# Prestige-based Peer Sampling Service: Interdisciplinary Approach to Secure Gossip *

Gian Paolo Jesi , Edoardo Mollona
Dept. of Computer Science
University of Bologna - Italy
E-mail: {jesi | mollona}@cs.unibo.it

Srijith K. Nair, Maarten van Steen
Dept. of Computer Science
Vrije Universiteit Amsterdam (The Netherlands)
E-mail: {steen | srijith}@cs.vu.nl

## ABSTRACT

The Peer Sampling Service (PSS) has been proposed as a method to initiate and maintain the set of connections between nodes in unstructured peer to peer (P2P) networks. The PSS usually relies on gossip-style communication where participants exchange their links in a randomized way. However, the PSS network organization can be easily modified by malicious nodes running a "hub attack", in which they achieve a leading structural position. From this prestigious status, the malicious nodes can severely affect the overlay and achieve several application dependent advantages. We present a novel method to overcome this attack and provide results from simulation experiments that validate our claim. This method is inspired by a simple technique used to detect social leaders in firm's organizations that is based on the social (structural) "prestige" of actors.

## Keywords

Gossip, peer sampling, security, SNA

## 1. INTRODUCTION

In large-scale distributed systems, such as P2P, there is a need to provide some method for sampling the network. This feature is needed, for example, to discover network properties like its topology, or to build and maintain robust overlays [3, 6, 15]. Usually, in such systems, each node maintains a set of so-called *local cache* or *view* of logical links (e.g., IP address and port number) to other nodes (neighbors). In dynamic conditions, where nodes constantly join and leave the system, these caches must evolve to reflect the changes triggered by the dynamism. A key requirement in this kind of systems is that the sampling should be uniformly random to ensure connectivity and resilience to crashes.

In the absence of malicious behavior, the local caches can be successfully maintained in a pseudo-random fashion using gossip-based approaches such as the Peer Sampling Service (PSS) [6], with distinct implementations (e.g., [6, 15]).

Unfortunately, the PSS can be exploited by potentially malicious nodes which wishes, in the worst case scenario, to defeat the overlay. As the status information is spread very quickly, non-malicious nodes may be too late in detecting that the overlay have been subverted and any attempt to recover could be useless. Figure 1(b) and (c) shows the effects of the presence of just 20 attackers interested in becoming the overlay's structural leaders in a 1000 nodes network. In the former subfigure the malicious nodes (i.e., the dark circles in the plot) became hubs as all the other nodes have logical links to only them. In the latter subfigure, these attackers leave the network after having achieved the structural positions, causing the overlay to become completely disconnected. What is worse, is that these effects can be achieved in a few gossip interactions and independent of the network size.

To solve this problem we have designed and tested (by simulation) a *prestige-based* Secure Peer Sampling Service (SPSS) based on heuristics inspired by a Social Network Analysis (SNA) technique [4, 16] used to measure the structural prestige of nodes in a network. The conceptual idea of becoming a structural leader goes beyond the boundaries of distributed systems; in fact, it is also an interesting research topic in firm's reorganization processes [12].

We show how techniques from SNA can be adopted in gossip-based distributed systems in order to solve the identified security issues. This approach is in line with others works that exploit social networks towards building robust systems [11]. We compare our approach with another from SPSS approach [8] and show its improved effectiveness in dealing with a larger set of attackers.

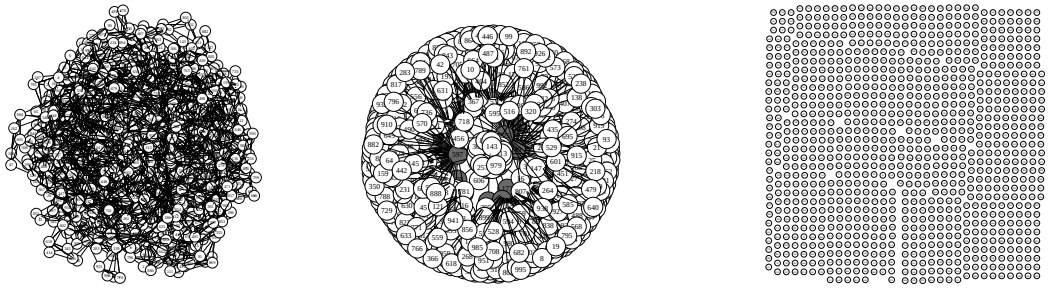
The remainder of this paper is organized as follows: in Section 2 we first describe our scenarios and the "hub attack" model [8] in which several colluding malicious nodes are sufficient to completely partition a network using a gossip based protocol. In Section 3, we focus on our specific problem and present the basic ideas behind our prestige-based solution. In Section 4, we describe the solution algorithm and then provide the results of our experimental evaluation in Section 5. Finally, we briefly survey related work and conclude with a summary, open issues and possible future work.

## 2. GOSSIP AND ATTACK MODEL

We consider a network consisting of a large collection of nodes that can join or leave at any time. Leaving the network can be voluntary or due to a crash. We assume the presence of an underlying routed network (e.g., the Internet) in which any node can, in principle, contact any other node. It is required that any node must be addressable by a unique *node identifier* (ID), such as an ⟨IP-address, port⟩ pair. We do not address the presence of firewalls and NAT routing among peers since it has been demonstrated that they

*The authors acknowledge financial support from the research project FIRB 2003 n° RBNE03HJZZ named "The evolution of clusters of firms: emerging technological and organizational architectures".


*SAC'09* March 8-12, 2009, Honolulu, Hawaii, U.S.A.



(a) healthy condition: no attackers (b) $k = 20$ attackers (c) $k = 20$ attackers after leaving the overlay

**Figure 1: Overlay topology in, respectively: (a) normal condition, when nodes are connected in a random fashion; (b) when the attackers (i.e., dark circles) are present in the system and become the "hubs" of the network; (c) the network becomes fully disconnected when the attackers, after polluting the other node's caches, have left the network. Network size is 1000 nodes.**

need not pose serious problems [5].

Due to scalability constraints, a node knows about only a small subset of other participants. This subset, which may change, is stored in a *local cache* of size $c$, while the node IDs it holds are called *neighbors*. This set provides the connectivity for a node in the overlay; the relation "who knows whom" induced by the neighborhood set defines the overlay *topology*. Real-world P2P applications provide a set of well-known, highly available nodes in order to provide a bootstrap facility which constitute the initial neighborhood set. In general, a *timestamp* is associated with each distinct ID stored in the cache to eventually purge "old" ID references according to an aging policy. This model does not rely on time synchronization, the time being measured in generic *time units* or *cycles* during which each node has the possibility to initiate a gossip exchange with another (randomly selected) neighbor.

When the content of the cache is corrupted by the presence of bogus IDs (i.e., *polluted* cache), the ability to communicate with the rest of the network can be severely compromised. In this case, a new initialization or *bootstrap* is required.

In our attack model, we consider a practical scenario in which a small set of *colluding* malicious nodes or attackers play in the system. The size $k$ of this set is equal to the cache size ($k = c$). *The goal of an attacker is to subvert the network in order to achieve a leading structural position* i.e., become a hub. The attack method involves the spreading of fabricated data through the messages gossiped among the participants in such as way as to affect the logical links of the overlay. We suppose the attackers are "clever", in the sense that they operate to avoid being easily discovered [10].

The attack algorithm, called the *hub attack* [8], can run any PSS implementation like any other well-behaving node, but the content of its malicious messages is filled with the IDs of other malicious nodes in the system. In addition, the *ts* for each ID is manipulated to ensure that it will not be dropped by a neighbor. This behavior is perfectly valid from a PSS point of view, as the only mandatory constraint is that cache entries be distinct. Surprisingly, this intrinsic weak integrity constraint complies to reality; in real-world P2P file-sharing systems (see [9,10,13]), when a peer receives an advertisement for an item, it does not usually verify its validity. Another kind of malicious cache attack have been proposed [7] but, in this work, we restrict to this one; however, the presented SPSS approach is also suited for this other cache attack.

In this paper, we consider a specific implementation of the PSS called NEWSCAST. However, our approach is generic and implementation independent. NEWSCAST is a gossip-based *topology manager* that builds and maintains a dynamic random graph. In this protocol, a node randomly selects one of its neighbors in order to subsequently exchange links from their respective caches. Note that such an exchange implies that (usually) the neighbor set of each node changes. Details can be found in [6].

# 3. PROBLEM AND SOLUTION GUIDELINES

The effect of gossiping malicious information is to mutate the randomly organized PSS overlay into a network having a completely different kind of organization that suits the attackers' aim. Our goal is to preserve the original organization of the PSS to avoid severe performance issues or the disruption of the overlay.

The main challenge is represented by the exponential growth of the infection which leads to a situation where each well-behaving node ends up with no neighbors other than the attackers. As a consequence, when a node is infected, it has no way to recover, especially in a fully distributed environment where there are no trusted parties to ask for help [8]. What is worse, is that just a single gossip exchange (initiated by a malicious node) may be sufficient to fully pollute a node's cache.

The basic idea and the main contribution of this work is to fit the PSS into a higher level framework which allows to: (a) play a PSS implementation as usual and (b) monitor the overlay and react to structure changes when required. The key point is that *the presence of the new framework is transparent* to the PSS and there is no way for a node to know how the information provided will be used by a receiving neighbor. In fact, a stochastic proportion of the gossip interactions could just be "explorative", while the others lead to standard PSS interactions. The task of the exploration is two-fold: on one hand it builds a particular sample of the current neighbor's surroundings, while on the other hand it collects node IDs that may become useful if and when the PSS cache becomes polluted by the spreading infection.

As there is no way to detect if a neighbor has actually played the PSS or not, this behavior generates a *dilemma* that could tremendously limit the attacker's power. As the attackers aim to mutate the overlay organization, they have to artificially promote them-

```
forever do
 PSS_done ← FALSE
 size ← [0 ... degree()]
 rndNSet ← RNDNEIGHBORSUBSET(size)
 wait(Δt)
 ∀ neighbor ∈ rndNSet do
  SENDSTATE(neighbor)
  n_state ← RECEIVESTATE()
  with prob. 1/size et !PSS_done do
   if CHECKIDS(n_state) do
    PSS_done ← TRUE
    apply PSS update.
  otherwise GETSTATISTICS(n_state)
 RECOVERSTATE()
```

(a) Active Thread

```
forever do
 n_state ← RECEIVESTATE()
 SENDSTATE(n_state.sender)
 if CHECKIDS(n_state) do
  apply PSS update.
 RECOVERSTATE()
```

(b) Passive Thread

**Figure 2: SPSS pseudo-code algorithm.**

selves by aggressively diffusing their IDs and they cannot hide this behavior.

Monitoring the overlay requires collecting statistics about the overlay structure, therefore a suitable *property* must be identified; this property has to be selected according to the particular nature of the overlay we address - random nature, in our case. The *nature of the property* we choose and the *way we check it* are the key points of our solution (see [10]).

For our solution we draw inspiration from social network analysis (SNA) and in particular we consider the notion of *prestige* of nodes in a directed network, where the nodes that receive more positive choices are considered prestigious [4]. In SNA, prestige is expressed as a particular pattern of social links. We adopt a (simple) technique to compute the *structural prestige* of a node in terms of *popularity* (or in-degree). The information about a node's in-degree is collected during a gossip exchange among neighbors, by looking at each entry in the received caches.

Since the network should be random, we expect that the average value of popularity is almost the same for each node. Detecting a node showing a popularity value far grater than the average means that it could represent a network hub. Each node builds its own knowledge about its surroundings and it does not share this information with its neighbors [10].

While in a firm's social context prestige is a desirable status (e.g., usually associated with higher respect, responsibilities and income), in our particular environment the prestige feature is an *undesirable* property. In a random overlay, the "egalitarian" nature of a P2P system is even more emphasized as it includes the homogeneity in terms of structural network position.

# 4. PRESTIGE-BASED SPSS ALGORITHM

As the main problem in preventing the hub attack is the exponential speed of infection, the main challenge is to let each node build a suitable *knowledge base* (KB) to possibly recover its local cache. Another SPSS approach [8] was based on building a KB according to a *quality rate* performed at each exchange. In such an approach, during each gossip, a node collects information regarding a single neighbor; it is a slow process mitigated by the fact that the probability of the actual state update is proportional to the quality value.

The approach we propose in this work can build a suitable KB in a much faster manner. In the following, we distinguish two distinct aspects:

**Dilemma mechanism**: Figure 2 shows the prestige-based SPSS algorithm using a pseudo-code language. The first important difference compared to the prototypical gossip scheme (see [3]) is the absence of the SELECTPEER() function which selects a random peer from the local cache. Here instead, a node can gossip with multiple neighbors in the same time unit (cycle); a sub-set of random neighbors, *rndNSet*, is selected from the cache by the RNDNEIGHBORSUBSET() function. This feature is similar to the behavior of cellular automata, which in turn can be considered as gossip systems as well [3]. In normal conditions - i.e., when the number of attackers is equal to the cache size ($k = c$) - just two gossips per cycle are sufficient to prevent the hub attack as, on average, one gossip cycle is dedicated to the PSS interaction and the other collects statistics about the overlay organization. The process of detecting the network organization is faster than the diffusion of malicious IDs.

Regardless of the number of neighbors selected by RNDNEIGHBORSUBSET(), the PSS state update policy can be executed only once (by the active thread) and with a probability of $\frac{1}{rndNSet.size()}$. The interactions with the other neighbors on the other hand, are used to collect prestige related information about the node's neighborhood. The presence of the $\frac{1}{rndNSet.size()}$ probability is the reason why a single gossip is not sufficient to prevent the attack, as it would disable both the dilemma and prestige mechanisms.

The proposed mechanism produces the dilemma in which a potentially malicious node A never knows how another neighbor B is going to use the information the malicious node provided. In fact, the more they pollute, serving malicious IDs during the exchanges, the more is the possibility of appearing as "suspect" in terms of popularity (prestige).

**Prestige mechanism**: The information collected during the gossip exchanges is used to build the KB required to detect with good accuracy the malicious nodes and to eventually repair the node's cache when it becomes polluted by the presence of malicious IDs. A table structure - the *Prestige Table* (PTABLE) - holds the following kind of tuple: [ node ID, #hits, TTL ], where each node ID detected is associated to a frequency value (#*hits*) and to a time-to-live (TTL) value expressing the time validity of the table entry. Essentially, each node explores its neighborhood and collects data about the popularity (or frequency expressed in #*hits*) with which the same node ID has been reported by the received caches. In just a single gossip, each node can collect a number of items equal to the current cache size ($c$ items). Though we do not pose any size restriction to the PTABLE, which can hence eventually grow up to the size of the network; this is very unlikely, as the entries are purged according to an aging policy, which decrements by 1 each entry's TTL at each cycle. An entry's TTL value is incremented (by 1) each time the same entry is detected.

As the attackers tend to acquire a network centric position, their prestige is likely to dominate over the other node's entries and its value will be far more than the average node's prestige. According to this simple idea, the PTABLE object maintains the average hits value: $\#hits_{avg}$; this value is used as a threshold to distinguish between potential attackers (i.e., $\#hits_A \geq \#hits_{avg}$ for a node A) and well-behaving nodes. When an entry expires (i.e., TTL=0), its ID is collected in a WHITELIST, as it can be (with high probability) considered a well-behaving node. If the same ID is detected in a neighbor's cache in a subsequent gossip, it is not removed from the WHITELIST until its #*hits* value has eventually reached the PTABLE $\#hits_{avg}$ value. These calculations and the management of the KB are handled by the GETSTATISTICS() function.

Any well-behaving node could play the PSS with an attacker that has not been detected yet, as its KB is not established enough or is even empty. The risk of having the cache polluted is always present and therefore a method to recover the node's cache is required. In Figure 2, the function RECOVERSTATE() represents this requirement. If the KB is not empty, it does not matter if not all the attackers are known. In the average case, it is sufficient to have a set of node IDs in the WHITELIST that are present in the PTABLE, but have a lower #*hits* value than the $\#hits_{avg}$ value, or that are not present at all in the PTABLE. When the KB is empty instead, the node's cache

could be completely polluted by an attacker. This would happen in the early stages of the protocol when a node has just joined the overlay. The only chance to recover, from such scenario, is to be contacted by another well-behaving and non polluted node. As the overlay is pseudo-random, on average, this can happen quite often.

The function CHECKIDS() is designed to perform basic cryptographic checks over the received IDs. The SPSS system requires a central Certification Authority (CA) which is not part of the protocol, but is required to provide the nodes the credentials to join the network. We cryptographically secure each ID structure using $[ID_A, ts_{creation}, ts_{expiration}, PK_A, \sigma]$, where $ID_A$ is $A$'s ID, the $ts$ are time-stamps, $PK_A$ is $A$'s public key and $\sigma$ is the digital signature on the message.

# 5. EXPERIMENTAL EVALUATION

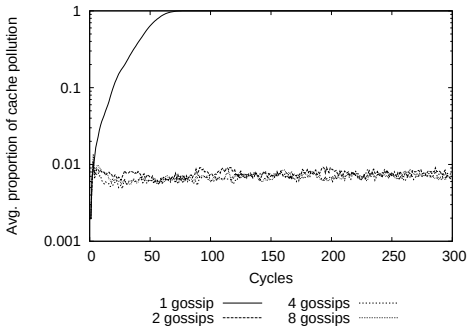



**Figure 3: The avg. pollution level in the caches is shown over time. Each plot represents a distinct number of gossips (i.e., 1, 2, 4, 8 respectively); $k = 20 = c$ malicious nodes are present. Network size is 10000.**

The prestige-based SPSS's evaluation reported in this section has the following goals: (a) how much time is required to achieve a stable and tolerable proportion of pollution in the node's cache and how the number of gossips per cycles influence this performance, (b) performance in a dynamic scenario and finally (c) comparison of the prestige-based SPSS with another fully distributed SPSS, based on a different approach [8], in terms of performance in extreme conditions. In the following evaluation, we consider that the pollution level is "tolerable" if the overlay graph does not split into clusters when the malicious nodes leave the network (see [8]). If not stated otherwise, the hub attack is performed by a set of $k$ colluding malicious nodes $k = c = 20$, where $c$ is the cache size the actual PSS implementation adopted (NEWSCAST). Due to the space constraints, only the results for the 10000 nodes overlay scenario will be reported, but the results are quite similar for smaller overlay sizes (e.g., 1000 and 5000 nodes).

Figure 3 shows the average pollution proportion in the node's caches. Each plot represents a protocol setup adopting a distinct number of gossips: 1, 2, 4 and 8. In this scenario, the overlay is static; in other words, we guarantee that during the life-span of the experiments no nodes will leave or join the network for any reason. When the nodes perform just a single gossip per cycle, the prestige SPSS is indistinguishable from the ordinary PSS, where no defense is available. Starting from the two gossips setup, the situation changes dramatically. The average pollution drops to a very low (and safe) 1-2%. Increasing the number of gossips further to 4 or 8 has a very marginal benefit, if any. It was seen that the communication cost, in terms of messages exchanged, grows much faster than the achieved pollution suppression benefit and 2 gossips are good enough to manage the attack.

In Figure 4, we consider the performance of the prestige-based SPSS in a *dynamic* setup, where at each cycle a set of nodes leaves the network and it is substituted by an equal amount of new nodes. However, this process involves only the well-behaving participants, while the malicious nodes are not affected and they pursue their malicious intent for the whole duration of the experiment. Three distinct churn rates are shown: 1%, 5% and 10% respectively. Two gossips per cycle are adopted.

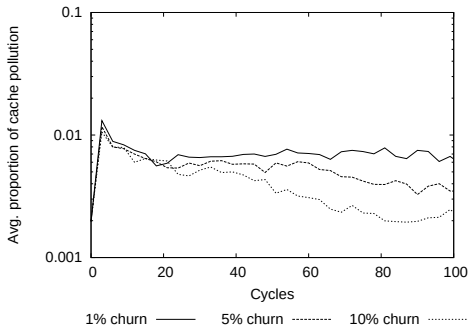



**Figure 4: Dynamic scenario: the avg. pollution level is shown over time according to distinct churn ratios. 2 gossips per cycle and $k = 20$ malicious nodes are adopted. Network size is 10000.**

It is interesting to see that the dynamism of the network actually *helps* the prestige mechanism to keep the cache pollution low. While it may seem counter-intuitive, the reason of this behavior is easily explained. As in the static scenario the attackers cannot subvert the system, in addition to that, well-behaving nodes are substituted at a constant rate and their caches are randomly initialized, minimizing the chance of joining the network with a polluted cache and neutralizing any previous success in polluting the previous ones. In fact, as the churn rate increases, the average pollution in the system decreases proportionally.

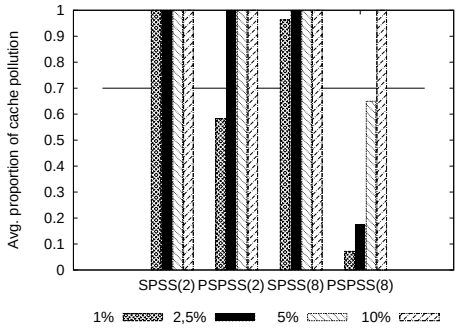



**Figure 5: Comparison among the prestige based SPSS and distinct a SPSS approach. The size of the network is 10000 nodes. The set of attackers is expressed as a percentage of the network size: 1%, 2,5%, 5% and 10%. The thick horizontal line represents the maximum tolerable pollution level.**

Figure 5 depicts an extreme attack scenario where the number of malicious nodes $k$ is expressed as a percentage of the overlay size and it is much higher than the cache size $c$ ($k >> c$); in particular, the percentages we selected are: 1%, 2,5%, 5% and 10% of the overlay size. We compared the prestige-based (SPSS) with another SPSS algorithm based on a different mechanism [8]. Each bar for each cluster represents a distinct percentage of attackers in

the network. The thick horizontal line represents the maximum tolerable pollution level over which the overlay runs the risk of being partitioned. To allow a fair comparison among the contenders, we adopted respectively 2 and 8 gossips setup for the former, versus 2 and 8 overlays setup for the latter SPSS. Essentially, the prestige-based SPSS can deal with the presence of a larger set of malicious nodes and its effectiveness is not limited to the standard $k = c$ case. However, the extreme scenario shows the benefit given by multiple gossips: the first setup (2 gossips) can just deal with 1% of attackers, while the second one (8 gossips) can tolerate 5% of attackers. The other SPSS instead always fails when $k >> c$.

The infection can spread to the whole network only if the cache size $c$ (and the number of explorative gossips) are too small compared to the attacker's population. Essentially, the attackers can successfully behave in a malicious manner if their population is sufficiently large, otherwise the more they pollute the more is the possibility to be discovered. When the set of attackers is too large, from a well-behaving node point of view, its overlay neighborhood looks random: the malicious caches it receives are fabricated from a large set of malicious IDs and they are effectively indistinguishable from a non malicious one without having a deeper knowledge of the overlay structure.

The prestige-based approach is also effective when more sophisticated malicious cache content (e.g., malicious node IDs are mixed with crashed or non existent node IDs) is adopted. However, we have not shown them in this work due to the space restrictions.

# 6. RELATED WORK

The previous SPSS approach [8], with which we made the comparison depicted in Figure 5, builds and manages distinct PSS overlays. By participating in multiple overlays, each node monitors the received cache patterns coming from distinct sources. The emergence of high frequency patterns may indicate the presence of the attackers and may trigger a topology reconfiguration and may blacklist the potential attackers. Our novel approach instead, does not need the extra overhead of maintaining multiple overlays, but each node just monitors its neighborhood. However, monitoring and playing the PSS is indistinguishable; this poses a dilemma for the attackers: if they pollute with brute force they will be discovered fast and with high probability; conversely, if they pollute more cleverly (e.g., at a lower rate) they will not be discovered, but they will hardly accomplish their malicious task.

In [1], the authors presents a sampling membership algorithm with which every node's sample converges to a uniform sample and can resists to the failure of a linear portion of the nodes. This approach defines and uses its own sampler algorithm, while we focus on securing an already existent sampling service (i.e., the PSS).

In [14], the authors introduces a fully decentralized approach for securing synthetic coordinate systems. They adopt a sort of social-like, vote-based approach in which each coordinate tuple must be checked by a (small) set of other nodes. For each node producing a coordinate tuple, the set of nodes that have to check and eventually approve that tuple is given by a hash function based on each node's unique identifier. The system is very resilient to attacks targeting instabilities and inaccuracies to the underlying coordinate system. However, this approach requires the presence of a DHT facility that adds complexity and may become an extra source of issues (e.g., DHT attacks).

Social network principles (e.g., *reciprocity* and *structural holes* [2]) are also adopted in JetStream [11] to optimize and build robust gossip systems. The basic idea is to make a predictable neighbor selection when gossiping to avoid unpredictable, excessive message overhead. In addition, the traditional scalability and reliability of gossip are maintained.

# 7. CONCLUSION

We have presented a novel SPSS mechanism which maintains the underlying random overlay in the presence of malicious nodes playing the hub attack. Our mechanism is based on a prestige-based SNA technique. The comparison with another SPSS method reveals the improved effectiveness of this approach when a larger set of malicious nodes ($k >> c$) are involved in the attack.

# 8. REFERENCES


[1] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine resilient random membership sampling. In *PODC*, June 2008.

[2] R. Burt. *Structural Holes: The Social Structure of Competition*. Harvard University Press, 1992.

[3] P. Costa, V. Gramoli, M. Jelasity, G. P. Jesi, E. Le Merrer, A. Montresor, and L. Querzoni. Exploring the Interdisciplinary Connections of Gossip-based Systems. *Operating System Review*, 41(5):51–60, October 2007.

[4] W. de Nooy, A. Mrvar, and V. Batagelj. *Exploratory Social Network Analysis with Pajek*. Cambridge, 2005.

[5] N. Drost, E. Ogston, R. V. van Nieuwpoort, and H. E. Bal. Arrg: real-world gossiping. In *HPDC*, pages 147–158, New York, NY, USA, 2007. ACM.

[6] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3):8, 2007.

[7] G. P. Jesi, D. Gavidia, C. Gamage, and M. van Steen. A Secure Peer Sampling Service. UBLCS 2006-17, University of Bologna, Dept. of Computer Science, May 2006.

[8] G. P. Jesi, D. Hales, and M. van Steen. Identifying Malicious Peers Before it's Too Late: A Decentralized Secure Peer Sampling Service. In *IEEE SASO*, Boston, MA (USA), 2007.

[9] J. Liang, N. Naoumov, and K. Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *INFOCOM 2006*.

[10] S. J. Nielson, S. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *IPTPS*, LNCS. Springer, 2005.

[11] J. A. Patel, I. Gupta, and N. Contractor. JetStream: Achieving predictable gossip dissemination by leveraging social network principles. In *NCA*, Cambridge, MA, USA, 2006.

[12] T. Rowley, D. Bherens, and D. Krackhardt. Redundant Governance Structures: an Analysis of Structural and Relational Embeddedness in the Steel and Semiconductor Industries. *Strategic Management Journal*, 21, 2002.

[13] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware*, Nov. 2001.

[14] M. Sherr, B. T. Loo, and M. Blaze. Veracity: A fully decentralized service for securing network coordinate systems. In *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)*, February 2008.

[15] S. Voulgaris, D. Gavidia, and M. van Steen. CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. *J. Network Syst. Manage.*, 13(2), 2005.

[16] S. Wassermann and K. Faust. *Social Network Analysis*. Cambridge University Press, 1994.