop-by-hop up to distance $d = d_{max}$ from the originator. A user performing this propagation is termed a *relay* r. When r receives (d, T) from her friend r_i, she inserts T into her list of received tokens. The relay r_i computes $T_j = H(T||j), j \in \{1, ..., |r|\}$, $j \neq i$, and sends $(d+1, T_j)$ to each of her friends $r_j \neq r_i$. Figure 2(b) presents the hash tree spanned by A's token propagation, where A plays the role of an originator.

The originator o can reconstruct the hash tree created during the token flooding phase by users at distance $d < d_{max}$. This way, the originator has a common token with each user at distance $d \leq d_{max}$. Each of these tokens is associated with an obfuscated relationship path. The tokens that are common between the originator and users at distance $d \leq d_{max}$ constitute the basis for discovering private relationship paths during the path discovery phase (explained in detail below). Algorithm 3 is used by the originator to reconstruct all tokens from the hash tree. The originator o constructs a hash tree based on the seed z, the maximum degree deg_{max} and depth d_{max}. Each token of the hash tree is associated with an obfuscated private relationship path. A token T_i at depth 1 is a token associated with the trivial obfuscated private relationship paths $(o, o_i, 1, ?)$, as the token T_i was sent to the friend o_i. A node T at depth d with ancestor T_i is associated with the obfuscated private relationship path $(o, o_i, d, ?)$.

Path Discovery Phase. The path discovery phase runs when two users u and v want to discover private relationship paths. Users u and v run a private set intersection protocol (as described in

Section 3) where u plays the role of the client entering the tokens in her hash tree and v plays the role of the server entering the tokens in her list of received tokens. The private set intersection outputs to u the tokens T_k in u's hash tree corresponding to the entries (u_k, d_k, T_k) in u's hash tree. This enables u to discover private relationship paths (u, u_k, d_k, v). A second private set intersection protocol with the roles of u and v inverted. That is, v plays the role of the client and enters the tokens in her hash tree as input, and u plays the role of the server and enters the tokens in her list of received tokens as input. This second private set intersection enables v to discover private relationship paths to u.

We consider the extended hexagonal topology, with users A and F running the path discovery phase. A and F run a private set intersection protocol where A enters the tokens in her hash tree as input, while F enters the tokens in her list of received tokens as input. A's hash tree contains tokens $T_1, \cdots, T_8$ depicted in figure 2(b). During the token flooding phase, F received tokens T_6, T_8. The private set intersection protocol outputs T_6, T_8 for A. T_6 enables A to learn $(A, B, 3, F)$. A knows that B is the bridge contact since A knows that T_6 was derived from T_1 that was handed to B. A further knows that F is at distance 3 since T_6 was derived from T_1 by applying three times a hash function to the seed z. Analogously, A discovers the private relationship path $(A, C, 3, F)$ from T_8. A second private set intersection protocol is run with the roles of A and F inverted. F learns the private relationship paths $(F, D, 3, A)$ and $(F, E, 3, A)$.

5.2 Extended Scheme

The basic scheme, unfortunately, suffers from a privacy leak: it is possible to learn whether discovered private relationship paths intersect at specific intermediate users. Assume that in the extended hexagonal topology, A runs the path discovery phase with F and subsequently with D. A discovers the private relationship paths $(A, B, 3, F)$, $(A, C, 3, F)$ based on T_6, T_8 and later $(A, B, 2, D)$, $(A, C, 3, D)$ based on T_3, T_7. Since A knows that T_3 is the ancestor of T_6 in the hash tree, A learns that D is the user at distance 2 in the private relationship path $(A, B, 3, F)$. Similarly, since T_7 and T_8 have the same ancestor, A learns that $(A, C, 3, D)$ and $(A, C, 3, F)$ have the same intermediate user at distance 2. The basic scheme also has a very large overhead since an originator computes a hash tree of degree deg_{max}. Since most users have fewer than deg_{max} friends, the basic scheme computes and stores many unnecessary tokens in the hash tree.

We present an extended scheme to address these shortcomings. The extended scheme utilizes a randomization technique to seal the privacy leak and an optimization to reduce the overhead associated with the computation of unnecessary tokens. The goal of the randomization technique is to prevent an originator from learning information about intermediate users on a private relationship path beyond what can be directly inferred from discovered private relationship paths. This holds even after the originator runs a path discovery phase with multiple users. In the extended scheme, we separate the computation of the token of a user v from the computation of tokens of intermediate users between v and the bridge contact. More specifically, for a given bridge contact, we separate the computation of tokens based on the distance to the originator. A user at distance $2 \leq d \leq d_{max}$ receives a token randomly chosen from the set of tokens computed for d.

Algorithm 4 Extended scheme. Token propagation. Originator o.

Generate a random number z
Send $(d = 1, T_i^1 = H(z||'1'||i))$ to o_i, $i = 1, \ldots, |o|$

Our basic observation towards designing the extended scheme is that since the bridge contact is disclosed with the private relationship path, the bridge contact can directly create tokens for all subsequent users at $d \leq d_{max}$. Since in practice $d_{max} = 3$, the bridge contacts creates tokens for distances 2 and 3. The bridge contact obtains the number of tokens for distance 3 by asking her friends about their degrees and summing these degrees. Tokens for distance 3 are computed separately from the ones for distance 2. The bridge contact sends to each of her friends one token for distance 2 and a number of tokens for distance 3 proportional to the friend's degree. The tokens transmitted are randomly chosen from the computed tokens. The friend stores the token for distance 2 and to each of her friends one of the received tokens for distance 3. This approach addresses the inefficiency of the basic scheme, because each bridge contact can inform the originator of how many tokens were distributed.

We now describe the extended scheme in more detail for the case of $d_{max} = 3$, which we consider to be the largest value for d_{max} that is viable in practice. Algorithm 4 presents the algorithm used by the originator o to construct and forward tokens to her friends. o generates a random number z and uses it to create tokens for her friends as $T_i^1 = H(z||'1'||i), i = 1, \cdots, |o|$ and sends $(1, T_i^1)$ to each of her friends. These friends constitute the bridge contacts. Algorithm 5 presents the algorithm used by a bridge contact $b = o_i$ to construct and send tokens. The algorithm assumes that b previously received the value of $|b_i|$ from each of her friends. When b receives a token T from o, b computes tokens for all subsequent users at distances 2 and 3. Tokens for distance 2 are computed as $T_i^2 = H(T||'2'||i)$, $i = 1, \cdots, p = |b| - 1$. Tokens for distance 3 are computed as $T_j^3 = H(T||'3'||j)$, $j = 1, \cdots, q = \sum_{i=1, b_i \neq o}^{|b|}(|b_i| - 1)$. Similarly to the basic scheme, each token encodes the distance to the originator. b sends to each of her friends b_i a token for distance 2 and $|b_i| - 1$ tokens for distance 3. These tokens are randomly chosen from tokens computed for distances 2 and 3 respectively. At the end, b informs o about (p, q) the number of tokens computed for distances 2 and 3. A subsequent user on the relationship path is termed as a *relay* r. Algorithm 6 presents the algorithm used by r to forward tokens to her friends. When r receives from a bridge contact $((2, T^2), T_i^3, i = 1, \cdots, |r| - 1)$, r stores T^2. r sends to each of her friends $(3, T^3)$, where T^3 is randomly chosen from the received tokens T_i^3. Figure 5.2 presents the hash tree spanned by A's token propagation.

Similarly to the basic scheme, the originator computes the hash tree of tokens received in the network, where each token is associated with an obfuscated private relationship path. Algorithm 7 presents the algorithm used by the originator. The algorithm takes as input (p_i, q_i), the number of tokens computed for distances 2 and 3 by each bridge contact o_i. For each bridge contact o_i, the originator constructs a token for distance 1 as $T_i^1 = H(z||'1'||i)$, where z is the previously generated seed. The originator constructs tokens for distance 2 as $T_j^2 = H(T^1||'2'||j), j = 1, \cdots, p_i$ and associates them with obfuscated private relationship paths $(o, o_i, 2, ?)$. Similarly, tokens for distance 3

Algorithm 5 Extended scheme. Token propagation. Bridge contact b.

Require: $|b_i|,\ i = 1,\cdots,|b|$
  for reception of $(1,T)$ from o **do**
    Insert T in list of received tokens

    $p = |b| - 1$
    $T_i^2 = H(T||'2'||i),\ i = 1,\cdots,p$

    $q = \sum_{i=1,b_i\neq o}^{|b|}(|b_i| - 1)$
    $T_j^3 = H(T||'3'||j),\ j = 1,\cdots,q$

    Shuffle T_i^2
    Shuffle T_j^3
    Send $(2,T_i^2)$ and $(|r_i| - 1)$ values from T_j^3 to r_i
    Send (p,q) to o
  end for

Algorithm 6 Extended scheme. Token propagation. Relay r.

  for reception of $((2,T),T_i^3,\ i = 1,\ldots,|r| - 1)$ from bridge contact b **do**
    Insert T in list of received tokens
    Shuffle T_i^3
    Send $(3,T_i^3)$ to $r_i,\ i = 1,\ldots,|r|,\ r_i \neq b$
  end for

  for reception of $(3,T)$ from r_i **do**
    Insert T in list of received tokens
  end for

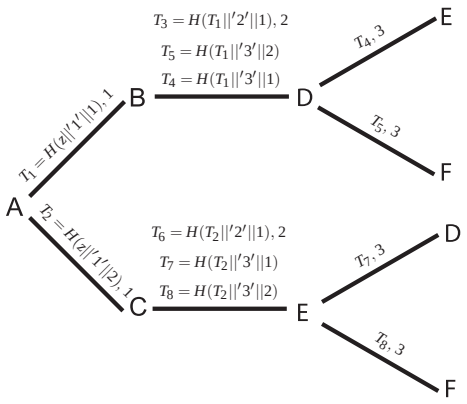


Fig. 3. Extended scheme. Obfuscated tokens propagated for originator A in the extended hexagonal topology.

are constructed as $T_j^3 = H(T^1||'3'||j), j = 1,\cdots,q_i$ and associated with obfuscated private relationship paths $(o,o_i,3,?)$. The hash tree constructed by this algorithm has a different structure than the

Algorithm 7 Extended scheme. Hash tree construction. Originator o.

Require: (p_i, q_i) number of tokens for distances 2 and 3 computed by bridge contacts o_i, $i = 1, \cdots, |o|$
  for $i = 1, \cdots, |o|$ **do**
    Insert $(o_i, 1, T^1 = H(z||'1'||i))$ in hash tree
    Insert $(o_i, 2, T^2_j = H(T^1||'2'||j))$, $j = 1, \cdots, p_i$ in hash tree
    Insert $(o_i, 3, T^3_j = H(T^1||'3'||j))$, $j = 1, \cdots, q_i$ in hash tree
  end for

one for the basic scheme. The depth of the tree is 2. The root z has degree $|o|$, whereas a token for distance 1 handed to o_i has degree $p_i + q_i$. For example, T_1 has 3 children: T_3, T_4 and T_5.

The path discovery phase runs similarly to the basic scheme. A private set intersection between u and v enables u to discover common tokens and learn private relationship paths.

In practice, a user does not inform others about her exact degree. Instead, the user adds to it a positive noise in order to further hide the network topology. To be consistent with the noise added, the user creates and sends dummy tokens as if she had additional friends. The amount of the noise to be added in order to appropriately hide the network topology is beyond the scope of this paper. It should be noted that the extended scheme does not require additional trustworthiness from the bridge contact. The bridge contact is free to deviate from the protocol in order to suppress private relationship paths containing her relationships, as any other intermediate user.

In our discussion, we considered $d_{max} = 3$. However, the scheme can be easily extended to a larger d_{max}.

6 Evaluation

In this section, we evaluate the extended scheme. Sections 6.1 and 6.2 provide a security and a privacy analysis. Section 6.3 analyzes the scheme overhead. Simulations based on real social network topologies are presented in Section 6.4. We consider $d_{max} = 3$.

6.1 Security Analysis

Completeness. Assume there exists a relationship path $(u, u_i, \cdots, v_j, v)$ of distance $d \leq d_{max}$. Completeness requires that when u and v run a path discovery phase, u discovers (u, u_i, d, v) and v discovers (v, v_j, d, u) in case all users on the relationship path consent to the discovery. In our protocol, a user is provided with the flexibility to suppress all private relationship paths corresponding to paths containing one of her relationships. This relationship may be extremely sensitive.

We focus on relationship paths with distance 3. The discussion can be easily adapted to relationship paths with distance 2. We consider the discovery by u, the one by v being very similar. We list sufficient conditions for u's discovery: (1) during the token flooding phase, v receives a well constructed token T originating from u and u computes T as part of her hash tree and associates it with the obfuscated private relationship path $(u, u_i, d, ?)$; and (2) during the path discovery phase, both u and v enter T to the private set intersection protocol instance where u plays the role of a client and v plays the role of a server.

The first condition holds if u sends a token associated with $d = 1$ to u_i, u_i receives an upper bound on $|v_j|$ from v_j, constructs and forwards as many tokens for $d = 3$ (one of these tokens being T) to v_j, and v_j forwards T to v. The second condition holds if u computes all tokens encoding $d = 3$ forwarded by u_i to v_j, and associates them with the obfuscated private relationship path $(u, u_i, 3, ?)$. u can perform such computation if u_i informs her about the total number of constructed tokens encoding $d = 3$. It can be seen therefore that the first condition holds if all users on the relationship path consent to the discovery by performing the required actions. Similarly, the second condition only depends on the consent of u and v.

In order to further clarify why completeness holds in our protocol, we consider one specific protocol deviation. Assume an intermediate user on the relationship path $(u, u_i, \cdots, v_j, v)$ does not forward tokens to subsequent users. u and v will not be able to discover the corresponding private relationship paths. This, however, does not break the completeness property. In our protocol, a user is provided with the flexibility of suppressing private relationship paths corresponding to a relationship path containing one of her relationships.

Path Integrity. Path integrity requires than an adversary M cannot alter the bridge contact or shorten the discovered path. In our protocol, u discovers a private relationship path (u, u_i, d, M) when running a path discovery phase with M if: (1) M owns a token that u associates with the obfuscated private relationship path $(u, u_i, d, ?)$ and (2) both u and M enter that token to the private set intersection protocol.

When M is at distance $d > d_{max}$ from u, M does not receive any token originating from u. M is only left with the option of generating random tokens. This attack is considered of limited scope. We now consider the existence of one relationship path $(u, u_i, \cdots, M)$ of distance $d \leq d_{max}$. M receives a token T corresponding to the private relationship path (u, u_i, d, M). Additionally, M has access to tokens corresponding to longer private relationship paths with the same bridge contact because of the hop-by-hop token propagation. T cannot be used to compute tokens corresponding to shorter obfuscated paths with a different bridge contact, thanks to the preimage resistance property of a cryptographic hash function. When there are multiple relationship paths of distance $d \leq d_{max}$ from u to M, M receives a token originating from u for each of these relationship paths. Here also, the preimage resistance property of a cryptographic hash function prevents these tokens from being used to compute a token corresponding to a shorter obfuscated private relationship path or one with a different bridge contact.

In our protocol, M can make a user v appear at a shorter distance to another user u. Consider a relationship path (u, u_i, M, v). M can forward its own token to v, making v appear at distance 2 to u. This misbehavior causes disturbance to the system, but does not directly benefit M.

6.2 Privacy of the Network Topology

We compare our protocol to an ideal scheme, which returns private relationship paths without disclosing any additional information. we first consider an honest but curious adversary that follows our protocol, but performs all possible computation on available data. Later, we consider a malicious adversary that deviates from our protocol.

Honest but Curious Adversary. During the token flooding phase, users learn no information about the network topology in an ideal scheme. In our protocol, we distinguish between the three roles played by each user *v*: an originator, a bridge contact and a relay. As an originator, *v* learns a noisy version of the number of users at distances 2 and 3, for each of her bridge contacts. We clarify that a user having two distinct relationship paths of distance 2 or 3 to *v* is counted twice in the number learnt by *v*. *v* receives this information directly from the bridge contacts as described in Algorithm 5. As a bridge contact, *v* does not learn additional information, compared to what *v* already learnt as an originator. As a relay, *v* can infer the number of users at distances 2 and 3 through a simple count of the received tokens. *v* already learnt this information as an originator. *v* does not learn whether two received tokens have the same originator, when *v* did not originate any of the two tokens. This is achieved thanks to the way tokens are constructed. However, *v* can recognize that she is the originator of a received token. *v* receives one of her tokens when *v* is on a cycle of distance $d \leq d_{max} = 3$. Cycles of distances 1 and 2 are trivial. The ones of distance 3 are the only interesting case. These cycles enable *v* to learn about the existence of a friendship relationship between any two of her friends. Although, in the ideal scheme, *v* does not learn about the existence of these relationships during the token flooding phase, *v* can easily learn about them by running a path discovery phase with her friends. Similarly, *v* does not know whether the same intermediate user (beyond direct friends) was involved in the forwarding of two received tokens, if *v* was not involved in the forwarding of any of the two tokens. *v* can be involved in the forwarding of a token either as a relay, bridge contact or originator. We consider again the example topology in Figure 1(a). Figure 6.2 presents *A*(lice)'s perception of the social network topology by the end of the token flooding phase. The figure does not depict the noise added by users to their degrees.

During the path discovery phase, the two parties *u* and *v* exclusively learn existing private relationship paths, in an ideal scheme. In our protocol, *u* learns the common tokens between her hash tree and *v*'s list of received tokens. The construction of these tokens and the randomization technique prevent these tokens from leaking information beyond existing private relationship paths to *u*. However, the perception of the social network topology gained by *u* by the end of the token flooding phase may help her gain some additional knowledge. Such knowledge is limited in case users have a large number of friends. It is further reduced by the noise added by users to their degrees. We consider the example topology in Figure 1(a). Assume that *A*(lice) runs a path discovery phase with *M*(ary) and subsequently with *G*(ary). *A* discovers $(A,B,3,M)$, $(A,D,3,M)$ and subsequently $(A,D,2,G)$. The tokens of *M* and *G* were randomly chosen from the total set of tokens computed by *D* for distances 2 and 3 respectively. This prevents *A* from learning whether *G* is *D*'s friend present in $(A,D,3,M)$. However, from Figure 6.2, *A* knows that $|D| = 3$. *A* gains more confidence up to whether *G* is *D*'s friend present in $(A,D,3,F)$, compared to the ideal scheme, where *A* does not know $|D|$. This confidence is limited in typical social network topologies where $|D|$ would be much larger. It is further reduced by the noise added by *D* to $|D|$.

Malicious Adversary. We examine relationship information that can be leaked to a malicious adversary, but not to a honest but curious one. More specifically, we are interested in misbehaviors that aim at breaking relationship privacy. In our protocol, relationship information can be learnt in

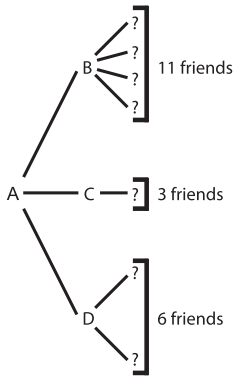


Fig. 4. *A*'s perception of the topology by the end of the token flooding phase for the topology from Figure 1(a).

two ways:(1) during the token flooding phase, through analysis of data received from friends; and (2) by running a path discovery phase with other users. During the token flooding phase, a malicious adversary M learns additional relationship information compared to an honest but curious one if it can influence the received data in a way that leaks private information. However, the only data that M can both influence and receive is the one propagated through cycles of distance $d \leq d_{max} = 3$ containing M. Even if M associates the smallest possible distance 1 with a transmitted token, other users will stop the propagation after d_{max} hops from M. Such misbehavior can help M discover friendship relationships between any of her friends. However, an honest but curious adversary already knows about these relationships, as was already discussed. During the path discovery phase with a user v, a malicious adversary can enter to the private set intersection protocol more or less tokens than prescribed in our protocol. When M plays the role of a server, this kind of misbehavior is not beneficial as the private set intersection protocol does not output anything to the server. As a client, M does not gain anything from entering less tokens to the private set intersection protocol. When M enters additional tokens, M learns whether these tokens are in v's list of received tokens. Because of the large space of tokens, we assume that M has a token T that is in v's list of received tokens only when M was involved in the construction or the propagation of T. We first consider when M does not deviate during the token flooding phase. Beyond tokens in its hash tree, M has access to the tokens it received and to the ones that can be constructed from them. Entering these tokens to the private set intersection protocol makes M discover private relationship paths to v of distance $d \leq d_{max} - 1$. These private relationship paths are also discovered in the normal run of the protocol. By deviating during the token flooding phase, M can only receive maliciously constructed tokens through cycles of distance $d \leq d_{max}$, as was already explained. Tokens propagated through these cycles do not help M learn additional information.

6.3 Overhead Analysis

We distinguish the overhead of the token flooding phase and that of the path discovery phase. The token flooding phase needs to run very infrequently. The path discovery phase runs when two users u and v need to discover private relationship paths *for the first time*. We introduce a new variable for

the purpose of simplifying the notation for this section. F_v^i is the fan-out of a user v at depth i. That is F_v^i is the number of relationship paths starting at v, of distance $d \leq i$. In the extended hexagonal topology, $F_A^1 = 2$, $F_A^2 = 4$ and $F_A^3 = 8$.

Token Flooding Phase. As an originator, v computes F_v^3 tokens in constructing the hash tree and transmits F_v^1 tokens. As a bridge contact, v computes $O(F_v^1 . F_v^2)$ tokens and transmits a similar number of them. v receives F_v^1 tokens. As a relay, v does not perform any computation. v receives $O(F_v^3 + F_v^1 . F_v^2)$ and transmits $O(F_v^1 . F_v^2)$ tokens. Therefore, during the token flooding phase, the overhead is $O(F_v^3 + 2F_v^1 . F_v^2)$ hash computation and exchange.

Path Discovery Phase. We evaluate the overhead when u discovers private relationship paths to v. The overhead originates from the private set intersection protocol run where u plays the role of the client and enters the F_u^3 tokens in her hash tree and v plays the role of the server and enters the F_v^3 tokens in her list of received tokens. From Section 3, the computation overhead of u consists of F_u^3 homomorphic encryptions and F_v^3 homomorphic decryptions. The F_u^3 homomorphic encryptions only need to be computed once per input set. The computation overhead of v consists of $O(F_u^3 + F_v^3 \ln \ln F_u^3)$ exponentiations. The communication overhead of this step consists of $O(F_u^3 + F_v^3)$ exchange of homomorphic ciphertexts. The overhead for v's discovery can be obtained through a similar analysis.

6.4 Simulations

We carried out our overhead analysis based on graphs of major social networking sites: Flickr, LiveJournal, Orkut, YouTube. The graphs were crawled by Mislove et al. [14] in late 2006. Table 2 presents statistics about the social network topologies used.

Token Flooding Phase. Figure 5 presents the computation and communication overhead during the token flooding phase to an individual user. It presents the cdf of the number of tokens computed and exchanged in logarithmic scale. For Flickr, LiveJournal and YouTube, about 90% of users exchange less than 10^5 hash values, which is equivalent to 2 MB, given that a hash value consists of 20 B. For Orkut, more than 75% of users exchange less than 10^6 hash values equivalent to 20 MB and more than 90% of users exchange less than 10^7 tokens, equivalent to 200 MB. Similar trend applies to the computation overhead.

Path Discovery Phase. We consider $F_u^3 = F_v^3$. Figure 6 presents the computation overhead when user u discovers private relationship paths to user v. The overhead follows a similar trend compared to the token flooding phase. The main difference is that u performs homomorphic decryptions and v performs exponentiations. These operations are more expensive than hash computations. The communication overhead is not depicted. It follows a similar trend compared to the token flooding phase. The difference is that the items transmitted are homomorphic ciphertexts and not hash values. It should be noted, however, that the path discovery phase only needs to run once between two particular users u and v. After the first run, u and v can mark the common tokens with the identity of the other party. u and v can also establish a shared symmetric key for future use.

	Flickr	LiveJournal	Orkut	YouTube
Number of users	1,846,198	5,284,457	3,072,441	1,157,827
Estimated fraction of user population crawled	26.9 %	95.4 %	11.3 %	unknown
Number of friend links	22,613,981	77,402,652	223,534,301	4,945,382

Table 2. Statistics about the social network topologies used.

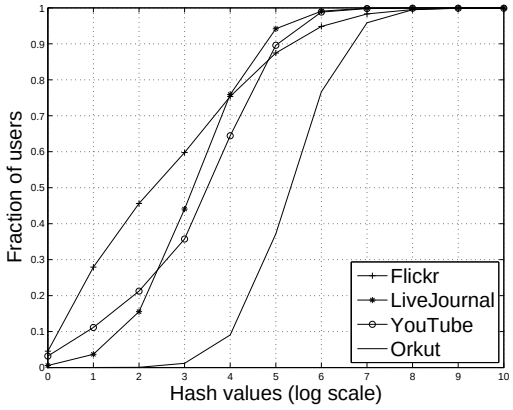
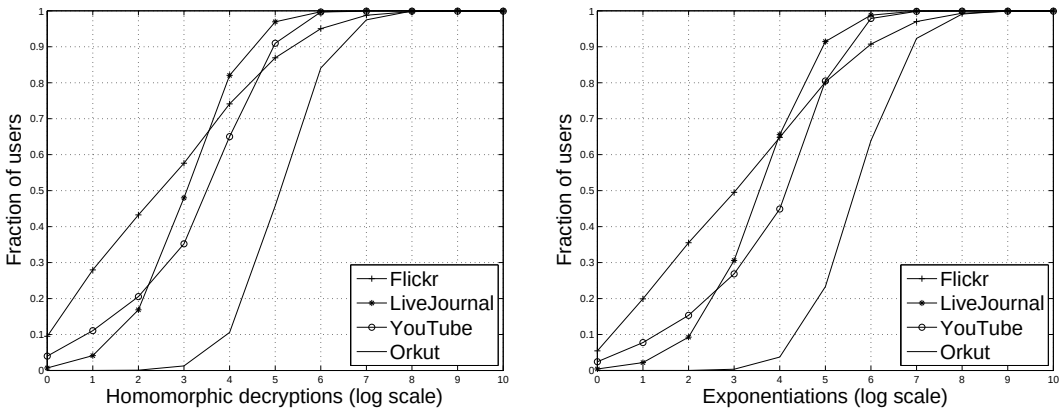


Fig. 5. Token flooding phase. Overhead per user in the number of hash values computed and exchanged.



(a) User u (b) User v

Fig. 6. Path Discovery phase. Computation overhead when user u discovers private relationship paths to user v.

7 Related work

In this section, we discuss decentralized social networks that were proposed. We then discuss previous schemes to discover relationship paths between users.

Several centralized social networking sites enable users to find relationship paths. For instance, LinkedIn, a professional social networking site, enables users to find private relationship paths to others. Unfortunately, these centralized sites know the entire topology and offer no privacy.

Decentralized social networks were proposed [1, 2] to circumvent the lack of interoperability among current social networking sites, where users store their profiles locally and directly communicate with their friends. We did not find any support for a discovery of private relationship paths in a fully decentralized manner in any of these works. A decentralized social network was also proposed by Popescu et al. [16] to resist government monitoring. The social network provided search capabilities for sensitive items, but unfortunately, discovery of private relationship paths is not supported.

The most closely related work is by Freedman and Nicolosi [7], and their preceding paper [9]. Techniques to verify social proximity between users are presented as a mechanism to whitelist emails from the social network of the recipient. The paper mainly focuses on verifying friend of friend relationships, i.e., a relationship path of distance 2, while only disclosing common friends to one party. The paper suggests an extension to check for longer relationship paths, but unfortunately, their extension discloses all the relationships on the relationship path at the time of verifying the social proximity.

Carminati et al. [5] propose techniques to discover relationship paths between users in a decentralized social network. The paper assumes an untrusted central node, and discloses to one party all the relationships on the discovered relationship paths. Domingo-Ferrer [6] propose a mechanism to discover private relationship paths in a decentralized social network. When u wishes to discover paths to v, u floods her social network at that time. A major issue of this approach is that the discovery to be arbitrarily delayed if intermediate users are offline. Moreover, after some time has elapsed, u cannot know whether there does not exist a relationship path, or simply that some intermediate user did not happen to be online.

Conclusion

Social networks are increasing in importance. The majority of current social networking sites rely on a centralized server, which unfortunately offer no privacy for users' sensitive data. Given the highly privacy-sensitive nature of social networking topology (i.e., friendship relationships), a challenge is how to construct privacy-preserving social networks that provide the ability to find relationship paths without disclosing superfluous relationships. We take this problem one step further and consider decentralized social networks, where users can discover relationship paths *offline* (in a privacy-preserving manner) with people they meet. Our proposed approach provides the property to users who casually meet to discover relationship paths without disclosing their private relationships. More efficient schemes are the subject of our future research.

ACKNOWLEDGMENT

This research was supported in part by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and MURI W 911 NF 0710287 from the Army Research Office, and by the iCAST project, National Science Council, Taiwan under the Grant NSC96-3114-P-001-002-Y. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, or the U.S. Government or any of its agencies.

We would like to express our gratitude to Ahren Studer and Bo-Yin Yang for useful feedback and interesting discussions. We would also like to thank Hazel Diana Mary for her help with the simulations.

References

1. foaf-o-matic. `http://www.foaf.com/`.
2. M. Ackermann, K. Hymon, B. Ludwig, and K. Wilhelm. Helloworld: An open source, distributed and secure social network. W3C Workshop on the Future of Social Networking, January 2009.
3. R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of ACM SIGMOD international conference on Management of data*, June 2003.
4. R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An online social network with user-defined privacy. In *Proceedings of ACM SIGCOMM*, August 2009.
5. B. Carminati, E. Ferrari, and A. Perego. Private relationships in social networks. In *International Conference on Data Engineering Workshops*, 2007.
6. J. Domingo-Ferrer. A public-key protocol for social networks with private relationships. In *Modeling Decisions for Artificial Intelligence*, 2007.
7. M. J. Freedman and A. Nicolosi. Efficient private techniques for verifying social proximity. In *Proceedings of International Workshop on Peer-to-Peer Systems*, 2007.
8. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proceedings of Advances in Cryptology – EUROCRYPT*, May 2004.
9. S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. RE: Reliable email. In *Symposium on Networked Systems Design and Implementation (NSDI)*, May 2006.
10. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of Advances in Cryptology – EUROCRYPT*, May 2008.
11. L. Kissner and D. Song. Privacy preserving set operations. In *Proceedings of Advances in Cryptology – CRYPTO*, 2005.
12. B. Laurie. Apres: A system for anonymous presence. `http://www.apache-ssl.org/apres.pdf/`, 2004.
13. M. Lucas and N. Borisov. Flybynight: mitigating the privacy risks of social networking. In *Proceedings of ACM workshop on Privacy in the Electronic Society (WPES)*, Oct. 2008.
14. A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the ACM SIGCOMM conference on Internet measurement (IMC)*, 2007.
15. A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, May 2009.
16. B. C. Popescu, B. Crispo, and A. S. Tanenbaum. Safe and private data sharing with turtle: Friends team-up and beat the system. In *Proc. 12th Cambridge International Workshop on Security Protocols*. Springer-Verlag, April 2004.