

# New Metrics for Reputation Management in P2P Networks

Debora Donato<sup>1</sup>  
debora@yahoo-inc.com

Carlos Castillo<sup>1</sup>  
chato@yahoo-inc.com

Mario Paniccia<sup>3</sup>  
mario\_paniccia@yahoo.it

Giovanni Cortese<sup>2</sup>  
g.cortese@computer.org

Maddalena Selis<sup>3</sup>  
maddalenaselis@yahoo.it

Stefano Leonardi<sup>3</sup>  
leon@dis.uniroma1.it

<sup>1</sup> Yahoo! Research  
Barcelona, Spain

<sup>2</sup> Consorzio Universita Industria <sup>3</sup> DIS - University of Rome  
Radiolabs - University of Rome  
Tor Vergata - Rome, Italy

“La Sapienza”  
Rome, Italy

## ABSTRACT

In this work we study the effectiveness of mechanisms for decentralized reputation management in P2P networks. We depart from EigenTrust, an algorithm designed for reputation management in file sharing applications over p2p networks. EigenTrust has been proved very effective against three different natural attacks from malicious coalitions while it performs poorly on particular attack organized by two different kinds of malicious peers. We propose various metrics of reputation based on ideas recently introduced for detecting and demoting Web spam. We combine these metrics with the original EigenTrust approach. Our mechanisms are more effective than EigenTrust alone for detecting malicious peers and reducing the number of inauthentic downloads not only for all the cases previously addressed but also for more sophisticated attacks.

**Categories and Subject Descriptors:** H.4 [Information Systems Applications]: Miscellaneous; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.3.4 [Information Storage and Retrieval]: Systems and Software – *Distributed Systems*

**General Terms:** Algorithms, Security.

**Keywords:** Peer-to-Peer, Trust, Reputation, Distributed Systems

## 1. INTRODUCTION

Decentralized environments, such as Peer to Peer (P2P) networks, are increasingly spreading through the Internet. Their open structure indeed offers the possibility of sharing a huge quantity of information and resources. The main strength and at the same time main problem of these kind of networks is the lack of a central authority that can provide information about the performance of individual peers, or guarantee and certify the quality of the shared resources. The concept of ‘shared resource’ or ‘performance’ is very

general here. In fact, it can encompass both application-level behavior of a peer node (or even of its user), such as quality of information the peer is sharing, or ‘infrastructure-level’ behavior, such as the bandwidth the peer has available, the response time in performing a certain transaction, availability of the peer, and the like. We seek mechanisms that allow peers of a network to collect information and estimate metrics, describing the performance and quality of other peers, without resorting to a centralized service. To fit with the characteristics of peer-to-peer networks, where individuals or coalitions of peers may alter the measurements to obtain some advantage, such mechanisms should be robust against malicious peers. Our aim is to enable a distributed algorithm that is run by autonomous untrusted agents to be sufficiently reliable, efficient and secure [10]. In the kind of applications we are interested in, we are often concerned with assisting a peer, which needs to engage with another peer to perform some kind of transaction, in the selection of a peer which has the desired characteristics with respect to some metric. Also, we need to assist him with the rating of the information coming from other peers. In general, any distributed application where there might be several providers of a service, would benefit from this work. Data overlays and collaborative content distribution systems have a need for a distributed metrics evaluation mechanism. The first scenario where we apply this approach is file sharing in a peer-to-peer network where a number of malicious peers provide corrupted data. The goal is to provide a quick access to a trustworthy source and above all to update and retrieve quickly trust ratings of peers. This requires the implementation of tools for distributing efficiently trust computation between a number of honest peers. The final goal is to implement a system that is able to alter the natural scheme of selection of the transacting peers in order to maximize the number of authentic downloads without unbalancing the load in the network.

**Our contribution:** We depart from EigenTrust, an algorithm proposed in [8] for file sharing in p2p systems. In this work we presented various metrics that can be easily integrated with EigenTrust to build a reputation system, that is, a system able to “assist agents in choosing a reliable peer to transact with when one or more have offered the agent a service or resource” [10]. Our main contributions are summarized as follows.

– We adapt Truncated PageRank, Bit Propagation [4] and BadRank [1] to the P2P file sharing framework.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AIRWeb '07, May 8, 2007 Banff, Alberta, Canada.  
Copyright 2007 ACM 978-1-59593-732-2 ...\$5.00.

- We introduce a number of new attack models that, to the best of our knowledge, have not been addressed before.
- We introduce a new metric, called **dishonesty**, which prevents malicious peers from lying.
- We show that our combined approaches are more efficacious than EigenTrust in reducing the amount of inauthentic downloads for all the cases in which EigenTrust alone is not sufficient.

The rest of the document is organized as follows. Section 2 describes previous work on P2P reputation management systems. In Section 3, we present the algorithms we modify and integrate in our reputation system. Section 4 presents the experimental framework and the metrics we evaluate. In Section 5, we present the five threat models introduced in [8] and two more sophisticated kind of attacks. In Section 6, we show how to apply the algorithms originally developed to detect Web spam in the context of file sharing. Experimental results are given in Section 7. Finally, Section 8 presents conclusions and discusses future works.

## 2. RELATED WORK

The most relevant previous work is contained in [8] which introduces the algorithm EigenTrust and shows how to aggregate the local trust assessments of all peers in the network in an efficient, distributed manner. In [2] several issues related to practical aspects of keeping peers honest in EigenTrust are presented. In [15] it is proposed a novel algorithm, PeerTrust, that combine several parameters such as feedback from peers and credibility of the feedback source into a general trust metric. Guha et al.[6] propose a framework for trust and distrust propagation on networks. The main contribution is to address the conceptual and computational difficulties to propagate both trust and distrust. The goal is to infer the largest number of pairwise trust relationships from the analysis of a sparse network of opinions, as opposed to our case in which we aim to define a global trust value for each single peer in the network. More recently, in [16], a novel protocol called SybilGuard for limiting the corruptive influences of sybil attacks is based on detecting small cuts in the graph of trust relationships between peers.

There is already ongoing work on incorporating the notion of trust into distributed ranking of Web pages. Specifically, the goal is to incorporate the concept of trustworthy peer in the meeting selection step of the JXP algorithm [13]. In this work we look at a different set of applications. A first possible scenario is file sharing in a peer-to-peer network. The goal is to provide a quick access to a trustworthy source and above all to update and retrieve quickly trust ratings of peers. This might require the implementation of tools for distributing efficiently trust computation between a number of honest peers.

An excellent taxonomy of concepts in p2p reputation and trust management is given in [10].

## 3. PRELIMINARIES

In this Section, we give some preliminary notions and briefly present the algorithms we use in our approach.

### 3.1 PageRank

Let  $A_{N \times N}$  be the citation matrix of graph  $G$ , that is,  $a_{xy} = 1 \iff (x, y) \in E$ . Let  $P_{N \times N}$  be the row-normalized citation matrix, such that all rows sum up to one, and rows

of zeros are replaced by rows of  $1/N$ . The PageRank vector is given by the stationary distribution of the Markov chain with transition matrix  $P$ . In [3] it is shown that PageRank can be described as a functional ranking, that is, a link-based ranking algorithm that computes a scoring vector  $S$  of the form:

$$S = \sum_{t=0}^{\infty} \frac{\text{damping}(t)}{N} P^t.$$

where  $\text{damping}(t)$  is a decreasing function of  $t$ , the lengths of the paths. In particular, for PageRank the damping function is exponentially decreasing, namely,  $\text{damping}(t) = (1-\alpha)\alpha^t$ .

### 3.2 EigenTrust

In a distributed environment, peers rate each other after each transaction. Each time peer  $i$  downloads a file from peer  $j$ , it may rate the transaction as positive ( $tr(i, j) = 1$ ) or negative ( $tr(i, j) = -1$ ). Peer  $i$  may rate a download as negative, for example, if the file downloaded is inauthentic or tampered with, or if the download was interrupted. We define a local trust value  $s_{ij}$  as the sum of the ratings of the individual transaction that peer  $i$  had with peer  $j$ :  $s_{ij} = \sum tr_{ij}$ . Equivalently, each peer  $i$  can store the number of satisfactory transactions it has had with peer  $j$ ,  $sat(i, j)$  and the number of unsatisfactory transactions it has had with peer  $j$ ,  $unsat(i, j)$ . Then,  $s_{ij}$  is defined:

$$s_{ij} = sat(i, j) - unsat(i, j).$$

In order to avoid malicious peers to assign arbitrarily high local trust values, it is necessary to normalize them. The normalized local trust value  $c_{ij}$  is defined as follows:

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}.$$

This ensures that all values will be between 0 and 1. The challenge for reputation systems in a distributed environment is how to aggregate the local trust values  $s_{ij}$  without a centralized storage and management facility. In [8] it is presented a reputation system that aggregates the local trust values of all of the users that uses the notion of transitive trust: a peer  $i$  will have a high opinion of those peers who have provided it authentic files. Moreover, peer  $i$  is likely to trust the opinions of those peers, since peers who are honest about the files they provide are also likely to be honest in reporting their local trust values. The idea of transitive trust, naturally inspired to the PageRank principle [11], leads to a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust values. The eigenvector computation can be efficiently performed in a distributed manner. The system proposed in [8] is able to decrease the number of inauthentic downloads even when up to 70% of the peers in the network form a malicious collective under a number of threat attacks. Normalizing the local trust values in this manner allows to perform the computation of EigenTrust without renormalizing the global trust values at each iteration (which is prohibitively costly in a large distributed environment). A natural way to aggregate the normalized local trust values in a distributed environment is for peer  $i$  to ask its acquaintances about their opinions about other peers, and weight their opinions by the trust peer  $i$  places in them:

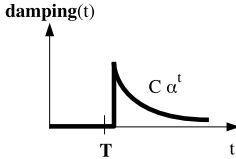
$$t_{ik} = \sum_j c_{ij} c_{jk}$$

where  $t_{ik}$  represents the trust that peer  $i$  places in peer  $k$  based on asking his friends. We can write this in matrix notation: if we define  $C$  to be the matrix  $[c_{ij}]$  of local trust values and  $t_i$  to be the vector containing the values  $t_{ik}$ , then ( $t_i = C^T c_i$ ), with  $\sum_j t_{ij} = 1$  as desired. We then ask the friends of the friends, therefore defining ( $t = (C^T)^n c_i$ ), after a large number  $n$  of iterations (under the assumptions that  $C$  is irreducible and aperiodic.) The vector  $t_i$  will converge to the left principal eigenvector of  $C$ , the vector of global trust values.

An important role for the convergence of the algorithm is played by **pre-trusted peers**, that is peers that are known to be trustworthy like the designers of the network. As we show in the Section 6, their presence ensures the convergence of the algorithms we introduce. A number of practical issues concerning the computation in a distributed and secure manner of the global EigenTrust vector are also discussed in [8]. These kind of solutions can be applied to the whole class of algorithms that are based on the propagation of local trust values to the set of neighbor peers with the goal of converging to a global measure of trust of the peers.

### 3.3 Truncated PageRank

Truncated PageRank is described in [4]. It is a link-based ranking function that decreases the importance of neighbors that are topologically “close” to the target node. This algorithm was originally introduced for demoting spam pages and is intuitively based on the consideration that most of the rank contribution for Web spam pages is originated from pages at very short hop distance. The idea is to consider a damping function that **removes the direct contribution of the first levels of links**, such as:

$$\text{damping}(t) = \begin{cases} 0 & t \leq T \\ C\alpha^t & t > T \end{cases}$$


where  $C$  is a normalization constant and  $\alpha$  is the damping factor used for PageRank. This function penalizes pages that obtain a large share of their PageRank from the first few levels of links; we call the corresponding functional ranking the **Truncated PageRank** of a page. The calculation of Truncated PageRank is described in detail in [4]. Essentially, this means that the Truncated PageRank can be calculated for free during the PageRank iterations.

### 3.4 Estimation of supporters

Following [5], we call  $x$  a **supporter** of page  $y$  at distance  $d$ , if the shortest path from  $x$  to  $y$  formed by links in  $E$  has length  $d$ . The set of supporters of a page are all the other pages that contribute to its link-based ranking.

The naive approach is to repeat a reverse breadth-first search from each node of the graph, up to a certain depth, and mark nodes as they are visited [9]. Unfortunately, this is infeasible unless a subset of “suspicious” node is known a priori. A method for estimating the number of supporters of each node in the graph is described in [4] which improves [12]. Also this algorithm was introduced for detecting spam. The general algorithm (described in detail in [4]) involves the propagation of a bit mask. We start by assigning a random vector of bits to each page. We then perform an iterative computation: on each iteration of the algorithm, if page  $y$

has a link to page  $x$ , then the bit vector of page  $x$  is updated as  $x \leftarrow x \text{ OR } y$ . After  $d$  iterations, the bit vector associated to any page  $x$  provides information about the number of supporters of  $x$  at distance  $\leq d$ . Intuitively, if a page has a larger number of supporters than another, more 1s will appear in the final configuration of its bit vector.

In order to have a good estimation,  $d$  passes have to be repeated  $O(\log N)$  times with different initial values, because the range of the possible values for the number of supporters is very large. This algorithm can be used to estimate the number of different peers contributing to the ranking of a given peer.

### 3.5 BadRank

BadRank [1] represents a reversion of PageRank, introduced in combination with PageRank in order to detect spam. It is based on the principle that if a page links to another page with a high BadRank, then also this page should be considered a page with negative characteristics. The difference with respect to PageRank is that BadRank is not based on the evaluation of inbound links of a web page but on its outbound links. Another fundamental feature is that the BadRank of a page  $i$  is given by a combination between the BadRank of all the pages that are pointed to by  $i$  and a value  $e(i)$  that reflect if the page was detected by a spam filter or not. The overall formula is the following:

$$br(i) = d \sum_{i \rightarrow j} \frac{br(j)}{indeg(j)} + (1 - d)e(i)$$

## 4. THE SIMULATION PLATFORM

In this section we briefly illustrate the simulation platform that has been proposed in [8] and adopted in [14] for evaluating the performances of EigenTrust under several threat attacks. We adopt this platform since it models several phenomena observed in practical p2p networks. This will also allow us to compare our algorithms with EigenTrust under different threat attacks.

The simulator [14] is based on a p2p network model with file-sharing peers able to issue queries for files, peers responding to queries, and files transferred between two peers to conclude a search process. When a query is issued by a peer, it is propagated by broadcast with hop-count horizon throughout the network (in the usual Gnutella way), peers which receive the query forward it and check if they are able to respond to it. We interconnect peers by a power-law network, a type of network prevalent in real-world P2P networks.

Concerning the content distribution model, it is considered a probabilistic interaction between peers. Peers are assumed to be interested in a subset of the total available content in the network, i.e., each peer initially picks a number of content categories and shares files only in these categories. We also assume that within one content category, files with different popularities exist, governed by a Zipf distribution. When the simulator generates a query, it generates the category and rank (or popularity) of the file that will satisfy the query. The category and rank are based on Zipf distributions. Each peer that receives the query checks if it supports the category and if it shares the file. Files are assigned probabilistically to peers at initialization based on file popularity and the content categories the peer is interested

(that is, peers are likely to share popular files, even if they have few files).

### Simulation execution.

The simulation of a network proceeds in simulation cycles: each simulation cycle is subdivided into a number of query cycles. In each query cycle, a peer  $i$  in the network may be *actively issuing a query*, *inactive*, or even *down* and not responding to queries passing by. Upon issuing a query, a peer waits for incoming responses, selects a download source among those nodes that responded and starts downloading the file. The latter two steps are repeated until a peer has properly received a good copy of the file that it has been looking for. Upon the conclusion of each simulation cycle, the global trust value computation is kicked off. Each experiment is run several times and the results of all runs are averaged, until there is convergence to a steady state (to be defined in the descriptions of the experiments). Initial transient states are excluded from the data.

### Metrics.

The metrics we use to assess the performance of the p2p system are the number of inauthentic file downloads versus the number of authentic file downloads: if the computed global trust values accurately reflect each peer’s actual behavior, the number of inauthentic file downloads should be minimized.

## 5. THREAT MODELS

Malicious peers can operate in different ways in order to subvert the system and gain high global trust. In [8] five possible threat model are presented.

**Threat Model A** Malicious peers always provide inauthentic files and assign high trust values to any other malicious peer they meet. This is done setting their local trust value as  $s_{ij} = \text{inauth}(j) - \text{auth}(j)$ . In this way malicious peers assess inauthentic file downloads instead of authentic ones.

**Threat Model B** Malicious always upload inauthentic files and collude with a set  $M$  of peers in the network. Each of the peers in  $M$  assign positive values to any other malicious peer in the clique it belongs to.

**Threat Model C** This model is also called *Malicious Collectives with Camouflage*. Malicious peers form a malicious coalition as in the previous case and upload inauthentic files in the  $f\%$  of the cases.

**Threat Model D** In this case the malicious peers are of two different types: part of them acts exactly as in the model B, the others, called *malicious spies* answer 0.05% of the most popular queries always uploading authentic files but assign trust value only to peers of type B. In this way, the spies gain trust global value that are able to somehow transfer to malicious peers.

**Threat Model E** Essentially, each malicious peer leaves the network after the upload of an inauthentic file and it enters the network again with a new identity.

**Threat Model F** The malicious disseminate one virus-laden file every 100 requests.

EigenTrust successfully addresses threat models A, B and C but badly performs in the case D since the efficient division of the work in this model allows the spies to increase more and more their global trust rank. The model E and F are

not addressed at all by EigenTrust. In the next Section we illustrate different metrics able to greatly reduce the rate of inauthentic file downloads in the case D. These techniques perform well also in the case of more sophisticated attacks.

**Malicious Smart Model** (or **Threat Model G** consistently with [8]) Malicious peers to further bedim their presence in the network, start to assign a local trust value also to a small fraction of good peers. Malicious peers can decide their **smartness degree**, that is, the fraction of good peers to which assign positive values. The configuration that results most strenuous to handle is the one in which the spies assign positive value only to the malicious peers, that instead assign positive value also to good peers.

**Malicious Smart Model with Camouflage** (or **Threat Model H**) The coalition uses the same schema of the Model G with the only exception that malicious peers upload inauthentic file in  $f\%$  of the cases.

## 6. ALGORITHMS

The set of interactions between peers is modeled by a **transaction network**, where a link from a node (peer)  $i$  to a node  $j$  is inserted every time  $i$  downloads a file from  $j$ . Each link is weighted with a positive value if the downloaded file was authentic, negative otherwise. In the most of the cases, we consider the **positive opinion network** in which a link is inserted from a node  $i$  to a node  $j$  only after the download of authentic files. Figure 7(a) shows the direct opinion network in the case of a coalition that follows the Threat Model D.

In this Section we show how to apply the algorithms originally developed to detect Web spam in the context of file sharing. These algorithms can be used instead of EigenTrust, or in combination with it, to dramatically minimize the number of inauthentic downloads. We want to stress that the behavior of each method, with the only exception of the last one, is strongly dependent from the Threat Model considered and that we elucidate in the following the applicability and the limitations of each of them.

### EigenTrust with Inverse EigenTrust.

We call **inverse network** the transpose of the positive opinion network. From now on, with direct and inverse network, we refer respectively to the direct opinion and inverse opinion network. If malicious peers behave following the threat model D, they are not reachable by good peers on the inverse network, as shown in Figure 7(b). Actually a link from a good peer to a malicious peer on the inverse network should correspond to a trust value assigned from such a malicious peer to the good one (conversely to the model D). This means that a random walk from pre-trusted nodes has a null probability to terminate in a malicious node. We call this probability Inverse EigenTrust and we calculate it, using the EigenTrust algorithm on the inverse network, starting from the pre-trusted nodes. The idea of computing pages rank, starting from a set of trusted pages, is presented in [7] where an algorithm, called TrustRank is used to individuate spam pages, that is, pages that receive most of their *trust mass* from nodes that are not in a predefined trustworthy set. Here this algorithm, conveniently modified, is applied on the inverse network to discover nodes that are not reachable from the pre-trusted nodes.

Our goal is to integrate the values generated by EigenTrust and Inverse EigenTrust, preserving the ranking generated by

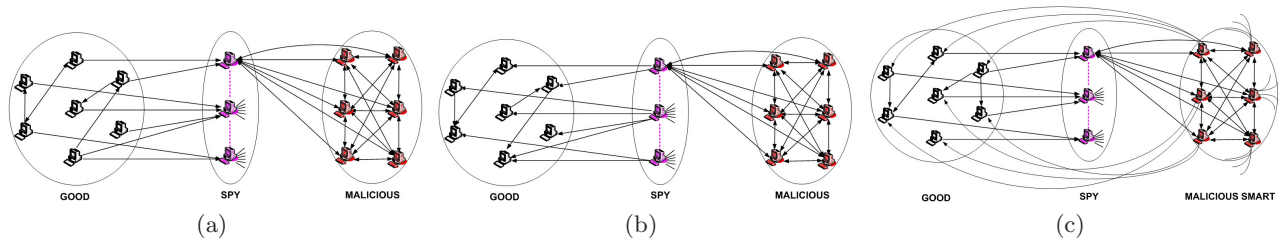


Figure 1: Graphic depiction of the model described in Section 5: (a) Treat Model D; (b) Threat Model D on inverse network; (c) Threat model G

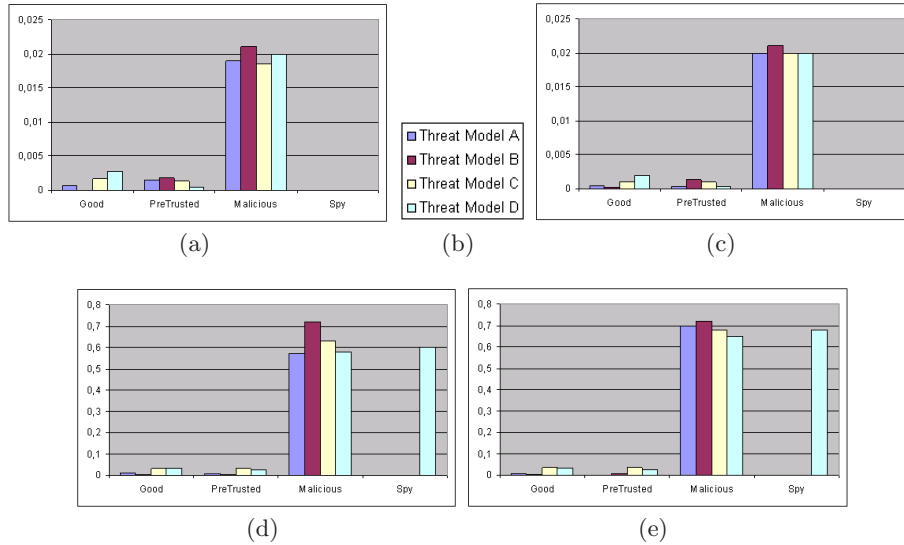


Figure 2: Badness after: (a) 25 cycles, (c) 50 cycles - Dishonesty after: (d) 25 cycles, (e) 50 cycles.

EigenTrust. A first possible approach is shown in Figure 3

**Require:** EigenTrust score vector  $ET$ , Inverse EigenTrust score vector  $I$

- 1: if  $I[i] > 0$  then
- 2: return  $ET[i]$
- 3: else
- 4: return 0
- 5: end if

Figure 3: Method 1 - Combination of EigenTrust and Inverse EigenTrust

In the case of the model G, this approach is not effective, since the cluster of malicious peers and spies is now connected to good peers on the inverse network for the effect of the *smart* positive values that malicious peers assign to good ones. Obviously as malicious peers increase their smartness degree, the Inverse EigenTrust of malicious peers approaches the Inverse EigenTrust of good peers. On the other hand, a direct effect of this kind of attack is that EigenTrust results more efficient, with respect to the case D, in decreasing the number of inauthentic downloads, since malicious peers dispel part of their assessments over a fraction of good peers. To cope with this trade-off, the Method 1 is modified introducing a threshold as shown in Figure 4. The main drawback of the Method 2 is that good peers with low Inverse

**Require:** EigenTrust score vector  $ET$ , Inverse EigenTrust score vector  $I$ , threshold  $tr = \sum_i \frac{ET[i]}{N}$

- 1: if  $I[i] \geq tr$  then
- 2: return  $ET[i]$
- 3: else
- 4: return 0
- 5: end if

Figure 4: Method 2 - Combination of EigenTrust and InverseEigenTrust with Threshold  $tr$

EigenTrust are assigned with a EigenTrust equal to 0. Nevertheless we experimentally observe that the rate of error is reasonably low for smartness values lower than 0.2.

### *EigenTrust with Truncated PageRank.*

Malicious peers receive positive opinions from the other members of the coalition (malicious and spy). This means that most of the *trust mass* is propagated starting from nodes at few hops distance. This behavior is observable also for Web link-based spam and a natural way to demote their rank is to not consider the rank mass coming from nodes within 1 or 2 hops. Truncated PageRank is an algorithm based on the previous observation. It can be straightforward used (alone or in combination with EigenTrust) to re-

duce the number of inauthentic downloads. The algorithm can be applied both on the direct opinion network and on the inverse one. Obviously the performances are constrained by the topology that the different kinds of attacks induce. The combined approach is shown in Figure 5.

**Require:** EigenTrust score vector  $ET$ , Truncated PageRank vector  $P$ , threshold  $tr$

- 1: **if**  $P[i] \geq tr$  **then**
- 2:     **return**  $ET[i]$
- 3: **else**
- 4:     **return** 0
- 5: **end if**

**Figure 5: Method 3 - Combination of EigenTrust and Truncated PageRank**

A difference with the Method 2 is that it is not possible to use the average Truncated PageRank as threshold because this quantity is always higher than the values experimentally obtained for the malicious peers. The value  $tr = 0.001$  has proved to well-perform during the simulations.

### EigenTrust with Estimation of Supporters.

The Bit Propagation algorithm for the Estimation of Supporters is introduced in Section 3.4. The assumptions done for the Web graph can be easily extended to the opinion network. Since a link from a peer  $i$  to a peer  $j$  exists each time that  $i$  wants to endorse  $j$ , we can say that the peer  $i$  is a **supporter** of the peer  $j$  at distance  $d$  if the shortest path from  $i$  to  $j$  has length  $d$ . Malicious peers supporters necessarily belong to the same coalition: they are negatively assessed by honest peers, since they spread corrupted files through the network. This means that a malicious peer obtains a high reputation because of the great number of supporters at short distance from it. The Bit Propagation algorithm can be used to perform an analysis of the connectivity of the opinion network in order to detect local anomalies. We combine it with EigenTrust as done for Truncated PageRank in Figure 6.

**Require:** EigenTrust score vector  $ET$ , Bit Propagation vector  $BP$ , threshold  $tr$

- 1: **if**  $BP[i] \geq tr$  **then**
- 2:     **return**  $ET[i]$
- 3: **else**
- 4:     **return** 0
- 5: **end if**

**Figure 6: Method 4 - Combination of EigenTrust and Bit Propagation**

### Badness and Dishonesty.

An alternative approach, inspired by BadRank [1], is possible if we explicitly consider, for each peer, **the negative opinion** vector. Exploiting the fact that each peer is prone to trust the opinions of peers for which has already built a positive reputation, we can say that if  $i$  trusts  $j$  and  $j$  distrusts  $k$  then, with high probability, also  $i$  should regard  $k$  as untrustworthy. Following the same arguments, used in [8] to build the EigenTrust vector  $T$  and exploiting the linearity of the product between matrices, we can define the **Global Badness** as:

$$\mathbf{negT} = \mathbf{D}^T * \mathbf{T}$$

where  $D$  is the normalized negative opinion matrix and  $T$  is the EigenTrust Rank. Each peer  $i$  has a global Badness given by

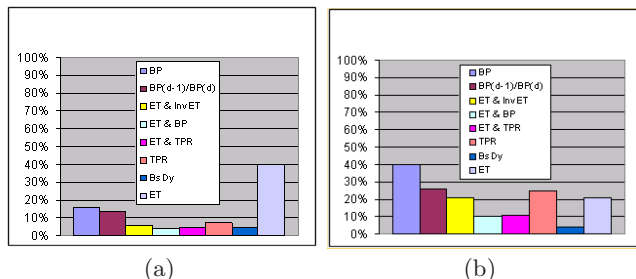
$$\mathbf{negT}_i = \sum_{j=1}^n \mathbf{negC}_{ji} * \mathbf{T}_j$$

Since peers are assigned with a probability of supplying corrupted files equals to 2%, the honest peers have a Badness value greater than 0, meanwhile the spy (since the strong assumption that they never provide corrupted files) always have Badness values equal to 0. The Badness for all the types of peers, respectively in the threat models A, B, C and D, are shown in Figure 2(b) after 25 simulation cycles and in Figure 2(c) after 50 simulation cycles. After only few cycles, the badness is able to differentiate between good and malicious peers but it does not help in discovering spies. For this reason we introduce the notion of **Dishonesty** as follows:

$$\mathbf{dishonesty}_i = \sum_{j \in P} \mathbf{negT}_j$$

where  $P$  is the set of peers that  $i$  have assessed. The dishonesty is high for all those peers which assess peers with high badness. As shown in Figure 2(d) and Figure 2(e), this measure is able to detect peers that are cheating in all the attacks presented in [8]. Since good peers can have a non-zero values of badness and dishonesty, we experience a number of false positive, that is, the algorithm can penalize also good peers. Obviously the main requirements of a profitable reputation management system must be the ability of minimizing the number of inauthentic downloads without penalizing good peers. The goal of minimizing the number of false positive can be pursued with a careful choice of the integration method, i.e. the strategy that allows to modify the probability with which a peer  $j$  is chosen for the download. A possible choice is to consider trustworthy all the peers with badness (and/or dishonesty) less than the average value of badness (and/or dishonesty). This method has the main disadvantage that penalizes an high number of good peers when the fraction of malicious is low. Anyway, tuning conveniently the thresholds allow to keep the percentage of false positives below 3%.

## 7. EVALUATION



**Figure 8: Reputation metrics for (a) threat model D and (b) threat model G**

All the experiments are done simulating a network with 100 good peers and 5 pre-trusted peers. In all the possible scenarios we consider, the good peers are assigned with

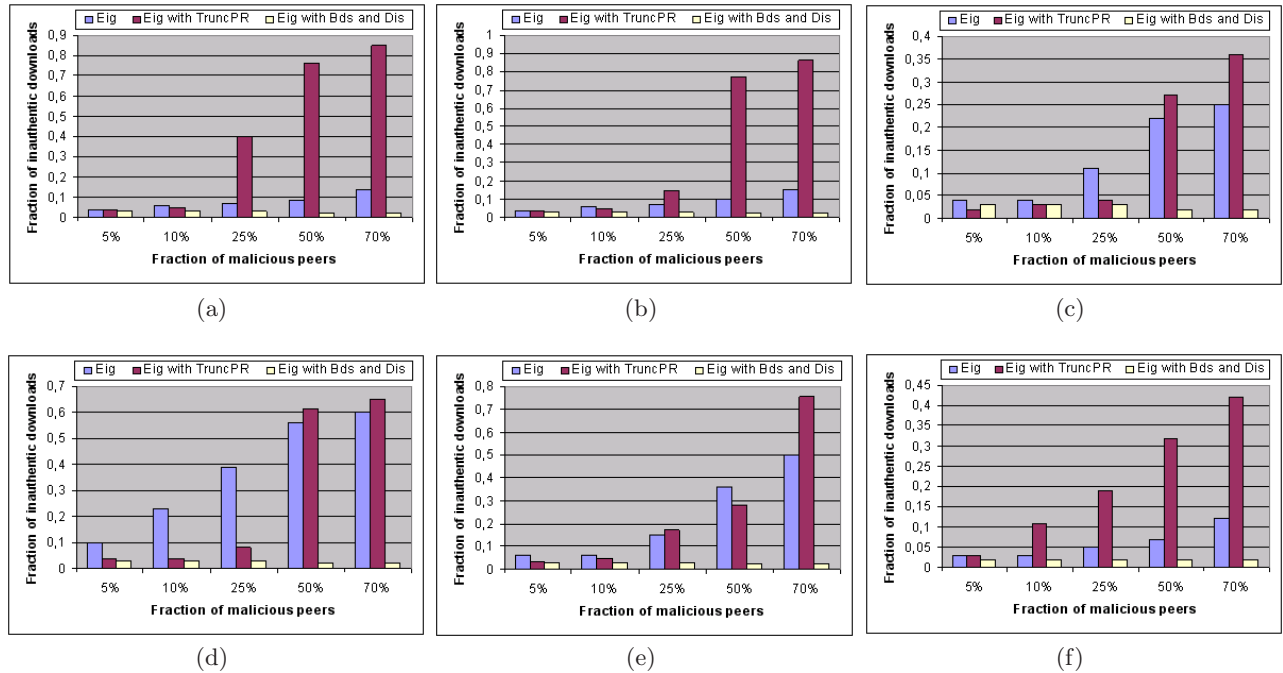


Figure 7: Threat model A,B,C,D presented in Kamvar et al. and G,H here introduced

a probability of supplying corrupted files equals to 2%. We fix the smartness degree equal to 1. We let the total number of colluding peers (malicious and spy) vary in order to analyze the behavior of all the proposed algorithms for different fractions of malicious peers. For each of the method presented in Section 6, we run a total number of 6 simulations of 50 cycles each. We consider the average ratio between the number of inauthentic downloads and the total number of downloads. The first 15 cycles of each run are considered initial transient states and are excluded from the data.

As said in the previous Section, EigenTrust, Truncated PageRank and Bit Propagation can be applied on the inverse and on the direct network, with performances that strongly depend on the threat model followed by the coalition. A straightforward consequence of this is that for what concerns the models presented in [8], the previous algorithms give best results if used on the inverse network, whereas, for the new threat models G and H, they work better on the direct network.

All the algorithms presented are successful in reducing the ratio between inauthentic and total downloads for threat model D, that was not properly addressed by EigenTrust. This is made evident in Figure 8(a), where we report the results of the algorithms used alone or in combination with EigenTrust. We also introduce as further measure the ratio between peers at distance  $d - 1$  and  $d$  or "bottleneck number"; this measure has been proved very effective in demote Web spam pages [4].

The threat model G is introduced as a possible reaction of the coalition to the new stronger reputation system. In this case, the combined approaches still result better than EigenTrust as shown in Figure 8(b), but the performance is lower than in previous cases. The only approach that is robust against all the types of attacks is the one based on badness and dishonesty. In Figure 7, we compare three methods: (i)

EigenTrust, (ii) Truncated PageRank combined with EigenTrust, (iii) Badness, Dishonesty combined with EigenTrust. We run different simulations varying the fraction of malicious peers. We can observe that

- The second method performs better than EigenTrust for fraction of malicious peers reasonably low but it is not stable when malicious peer fraction increases.
- Surprisingly the last method based on badness and dishonesty is always better than EigenTrust even for percent of malicious equal to the 70% of the global number of peers in the system.

The measure of dishonesty is able to single out, after few cycles, all the peers that are lying in assessing the other peers. Hence this method can penalize both the malicious peers and the spies. Lying is not more a profitable strategy for malicious peers, since they are immediately isolated in the system. We then consider a modified version of the threat model A, C and D in which the peers are malicious but not dishonest. We call these new threat models respectively, A', C', D+A', D+C'. The results are shown in Figure 9 and Figure 10. In the Table 1, we summarize all the results for the method based on badness and dishonesty.

## 8. CONCLUSIONS

The problem we address in this paper is to limit the effects that misbehaving agents or peers can cause to the global network or to single peers when defecting from the established protocol. We propose different techniques able to deal against different kinds of attacks that EigenTrust is not able to defeat.

This is a preliminary work. We are seeking for other approaches able, for example, to act on malicious peers that

Malicious	A	B	C	D	A'	C'	D+A'	D+C'	G	H
5%	4%/3%	4%/3%	4%/3%	10%/3%	4%/4%	3%/3%	5%/4%	3%/3%	6%/3%	3%/2%
10%	6%/3%	6%/3%	4%/3%	23%/3%	5%/5%	3%/3%	9%/5%	4%/3%	6%/3%	3%/2%
25%	7%/3%	7%/3%	11%/3%	39%/3%	5%/5%	6%/5%	22%/6%	7%/5%	15%/3%	5%/2%
50%	8,5%/2%	10%/2%	22%/2%	56%/2%	8%/7%	9%/7%	30%/8%	9%/6%	36%/2%	7%/2%
70%	14%/2%	15.5%/2%	25%/2%	60%/2%	13%/11%	11%/10%	42%/12%	18%/13%	50%/2%	12%/2%

Table 1: EigenTrust compared with EigenTrust with badness and dishonesty

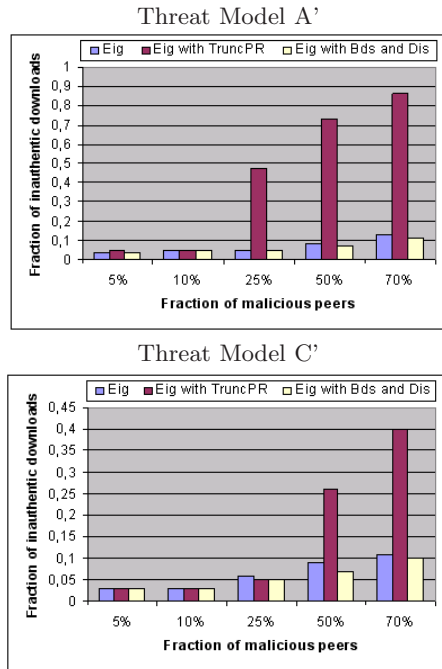


Figure 9: Threat Models A', C'.

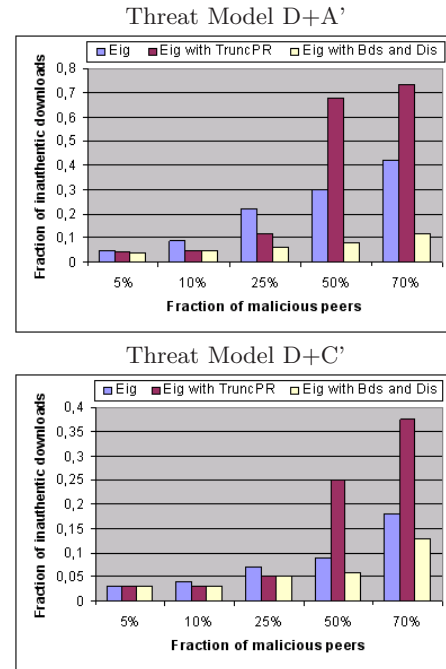


Figure 10: Threat Models D+A', D+C'.

collude in order to discredit a subset of honest peers. Moreover we want to implement more general mechanisms; the goal is to obtain an adaptive strategies for the thresholds that could be applied to any kind of attack or even to a combination of them.

## 9. REFERENCES

- [1] <http://ewwws.com/pr/przero.php>. PR0 - Google's PageRank 0 Penalty.
- [2] Zo Abrams, Robert Mcgrew, and Serge Plotkin. A non-manipulable trust system based on eigentrust, July 2005.
- [3] Ricardo Baeza-Yates, Paolo Boldi, and Carlos Castillo. Generalizing PageRank: Damping functions for link-based ranking algorithms. In *SIGIR*, Seattle, USA, 2006. ACM Press.
- [4] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *WebKDD*, Pennsylvania, USA, 2006. ACM Press.
- [5] A. Benczúr, K. Csalogány, T. Sarlós, and M. Uher. Spamrank: Fully automatic link spam detection. In *AIRWeb*, Chiba, Japan, 2005.
- [6] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW*, New York 2004.
- [7] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web spam with TrustRank. In *VLDB*, pp. 576–587, Toronto, Canada, 2004.
- [8] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, pp. 640–651, 2003.
- [9] R. J. Lipton and J. F. Naughton. Estimating the size of generalized transitive closures. In *VLDB*, pp. 165–171, San Francisco, CA, USA, 1989.
- [10] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, March 2006.
- [11] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. Tech. Rep, Stanford Digital Library Technologies Project, 1998.
- [12] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. ANF: a fast and scalable tool for data mining in massive graphs. In *KDD*, pages 81–90, New York, NY, USA, 2002.
- [13] J.Xavier Parreira, D. Donato, S. Michel, and S. Weikum. Efficient and decentralized pagerank approximation in a peer-to-peer web search network. In *VLDB*, Korea, 2006.
- [14] M. Schlosser and S. Kamvar. Simulating a p2p file-sharing network, 2003.
- [15] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE TKDE*, 16(7):843–857, July 2004.
- [16] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278, 2006.