Controlling Spam Emails at the Routers

Banit Agrawal, Nitin Kumar, Mart Molle
Department of Computer Science & Engineering
University of California, Riverside, California, 92521
Email: {bagrawal,nkumar,mart}@cs.ucr.edu

Abstract— **Like it or not, unsolicited bulk commercial email (aka "spam") has become a regular menu item on the Internet information diet. Every day, millions of people find their email in-boxes clogged with vast quantities of spam. Moreover, the daily replenishment of all those in-boxes with new spam also consumes significant amount of network bandwidth. Dealing with spam is like fighting a battle against a large army; the most effective approach is to employ multiple tactics. However, almost all spam control methods that have been proposed and implemented follow the same basic theme of establishing a "front line" of defense at the end-user level. Thus, in this paper we propose a method for blocking the supply lines. More specifically, we identify spam at the router level and control it via rate limiting.**

Spam identification is done in two phases. In the first phase, we identify the bulk stream of email messages and in second phase we apply Bayesian classifier to identify whether it is a spam. If a bulk email stream is classified as a spam then we rate limit it (e.g no more than one copy per minute). Our proposed method exploits the short timespan delivery and bulkiness of spam emails. We use publicly available spam corpus to evaluate our proposed scheme and in the other set of experiments, we work on one month sanitized log of our department emails to provide the representative results.

I. Introduction

The majority of work on controlling email spams is directed at the recipient level. Using a variety of heuristics, these techniques are quite successful in identifying and/or blocking the delivery of spam to recipient's email in-box [1][2][3][4]. However, even if the spammers do not succeed in reaching the recipient's eyeballs, or occupying disk space on the recipient's mail server, they are still free to consume large amounts of network bandwidth in the process. As shown in the analysis of Brightmail's Probe Network [5], the percentage of total Internet emails identified as spam has increased from 16% in June 2002 to 50% in August 2003 and the trend is likely to continue. The statistics obtained by Excedent's white paper [2] shows that 45% of global email traffic is spam and various companies are spending about 30 billion dollars per year to control spam emails in order to make their internal network spam-free. Therefore, the challenge here is to protect the network resources from abuse by spams, not just the end users.

In this paper, we propose 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 mail delivery traffic from other traffic for further processing. Whenever it detects email delivery traffic, it invokes the first phase of our algorithm and attempts to match the content of the incoming message against a cache of recently-seen candidate messages. If it succeeds in finding a match, we 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.

II. 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 [6] and the Simple Mail Transfer Protocol (SMTP), as specified in RFC 2821 [7].

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. Many vendors provide content-addressable memory (CAM) as the co-processors to 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.

III. 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 cache size*. We also keep a time-stamp of each message stored in the primary message cache 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.

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 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. To find a small pattern in a big string, we employ Boyer Moore's algorithm [8], which is much more efficient in complexity than other brute force algorithms.

IV. 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 tokens[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 spamming techniques 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 [9] to get an overall estimate of spamminess of an email message.

V. Recipient-side Notification

We provide a mechanism to inform the end-user of spam emails so that appropriate actions can be taken at the recipient side. We use one reserved flag of TCP header to mark the spam email. Once a spam message is found at the router, the reserved flag is set. It is the end-user's prerogative to utilize this feature to filter spam emails at the recipient side. Generally while sending TCP/IP packets, all the reserved flags of TCP headers are unset. Spammers might not want to set the TCP header's reserved flag because it defeats the purpose for the spammer. Therefore, the spammer will send email messages with TCP header's reserved flag unset. The spam message passes through various routers in Internet. Our ratelimiter at the edge router will classify the spam message and set the appropriate flag. As this flag is set by the router and spammers don't have any control on the router, an efficient classification can be achieved.

VI. Testing

The operation of the algorithm within a router was represented by simulating the execution of the router's task scheduler loop. We created a stream of IP packets from

[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.

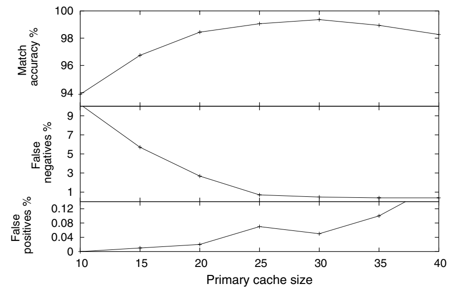

Fig. 1. Tuning the primary message cache size

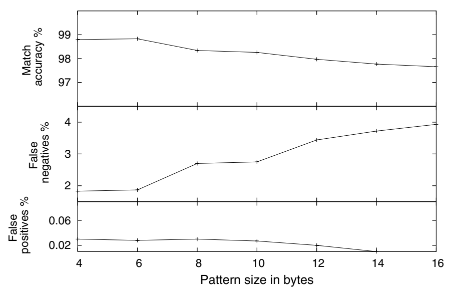

Fig. 2. Tuning the message pattern size

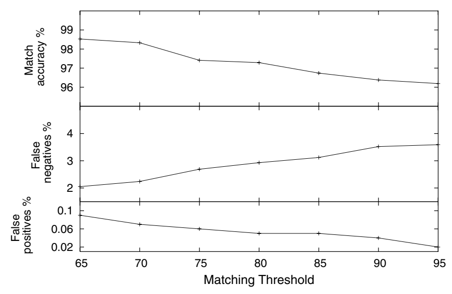

Fig. 3. Tuning the matching threshold

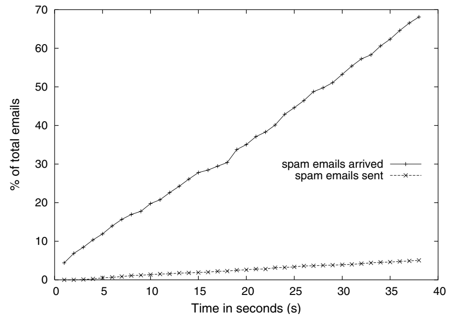

Fig. 4. Rate-limiting spam emails

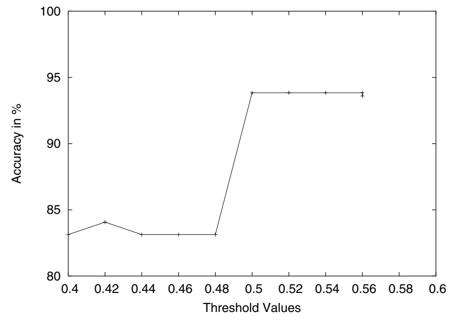

Fig. 5. Accuracy observed on various Threshold values

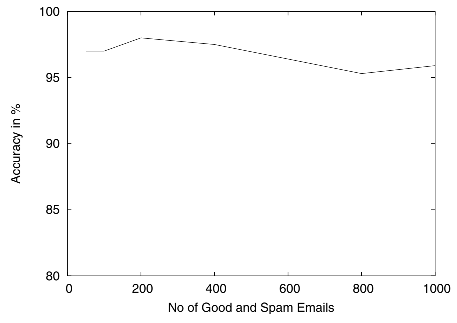

Fig. 6. Bayesian matching accuracy

publicly available archives of spam and good emails. In the router dispatch loop, we fetch an IP packet at each simulation step. We then feed it to the packet classifier. The classifier checks to find if its a SMTP packet and then it sends the packet to SMTP parser. SMTP parser determines whether the received packet is a message packet or not. Upon a match, it calls content matching algorithm to find if this message is a bulk message or not. On a match, the email message is sent to the second phase of Bayesian classification. The training of Bayesian classifier was done using an archive of 1513 good and 2401 spam emails obtained from [10] (October 2002). Testing was done using an entirely different set of 1000 emails(both good and spam emails)taken from the archive [11] and from [10](February 2003). The evaluation of the algorithm was based on: a) *Accuracy*; b) percentage of false positives; and c) percentage of false negatives. *Accuracy* is defined by the following formula:

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

False positive is defined as the percentage of good emails identified as spams, whereas *False negative* is defined as the percentage of spam emails identified as good emails.

A. Content Matching Phase

Our content-matching algorithm includes various control parameters, such as primary cache size, secondary cache size, pattern size, rate-limiter rate, and matching threshold to fine tune the performance of our content-matching algorithm. A typical value of 100 for primary cache size, 20 for secondary cache size, 75% for matching threshold and 60 seconds for ratelimit timegap is used in our analysis. The effects of altering these parameters are shown in Figures 1–4.

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 1. We see that the *Accuracy* is generally 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%). We see very little sensitivity to the size of the secondary cache, which suggests that further tests using additional message traces needs to be done before we can reach any conclusions.

Varying the pattern size. Figure 2 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. 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. Matching threshold is the lower bound of match percentage 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 the three metrics with varying matching threshold is shown in figure 3.

TABLE I

EFFECT OF PRIMARY CACHE SIZE

Primary cache size	Size match Percentage	Size+ from Percentage	Size+ time Percentage	Average time difference seconds (s)
40	61.018	39.601	37.945	21.531
60	67.969	41.358	38.280	21.659
100	77.752	43.932	38.741	21.773
500	97.154	52.140	39.416	21.898
1000	98.995	54.557	39.472	21.904
5000	99.141	54.885	39.477	21.904

TABLE II

EFFECT OF SECONDARY CACHE SIZE

Secondary cache size	Size match Percentage	Size+ from Percentage	Size+ time Percentage	Average time difference seconds (s)
10	75.886	43.190	37.165	19.287
20	77.752	43.932	38.741	21.773
40	80.087	44.286	39.303	24.137
100	84.022	44.846	39.950	24.774

TABLE III

EFFECT OF SIZE-MATCH PERCENTAGE

Match Percentage	Size match Percentage	Size+ from Percentage	Size+ time Percentage	Average time difference seconds (s)
0.5	63.003	39.228	29.538	23.547
1	77.752	43.932	38.741	21.773
2	90.688	48.503	50.232	21.344
3	95.501	50.619	55.889	22.088
4	97.473	52.221	59.681	22.507

TABLE IV

EFFECT OF RATELIMIT TIMEGAP

Ratelimit timegap in seconds (s)	Size match Percentage	Size+ from Percentage	Size+ time Percentage	Average time difference seconds (s)
30	77.519	43.752	30.387	10.081
60	77.752	43.932	38.741	21.773
120	78.203	44.039	47.810	38.562
240	78.812	44.144	55.379	65.489
600	79.747	44.402	65.330	153.28

Rate-limiting spam emails at edge router. 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 figure 4. From the figure, we can see that 60% of the spam messages are ratelimited at the end of simulation.

B. 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.

Varying the Number of Spam and Good emails. The Accuracy was calculated by varying the number of unseen spam and good emails and plotted in figure 6. 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. Figure 5 shows the accuracy obtained by considering various threshold values. A threshold value of 0.5 is taken for this experiment.

C. Analysis of Realtime Email Logs

In this subsection, we provide a detailed analysis of real-time email logs. We use one month log of our department emails for the experimental analysis. The size of the email, *from* field, and timestamp of the email are the only attributes used in our analysis. We provide the justification of short time span feature of bulk emails in temporal domain by selecting different values of rate-limiting timegap. We use different form of bulk classification as described below:

- When the size of the email matches (”size”)
- When the size and the ”from” field of the email matches (”size+from”)
- When the size matches and the new email falls in the ratelimit time imposed by the matched email's timestamp and ratelimit timegap (”size+time”).

We also provide the average time difference of bulk emails, when the bulk email falls in the ratelimit timegap. A typical value of 100 for primary cache size, 20 for secondary cache size, 1% for size match percentage and 60 seconds for ratelimit timegap is used to for our analysis.

1) Varying the size of message caches: We vary the primary cache size from 40 to 5000 to find its effect on identifying bulk emails. From the Table I, it is evident that there is a slight increase in the percentage of bulk emails when the primary cache size is increased from 1000 to 5000. We observe that it can identify 38.741 % ”size+time” bulk emails of the total emails with a reasonable primary cache size of 100. We vary the secondary cache size from 10 to 100 and the results are shown in Table II. With a reasonable secondary cache size of 40, we can identify about 39% bulk emails. It is evident from table II that there is a constant increase in the percentage of identified bulk emails with increase in the size of secondary cache.

2) Match Percentage: We vary the size-match percentage from 0.5% to 4% and the result is tabulated in Table III. If we increase the size match percentage, the chances of getting more matches becomes high. This is reflected in table III as there is constant increase in all three forms of classified bulk emails with increasing size match percentage.

3) Ratelimit timegap: This parameter mostly affects the ”size+time” bulk emails. We vary the rate limit time gap from 30 seconds to 600 seconds. The corresponding results are shown in Table IV. For ratelimit timegap of 60 seconds, we effectively send 64.1 % less bulks by ratelimiting them.

4) Cumulative Size Distribution of Emails: We also analyze the cumulative size distribution of bulk emails (size+from) and temporal bulk emails (size+time). The plot of the cumulative size distribution for ”size+from” bulk emails and temporal bulk emails are shown in figure 7 and figure 8 respectively.

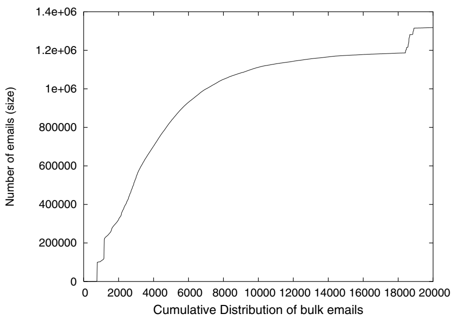

Fig. 7. Cumulative size distribution of bulk emails (size+from)

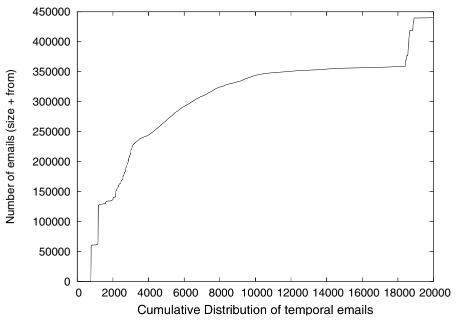

Fig. 8. Cumulative size distribution of temporal bulk emails

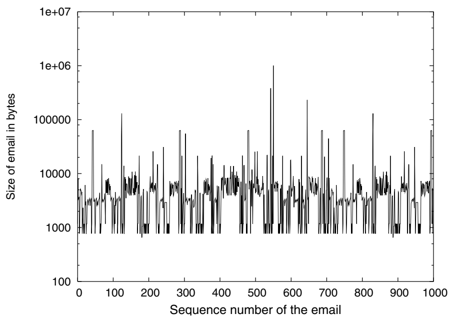

Fig. 9. Arrival distribution of temporal bulk emails

From the figure 7 and 8, it is evident is that most of the bulk emails are in the range of 2000 to 8000 bytes in size.

5) Arrival Distribution of Emails: The arrival distribution of emails give true behavior of incoming emails in temporal domain. We plot the size of the email verses the timestamp of the email. The plot of the arrival distribution for the temporal bulk emails is shown in Figure 9. We can see that there are lots of horizontal portions in the figure, which demonstrates the short-time span feature of the bulk emails.

VII. Related Work

Spam is a growing problem for email users and many solutions have been proposed. These solutions vary from a postage fee for email to Turing tests to simply not accepting email from people you don't know. 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, the domain name, 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. Since the accuracy of these techniques is not 100%, some rare legitimate emails may not get delivered. TarProxy [12] is a method for throttling connections between spammer and an SMTP server by slowing the rate at which the spammer can send spam. Most of the filtering techniques are [1] Naive Bayesian classification, Memory-based approach, Markov chains [3], and Support Vector Machines (SVMs) [4]. Bayesian spam classification technique claims an accuracy level of about 99% [1], but it has a shortcoming of considering the independence of features. Vipul's razor [13] is a good example for collaborative filtering. It is a distributed, spam detection and filtering network, which establishes a constantly updating catalog of spam in propagation. 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-limits the spam emails. It also notifies the end-user of spam emails, identified at the router, by setting a particular TCP header's reserved flag.

VIII. Conclusion and Future Work

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. 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.

Nevertheless, there is still plenty of scope for improvement. Our scheme can be complemented with Vipul's razor [13] scheme to achieve a better collaborative filtering platform. Our proposed scheme can also be used to detect viruses by storing virus signatures separately. Virus detection along with the spam detection at the router level can be a great step towards building a secure Internet backbone.

Acknowledgment

We thank Victor Hill of UC, Riverside for providing us with one month real-time sanitized log of department emails.

References

[1] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian Approach to Filtering Junk E-mail," in *Learning for Text Categorization*. 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] B. Yerazunis, "The Spam Filtering Plateau at 99.9% Accuracy and How to Get Past It." in *MIT Spam Conference*, 2004.

[4] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1998, pp. 137–142.

[5] Brightmail Probe Network, 2004, http://www.brightmail.com/.

[6] P. Resnick, "Internet Message Format, RFC 2822," April 2001.

[7] J. Klensin, "Simple Mail Transfer Protocol (SMTP), RFC 2821," April 2001.

[8] R. Boyer and J. Moore, "A fast string searching algorithm," in *Communications of the ACM., 20(10)*, 1977, pp. 762–772.

[9] T. Bayes, "An Essay towards solving a Problem in the Doctrine of Chances," in *Philosophical Trans. of Royal Society of London, 53*, 1763.

[10] Spamassassin.org, "Spam Mails Archive," 2002-2003, http://spamassassin.org/publiccorpus/.

[11] SpamArchive.org, "Spam Mails Archive," 2004, http://www.spamarchive.org/.

[12] M. Lamb, "TarProxy: a Statistically-Driven SMTP Tarpit," in *MIT Spam Conference*, 2004.

[13] V. V. Prakash, "Vipul's razor," 2004, http://razor.sourceforge.net/.