Controlling Spam E-mail at the Routers

Banit Agrawal Nitin Kumar Mart Molle

Department of Computer Science & Engineering
University of California, Riverside, CA, 92521, USA
email: {bagrawal, nkumar, mart}@cs.ucr.edu
Telephone: (+1) 909-787-7354
FAX: (+1) 909-787-4643

Abstract. Like it or not, unsolicited bulk commercial email (aka "spam") has become a regular menu item on the Internet information diet. To combat the daily onslaught of spam clogging people's email inboxes, much work is being done on the development of effective spam control methods, most of which follow the same basic theme of establishing a "front line" of defense at the end-user level. However, dealing with spam is like fighting a battle against a large army; the most effective approach is to employ multiple tactics. Thus, in this paper we propose a method for blocking the supply lines. More specifically, since the daily replenishment of all those in-boxes with new spam consumes a significant amount of network resources, we describe a mechanism to allow network administrators to impose rate controls on bulk email delivery. In our approach, we separate SMTP email delivery traffic from other types of traffic at the router. We then apply a two-step process to the email delivery traffic, which first identifies bulk email streams by comparison with a cache of recently-seen emails, and then uses a Bayesian classifier to decide whether or not a particular bulk emails stream is spam. If a bulk email stream is classified as a spam, we then rate limit it (e.g., no more than 1 copy per minute).

Keywords: spam, bulk email, filtering, rate-limiting, bayesian, router

1 Introduction

The majority of work on controlling email spams is designed to identify spam *after* it has already been delivered to the email inbox belonging to the intended recipient. Using a variety of heuristics, these techniques are quite successful in identifying the bulk email as spam, redirecting the email to a secondary "junk" mailbox, or even deleting the email without human intervention [1–4]. However, even if these techniques are successful — so the spammer's email never reaches the target recipient's eyeballs, nor occupies any disk space on the recipient's mail server — there is nothing to prevent the spammers from wasting large amounts of other people's network bandwidth while attempting to deliver their email.

This network abuse by spammers comes in two different forms. First, the sheer volume of spam email consumes a significant amount of bandwidth from all Internet Service Providers. For example, a recent study by Excedent [2] found that 45% of global email traffic is spam. They also found that major companies are spending

around 30 billion dollars per year in order to eliminate spam emails from their internal networks. Second, some spammers illegally use other people's equipment to send their bulk emails, rather than buying necessary the computers and network connectivity for this purpose. In the past, this was often done by finding a poorly configured mail server that was willing to act as an open mail relay, and sending it one copy of a bulk email message together with instructions to deliver it to a long list of recipients. More recently, however, they have switched to a new strategy of compromising the security of other people's Internet-connected computers, and converting them to remote-controlled spam generators [5]. In both cases, the vast majority of the costs associated with sending spam emails is borne by an innocent third party, rather than the spammer. Therefore, the challenge here is to protect the network resources from abuse by spams, not just the end users.

This paper proposes a mechanism to control spams at the router level. Our approach utilizes the fact that spams are generally sent to multiple recipients with few alterations to a common message content. Thus, our router segregates SMTP mail delivery traffic from other traffic for further processing. This processing could be done within the router, if it were provided with onboard computing resources (for example, by adding a programmable network processor to the network interface line card), or by redirecting the SMTP traffic for offboard processing on a nearby computer (for example, a Linux PC). Whenever we detect email delivery traffic, we invoke the first phase of our algorithm and attempt to match the content of the incoming message against a cache of recently-seen candidate messages. If it succeeds in finding a match, we then invoke the second phase of our algorithm, which consists of Bayesian classification of the bulk message. If the message stream qualifies as spam, we rate-limit its delivery by resetting the TCP session if the elapsed time between consecutive copies falls below our minimum delay threshold. Therefore, this technique introduces the extra costs and burdens to the spammer and minimizes the abuse of the rest of the Internet network traffic.

2 Related Work

Spam is a growing problem for email users, and many solutions have been proposed, from a postage fee for email to Turing tests to simply not accepting email from people you don't know. Spam filtering is one way to reduce the impact of the problem on an individual user (though it does nothing to reduce the effect of network traffic created by spams). In its simplest form, Spam filtering is a mechanism to identify and filter spam messages. Anti-relaying filters prevents the mail server from serving as a promiscuous relay. It does not block incoming spams, but it prevents an authentic mail server from spamming others.

Blocking by IP subnet or number, Blocking on the domain name, Blocking on unresolvable domain names, Header filtering triggered by invalid headers, Checking To: address of the recipient in the header, are some of the common techniques used today. These techniques completely block spam emails. Since the accuracy of these techniques is not 100%, some rare legitimate mails are not delivered. TarProxy is a method for throttling connections between the spammer and an SMTP server by slowing the rate at which a spammer can send spam [6]. The spam filtering techniques based on the

content of the email utilizes the frequency of words, the sequence of words, the frequency of the sequences, weighted values for particular words, weightings for various features such as suspicious "FROM" part, and many more features related to the message headers. These widely used techniques include various forms of Bayesian filtering techniques [1] such as Naive Bayesian classification, Memory-based approach, Markov chains [3], and support vector machines (SVMs) [4]. Bayesian Spam classification technique claims an accuracy level of 99% [1], but it has a shortcoming of considering the independence of features. Our approach is fairly a new idea of detecting and rate limiting spams using an efficient content matching Algorithm based on tunable pattern size and Bayesian classifier. This technique is employed at the router level, which effectively utilizes the network bandwidth, never blocks the legitimate emails, and rate-limit the spam emails.

3 Controlling E-mail spamming at Router

If one user (say "Alice") wants to send an email message to another user (say "Bob"), then Alice's mail looks up the address for Bob's mail server and opens a direct TCP connection to port 25 on Bob's server. At this point the two mail servers carry out the email delivery transaction, according to RFC 2822 [7] and the Simple Mail Transfer Protocol (SMTP), as specified in RFC 2821 [8]. Figure 1 illustrates the complete routing process.

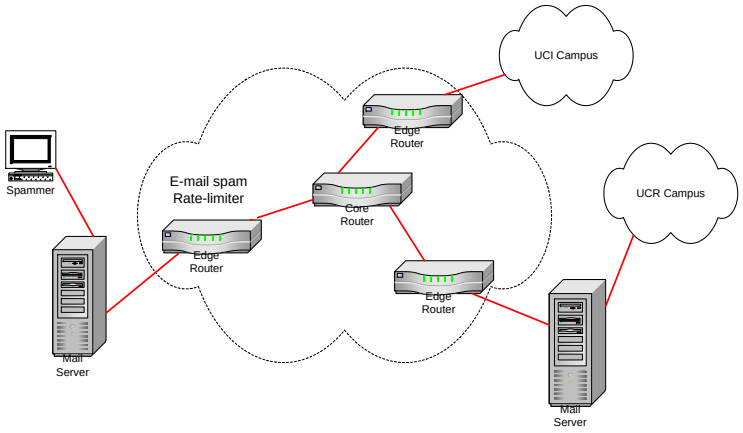

Fig. 1. Email packet Routing in Internet

During this email delivery transaction, the entire dialog between the two mail servers is visible to all routers along the path between Alice and Bob. If any of those routers wanted to block the transaction, it could simply force the TCP session to close by sending a TCP reset segment to both parties. Thus, we have an opportunity for controlling spam at the router level, by:

– *monitoring* all SMTP sessions passing through a router,
– *classifying* each SMTP session as (unappealing) spam or (wholesome) good, and finally
– *policing* the spam traffic to limit its resource consumption.

Note that we are *not* advocating the policy of completely blocking the delivery of all emails that our algorithm classifies as spam, which would be hard to defend and quite possibly illegal. Instead, we suggest imposing a limit on the number of copies of bulk emails we accept per unit time.

Any proposal to increase the amount of processing at an Internet router must include an assessment of the cost of supporting it. Fortunately, special-purpose *network processors* are becoming commercially available for many high-speed networking applications, which offer router architects access to significant processing power, flexibility, and ease-of-use/reuse. Network processors can also be used to do the application layer processing along with other lower layers processing for many diverse networking applications such as content-based load balancing, md5, encryption, hashing etc. Many vendors provide content-addressable memory (CAM) as the co-processors which can be used with the network processor to accomplish high speed data search.

The email rate-limiting can be added to edge routers at low cost using any of the powerful network processors and high-speed CAM co-processors, which provides the flexibility and programmability to the router architects. As another alternative, we can simply configure the router to forward SMTP traffic to an external computer for offline processing, where we apply our rate-limiter algorithm and control the spam email. The rest of the network traffic is undisturbed and sent through high-speed data path.

4 Content Matching Phase

In this first phase of identification process, we attempt to find a match between each new incoming email message and a cache of previously-seen email messages. The goal here is to correctly classify each message as containing either *repeated* or *unique* content, and without excessive computation or storage requirements. Clearly we can only increase the probability of correctly identifying a repeated message by testing it against more samples of known repeated content. However, pattern matching between two repeated messages is expensive because of variable-length header information that is specific to each recipient. Thus, we use sampling (described below) rather than a full text comparison to detect a match, which can lead to a problem of increasing the number of false positives if we test it against too many samples of unique content.

To minimize these problems, we use a two-level cache structure, which exploits the "bulk delivery in a short time span" property of spam emails to bias the content of our cache structure towards repeated messages. The *primary message cache* is used to store one prototype for each of the k most-recently seen *repeated message types* in LRU order, where k is the tunable *primary window size*. We also keep a time-stamp of each messages stored in the primary message window to rate-limit the spam mails. Conversely, the *secondary message cache* is used to store the l most-recently seen *new candidates for a repeated message type* in FIFO order.

Algorithm 1 Content Matching Algorithm

```
1: match←0;
2: count←0;
3: patterns[]←divide incoming msg into fixed-size patterns;
4: for each element of k-size cache do
5:    for each P in patterns[]; do
6:       match←Look for P in the cache element; // Boyer Moore Algorithm
7:       if match then
8:          count← count + 1;
9:       end if
10:   end for
11:   if count >threshold_count then
12:      Do Spam Processing;
13:      Return;
14:   end if
15:   count←0;
16: end for
17: Do Other Processing;
```

These two caches are used in the following way. Whenever the router receives a new email message, it is compared against all stored message prototypes in both caches. If it matches an entry in either message cache, the new message is classified as repeated content and passed to phase 2 for further processing. Thereafter, we transfer the cached message prototype to the primary cache (if the matched entry was in the secondary cache), and then update the LRU structure of the primary cache. If the new email message does not match any stored message prototypes, it is classified as unique content and avoids further spam processing. In addition, we save a copy of the message in the secondary cache in case it is our first prototype for a new stream of repeated messages. Thus, the secondary message cache acts as a filter before the primary message cache, and new repeated message prototypes can only be added to the primary message cache following a match in the secondary cache.

Our content matching algorithm is optimized for detecting spam messages, which are generally sent in bulk to many different users on different networks with a little personalization to the common content. In order to detect this, we partition each new email into multiple fixed-length substrings called *patterns*. (The pattern length is a tunable parameter, which needs to be optimized.) The set of patterns is now tested against all messages stored in the cache; if a sufficiently high percentage of its patterns match a single cached message, then we declare it to contain repeated content.

Boyer Moore's (BM) Algorithm. To find a small pattern in a big string, we employ Boyer Moore's algorithm [9], which is much more efficient in complexity than other brute force algorithms. The Boyer Moore algorithm solves the pattern matching problem by repeatedly positioning the pattern over the text and attempting to match it. For each positioning that occurs, the algorithm starts matching the pattern against the text from the right end of the pattern. If no mismatch occurs, then the pattern has been found, otherwise the algorithm takes a shift that is an amount by which the pattern will

be moved to the right until a new matching attempt is undertaken, where the first character of pattern and leading character in text matches. The complexity of this algorithm is quite good [9]: the best time to find a pattern of length M embedded within a text of length N is $O(N/M)$.

5 Bayesian Spam Classification Phase

In this section, we describe our spam classification method. We used Bayesian classifier to identify spam emails. Bayesian is a simple, self learning and multi-lingual method which takes entire message into account. In general, it consists of two phases - Training and Testing.

Training phase: Each incoming email is reduced into a set of unique token[1] to assemble an initial list of tokens. The count of good and bad emails containing these tokens is maintained in the *initial-token-set*. If an incoming email consists of a new token which is not already present in the *initial-token-set*, then the new token is added to the *initial-token-set* and the count is updated. This keeps our *initial-token-set* always updated with the new words used by spammers. It helps making the algorithm more robust against the new technique adopted by spammers.

Testing phase: A new email message is tokenized to convert it to a set of tokens $\{t_1, t_2,..\}$. Next, use each token t_i (which is present in the *initial-token-set* generated in the training phase) to calculate a conditional probability that this message is spam, given its inclusion of token t_i. New tokens, not already present in the *initial-token-set* are added in the *initial-token-set* and their count is updated. Once the individual probability of each token has been generated, we combine them using the Bayes Theorem [10] to get an overall estimate of spamminess of an email message.

6 Testing

We implemented the complete algorithm as a set of user-level application programs. The source code for the first phase was written in C and the second phase in Java on a Linux platform. The operation of the algorithm within a router was represented by simulating the execution of the router's task scheduler loop. Each time the task scheduler receives another IP packet, it invokes our algorithm. We created a stream of IP packets from publically available archives of spam and good emails.

In the router dispatch loop, we fetch a packet at each simulation step. Then we feed it to the packet classifier. The classifier checks the source and destination port of the packet and if it equals to 25, it send the packet to the SMTP parser. SMTP parser determines whether this packet is a message packet or not. If this is a message packet, it calls content matching algorithm to find if this message is a bulk message by comparing it with messages stored in an email message window.

[1] Token is a selected word from an email message. HTML tags, commonly used English words (i.e articles, prepositions, conjunctions, etc) are not considered as tokens, since they are likely to appear in both good and bad emails.

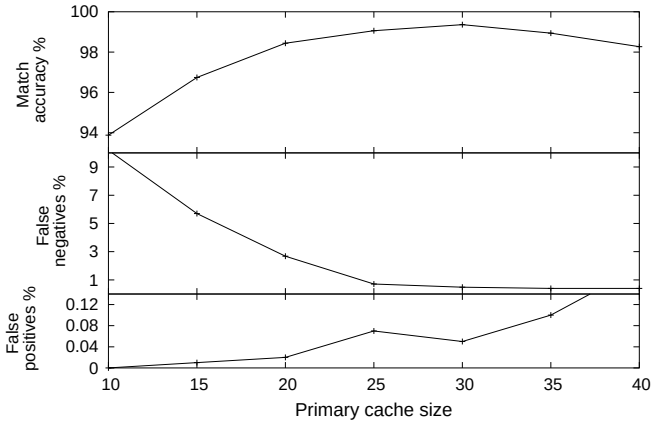

Fig. 2. Tuning the primary message cache size

The training was done using a total of 1513 good and 2401 spam emails obtained from [11] (October 2002). Testing was done using an entirely different set of 1000 emails(both good and spam emails)taken from the archive [12] and from [11](February 2003). The evaluation of the algorithm was based on: a) accuracy; b) percentage of false positives; and c) percentage of false negatives. The Accuracy is defined by the following formula:

$$Accuracy = \frac{1}{2}\left(\frac{Correctly\ classified\ GoodEmail\ Count}{Total\ GoodEmails} + \frac{Correctly\ classified\ Spam\ Count}{Total\ Spams}\right) \cdot 100 \quad (1)$$

6.1 Content Matching Phase

Our content-matching algorithm includes various control parameters, such as primary message cache size, secondary message cache size, pattern size, rate-limiter rate, and matching threshold to fine tune the performance of our content-matching algorithm. The effects of altering these parameters are shown in Figures 2–5.

Varying the size of the message caches. The sensitivity of the three performance metrics to the primary message cache size is shown in the figure 2 for a small corporate environment. We see that the Accuracy is generally quite high, and rises to a broad maximum of approximately 99% for cache sizes between 25 and 35. At the same time, we see that the percentage of false negatives drops significantly (from approximately 10% to almost zero) as we increase the size of primary message cache. However, this improvement is counterbalanced by a much smaller increase in the percentage of false positives (from almost zero to approximately 0.15%). Thus, keeping in mind that the processing time for phase I increases linearly with size of the primary cache, we conclude that a moderate cache size of about 30 messages can be used for a small corporate environment. In future, we are planning to deploy this technique in a real-life environ-

ment to assert more pragmatic findings. We see very little sensitivity to the size of the secondary cache, which suggests that further testing using additional message traces needs to be done before we can reach any conclusions.

Varying the pattern size. Figure 3 shows the sensitivity of our three performance metrics to the pattern size used for content matching. Usually a spammer sends multiple copies of a mail by making few alterations. The size of pattern is an important performance parameter as the content of match for such mails depends upon the pattern size chosen. Once again, we see that Accuracy is always very high, whereas the probability of false positives is always extremely low while the probability of false negatives is small but increasing.

Varying the matching threshold. The performance was calculated by varying the match threshold. It is the lower bound of percentage of match obtained for an incoming mail to be qualified as an email message. If we keep the threshold value very low, then the chances of false positives increases. The plot of various metrics with varying matching threshold is shown in figure 4.

Rate-limiting spam emails at edge router. The arrival rate of spam is rate limited to effectively utilize the network bandwidth. This parameter is an important factor in measuring the performance. We plot the percentage of spam messages received at the edge router and the percentage of the spam messages sent by the router. The plot is shown in the figure 5. From the figure 5, we can see that the percentage of spam messages is much less than the spam messages received by the router.

6.2 Bayesian Spam Classification Phase

A mail is declared as spam if the estimate of spamminess of the email is greater than a threshold value; otherwise it is declared as a good email.

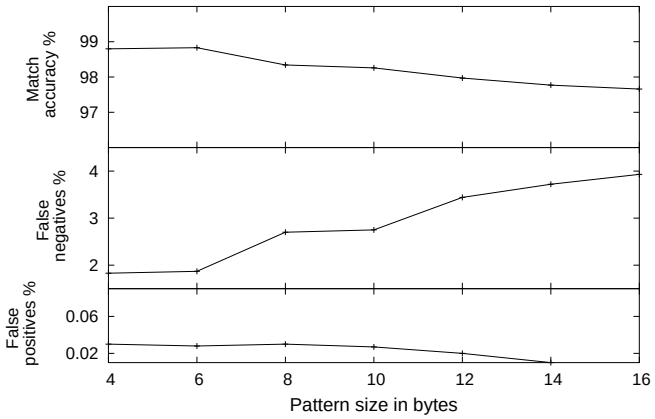

Fig. 3. Tuning the message pattern size

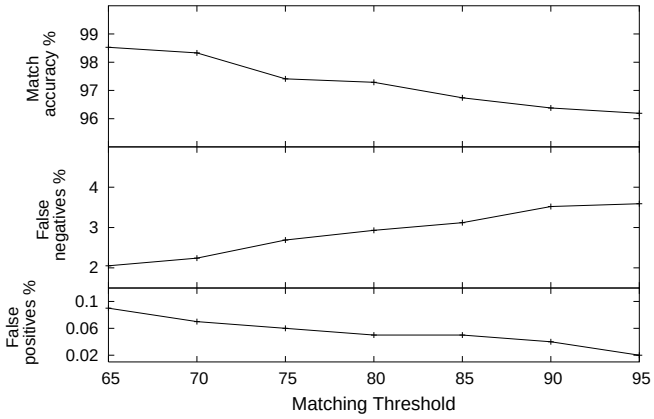

Fig. 4. Tuning the matching threshold

Varying the Number of Spam and Ham emails. The Accuracy was calculated by varying the number of unseen spam and good emails and plotted in figure 7. The accuracy is found to be 97% on an average.

Varying the threshold value. Threshold value is used for classification of an email as a good or spam email based on the Indicator of spamminess. The figure 6 shows the accuracy obtained by considering various threshold values. For this project a threshold value of 0.5 was considered.

Number of false positives and false negatives. We obtained almost negligible false-positives by the Bayesian classifier. The detailed experimental results are provided in table 1.

Table 1. Accuracy of Bayesian Algorithm

Case	No of unseen good emails	No of unseen spam emails	Accuracy Accuracy	No of False Positives	No of False Negatives
1	50	50	97.0%	0	0
2	100	100	97.0%	0	2
3	200	200	98.0%	1	6
4	400	400	97.5%	1	17
5	800	800	95.3%	0	71
6	1000	1000	95.8%	3	121

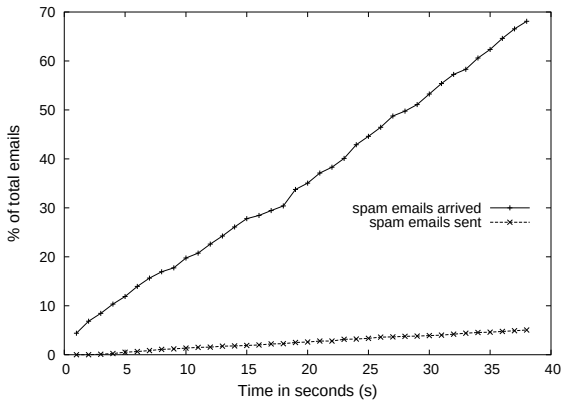

Fig. 5. Rate-limiting spam emails

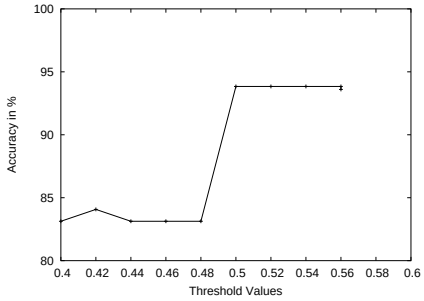

Fig. 6. Accuracy observed on various Threshold values

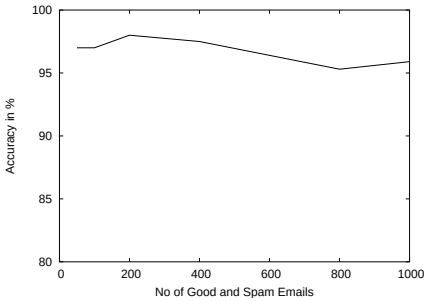

Fig. 7. Bayesian matching accuracy

7 Conclusion and Future Work

This work demonstrates that a significant amount of spam controlling can be successfully achieved at the router level. This approach not only protects the end-users from excessive volumes of unsolicited mails, but also limits the network congestion caused by spams. In this paper, we have implemented a two phase approach to detect spam at the router level. First phase identifies the bulk mail by pattern-matching and the second phase applies Bayesian classifier on the identified bulk mail to classify it as a spam. The experiment conducted on various spams collected from different sources gave an average accuracy level of 97%. The method was most effective with respect to "False Positives". The number of "False Positives" were almost negligible (Table 1). This can be implemented by using any network processors tailored for high-speed networking applications, which provides the programmability, flexibility, and good efficiency. Our approach forces extra costs and burdens for spammers to send spams and controls the further abuse of the Internet traffic.

Nevertheless, there is still plenty of scope for the improvement. For example, we can improve our spam classifi er by supplementing it with a dictionary-based approach to identify spam emails. It can be tricky to handle multi-lingual emails and other emails which might have some spelling mistakes done intentionally by spammer. Although, most of the spam emails contains some of the English dictionary words and others are just the words that are random strings of characters. We can also add some features at the routers to notify the end-users of identifi ed spam emails that was rate-limited at the router.

References

1. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian Approach to Filtering Junk E-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
2. Excedent's White Paper: Spam DNA Filtering Version 2.00 (2003) www.excedent.com/white-papers/Spam-Filtering.pdf.
3. Yerazunis, B.: The Spam Filtering Plateau at 99.9% Accuracy and How to Get Past It. In: MIT Spam Conference. (2004)
4. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In Nédellec, C., Rouveirol, C., eds.: Proceedings of ECML-98, 10th European Conference on Machine Learning, Chemnitz, DE, Springer Verlag, Heidelberg, DE (1998) 137–142
5. Federal Trade Commission Consumer Alert: Who's Spamming Who? Could it be You? (2004) http://www.ftc.gov/bcp/conline/pubs/alerts/whospamalrt.htm.
6. Lamb, M.: TarProxy: a Statistically-Driven SMTP Tarpit. In: MIT Spam Conference. (2004)
7. Resnick, P.: Internet Message Format, RFC 2822 (April 2001)
8. Klensin, J.: Simple Mail Transfer Protocol (SMTP), RFC 2821 (April 2001)
9. Boyer, R., Moore, J.: A fast string searching algorithm. In: Communications of the ACM., 20(10). (1977) 762–772
10. Bayes, T.: An Essay towards solving a Problem in the Doctrine of Chances. In: Philosophical Transactions of the Royal Society of London, 53. (1763)
11. Spamassassin.org: Spam Mails Archive (2002-2003) http://spamassassin.org/publiccorpus/.
12. SpamArchive.org: Spam Mails Archive (2004) http://www.spamarchive.org/.