Anonymous Opinion Exchange over Untrusted Social Networks

Mouna Kacimi
Max Planck Institute for Informatics
Saarbrucken, Germany
mkacimi@mpi-inf.mpg.de

Stefano Ortolani
Vrije Universiteit
Amsterdam, The Netherlands
ortolani@few.vu.nl

Bruno Crispo
University of Trento
Trento, Italy
crispo@disi.unitn.it

ABSTRACT

Social networks are the fastest growing Internet applications. They offer the possibility to get in touch with current friends, discover where the old ones are, and make new ones. While these applications are a great enabler for our social life, they are also well known to fall short on privacy. The lack of adequate privacy enhancing technology is particularly important in these applications due to the nature of information they deal with, and the fact that many users are underage. This paper provides a contribution in this direction by presenting a protocol, tailored for social network applications, that allows users to ask and/or submit personal opinions while preserving their anonymity.

1. INTRODUCTION

Online social networks are one of the most successful stories in the life of Internet. For an increasing number of people, a big part of their social life is now entirely built and developed online. The most popular platforms claim tens of millions of users, where a good portion of them spend several hours a day using such platforms. Users can publish their personal content and share it with other people in the network. Moreover, they can create social relations with their friends (from real life), discover new people with common interests, express opinions and exchange comments about disparate subjects and topics. Examples of social network platforms include Facebook, MySpace, Flickr and Orkut.

The impact such networks may have on users' privacy raises serious concerns [16, 12]. Most users share and exchange private information without a clear understanding of the possible consequences. The reason is that users are not in control of where information will be stored and how it is used by other users and by the platform. Thus, privacy enhancing technologies (PETs), specifically tailored for online social networks, are sorely needed.

This paper presents a protocol that allows anonymous opinion exchange among users. Such anonymity is still needed today to avoid serious consequences of what people could say. For example, in countries with oppressive regimes, citizens are not free to express disagreement or criticism about their government. Thus, anonymity in such environments is an important step towards freedom of speech. Furthermore, by implementing anonymous opinion exchange, the protocol addresses the problem of the *spiral of silence.* [1]

Existing solutions [5, 6, 3] providing anonymous information sharing rely on a global knowledge of the network topology. This knowledge cannot be realistically assumed in online social network applications. Additionally, they require the requestor to know the responder in advance . For these reasons, existing solutions cannot be adopted to solve the problem presented in this paper. In social networks, users do not know the connectivity graph. Thus, the proposed protocol relies only on local knowledge for transmitting information. Additionally, since the protocol aims to preserve anonymity, the requestor does not know who the responder (if any) will be.

This paper proposes a new *friends-to-friends* delivery protocol allowing anonymous and private sharing of opinions between a large number of users. This mechanism is deployed (1) over an untrusted network and (2) in the presence of a central infrastructure, such as the servers of the online platform. To achieve anonymity of both requestors and responders, queries and answers are forwarded via trusted links (friends). Queries and answers travel hop-by-hop in the friendship graph. Each

[1]As the *spiral of silence* theory[11] asserts, a person belonging to the minority is less likely to voice an opinion on a topic for fear of reprisal or isolation from the majority.

intermediary node in the graph changes the requestor's address in the query to its own address before forwarding it to its friends. The same process is done for the answers. In this way, no user can know where queries and answers come from. The protocol guarantees responder's anonymity against any other user of the network and also against the central platform. By contrast, requestor's anonymity is protected against any other user of the network, but not against the platform that (potentially) can observe and store all the traffic of the network. Additionally, the privacy of queries and answers is preserved against users that are not traversed in the communication paths, and against the central platform.

The key innovation of the proposed protocol is that it relies on an existing real-world social network that captures real friendship relations. Relying on real friendships to exchange encryption keys and to deliver messages increases the robustness of the system against attacks. Since users are friends in real-world, they can exchange cryptographic keys by existing out-of-band channels, without the need to establish an additional secure infrastructure for this purpose.

Messages are transmitted via trusted friendship links. One way of violating the system is to inject an attacker along the message path. Attackers can easily create fake accounts, though they cannot create fake relations since they represent real friendships. It is also important to note that, using this protocol, the damage of a misbehaving friend is confined to a single hop and does not propagate beyond that.

The rest of the paper is organized as follows. Section 2 introduces the system model used for anonymous exchange of opinions in a social network. Section 3 explains the attack model. Section 4 describes the main concepts of the protocol. Section 5 presents some security analysis. Section 6 presents experiments with a real-world dataset to study the feasibility of the protocol. Section 7 presents related work and puts it into context and Section 8 concludes the paper.

2. THE SYSTEM MODEL

This section outlines the system model used to exchange private information in a social network. This model reflects, as much as possible, how social network platforms (e.g., Facebook) are used and implemented. The platform is considered to be logically centralized. Thus, it can monitor, store, block, inject and modify any message that goes through it. Users access the platform by means of a client (i.e., browser) that implements encryption and decryption.

Formally, the network of friends is modeled as a graph consisting of N nodes. Behind each node i there is a human user (the node's **owner**). The edges of the graph represent friendship relations between users' nodes. A user owning a node i is willing to share her opinions only with nodes owned by people she trusts - this is denoted as *i's* **friend subset** - F_i. It is assumed that the friendship relation is commutative; for any two nodes i and j, if i is in F_j, then j is in F_i. However, friendship is not transitive (the friend of a friend is not automatically a friend). Since these friendship relations exist in real life, it is assumed in this model that friends share pairwise symmetric keys exchanged via out-of-band mechanisms. The key is then inserted and stored locally in the client, and never shared with or exchanged using the platform.

The protocol, proposed in this paper, is envisaged to be used for exchanging textual queries and answers. Queries, submitted by users, concern some topics that are generally not openly discussed. Examples of queries are: *how is the working environment at Pyanair?* or *what is the salary of a junior programmer at Groogle?*. Upon receiving these questions, users can anonymously give answers to avoid bad consequences [18]. Examples of answers are: *As a flight attendant at Pyanair, I have a very busy schedule and I can be called in anytime. Moreover, the salary is one of the lowest compared to other airlines*, or *I work at Groogle and I get 60K Euro*

The maximum length l of queries and answers is fixed (i.e. 50Kbyte). A query/answer always fits into a logical block of size l before being encrypted. In case queries/answers are shorter, the block is filled with padding. This prevents well known attacks relying on the different lengths messages may have.

3. ATTACK MODEL

A successful attack manages to reveal the identity of a requestor and/or a responder. We will show that our protocol can protect the anonymity of both requestors and responders against any other user of the system. This includes friends involved in message forwarding and users who try to violate the system. Additionally, the protocol guarantees responders' anonymity against attacks mounted by the platform. However, it does not protect requestor's anonymity against the platform. Recall that the platform is a very powerful attacker since it knows all relations between all users, and it can monitor and store all exchanged messages. Moreover, it can create fake accounts; however, it cannot create fake relations since they are based on real friendships. The platform can mount denial of service attacks(DoS) by blocking messages to be delivered. Since these attacks do not break users' anonymity, protection against them is outside the scope of this paper.

The protocol assumes that the underlying cryptographic algorithm is secure. Encryption and decryption of messages are executed on the user's client and not on the platform. The security of such clients is assumed and it is beyond the scope of this paper. Recall that the main goal is to protect the anonymity of requestors and responders and not the confidentiality of the answers. Thus, it is worth mentioning that the protocol also provides confidentiality of the answers against the platform and against eavesdroppers (other users that are not in the path traversed by the query).

4. THE PROTOCOL

This section presents a protocol for anonymous opinion exchange in social networks. The first step toward ensuring privacy of communication is to encrypt messages exchanged by users. They can then be safely propagated via the network, and decrypted by the local clients. Each user can establish a cryptographically secure connection between its node and all the nodes in its friends' subset. Since the owners of the nodes are assumed to be friends in real life, the shared secrets needed to establish these secure connections can be agreed-upon by out-of-band means. Once established, the inter-friends secure communication links are used for exchanging opinions

Users are interested in getting opinions not only from their friends but also from other people in the network. Since secure connections can be established only between friends, all messages, including queries and answers, are propagated hop-by-hop from friend-to-friend. A node, receiving a message from one of its friends, changes the sender address in the message to its own address. In this way, each intermediary node does not know where queries and answers come from.

When a user submits a query to the network, she gets back a set of answers. To protect the anonymity of responders against external observers a message transmission mechanism based on the concept of k-anonymity [17], is proposed. For each query, the requestor creates a message that contains the query and k fake answers. This message will travel hop-by-hop via friendship links. Every node in the path that has an answer replaces a fake answer by a correct one. When the search is terminated the message travels back to the requestor. In this way, during its life time in the network, a message always contains k answers. At the beginning of the transmission all the k answers are fake, and at the end they can be all correct, all fake, or a mixture of both. Recall that queries and answers travel within a block of a fixed size, so an attacker cannot distinguish them or try to infer something based on their lengths.

As described previously, messages are propagated from friend-to-friend. The requestor node chooses one of its friend nodes to send the message to. When the friend's node receives the message, it chooses one of its friend nodes (except the one from which the message comes from) and forwards the message to it. The process is repeated until a given stopping condition is verified. To protect involved parties, friends to which messages are forwarded must be randomly chosen. The reason is that users involved in the transmission process do not know, and should not know, which user (or which path) can provide answers. The randomization of friends' selection calls for adopting the random walk strategy. The random walk is inherently a sequential process that visits the nodes of a graph one after the other in a random order. A nice property that makes the random walk appealing for this application is that it eventually visits all nodes of the graph [8]. Thus, from a given node i, any node j can (eventually) be reached with a lower overhead comparing to flooding-based protocols [14]. The standard random walk (with one walker) increases the user-perceived delay of successful searches by an order of magnitude [2]. Therefore, instead of sending just one query message, the requestor's node sends r messages, and each message takes its own random walk. The walkers are independent; they may take the same path multiple times, which increases the number of duplicated messages in the network. To solve this problem, a *state keeping* mechanism is used. *State keeping* consists of keeping track of the friends that has been selected by other random walkers of the same query. When a node receives a message, it selects (if possible) a friend among the ones which have never been selected by other walkers. The state keeping mechanism reduces the likelihood that a random walk traverses the same path twice. Note that state information is kept by the nodes for a very short time.

Four main components of the protocol were described: *encryption*, *hop-by-hop communication*, *k - anonymity* and *random walks*. Based on these concepts, the protocol for one walker is described in the following paragraphs. The same process is replicated for r walkers.

• The user starts by initiating a query. Then, the user's node constructs a message containing the query and a set of k fake answers of the same size.

• When the node generates the message it starts a friend discovery phase during which, it discovers online friends. From the set of online friends, the node randomly chooses one friend to which it forwards the message after having encrypted it with the key it shares with that friend. When the message arrives at a particular node, it is first decrypted and then the following cases are distinguished:

1. If the node has an answer to the query and there is at least one fake answer in the received message, it adds its correct answer by replacing a fake answer. Then, it forwards the message to one of its friends. A node which adds a correct answer in the message needs to forward it to protect its anonymity.

2. If the node has an answer to the query and there is no fake answer in the received message, this means that the maximum capacity of the message has been reached. Therefore, the node sends back the message to the requestor.

3. If the node has no answer and there is no correct answer in the received message, it always forwards it. To avoid infinite walking, in case no answers can be found in the path, a TTL is taken into account only in this case. If the message does not have a correct answer and its TTL is 0, each node that receives it decides either to drop the message or forward it to the next friend with the same probability. To avoid some possible attacks there is no fixed or globally agreed TTL, but rather the TTL is randomly chosen by the requestor.

4. If the node has no answer and there is at least one correct answer in the received message, it can either send back the message to the requestor, or forward it to the next friend with the same probability.

• Each node maintains a query table with queries it has forwarded but for which the query answer process has not yet completed. Each query table entry maintains information about the node which has forwarded the query message to the current node. This information is used to return the message to the requestor. Note that in order to be able to recognize correct answers, fake answers have an agreed format (e.g., a sequence of all 1's).

Figure 1 shows an example of a user A sending a message with 4 fake answers. The user A sends the message to his friend B. User B adds a correct answer in the message and forwards it to C. User C also adds a correct answer in the message and forwards it to D. User D does not have an answer and decides to forward the message to E. User E does not have an answer and decides to send the message back to user A using the hop-by-hop mechanism. User A gets back the message with 4 answers: 2 correct ones and 2 fake ones.

5. SECURITY ANALYSIS

5.1 Protecting from friends

Attacks from friends are actually the most serious attacks for the proposed application scenario, since users will be very reluctant to use the protocol if it fails to protect their anonymity against their close friends. To protect requestor anonymity, the TTL is randomly chosen in the protocol. Thus, when the requestor sends a message to the node of its first hop friend, the node cannot distinguish if (1) it received the message from the requestor or (2) it received an already exiting message

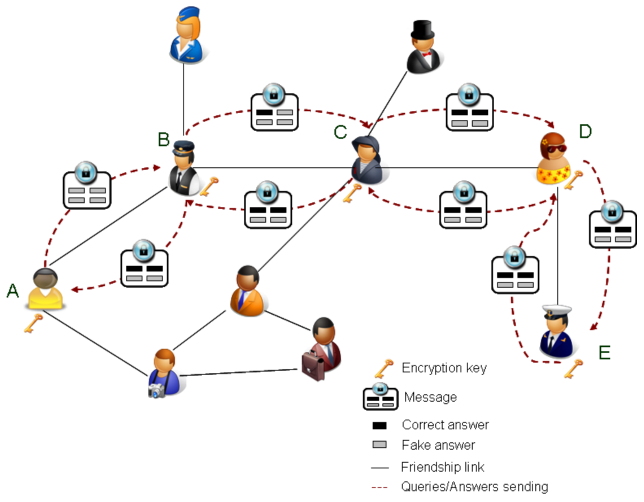

Figure 1: Asking anonymous opinions in a social network)

with no correct answers and with TTL different from zero. To protect the responder anonymity, a node that inserts a correct answer does not directly send back the message. In this way, the attack where a friend can deduce that the next node is actually a responder is prevented. This attack consists of forwarding a message containing n correct answers and receiving back from the node to which the message has been forwarded a message with $n + 1$ correct answers. In this case, the friend can deduce that this node is actually a responder. This attack is prevented when the responder always forwards the message to the next hop after it inserted a correct answer. Since the next hop node will choose, with probability of 50% , to send back or forward the message further, the attacker, in the best case, has no more than 50% chance of guessing the right responder. Thus, the protocol offers plausible deniability.

5.2 Protecting from the platform

Suppose the social platform wants to identify at least one of the users who answered a query. Recall that the query is sent through the network using a message, for each walker, containing the query and k answers of the same length. At the beginning of the query life the answers are fake. Then, they can be replaced by correct ones when the message travels. As the platform is a global observer, it can gather all messages and subsequently try to identify which user answered the query by means of comparing them all. As the protocol encrypts each message in a hop-by-hop manner, the platform cannot read them. The platform could also apply some traffic analysis based on messages lengths. However, because messages with queries and answers (fake or correct) have all the same fixed length, the platform cannot identify which message is which. Therefore no

Table 1: Key statistics of the Flickr social network used for the evaluation

metric	value
#nodes	2841
total #edges	23153
avg. node degree	16.2991
std. dev	22.9511
max degree	340
median degree	7.0

user can be accused by the platform of being a responder.

Another possible attack, carried out by the platform, would be to monitor encrypted messages since they are the suspicious ones. The platform considers, as a responder, each node that starts sending back a message toward the requestor. The protocol protects against this type of attack (1) by not using any fixed TTLs and (2) by using a termination algorithm that introduces uncertainty on how far is the responder from the node that starts sending back answers (hop-by-hop) to the requestor. The protocol does not protect requestor anonymity from the platform. The platform can monitor all messages and perform traffic analysis. Thus, it can identify the requestor by observing which node starts a new encrypted thread.

5.3 Protecting from other users

Protection of the anonymity of responders against other users that are not traversed is guaranteed by the same arguments used for protecting against the platform. As described in the previous section, the anonymity of requestors can be broken only by performing traffic analysis. Since normal users cannot monitor messages exchanged by other users they cannot break, requestors' anonymity.

5.4 Confidentiality of queries and answers

The confidentiality of each message (queries and answers) against possible attacks from the platform and from eavesdroppers (users that are not traversed by the query) is preserved by means of encryption. Each message is encrypte, on the local client with different keys at each hop. As it is assumed that a friend-to-friend relation is symmetric, each pair of friends F_1 and F_2 can trivially share a secret key K_{F_1,F_2}. The two primitives $Encrypt_{K_{F_1,F_2}}$ and $Decrypt_{K_{F_1,F_2}}$ are then implemented by any sufficiently robust symmetric cipher.

6. EVALUATION

One of the main assumptions of this approach is that for sufficiently large social communities, friendship relationships form a connected graph. We evaluated the performance of the proposed approach on a dataset

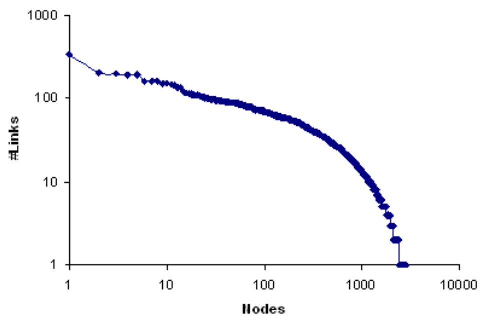

Figure 2: Distribution of node degrees

crawled using snowball sampling from the Flickr social network website (http://flickr.com/). The crawled network has a total of 2,841 users and 23,153 symmetric friendship connections. The resulting graph is connected where any node i can be reached from any other node j. As shown in Figure 2, the node degrees of the *Flickr* graph follow roughly a power-law distribution. Additional statistics about this graph are shown in table 1.

A user submits questions to people via her friendship relations. Answers to those questions are anonymous. They represent personal opinions about the query subject. Consider S as the set of subjects for which users ask questions. For each subject S_i, there are U users in the network that know the subject and $\alpha\%$ of them are willing to give opinions. Considering that these subjects are private, the group of related users are not explicitly defined. Note that users belonging to the same group are not necessarily friends.

Since the proposed protocol deals with private information, it is very hard to make assumptions on the distribution of user groups, subjects, and opinions. For this reason, the focus in this evaluation is on a case where realistic assumptions can be made. To this end, consider the type of questions that ask opinions about people in the network. The group of users that may give opinions about a given person A are naturally her friends. This can also be extended to acquaintances (friends of friends) by assuming that opinions can also be given by people who are two hops far from A. Note that other users who are neither friends nor acquaintances of person A can also have opinions built from previous searches or experiences.

In this evaluation, 100 questions about people in the social graph and random requestors have been generated. The queries were run with different configurations of the protocol to measure its performance. Particularly, its cost and the rate of successful queries. This two measures have been computed while varying (1) the number of walkers r initiated by the requestor (2) the percentage α of users who are willing to give opinions.

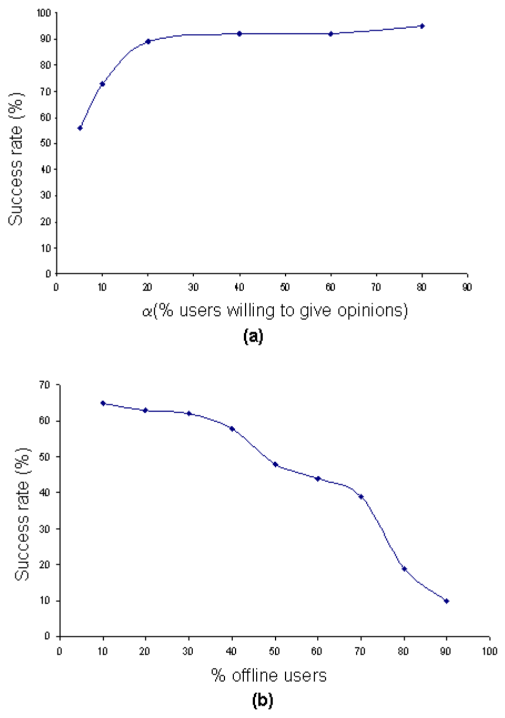

Figure 3: Success rate

A query is classified as successful if the requestor gets at least m answers for that query. The parameter m depends on the user's needs. In this evaluation, m is set to 5 answers. Note that, for a given query, m is set to the total number of existing answers in the network in case it is less than 5.

The results in table 2 show the protocol cost and the rate of successful queries depending on the number of walkers. They show that with only 5% of people giving opinions there is a high rate of successful queries when the number of walkers $r \geq 16$. Note also that this rate increases with the number of walkers. At the same time, a high number of walkers increases the network overhead in terms of number of messages and hops. Figure 3-a shows the rate of successful searches for different values of α. The results indicate that with 16 walkers and only 5% of people who are willing to give opinions, 56% of queries can be answered successfully.

In social networks people are not always online. This behavior can have an impact on the effectiveness of the protocol. We have evaluated the protocol in a dynamic environment where we have measured the rate of successful queries in the presence of churn. Figure 3-b shows that when using 32 walkers there is a reasonable rate of successful queries of 48% when only 5% of people are willing to give opinions and 50% random users (including those who might give opinions) are offline. These results show that the protocol is feasible in dynamic social networks.

7. RELATED WORK

Gross et al. [7] studied privacy in social networks. They showed how users often underestimate the disclosure of privacy by using such platforms. Several works tried to assess the privacy disclosure of different activities performed on platforms like Facebook. As messaging is one of the most common activities [4], Lucas et al. [9] proposed a system to mitigate the privacy risks of sending messages among a group of users. They propose an application tailored for Facebook where, by means of an external platform, encrypted messages could be delivered between users. However, they do not address the problem of preserving the anonymity of users within social network platforms. To tackle this problem, the protocol, presented in this paper, applies the concept of responder k-anonymity where an adversary can only narrow the list of suspects down to a set of k users. Moreover, external resources that users have to trust are not involved.

Manna et al. [10] use friendship relations to define access lists on published data. Popescu et al. [13] propose *Turtle* where they exploit friend-to-friend relations to share private data on a peer-to-peer network. Turtle ensures requestor and responder anonymity w.r.t. a local eavesdropper. It relies on trusted friendship relations to exchange symmetric keys for message encryption. The approach takes the main idea of friend-to-friend relation and extend the query protocol by means of random walks. As Turtle uses flooding to spread the query across the graph, the solution proposed in this paper significantly improves the efficiency of the protocol; moreover responder's anonymity w.r.t. a global eavesdropper is provided.

The concept of requestor's anonymity was first introduced in Crowds [15] where groups of users collectively issue requests on behalf of its members. The originator of the message is then disguised. Later on, Ahn et al. [1] introduced the concept of requestor and responder k-anononymity[2] to provide protection against a global eavesdropper. Dingledine et al. [3] propose Tor, a circuit-based low-latency anonymous communication service. The protocol requires users to establish end-to-end circuit where the requestor knows the responder in advance. This mechanism cannot be used in the setting of this paper since the requestor should not know who the responder is.

8. CONCLUSION

This paper introduces a protocol to provide anonymous exchange of opinions over untrusted social networks. The protocol is based on a *friend-to-friend* delivery mechanism allowing private and secure sharing of

[2]The concept of k-anonymity was originally formulated by Sweeney in [17]

Table 2: Protocol Performances ($\alpha = 5\%$)

metric	$r = 2$	$r = 4$	$r = 8$	$r = 16$	$r = 32$	$r = 64$	$r = 128$	$r = 256$
# msgs	96.81	211.18	458.63	951.99	2215.64	5268.35	12227.83	27185.54
avg # msgs/node	1.03	1.10	1.23	1.46	2.02	3.23	5.73	10.65
% visited nodes	3.19%	6.45%	12.42%	21.67%	36.83%	55.80%	74.37%	89.58%
avg # answers	1.17	2.1	4.01	7.29	12.35	17.58	20.77	22.09
% successful queries	8%	10%	59%	87%	94%	99%	100%	100%

sensitive information between a large number of users.

The main adversary of this protocol can be the social network platform itself since it has global knowledge about the network traffic. This paper has focused on how to protect responders' anonymity against the platform. In future work the problem of protecting requestors' anonymity from the platform as well will be addressed.

9. REFERENCES

[1] L. Ahn, A. Bortz, and N. Hopper. k-anonymous message transmission. *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, Oct 2003.

[2] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *The ACM SIGCOMM02 Conference*, pages 177–190, August 2002.

[3] R. Dingledine and P. S. N. Mathewson. Tor: The second-generation onion router. *In Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[4] A. Felt and D. Evans. Privacy protection for social networking apis. *W2SP 2008: Web 2.0 Security and Privacy 2008*, Jan 2008.

[5] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. *In Proceedings of the 9th ACM Conference on Computer and Communications Security , year=2000.*

[6] N. M. G. Danezis, R. Dingledine. Mixminion: Design of a type iii anonymous remailer protocol. *In Proceedings of the IEEE Symposium on Security and Privacy*, 2003.

[7] R. Gross and A. Acquisti. Information revelation and privacy in online social networks (the facebook case). *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[8] L. Lov and O. P. Erdos. Random walks on graphs: A survey. *In Eighty, Vol. 2*, 1993.

[9] M. Lucas and N. Borisov. Flybynight: mitigating the privacy risks of social networking. *Proceedings of the 7th ACM workshop on Privacy in the electronic society table of contents.*, Jan 2008.

[10] M. Mannan and P. Oorschot. Privacy-enhanced sharing of personal content on the web. *WWW '08: Proceeding of the 17th international conference on World Wide Web*, Apr 2008.

[11] Noelle-Neumann. The spiral of silence: a theory of public opinion. *Journal of Communication. 24, 43-51*, 1974.

[12] G. P. of Jaiku Raises New Privacy Issues. http://www.nytimes.com/2007/10/22/technology/22wireless.html, 2007.

[13] B. Popescu, B. Crispo, and A. Tanenbaum. Safe and private data sharing with turtle: Friends team-up and beat the system. *Proc. of the 12th Cambridge Intl. Workshop on Security Protocols*, 2004.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker. Search and replication in unstructured peer-to-peer networks. *In Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2002.

[15] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[16] N.Y.T. One Friend Facebook Hasnt Made Yet: Privacy Rights. http://www.nytimes.com/2008/02/18/opinion/18mon4.html, 2008.

[17] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Internation Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):571–588, 2002.

[18] Telegraph.co.uk. Office worker sacked for branding work boring on facebook http://www.telegraph.co.uk/scienceandtechnology/technology/facebook/4838076/office-worker-sacked-for-branding-work-boring-on-facebook.html,
2009.