

This report summarizes the results of phases 2 and 3 of the project FA 384018 "Spamabwehr" of the Institute of Distributed and Multimedia Systems at the University of Vienna, funded by Mobilkom Austria, UPC Telekabel and Internet Service Providers Austria (ISPA).

Copyright: ©2005 by University of Vienna. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means without the prior permission of the authors. The Institute of Distributed and Multimedia Systems at the University of Vienna does not guarantee the accuracy, adequacy or completeness of any information and is not responsible for any errors or omissions or the result obtained from the use of such information.

About the Authors: Project "*Spamabwehr*" was launched in summer 2004 at the Department of Computer Science (Distributed Systems group) which, due to internal restructuring at the University of Vienna, recently became the new Institute of Distributed and Multimedia Systems at the Faculty of Computer Science.

Contact for Project "Spamabwehr":

phone: +43-1-4277-39652

e-mail: wilfried.gansterer@univie.ac.at, spamabwehr@ani.univie.ac.at

Institute of Distributed and Multimedia Systems
University of Vienna
Lenaugasse 2/8, A-1080 Vienna (Austria)

Contents

Executive Summary	6
1 Introduction	7
1.1 Background	7
1.2 SMTP-Based Spam Defense	8
1.3 Cost-Based Spam Prevention	8
1.4 Synopsis	8
2 SMTP-Based Spam Defense	10
2.1 Ideas Pursued	10
2.1.1 Objective	10
2.1.2 Related Work	11
2.2 Potential and Limitations	11
2.2.1 Trivial Tests	11
2.2.2 Non Fault Tolerant Mail Servers	12
2.2.3 Open Relay and Open Proxy Tests	13
2.3 EPF – An Extended SMTP Server	14
2.3.1 Architectural Overview	15
2.3.2 Implementation	16
3 Cost-Based Spam Prevention	19
3.1 Cost-based Approaches	19
3.1.1 Scientific Work	20
3.1.2 Economic Background	20
3.1.3 General Information and Facts	22
3.1.4 Cost Factors for Spammers	24
3.2 The Economics of a Spammer	25
3.2.1 Spammer Rents a Server	25
3.2.2 Spammer as Sales Agent	26
3.2.3 Spammer as Online Marketer	26
3.3 Increasing Sender Costs	28
3.3.1 Money Based Costs	28
3.3.2 Time Costs (Delay)	28

4 Experiments	34
4.1 Test Data	34
4.1.1 Live Streams	34
4.1.2 Offline Evaluations	35
4.2 Experimental Results	35
4.2.1 Evaluation of SpamAssassin’s Header-Based Tests	35
4.2.2 Live Stream Evaluation of EPF	37
4.2.3 Offline Evaluation of Open Proxy and Open Relay Tests	38
4.3 Performance Issues	39
5 Conclusions	40
5.1 Standard SMTP Transfer Process	40
5.2 Spammers’ Business Model	41
5.3 Future Directions	41
Appendix: Prototype Installation Notes	43

Executive Summary

This report summarizes the findings and results of phases 2 and 3 of the project "FA 384018 Spam-Abwehr" ("Spam-Defense") which was launched in July 2004 at the Institute of Distributed and Multimedia Systems at the University of Vienna. The project is supported by Mobilkom Austria, UPC Telekabel and Internet Service Providers Austria (ISPA).

This document is structured as follows.

Chapter 1 introduces the background and the main topics discussed in this report.

Chapter 2 describes our findings regarding to possible anti-spam strategies based on the current SMTP. We illustrate that standard SMTP does not provide a suitable infrastructure for spam defense methods. There are only very few basic properties of the SMTP-based e-mail transfer process which can provide reasonably reliable indications about whether an e-mail should be considered spam or not. We have implemented a prototype of an extended SMTP server, *extended Postfix (EPF)*, which includes checks for these basic properties into the standard SMTP transfer process. Experimental observations and a preliminary evaluation are described in Chapter 4.

In the vast majority of cases the sending of spam is motivated by economic incentives (profit from advertizing). Chapter 3 analyzes spammers' business models in order to investigate new methods which try to fight the spam problem at its source (pre-send methods). Our findings indicate that (fortunately) there is a relatively wide gap between the number of e-mail messages a spammer has to send out in order to be profitable and the number of e-mail messages sent out by a "regular" user. This gap provides the opportunity for developing pre-send antispam methods which harm the spammers' business model without affecting the regular user at all. We point out that a *comprehensive* approach (comprising improved post-send methods as well as those new pre-send methods) is required to achieve satisfactory performance of spam defense systems.

Chapter 4 summarizes the experiments we performed in phases 2 and 3 of project "Spamabwehr". These comprise an evaluation of SpamAssassin's header-based tests and a preliminary evaluation of the components of our EPF prototype (partly based on offline testing data). This includes a detailed in-depth analysis of greylisting regarding both spam and ham messages as well as the performance of open proxy and open relay tests.

Finally, we summarize our conclusions and future plans in Chapter 5 and give a brief overview of the installation of our EPF prototype in the appendix.

Chapter 1

Introduction

This report summarizes the results of the second and third phase of the research project FA384018 “Spamabwehr”. The investigations summarized here are based on the results of the first phase of this project, documented in [1].

1.1 Background

At the beginning of the project “Spamabwehr” several objectives were defined. The main target audience are internet service providers (ISPs). In their special point of view the focus is on aspects such as server-side solutions, early detection and prevention, and the reduction of resource demand caused by spam. In this context, “resource demand” means all overhead caused by spam – including the time needed for individuals to distinguish important e-mail from spam as well as the storage requirements for spam messages.

In our first report [1] we provided an overview and an assessment of currently available antispam methods. We surveyed the state-of-the-art in antispam methods and provided a comprehensive categorization of existing approaches.

After some in-depth research it became obvious that most of the methods currently available are not able to fully cope with the problem and do not achieve the desired results. Most of the solutions currently available belong to the category of “post-send” filters, i. e., they screen and classify messages *after* they have been sent off at the sender’s side; usually, also *after* they have arrived at their target host. Obviously, the post-send approach has serious disadvantages in terms of resource demand. Classification and filtering of spam is happening *after* the e-mail has been fully accepted at the mail server. Thus, the overhead caused by spam is hardly reduced, it is only shifted from the end-user to the ISP.

On the basis of the state-of-the-art survey [1] our focus in phases 2 and 3 of project “Spamabwehr” was on the possibilities for SMTP-based (Simple Mail Transfer Protocol) spam defense as well as on cost-based methods for spam *prevention*. In this report, we summarize our main results in these two central research directions, pursued since January 2005. Our agenda in this phase of the project comprised to focus areas – SMTP-based spam defense and cost-based spam prevention.

1.2 SMTP-Based Spam Defense

Our investigations in this area tried to answer the following question:

- Based on standard SMTP, can we design antispam methods which act in the transfer process of an e-mail message, at the receiving SMTP server, and thus allow us to reduce the waste of resources caused by spam?

We decided to emphasize this topic in the second and third phase of “Spamabwehr I” for several reasons. Firstly, from a practical point of view, it is currently unrealistic to design any spam defense concepts which are not based on standard SMTP. Thus, in order to have a solid foundation for any new developments, it is essential to carefully investigate the potential and limitations of spam defense methods in the e-mail transfer process. Secondly, early antispam actions at the receiving server seemed to have the biggest potential for reducing the associated overhead (reject before final acceptance, etc.).

In this report, we also describe a first prototype implementation of an extended SMTP server which integrates some basic spam defense features into the standard SMTP e-mail transfer process.

1.3 Cost-Based Spam Prevention

In order to successfully address the spam problem, it is indispensable to analyze the underlying motivation and incentives. This forms the foundation for developing methods which tackle the problem at its roots. Thus, the central question to be addressed is the following:

- What methods can be developed to *prevent* spam by attacking the problem at its source, i. e., by harming the spammers’ business model which is the main motivation for the spam phenomenon?

Based on a careful analysis of the business model used by spammers it is possible to design methods which harm this business model by increasing the costs for spammers (*without* affecting “regular” customers!). The objective is to create an environment where sending spam messages is too expensive, becomes unprofitable, and therefore is not of interest any more. Several solutions have been proposed to achieve this objective. In this report, we analyze and evaluate the most promising ones. The results of these activities will be the foundation for further work to be carried out in the continuation project “Spamabwehr II”.

1.4 Synopsis

In Chapter 2 we discuss the potential and limitations of spam defense methods which utilize features of the standard SMTP e-mail transfer process. Moreover, we describe the concept of an extended SMTP server *EPF* (Extended Postfix) which integrates a few basic structural spam detection techniques. These techniques do not depend on fast changing properties of spam, but are motivated by basic characteristics of the underlying infrastructure for e-mail transfer.

In Chapter 3 we give an analysis of the business model underlying spamming activities. Moreover, we provide a detailed evaluation of existing cost-based antispam methods on the basis of the results of this analysis.

In Chapter 4 we report on our experimental evaluation of SMTP-based and related antispam methods. This includes an evaluation of the header tests included in SpamAssassin as well as an evaluation of the prototype of our extended SMTP server EPF.

This report not only summarizes our answers to the questions stated above. The insight gained is also very useful in the process of identifying features to be included in a new adaptive and self-learning profile based approach for spam defense, which will be pursued further in the continuation project “Spamabwehr II”. Accordingly, Chapter 5 summarizes our results and also outlines the most important topics of our ongoing and future work.

Chapter 2

SMTP-Based Spam Defense

The Simple Mail Transfer Protocol was originally developed in 1982 [2]. At this time the spam problem was nonexistent and therefore measures for defeating unsolicited bulk e-mail (UBE) were not taken into consideration when the protocol was designed. Attention was given to reliability and robustness. Authentication, integrity checks and measures for limiting transfer of e-mail messages were not addressed. SMTP does not provide any means for distinguishing “regular” e-mail from spam or for blocking spam.

The SMTP had already been around for a while and was established worldwide when spam messages started to become a problem. Thus, there is a big inertia which makes it almost impossible to change or replace the protocol for the purpose of spam defense. Therefore different approaches like filters were implemented looking for certain keywords in mail messages. As those static lists stopped working effectively the lists of keywords were extended, Bayesian filters were introduced resulting in an increased need for computing power to perform those tests. This has led to another point where a lot of time and money is used to remove spam messages from the users’ mailboxes.

2.1 Ideas Pursued

From a legal point of view, messages have to be delivered as soon as they are accepted by a SMTP server. Thus, any spam defense measures which have the objective of saving resources have to intervene *before* the SMTP dialog is completed.

Therefore a different solution has to be found, preferably allowing mail servers to selectively accept ham messages and refuse spam messages before delivery.

2.1.1 Objective

To describe it as simple as possible, an e-mail server has just one goal: Delivering messages. To be precise it has to deliver messages as fast as possible, without dropping any of them. This implies that the resource demand for every single message should be reduced as much as possible. Spam messages, which constitute a dominating part of today’s e-mail traffic, cause an enormous waste of resources. Thus, it would be desirable to recognize them *as early as possible*. An ideal

solution would immediately distinguish spam messages from ham messages (before accepting the data in the SMTP dialog), refuse accepting spam, and accept only ham. It was one of the objectives of our activities described in this chapter to get as close to this ideal solution as possible.

2.1.2 Related Work

In response to the growing amount of unsolicited bulk e-mail a few countermeasures that are based on data used within the SMTP dialog have been developed.

The most popular and easiest method to block spam is the use of DNS (Domain Name System) based *real time blacklists* (RBL) but their efficiency is hard to estimate. Experimental results in terms of detection rate range from 10% [3] to 80% [4]. Another frequently used method is *greylisting* [5] that rejects spam from non fault tolerant e-mail servers. Performance analysis shows that greylisting delays 98% of spam but also 40% to 51% of ham messages [6].

Recently developed domain based authentication methods such as SPF [7] (Sender Policy Framework), Caller-ID [8] and DomainKeys [9] try to provide functionality for allowing the verification of the integrity of domain information submitted within the SMTP dialog or recorded in the e-mail header.

A completely different approach to defeat spam in the transfer process is the so called *Tar Pit* (*Teergrube*, [10]). “Tarpitting” (“teergrubing”) is not a method for classifying e-mail messages. Instead, it tries to bind the resources and to slow down the spammer’s mail server by including artificial delays in the SMTP dialog.

Unfortunately all these countermeasures, similar to content based methods, have weaknesses allowing to bypass them which is detailed in the next section.

2.2 Potential and Limitations

A central problem of all SMTP-based approaches is that almost all information which is exchanged between client and server can be forged; the only exception is the IP (Internet Protocol) address of the client.

A reliable method for classifying e-mail based only on information exchanged during the SMTP dialog (*not* including the data of the message body) does not exist [11], but there are some countermeasures that can reduce the amount of spam. The main problem of SMTP-based as well as general spam defense is a possible increase in false positives. This can lead to a total loss of ham e-mail from specific senders due to permanent refusal as most SMTP-based techniques lead to a blocking of the sending mail servers and not of individual messages. In the following chapter we summarize which information can be used to classify spam before accepting the message data.

2.2.1 Trivial Tests

Within the current SMTP dialog the results of only three (rather trivial) tests have some relevance for classifying incoming e-mail:

- Does a HELO/EHLO parameter have a value?

If the answer is yes then no conclusion is possible if the message is spam or not. If the answer is no the message does not necessarily have to be spam but the SMTP client does not act compliant to the defined standards. RFC2821 [12] (Request for Comment) requires that the argument field contains a client identification. If available, the fully-qualified domain name of the SMTP client is used. In most cases it is not possible to verify the argument by performing reverse and forward DNS lookups. A syntactical analysis of the specified parameter, if it is actually according to a fully-qualified domain name, does not allow a classification, too. Therefore, the only information that can be used is whether any value is given or not. A detailed discussion is available in [11].

- Does the Sender Domain of the “MAIL FROM:” address resolve in A- or MX-Records?

The “MAIL FROM:” address specifies a reverse-path to the SMTP client and is used for the delivery of error messages. Therefore obviously invalid addresses can be rejected. If the domain resolves in A- (Address) or MX-Records (Mail Exchanger) the domain actually exists. This does not mean that the domain complies with the sender’s domain and that the specified mailbox within the domain actually exists. If the domain can not be resolved messages can be rejected. The reverse-path of error messages may be null in order to avoid mail loops. It has to be pointed out that a negative verification through DNS may also be caused due to temporarily unavailability of the DNS infrastructure.

- Do the specified recipient mailboxes exist?

If the specified recipient mailbox exists messages can be accepted. E-Mail messages to invalid recipient mailboxes should be rejected within the SMTP dialog, because those delivery attempts are either automatically performed (spam) or misspelled addresses (the message has to be sent again by the (human) sender).

2.2.2 Non Fault Tolerant Mail Servers

A large amount of unsolicited bulk e-mail is sent out with spam tools which often do not act compliant to common standards. They are primarily designed to send out bulks of messages within a preferably short period of time using an often restricted and incomplete (and therefore cheap) SMTP server implementation. The main principle is “fire and forget” using a “quick and dirty” implementation of an SMTP server. RFC compliance and fault tolerance are often not implemented. This fact can be used to reject spam e-mail.

Greylisting

Greylisting provides an aggressive method to incorporate this feature by refusing messages from non fault tolerant SMTP servers. Because spammers often do not know if their recipient addresses actually exist, they do not try to resend messages if an error occurs during the transmission process. Unfortunately, this approach has some important drawbacks summarized in the following.

Greylisting involves a temporary refusal of e-mail messages, which affects both ham and spam. Some SMTP servers do not try to redeliver their messages after receipt of a reply code implying temporary unavailability (category 4xy, transient negative completion reply codes) [12]. Therefore it is necessary to use a proprietary whitelist. Another problem appears with server farms because in the case of load balancing it is not guaranteed that succeeding delivery attempts are made from the same IP address. In this case greylisting does not recognize RFC compliant behaviour and messages are refused more than once. See [13] for a more sophisticated implementation that avoids that problem. Greylisting is also completely useless if Spammers resend their messages. These facts make greylisting inefficient and probably not a good solution for the long run, respectively.

Within the concept of greylisting different approaches can be considered. As the temporary refusal may happen more or less at any time, messages may either be refused as soon as the connection is established in the first place, or after the data is transmitted. In the first case only a minimum of data is sent over the network, while it is impossible to verify the content of the message. In case of the second approach the whole data can be sent through a quick preliminary check to figure out if greylisting is required or not. This may increase the speed of message delivery if a message is clean, but also increases the network traffic if a ham message is wrongly considered to be spam and has to be resent.

2.2.3 Open Relay and Open Proxy Tests

An *open relay* is a host that accepts mail from any source and delivers it to any target. An *open proxy* similarly accepts requests from any source but does not implement the SMTP. Instead an open proxy establishes a connection between a specific source and target and just forwards packet traffic.

Open relays and/or open proxies can be exploited by spammers to send out e-mail messages via uncontrolled intermediate hosts (circumventing any restrictions potentially set up by “regular” e-mail providers). This leads to a shift of the consumption of resources, such as bandwidth and computation time away from the spammers’ to other parties’ infrastructure. Assuming that the existence of open relays/open proxies is either unintended (the result of inadequate system administration) or due to devious intentions and that in general only spammers have a major interest in using such infrastructure, we decided to include open relay/open proxy tests in our prototype.

The test works as follows:

1. During the SMTP dialog the receiving MTA (Mail Transfer Agent) tries to relay a test message through the connected SMTP client.
2. If the SMTP client accepts the test message for delivery, the MTA temporarily rejects the e-mail message. Some MTAs are accepting messages for foreign domains but discard them later. An open relay is not verified until the test message has been delivered to the target. If the MTA accepts but does not deliver the message to the final target the SMTP client has to be informed with an error message [14]. Due to the undefined period of time the delivery process takes blocking the SMTP client during the SMTP session is not possible.

3. If the SMTP client rejects the test message within the SMTP dialog, the host is verified as Closed Relay and added to a whitelist.
4. If the test message is relayed and delivered successfully through the SMTP client, the host is verified as an open relay and is added to a blacklist.

This scheme has several disadvantages. First, the test message leads to higher network traffic. Second, ham messages are possibly delayed. Another drawback is that the test message itself could be identified as spam leading to the SMTP server itself getting blacklisted. The message could be seen as an attempt to exploit the tested SMTP server. Test messages are also sent out unsolicited and at least at the beginning of operation in bulk because any new connected IP address causes a test process. A cache can reduce this negative effect. Nevertheless an operation of open relay tests within an SMTP server for the organization's usual e-mail traffic has to be handled with care. A deployment of a separate test server should be considered.

Besides identifying open relays it is possible to evaluate if a connecting client shows the behaviour of an open proxy. An open proxy is a service that like an open relay accepts requests from any source. A request hereby refers to connection establishment to a specific server. Generally spammers use SOCKS [15] and HTTP/HTTPS [16] (Hyper Text Transfer Protocol) Proxies to deliver their messages [17]. An open proxy test does not require sending an e-mail message. The establishment of a TCP (Transmission Control Protocol) connection from a local IP address through the host to the own SMTP server is enough for verification. If the connection establishment fails it cannot be guaranteed that the host is not an open proxy. The interruption for example may also be caused by time-out failures. It is also generally not known on which port the open proxy is available so checks are limited to standard communication endpoints. Another problem is that the performance of SMTP servers is reduced when tests are performed. Open proxy tests are only suitable for SMTP servers with low traffic even if they consume far less resources than open relay tests.

An alternative to open proxy and open relay tests is the deployment of real time blacklists. In contrast to the on-the-fly tests that were discussed in this chapter an online list is consulted whether to accept or reject a message. Before a query takes place it is recommended to verify that the blacklist's listing policy corresponds with the server's own anti-spam policy. To avoid false positives a rather restrictive provider should be chosen.

2.3 EPF – An Extended SMTP Server

Based on these concepts we designed an extended SMTP server that incorporates open proxy tests, open relay tests and greylisting implemented in a layered architecture. Our prototype, an extension to the Postfix mail server [18], is called *extended Postfix* (*EPF*), and thus fully conforms with standard SMTP. Postfix is a fast, secure and easily extensible open source mail transfer agent for Unix operating systems. Its modular architecture allows rapid development of extensions or plugins implemented in different programming languages. Postfix refers to plugins as *policy servers*. Each incoming connection can be delegated to a *policy server* and then be accepted or rejected according to the plugins return value.

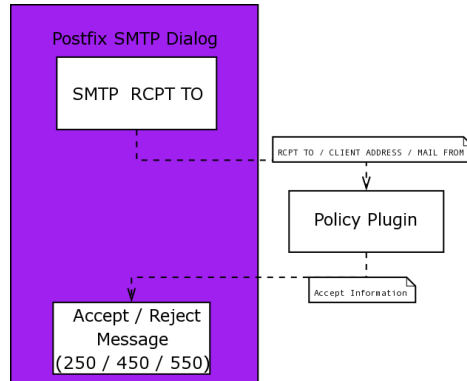


Figure 2.1: Positioning of plugin within the SMTP dialog

We developed a plugin to incorporate open proxy checking, open relay testing and greylisting. It is based on the existing *Grinch* and *Proxycheck* implementations [19, 20]. The main motivation for this prototype implementation is to do all active tests that are possible using all information which is available before the actual data of an e-mail is transferred. In the current prototype, no lookups of existing RBLs for open relays or open proxies are performed, although such extensions are easily possible.

Our extension is plugged into the SMTP dialog after the “RCPT TO:” command. Fig. 2.1 shows how the plugin is called from within Postfix. This architecture makes it possible to return any desired SMTP command to Postfix, which itself returns this command to the connecting MTA.

2.3.1 Architectural Overview

An example of the data submitted to our Perl plugin (i.e. all data submitted to the local Postfix server in the SMTP dialog including “RCPT TO:”) is shown in Listing 2.1. The *sasl* and *ccert* attributes specify information about authentication and are not relevant for our purposes. The *request* and *protocol* attributes show information about the kind of policy (for in- or outgoing mail) and the protocol type and state, respectively.

Our plugin implementation only uses a triple consisting of *sender*, *recipient*, and *client_address*, e.g.

foo@bar.tld/bar@foo.tld/1.2.3.4

where *foo@bar.tld* is the “MAIL FROM:” value, *bar@foo.tld* the “RCPT TO:” entry, and *1.2.3.4* the IP address of the connecting mail server (or potential spam tool). The other entries shown in Listing 2.1 are currently not utilized by any of our checks.

Listing 2.1: Example data submitted before the SMTP “RCPT TO:” command

```
request=smtpd_access_policy
protocol_state=RCPT
protocol_name=SMTP
```

```

helo_name=some.domain.tld
queue_id=8045F2AB23
sender=foo@bar.tld
recipient=bar@foo.tld
client_address=1.2.3.4
client_name=another.domain.tld
instance=123.456.7
sasl_method=plain
sasl_username=you
sasl_sender=
ccert_subject=solaris9.porcupine.org
ccert_issuer=Wietse Venema
ccert_fingerprint=
                C2:9D:F4:87:71:73:73:D9:18:E7:C2:F3:C1:DA:6E:04
size=12345
[empty line]

```

Fig. 2.2 depicts the main sequence of tests performed. The sender of every incoming triple is looked up in a whitelist, and if the address is found the message is accepted. If the accepted message is identified as a previously sent open relay test message the source is confirmed as open relay and added to a blacklist.

After that, EPF checks if the message originates from an open proxy. Therefore the address of the connecting mail server is looked up in the internal open proxy white- and blacklists. If it can be found there, the message is either accepted or rejected. If the sending mail server is not found in any of those lists, the open proxy test is performed. According to that test a message is either accepted and added to the whitelist or rejected and added to the blacklist, respectively.

The open relay test module works analogously. The main difference is that an open relay may takes more time. Thus, we combine the open relay testing with greylisting: If the sending host accepts the relay test message, then his original message sent to EPF is temporarily rejected. If the relay test message arrives back at EPF, then the sending host is proven to be an open relay and it is put into the open relay blacklist.

If a message can not be rejected because of the results of one of those tests, the triple is passed to the greylisting module. Every incoming triple is looked up in the greylisting database. If it is not already there, it is added and the message is temporarily rejected (SMTP code 450). If it is found in the greylisting database, a message is accepted if it was resent after a certain minimal retry time (60 seconds per default), but temporarily rejected if it was resent within this period (in order to handle multiply sent out e-mails).

2.3.2 Implementation

We used Perl [21] to implement our plugin, mainly because of Perl's rapid development capabilities. EPF runs on all architectures that Postfix and Perl run on, i.e. all Unix operating systems. It is configured as a Postfix *policy server* in the mail servers's config files. Except Postfix it depends on some Perl packages like *Cache-Cache* and *Net::SMTP*.

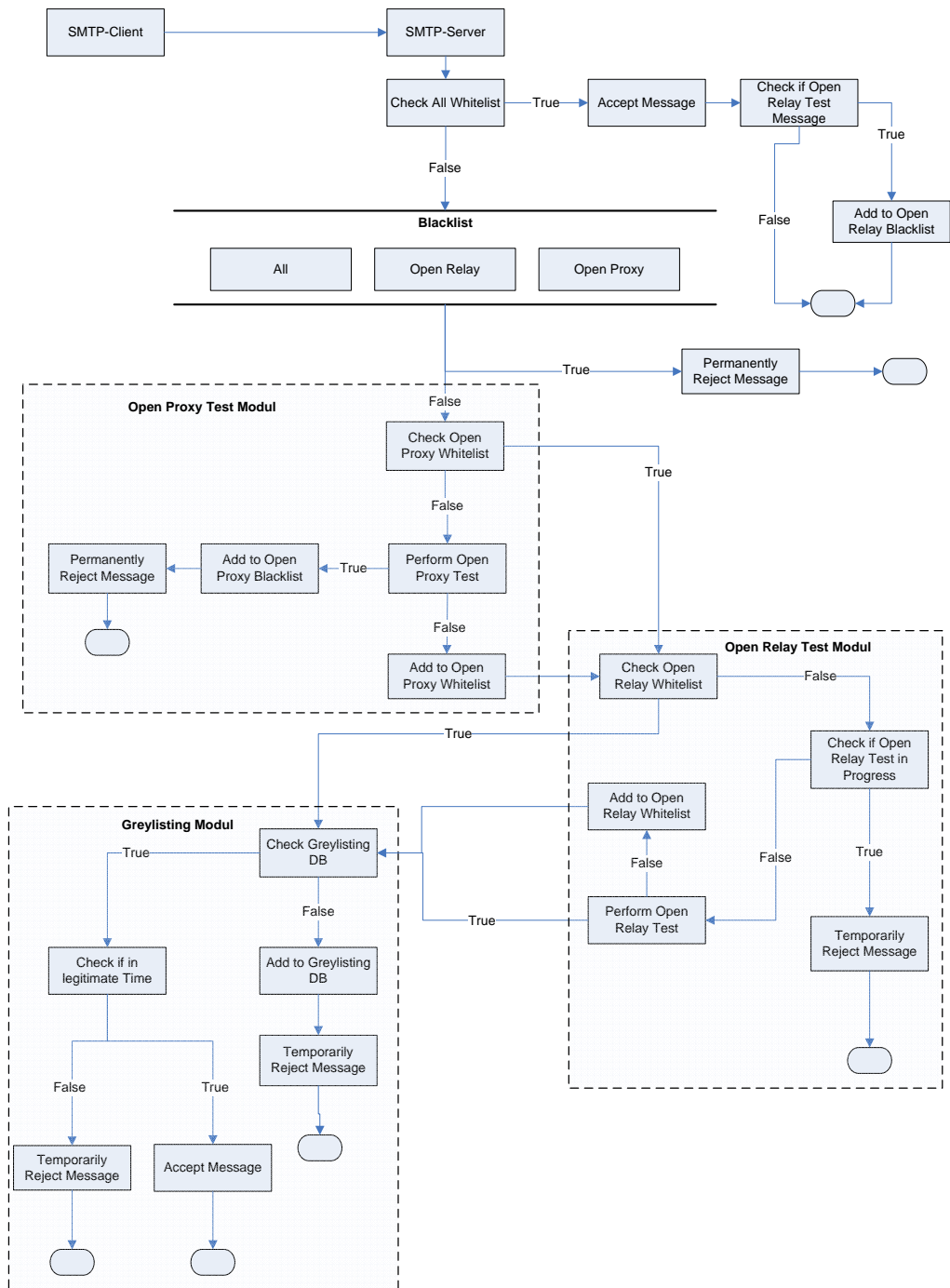


Figure 2.2: Prototype main functionality

Once installed, every incoming connection is handled via this plugin and accepted or rejected according to its output. All necessary settings like server addresses can be done in the *mail.pl* file. Moreover EPF can simply tag messages and deliver everything regardless of its output. More details about installing EPF can be found in the appendix.

Chapter 3

Cost-Based Spam Prevention

In this chapter we focus on pre-send spam prevention methods. Since the main motivation of spammers is to do business and to make profit, the most promising among these methods are cost-based. “Old” protocols like SMTP allow them to hide their identity easily, as described in Chapter 2. We give an overview over the economic background of the spam phenomenon and analyze three possible spam business models. This analysis shows that in the absence of countermeasures against spamming the senders of spam can make huge profits. In Section 3.3 we discuss two technical solutions for increasing the costs for spammers and thus harming or destroying their business model. The basic idea is to slow down or delay the sending of spam without affecting normal users (too much).

3.1 Cost-based Approaches

According to our anti-spam characterization tree (see [1, Chapter 2]), methods can be divided into pre-send and post-send ones. Post-send methods offer a big advantage for categorization: There is more utilizable information available. Unfortunately they have major disadvantages, too: the message must be received before the classification can be done and the traffic, transportation and storage cost are spent already.

Pre-send methods on the other side would be the perfect anti-spam method, but suffer from a major information imbalance – the sender has much more information about a message than the receiver does and how should the receiver know if he is interested in a specific message or not before he receives it?

There are two basic approaches in the pre-send category: strategies to increase the risk for sending UBE/UCE (Unsolicited Commercial E-mail) in the form of stricter legal regulations and stricter enforcement of these regulations, and strategies to increase the costs for sending e-mail, which we will discuss in this chapter. Legal regulations will not be discussed in this part of the report.

The simple explanation for the recent dramatic rise in the amount of spam sent is an economic one – it seems to be easy to make a lot of money with this kind of advertising. The price per piece for the sender in a traditional marketing campaign is always higher than the price for a receiver to throw the letter or brochure away. The sender always pays the price for

transportation and therefore marketers try to address only potential customers. By using modern mass communication methods, the cost for sending a message is spread amongst three different parties - the sender, the receiver and the community. If it is possible to shift the whole cost back to the sender, then the spammers' business model will sustain damages and they will give up UBE in favor of more tightly focused marketing.

There are two basic strategies for increasing costs. On the one hand, there are technical solutions, which use some kind of delaying of each message and on the other hand, there are money-based solutions which suggest some kind of monetary fees for each e-mail. A basic description of these methods can be found in our previous report. Here we want to highlight the major advantages and disadvantages of these methods and we show some experimental results with parameters used in scientific research and their influence on the balance of a spammer's business.

The following section gives a short overview about scientific work done in this area, the economic background and the major facts; the next sections summarize some results made during our experiments and the most important advantages and disadvantages of these models and give a short overview over further possible research objectives.

3.1.1 Scientific Work

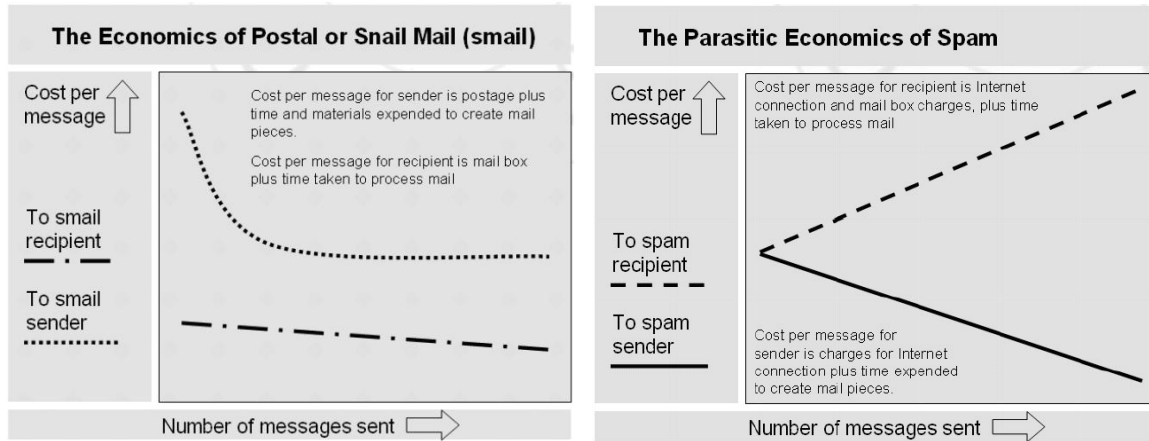
In contrast to post-send methods, the area of pre-send methods is not investigated that much in scientific work. Taughannock Network [22] describes the major facts about E-postage and concludes that these systems will fail due to unforeseen both technical and social barriers. T. Loder et al. [23] suggest a screening and signaling mechanism for senders and receivers wherein people who knowingly misuse communication should have higher communication cost and conclude that a mechanism like this may even be superior to a perfect filter that costs nothing. R. Kraut et al. [24] found out, that charging fees on messages causes senders to be more selective and to send fewer messages, which means variable internet/online rates are better than flat rate packages.

3.1.2 Economic Background

The classic marketing-mix consists of four different components – price, product, place and promotion. Basic promotion methods are media advertising (television, magazines, Internet, e-mail, radio), personal selling, non-personal communication (persuasion advertising – competitions, free samples) and other promotional types like public relation exercises and free publicity.

For these marketing methods, the costs associated with every step are significant and increase proportionally with the number of potential customers reached. Revenue is only created by selling real products or services – initial investment is necessary for advertising in order to make profit afterwards. In the following, we outline the different cost structures of snail mail (letter post) and e-mail.

Snail Mail vs. E-Mail: The production of one letter is expensive because it must be produced by hand, must be brought to the post office and the postal charges have to be paid. Due to



(a) The economics of postal mail [25]

(b) The parasitic economics of spam [26]

Figure 3.1: Comparison of postal mail and spam mail

printing and handling efficiencies and bulk postage rates the cost per message declines as volume increases till a lower bound, where all efficiencies are exploited. What does this mean to the recipient? The per message cost is low and relatively fixed for a recipient, because most people check their snail mail once a day and it is easy to identify brochures unworthy for a single individual to read. Fig. 3.1(a) shows the different cost functions and the fact, that costs for a sender are always higher than cost for recipients.

In contrast, most mail users are usually checking their messages a few times a day, which makes sorting out the “good from the bad” much more time consuming and therefore more expensive. Spammers are constantly coming up with new sorts of messages and change their behavior over time, which makes it very difficult to distinguish between normal and spam e-mail¹ – so message costs rise with volume. On the contrary, there are some initial costs for spammers, but message costs decline rapidly with volume. Fig. 3.1(b) shows the cost curves for e-mail, outlining that the recipient must pay a significantly higher part of the cost per message than the spammer.

Example: A business commissions a marketing company to sell software programs for 49.95 Euro and grants 19 Euro per sold item. A traditional marketing campaign with 5,000 brochures would cost about 5,000 Euro [25]. To cover these expenses the marketing company must have a return rate above 5%, which equals 263 sold pieces so every piece above this will bring 19 Euro of revenue.

Now let us compare this traditional campaign with a new e-mail based one. The amount of sold items of software is directly dependent on the response rate, which is a priori difficult to calculate. Graham [27] estimates a response rate of 0.000015% and DoubleClick [28] gives a rate of 0.35% of orders per delivered e-mail. The cost per spam e-mail ranges from 0.00000492 [26]

¹This is one reason why it is very difficult to create a 100% accurate mail filter.

<i>Example</i>	<i>Response rate [%]</i>	<i>Mails per order</i>	<i>Cost per mail [Euro]</i>	<i>Total costs [Euro]</i>	<i>Revenue [Euro]</i>	<i>Profit [Euro]</i>
<i>1</i>	<i>0.000015 %</i>	<i>66667</i>	<i>0.00000492</i>	<i>0.33</i>	<i>19.00</i>	<i>18.67</i>
<i>2</i>	<i>0.000015 %</i>	<i>66667</i>	<i>0.004</i>	<i>266.67</i>	<i>19.00</i>	<i>-247.67</i>
<i>3</i>	<i>0.35 %</i>	<i>286</i>	<i>0.00000492</i>	<i>0.001</i>	<i>19.00</i>	<i>19.00</i>
<i>4</i>	<i>0.35 %</i>	<i>286</i>	<i>0.004</i>	<i>1.14</i>	<i>19.00</i>	<i>17.86</i>

Table 3.1: Influence of response rate and cost per e-mail on spammers’ profit

to 0.004 Euro [29].

Table 3.1 shows that all economic models are dependent on the input parameters, in this example the response rate and cost per mail. As it is possible to send 691,200 messages from a 256 KBit/second upload line (no BCC²), average message size of 4 KB), the possible profit for example one, three and four would be between 193 and 43 164 Euro per day, whereas losses for the spammer in case two would be tremendous.

High cost per message and a low response rate are the major factors to reduce spammers’ revenue. The following section summarizes general facts and data about the most important input parameters found in literature.

3.1.3 General Information and Facts

Radicati [30], a market research corporation, estimates a number of 76.8 Billion sent messages per day in their quarterly report (Q4 2004) and this rate will rise up to 147.5 Billion in the year 2008 (cf. Fig 3.2). TNS Infratest [31] estimates the total number of internet users with 750 Million in 2004, which means that every user gets 104 messages per day. Postini estimates the spam percentage with 80% [32], which equals 61.44 Billion spam messages per day or 82 per user.

The number of computers with DNS entries is according to [33] 285,139,107, which means that around 270 messages are sent per day and computer. About 40% of the world’s spam (that is 24.58 Billion) is sent out from “zombie computers”. Fig. 3.3 shows the number of internet users from 1991 until 2003 [31].

Response rate: One of the most important factors in spamming is the response rate, which depends on different hard to preview factors. TNS [31] figures out that e-mail advertising is still a growing market, but customer acceptance is going down for two main reasons - the high quantity of spam and the poor quality of marketing. Nevertheless, the revenue per e-mail is estimated with 0.26 US\$. DoubleClick observed in its quarterly report [28] that 90.6% of the messages have been delivered to the recipient, 32.6% opened it and 8.0% followed a link advertised in the message (cf. Fig. 3.4(a)).

Much more importance has the click-to-purchase conversion rate and the orders per delivered message shown in Fig. 3.4(b).

²Blind Carbon Copy

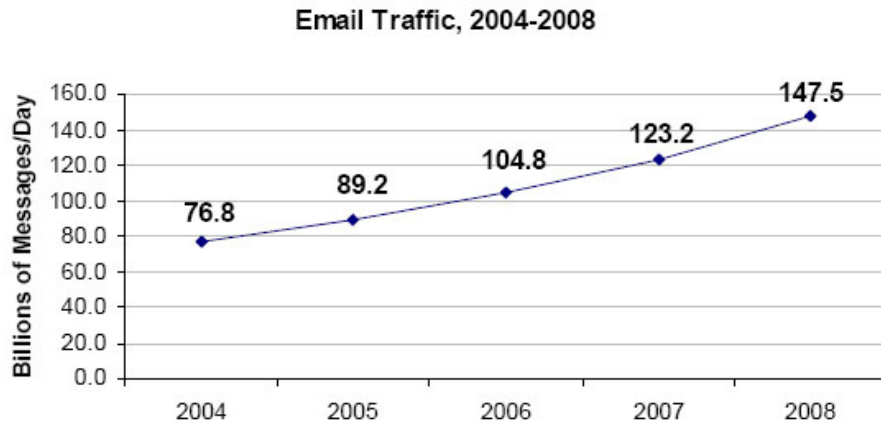


Figure 3.2: E-mail traffic 2004 – 2008 [30]

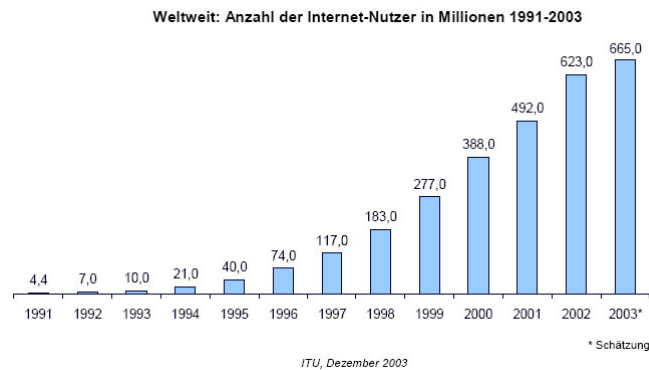
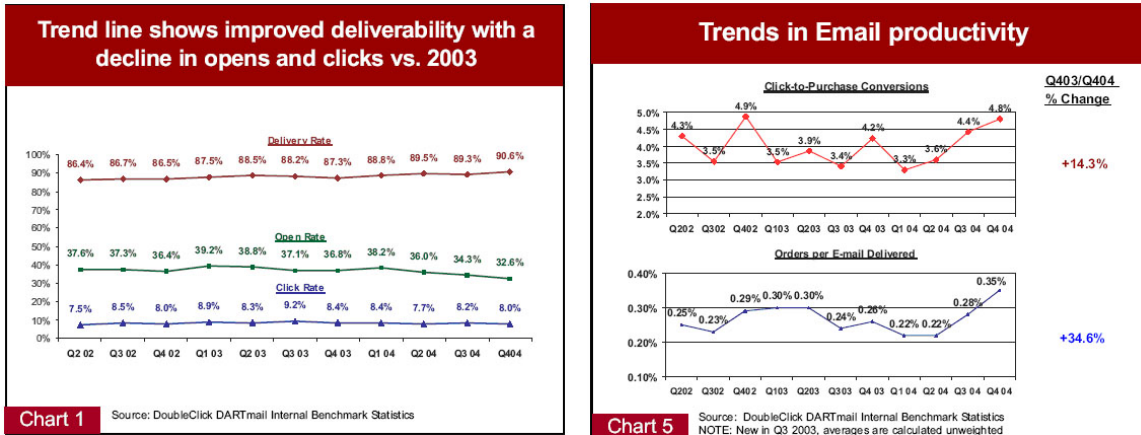


Figure 3.3: Number of worldwide internet users [31]

There are two kinds of response rates reflecting the order per e-mail ratio – the response rate to “normal, legal” online marketing and for spam campaigns. Portmann [34] estimates the rate depending on target groups between 0.5% and 10%, other sources [35], [36] report rates between 1% and 3% and estimated costs per message between 1 and 2 US\$. The response rates for spam campaigns vary from 0.00001% [37] to 0.35% [28]. The true cost per spam e-mail is very difficult to calculate and no spammer would publish them. Some sources [38], [36] report 0.004 Euro per sent spam message, but the true cost would be much less.

To find out the relevant parameters, we would have to start a spam campaign by ourselves, which we did not due to legal restrictions. Therefore, we tried to do our best, took some of the mentioned parameters and put it into three example models, which we want to describe in the following sections. All scenarios describe a simple spam models (individual spammer) – one is renting a server in a country with no legal restrictions, one sends spam using a usual home PC and connection and one acts as a sales agent.



(a) Ratio of delivery rate, open rate and click rate [28] (b) Number of orders per e-mail delivered [28]

Figure 3.4: Trends in mail productivity and deliverability

Example: A single computer using an ADSL line with 256 KBit/second upload limit, may send out 691,200 messages per day (no BCC, average message size 4 KB). The mean online time lies around three hours per day (in Austria, source [39]), during these three hours it is possible to send out 86,400 messages. To send out the estimated 24.5 Billion messages, 284,500 computers must be hacked. If ISPs use limits like 100 messages per day in sending e-mail (independent from the method used), 82 Million computers must be hacked, that is 28% of all computers with DNS entries.

3.1.4 Cost Factors for Spammers

This section summarizes the central cost factors for spammers. Some of these costs are very difficult to find out and to verify – so in this case we have to rely on estimations. As mentioned in our previous report, there is a big difference in cost structure for a spammer who is only doing advertisement and a spammer who acts as a retailer. As the predominant part acts as advertisers, we took two advertisers and one retailer.

Cost factors: Costs for a single spammer can be divided into four cost types – hardware, software, labor cost and operating cost (cf. Fig. 3.5). Typical hardware cost for computers and peripheral devices are easy to elicit and relatively constant over time. Normal software (e.g. system software) is not treated in this report; we try to focus on spam specific applications: e-mail marketing software, remailer³, mailharvester⁴ and BulletFreeWebHosting⁵. Labor cost summarize all operations from installing the operational system to the creation of a spam

³Anonymous remailer, i. e., an SMTP server that allows sending anonymous e-mail messages.

⁴A mail harvester collects e-mail addresses from web pages or directories.

⁵Hosting without the problem of dealing with complaints or the shut down invoked by the provider.

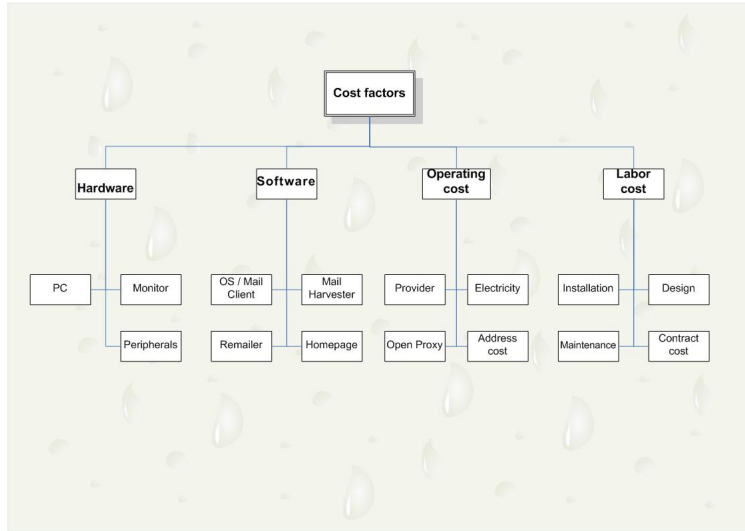


Figure 3.5: Cost categories for spammers [29]

mail. We do not treat labor cost separately, because all described models are based on a single spammer. Further information can be taken from [29].

Revenue factors: Revenue can be made with two different payment schemes – first, a kind of pay per sold item system (compare example in Section 3.1.2) and a pay per campaign system. Most information can be found about the second type – data vary from 0.000399 [40] to 0.09 US\$ [41] per sent e-mail.

3.2 The Economics of a Spammer

The previous section described the most important cost and revenue factors for spam campaigns. Some of these factors like hardware and transmission costs are well documented and do not change significantly over time, but the two most important factors – the response rate and the true cost per e-mail are difficult to handle. We describe possible scenarios for three individual spammers – one renting a server in Asia, one reseller advertising via open relay and one spammer who uses his home PC and a leased line.

3.2.1 Spammer Rents a Server

The following section shows an example of a spammer that uses a rent server. All data was taken from [40], the product name is LegalMail and it offers a Windows 2000 server located in Asia. The key features are as following: it offers a remote interface for transmitting data, e-mail addresses and the message; it grants sending a minimum of one million messages per day (or up to a tenth fold if addresses are well prepared); there is a integrated spam check (where messages can be checked upon their spaminess) and as a special feature dynamic IP address changing

	<i>Fixed costs [Euro]</i>
<i>1 computer with peripheral devices</i>	<i>27.40</i>
<i>Lease costs server</i>	<i>996.85</i>
<i>Electricity</i>	<i>1.00</i>
<i>Internet access cost</i>	<i>35.00</i>
<i>Total:</i>	<i>1060.25</i>

Table 3.2: Fixed cost per month for renting a server in Asia [40]

every ten minutes. Additionally they sell e-mail addresses. Table 3.2 shows the results in Euro on a monthly basis. (The price for the computer including peripherals is estimated with 1000 Euro and a usage duration of 36 month, electricity cost of the server are covered with renting costs).

Together with 12 million e-mail addresses bought at [40] for 184.14 Euro the total cost in this example is 1,244.39 Euro with a guaranteed transmission of 30 million messages per month (per message 0.0000415 Euro). As according to [35] the mean revenue per message is 0.00434 Euro – the profit in this scenario would be 128,955 Euro per month.

3.2.2 Spammer as Sales Agent

The following example gives an impression how spammers’ businesses operate. The description is based on the interview [42] with an anonymous spammer who runs a rather small-scale operation. More details can be found in [1, Section 1.3.1.3].

Table 3.3 shows the cost factors in Euro for a single spammer on a daily basis.

In this case spamming costs 97.73 Euro per day. With this kind of account it is possible to send out 691 200 spam messages – so the cost per e-mail is around 0.000130 Euro. With the same revenue as in the previous section the spammer will earn 0.00421 Euro per message or 2 910 Euro per day or around 87 300 Euro per month.

3.2.3 Spammer as Online Marketer

This section shows a simple example of a spammer, who uses his own home PC and a leased line for spamming. This example is naturally dependent on the general terms and conditions of the ISP, a normal ISP would monitor outgoing traffic and will recognize such a campaign and stop it due to complaints.

With the given revenue per e-mail of 0.00434 Euro, the spammer must send 863 messages a day for balancing costs and profits. With the given maximum rate of 691 200 messages per day, the cost per e-mail is 0.00000541 Euro, the theoretical daily profit would be 2,996 Euro, which means 89 882 Euro per month. To avoid closing of contract with your ISP, it is possible to use so called free mail accounts. Goodman et al. [43] give an example how much the opening of a free mail account costs – they suggest that opening would last about one minute (labor cost equal 10 Euro per hour) – so there are total cost of 0.16 Euro per account. Assuming that every

<i>Hardware</i>			
<i>Type of cost</i>	<i>Single cost</i>	<i>Economic life</i>	<i>Cost per day</i>
<i>Average computer</i>	<i>700.-</i>	<i>3 years</i>	<i>0.65</i>
<i>Average Monitor</i>	<i>300.-</i>	<i>3 years</i>	<i>0.28</i>
<i>Peripherals</i>	<i>100.-</i>	<i>3 years</i>	<i>0.09</i>
<i>Sum hardware:</i>			<i>1.02</i>
<i>Initial costs</i>			
<i>Adresses</i>	<i>300.-</i>	<i>3 years</i>	<i>0.28</i>
<i>Sum initial costs:</i>			<i>0.28</i>
<i>Operating costs</i>			
<i>ISP cost</i>	<i>49.-</i>	<i>-</i>	<i>1.63</i>
<i>Electricity</i>	<i>0.14/KWh</i>	<i>-</i>	<i>1.18</i>
<i>Open proxy cost</i>	<i>0.000125</i>	<i>For 691 200 e-mails</i>	<i>86.40</i>
<i>Sum operating costs:</i>			<i>88.32</i>
<i>Total:</i>			<i>89.62</i>

Table 3.3: Cost factors – single spammer as sales agent, monthly basis [29]. Costs for electricity according to österreichische Bundeswettbewerbbehörde, Dezember 2004.

<i>Appellation</i>	<i>Single cost</i>	<i>Economic life</i>	<i>Cost per day</i>
<i>Average computer</i>	<i>700.-</i>	<i>3 years</i>	<i>0.65</i>
<i>Average Monitor</i>	<i>300.-</i>	<i>3 years</i>	<i>0.28</i>
<i>Peripherals</i>	<i>100.-</i>	<i>3 years</i>	<i>0.09</i>
<i>Electricity</i>	<i>350W/h</i>	<i>Price per KWh=0.14</i>	<i>1.18</i>
<i>Internet Account</i>	<i>49.-</i>	<i>One month</i>	<i>1.63</i>
<i>Sum:</i>			<i>3.74</i>

Table 3.4: Cost factors – single spammer as online marketer, daily basis [29]. Costs for electricity according to österreichische Bundeswettbewerbbehörde, Dezember 2004.

new free mail account is closed after 24 hours due to complaints, the total cost per day are 3.56 Euro and therefor similar to the above example.

Summary: This section showed three examples of spammers doing rather small-scale business, further work must be done to investigate professional spammers with multiple computers and many employed technicians. The results show that this business must be very profitable, because their profit (before tax and without their labor cost) is between 89,760 and 128,955 Euro per month. So how can we harm their business? There are two factors, which have significant importance – the response rate, which possibly may be influenced by a better intelligence of e-mail readers, and the number of messages that can be sent in a specific time interval. The number of possible messages can be influenced – a few possible solutions are presented in the following section.

3.3 Increasing Sender Costs

The business models described in the previous section show that if there are no countermeasures against spamming, spammers would never stop their business due to economic reasons. Methods, filtering out spam at the client side must reach a very high rate to be effective compared to the data computed in Section 3.2. The conclusion is to find methods to avoid spamming, namely raise costs for sending every single message or limit sending capabilities with technical solutions.

3.3.1 Money Based Costs

The basic idea behind money-based solutions is to “pay” some amount of a possibly symbolic currency for each e-mail to be sent. One solution to this problem is the Lightweight Currency Protocol (LCP) [44] – a micro payment system described previously in [1]. It forces spammers to send their e-mail more selectively and only to potential customers. Therefore, it will not stop spamming, but it will bring back spammers to work like serious online-marketers and therefore will raise cost per e-mail. There are still some shortcomings (lack of a global micro payment system) and this method will not be sufficient used as a single mechanism. It may be one possible add-on for our second project.

3.3.2 Time Costs (Delay)

Within technical solutions, the sender of an e-mail is required to compute a moderately expensive function – a so-called pricing function – before an e-mail is sent. The cost per message therefore is paid through a time delay. The amount of time consumed should be transparent for a normal user, but should be very disturbing for a spammer, because it reduces the number of potential customers reached per unit of time. In the following, we want to take a closer look at Hashcash and Memory-Bound functions.

Hashcash

A closer description of Hashcash can be found in [1, Section 2.2.1.1.], [45] and [46]. Here, we want to focus on the performance of Hashcash on different computers and their influence on the business model of a spammer. Hashcash computes partial hash collisions – the time to compute such a collision is dependent on two factors – the length of the collision and the performance of the CPU. Table 3.5 shows the results of our test runs on different computers with different collision lengths.

For computing we used Adam Back’s Hashcash.exe, we made 10 runs and took the mean values. The table shows that a slow laptop computer would need 590 seconds to compute a 28 bit partial collision, whereas a modern fast CPU (Athlon 64) just needs 28 seconds. Fig. 3.6 shows the results graphically.

The business models described in 3.2 show that spammers are dependent on the amount of spam sent. Hashcash slows down the speed and therefore will reduce the profit. The remaining question is: how strong is the influence?

Computer	Processor	Clock rate	10 bit	15 bit	20 bit	22 bit	24 bit	26 bit	28 bit	30 bit
Laptop	MMX	233MHz	0.16	0.25	2.43	11.08	37.31	171.70	590.38	n/a
Desktop	PIII	450 MHz	0.14	0.17	1.04	4.22	21.44	42.48	355.65	1400.00
Laptop	P M	1.7 GHz	0.03	0.06	0.28	0.99	5.27	8.97	43.40	135.97
Desktop	P IV	2.4 GHz	0.05	0.06	0.40	1.74	3.47	21.78	178.73	395.99
Desktop	Athl.XP	2600+	0.04	0.05	0.41	1.01	4.05	18.71	90.90	274.45
Desktop	Athl.64	3500+	0.03	0.03	0.26	0.96	2.59	20.57	28.18	293.01

Table 3.5: Computation time in seconds for different partial collisions on different CPUs

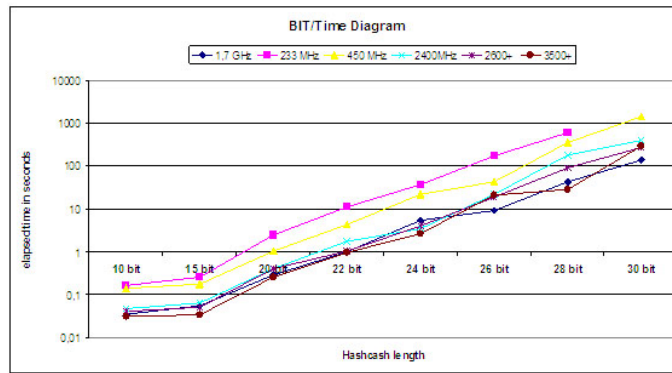


Figure 3.6: Bit length to elapsed time ratio

Table 3.6 shows the influence of the used strength of collision on the maximum e-mail sent (basic assumption ADSL lease line 256 KBit/sec upload, compare Section 3.1.2).

Fig. 3.7 shows the influence of the usage of Hashcash on the profit of the spammers' (rent a server) business (compare Section 3.2.1, cost per e-mail = 0.0000415, revenue per e-mail = 0.00434). The results for the other models are similar. Table 3.7 shows the possible revenue in

Clock rate	10 bit	15 bit	20 bit	22 bit	24 bit	26 bit	28 bit	30 bit
MMX 233MHz	540000,00	345600,00	35614,18	7797,83	2315,80	503,21	146,35	n/a
PIII 450MHz	626086,96	499421,97	82997,12	20483,64	4030,60	2033,66	242,94	61,71
PM 1700MHz	2541176,47	1570909,09	304225,35	87184,66	16382,25	9636,40	1990,69	635,45
P4 2400MHz	1878260,87	1350000,00	215461,35	49683,73	24906,31	3966,40	483,40	218,19
Ath. XP2600+	2107317,07	1728000,00	212285,01	85714,29	21359,70	4619,09	950,47	314,81
Ath. 643500+	2787096,77	2618181,82	337500,00	90093,85	33410,67	4200,50	3066,22	294,87

Table 3.6: Possible number of outgoing messages depending on CPU and collision size

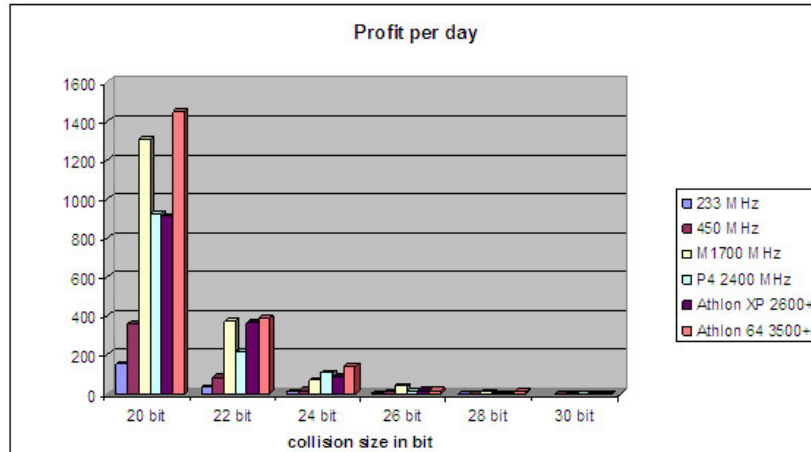


Figure 3.7: Possible profit per day and computer (spammer rents server)

Euro per day for different computers and some possible Hashcash sizes.

Conclusion: Employing Hashcash reduces the output of messages per computer depending on the Hashcash size. As an example let us take a 26-bit collision. A fast computer (spammers would probably use fast machines) is able to compute 4200 partial collisions (one collision equals one message to one recipient) a day, which might produce around 18 Euro profit per day (cf. Table 3.7). A normal user with a slow computer is able to send 503 e-mails per day, which is more than needed by the average user (compared to the data in Section 3.1.3, where every user receives 104 messages on average), but he has to wait 171 seconds until it is sent. During computation the working load of the CPU is close to 100%, but Hashcash can be run in low priority mode and stamps can be produced in advance.

With this reduction of revenue the business model of the spammers is damaged while normal users usually do not feel that they have limited sending capabilities. The computation of a Hashcash stamp is secure and can not easily be forged. The major disadvantages of Hashcash are a potential “unfairness” due to differences in processing speed, and some potential acceptance problems (users tend not to like the idea of having to run rather compute intense jobs in the background).

Memory-bound Functions

CPU-bound pricing functions suffer from a possible mismatch in processing speed on different machines, so Borrows [47] suggested an alternative computational approach based on memory latency. He suggested designing a pricing function that requires a large number of scattered memory access, because memory latencies do not vary as much from between computers as clock speeds do. The goal is to cause the sender to incur some cache misses, which takes some amount of time. Further information can be found at Rosenthal [48] Dwork et al. [49] who analyzed a possible function called MBound and tested different computer architectures.

<i>CPU</i>	<i>Online Marketer</i>	<i>Server Renter</i>	<i>Sales Agent</i>
<i>Hashcash size = 20 bit</i>			
<i>233 MHz</i>	<i>154,39</i>	<i>153,09</i>	<i>149,54</i>
<i>450 MHz</i>	<i>359,80</i>	<i>356,76</i>	<i>348,50</i>
<i>M1700 MHz</i>	<i>1318,84</i>	<i>1307,71</i>	<i>1277,44</i>
<i>P4 2400 MHz</i>	<i>934,04</i>	<i>926,16</i>	<i>904,72</i>
<i>Athlon XP 2600+</i>	<i>920,27</i>	<i>912,51</i>	<i>891,38</i>
<i>Athlon 64 3500+</i>	<i>1463,09</i>	<i>1450,74</i>	<i>1417,16</i>
<i>Hashcash size = 22 bit</i>			
<i>233 MHz</i>	<i>33,80</i>	<i>33,51</i>	<i>32,74</i>
<i>450 MHz</i>	<i>88,80</i>	<i>88,05</i>	<i>86,01</i>
<i>M1700 MHz</i>	<i>377,95</i>	<i>374,76</i>	<i>366,09</i>
<i>P4 2400 MHz</i>	<i>215,38</i>	<i>213,57</i>	<i>208,62</i>
<i>Athlon XP 2600+</i>	<i>371,58</i>	<i>368,44</i>	<i>359,91</i>
<i>Athlon 64 3500+</i>	<i>390,56</i>	<i>387,27</i>	<i>378,30</i>
<i>Hashcash size = 24 bit</i>			
<i>233 MHz</i>	<i>10,04</i>	<i>9,95</i>	<i>9,72</i>
<i>450 MHz</i>	<i>17,47</i>	<i>17,33</i>	<i>16,92</i>
<i>M1700 MHz</i>	<i>71,02</i>	<i>70,42</i>	<i>68,79</i>
<i>P4 2400 MHz</i>	<i>107,97</i>	<i>107,06</i>	<i>104,58</i>
<i>Athlon XP 2600+</i>	<i>92,60</i>	<i>91,81</i>	<i>89,69</i>
<i>Athlon 64 3500+</i>	<i>144,84</i>	<i>143,62</i>	<i>140,29</i>
<i>Hashcash size = 26 bit</i>			
<i>233 MHz</i>	<i>2,18</i>	<i>2,16</i>	<i>2,11</i>
<i>450 MHz</i>	<i>8,81</i>	<i>8,74</i>	<i>8,54</i>
<i>M1700 MHz</i>	<i>41,77</i>	<i>41,42</i>	<i>40,46</i>
<i>P4 2400 MHz</i>	<i>17,19</i>	<i>17,05</i>	<i>16,65</i>
<i>Athlon XP 2600+</i>	<i>20,02</i>	<i>19,86</i>	<i>19,40</i>
<i>Athlon 64 3500+</i>	<i>18,21</i>	<i>18,06</i>	<i>17,64</i>
<i>Hashcash size = 28 bit</i>			
<i>233 MHz</i>	<i>0,63</i>	<i>0,63</i>	<i>0,61</i>
<i>450 MHz</i>	<i>1,05</i>	<i>1,04</i>	<i>1,02</i>
<i>M1700 MHz</i>	<i>8,63</i>	<i>8,56</i>	<i>8,36</i>
<i>P4 2400 MHz</i>	<i>2,10</i>	<i>2,08</i>	<i>2,03</i>
<i>Athlon XP 2600+</i>	<i>4,126</i>	<i>4,09</i>	<i>3,99</i>
<i>Athlon 64 3500+</i>	<i>13,29</i>	<i>13,18</i>	<i>12,88</i>
<i>Hashcash size = 30 bit</i>			
<i>233 MHz</i>	<i>nc</i>	<i>nc</i>	<i>nc</i>
<i>450 MHz</i>	<i>0,27</i>	<i>0,27</i>	<i>0,26</i>
<i>M1700 MHz</i>	<i>2,75</i>	<i>2,73</i>	<i>2,67</i>
<i>P4 2400 MHz</i>	<i>0,95</i>	<i>0,94</i>	<i>0,92</i>
<i>Athlon XP 2600+</i>	<i>1,36</i>	<i>1,35</i>	<i>1,32</i>
<i>Athlon 64 3500+</i>	<i>1,28</i>	<i>1,27</i>	<i>1,24</i>

Table 3.7: Revenue for different computers/Hashcash sizes [29]

Processor	CPU clock	OS	L2 Cache	Memory	Hashcash, 22 bit [s]	Mbound, 15 bit [s]
PIV	3.06 GHz	Linux	256 KB	4 GB	4.44	9.24
PIV	2.0 GHz	Win XP	256 KB	512 MB	8.48	12.17
PIII	1.2 GHz	Win XP	256 KB	512 MB	9.81	9.15
PIII	1.0 GHz	Win XP	256 KB	512 MB	11.85	9.70
Mac G4	1000 MHz	OSX	256 KB	512 MB	8.26	17.93
PIII	933 MHz	Linux	256 KB	512 MB	9.55	9.70
PII	266 MHz	Win 98	512 KB	96 MB	45.15	24.43

Table 3.8: Hashcash vs. MBound [49]

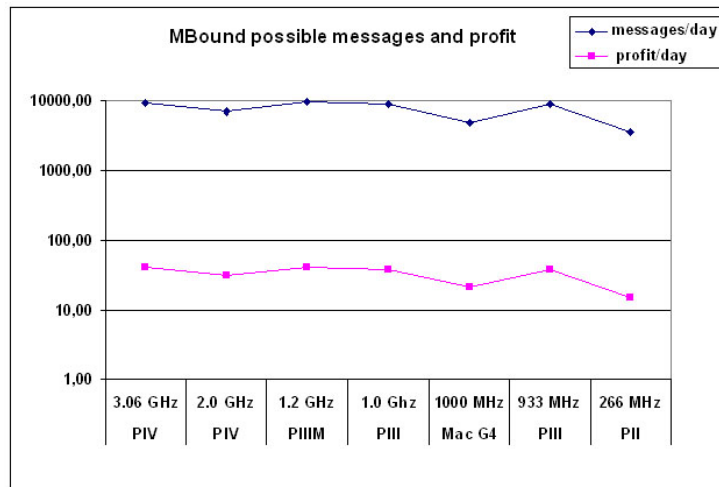


Figure 3.8: Messages sent/profit per computer with MBound 15 bit (spammer rents server)

Table 3.8 summarizes the results taken from Dwork et al.

Table 3.8 shows that the time to compute a 22 bit partial Hashcash collision varies from 4.44 to 45.15 seconds which equals a difference of a factor of 10, while the difference for MBound is only a factor of 2.67.

Fig. 3.8 shows the results for the usage of MBound under the assumption of a 256 KB upload line (compare 3.2.1, cost per e-mail = 0.0000415, revenue per e-mail = 0.00434).

The results for the other models are similar. Table 3.9 shows the possible revenue in Euro per day for different computers and some possible Hashcash sizes.

Conclusion: The influence on the spammers business model is similar to the one of Hashcash, but a memory-bound approach is potentially “fairer”, i. e., it smoothes the differences in hardware performance. However, this approach is not yet so well developed and documented, some more research is needed into suitable memory-bound functions.

		<i>Rent server</i>	<i>Sales agent</i>	<i>Marketer</i>
<i>Processor</i>	<i>CPU clock</i>	<i>profit/day</i>	<i>profit/day</i>	<i>profit/day</i>
<i>PIV</i>	<i>3.06 GHz</i>	<i>40.19</i>	<i>39.26</i>	<i>40.54</i>
<i>PIV</i>	<i>2.0 GHz</i>	<i>30.52</i>	<i>29.81</i>	<i>30.78</i>
<i>PIIIM</i>	<i>1.2 GHz</i>	<i>40.59</i>	<i>39.65</i>	<i>40.93</i>
<i>PIII</i>	<i>1.0 Ghz</i>	<i>38.29</i>	<i>37.40</i>	<i>38.61</i>
<i>Mac G4</i>	<i>1000 MHz</i>	<i>20.71</i>	<i>20.23</i>	<i>20.89</i>
<i>PIII</i>	<i>933 MHz</i>	<i>38.29</i>	<i>37.40</i>	<i>38.61</i>
<i>PII</i>	<i>266 MHz</i>	<i>15.20</i>	<i>14.85</i>	<i>15.33</i>

Table 3.9: Spammers' Profit per day, different cost models [29]

Chapter 4

Experiments

In this chapter we describe some experiments with different SMTP-based and related methods. This comprises an evaluation of the header tests implemented in SpamAssassin [50] as well as an evaluation of the modules of our EPF server.

The evaluation of SpamAssassin's header tests can be performed offline with stored e-mail data, whereas the evaluation of the modules of our EPF server ideally would require *live* e-mail streams. Since we did not have fully satisfactory live streams available so far, some of our experiments are of preliminary nature. In the following, we give more detailed information about the test data used in our experiments.

4.1 Test Data

In an ideal setting, live streams of spam messages as well as of ham messages are available for evaluation of antispam methods. This is especially important in the case of SMTP related methods.

4.1.1 Live Streams

Full evaluation of our EPF server is only possible with some live traffic. However, due to organizational, technical and other difficulties as well as resource limitations, so far it was not possible to produce appropriate live streams. In the case of spam streams, the main question is how to organize the redirection of such a stream from the servers of our project partners to our server. Despite close contact with them we were not yet able to establish this. In the case of ham streams, the situation is even more complicated. First, we would need many volunteers participating in a live test of our method in order to produce a substantial live stream. Second, there are no “ham traps”, i. e., in contrast to spam traps it will never be possible to generate a stream of guaranteed ham messages.

Given these limitations, we decided to use two strategies for evaluating the methods considered in this report: (*i*) using a (small) live spam stream from spam traps available to us for the evaluation of those techniques which absolutely cannot be evaluated with offline data (especially

greylisting), and (ii) use stored offline e-mail data for the other techniques. The results achieved for this live stream are summarized in Section 4.2.2.

4.1.2 Offline Evaluations

The test data used in the offline evaluations mostly originates from our project partners. We used this data for evaluating SpamAssassin's header tests (see Section 4.2.1) and for getting a rough picture about the effectivity of open proxy and open relay tests (see Section 4.2.3).

4.2 Experimental Results

In this section, we summarize our experimental results. First, we discuss the evaluation of SpamAssassin's header-based tests. Afterwards we summarize the preliminary results achieved with our own prototype as well as of our offline experiments with open proxy and open relay checks.

4.2.1 Evaluation of SpamAssassin's Header-Based Tests

Spammers tend to manipulate header entries in order to conceal their identity. SpamAssassin Version 3.0.2 [50] contains 203 tests which concern the header of an e-mail (excluding the subject line). As all SpamAssassin tests, these are post-send tests which are applied after the message has been accepted by the receiving mail server. Since the header information is transferred as part of the data command in the SMTP dialog, header-based approaches require the complete content of the e-mail. As it has been shown in [11], any information in the e-mail header can be forged, except the IP address of the SMTP client recorded by the local MTA. Detecting spam based on header information is limited to poor forgeries and cannot achieve satisfying results in the long run. To confirm this we evaluated the performance of SpamAssassin's header rules.

We took two test samples each consisting of 1000 spam messages. One sample was collected in August 2004, the second one in April 2005. Figure 4.1 depicts clearly that the detection rate achieved by header tests significantly decreases. While in August 2004 around 60 percent of the messages were classified correctly, the detection rate decreases with time, and in April 2005 it was already down at 7 percent! This clearly illustrates the fact that many of the rules implemented in SpamAssassin have an *ad hoc* nature. Once it is known which rules are used, spammers can easily react and bypass them.

This dynamics is also reflected in a significant decrease in the number of rules triggered during the test phase. Figures 4.2 and 4.3 show that while the test set from August 2004 triggers 47 percent of the header rules, for the newer test set from April 2005 the portion decreases to 21 percent.

Relevance of Individual SpamAssassin Header Rules

Another interesting aspect is the high variation of relevance in the rule sets of SpamAssassin, in particular in the header rules, in particular the large portion of rules which are only marginally relevant (on average). With a small number of selected rules nearly the same detection rate

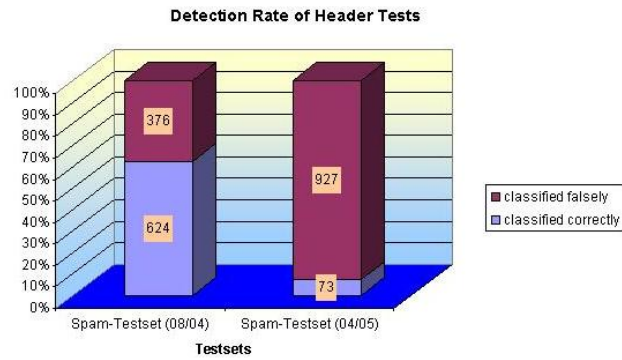


Figure 4.1: Detection rate of header tests

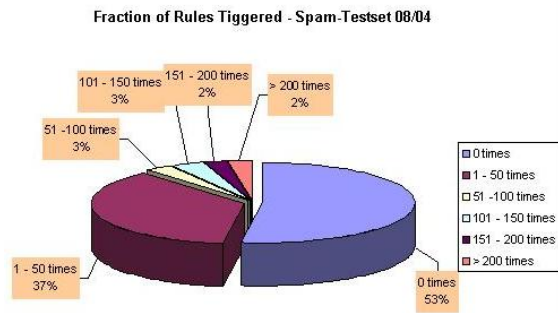


Figure 4.2: Fraction of rules triggered - spam-test set 08/04

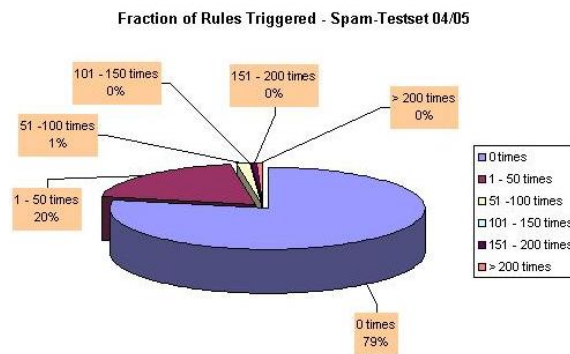


Figure 4.3: Fraction of rules triggered - spam-test set 04/05

Total connections	233	
Rejected	231	99.1 %
Accepted	2	0.9 %
<i>Greylisted</i>	109	46.8 %
<i>Open proxies (total)</i>	50	21.5 %
<i>(tested)</i>	36	15.5 %
<i>(blacklisted)</i>	14	6.0 %
<i>Open relays</i>	0	0.0 %
<i>Accepted</i>	2	0.9 %
<i>Connection errors</i>	72	30.9 %

Table 4.1: EPF results using a spam trap from March 22 to May 9, 2005

results can be achieved as with the entire set. A closer look reveals that with only 10 to 15 out of the 203 rules about 90% of the detection rate of the total rule set can be achieved for both test sets [11].

These observations confirm the ad hoc character of most of the 203 header rules and their very limited range of applicability. Many of these rules were added to cope with special individual messages, and therefore cannot increase the detection rate in the same way as more general criteria do.

In summary, this illustrates that the header based rules integrated in SpamAssassin do not provide a persistent and efficient method. The reason is that these rules are too specific, case study oriented and too easy to bypass if known.

4.2.2 Live Stream Evaluation of EPF

Based on the observations made with SpamAssassin’s header rules, our focus was on identifying more fundamental criteria accessible in the SMTP transfer process, as described in Sections 2.1 and 2.3. Here, we report on the results of a preliminary evaluation of our EPF server developed in phases 2 and 3 of the project.

Due to the methods used in our EPF server, it is essential to test it with *live* traffic. As summarized in Section 4.1, we used spamtrap addresses for creating a live spam stream. Unfortunately, the volume of this spam stream available to us is (still) rather small and thus so far we can only present preliminary results.

Table 4.1 summarizes our experimental results from March 22 until May 9, 2005. In total, 233 SMTP connections to spam traps at our server have been established in this period. In this rather small sample, only a small fraction of the incoming connections originated from open proxies and none from open relays. Nevertheless, almost *all* of the incoming spams were rejected. Astoundingly, *greylisting* blocked almost all other connections. This illustrates that the vast majority of connecting mail servers used by the spammers were configured/set up conforming to the recommendations of the SMTP standard [12].

For comparison, Table 4.2 summarizes a selection of outgoing e-mail servers from which we sent ham messages to our EPF server. All these outgoing servers have been configured

<i>Server Name</i>	<i>Type</i>	<i>Server Farm</i>	<i>Connections</i>	<i>retries after [min]</i>
<i>mailbackup.inode.at</i>	<i>exim</i>	<i>yes</i>	<i>3 (2)</i>	<i>0, 30</i>
<i>viefep20-int.chello.at</i>	<i>InterMail</i>	<i>yes</i>	<i>5 (4)</i>	<i>1, 2</i> <i>15, 20</i>
<i>main.blackbox.net</i>	<i>sendmail</i>	<i>no</i>	<i>2</i>	<i>2</i>
<i>fv-win.at</i>	<i>qmail</i>	<i>no</i>	<i>2</i>	<i>6</i>
<i>winf.htu.tuwien.ac.at</i>	<i>postfix</i>	<i>no</i>	<i>2</i>	<i>35</i>
<i>einstein.ani.univie.ac.at</i>	<i>exchange</i>	<i>no</i>	<i>2</i>	<i>2</i>
<i>gmx.net</i>	<i>qmail</i>	<i>no</i>	<i>2</i>	<i>6</i>
<i>telis.winf1.at</i>	<i>qmail</i>	<i>no</i>	<i>2</i>	<i>6</i>
<i>aristophanes.winf1.at</i>	<i>Mercury</i>	<i>no</i>	<i>2</i>	<i>30</i>
<i>bay23-f28.bay23.hotmail.com</i>	<i>MS SMTPSVC</i>	<i>no</i>	<i>2</i>	<i>3</i>
<i>nemo.ani.univie.ac.at</i>	<i>postfix</i>	<i>no</i>	<i>2</i>	<i>32</i>
<i>uhura1.kom.tuwien.ac.at</i>	<i>supermail</i>	<i>yes</i>	<i>3 (2)</i>	<i>16, 26</i>
<i>klutz.cs.utk.edu</i>	<i>postfix</i>	<i>no</i>	<i>2</i>	<i>28.5</i>
<i>networld.at</i>	<i>sendmail</i>	<i>no</i>	<i>2</i>	<i>ca. 5</i>
<i>gmail.com</i>	<i>sendmail</i>	<i>yes</i>	<i>3 (2)</i>	<i>5, 10</i>
<i>aon.at</i>	<i>postfix</i>	<i>no</i>	<i>2</i>	<i>ca. 15</i>
<i>sccm.Stanford.EDU</i>	<i>exim</i>	<i>no</i>	<i>2</i>	<i>2</i>
<i>stanfordalumni.org</i>	<i>postfix</i>	<i>no</i>	<i>2</i>	<i>4</i>

Table 4.2: Mail servers and their greylisting compatibility. “*Server Farm*” means that several computers (and therefore different IPs) process messages for a certain domain. “*Connections*” lists the number of established connections and, if applicable, the number of different sources.

conforming to the recommendations of the SMTP standard and thus passed our greylisting test (and, of course, also the open proxy and open relay checks). The table shows the name of the sending machine, the type of mail server installed (if indicated in the SMTP dialog), whether the e-mail came from a server farm, the number of connections needed (more than two only in the case of server farms) and a rough estimation of the time the servers waited before they resent after receiving a temporary error message. From the 18 outgoing SMTP servers tested not a single message was blocked by the greylisting of our EPF server. On the contrary, all ham messages were delivered within at most one hour.

4.2.3 Offline Evaluation of Open Proxy and Open Relay Tests

Due to the relatively low amount of messages available for online testing we decided to complete our evaluations using stored offline data where possible. We extracted sender IP addresses from the headers of stored spam messages (offline) and performed open proxy and open relay tests for these addresses. A potential weakness of this approach is the fact that being an open proxy or an open relay may be a *temporary* feature of an e-mail server and open proxies or open relays may be closed down soon after sending messages. In order to reduce the risk of wrongly classifying hosts, they were tested within a period of at most 48 hours after actual receipt of the message.

Altogether, we extracted 3000 IP addresses from spam messages and tested whether the

corresponding hosts showed the behaviour of open proxies or open relays. In this sample, only a small fraction (1.6%) was identified as open proxy while no host could be identified as open relay. As mentioned before, due to the fact that it was not possible to perform these tests on larger samples in real time, the results are influenced and potentially somewhat flawed by the delay between the actual transfer of the e-mail and the examination of the source.

4.3 Performance Issues

For any solution to be employed in large-scale environments performance is an important issue. Servers have to be able to deal with large amounts of messages without stalling. The methods we investigated in this report clearly raise a few performance questions.

By far the most resource consuming test is the open relay check. It includes the sending of a test message to the connecting mail server and waiting for its return to the local machine. This uses not only resources on both, our own machine as well as on the connecting mail server but also holds the risk of getting blacklisted (if the primary mail server is used for the check).

The other two components used are less performance critical. The open proxy check needs fewer resources than the open relay check because no messages have to be sent (as mentioned in Section 2.2.3). Greylisting causes delays in the arrival of e-mail messages, but does not imply performance problems. Spam messages that are not resent due to a badly implemented spammer tool or due to a misconfigured mail server do not use any resources at all since it consists of receiving the data shown in Listing 2.1 and sending back one line of data, namely the temporary error code. Incoming ham may be rejected once, involving the same small amount of data to be sent back, but will be accepted from then on.

In order to minimize the overhead caused by open relay, open proxy tests and by greylisting they can be implemented in a *dynamic form* based on dynamically updated lists with an expiration date for each entry. We will integrate this improvement in the next version of EPF.

Chapter 5

Conclusions

On the basis of our comprehensive overview [1] of the current state-of-the-art in spam defense we have summarized our investigations into two important research directions: (i) the potential and the limitations of approaches based on the standard SMTP transfer process, and (ii) the spammers' business models which are the motivation underlying the spam phenomenon.

5.1 Standard SMTP Transfer Process

We have thoroughly investigated the potential and the limitations of spam defense methods based on the transfer process of e-mail messages using standard SMTP. Our results can be summarized as follows (for more details see also [11]):

- The standard SMTP protocol is not able to support an infrastructure for efficient spam defense methods.
- Methods using ad-hoc rules for SMTP related header information (e.g., as implemented in SpamAssassin) can achieve reasonable results *temporarily*. However, their performance deteriorates quickly with time. Thus, the maintenance costs for achieving satisfactory performance over a longer period of time with such approaches tend to be prohibitive.
- There are only very few basic properties of the SMTP-based e-mail transfer process which can provide reasonably reliable indications about whether an e-mail should be considered as spam or not.
- We have implemented a prototype of an extended SMTP server which integrates checks for these basic properties into the standard SMTP transfer process.
- First experimental experience gained with this prototype confirms theoretical analysis: The significance of these basic properties may not always be high enough to justify the rejection of an e-mail message. However, they are important features to be included into a profile which forms the basis for our profile-based spam defense currently under development.

With these findings and with the implementation of the EPF prototype our research activities in the area of SMTP-based spam defense are completed. These results will be important building blocks in the development of a comprehensive profile-based antispam methodology (see Section 5.3).

5.2 Spammers' Business Model

In order to investigate new methods which try to fight the spam problem at its source (pre-send methods), we performed a careful investigation of the motivation behind the phenomenon spam – the business dynamics of spam.

Our results described in this report can be summarized as follows:

- In the vast majority of cases the sending of spam is motivated by economic incentives (profit from advertizing).
- The business model of spammers is based on sending out big amounts of e-mail messages, which currently can be done very cheaply.
- There is a wide gap between the number of e-mail messages sent out by a spammer in order to be profitable and the number of e-mail messages sent out by a “regular” user.
- This gap provides the opportunity for developing pre-send methods which harm the spammers' business model without affecting a “regular” user. Our first steps in this direction will be continued in the project “Spamabwehr II”.
- Those pre-send methods are expected to substantially reducing the spam problem by reducing the amount of spam. However, due to the nature of the internet and the current e-mail infrastructure they will not suffice to fully control the problem. A comprehensive approach is required comprising improved post-send methods as well as those new pre-send methods to achieve satisfactory performance of spam defense systems.

5.3 Future Directions

Important future research directions include innovative approaches for controlling outgoing e-mail traffic (new memory-bound functions, transparent sending delays, different “payment” systems, etc.), advances in adaptive and highly accurate classification techniques for incoming e-mail traffic, and, last but not least, the investigation and development of a methodology for combining those approaches into a comprehensive spam defense system which will lead to a reduction of the spam portion in e-mail traffic to a negligible level.

We will pursue all these directions in the context of the continuation research project “Spamabwehr II”. Our ideas for the profile-based classification technology will be the core of a comprehensive spam defense system as mentioned before.

Beyond the topics we plan to pursue in the course of the next months, several other areas deserve special attention. Among these, we certainly need to highlight efforts for developing

new e-mail transfer protocols and also initiatives for establishing better and more uniform legal regulations. Activities in these areas certainly require a broader base (institutionally as well as geographically) than that of our project. Nevertheless, we will continue to closely monitor developments in these areas and adapt our research activities accordingly.

Appendix

Prototype Installation Notes

We presented a Perl prototype to implement greylisting, open proxy and open relay checking. EPF is implemented as plugin for the Postfix SMTP server and therefore requires an existing, working Postfix installation. The installation of this prototype is described in the following.

The file *epf-0.1.tar.gz* contains the following files:

- Database.pm - database related module
- DatabaseStatics.pm - contains static settings for database access
- main.pl - the main script containing all application logic
- proxycheck - binary checking whether given hosts are open proxies or not
- reportopenrelay.pl - used to catch returning test mails from open relays

This archive file is available on request from the authors of this report. In the following we will give step by step instructions to setup our prototype for an existing Postfix installation.

- Create a folder */var/mta* for the database files and change its ownership to nobody *<chown nobody /var/mta>*. Note that all contents of mta must be deleted if you want to reset the databases (e.g. *<rm -r /var/mta/*>*)
- Create a folder */usr/bin/postfixscripts/proto*
- Copy all files from the archive file into this folder
- Install required Perl modules
 - install the Perl package *Cache*, e.g. via CPAN

Moreover, the following settings in Postfix configuration files must be done.

- Install Postfix (tested with postfix-2.1.5-r2)

- You have to be superuser to change Postfix configuration files
- Start with checking your Postfix installation (only continue if no errors are reported, i.e. your installation is ok)
 - * E.g. type `<postfix check>` etc.
- Create a file `/etc/postfix/recipient_checks` (i.e. the postfix whitelist) containing:


```
postmaster@    OK
relaycheck@    OK
```
- Create a `.db` file by typing: `<postmap /etc/postfix/recipient_checks>`
- Add the following to the `/etc/postfix/master.cf` file:


```
# first line
policy-proto  unix  -  n  n  -  -  spawn
# second line
    user=nobody argv=/usr/bin/perl
    -w /usr/bin/postfixscripts/proto/main.pl
```
- Add the following to the `/etc/mail/aliases` file (on one line):


```
relaycheck: "| /usr/bin/perl
/usr/bin/postfixscripts/proto/reportopenrelay.pl"
```
- Submit the changes by typing: `<newaliases>`
- Add the following to the `/etc/postfix/main.cf` file:


```
unknown_address_reject_code = 550

# clients == incoming IPs
smtpd_client_restrictions =
    check_recipient_access hash:/etc/postfix/recipient_checks,
    reject_unknown_sender_domain,
    check_policy_service unix:private/policy-proto

unknown_address_reject sets the default action for connections from unknown addresses,
reject_unknown_sender_domain applies that action to incoming connections.
```
- Restart and recheck Postfix
 - `<postfix check && /etc/init.d/postfix restart>`
- Remember not to use metalog as system logger, it has problems with the Perl routine for system logging. Use, e.g., syslog-ng!

List of Figures

2.1	Positioning of plugin within the SMTP dialog	15
2.2	Prototype main functionality	17
3.1	Comparison of postal mail and spam mail	21
3.2	E-mail traffic 2004 – 2008	23
3.3	Number of worldwide internet users	23
3.4	Trends in mail productibility and deliverability	24
3.5	Cost categories for spammers	25
3.6	Bit length to elapsed time ratio	29
3.7	Possible profit per day and computer (spammer rents server)	30
3.8	Profit with MBound 15 bit	32
4.1	Detection rate of header tests	36
4.2	Fraction of rules triggered - spam-test set 08/04	36
4.3	Fraction of rules triggered - spam-test set 04/05	36

List of Tables

3.1	Influence of response rate and cost per e-mail on spammers' profit	22
3.2	Fixed cost per month for renting a server in Asia	26
3.3	Cost factors – single spammer as sales agent	27
3.4	Cost factors – single spammer as online marketer	27
3.5	Computation time for different partial collisions on different CPUs	29
3.6	Number of outgoing messages depending on CPU and collision size	29
3.7	Revenue depending on Hashcash size	31
3.8	Hashcash vs. MBound	32
3.9	Spammers' Profit per day, different cost models [29]	33
4.1	EPF results for a spam trap	37
4.2	Mail servers and their greylisting compatibility.	38

Bibliography

- [1] Gansterer, W., Ilger, M., Lechner, P., Neumayer, R., Strauß, J.: Anti-spam methods - state of the art. Technical report, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna (2005)
- [2] Postel, J.: RFC821: Simple Mail Transfer Protocol. Technical report, Internet Engineering Task Force (1982) <ftp://ftp.rfc-editor.org/in-notes/rfc821.txt>.
- [3] MXLogic: Spam classification techniques (2004) <http://www.mxlogic.com>.
- [4] Jung, J., Sit, E.: An empirical study of spam traffic and the use of dns black lists. Technical report, MIT Computer Science and Artificial Intelligent Laboratory (2004) <http://nms.lcs.mit.edu/papers/imc2004-spam.pdf>.
- [5] Harris, E.: The next step in the spam control war: Greylisting. Technical report, PureMagic Software (2003) <http://projects.puremagic.com/greylisting/whitepaper.html>.
- [6] Twining, R.D., Williamson, M.M., Mowbray, M., Rahmouni, M.: Email Prioritization: reducing delays on legitimate mail caused by junk mail. Technical report, HP Laboratories Bristol (2004) <http://www.hpl.hp.com/techreports/2004/HPL-2004-5.pdf>.
- [7] Lentczner, M., Wong, M.W.: Sender Policy Framework: Authorizing Use of Domains in MAIL FROM. Technical report, The Internet Society (2004) <http://www.ozonehouse.com/mark/spf/draft-lentczner-spf-00.html>.
- [8] Microsoft Corporation: Caller ID for E-Mail, The Next Step to Deterring Spam. Technical report, Microsoft Corporation (2004) <http://www.microsoft.com/downloads/details.aspx?FamilyID=9a9e8a28-3e85-4d07-9d0f-6daeabd3b71b&displaylang=en>.
- [9] Delany, M.: Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys). Technical report, Yahoo (2005) <http://www.ietf.org/internet-drafts/draft-delany-domainkeys-base-02.txt>.
- [10] Donnerhacke, L.: Teergruben FAQ (2005) <http://www.iks-jena.de/mitarb/lutz/usenet/teergrube.html>.
- [11] Lechner, P.: Das Simple Mail Transfer Protokoll und die Spamproblematik. Master's thesis, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna (2005)

- [12] Klensin, J.: RFC2821: Simple Mail Transfer Protocol. Technical report, Internet Engineering Task Force (2001) <ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt>.
- [13] Schweikert, D.: Postgrey - postfix greylisting policy server. Internet, Open Source (2004) <http://isg.ee.ethz.ch/tools/postgrey/>.
- [14] Braden, R.: RFC1123: Requirements for Internet Hosts - Application and Support. Technical report, Internet Engineering Task Force (1989) <ftp://ftp.rfc-editor.org/in-notes/rfc1123.txt>.
- [15] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., Jones, L.: RFC1123: SOCKS Protocol Version 5. Technical report, Internet Engineering Task Force (1996) <ftp://ftp.rfc-editor.org/in-notes/rfc1123.txt>.
- [16] R. Khare, S.L.: RFC2817: Upgrading to TLS Within HTTP/1.1. Technical report, Internet Engineering Task Force (2000) <ftp://ftp.rfc-editor.org/in-notes/rfc2817.txt>.
- [17] Boneh, D.: The difficulties of tracing spam email. Technical report, Department of Computer Science, Stanford University (2004) http://www.ftc.gov/reports/rewardsys/expertrpt_boneh.pdf.
- [18] Venema, W.: Postfix mail transfer agent. Internet, Open Source (2005) <http://www.postfix.org>.
- [19] Tokarev, M.: proxycheck: Open proxy checker (2004) <http://www.corpit.ru/mjt/proxycheck.html>.
- [20] Mack, D.: Grinch open relay checker (2004) <http://www.zonque.org/projects/grinch/>.
- [21] Wall, L.: Perl, the practical extraction and report language. Internet, Open Source (1987) <http://www.perl.org>.
- [22] Taughannock Networks: An overview of e-postage (2003) <http://www.taugh.com/epostage.pdf>.
- [23] Loder, T., van Alstyne, M.W., Wash, R.: An economic answer to unsolicited communication. In: ACM Conference on Electronic Commerce. (2004) 40–50 <http://doi.acm.org/10.1145/988780>.
- [24] Sunder, S., Cronin, M.A., Kraut, R.E., Morris, J., Telang, R.: Markets for attention: Will postage for email help? Technical Report ysm394, Yale School of Management (2003) <http://ideas.repec.org/p/ysm/somwrk/ysm394.html>.
- [25] Cobb, S.: The economics of spam (2003) <http://www.cobb.com/spam/index.html>.
- [26] Geller, T.: The true cost of spam (2005) <http://www.website101.com/SpamFilter/spam-costs.html>.

- [27] Graham, P.: Stopping spam (2003) <http://www.paulgraham.com/stopspam.html>.
- [28] DoubleClick, Inc.: Doubleclick q4 2004 email trend report (2005)
- [29] Strauß, J.: Analyse technischer Lösungen zur Spamvermeidung und ihrer betriebswirtschaftlichen Implikationen. Master's thesis, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna (2005)
- [30] The Radicati Group, Inc.: Radicati market numbers summary update, Q4 2004 (2005) <http://www.radicati.com>.
- [31] TNS Infratest Business Intelligence: 7. Faktenbericht - Monitoring Informationswirtschaft (2004) <http://hhi.corecom.com/judge-antispam.pdf>.
- [32] Postini Inc.: Email security annual review & threat report 2005. Technical report, Postini Inc. (2005) <http://www.postini.com/whitepapers/?WPID=25>.
- [33] Internet Systems Consortium: Internet domain survey, jul 2004 (2004) <http://www.isc.org/index.pl?ops/ds/>.
- [34] Portmann, C.: Werbebriefe, die nicht im Papierkorb landen. KMU-Magazin (2004) <http://www.scoremarketing.ch/WissensBox/pdfs/lesenswert/werbebriefe-kmu0404.pdf>.
- [35] McCloskey, B.: Live from ad:tech. Email-Insider (2004) http://www.mediapost.com/dt1s_dsp_EmailInsider.cfm?fnl=041110.
- [36] Judge, P.Q.: Fighting spam: Anti-spam technologies (2003) <http://hhi.corecom.com/judge-antispam.pdf>.
- [37] Westley, C.: The economics of spam. The Freeman (2003) <http://www.fee.org/vnews.php?nid=5662>.
- [38] netz98 new media GmbH: E-mail marketing system (2005) <http://www.mailenstein.de>.
- [39] Statistik Austria: IKT - Einsatz in Haushalten. Technical report, Statistik Austria (2004) <ftp://www.statistik.at/pub/neuerscheinungen/2005/ikt2004.pdf>.
- [40] Bulk Email Superstore: Online marketing system (2005) <http://www.americaint.com/>.
- [41] Hitt, J.: Confessions of a spam king. New York Times (2003) <http://www.gwtools.com/gwguardian/prodlit/Confessions%20of%20a%20Spam%20King.pdf>.
- [42] Zenger, R.: Confession for two: a spammer spills it all (2004) <http://rejo.zenger.nl/abuse/1085493870.php>.
- [43] Goodman, J.T., Rounthwaite, R.: Stopping outgoing spam. In: ACM Conference on Electronic Commerce. (2004) 30–39

- [44] Turner, D., Ross, K.: The lightweight currency protocol (2003) <http://cis.poly.edu/~ross/papers/draft-turner-lcp-00.txt>.
- [45] Federal Information Processing Standards Publication: Secure hash standard (1995) <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- [46] Back, A.: Hashcash - a denial of service counter-measure (2002) <http://www.hashcash.org/papers/hashcash.pdf>.
- [47] Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. In: Network and Distributed System Security Symposium Conference Proceedings: 2003. (2003) <http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/2.pdf>.
- [48] Rosenthal, D.: On the cost distribution of a memory bound function. In: LOCKSS TR2003-02. (2003)
- [49] Dwork, C., Goldberg, A., Naor, M.: On memory-bound functions for fighting spam. In: Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003). (2004)
- [50] Apache Software Foundation: Spamassassin open-source spam filter. Internet, Open Source (2004) <http://spamassassin.apache.org/>.