

# Establishing Trust Between Mail Servers to Improve Spam Filtering

Jimmy McGibney and Dmitri Botvich

Telecommunications Software & Systems Group, Waterford Institute of Technology,  
Waterford, Ireland.  
{jmcgibney, dbotvich}@tssg.org

**Abstract.** This paper proposes a new way to improve spam filtering based on the establishment and maintenance of trust between mail domains. An architecture is presented where each mail domain has an associated trust manager that dynamically records trust measures pertaining to other domains. Trust by one mail domain in another is influenced by direct experience as well as recommendations issued by collaborators. Each trust manager interacts with local spam filtering and peer trust managers to continuously update trust. These trust measures are used to tune filter sensitivity. A simulation set-up is described with multiple nodes that send and receive mail, some of which is spam. Rogue mail servers that produce spam are also introduced. Results of these simulations demonstrate the potential of trust based spam filtering, and are assessed in terms of improvements in rates of false positives and false negatives.

## 1 Introduction

Unsolicited bulk e-mail, or spam, is probably the greatest single nuisance for users of the Internet. Despite significant anti-spam efforts and the development of powerful spam filtering technologies, the incidence of spam remains stubbornly high. Estimates of the incidence of spam as a proportion of total email traffic vary widely, with the highest estimates close to 90% and the lowest still above 50%.

The main anti-spam techniques in practical use are based on message content, domain name system (DNS) blocklists and collaborative filtering databases. *SpamAssassin* [1], for example, processes each incoming mail and assigns it a score based on a combination of values attributed to possible spam indicators. The higher the score, the more likely it is that the mail is spam. A score threshold is then used to filter mail. DNS blocklisting is a complementary approach where mail is simply filtered based on where it comes from.

Countering spam is difficult though. Spammers are quite resourceful and adapt to filtering advances. Anti-spam techniques also evolve and improve to meet these new challenges, but there is usually some delay in propagating updates. With content filters, it is difficult to avoid having false positives and false negatives. A false positive occurs when a genuine mail message scores above the threshold and is flagged as spam. A false negative occurs when spam scores below the threshold and is accepted. There are also difficulties with blocklists, such as when well-intentioned mail servers

are attacked and exploited, or when they fall somewhere in between – for example, where servers are well managed but client machines are not patched frequently and are at risk of hijack by spammer rootkits.

This paper proposes a new approach to establishing and maintaining trust between mail domains and its application to improve spam filtering. In this approach, mail domains dynamically record trust scores for other mail domains; trust of one in another is influenced by direct experience of the domain (i.e. based on mail received from its server) as well as recommendations issued by collaborating domains. As well as modelling trust interactions between mail domains, we explore how mail filtering can use these trust values with existing mail filtering techniques. We also consider the case of rogue mail domains that issue false recommendations. We also report on experimental simulations that measure the effectiveness of our approach by examining to what extent we have achieved a reduction in false positives and false negatives.

Note that we focus on mail transfer agents (MTAs) rather than user clients in this paper. We interchangeably use the terms *mail server*, *mail domain* and, simply, *node* in place of MTA throughout this paper.

The remainder of this paper is organised as follows. The next section summarises related work and establishes the novelty of the work reported upon in this paper. In section 3, we summarise the principles of trust management that relate to this work, identify requirements for a trust-based anti-spam system and describe the architecture of our system. Section 4 discusses simulations that attempt to assess the system's effectiveness and section 5 concludes the paper.

## 2 Related Work

Several innovative new anti-spam techniques have been proposed, though without widespread practical application as yet. An example is the use of micro-payments for sending mail. The idea is that cost is negligible for normal users but punitive for bulk mail [2]; in one variation, the cost is more significant but is refundable if not spam [3]. Another idea is to require the sending client to solve a computational challenge for each message, greatly slowing bulk mail generation. Additional tricks, such as obfuscation of published email addresses and the use of *human interactive proofs* [4] in email account creation systems, try to frustrate spammers by making it harder to automate their activities.

The remainder of this section discusses other work that specifically uses collaborative techniques to fight spam.

Some widely-implemented spam filters use *centralised* trust information. For example, *SpamAssassin* [1] has a facility to make use of, in addition to other measures, collaborative filtering databases where trust information is shared and used to help to detect spam. Our approach differs from this, in that trust information is in our case generally managed independently by each mail domain and shared as desired. If required though, a centralised database may be modelled as a node in our network, and each mail domain may assign a trust value to it if it wishes.

Kong et al. [5] present a collaborative anti-spam technique that is different from ours in that the system focuses on end user email addresses. When a user flags a mail

as spam, this information is made available to other users' spam filters, which is useful as the same spam messages are usually sent to a large number of users. Golbeck and Hendler [6] present a technique, inspired by social networks, that allows end users to share ratings information. This technique requires direct user interaction. Neustaedter et al. [7] also use social networks, but to assist more general email management, not limited to spam filtering. Han et al. [8] also focus on users that generate spam, but more specifically *blog comment spam*.

Foukia et al. [9] consider how to incentivise mail servers to restrict output of spam. Their approach is agent based – each participating mail server has an associated Federated Security Context Agent that contributes to, and draws on, an aggregated community view. They also use quotas to control the volume of mail output by a server in an attempt to prevent temporary traffic bursts that are typical of spammer activity.

Though the idea of auto-tuning spam filters is not entirely new, our method of auto-tuning is different from other approaches, for example from Androutsopoulos et al. [10] who use game theory to model interaction between spammers and email users.

### 3 Trust-Based Approach to Spam Protection

#### 3.1 Modelling Trust

A well-known definition of trust is “*a particular level of the subjective probability with which an agent will perform a particular action*” (Gambetta, [11]). Trust is primarily a social concept and, by this definition, is personalised by the subject.

Almost any transaction between entities requires the establishment of trust between them. The decentralised nature of many Internet services means that a model of trust is necessary for effective operations. The scope for hierarchical top-down solutions is limited due to the lack of centralised control, the desire for privacy or anonymity, and the increased use of services in a one-off ad hoc fashion. We can identify some specific requirements for a trust-based anti-spam system:

- *Compatibility with existing infrastructure.* The existing email system should not require changing.
- *The meaning of trust.* Trust is defined as being between two nodes, in this case mail servers – i.e. node  $i$  has a certain level of trust in node  $j$ . Each node has access to a measure of level of trust in each other node. Each node should be capable of managing its own trust level independently.
- *Trust updates based on experience.* It is possible for trust to be built up by experience. Although there may initially be little or no trust between node  $i$  and node  $j$ , it must be possible to establish trust based on interactions between them.
- *Trust updates based on recommendations.* Node  $i$ 's level of trust in node  $k$  may be influenced by node  $j$ 's level of trust in node  $k$  (communicated by node  $j$  to node  $i$ ).
- *Robustness against attack, including collaboration between spammers.* Spammers tend to adapt to new anti-spam systems. For a new trust-based approach to be effective, it should be difficult for spammers to corrupt it. This could include the possibility of spammers actively participating in the trust system, possibly in concert, issuing false recommendations.

- *Stability.* The system should be stable – i.e. consistent treatment of mail, few oscillations in state, and rapid convergence to new state on changes. There should be a way to allow a poorly behaved domain to regain trust following recovery from a hijack or change in its administration policy.

### 3.2 Trust Overlay Architecture

We propose the overlay of a distributed trust management infrastructure on the Simple Mail Transfer Protocol (SMTP) mail infrastructure, with a view to using trust information at each domain to assist with spam filtering. Note that we do not propose any changes or extensions to SMTP or other mail protocols – rather, mail domains exchange mail as normal, but this is overlaid with a *trust management layer*. For each mail domain, there is a logical *trust manager* operating at this layer.

This trust management overlay works as follows. Each mail domain with associated spam detection, operating at the mail transport layer, records mail statistics including incidences of spam and reports this direct experience to its associated trust manager. The trust manager uses this data to form measures of trust about the sources of mail reported upon. This local trust information is then shared with other peer trust managers in the form of recommendations. Each trust manager independently decides how to handle these recommendations, using a trust transitivity algorithm to assist in differentiating between recommendations from well-intentioned nodes and those from those that are unreliable or deliberately false. This trust information maintained by the trust manager is then fed back to the mail domain at the mail transport layer to allow it to re-tune its spam filters (typically by raising or lowering thresholds based on trust) to be more effective.

The mechanics of these interactions requires a trust management overlay protocol. The mechanics of such a trust management overlay protocol, and its associated architecture, have already been described by these authors [12]. There are three major types of communications, as follows:

1. *Experience report: Mail host* → *Trust manager*. The mail host has an associated spam filter. All mail is processed by this spam filter and flagged as spam or else accepted. This experience information is made available to the trust manager.
2. *Trust recommendation: Trust manager* ↔ *Trust manager*. Nodes' trust managers collaborate to share trust information with one another. Trust information from a third party node may be based on its experience of mail received from the node in question and/or reputation information that it has gleaned from other nodes. This relates to trust transitivity.
3. *Policy update: Trust manager* → *Mail host*. The third part of this collaboration architecture is responsible for closing the loop. Direct experience is recorded by nodes and shared between them. The result of this experience and collaboration is then used to inform the mail host to allow it to operate more effectively.

**Initialisation.** Note that choosing an initial value of trust to assign to such new arrivals is a non-trivial task. The first option is to initialise the trust level at zero so that nodes have no trust initially and must earn it, thus removing the threat of the so-

called *Sybil attack* [13]. The second option is to assume some *default trust* exists and initialise the trust level at some value greater than zero for previously unknown nodes. The great benefit of Internet mail is in being able to advertise a mail address and receive mail from anyone, even if this person is previously unknown, so the second option is perhaps the better, though we will use simulations and experience to best determine this. In our experiments, we choose a modest initial value of 0.2.

**Distribution of Trust Information.** It is important, for reasons of scalability and reliability of reputation information, to consider how often and to whom trust advertisements are provided.

- *When to issue trust advertisements?* If the trust level in a node is significantly increased or (more likely, due to sudden appearance of spam) decreased, then this should be advertised. Trust advertisements may also be issued periodically.
- *To whom?* Trust advertisements are restricted to nodes that are defined as within the neighbourhood of the issuer. A node may define its neighbourhood for trust advertisements as it wishes. The set of neighbours could contain, for example, most frequent contacts, most trusted contacts, most reliable recommenders, or nodes located nearby.

### 3.3 Using Trust Scores to Filter Email

Assume that a spam filter applies some test to incoming email. In each case, a decision is made whether to accept the mail. A negative result (in the spam test) means that the mail is accepted and a positive result means that the mail is marked as spam.

Most spam filters combine a variety of measures into a suspicion score and compare this with a *pre-defined* threshold. This threshold is a fixed value, which may be tuned manually. In our system, we attempt to improve spam filtering by allowing the threshold to vary. The threshold level depends on the trustworthiness of the node that sent the message. So, we use the sender's trust score (as perceived by the receiver) to define the threshold – i.e. the more trusted a node is, the higher we set the threshold for marking a received mail message as spam. There are several ways to cause this threshold to vary. In our initial experiments, the threshold for mail received from a domain is simply a linear function of the trust score of that domain (as the mean of the trust score range is 0.5 and the default threshold for *SpamAssassin* is 5, we set the threshold to be simply ten times the trust score).

The motivation for this is that mail received from nodes that are known to be well managed and are unlikely to produce spam has an increased likelihood of passing through the filter, reducing the incidence of false positives. There is also scope for reduced processing load if less stringent spam filtering is required in these situations.

In practice, the dynamics of trust applied to spam filtering allows an organisational mail domain to process email for spam in a way that depends on its trust in the sending node. In some cases, this trust level will be somewhere in the middle, between trusted and untrusted. Table 1 illustrates the possible effects of maintaining trust scores for a variety of mail domains, and a desired implicit classification.

**Table 1.** Possible effects of recording trust scores for a variety of mail domains

Trust score range	Category of mail domain	Spam filter threshold (Trust score * 10)	Effect
1.0	Internal host	10	All mail accepted
0.8 – 1.0	Business partner	8 – 10	Most mail accepted
0.7 – 0.8	University	7 – 8	Mail checked for spam but mostly ok
0.5 – 0.7	Popular ISP	5 – 7	Mail checked for spam
0.3 – 0.5	Public webmail service	3 – 5	Mail checked thoroughly for spam
0.1 – 0.3	Lazy configuration	1 – 3	Spam quite likely
0 – 0.1	Open mail relay; known spam source	< 1	Most mail flagged as spam

## 4 Simulations and Results

This section reports on initial simulations that aim to evaluate whether this new proposed approach is likely to be worthwhile.

In summary, the objectives of these experiments were to see if the rates of false positives and/or false negatives could be reduced, as well as to observe the overall stability of the spam filtering system as trust levels converge to steady-state levels.

### 4.1 Generation of Test Emails

Email traffic is inhomogeneous in practice. Some mail domains are much more active than others. Mail domains tend to communicate mostly with selected mail domains for business, social or geographical reasons. Spammers produce email in vast quantities compared to regular email users.

In our simulations, each node has a *neighbourhood* defined. This is a set of nodes that are somehow close to the node in question, with the expectation of above average frequency of communication.

Our generated email has the following statistical properties. For some experiments, every node is a neighbour of every other node. For other experiments, we use a sparse neighbourhood definition. This second, more sparse, neighbourhood of each node is built randomly as follows:

1. Initially, the neighbourhood of each node is the empty set.
2. Choose two nodes at random. Add one to the neighbourhood of the other.
3. Repeat (2) until all nodes are connected via neighbour relations (if the set of nodes is modelled as a graph with neighbourhood relationships as edges, repeat until the graph is connected).

From each node, 50% of emails are sent to a randomly chosen neighbour. The other 50% are sent to any randomly chosen node.

Each email contains a value  $S'$  that models aggregated indicators of spam, in the style of *SpamAssassin*. This value is used by the receiving node to test for spam.  $S'$  has a Gaussian (normal) probability density function with mean  $\mu$  and standard deviation  $\sigma$ . Mail that is actually spam tends to have a relatively high value of  $\mu$ . Normal mail tends to have a lower value of  $\mu$ . The standard deviation determines the tendency for the filtering system to be prone to false positives and false negatives.

For the purpose of our experiments, whether or not an email is actually spam is indicated by a binary value communicated separately to the receiver. This is not used in spam detection, but is used afterwards in evaluating how well the spam filter worked.

## 4.2 Trust Establishment and Maintenance in the Experiments

We denote the *local trust* that node  $i$  has in node  $j$  as  $T_{i,j}$ ,  $0 \leq T_{i,j} \leq 1$ . Consistent with Gambetta's definition [11], this is the probability assigned by node  $i$  that an email from  $j$  can be trusted to be spam free. Initially,  $T_{i,j} = x$ , where  $x$  is the default trust.

**Updating Local Trust Based on Direct Experience.** On receipt of an email from node  $j$ , node  $i$  applies a test based on its contents and a threshold (which is function of  $T_{i,j}$ ) to determine whether it is spam. Set the binary value  $S$  to 1 if spam is found and 0 otherwise. In our simulations, we use an exponential averaging technique to update the local trust that node  $i$  has in node  $j$  as follows:

$$T_{i,j} = \alpha S + (1 - \alpha)T_{i,j} , \quad (1)$$

where parameter  $\alpha$  that can be viewed as the *rate of adoption of trust*,  $0 \leq \alpha \leq 1$ . Note that having  $\alpha$  set to 0 means that the trust value is unaffected by the experience. Having  $\alpha$  set to 1 means that local trust is always defined by the latest experience and no memory is retained. The higher the value of  $\alpha$ , the greater the influence of recent mails from a node on the trust value recorded for that node. Lower values of  $\alpha$  encourage stability of the system. If a succession of experiences of mail from node  $j$  return the same spam determination,  $S$ , then the trust value  $T_{i,j}$  converges towards  $S$  (towards 0 for a sequence of spam or towards 1 for a sequence of normal messages).

**Updating Local Trust Based on Recommendation by Another Node.** Node  $i$  may receive a message from a third party node,  $k$ , indicating a level of trust in node  $j$ . This can be modelled as node  $i$  adopting some of node  $k$ 's trust level in node  $j$ . As well as introducing a new parameter  $\beta$  indicating the level of influence of recommender trust on local trust, we also use  $T_{i,k}$ , how much domain  $i$  trusts domain  $k$ . In our simulations, we use an exponential averaging technique to update trust as follows:

$$T_{i,j} = \beta T_{i,k} T_{k,j} + (1 - \beta T_{i,k}) T_{i,j} , \quad (2)$$

where  $\beta$  is a parameter indicating the level of influence that recommender trust has on local trust,  $0 \leq \beta \leq 1$ . Note that, the larger the value of  $T_{i,k}$ , (i.e. the more  $i$  trusts  $k$ ), the greater the influence of  $k$ 's trust in  $j$  on the newly updated value of  $T_{i,j}$ . Note that, if  $T_{i,k} = 0$ , (i.e.  $i$  has no trust in  $k$ ), this causes  $T_{i,j}$  to be unchanged.

### 4.3 Initial Results and Analysis

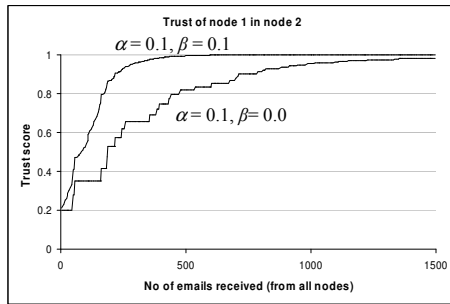
In this subsection, we present initial results of our experiments. Firstly, we examine how each node draws on both direct experience and recommendations from collaborating nodes to reach a stable trust evaluation of another node. Secondly, we observe the effects of feedback of trust values for more effective spam filtering.

**Convergence of Trust.** How trust converges to stable values in our system depends on a number of factors, such as the size of the network, the means of updating trust, the extent of spam, the average size of each node’s neighbourhood, and the starting point (how trust scores are initialised). The means of updating trust relates to whether we use direct experience only or how a mix experience and reputation inputs are combined. In our system, we use exponential averaging with parameters  $\alpha$  and  $\beta$  respectively, so the choice of these values influences convergence.

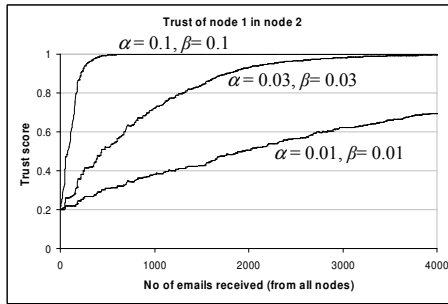
Results shown here vary just some of these parameters. Each of the figures below is based on a network of fifty nodes. The default trust is set to 0.2 for all nodes.

Fig. 1 compares the option of using direct experience only to update trust with the combination of direct experience and frequent recommendations from neighbours. In this example, there are fifty good nodes in the network and there is no spam, meaning that all trust values should eventually converge to 100%. Setting parameter  $\beta$  to zero removes any effect of recommendations. Note that, in the direct experience only case ( $\alpha = 0.1, \beta = 0$ ), trust level converges less smoothly – each jump occurs when a (non-spam) mail is received from the node in question. Recommendations allow trust values to be updated on receipt of mail by *another* node.

Fig. 2 examines the effect of the sizes of parameters  $\alpha$  and  $\beta$  on convergence. Not surprisingly, the higher their values, the faster is convergence. It is important not to set these values too high though to avoid the risk of trust values oscillating widely.



**Fig. 1.** Effect on convergence of using recommendations to update trust score



**Fig. 2.** Effect on convergence of varying parameters  $\alpha$  and  $\beta$

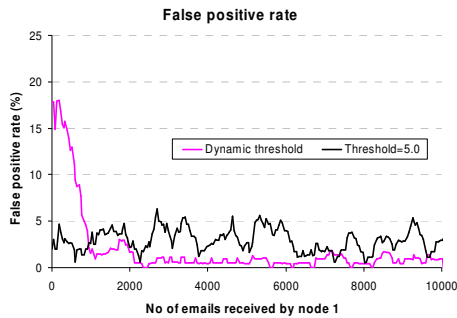
These illustrations just consider trust between good nodes. What happens when some nodes occasionally produce spam (perhaps due to poor configuration, or due to relatively open access policies)? Experiments with a small number of nodes producing a varying level of spam have shown that the trust recorded by a good node for each of these unreliable nodes converges to a different value, depending on the level

of spam. This is encouraging as we can envisage trust filters that work by subjecting mail to a degree of scrutiny that is appropriate to the trust level in the source node.

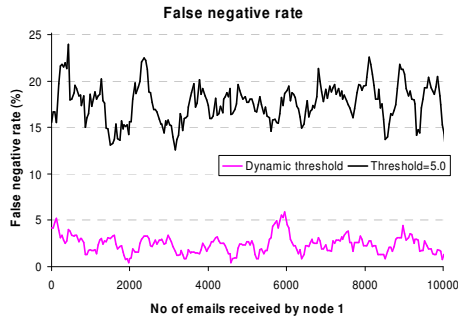
**Influence of Trust on Effectiveness of Filtering.** The main objective of the work described in this paper is to see if we can get an improvement in the effectiveness of spam filtering by applying trust scores. The figures below show how this has been achieved in an illustrative case. In this experiment, a network of fifty nodes is chosen; also there is a single spammer who is responsible for 50% of all email generated in the system. Trust convergence for normal nodes is moderately fast, with parameters  $\alpha$  and  $\beta$  both set to 0.03. Furthermore, the neighbourhood of each node consists on average of one-seventh of all nodes.

For this experiment, we choose relatively flat (but distinct) probability density functions for spam indicators for both spam and non-spam email. For spam mail, the aggregate spam indicator has a Gaussian (normal) distribution with a mean of 8.0 and a standard deviation of 4.0. For non-spam, the mean is 1.0 and the standard deviation is 2.0. This attempts to model the fact that parameters of non-spam mail tend to deviate less than those of spam (and hence fewer false positives than false negatives).

As already mentioned, most spam filters combine a variety of measures into a suspicion score and compare this score with a pre-defined threshold. For our experiments, a fixed threshold of 5.0 is chosen (*SpamAssassin* default) and used as a benchmark. As can be seen in Fig. 3 and Fig. 4 below, a significant reduction in both false positives and false negatives can be achieved with auto-tuning of the threshold (based on trust values). Auto-tuning is of course most effective in a steady-state situation when trust values have stabilised. A range of other predefined threshold values were also tried, but with no better results than the value of 5.0 shown. Choosing a higher predefined threshold causes an increase in false negatives and choosing a lower predefined threshold causes an increase in false positives.



**Fig. 3.** Comparison of dynamic vs. fixed threshold: impact on rate of false positives



**Fig. 4.** Comparison of dynamic vs. fixed threshold: impact on rate of false negatives

## 5 Conclusions and Future Work

We have described a collaboration system for mail domains and how this can be used to make spam filtering more efficient and more effective. The collaboration system is

lightweight, and relies on the decentralised maintenance of simple trust scores by individual mail domains. Stability and speed of convergence are influenced by a number of tuneable parameters, and our initial simulations have investigated various parameter settings for email traffic with certain statistical characteristics.

Our simulations have also shown (again for systems and email traffic with specific statistical characteristics) that using trust measures to dynamically refine spam filter thresholds can improve effectiveness by reducing false positives and false negatives.

Further work is required to assess the applicability of this approach to situations with various topological and email traffic patterns. It is proposed to use real email data sets such as [14] to more realistically model incidence of spam and examine performance issues with our system. There is also significant scope for refinement of the system's dynamics, including how recommendations and new experiences are interpreted and used to update trust scores. Further experiments are required to explore the effects of various ways to define node neighbourhoods. It should also be possible to improve filter throughput as mails from trusted domains may require less processing.

It would also be very interesting to examine how spammers might try to get around this system, either by looking for weaknesses in the system's dynamics, or by collaborating with each other.

## References

1. Schwartz, A.: SpamAssassin. O'Reilly (2004)
2. Goodman, J., Rounthwaite, R.: Stopping outgoing spam. In Proc. ACM Conference on E-Commerce, New York (2004)
3. Abadi, M., Birrell, A., Burrows, M., Dabek, F., Wobber, T.: Bankable postage for network services. In Proc. Asian Computing Science Conference, LNCS 2896 (2003) 72-90
4. Naor, M.: Verification of a human in the loop or identification via the Turing test. Unpublished manuscript: <http://w.wisdom.weizmann.ac.il/~naor> (1996)
5. Kong, J., Rezaei, B., Sarshar, N., Roychowdhury, V., Oscar Boykin, P.: Collaborative spam filtering using e-mail networks. IEEE Computer, Vol. 39, No. 8 (2006) 67-73
6. Golbeck, J., Hendler, J.: Reputation network analysis for email filtering. In: Proc. Conf. on Email and Anti-Spam (2004)
7. Neustaedter, C., Bernheim Brush, A., Smith, M., Fisher, D.: The social network and relationship finder: social sorting for email triage. In Proc. Conf. on Email and Anti-Spam (2005)
8. Han, S., Ahn, Y., Moon, S., Jeong, H.: Collaborative blog spam filtering using adaptive percolation search. In Proc. International World Wide Web Conference. Edinburgh (2006)
9. Foukia, N., Zhou, L., Neuman, C.: Multilateral decisions for collaborative defense against unsolicited bulk e-mail. In K. Stolen et al. (Eds.): iTrust 2006, LNCS 3986 (2006) 77-92
10. Androutopoulos, I., Magirou, E., Vassilakis, O.: A game theoretic model of spam e-mailing. In Proc. Conf. on Email and Anti-Spam, Stanford (2005)
11. Gambetta, D.: Can we trust trust? In: D. Gambetta (Ed.), Trust: making and breaking cooperative relations. Blackwell (1988) 213-237
12. McGibney, J., Botvich, D.: A trust overlay architecture and protocol for enhanced protection against spam. In Proc. Conf. on Availability, Reliability & Security, Vienna (2007) 749-756.
13. Douceur, J.: The Sybil attack. In: Proc. International Workshop on P2P Systems (2002)
14. Klimt, B., Yang, Y.: The Enron corpus: a new dataset for email classification research. Proc. European Conf. on Machine Learning (2004) 217-226