2012 32nd IEEE International Conference on Distributed Computing Systems

Robust overlays for privacy-preserving data dissemination over a social graph

Abhishek Singh, Guido Urdaneta
University of Oslo, Norway
{*abhi,guidoau*}*@ifi.uio.no*

Maarten van Steen
VU University Amsterdam, The Netherlands
The Network Institute and Department of Informatics
steen@cs.vu.nl

Roman Vitenberg
University of Oslo, Norway
romanvi@ifi.uio.no

***Abstract*—A number of recently proposed systems provide secure and privacy-preserving data dissemination by leveraging pre-existing social trust relations and effectively mapping them into communication links. However, as we show in this paper, the underlying trust graph may not be optimal as a communication overlay. It has relatively long path lengths and it can be easily partitioned in scenarios where users are unavailable for a fraction of time.**

Following this observation, we present a method for improving the robustness of trust-based overlays. Essentially, we start with an overlay derived from the trust graph and evolve it in a privacy-preserving fashion into one that lends itself to data dissemination. The experimental evaluation shows that our approach leads to overlays that are significantly more robust under churn, and exhibit lower path lengths than the underlying trust graph.

***Keywords*-privacy; online social networks; peer-to-peer.**

I. Introduction

The Internet is being revolutionized by the interest-based data exchange within social communities. Millions of users in free online social networks (OSNs) are producing staggering volumes of data: Facebook receives more than 30 billion shared items every month while Twitter receives more than 55 million tweets each day.

However, numerous incidents indicate that free centralized services cannot be trusted to safeguard sensitive OSN data. As observed in [1], concentrating the personal data of hundreds of millions of users under a single administrative domain leaves users vulnerable to large-scale privacy violations via inadvertent disclosures and malicious attacks. Furthermore, OSN terms of service often give the provider the right to reuse users' data in any way the provider sees fit.

Consider a worldwide community of patients with the same chronic illness trying to support each other with information, job seekers in a particular professional area or geographical region mutually sharing advices, or a group of dissidents in a country that limits freedom of expression attempting to reach out to a broader audience. It is of utmost importance for users in such groups to be able to exchange data in a privacy-preserving fashion, that is, without disclosing their identity, personal details of their profile, or social friendship relations between the users. Unfortunately, existing free OSNs or generally speaking, architectures with central entities that can be compromised do not endow such groups with a viable solution.

Decentralized friend-to-friend (F2F) networks such as Freenet [2] pose a strong potential for catering to the needs of such groups. These networks leverage the existence of the underlying social friendship relations between the users and effectively map them into communication links. While direct data exchange occurs between friends only, a user A may relay messages between her friends B and C that are not friends of each other. Since in many cases, the social graph corresponding to a large community of users is connected, this mechanism allows for dissemination in large groups. Social relations provide additional incentives for the nodes to contribute their resources and participate in the dissemination. Unlike other kinds of private P2P networks, users in a F2F network cannot find out who else is participating beyond their own circle of friends, so that F2F networks can grow in size without compromising their users' anonymity. This also mitigates the harm to privacy if one of the user nodes is hijacked or willingly relaying information to a third party.

However, dissemination over a communication graph that simply mimics the social one is not very effective, as we show in this work. Although the social graph is connected, its connectivity is much weaker compared to random graphs with the same number of nodes and edges. Even moderate churn typical in online P2P networks results in degraded connectivity and significant graph partitioning, not to mention the effect of further decreased node availability if some of the users are using mobile devices. Moreover, existing social graphs exhibit a higher diameter than random graphs of the same size, which leads to slower data propagation.

The gist of our approach in this work is to bootstrap the communication overlay using the social graph but augment it with additional links between pairs of nodes that correspond to the users not related by social ties, as shown in Figure 1. These additional links should provide an abstraction of privacy-preserving routing between pairs of *untrusted* nodes that cannot trust each other when it comes to anonymity or even learn of each other's identity. They are maintained in presence of churn in such a way that (a) sending data over these links does not violate privacy of the nodes, (b) our mechanism for establishing these links is itself privacy

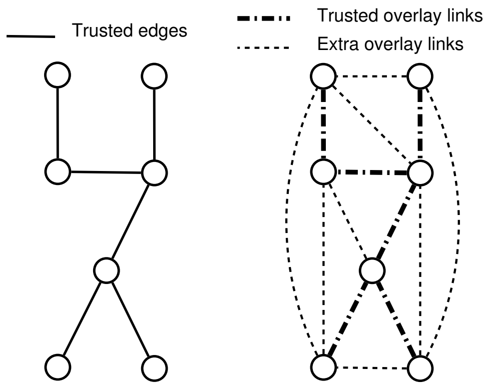

Figure 1: Example of a trust graph and derived communication overlay. Note that the overlay has links that correspond to edges in the trust graph, and extra links that help make the overlay more robust.

preserving and it does not impose significant overhead, and (c) the resulting overlay becomes robust under churn as well as having a low diameter. This way, we retain the above mentioned advantages of F2F networks while enhancing robustness of the overlay. For example, it is possible to leverage such an overlay towards an efficient and scalable implementation of reliable and privacy-preserving message broadcast by using controlled flooding, epidemic dissemination, or an additional routing layer. In essence, this will present a highly sought solution for the scenarios of privacy-preserving data exchange within communities, as presented above.

To this end, we propose a service for overlay maintenance and privacy-preserving routing between pairs of untrusted nodes. The service utilizes the idea of node pseudonyms, which serve the role of an anonymous address for the node. A major building block in our design is a pseudonym-based routing facility that is responsible for generating pseudonyms. When the pseudonym of node A reaches a node B not trusted by A, the facility allows B to communicate with A in a privacy-preserving fashion.

Our overlay maintenance service distributes pseudonyms using a privacy-preserving gossiping protocol. Each node only uses a carefully selected small subset of all received pseudonyms so that the overlay has a limited fan-out and the distribution of links approximates that of a random graph. Finally, our service periodically renews pseudonyms in order to further enhance privacy. This is implemented by a TTL-style mechanism with expiration values carefully chosen so as to provide better privacy without hurting the robustness of the overlay under a typical churn.

We analyze the privacy protection that our service provides under a variety of threat scenarios. In particular, system nodes may play the role of internal observers while providers of a communication infrastructure (e.g., an ISP) may act as external observers. Both types of observers may spy upon the nodes in the system. Furthermore, internal observers may collude in an attempt to collectively gain knowledge about the identities and relations beyond the sum of knowledge that is available to each of them. We conclude that our service is not vulnerable to obvious attacks and is able to satisfy our privacy requirements under several reasonable threat models. Furthermore, using the service does not worsen privacy protection compared to existing F2F systems that only utilize links between trusted nodes.

We experimentally evaluate a number of aspects of our service: (a) the characteristics of the resulting overlay, (b) the overhead of overlay maintenance, and (c) the speed of convergence towards a robust overlay. We also conduct a sensitivity analysis wrt a number of settings affecting the execution of different protocols within our service. The evaluation is carried out using a number of different social graphs sampled from the Facebook's social graph. The results show that our service efficiently generates robust overlays while incurring acceptable overhead.

II. Definitions and goals

Our main goal is to produce an overlay protocol that facilitates robust privacy-preserving data dissemination for a group of users that have trust relationships among them. Our idea is that this overlay can serve as a substrate on top of which high-level social applications such as micro-news, mailing lists and group chat can be built. In this section we attempt to present in precise terms what this problem entails.

A. System model

The system consists of a set U of nodes, each node being managed by a unique user. We do not consider direct communication channels between the nodes because in order to allow for privacy-preserving data exchange, a viable solution may have the nodes communicating through available third-party services. Instead, we say that the nodes are connected to the Internet, possibly intermittently as detailed in Section II-D. Nondisclosure of the nodes in U is one of the central privacy-preservation requirements, as specified in Section II-C.

B. Trust graph

A group of users participating in a privacy-preserving data-dissemination application can be modeled as a graph where each vertex represents a node, and each edge represents a trust relationship (e.g., friendship) between a pair of users. In the context of the problem we consider, trust between two users a and b refers to the fact that a and b can rely on each other not to violate the privacy of their mutual identities or communication as detailed in Section II-C. In practice, such a trust graph may correspond to existing large-scale social graphs of users in a particular social network such as Facebook, or community, such as the community of patients with the same chronic disease.

Trust relationships in our system are symmetric, but not transitive. That is, if a is b's friend, then b is a's friend; but if a and b trust each other and b and c trust each other, that does not imply that a and c trust each other. For simplicity, we assume that the trust graph is connected. If it is not, then the problem of privacy preserving dissemination would apply to each of the connected subgraphs separately.

In practice, the trust graph may evolve over time due to addition of new nodes as well as addition or removal of trust relations. While addition of nodes or edges does not raise privacy concerns, it is challenging to define meaningful privacy requirements when trust is revoked and edges are removed from the trust graph. In this paper we only consider trust graphs that do not change over time, and leave the study of mutable trust graphs as future work.

In the rest of the paper, we refer to a pair of nodes controlled by users with a trust edge between them as *trusted nodes*. For succinctness, we also say that such nodes trust each other, or that they are *trusted peers* of each other.

C. Privacy

Our most important privacy requirement is to prevent the disclosure of the list U of participating nodes and their corresponding users. We assume that, initially, each node is configured with the list of its trusted peers, and that the only knowledge each node has about U is that its trusted peers are part of U. Furthermore, we assume that no node will intentionally disclose the participation of any of its trusted peers in the system to any third party. We also assume that no other entity has any knowledge about U when the system starts.

The second privacy requirement is to guarantee nondisclosure of the edges of the trust graph, which correspond to relations between the users controlling the nodes. For example, if a trusts b and c, a should not be able to establish whether b and c trust each other.

Another important privacy requirement refers to the security of the data disseminated within the overlay. For example, disseminated data should be readable only by the members of U. Since we focus on overlay maintenance in this work, application-specific protocols for disseminating application data are beyond the scope of this paper.

D. Failure model

We assume that the nodes can leave (e.g., voluntarily or by crashing) and rejoin the system at any time. Nodes can also get temporarily disconnected from the rest of the system (e.g., due to a failed Internet connection).

When a node comes online for the first time, the only information it possesses is the set of its adjacent nodes in the trust graph. (This information is initially obtained by the node through means of communication external to the system.) When a node rejoins the system, it retains the state data that it had prior to the failure or disconnection.

We assume that all the nodes follow the proposed communication protocols correctly due to the shared incentive of achieving privacy-preserving data dissemination. In particular, they do not actively try to disrupt the robustness of the system. However, system nodes may play the role of internal observers while providers of a communication infrastructure (e.g., an ISP) may act as external observers. Both external and internal observers may apply passive techniques such as traffic analysis [3] with the purpose of gaining knowledge about U. Furthermore, internal observers may collude in an attempt to collectively gain knowledge about U beyond what is available to both of them. We assume that the application disseminating the data using overlay links encrypts the messages so that external observers cannot read their content.

E. Problem statement

The problem we need to solve is as follows: Given a set U of nodes, and a trust graph that represents trust relationships among the users controlling such nodes, find a protocol for building and maintaining an overlay network that satisfies three main properties:

- *Privacy preserving overlay links:* The links created by the protocol should be privacy-preserving so that the application disseminating the data over these links would not be disclosing node identities or relations as defined in Section II-C.
- *Privacy preserving overlay maintenance:* The protocol for overlay maintenance should not be disclosing node identities or relations as defined in Section II-C.
- *Robustness:* The overlay remains connected and has relatively short path lengths in presence of realistic node churn. More precisely, our main goal with regard to robustness is to minimize the probability that the overlay gets partitioned due to offline nodes. If partitioning does occur, we want to minimize the number of online nodes that get disconnected from the largest connected component of the overlay.

Observe that we cannot use a centralized node directory service in our solution because the latter can be compromised (consider data leaks from Facebook or other social networking sites). The role of the trust graph in the problem definition is that the mutual knowledge of the neighbors in the trust graph is an appealing way of bootstrapping overlay links between the nodes that would otherwise be unable to learn of each other.

The immediate impediment on the way to solving the problem is that a direct communication channel between any pair of trusted nodes might be monitored by a passive external observer (e.g., an ISP). A naive data exchange over these channels may reveal both the identities of the nodes and the fact that there is a trust relation between them. Therefore, it is necessary to create an indirect communication link that prevents observers from learning that the trusted nodes are exchanging data in the context of our application.

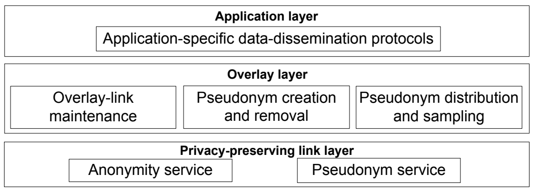
Application layer
Application-specific data-dissemination protocols
Overlay layer
Overlay-link maintenance
Pseudonym creation and removal
Pseudonym distribution and sampling
Privacy-preserving link layer
Anonymity service
Pseudonym service

Figure 2: Architecture for privacy-preserving data dissemination

Fortunately, this particular challenge can be resolved by existing anonymity systems [4].

However, establishing overlay links between trusted nodes is not enough to make the overlay robust. As we show in our experimental evaluation in Section V, in typical trust graphs, a significant fraction of the online participating nodes may get disconnected from the largest connected component of the overlay even for moderate node churn. Furthermore, we also show that path lengths in existing trust graphs are substantially longer compared to random graphs of the same size and average fan-out.

Hence, we also have to create overlay links between untrusted nodes (i.e., not connected by an edge in the trust graph). In this case, the link must not only protect against external observers, but it must also prevent both nodes from learning each other's IDs. Creation and maintenance of such links is the main focus of our work.

III. Solution

A. Overview

Our solution for privacy-preserving data dissemination leverages trust relationships among users and aims to improve the robustness of a trust-based overlay by augmenting it with additional links, such that the augmented graph has properties more similar to those of random graphs. Random graphs are known to exhibit good failure resilience and short path lengths.

Figure 2 shows the general architecture of our approach. The lowest-level layer of our architecture is a *privacy-preserving link layer* consisting of an *anonymity service* and a *pseudonym service*.

The anonymity service allows any node to create privacy-preserving links to any destination whose ID is known. Privacy-preserving links allow nodes to exchange messages such that external observers monitoring communication channels are unable to discover that either node participates in the system. Application using our service and sending messages through privacy-preserving links must employ end-to-end encryption in order to prevent potential observers from reading message content. The main use for the anonymity service in our proposed solution is to establish links between trusted nodes, which know each other's IDs. Each node n uses this service to establish links with its adjacent nodes in the trust graph when n comes online. Henceforward, we refer to the set of privacy-preserving links built in this way as the *trusted links* of n.

Alongside the anonymity service, we also employ a pseudonym service, which allows any node to create pseudonyms and to establish privacy-preserving links to any destination for which a pseudonym is known. We define a pseudonym $P(n)$ of a node n as an address that any other node m can use in conjunction with the pseudonym service to build a link to n such that n's ID is not disclosed to m and vice versa, and that the participation of both m and n in the system remains undisclosed to external observers. We use the pseudonym service in our solution to establish links between nodes that do not trust each other and hence must not be able to learn each other's IDs. For any node n, we refer to the set of privacy-preserving links established by n using the pseudonym service as the *pseudonym links* of n.

The set of *overlay links* of a node n (denoted n.*links*) is the union of its *trusted links* and *pseudonym links*. Note that when a node rejoins the system after becoming offline temporarily, it re-establishes the overlay links it had before becoming offline. Similarly, overlay links to nodes that go offline are not removed. Such links become operational again when the corresponding nodes rejoin the system. In essence, individual links in our system do not provide strong reliability guarantees; it is the redundant connectivity due to the collection of the links that makes the overlay robust.

The anonymity and pseudonym services can be deployed on either participating nodes or a third-party infrastructure with a higher availability. In Sections III-B we discuss how these services can be realized.

The next layer in our architecture, which we call the *overlay layer*, is responsible for the creation and maintenance of overlay links and for providing upper layers with methods to use those links to implement data-dissemination protocols.

Maintaining trusted links is a straightforward operation. Initially, every node knows the IDs of its neighbors in the trust graph and can therefore use the anonymity service to establish trusted links. On the other hand, nodes initially have no knowledge about pseudonyms, and thus cannot readily create pseudonym links. To solve this problem, the overlay layer executes a maintenance protocol that takes care of the creation and removal of pseudonyms, the distribution of pseudonyms across the overlay, and the addition and removal of pseudonym links such that the resulting overlay resembles a random graph.

For pseudonym creation and removal we use a scheme based on giving pseudonyms a limited lifetime. We describe the scheme in Section III-C. We distribute pseudonyms across the overlay using a gossiping algorithm. Each node applies a sampling mechanism over the pseudonyms that are distributed to the node in order to decide which pseudonym links the node should have. We describe our protocol for pseudonym distribution and sampling in Section III-D. In Section III-E we discuss how our overlay maintenance protocol preserves privacy under various threat models.

B. Anonymity and pseudonym services

The anonymity service can be realized using existing solutions based on the concept of mix networks [5]. Mix networks allow the implementation of privacy-preserving links between two nodes by employing a set of relay nodes. The sender node applies multiple layers of encryption to the message and sends it to the first relay in the mix, each relay removes one encryption layer and passes the message to the next relay until it finally reaches the destination. Alternatively, each relay may add an encryption layer, and the destination can apply all the decryption operations. The latter scheme is commonly used to send responses back to the original sender in a data exchange.

In a mix network, no node knows its position in the chain, so they cannot know if the previous node is the sender or another relay, or if the next node is the destination or another relay. It is also difficult for an external observer who can monitor communication channels to associate the sender with the receiver. A recent survey [4] describes in great detail the state of the art in mix-based anonymity systems.

In practice, higher-level application requirements will dictate the type of anonymity service to use, with a trade-off between latency requirements and threat models for external observers being a decisive factor. Some anonymity services specialize in low-latency [6], [7] applications such as chat or web browsing, at the cost of supporting weaker threat models; while systems supporting stronger threat models [8], are practical only for noninteractive higher-latency applications such as email. The reason is that timing analysis techniques [9] can be much more effective when a message needs to move quickly through a mix network in order to achieve low latency. High-latency systems, on the other hand, can apply additional mechanisms to defend against such techniques, such as introducing random delays.

Pseudonym services can generally be realized with the help of an anonymity service. A few deployed anonymity services have a stub for this extra functionality built in. Examples are I2P's "eepsites" [7] and Tor's "hidden services" [6], in which a node n wishing to be contacted establishes a mix network, with the address of the last relay acting as a pseudonym. A node wishing to contact n can send a request to the last relay (i.e., the pseudonym) and negotiate a separate mix for further communication.

Another possible way to realize a pseudonym service is to use the anonymity service together with a third-party distributed storage service (e.g., email or a DHT). In this case, pseudonyms would be storage-service addresses (e.g., email addresses or DHT IDs). Under this approach, a sender node m can send a message to a receiver node n by storing data at the appropriate pseudonym address, and the receiver n can obtain new messages by regularly polling the storage service. Both n and m need to access the storage system through the anonymity service in order to protect their IDs.

Existing anonymity services are known to provide high availability [6]. When considering the overlay robustness, we assume that a link established by these services always remains operational provided that both nodes are online.

C. Pseudonym creation and lifetime

Every node creates a pseudonym to represent itself when it starts. Pseudonyms always have a limited lifetime, so that whenever a pseudonym expires all pseudonym links involving the expired pseudonym are removed from the overlay. Therefore, every node must periodically create a new pseudonym.

The pseudonym service guarantees that the pseudonym of a node remains valid even if the node goes offline and comes back, provided that the pseudonym has not expired.

We believe that having ephemeral pseudonyms can help improve the privacy of our system against certain types of external observers. Intuitively, an observer who can monitor traffic corresponding to a single pseudonym link will gather only a limited amount of data for traffic analysis. In order to gather data corresponding to a specific node for a long time, the observer will need to be able to monitor many more communication channels. Moreover, even if the observer is powerful enough to monitor all required channels, its traffic-analysis problems might be more difficult to solve, as it will be necessary to detect that messages flowing through various pseudonym links at various points in time correspond to the same node.

Ephemeral pseudonyms can also make it easier to defend against replay attacks, where the same message is sent repeatedly to a pseudonym in order to, for example, discover repetition patterns at the end of the mix network used for implementing the link. A common technique that mix relays can apply to prevent replay attacks is to remember hashes of all messages that have been relayed to each pseudonym, and drop repeated messages. If pseudonyms expire, then the space requirements for the effective implementation of this defense become bounded for each pseudonym.

Ephemeral pseudonyms can also improve the quality of the overlay in the case when a node goes offline permanently or for a long time. In this case, all pseudonym links involving the offline node will eventually be removed after the corresponding pseudonyms expire. In our experimental evaluation in Section V, we analyze the effect of the pseudonym lifetime on the quality of the overlay.

Pseudonym lifetime is an important system parameter and it must be set such that it is longer than the time nodes are expected to be offline before rejoining the system. For example, in an application where nodes are expected to be offline for 8 hours before rejoining the overlay, it would be reasonable to have pseudonym lifetimes of, for example, 24 hours. The reason is that when a node rejoins the system, it is important that it has some valid pseudonym links in addition to its trusted links. If all the pseudonym links of a node have expired by the time it rejoins the system, its only valid links will be the trusted links. Hence, the overlay will behave more like the trust graph than like a random graph. Note that the time a node remains online, while important for the general quality of the overlay, is not important for determining the appropriate value for pseudonym lifetime.

In our experimental evaluation in Section IV we study the effect of the ratio between pseudonym lifetime and the average time nodes spend offline before rejoining the system. The higher this ratio the more the overlay resembles a random graph. This ratio also introduces a trade-off between good data-dissemination properties and better privacy properties under certain threat models.

In this paper we consider pseudonym lifetime a global system parameter with the same value for all nodes. However, it might be better to let each node adapt the lifetime of its pseudonyms based on the availability characteristics of the other participating nodes.

D. Pseudonym distribution and sampling

Our protocol for pseudonym distribution and sampling is based on gossiping algorithms, using ideas from shuffling [10], [11] and membership-sampling [12] protocols. The basic idea is that nodes periodically exchange pseudonyms through their overlay links, and apply a sampling mechanism to the pseudonyms they receive in gossip exchanges in order to select candidates for pseudonym-link creation.

1) Shuffling protocol: Each node n maintains a pseudonym cache of a configurable size. The cache is empty when the system starts.

Periodically, n selects a link from $n.links$ uniformly at random and executes a shuffling protocol with the node m at the other end of the link. Each of the two nodes sends an encrypted message containing a set of up to ℓ pseudonyms to the other, ℓ being also configurable. The set includes one node's own pseudonym and up to $\ell-1$ pseudonyms from the node's cache. Upon receiving a set over the link, the node updates its own cache to include all entries in the received set (with the exception of its own pseudonym, if present). The cache replacement policy is similar to that employed in [11]. Additionally, all pseudonyms in the received set, whether already in the cache or not, are sampled as we describe next.

2) Pseudonym sampling: A node does not need to have an overlay link to all of the pseudonyms in its cache. In our solution, each node establishes pseudonym links with a carefully selected sample of the pseudonyms received by the shuffling protocol. The maximum allowed number S of per-node pseudonym links governs the balance between potentially higher overhead and better overlay robustness. In our solution, S varies across the nodes so that all nodes will have a similar number of overlays links including trusted links. In particular, nodes that are very well connected in the trust graph (i.e., hubs) do not need the extra random links to ensure connectivity under churn.

The goal is to select a pseudonym sample in such a way that the resulting overlay is similar to a random graph. Our sampling protocol satisfies this requirement under the assumption that each pseudonym is a random p-bit sequence. In practice, if pseudonyms cannot be represented as random bit sequences, a similar effect can be achieved by adding some random bits to the original representation of a pseudonym and then applying a cryptographically strong hash function.

Each node n keeps its sampled pseudonyms in a list $n.L$ with S slots. Each slot contains a pair (P, R) where P is either a sampled pseudonym or an empty value, and R is a reference value. When n starts, it computes S random p-bit sequences and assigns each sequence to be a reference value for one of the slots in the list. The reference values are never removed or changed afterwards. We say that a slot (P, R) where P is empty is an empty slot. All slots are empty when the system starts. Pseudonyms are automatically removed from $n.L$ when they expire, and their corresponding slots become empty. Note that if n goes offline, the state of $n.L$ when n rejoins the system is the same as before n became offline, except for any pseudonyms in $n.L$ which might have expired during the offline period and are therefore removed.

Whenever a node n receives a pseudonym P' as a result of the execution of the shuffling protocol, n traverses $n.L$ and, for each entry (P, R), it replaces P with P' if (1) the slot (P, R) is empty, or (2) P' is numerically closer to R than P is, or (3) P' is numerically as close to R as P is, but P''s expiration time is later than P's. After traversing $n.L$, n updates $n.links$ to include only pseudonyms appearing in at least one slot of $n.L$.

The main advantage of this sampling method, which is based on the Brahms protocol [12], is that the set of pseudonym links for a node n will always be a random sample of all the pseudonyms n has received by means of the shuffling protocol, regardless of how frequently any pseudonym is received.

E. Privacy preservation

In this section, we analyze the privacy protection our protocol provides under various threat scenarios involving both participating nodes (e.g., internal observers) and external observers. We assume each participating node is able to observe only the traffic arriving at it even though the nodes can collude.

The analysis shows that our system is not vulnerable to obvious attacks and is able to satisfy our privacy requirements under several reasonable threat models. Furthermore, we believe that using our system is not worse in terms of privacy protection than using an overlay that uses only links between trusted nodes, and that successfully compromising privacy would require a powerful adversary able to subvert the underlying anonymity and pseudonym services.

1) Single internal observer: A single node n which is not a cut vertex in the trust graph has very limited capability to derive meaningful information about the trust graph with certainty better than a random guess. In particular, n cannot determine the ID of any participating node, as IDs are never propagated. Similarly, n does not have enough information to discover any nonincident edge in the trust graph, including edges between nodes adjacent to n in the trust graph.

2) Multiple colluding internal observers: A set of colluding nodes that do not form a vertex cut in the trust graph and that do not represent a large majority of the set of participating nodes also have limited capability to derive meaningful information about the trust graph. As in the case of a single internal observer, these nodes cannot determine the ID of any other participating node other than their neighbors in the trust graph.

Such a set of nodes can use timing analysis to detect the presence of an overlay link between any pair of their adjacent nodes, but it would be difficult to determine if the overlay link is a trusted link. Suppose observer nodes n and o are adjacent to a and b, respectively. Then n can produce a pseudonym P and send it only to a. If a gossips P to b in the next gossip round and b gossips P to o in the next round as well, then n and o can reasonably assume that an overlay link exists between a and b. Such a sequence of events, while possible, is unlikely to occur. It would be necessary that a chooses to propagate P quickly among all the pseudonyms in its cache, and that it chooses to propagate it to b. Then b has to do the same. The attack becomes easier as the number of colluding nodes connected to b (but not connected to a) grows, since the probability that b will choose a colluding node becomes higher. Determining whether the detected link is trusted or not is difficult, as it would require repeated successful execution of the attack over a long time, which is highly unlikely to occur.

3) Set of internal observers that are a vertex cut in the trust graph: When a set of colluding internal observers forms a vertex cut in the trust graph, then it has the possibility to control the flow of pseudonyms from one part of the graph to the other. If this set maliciously deviates from the protocol and sends only pseudonyms created by the set, then it can detect the existence of overlay links between adjacent nodes using the timing-analysis technique described in the previous section. Again, it is not possible to know if a and b have an edge in the trust graph, but they know that the trust path between a and b includes only nodes from their part of the graph, which may increase the likelihood that the link is trusted. In particular, if a and b are the only nodes in that part of the graph, then the malicious set can know for sure that there is a trust edge between a and b.

4) Estimating the size of the overlay: If the number of nodes in the system is small, then all nodes will eventually see all pseudonyms in the system before they expire, which allows nodes to estimate the number of participating nodes. This, however, does not violate our privacy requirements.

5) External observers: A full analysis of the resistance of our system against external observers is out of the scope of this paper, as it depends on the specific characteristics of the anonymity and pseudonym services employed. However, we can reasonably state that our protocol does not have any particular characteristic that makes it more vulnerable against external observers than, for example, using an overlay with only trust links running an application-level data-dissemination protocol. In particular, our pseudonym distribution protocol does not depend on very low latencies, as it is not an interactive application. For example, for a pseudonym lifetime of a few hours, pseudonym propagation times in the order of minutes are more than acceptable, which gives the opportunity to apply many effective anonymity techniques. In addition, we apply techniques such as ephemeral pseudonyms and end-to-end encryption, which make it harder for external observers to successfully perpetrate certain attacks. We believe that most constraints for the anonymity service will come from additional application-level requirements, than from our overlay maintenance protocol.

6) External observer colluding with a participating node: We can reasonably argue that controlling a participating node that only acts as an internal observer gives no strong benefit to an external observer. The reason is that, due to the use of multiple encryption layers, it is exceedingly difficult to associate a known message sent by a participating (colluding) node with any message circulating over a mix network, just by looking at the content of the message. Unless the colluding participating node is allowed to deviate from the protocol, timing analysis by the external observer does not benefit from colluding with a participating node.

If the colluding node does deviate from the protocol by producing anomalous messages or messages which are timed anomalously, the external observer might gain some advantage for traffic analysis, although the scheme that the adversary should use does not look obvious. Furthermore, it seems that any advantage an adversary gains by having an internal colluder in our protocol is not greater than the advantage gained by having an internal colluder with other protocols using the same anonymity service, including an overlay based exclusively on trust links running application-level data dissemination protocols.

IV. Experimental settings

In this section, we present the settings we used for evaluating our system. We implemented our protocols in a custom event-based simulation environment. Our inputs are based on a real-world social graph sampled from the Facebook social networking site, and synthetic churn models.

In our setup, we assume the existence of ideal anonymity and pseudonym services which allow the creation of privacy-preserving links that are reliable and have both low latency and high bandwidth. This means that all messages sent through an overlay link are delivered in a short time, provided that both ends of the link are online.

In all cases we use the shuffling period as our time unit. The reason is that all important interactions in our system depend on this period. Note, however, that our simulations are not based on rounds, but on events, which can occur at any time within the duration of a single shuffling period.

A. *Trust graphs*

We believe that social graphs from real-world social-networking sites such as Facebook provide a good representation of trust among users. In our evaluation we use a graph obtained by Wilson et al. [13] by crawling Facebook. This graph has about 3 million nodes and 28 million edges, and its degree distribution follows a power law.

Our system is intended for privacy-preserving applications in which the number of users is not likely to be as large as 3 million, thus in our evaluation we use smaller trust graphs sampled from the crawled Facebook graph.

Our sampling mechanism starts at a random node and adds additional nodes by traversing the graph following (some of) the contacts of each node until reaching a pre-established number of nodes. The edges of the sampled trust graph are all the edges among the selected nodes in the original Facebook graph. This method mimics an invitation model for participating in the group, which is common in real-world applications where privacy is a concern.

Our sampling method has a parameter f, which controls the number of neighbors in the Facebook graph to add to the sample when visiting each node during graph traversal. More specifically, when we visit a node n during the traversal, we add to the sample $\max(1, f \times |\delta(n)|)$ random neighbors of n which have not yet been visited. These newly added nodes are in turn visited in a breadth-first manner. $|\delta(n)|$ is the degree of node n. Note that using $f = 1$ is the same as a full breadth-first traversal, equivalent to users persuading all their friends to join the group; using $f = 0$ is equivalent to a depth-first traversal, roughly equivalent to each node inviting one friend; and $0 < f < 1$ is a partial breadth-first traversal, equivalent to users inviting some of their friends.

We evaluated our sampling method for several values of f and found that, for a given value of f, it produces similar graphs even if the graph traversal starts from different nodes. We also found that the variation between sampled graphs is lower for smaller values of f.

B. *Churn Model*

We model churn in our system using a scheme proposed by Yao et al. in [14]. This model describes the behavior of a node that alternates between online and offline states using a separate probability distribution for the time spent in each state. Yao et al. consider exponential and Pareto distributions as good candidates for individual online/offline time distributions. In this paper, we use only exponential distributions, which have a single parameter that represents the distribution's mean. Hence, in our model, each node has two parameters: mean online time and mean offline time.

Parameter	**Default**
Number of nodes in trust graph	1000
Trust-graph sampling parameter (f)	0.5
Mean offline time in shuffling periods (T_{off})	30 sp
Pseudonym lifetime	$3 \times T_{\mathit{off}}$
Size of pseudonym cache	400
Number of pseudonyms exchanged during a shuffle (ℓ)	40
Target number of overlay links per node	50

Table I: Default values for system parameters

For each experiment reported in this paper we give all nodes the same average availability by giving the same parameters for mean online time and mean offline time to the corresponding exponential distributions that model individual node behavior. We define the average availability of a node as $\alpha = T_{on}/\left(T_{on} + T_{\mathit{off}}\right)$ where T_{on} is the mean online time, and T_{off} is the mean offline time.

C. *Performance metrics*

We measure the robustness of our overlay by looking at its connectivity, average path length, and degree distribution. Since all communication through overlay links can be bidirectional, we use undirected-graph metrics.

To evaluate connectivity, we measure the fraction of nodes that are not part of the largest connected component of the overlay. If the overlay is connected, then this fraction is zero. Our goal is to keep this fraction as low as possible.

We use a normalized path-length metric, which we define as the average path length in the largest connected component of the graph (considering only online nodes), divided by the number of nodes in that component and multiplied by the total number of nodes in the graph (including offline nodes). This metric prevents the reporting of misleading short average path lengths in heavily partitioned graphs where the largest component may be small.

We also evaluate the degree distribution taking into account only online nodes. This metric helps understand the topology of the graph and can reveal weaknesses of the overlay, such as a large number of nodes with low degree.

Other performance metrics that we use are the time it takes the system to converge to a highly connected overlay under churn, and the overhead of the overlay over time in terms of the number of overlay links that need to be replaced on each shuffling period either due to pseudonym expiration or the sampling of better pseudonyms.

D. *Default system settings*

In this section we discuss the default settings we used in our simulation experiments, which we summarize in Table I.

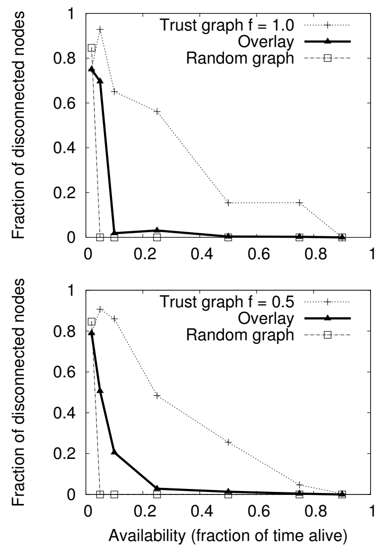

Figure 3: Connectivity for different trust graphs

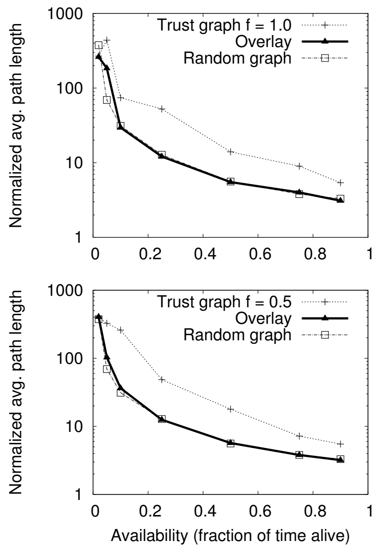

Figure 4: Normalized average path length for different trust graphs

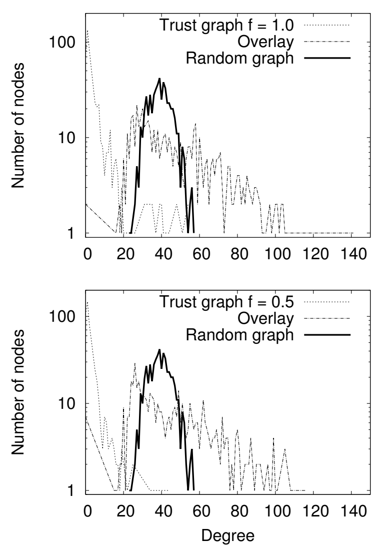

Figure 5: Degree distribution for different trust graphs, $\alpha = 0.5$

We used trust graphs of size 1000. We also experimented with other sizes, but we observed that, for a given value of the graph-sampling parameter f, graph properties do not change significantly. We present results for trust graphs sampled with different f values, but in experiments where we study the effect of other parameters our default configuration is to use $f = 0.5$.

We do not have a default churn setting, as in most of our analyses we study multiple availability scenarios. However, in all cases we assume a mean offline time T_{off} equivalent to 30 shuffling periods, and we adjust the mean online time T_{on} to obtain the availability setting that we want to evaluate.

The default value we use for pseudonym lifetime is 90 shuffling periods, which is equivalent to 3 times the value we use for T_{off}. In our evaluation we study the sensitivity of the system to the pseudonym lifetime. However, since the impact of this parameter greatly depends on the mean offline time of nodes, we prefer to use the ratio r between pseudonym lifetime and the mean offline time.

With regards to parameters for pseudonym distribution, in all cases we use a cache size of 400 and have nodes send up to $\ell = 40$ pseudonyms on each shuffle exchange. With respect to pseudonym sampling, we use a target number of overlay links per node of 50. This means that each nodes tries to establish overlay links with up to 50 peers. The actual degree of any node n may be higher than 50 due to links to n established by other peers. The degree can also be higher than the target if n has a large number of trusted links.

V. Experimental results

In this section we present the results of our experimental evaluation. We studied our system in two main scenarios under churn: i) with different trust graphs, and ii) with various pseudonym lifetimes. We have analyzed our system in other scenarios, but, due to space limitations, we report here only the most relevant results.

In all experiments we used the settings presented in Section IV-D unless otherwise stated. Our results show the state of the system after the reported metrics have reached stable values. The only exception is when we report how metrics evolve over time.

Our results show that our system is able to produce robust overlays under many demanding churn conditions. Robustness is achieved quickly even when the overlay needs to be reconfigured constantly due to pseudonym expiration.

A. Behaviour with different trust graphs

Our first set of experiments evaluates how our system behaves under different trust graphs. We sampled two 1000-node trust graphs from the Facebook graph, using the values 1 and 0.5 for the sampling parameter f, and evaluated the system for various values of average node availability α. For $f = 1$ the sampled graph has 5649 edges, and for $f = 0.5$ the number of edges is 3277.

Figure 3 shows that, as average node availability decreases, both trust graphs start to exhibit serious connectivity problems, whereas the overlay exhibits high connectivity for availability values as low as 0.25. For $f = 1.0$, our overlay exhibits good connectivity even for $\alpha = 0.125$. In this case the higher number of trust links with respect to the case in which $f = 0.5$ helps achieve good connectivity even in such a demanding churn scenario.

We can see in Figure 4 that the normalized average path length of our overlay is significantly lower than that of the trust graph, and closely matches a reference random Erdös-Rényi graph of similar size for all availability values.

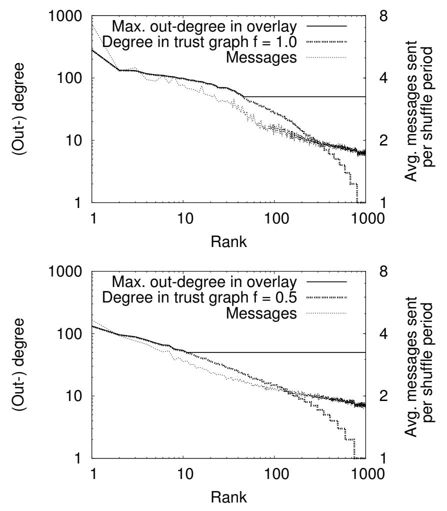

Figure 6: Average number of messages sent per shuffle period by each node for $\alpha = 0.5$ and two different values of the sampling parameter f. Nodes are ranked according to their degree in the trust graph. The figure also shows the maximum out-degree of the overlay for each node.

Figure 5 shows the degree distribution for $\alpha = 0.5$. The addition of pseudonym links shifts the degree distribution of the trust graph to the right. While it is clear that the overlay is close to the random graph, its degree distribution is not as concentrated around the average due to the presence of trust links, which have a skewed distribution.

Finally, Figure 6 shows the average number of messages sent per shuffle period by each node during the time the node was online. The figure also shows the maximum out-degree of each node in the overlay. Note that actual out-degree of a node can vary over time due to expiration of its pseudonym links. At each shuffle period, each node sends a shuffle request to one its neighbors in the overlay and it replies to shuffle requests that it receives from other nodes. The average number of messages sent per shuffle period per node across the whole overlay is 2: one message for a shuffle request generated by each node, and one message for the corresponding response. However, response messages are not necessarily equally distributed across all nodes. As the figure shows, nodes with more overlay neighbors send more messages per shuffle period than nodes with fewer neighbors. The reason is that such nodes have a higher probability of receiving shuffle requests because they are referenced by more peers in the overlay and, therefore, have to send more responses. In the case of nodes with the same maximum out-degree in the overlay, nodes with more trusted peers receive, on average, a slightly higher number of shuffle requests because they have fewer pseudonym links, which can be removed due to pseudonym expiration.

B. *Effect of pseudonym lifetime*

In this set of experiments we aim to ascertain the effect of pseudonym lifetime on the connectivity of the overlay for various availability values. As discussed earlier, lower pseudonym lifetimes might be better from a privacy perspective but higher pseudonym lifetimes are better for robustness.

In these experiments we vary the ratio r of pseudonym lifetime and mean offline time. We use the values 1,3,9 and infinite, which represents pseudonyms that do not expire.

Figure 7 shows that, as expected, higher pseudonym lifetimes result in more robust graphs. For $r = 9$ or infinite, the overlay closely resembles the random graph. For $r = 3$, connectivity degrades significantly for $\alpha = 0.125$, and for $r = 1$, significant degradation is observed for $\alpha = 0.25$. Having $r = 1$ results in a graph with poor connectivity as most pseudonym links for nodes that go offline expire before they rejoin. This result in behavior that tends to resemble the trust graph more than a random graph.

Figure 8 shows how the connectivity of the overlay evolves over time for $\alpha = 0.25$. It can be seen that for r values of 3 and 9, the overlay significantly improves its connectivity after a few shuffling periods and stabilizes at (nearly) full connectivity after little more than 200 shuffling periods. The trust graph, on the other hand, has a semi-stable fraction of around 70% of disconnected nodes over the whole experiment.

Figure 9 shows the average number of links that are replaced per shuffling period on the set of pseudonym links of each node. These replacements can occur either due to pseudonym expiration or due to the sampling of pseudonyms that improve the randomness of the overlay. The figure shows that, if pseudonyms do not expire, nodes quickly find the best overlay links do not need to make any further changes. In this case, nodes could easily stop executing the shuffling protocol after detecting the stabilization. For $r = 3$ we see that the frequent pseudonym expiration makes nodes replace pseudonym links at a rate of around 9 links per shuffling period. In the case of $r = 9$ we see an oscillatory behavior during the starting phase of the experiments. This is caused by the fact that all the nodes that are online when the experiment starts create their pseudonyms at the same time. When all these pseudonyms expire, nodes need to replace all the expired links with new links. The oscillations disappear once nodes go offline and rejoin at random points in time. Since pseudonym lifetime is higher than for $r = 3$, the number of link replacements per shuffling cycle is lower and stabilizes around a rate of 4 links per shuffling period.

VI. Related Work

The most common approach for building decentralized privacy-preserving communication overlays is to use F2F networks in which the overlay includes only links between nodes who trust each other. Some examples of this approach are Turtle [15], and Freenet's so-called darknet mode [2]. These systems normally focus on providing facilities for file sharing. As we have shown, overlays built with this approach do not provide optimal properties for data dissemination.

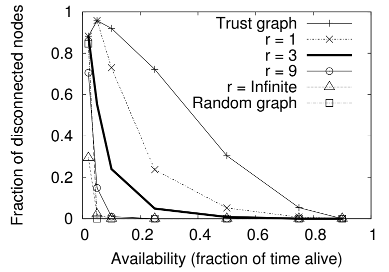

Figure 7: Connectivity for different pseudonym lifetimes

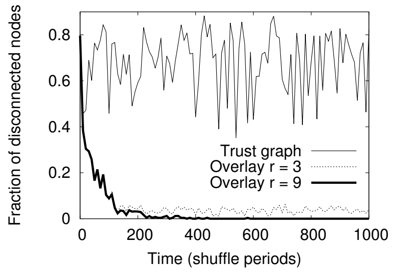

Figure 8: Connectivity over time for $\alpha = 0.25$

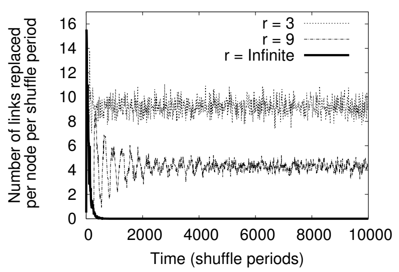

Figure 9: Number of links replaced per node per shuffle period for $\alpha = 0.25$

Another approach for enabling privacy-preserving decentralized group communication is to use social relationships to control who is able to participate in the group, but do not take such relationships into account for building the communication overlay. One example of this is Whisper [16], which allows building private invitation-only groups using random overlays that have privacy-preserving links based on mix networks. However, Whisper's privacy model is limited to preventing members of one group from learning the identities of members of other groups. No attempt is made to protect the identity of members within a single group. Another example is Membership-Concealing Overlay Networks (MCONs) [17]. As in Whisper, membership in a MCON is by invitation, but their primary goal is to protect the identity of users even from other participants. Their approach is to organize nodes in a DHT, such that each node is connected to a limited number of other participants. This helps prevent "celebrity" attacks, in which compromising a hub in the social graph allows the attacker to gain significant information about the whole system. However, the degree limitation in MCONs is achieved with the help of a trusted online central authority, which might also be compromised.

Our approach of building a robust overlay is similar in goal to the approach of building overlays that are guaranteed to be expander graphs [18]. However, it is not clear how to use such an approach in our privacy preserving context.

VII. Conclusion

In this paper we have presented a service for maintaining an overlay for privacy-preserving data dissemination for a group of users connected by trust-based social relationships. Our system leverages trust relationships among users to bootstrap a communication overlay, whose robustness is then improved with the addition of extra privacy-preserving links that make the overlay resemble a random graph. We evaluated our system experimentally using trust graphs sampled from Facebook's social graph and found that our system produces robust overlays and provides fast convergence while incurring acceptable overhead.

VIII. Acknowledgements

The work reported in this article was supported by the Tidal News project. Also, we thank Hein Meling for his helpful comments concerning this article.

References

[1] A. Shakimov, H. Lim, R. Caceres, L. Cox, K. Li, D. Liu, and A. Varshavsky, "Vis-à-vis: Privacy-preserving online social networking via virtual individual servers," in *COMSNETS*, 2011.

[2] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel, "Private communication through a network of trusted connections: The dark freenet."

[3] J.-F. Raymond, "Traffic analysis: protocols, attacks, design issues, and open problems," in *PET Workshop*, 2001.

[4] G. Danezis and C. Diaz, "A survey of anonymous communication channels," Microsoft Research, Tech. Rep., 2008.

[5] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *CACM*, vol. 24, 1981.

[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *SSYM*, 2004.

[7] B. Zantout and R. Haraty, "I2p data communication system," in *ICN*, 2011.

[8] G. Danezis, R. Dingledine, D. Hopwood, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *IEEE Symposium on Security and Privacy*, 2003.

[9] V. Shmatikov and M. Wang, "Timing analysis in low-latency mix networks: attacks and defenses," in *ESORICS*, 2006.

[10] A. Stavrou, D. Rubenstein, and S. Sahu, "A lightweight, robust p2p system to handle flash crowds," in *ICNP*, 2002.

[11] S. Voulgaris, D. Gavidia, and M. Steen, "CYCLON: Inexpensive membership management for unstructured P2P overlays," *JNSM*, vol. 13, no. 2, 2005.

[12] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, 2009.

[13] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *EuroSys*, 2009.

[14] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured p2p networks," in *ICNP*, 2006.

[15] B. Popescu, B. Crispo, and A. Tanenbaum, "Safe and private data sharing with turtle: Friends team-up and beat the system," in *Security Protocols*, ser. LNCS, 2006, vol. 3957.

[16] V. Schiavoni, E. Riviere, and P. Felber, "Whisper: Middleware for confidential communication in large-scale networks," in *ICDCS*, 2011.

[17] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim, "Membership-concealing overlay networks," in *CCS*, 2009.

[18] M. Naor and U. Wieder, "Novel architectures for p2p applications: The continuous-discrete approach," *ACM Trans. Algorithms*, vol. 3, no. 3, Aug. 2007.