Assessing Efficiency of Trust Management in Peer-to-Peer Systems

R. Aringhieri E. Damiani S. De Capitani di Vimercati P. Samarati
Università di Milano
Dipartimento di Tecnologie dell'Informazione
Via Bramante 65, 26013 Crema (CR), Italy
{aringhieri,damiani,decapita,samarati}@dti.unimi.it

Abstract

P2P applications support exchanging resources while preserving total or partial anonymity of both requestors and providers. However, concerns have been raised about the possibility that anonymity may encourage malicious peers to spread tampered-with resources (e.g., malicious programs and viruses). A considerable amount of research is now being carried out on the development of trust and reputation models in P2P networks. In this paper, we assess the efficiency of our approach to the design of reputation systems involving flexible techniques for collecting and aggregating peers' opinions via comparison with probabilistic approaches.

1. Introduction

A key requirement for large scale Peer-to-Peer (P2P) networks is allowing for different degrees of anonymity during interactions. In particular, full anonymity is widely acknowledged to be of paramount importance for establishing free marketplaces in many application environments [7]. On the other hand, anonymity is critical in presence of rogue peers, and there is increasing interest in systems capable of keeping the consequences of hostile behavior under control. In our previous work [5] we described a distributed voting algorithm for collecting other peers' views on a proposed transaction based on a two-step-technique: *i)* we poll the community of P2P users for collecting the different reputations on a candidate peer p, and *ii)* we merge the different opinions in a single value by means of a fuzzy aggregation [10]. The result of the vote is a quantitative estimation summarizing peers' evaluations that, taken individually, are subjective, dynamic, and often uncertain. In [4] we proposed our fuzzy aggregation as a community-based operational definition of *trust*. Other approaches [14] tried to deal with multiple types of trust based on a set of different interaction *pragmatics*; others [15] dynamically assign to each peer a *trust rating*, that is, a reputation based on the peer's previous performance on the network and store it at a suitable place. Robust trust reputation, based on a probabilistic model, have been also used to the problem of user agents selecting processor agents to processor tasks [11].
Any community-based technique for computing trust values must take into account the fact that fully anonymous environments are fundamentally different from centralized ones, inasmuch some basic assumptions (e.g., opinions' trustworthiness) cannot be made without adopting strong countermeasures against forgery. On the other hand, these countermeasures should not aggravate the computational cost of the voting scheme. In this paper, after briefly recalling our P2P reputation system,[1] we discuss its efficiency via a set of extensive simulations and we also compare it with the recent probabilistic approaches to trust computation called *EigenTrust* [9]. The paper is organized as follows. In Section 2 we describe our representation of trust [2, 4] and describe our protocol for community-wide trust computation. In Section 3 we present a set of simulations aimed at assessing the efficiency of our solution and comparing it with *EigenTrust*. Finally, in Section 4 we draw the conclusion.

2 Trust and Reputation Protocol

P2PRep is a reputation-based protocol, formalizing the way each peer stores and shares with the community the reputation of other peers [5]. P2PRep runs in a fully anonymous P2P environment, where peers are identified using self-assigned *opaque identifiers* (e.g. a digest of a public key for which only the peer itself knows the corresponding private key). For the sake of simplicity, here reputation and trust are represented as fuzzy values in the interval $[0, 1]$. Our approach can however be readily extended to more complex array-based representations taking

[1] Note that while in this paper we assume reputations to be associated with peers, the approach can also be applied to the exchange of opinions on resources [6] and on many other aspects (e.g., quality of resources, opinions on specified parameters, and so on).

into account multiple features [2]. Protocol P2PRep consists of four phases. In Phase 1, a requestor r locates available resources sending a `Query` broadcast message. Other peers answer with a `QueryHit` message notifying r that they may provide the requested resource. Upon receiving a set of `QueryHit` messages, r selects an offerer o and polls the community for any available reputation information on o sending a `Poll` message. `Poll` messages are broadcasted in the same way as `Query` messages. All peers maintain an *experience repository* of their previous experiences with other peers. When a peer receives a `Poll` message, it checks its local repository. If it has some information to offer and wants to express an opinion on the selected offerer o, it generates a vote based on its experiences, and returns a `PollReply` message to the initiator r. As a result of Phase 2, r receives a set *V* of votes, some of which express a good opinion while others express a bad one. In Phase 3, r evaluates the votes to collapse any set of votes that may belong to a clique and explicitly selects a random set of votes for verifying their trustworthiness [5]. In Phase 4 the set of reputations collected in Phase 3 is synthesized into an aggregated community-wide reputation value. Based on this reputation value, the requestor r can take a decision on whether accessing the resource offered by o or not (Phase 5). After accessing the resource, r can update its local trust on o (depending on whether the downloaded resource was satisfactory or not). While a naive implementation of P2PRep can be expensive in terms of storage capacity and bandwidth, this cost can be minimized by applying simple heuristics. The amount of storage capacity is proportional to the number of peers with which the initiating has interacted. With respect to the bandwidth, it is easy to see that P2PRep increases the traffic of the P2P network by requiring both direct exchanges and broadcast requests. It is however reasonable to assume that the major impact of the protocol on network performance is due to broadcast messages and their answers. To overcome this issue, several heuristics can be applied. For instance, *intelligent routing* techniques can be applied for enabling custom forwarding of poll packets to the "right" peers. Vote caching is another technique that can be applied to improve the effectiveness of P2PRep. Finally, P2PRep scalability depends on the technique used for vote aggregation. Section 3 will present a set of simulations showing the details of P2PRep behavior.

2.1 Reputation model

In P2PRep reputations are managed at two levels: *local* and *community-wide* reputation. Local reputation is based on each individual peer's direct experience of interactions with another peer, while community-wide reputations represent the synthesis resulting by aggregating multiple opinions about a peer. In the remainder of this Section, following [4] and [2], we recall our fuzzy technique for computing local and community-wide reputations.

Let $r_{i,j}$ be the local reputation resulting from direct interactions between peer i and peer j. For each interaction, we model the transaction outcome $t_{i,j}^{(n)}$ as follows: $t_{i,j}^{(n)} = 1$ if the outcome was satisfactory, $t_{i,j}^{(n)} = 0$ otherwise. We use a fuzzy value to express local reputations to take into consideration the fact that transactions can be heterogeneous for importance, resource value, and so on. Each local reputation is initialized after the first interaction by taking the value of $t_{i,j}^{(1)}$. At any time $n > 1$, the local reputation is updated based on the outcome of the n-th transaction as follows.

$$r_{i,j}^{(n)} = \begin{cases} t_{i,j}^{(n)} & \text{if } n = 1 \\ \alpha^{(n)} r_{i,j}^{(n-1)} + (1 - \alpha^{(n)}) t_{i,j}^{(n)} & \text{if } n \geq 2 \end{cases} \quad (1)$$

where $\alpha^{(n)}$, a value between 0 and 1, is the aggregation *freshness*,[2] that is, the importance of past transactions in relation with the last one. If $\alpha^{(n)} \simeq 1$ past experience will have a very high importance and the last transaction has a little role in reputation evaluation; if $\alpha^{(n)} \simeq 0$ then last experience is merely forgotten. Note that our freshness value is not static, but can change at any single interaction, according to circumstances. In particular, for the first encounters of i with j, (i.e., for low values of n), freshness should remain high while it can decrease as n grows and therefore i has acquired enough experience on j. Note however that freshness should never become too low, since this would mean having a blind trust in other peers. In our approach, freshness evolution relies on feedback, by checking whether the current reputation value of a peer would give an accurate prediction of the result of the next transaction with it. If this is the case, reputation value is about right and freshness should not increase; otherwise, the current reputation is considered unreliable and α is incremented. Our approach follows a well-known technique for feedback control, that quickly stabilizes to a fair and efficient setting [8]. While other feedback strategies such as *MIMD* (Multiplicative Increase/Multiplicative Decrease) could be adopted, they are known to be less robust and need careful tuning to avoid oscillatory behavior [3]. More specifically, if we consider the reputation $r_{i,j}^{(n-1)}$ to be a prediction of the outcome of i's next download from j, the *accuracy* of this prediction can be computed with a Boolean function which returns 0 (wrong prediction) if the value of $r_{i,j}^{(n-1)}$ differs from the actual outcome $t_{i,j}^{(n)}$ more than a given error threshold E,

[2]Note that different values of freshness are used for different peers and therefore a peer i will use a different α for each different peer j. For readability, we omit the subscripts when they are clear from the context.

and returns 1 (right prediction) otherwise. Namely,

$$\mathrm{Acc}_{i,j}^{(n)} = \begin{cases} 1 & \text{if } \mid r_{i,j}^{(n-1)} - t_{i,j}^{(n)} \mid < E \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

This accuracy value is then used to determine a coefficient $\beta^{(n)}$ taking into account past experience and the outcome of the last transaction as follows. The coefficient is initialized to $\beta^{(0)} = 0$ and is updated at subsequent times as:

$$\beta^{(n)} = \frac{\beta^{(n-1)} + Acc_{i,j}^{(n)}}{2} \quad (3)$$

Finally, $\alpha^{(n)} = \frac{\beta^{(n)}}{2}$.

If the degree of similarity $Acc_{i,j}^{(n)} \simeq 1$, then $\alpha^{(n)}$ will increase, granting more importance to past history[3].

If the requestor has no previous experience on a peer j or the local reputation is still considered not reliable enough (i.e., for low values of n), the peer will, by using P2PRep run a poll and inquire other peers about j's reputation for them. Here, we assume the vote expressed by each peer k participating in the poll on j to be its local reputation $r_{k,j}$ of j. The question now is how the poller should aggregate the different votes received to produce a synthesized value.[4] A basic requirement for aggregating opinions is that if the pool peers is stable and they maintain identical beliefs across all transactions from a given instant t_0 onwards, then all interactions will asymptotically conform to these beliefs. In other words, if the majority of peers considers that peer j has a bad reputation, then j will be in the end excluded from all transactions.[5] This property, called *unanimity*, is often considered a minimal standard of acceptability for an opinion aggregation operator, and is held both by the weighted mean and the geometric mean [1]. The simplest aggregation available is the arithmetic average of the votes received. However, arithmetic average performs a rough compensation between high and low values, not taking into account different variations between individual opinions that may characterize different polls. Luckily, arithmetic means are not the only aggregation functions usually used in opinion pooling and many other combination methods had been studied. The *OWA (Ordered Weighted Average)* operator, introduced by Yager in [13], allows the decision maker to give different importance to the values of a criteria. The main difference between OWA and the arithmetic means consists in the *separability* of the aggregation function: OWA considers that the influence of each contribution on the result is not directly separable, but depends on the other contributions. For instance, a very good reputation value in the midst of low ones should be treated differently than a good value accompanied in the poll by fair ones. Technically, an OWA operator is a weighted average that acts on an ordered list of arguments and applies a set of weights to tune their impact on the final result. Namely, in our setting, we get

$$\lambda_{OWA} = \frac{\sum_{k=1}^{n} w_k r_{t_k,j}}{\sum_{k=1}^{n} w_k} \quad (4)$$

where n is the number of reputations to be aggregated considered in decreasing order, that is, assuming $r_{t_1,j} \geq r_{t_2,j} \geq ... \geq r_{t_n,j}$ and $[w_1\ w_2 \ldots\ w_n]$ is a weighting vector. The behavior of this operator is largely determined by the choice of weights. For instance, the result could be based on the most frequent values simply by assigning lower weights to extreme ones. Alternatively, high weights can be given to extreme values to increase the operator responsiveness to them.[6] In our case, we set the OWA weights *asymmetrically*, since our aggregation operator needs to be biased toward the lower end of the interval, increasing the impact of low local reputations on the overall result. The reason is that we assume that peers are usually trustworthiness and a malicious behavior is the exception. According to this assumption, low local reputations should be considered as relevant and their impact on the overall reputation should be significant. Of course, there are many ways to do this, for example, by increasing weights linearly or non-linearly with the position k of the corresponding opinion $r_{t_k,j}$ in the OWA ordered set of arguments. Also, it is possible to give a bonus to multiple occurrences of the same weight (*group votes*). This can be done by defining the operator on a (usually small) set of different reputation values d rather than on the (usually large) number of peers, as follows:

$$\lambda = \frac{\sum_{i=1}^{d} w_i (v_i)^{1/|V_i|}}{\sum_{i=1}^{d} \mid V_i \mid w_i} \quad (5)$$

where $V_1, \ldots, V_d$ is a partitioning of the set of votes grouping together votes with the same value v_i, and where v_i are considered ordered, that is, $v_1 > \ldots > v_d$.

Since our local opinions take values in the unit interval $[0, 1]$, in principle there is no reason to favor group votes; however, computational efficiency and values' rounding to a fixed number of decimal may make this a viable solution for practical implementations. In the algorithm shown

[3]Note that, of course, this accuracy could also be defined as a fuzzy function, for example, by considering that not all transactions have the same importance. We shall not elaborate on this possibility in this paper, assuming that the crisp β coefficient summarizes all context representation.

[4]Note that since the individual reputations are fuzzy, their fuzzy aggregation will also be a value in the unit interval.

[5]Of course, the delay can be arbitrarily long if communications on the network are slow.

[6]With the OWA operator the decision makers can express their preferences in relation of the values of the criteria, they cannot express preferences between criteria. This drawback is important when criteria are heterogeneous and is usually solved using the *WOWA (Weighted OWA)* operator [12] instead of OWA. Here, however, we are in a homogeneous scenario, so using OWA looks perfectly safe.

below, we simply partition the unit interval in $d+1$ subintervals and use their extreme values (discarding 0 and 1) as a (linearly increasing) set of weights for aggregating the d distinct reputation values to be aggregated via the OWA operator. In other words, we set:

$$\lambda = \frac{\sum_{i=1}^{d} \frac{i}{d+1} v_i \mid V_i \mid}{\sum_{i=1}^{d} \frac{i}{d+1} \mid V_i \mid} \quad (6)$$

We include local reputation in the computation by adding a new class $V_{d+1} = r_{p,j}$ and associate with it the highest possible weight, with weights now computed for $d+1$ values. The final definition of our aggregation operator for P2PRep is the following one:

$$R_{p,j} = \frac{\sum_{i=1}^{d+1} \frac{i}{d+2} v_i \mid V_i \mid}{\sum_{i=1}^{d+1} \frac{i}{d+2} \mid V_i \mid} \quad (7)$$

3 Simulation Model

While implementations of P2PRep are available [6], our protocol's large-scale efficiency and effectiveness as compared to other proposed approaches can only be assessed via simulation. In this Section, after describing the simulation model we adopted for P2PRep we describe the simulation experiments that we carried out to investigate the efficiency of our solution, reporting and discussing the main results obtained.

3.1 Model description

Our simulation model refers to a P2P network where each peer is reachable from all others[7] and does not take into account delays due to message routing. Over this broadcast network, we simulate a set of queries, each asking for a randomly chosen resource. For each query, the peer querying the network is randomly chosen (with a uniform probability distribution) over all available peers. Then, a preferred offerer o is selected, randomly choosing some peers among those having the resource required. In our simulation, a malicious peer is more likely to be selected as the offerer o than a well-behaved one. The main settings of our simulation model are the following: the number of peers P in the network is uniformly distributed in $[300, 400]$; the number of malicious peers M, $M \subset P$ is the 40% of $|P|$; the number of different kinds of resources is 20; the $\max$ poll cardinality is uniformly distributed in $[5, 15]$. We also assume all well-behaved peers i participate in a poll on offerer o by returning the local reputation $r_{i,o}$ if such a value is recorded; no response is returned otherwise. Moreover, we modelled

[7] In real P2P systems this condition is only verified within a fixed *horizon*.

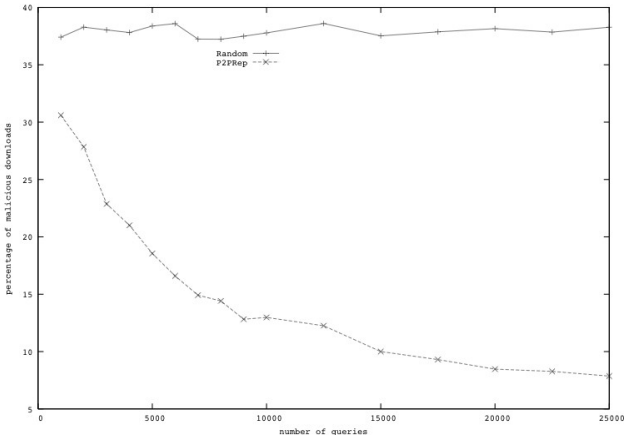

Figure 1. The performance of our solution

the behavior of malicious peers in M by assuming that: *1)* malicious peers provide only malicious resources; *2)* malicious peers respond to the polling on a peer o by always providing a (malicious) 1 reputation if $o \in M$, and by providing a genuine opinion, otherwise. Our simulation consists of a number of repeated experiments, each one evaluating a different and randomly generated scenario. We have set the total number of experiments to 50 while the number of queries for each experiment ranges from $1,000$ to $10,000$ with an increment of $1,000$. Higher values are not infrequent on real P2P systems like Gnutella [6]; however we chose this range to enforce the assumption that the set of peers remains more or less stable across the experiments. To perform a simple comparison, the model provides also a random policy in which the offerer is randomly chosen and reputation checks are not performed.

Number of queries	OWA	Arithmetic mean
5000	18.55	25.02
7500	14.01	20.88
10000	12.98	19.15
12500	12.25	18.76
15000	10.00	16.54
17500	9.30	15.81
20000	8.47	15.05
22500	8.27	14.63
25000	7.86	14.21

Table 1. *OWA* vs. arithmetic mean: Percentage of malicious downloads

The performance of P2PRep with fuzzy aggregation is shown in Figure 1: the fuzzy solution has a slow start but the percentage of malicious downloads decreases as the quality of the network reputation increases following the dif-

fusion of information about malicious peers. Although it steadily decrease, we note that after a number of queries (about 15000 in our experiments) the percentage of malicious downloads tends to become more stable.

Table 1 reports a comparison between OWA and arithmetic means showing the beneficial impact of using a fuzzy "intelligent" aggregation operator with respect to a flat one like arithmetic mean.

3.2 Comparisons with EigenTrust and Random Policy

We now compare the behavior of P2PRep with fuzzy aggregation with respect to *EigenTrust* [9], a probabilistic approach to trust computation. In our experiments we have set the number of pre-trusted peers equal to 5% of $|P|$. The results are reported in the Figure 2.

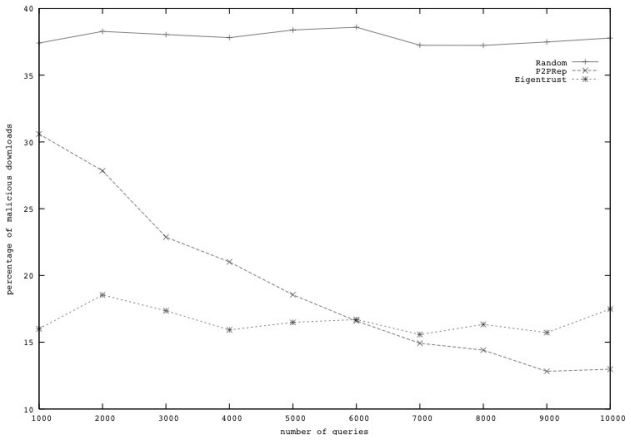

Figure 2. Comparison with Eigentrust

Our analysis shows that EigenTrust ensures good performance even for a small number of transactions. However, the very same results show that a single variation has a small impact and EigenTrust cannot improve much over time. Although it has a slower start, our fuzzy solution overtakes EigenTrust after 6000 queries.

3.3 Changing P2P population

We are now ready to investigate the effects on P2PRep of an high mortality rate, which is a typical feature of P2P environments. Namely, we consider three different scenarios. In scenario S_1 we increase the number of malicious peers by changing rogue peers into well-behaved ones. Vice versa, in S_2 the number of rogue peers is decreased by changing them into well-behaved peers. Finally, the third scenario S_3 is a mix of S_1 and S_2, that is, rogue peers change into well-behaved ones and vice versa. Referring to our simulation model, we introduce two new further parameters: a population change is carried out every q_{change} queries and a peer changes its status with p_{change} of probability. Table 2 reports the results of our experiments setting $q_{change} = 2500$ and $p_{change} = 10\%$. Column "no changes" refers to the scenario in which the status of peers do not change. The first two scenarios models the case in which peer' population drastically changes. In S_1, the number of malicious downloads using P2PRep are about one half of those with random policy. On the contrary, in S_2 they decreases up to five times. Scenario S_3 models high turnover in peer' population: P2PRep confirms its robustness showing a percentage of malicious downloads greater about 1% than scenario with no changes. Finally, we observe that comparison with EigenTrust is not possible since it requires stable peer' population in order to guarantee the convergence of the probabilistic model.

number of queries	no changes		S_1		S_2		S_3	
	rand	P2PRep	rand	P2PRep	rand	P2PRep	rand	P2PRep
10000	37.78	12.98	47.05	21.72	33.14	11.87	38.18	13.32
12500	38.60	12.25	49.95	25.08	31.61	9.68	37.73	10.56
15000	37.52	10.00	51.88	24.65	29.98	8.75	37.78	9.50
17500	37.87	9.30	53.93	24.30	28.33	6.99	38.40	9.17
20000	38.15	8.47	55.84	26.80	26.53	6.13	38.02	8.72
22500	37.85	8.27	57.70	26.21	25.57	5.77	38.27	8.81
25000	38.27	7.86	59.65	28.85	24.99	5.17	37.63	8.16

Table 2. Scenarios evaluation: Percentage of malicious downloads

4 Conclusions

Voting systems are at the base of the design of reputation systems in fully anonymous P2P environments. To be efficient as well as effective, vote collection and aggregation must rely on advanced flexible techniques for collecting and aggregating peers' opinions. In this paper, we discussed the efficiency of our voting protocol P2PRep when used in association with a OWA fuzzy aggregation operator, comparing it with *EigenTrust* a probabilistic approach. Further research will be aimet at improving the simulation system including a more complex network model to better evaluate the impact of communication overhead.

Acknowledgments

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian Ministry of Research Funds for Basic Research (FIRB) within the KIWI and MAPS projects.

References

[1] J. Aczél and C. Alsina. On synthesis of judgements. *Socio-Econom Planning Science*, 20(6):333–339, 1986.

[2] R. Aringhieri, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *Journal of the American Society of Information and Software Technology*, 2005. To appear.

[3] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance. *Journal of Computer Networks*, 17(1):1–14, 1989.

[4] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, M. Pesenti, P. Samarati, and S. Zara. Fuzzy logic techniques for reputation management in anonymous peer-to-peer systems. In *Proc. of the Third International Conference in Fuzzy Logic and Technology*, Zittau, Germany, 2003. Zittau, Germany.

[5] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in P2P systems. *IEEE Transactions on Data and Knowledge Engineering*, 15(4):840–854, 2003.

[6] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proc. of the 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, 2002. Washington, DC, USA.

[7] E. Damiani, R. Khosla, and W. Grosky. *Human-centered E-business*. Kluwer Academic Publisher, 2003.

[8] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 18(4):314–329, 1988.

[9] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of the Twelfth International World Wide Web Conference*, Budapest, Hungary, 2003. Budapest, Hungary.

[10] G. Klir and T. Folger. *Fuzzy Sets, Uncertainty, and Information*. Prentice-Hall, 1988.

[11] S.Sen and N. Sajja. Robustness of reputation-based trust: Boolean case. In *Proc. of AAMAS*, pages 288–293, Bologna, Italy, 2002. Bologna, Italy.

[12] V. Torra. The weighted owa operator. *International Journal of Intelligent Systems*, 12(2):153–166, 1997.

[13] R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[14] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, USA, 1993. Oakland, CA, USA.

[15] B. Yu and M. Singh. A social mechanism for reputation management in electronic communities. In *Proc. of the 4th International Workshop on Cooperative Information Agents*, Boston, USA, 2000. Boston, USA.