TOKEN BUCKETS FOR OUTGOING SPAM PREVENTION

Wilfried N. Gansterer, Helmut Hlavacs, Michael Ilger, Peter Lechner, Jürgen Strauß
Institute for Distributed and Multimedia Systems, University of Vienna
Lenaugasse 2/8, A-1080 Vienna, Austria
e-mail: wilfried.gansterer, helmut.hlavacs, michael.ilger, peter.lechner, juergen.strausz@univie.ac.at

ABSTRACT
In this paper we propose a concept for preventing unsolicited bulk e-mail (UBE, "spam") at the *outgoing* SMTP server. In particular, we take the point of view of an Internet Service Provider (ISP), who wants to control the e-mail traffic going out of its network. A central component of our concept contains the adaptation of a token bucket mechanism which allows the ISP to dynamically limit the number of e-mails sent out by each user. If adjusted properly, this strategy harms the business model of spammers *without* affecting regular e-mail users. In combination with a complementary component, our concept makes it possible to completely eliminate the sending of spam from the network of an ISP.

KEY WORDS
outgoing spam, token bucket, ISP

1 Introduction

Many strategies for addressing the problem of unsolicited bulk e-mail (UBE, "spam") have been proposed. Most of them have an "ad-hoc" nature and lack a lasting effect, because spammers easily find ways to circumvent them. Our goal is the investigation of techniques which are based on the foundations and principles of the current e-mail infrastructure in order to develop longer lasting, effective solutions.

The focus of this paper in particular is on potential solutions for detecting and preventing spam in *outgoing e-mail traffic*, i. e., on technology applicable on outgoing mail servers. This is of particular interest for Internet Service Providers (ISPs) for various reasons: (i) Above all, instead of "treating symptoms" *after* "damage" has been done (unexpected overload in bandwidth and storage capacity as well as loss of end-user productivity), approaches for controlling outgoing e-mail traffic fight the spam problem *at its source*. Thus, they need to be an integral part of any comprehensive anti-spam solution. (ii) Moreover, this approach helps the ISP avoiding bad publicity caused by spam originating from its network and/or by becoming blacklisted. (iii) Last, but not least, the workload on the outgoing mailservers of the ISP is reduced and can be controlled and adjusted better.

Related Work. Anti-spam methods can be categorized according to their point of action in the mail transfer process. Consequently, we distinguish three classes of approaches. *Pre-send methods* act *before* the e-mail is transported over the network, whereas *post-send methods* act *after* the e-mail has been transferred to the receiver. A third class comprises *new protocols*, which subsumes approaches based on modifying the transfer process itself. A more detailed categorization is given in [1].

Most of the existing research in the area of anti-spam methods has focused on post-send methods, more specifically, on filtering messages at the receiver side. Important examples are black- and whitelisting, Bayes filters [2], or rule sets (SpamAssassin [3]). From the point of view of the individual user, some post-send methods are able to achieve reasonably satisfactory results *provided* they are trained, tuned and maintained permanently. This effort, required for maintaining satisfactory performance of post-send methods, is one of their main disadvantages. An even more substantial drawback is the fact that these methods act – by definition – after the message has been transferred and received. Consequently, a big part of the damage caused by spam is done *before* post-send methods can become active. They are not able to reduce the overhead, the waste of bandwidth, processing power, memory and time caused by spam. Moreover, privacy issues may arise since most of them have to access the message content for satisfactory results.

Suggestions for new protocols extending or replacing SMTP have been made in order to overcome its deficiencies in the context of the spam problem, for example, AMTP [4] or IM2000 [5]. The main problem is that a common worldwide agreement on changing the protocol used for e-mail transfer and a coordinated transition seem unrealistic in the foreseeable future.

It is essential to develop strategies which fight the problem at the source in order to avoid the overhead and waste of resources caused by spam. Thus, our focus in this paper is on pre-send methods. Existing research in this area can be grouped into two categories: (i) Strategies for identifying sources of spam and, based on the results, for blocking those sources; and (ii) strategies for harming the business model of spammers by increasing the costs associated with sending out (spam) e-mail.

The first category comprises simple methods, such as relying on user complaints and shutting down spamming computers, but also more sophisticated ones. For example, one can try to detect abnormal behavior of sending users or

sources of spam by examining log data produced by outgoing and incoming mail transfer agents (MTAs). Heuristics for the latter are described in [6] and [7]. Besides log data, network traffic on the TCP layer can be used to gather information about spamming machines. A link analysis technique described in [8] identifies nodes that are anomalous in behavior. The underlying assumption is that e-mail servers form a community interacting with each other and that nodes which are not part of the community (for example, spam sources) differ in their behavior. The strength of such network traffic based approaches is certainly that usually the message itself does not have to be accessed. However, some of these approaches assume a lot of detailed information about the network structure. Moreover, a major disadvantage is that they are mostly *reactive* (a host is identified as spam source *after* spam was sent out).

It seems more efficient to investigate approaches which prevent spam messages from being sent out in the first place. The basic idea is to compromise the underlying business model. Conceptually, these techniques can be further divided into two groups: (i) approaches which allow unlimited sending of e-mail, but increase the costs for the sender; and (ii) methods which in some way limit the number of messages a user can send. An increase in costs can, for example, be achieved by micro-payment models [9] or by imposing computational costs for sending e-mail messages. Important representatives of the latter approach are CPU bound [10] and memory bound techniques [11, 12]. A combination of three possible techniques is proposed in [13], comprising HIP (Human Interaction Proof) challenges, computational challenges and paid subscription.

Synopsis. The remainder of this paper is organized as follows. In Section 2 we outline the basic structure of our approach, in Section 3 we review the basics of the token bucket concept, in Section 4 we describe how we apply this concept to spam defense, and in Section 5 we discuss a complementary component of our concept designed for controlling outgoing e-mail traffic which does not go through the outgoing mailserver of the ISP. Section 6 concludes our paper.

2 Basic Structure of our Approach

Existing approaches for preventing outgoing spam have various drawbacks. In particular, they impose restrictions which may also affect regular e-mail users. Ideally, one would like to have a method which (i) limits the sending of e-mails in such a way that the business model of spammers is harmed while (ii) its effects are *unnoticed* by regular e-mail users. Investigations of the business models underlying spamming indicate that this goal can be reached [14]. Our concept, outlined in the following, is designed accordingly. In order to motivate its structure we need to analyze potential spamming scenarios first.

Let us assume that an individual S intends to send spam to a number of recipients. S is connected to the Internet through ISP X. We distinguish two relevant scenarios:

1. S sends e-mail through the outgoing mail server of X.
2. S connects to an open proxy, an open relay or to a third party e-mail provider *outside* X's network directly and uses it to send out e-mail.

In principle, two more situations are conceivable. Theoretically, S could connect to an open proxy, to an open relay or to a third party e-mail provider *inside* X's network and use it to send out e-mail. S could also run his own e-mail server (or a spamming tool) locally. However, ISPs tend to prohibit the installation of an open proxy, open relay or of a private e-mail server inside their network. Consequently, the latter two situations involve questions concerning the enforcement of the internal structure and regulations of the ISP and are thus beyond the scope of this paper.

The strategy we propose comprises two core components in order to handle both of the relevant scenarios.

- A token bucket component limits the flow of e-mail through the outgoing mail server of X. If parameterized properly, this happens unnoticed by the regular e-mail user.
- A component consisting of whitelists (or blacklists) restricts or inhibits the outflow of e-mail through channels other than the outgoing mail server of X (open proxies, open relays or third party e-mail providers).

3 Review of the Token Bucket Concept

The concept of leaky/token buckets was originally developed in the context of network traffic shaping. It turns out that it has several properties which are attractive in the context of spam defense: It is very simple, both to implement and to handle, and due to its simplicity it is very efficient and does not cause any severe performance overhead. One of its main attractions is the fact that a *flexible* and *adaptive* limitation of the outflow of e-mail messages becomes possible which can be tuned so that the spammers' business model is harmed while the regular e-mail user is virtually not affected at all (see Section 4).

This motivated our work on adapting the token bucket concept to the context of anti-spam methods. In order to develop this adaptation, we first review the original concept.

Quality of Service in Networks. When sending a stream of packets from a source to a destination, a certain level of Quality of Service (QoS) may be required from the network, which can be characterized by four parameters: *reliability*, *delay*, *jitter* and *bandwidth*. As a rule of thumb, QoS is worsened if the traffic data is sent in large data bursts rather than as a continuous stream. Overlapping bursts soon fill up the router queues, thus increasing delay and jitter. Due to full queues, packets are dropped, therefore decreasing reliability and goodput.

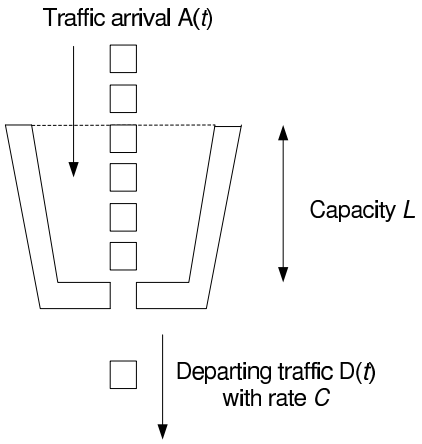

Figure 1. A leaky bucket.

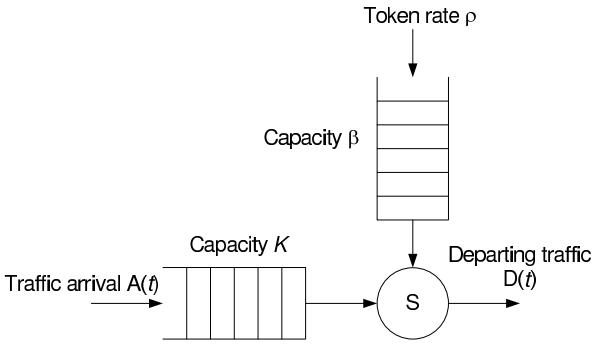

Figure 2. A token bucket.

Several techniques for guaranteeing a certain QoS have been defined in the past. In general, a process sending packets through a network may be forced to describe its QoS needs by using a traffic specification. The sender and the network then agree on a QoS contract, the sender promising not to send more traffic than defined in the traffic specification, while the network promises to offer a minimum QoS with respect to the four parameters mentioned above, at least with high probability (the contract may be violated sometimes).

At the network ingress, the network usually uses a process called *traffic conditioning* to make sure that the incoming traffic does not violate the allowed traffic size. The two main mechanisms for assuring this are called *policer* and *shaper*. A policer identifies packets which violate the agreed traffic specification, mainly because the observed traffic exceeds the agreed traffic volume per time unit. A shaper deals with regular traffic, but may also be used as a policer. It is a mechanism for forcing packets to comply with a given traffic specification.

Usually a shaper consists of a finite queue and server which forwards incoming packets. If the queue is full, incoming packets are discarded. Often the shaper restricts the immediate or long term bitrate of its outgoing traffic. For doing this, the two models *leaky bucket* and *token bucket* have become popular in the past.

Leaky Bucket. A leaky bucket [15] is an abstraction for imposing a strict output bitrate to the forwarded traffic (see Fig. 1). In this model, the shaper is assumed to be a bucket being able to store L bits. Packets entering the bucket gradually fill it up until the maximum is reached. At the bottom, the bucket has a hole where bits drop out at a fixed bitrate. If the next packet to be forwarded consists of b bits, the packet is forwarded as soon as b bits have left the bucket.

A drawback of this method is the fact that senders cannot save bitrate in times they have nothing to send, and spend it when a lot of data is to be transmitted. An advantage is that the shaped traffic does not exhibit any bursts but is completely smooth. Also, it is obvious that a leaky bucket with small L in essence represents a policer.

Token Bucket. A shaping algorithm with inherent bitrate saving capabilities is given by a *token bucket* (see Fig. 2).

Here, two buckets exist, one for storing incoming traffic in a queue with a capacity of K bits, another one for storing a maximum of β so-called *tokens*. Each incoming bit needs to remove a token from the token bucket to be forwarded, and the token bucket is filled with a rate of ρ tokens per time unit [16]. If the token bucket is full, no more tokens can be added to the bucket. If the traffic queue is full, incoming packets are dropped. If a sender does not send data for some time, the token bucket will fill up. Once the token bucket holds $0 \leq n \leq \beta$ tokens, then up to n incoming bits may be forwarded in a single burst (maximum burst size). However, in the long run, the committed sustainable bitrate is limited by ρ.

More formally, the traffic arriving at a shaper may be characterised by its *arrival process* $A(t)$, which defines the number of bits that have arrived at the shaper at time t [17]. Likewise, the number of bits that have left the shaper at time t is called the *departure process* $D(t)$. Clearly, $A(t) \geq D(t)$, and the number of bits stored in the traffic queue (the *backlog*) at time t is $X(t) = A(t) - D(t) \geq 0$. An arrival process is limited by an envelope $E(t)$, if for all t and τ with $0 \leq \tau \leq t$ we have $A(t) - A(\tau) \leq E(t - \tau)$, i. e., the number of bits arriving during any time interval $t - \tau$ is limited by $E(t-\tau)$. Moreover, a shaper has an envelope $E(t)$ if its departure process $D(t)$ is limited by the envelope $E(t)$. Envelopes for leaky and token bucket shapers are shown in Fig 3.

Using network calculus, the following result can be derived: If a shaper is limited by an envelope $E(t)$, then for any arrival process $A(t)$, the shaper departure process is given by

$$D(t) = A(t) \otimes E(t) = \inf(A(\tau) + E(t - \tau)), \quad \tau \in \mathbb{R},$$

where the operation $\otimes$ is called (min,+) convolution.

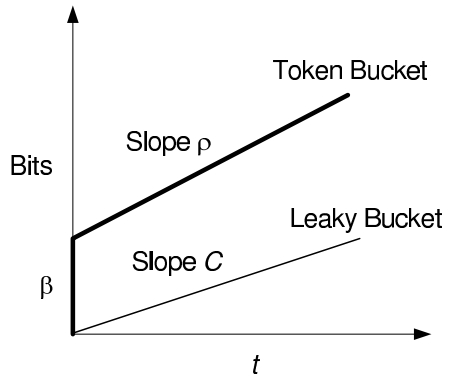

Figure 3. Shaper envelopes for token bucket (thick line) and leaky bucket (thin line).

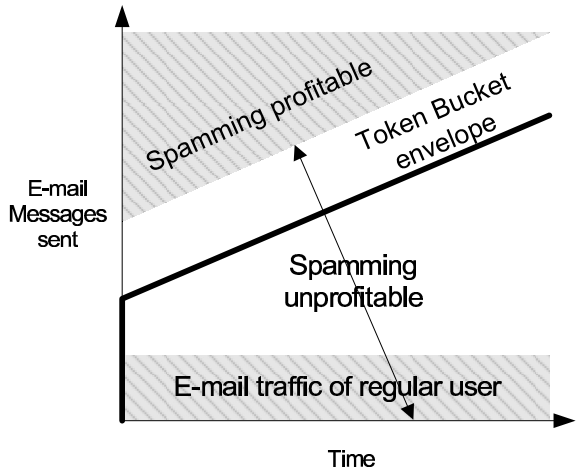

Figure 4. Token bucket envelope spearating the region of regular e-mail from the region of profitable spamming.

4 Token Buckets for Spam Defense

The token bucket concept for generic network traffic can be adapted to e-mail traffic (which can be considered a special form of network traffic) in a straightforward way. Instead of considering the number of packets/bits sent over time, we consider the number of e-mail messages sent over time. Alternatively, an e-mail may also be viewed as a number of data units ("packets") to be transferred. This makes it possible to account for different e-mail sizes.

The profit generated by spamming increases with the number of e-mail messages sent out. In other words, if one wants to render spamming unprofitable, outgoing e-mail traffic has to be shaped in a certain way such that profitable regions are not reached (see Fig. 4). In the following, we describe how to design a token bucket shaper for outgoing e-mail traffic such that regular e-mail users are within the envelope, whereas (profitable) spamming is outside the envelope.

How Many Tokens per Message ? When applying a token bucket strategy to spam prevention, the system can either use one token per message or it can determine the number of tokens needed based on the size of the e-mail (which corresponds to using one token per fixed amount of data to be sent). In this paper we focus on the former variant since the essential information to be transferred in spam tends to be compact (often only a URL), and thus most spam messages are usually rather short.

Design of the Envelope. The commercial success of spamming is related directly to the number of spam messages sent out per time unit. This is explained by the underlying business model: Spammers try to send out big numbers (millions) of small-sized messages. Although the relative response rate tends to be rather low – between 0.00001 % and 0.35 % (see [14] and references therein), this is sufficient in terms of absolute numbers (assuming a fixed income associated with each response) as long as the number of messages sent out is large enough. Due to this dependency on high sending rates, a limitation of the number of outgoing messages is an effective measure. As mentioned before, the concept we use to impose such a limit is a token bucket. In order to illustrate the benefits and to address the central question – how to design the envelope so that the impact on regular users is negligible – we take a closer look at some statistical data.

In case studies based on data from real cases available in the literature [14], one can see that without limitations in the number of messages sent out a single spammer can create large daily revenues (basically only limited by the bandwidth of the internet connection). A limit of a few hundred messages per day is normally sufficient to reduce the spammer's potential revenue below the marginal return. Roughly speaking, a spammer's operation does not start being profitable before he sends out at least in the order of several thousand messages per day.

Restricting Outgoing E-mail Traffic. A simple and straightforward solution is to put a *static limit* on the number of e-mails which can be sent out per time unit, for example, allowing each user to send out at most 100 e-mail messages per day, or on the number of recipients for a single e-mail (cf., for example, the terms of service of Yahoo). Obviously, such rigid limits easily cause notable restrictions for regular e-mail users, because the sending of e-mail is usually not distributed uniformly over a day. Generally speaking, for the success and widespread acceptance of any measure against spam it is crucial to be as transparent and imperceptible as possible for a regular e-mail user.

In summary, the objective has to be the following: Limit the number of e-mail messages sent out in order to compromise the spammers' business model *without* negatively affecting a regular e-mail user (ideally, a regular e-mail user should not notice anything). A token bucket mechanism is a suitable strategy for achieving this goal, because it provides a very flexible way of limiting the number of e-mail messages which can be sent. It can accommodate

for traffic bursts, and, if parametrized and adjusted properly, can achieve the objectives formulated above. In the following, we will discuss proper parametrization and adjustment of a token bucket mechanism for controlling outgoing e-mail traffic.

Parametrization. As summarized in Section 3, a token bucket is specified by two parameters: capacity β and token rate ρ. In order to determine good choices for these two parameters, a rough estimate of the number of e-mail messages a regular user (not a spammer) sends out per day is needed.

Data from ISPs indicates that this number tends to be below 20 messages per day (averaged over all customers, see [7]). A *global* estimate can be found on the basis of estimates for the total daily volume of e-mail messages, the total number of internet users, and the overall percentage of spam. Radicati[1] around 130 Billions e-mail messages are sent out per day. The number of internet users worldwide is roughly 950 Millions,[2] and Postini[3] estimates that more than 80 % of e-mail messages are spam. Based on these numbers, we can estimate that around 26 Billions non-spam ("ham") messages are sent out worldwide per day. Consequently, on average every internet user sends fewer than 30 ham messages per day.

We (conservatively) estimate the average sending rate $\bar{s}$ per user as $\bar{s} \approx 50$ messages per day. Due to legitimate (ham) mass mailers, the sending rate of most users can be expected to be significantly lower. Consequently, with a choice of $\beta = 2 \cdot \bar{s}$ and $\rho = 2 \cdot \bar{s}/86\,400$ (per second) we can limit the outgoing traffic to at most 100 messages per day, which is clearly below the volume needed for commercially successful spamming. At the same time, regular users would usually not notice any restrictions, because they may send out single bursts of up to twice their average message volume.

Implementation. Our token bucket approach is implemented as policy delegation server for the open source mail server Postfix [18]. For each user, a triplet consisting of (i) the mailbox name, (ii) the time t_0 of the last successful submission of the "RCPT TO:" command (SMTP status code "250"), and (iii) the number T of tokens currently available in his bucket is stored. Every time a client transfers a message to the outgoing mail server, a plug-in is triggered after the execution of the "RCPT TO:" command within the SMTP dialogue. If the e-mail has r recipients, the plug-in is executed r times.

Each execution of the plug-in computes the number of available tokens based on the following parameters: capacity β (= maximum number of tokens available per user), token consumption per recipient T_c, and token growth per time unit ρ.

Based on the current system time t (when the sending takes place) and on t_0, the new amount T of tokens currently available in the bucket can be computed using the following equation:

$$T(t) = \min\left(T(t_0) + (t - t_0) \cdot \rho, \beta\right)$$

This number is the basis for the decision whether the e-mail can be sent to a certain recipient or not. If $T(t) - T_c \geq 0$, the recipient is accepted, the number of available tokens is updated in the user's triplet as $T := T(t) - T_c$, and the plug-in returns status code "250 OK" to the mail server. If $T(t) - T_c < 0$, the e-mail cannot be sent, the recipient is refused, and the plug-in returns "554 Not enough tokens available" back to Postfix. Alternatively, one could *delay* sending of this e-mail until enough tokens are available.

5 A Complementary Component

Obviously, the token bucket component described in Section 4 can only control the e-mail traffic passing through the ISP's outgoing e-mail server. As mentioned in Section 2, a spammer can circumvent this mechanism by connecting to an open proxy, an open relay or to a third party e-mail server outside the ISP's network and sending his messages through one of these channels. Thus, we extend our concept by another component which allows one to control these "loopholes". We discuss two possible solutions.

Filter-out vs. Filter-in. The first one is a "filter-out" approach. This means, that the ISP *blocks* connections to certain IP addresses from its network. This involves maintaining (black)lists of known (or suspected) open proxies or open relays. This approach is generally applicable and is relatively easy to implement, because it does not require interaction with individual users. However, being an open proxy/open relay is a dynamic feature and it is difficult and rather costly to have complete blacklists available.

The second one is a "filter-in" approach. This means, that the ISP allows outgoing connections *only* to certain registered IP addresses on a "whitelist" (for each customer individually). While this approach has a higher initial overhead in terms of implementation and user interaction, it offers more completeness, because certain inspections can be integrated into the registration process.

Practical Realization. In practice, the ISP has to check connection attempts going out of its network for IP addresses on the blacklist (for filter-out) or on the whitelist (for filter-in) and decide accordingly whether to allow the connection or not.

The filter-out approach requires an efficient procedure for maintaining a reliable, up-to-date and comprehensive list of open proxies and open relays. Although several such lists are publicly available (for example, [19]), it is virtually impossible to have a complete list of all currently open proxies and open relays which could potentially be used for spamming. In order to complement and extend publicly available lists, a feedback mechanism from a spamfilter for incoming e-mail traffic could be used. This is subject of ongoing work.

[1] `www.radicati.com`
[2] `www.internetworldstats.com`
[3] `www.postini.com`

Overall, however, the filter-in approach based on a whitelist is more appealing in terms of performance. Each user's whitelist can be established as part of the registration procedure. The user would be asked to register the IP addresses he intends to connect to. During this registration process as well as in regular intervals afterwards the ISP can check whether these whitelisted IP addresses continue to be "trusted" hosts.

6 Conclusion

We have presented a concept for dynamically limiting the number of outgoing e-mail messages, to be implemented at an outgoing e-mail server. It is based on the adaptation of a token bucket mechanism. This concept has various advantages over existing approaches. Most importantly, it provides a very flexible and adaptive way of limiting outgoing e-mail traffic. In contrast to static approaches, traffic bursts can be taken into account. We have discussed how to parameterize and tune our concept such that the restrictions it imposes are virtually unnoticed by regular e-mail users while at the same time the business model of spammers is compromised. Thus, in contrast to existing approaches, our strategy does not affect all e-mail users, but selectively targets spammers.

Moreover, we have discussed a whitelisting component extending our concept in order to be able to also control outgoing e-mail traffic which does *not* go through the e-mail server of the ISP. In total, our combination of a token bucket mechanism and the whitelisting component makes it possible for an ISP to prevent the sending of spam out of its network. If widely used by ISPs, this strategy has the potential of significantly reducing the amount of spam on the Internet.

Acknowledgments. We would like to express our gratitude to Internet Privatstiftung Austria, mobilkom austria, UPC Telekabel, and Internet Service Providers Austria for supporting this research.

References

[1] W. Gansterer et al., Anti-spam methods – state-of-the-art, Tech. rep., Institute for Distributed and Multimedia Systems, University of Vienna, 2004.

[2] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, & C. D. Spyropoulos, An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages, *SIGIR '00: Proc. 23rd annual international ACM SIGIR conference on research and development in information retrieval*, Athens, Greece, 2000, 160–167.

[3] The Apache SpamAssassin Project, `http://spamassassin.apache.org/`.

[4] W. Weinman, Authenticated mail transfer protocol, 2003, `http://amtp.bw.org/`.

[5] D. J. Bernstein, Internet mail 2000, `http://www.im2000.org`.

[6] R. Clayton, Stopping spam by extrusion detection, *1st Conf. on Email and Anti-Spam (CEAS)*, Mountain View, USA, 2004.

[7] R. Clayton, Stopping outgoing spam by examining incoming server logs, *2nd Conf. on Email and Anti-Spam (CEAS)*, Stanford, USA, 2005.

[8] P. Desikan, & J. Srivastava, Analyzing network traffic to detect e-mail spamming machines, *Proc. Workshop on Privacy and Security Aspects of Data Mining*, Brighton, UK, 2004, 67–76.

[9] D. Turner, & D. Havey, Controlling spam through lightweight currency, *Proc. Hawaii Intern. Conf. on Computer Sciences*, Honolulu, USA, 2004.

[10] A. Back, Hashcash – a denial of service countermeasure, Tech. rep., 2002, `http://citeseer.ist.psu.edu/back02hashcash.html`.

[11] M. Abadi, M. Burrows, M. Manasse, & T. Wobber, Moderately hard, memory-bound functions, *ACM Trans. Inter. Tech.*, *5*(2), 2005, 299–327.

[12] C. Dwork, & M. Naor, Pricing via processing or combatting junk mail, *Proc. 12th Annual International Cryptology Conference on Advances in Cryptology*, Lecture Notes in Computer Science, vol. 740, Springer-Verlag, 1992, 139–147.

[13] J. Goodman, & R. Rounthwaite, Stopping outgoing spam, *Proc. 5th ACM Conf. on Electronic Commerce*, New York, USA, ACM Press, 2004, 30–39.

[14] J. Strauss, Analyse technischer Lösungen zur Spamvermeidung und ihre betriebswirtschaftlichen Implikationen, Master's thesis in Business Informatics, Institute for Distributed and Multimedia Systems, University of Vienna, 2005.

[15] J. Turner, Leaky bucket, *IEEE Communications Magazine*, *24*, 1986, 8–15.

[16] W. Stallings, *High-speed networks and internets, 2nd ed.* (Englewood Cliffs, NJ: Prentice Hall, 2002).

[17] A. Kumar, D. Manjunath, & J. Kuri, *Communication networking* (San Francisco, CA: Morgan Kaufmann, 2004).

[18] W. Venema, Postfix mail transfer agent, 2005, `http://www.postfix.org`.

[19] List of real-time spam black lists (rbl), `http://www.email-policy.com/Spam-black-lists.htm`.