Comparing Anti Spam Methods

Anders Wiehes

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2005

Institutt for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Most Internet users use mail to communicate electronically. They depend on the mail system to deliver their mails to the recipient. Spam has made the mail system more unreliable because mail can get falsely caught by spam filters on the way to the recipient or mail can drown among spam in the recipients inbox.

The goal of the Internet community should be to work toward a more usable Internet with less spam. Possible ways to do this are through the law and the legal system, technical solutions and user awareness. This thesis looks at and compares different technical methods used to detect spam when it is sent to the receiving Internet service provider or when it has entered the receiving user's inbox.

An experiment has been conducted to gain new knowledge about how the anti spam methods compares with each other on a real system handling over 300,000 emails in a month. The results of the experiment are presented and solutions for anti spam systems is also discussed in this thesis.

Sammendrag

Mange Internett-brukere bruker epost til å kommunisere. De er avhengige av at epostene de sender når mottakeren. Spam har gjort at epostsystemet har blitt mer upålitelig på grunn av at epost kan feilaktig bli stoppet i spam-filtere eller at de kan drukne i spam i mottakers innboks.

Internett-samfunnet bør ha som mål å jobbe mot et Internett med mindre spam. Dette kan gjøres via lover og rettssystemet, tekniske løsninger og brukeropplysning. Denne masteravhandlingen ser på og sammenligner forskjellige tekniske løsninger for å detektere spam når den blir sendt til mottakers Internett-tilbyder eller når den har havnet i mottakers innboks.

Det har blitt utført et eksperiment for å utvikle ny viten om hvordan forskjellige antispammetoder gjør det i forhold til hverandre på et produksjonssystem som behandler over 300,000 eposter på en måned. I denne rapporten blir resultatene av eksperimentet presentert og forslag til antispaml sninger blir også diskutert.

Acknowledgements

I would like to thank the people that contributed to this study. This study would not be possible without your help. I would especially like to thank the employees at Gjøvik University College who helped and supported this study. Erik Hjelmås was my guidance professor and he helped with the report and gave hints and ideas for this study. The Information Technology department trusted me to configure and run the experiment email server handling incoming email for one month. Jonny Birkelund, Stian Husemoen and Jørgen Hanssen at the Information Technology department were easy working and cooperating with. Also thanks to employees and students who gave user feedback on the email they received.

I would also like to thank the open source community for software used in the experiment. My source code modifications, experiment configuration and documentation will be made freely available so others can use and improve it. I hope the community will benefit from this.

My very good friends Fredrik Skarderud, Ole Kasper Olsen, Ole Martin Dahl and Torkjel Søndrol also deserve a big thank you for fun discussions and support.

Contents

1 Introduction

A high percentage of email users knows what spam, trash or junk mail is, email you do not want and do not need. Spam has grown just as fast as the Internet to become a world wide plague. The purpose of spam can be to make money of a product, create unnecessary traffic or bring down email servers. Whatever purpose, it's usually the average email user that looses resources.

1.1 Definition of Spam

Spam is an expression from a Monty Python sketch [24]. It has become a word for junk or unsolicited email. The purpose of spam can be to make money of a product, get personal information from users or spread viruses.

Junk mail was already an issue in 1975 when Joe Postel wrote a Request for Comments on the junk mail problem [30]. The problem has been growing since then and now every email user knows what spam or junk mail is.

1.2 Need of This Study

Spam steals valuable time for people since email users and system- and network administrators need to spend time handling spam. There are also direct expenses like bandwidth and storage space. And possibly the most important problem is that genuine information can be lost because of a false positive in a spam filter or because it ”drowned” in spam in a users mailbox.

Atkins [2] estimates that every US email user has direct expenses of \$30-50 per year, every employee costs \$730 in lost productivity per year and that the total cost to US corporations is \$8,900,000,000 because of spam. He also estimates that a dial-up Internet Service Provider (ISP) has a cost of \$2,000 to \$10,000 for dealing with a single spamming customer.

It's clear that companies and individuals using email could save huge amounts of time and money if they could avoid spam. ISPs that injects spam into the Internet can also save huge amounts of money if the problem is reduced.

1.3 Purpose of This Study

The purpose of this study is to produce new knowledge about how to reduce spam for email servers and email users. Results contributing to this knowledge should be helpful to email server administrators and users who see spam as a problem. Both when email get falsely caught in spam filters and when spam floods the inbox. The results should also contribute to people having more trust in the email system.

1.4 Statement of The Problem

State of the art technical anti spam features include spam filters based on email content, like most spam filters built into popular email clients, and other countermeasures like checking if the sender complies to Internet standards for sending mail.

It is interesting to see how the various anti spam methods compare to each other. Methods can mark different or equal email as spam. They can also have different false positive and false negative ratios.

1.5 Research Questions

The following needs to be researched in order to find answers to the statement of the problem:

- Where and how does the Internet mail system allow filtering of spam?
- Which spam filters and spam countermeasures are the most effective?
- To what extent is it possible to reduce spam in a secure way without the risk of loosing information?

1.6 Delimitations

This thesis will focus on technical aspects of the spam problem in email. Legislation and other non-technical issues will not be a big part of this thesis.

It will also focus on solutions that do not alter existing email addresses or impose extra work on the sender, like acknowledging that the email is sent by a human. Standard compliance, open source solutions and automatic solutions requiring little or no user interaction will be favored.

1.7 Method

A mixed research approach [7] is needed in order to solve the research problem. This approach combines qualitative and quantitative research methods. Literature study, a qualitative method, and experiment, a quantitative method, will be used when trying to find answers to the research questions.

1.8 Outline of The Chapters

Chapter 2 gives an overview of the how the email system on the Internet works. It also explains different technical possibilities for detecting, filtering and removing spam in the on the Internet. Expressions like false accept and false reject are also covered. Chapter 3 gives a review of the relevant literature and scientific work related this thesis. Chapter 4 explains the experiment design and results. Answers to the research questions found in the literature and the experiment results are discussed in Chapter 5. Suggestions to anti spam solutions is in Chapter 6. Chapter 7 gives a conclusion of the study and Chapter 8 suggests future work. Appendix A describes how to get documentation and software used in the experiment.

2 Theory

This chapter will cover the theory for this thesis. It will explain the terminology needed for understanding the topics covered. It will also give an overview of different parts in an email and how an email is sent through the Internet. Different possibilities related to this thesis for detecting spam between the sender and the recipient of an email is also covered.

2.1 Definition of Spam and Ham

Spam originates from a Monty Python sketch [24] and is commonly used when referring to junk or unsolicited email. This can include email containing virus, chain letters, advertisement, political advocacy or fraud attempts. Ham is used when referring genuine email, the opposite of spam.

2.2 The Internet Mail System

Request for Comments (RFC) 2821 [19] describes the Simple Mail Transfer Protocol (SMTP) which is used when sending mail on the Internet. This protocol has been in use for several years. The first simple mail transfer protocol RFC appeared in 1981. SMTP RFCs which supersede the first one are backward compatible, new functionality are added with extensions to the protocol.

2.2.1 Email - Header and body

An email consist of a header and a body. The sender of the email will make the content for both the header and the body. This is usually done within a Mail User Agent (MUA). The following is an email with header and body separated with the first blank line:

```
From: Joe <joe@example.com>
To: Jane <jane@example.net>
Cc: Bill <bill@example.org>
Bcc: George <george@example.org>
Date: Sat, 01 Jan 2005 12:00:00 +0100 (CEST)
Subject: Example email

Hello!

This is an example email with a header and a body.
```

Headers are defined in RFC 2822 [32]. The *date* header specifies the date and time when the author completed and sent the message. *From* and *reply-to* specifies the author's address and optionally the address where replies to the message should be sent. *To*, carbon copy (*Cc*) and blind carbon copy (*Bcc*) specifies the message recipients. *To* and *Cc* fields are shown in the message the recipients receive while the *Bcc* field is not shown. *Message-ID*, *In-Reply-To* and *References* specify this email's ID and optionally the IDs of the email replied to if this email is an answer to a previously sent email.

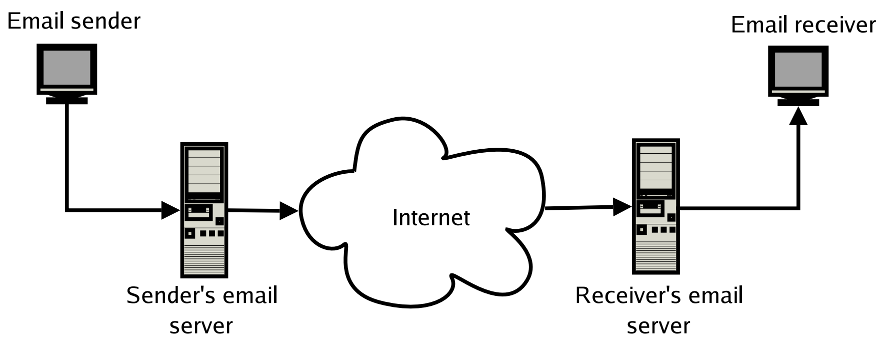

Figure 1: The route of an email from the sender, through the Internet to a receiver

2.2.2 Sending an email

Once the sender has finished making the header and the body of an email it will connect to an SMTP server, usually an SMTP server hosted at the sender's ISP, and send the email. This can be compared to delivering a letter to a postman. The email can then travel between many SMTP servers, like the letter can travel between many other postmen, before it is put in the receiver's mailbox. Figure 1 shows an example of how an email travels through the Internet.

The last SMTP server on the sender side will look up the email server at the receiver side. This is done with an address lookup in the Domain Name System (DNS) for the receiver domain. The lookup will return an IP address which the sender SMTP server connects to. The communication with a Sendmail server, sending the example email from 2.2.1, is shown below. Lines beginning with S after the line number are those sent by the sending end and lines beginning with R after the line number are those replied by the receiving end.

```
01 R: 220 smtp.example.net ESMTP Sendmail 8.12.11/8.12.11; Sat, 01 Jan 2005 12:00:00 +0100
02 S: EHLO smtp.example.com
03 R: 250-smtp.example.net Hello smtp.example.com [1.2.3.4], pleased to meet you
04 R: 250-ENHANCEDSTATUSCODES
05 R: 250-PIPELINING
06 R: 250-8BITMIME
07 R: 250-SIZE
08 R: 250-DSN
09 R: 250-ETRN
10 R: 250-AUTH DIGEST-MD5 CRAM-MD5
11 R: 250-DELIVERBY
12 R: 250 HELP
13 S: MAIL FROM: <joe@example.com>
14 R: 250 2.1.0 <joe@example.com>... Sender ok
15 S: RCPT TO: <jane@example.net>
16 R: 250 2.1.5 <jane@example.net>... Recipient ok
17 S: DATA
18 R: 354 Enter mail, end with "." on a line by itself
19 S: From: Joe <joe@example.com>
```

```
20 S: To: Jane <jane@example.net>
21 S: Cc: Bill <bill@example.org>
22 S: Date: Sat, 01 Jan 2005 12:00:00 +0100 (CEST)
23 S: Subject: Example email
24 S:
25 S: Hello!
26 S:
27 S: This is an example email with a header and a body.
28 S: .
29 R: 250 2.0.0 <queue ID> Message accepted for delivery
30 S: QUIT
31 R: 221 2.0.0 smtp.example.net closing connection
```

The addresses in the header of the email does not have to be the same as those specified as the envelope addresses. The envelope address and the header address can be compared with a paper envelope with a paper letter inside where one can easily write different addresses. The postman will deliver the envelope to the address written on the envelope like the email will be delivered to the envelope address. The addresses shown in MAIL FROM and RCPT TO at lines 13 and 15 are envelope addresses and specify the envelope address sender and receiver. The SMTP standard has not defined authentication of the sender, therefor email addresses can easily be faked. Section 2.3 describes some suggestions for new standards that adds email sender authentication functionality.

For every standard compliance server an email passes through, a *received* header is added. This header usually describes which email address the email has been received for and which IP address the email is coming from. This makes it possible to track the route of an email. Faked received headers can also be inserted into the email causing difficulties when trying to track it's origin.

Spammers want to hide their identity and origin of their spam [23] and send as many email messages as possible. This can be accomplished by using insecure SMTP servers, proxies or cracked machines connected to the Internet which will forward their spam [29]. As shown, there are not many obstacles in the SMTP standard.

2.2.3 Different Approaches to Spam Filtering

Figure 1 shows the route of an email from sender to receiver. There are many ways to eliminate or reduce spam [28]. One technical way to stop spam is to deny outgoing email sessions from the receiver network and make every host within that network use a controlled, administered and secure email server when sending email. This will also help stop abuse from outside when a third party uses a network's resources to relay email.

Spam can also be detected and filtered at the recipient's end. It can be stopped before it enters the recipient's system or it can be filtered after it has been accepted and entered the recipient's system. Filtering before it has entered the recipient's system is done by issuing error codes in the email delivery session shown in subsection 2.2.2. Depending on whether it is a permanent or temporary error message, a genuine and standard compliant sender system will try to deliver the message after a delay if it is a temporary error message. Failure to deliver an email to the recipient will often result in a warning to the sender.

When email is accepted by the recipient's system, it is not common to notify the sender if the email is caught by a spam filter. Therefor, the risk of losing information, when using a filter after the email is accepted, is higher. Technical anti spam methods like greylisting is also impossible at this stage.

2.3 Anti Spam Methods

There are a variety of anti spam methods. Some are based on email content and others are based on protocol and other technicalities. Centralized and decentralized anti spam methods are also covered. Every method has their strengths and weaknesses. This section will describe methods that fits the delimitations from section 1.6 of this thesis.

2.3.1 Greylisting

Greylisting [16] is designed to have minimal impact on users and require minimal maintenance. It have to be running on all the email servers published by a domain to be effective. Three pieces of information are used by greylisting.

- The IP address of the host attempting the delivery
- The envelope sender address
- The envelope recipient address

These items combined are called triplets. Greylisting operation can be summarized with a simple rule: *If we have never seen this triplet before, then refuse this delivery and any others that may come within a certain period of time with a temporary failure.* Since the SMTP protocol has built in support for temporary failures, genuine SMTP servers will try to deliver undelivered email until it is delivered. Greylisting will then eventually accept the email since the email's triplet is known.

Greylisting stops spam from spammers using ”dumb” software when spamming. ”Dumb” software is software that tries to deliver email once or more times within a limited time period and doesn't look at the SMTP protocol responses to see if the delivery attempt is accepted. For a spammer to circumvent greylisting, the cost of spamming will rise. More resources is needed and standard compliance software must be used.

In a personal correspondence with the email administrators at universities in Norway and Uninett, the Norwegian research network, we got valuable information about greylisting from those who used it actively. For example experiences with different greylist implementations and a list of networks that should be white-listed. This information is used in the experiment design phase.

2.3.2 Sender Policy Framework

Sender Policy Framework (SPF) [37] is a new system not yet standardized in the form of a RFC. But major email providers like Gmail and Hotmail has started to adapt it. SPF aims to prevent worms, viruses and spam from forging arbitrary email addresses as the envelope sender in SMTP.

Two things must be done in order for SPF to be effective. The first is that email administrators must publish the IP addresses that are allowed to send email with an envelope from address in their domains. This is done with the Domain Name System (DNS) which is best known for translating host names to IP addresses on the Internet.

DNS is a distributed database well suited to publish small amounts of information related to domains and IP addresses. The second thing that must be done in order for SPF to be effective is checking incoming mail. Email servers or email clients must also check incoming email to see if the envelope from address is an allowed address according to the domain's SPF records. If the sender's IP address doesn't match the ones in the SPF record, the mail can be considered forged and rejected or put in a spam folder.

Even if every domain has published SPF records and every incoming email server on the Internet checks incoming email against the envelope from address, spam is still an issue. SPF just makes faking the envelope mail from address impossible. To circumvent SPF, spammers could for example buy a new domain, publish their own SPF records for that domain and start spamming with an address in the newly bought domain.

2.3.3 DomainKeys

DomainKeys [38] is a new system not yet standardized in the form of a RFC, but major email providers like Yahoo has started to adapt it. DomainKeys aims to prevent worms, viruses and spam from forging arbitrary sender email addresses. It also aims to detect if some of the other headers and the body are modified after an email has left the sender system.

Each email is signed by the sending system when using DomainKeys. This requires the administrator of the sender system to generate a private and public key pair. The private key is made available to the sender server so it can sign each outgoing message. The public key is made available via DNS records in the sender domain so each recipient can retrieve it easily and use it to check the signature. A policy for a domain using DomainKeys must also be made available via DNS. It says to what extent a domain is using DomainKeys. If a recipient receives an email which is not signed and the policy in the sender domain is to sign all email, the email can be rejected.

The signature is appended as a new header in the email when it is sent. The new header contains, among other things, which other fields are signed, algorithms used to sign and the signature itself. Header fields which can be changed or added at a later time like the received headers are not signed. This means that email to mailinglists will also be valid after it has been redistributed by the list as long as the original headers and body are not changed. The following shows the header added to an email when using DomainKeys:

```
DomainKey-Signature: a=rsa-sha1; q=dns; c=nofws;
        s=beta; d=gmail.com;
        h=received:message-id:date:from:reply-to:to:subject:
          mime-version:content-type:c+ontent-transfer-encoding;
        b=b1FKPFeCr0M034uurDUwQwIgbkgd9dEpAnKblRw1mxfq/l2grBtmpn
          Y50Ue7UW8I0u5Kep31SbWrI+z1z8LgU7YZQyxpf54fY6TSWQA1CJrk
          h58sTATcdQeY+0XGZLhEee04RwtQwY5rV68KHnC0otz59zAZo+u8Tg
          JiBUqD2+v4k=
```

DomainKeys will stop fake sender addresses and manipulation of an email after the user has sent it. If it is widely deployed, phising and fraud or viruses using fake sender addresses can be reduced or eliminated. For other types of spam, the spammer can register new domains and use them as the sender domain.

2.3.4 Real Time Black Lists

Real Time Black Lists (RBL) [29] are often distributed via DNS and contains IP addresses of the hosts that are thought to be insecure or send spam. Like the name says, the lists are updating in real time meaning that when an IP address is added, users of the black list are able to benefit from this almost instantly. Each host sending email is checked by the recipient via DNS to see if the host sending the email is black listed. If the host is black listed, the email can be considered spam.

Each black lists can return different values to a query. Values can for example describe how sure the black list is or the reason the IP address is added to the black list. Black lists also has different rules for when an IP address is added. Rules can be open proxies that can be abused to send email, open email servers that everyone can relay via or that users have reported spam messages coming from that IP address.

DNS real time black lists operates at the network level. This means that if an email server is black listed because it can be abused or some of the users using it has abused it, the rest of the users are also black listed. This will in most cases make the administrator prioritize fixing the email server configuration, but it also means that black lists can have a high false positive ratio.

2.3.5 Razor

Razor [31] uses a distributed hash database and identifies spam by email content. A method which is related to Razor is described in [10] by Deepak and Parameswaran. With Razor, a hash is generated from the content and checked against dedicated Razor servers on the Internet. The Razor servers are updated by users who report the hash of spam they receive.

Since hash algorithms generate a totally different hash value when only one character in the input is changed, they do not work against spam that insert random string in the spam. Razor therefor uses a fuzzy signature matching algorithm [9] called Nilsimsa. Three criteria must be met for a fuzzy hash algorithm used in this setting:

- The digest identifying each message should not vary significantly for changes that can be produced automatically.
- The encoding must be robust against intentional attacks.
- The encoding should support an extremely low risk of false positives.

This algorithm creates a statistical model of each message. The model is based on message body minus any slight text mutations it finds. Razor can also only hash for example the seven first lines or the last five lines of an email to cope with big chunks of random data.

Razor can also return false positive spam status. It depends on how good the algorithms for detecting similar email content and it depends on correct user feedback of spam email. For Razor to be able to return as low false negative spam status as possible, it depends on users giving feedback when they receive spam.

2.3.6 Distributed Checksum Clearinghouse

Distributed Checksum Clearinghouse (DCC) [34] uses the same fuzzy hash algorithms as Razor described in subsection 2.3.5. Unlike Razor it does not depend on user feedback. Every email server or email client that is DCC enabled reports the message it receives to

DCC servers. The DCC servers then count how many similar email message that has been reported and sends this number back to the email server or client when asked. When a high number is sent back to the email server receiving an email, it is a bulk email and can be considered spam.

The only spam indicator this system has is the amount of email servers or clients that has seen an email. Ham that is sent to a large number of users, like a newsletter, will be marked as spam. White-listing or include other systems is therefor a necessity when using this system to avoid a high rate of false positives.

2.3.7 Spamassassin

Spamassassin [29] is a ruled based spam filter. It has an extensive set of built in rules to identify spam. Each rule is checked against an email to determine whether it is spam. When a rule matches the email, a number of points determined by the rule are added to a total sum for the email. When every ruled is checked the sum is compared to a user or default defined threshold. A number higher or equal than the threshold means that the message is spam and a number lower than the threshold means that the message is ham.

The rules and the score assigned to each rule defined in Spamassassin are static within a Spamassassin version. An example rule from Spamassassin version 3 is if the content offers an alert about a stock, this would add 2.362 points to the total sum. If an email also contains the word ”free” in the from address, this would add 0.194 to the total giving a sum of 2.556 points. Spamassassin also has the possibility, if enabled, to use rules that takes input from other methods like Razor described in subsection 2.3.5. For the rest of this thesis, Spamassassin will only be used with the score based rules, the thing that is unique for Spamassassin.

The user can configure the Spamassassin threshold and by that reduce or increase the false positive ratio Spamassassin will produce. But reducing false positives will also increase false negatives.

2.3.8 Other Methods

There are other popular anti spam methods that require more user feedback than the previously described. They usually require a learning period, where the user reports spam and ham to the system, before they can be effective.

Paul Graham [14] was the first to describe Bayesian spam filtering. Bayesian filters calculate the statistical probability of email being spam based on previous analyzes of spam messages, done by the user training the filter. These probabilities and frequencies are then collaborated into rules that are applied to incoming email to detect if it is spam. An improved method is to use Bayesian Noise Reduction (BNR), a data sanitation technique, on the emails before they are fed to the Bayesian spam filter. This will clean the email of mutations, random data and noise-based words. Using BNR combined with a Bayesian filter will improve the accuracy of the filter [15]. DSPAM is an example of a software implementation of this.

2.4 False Positives and False Negatives

There are four scenarios for an outcome when a spam filter or another countermeasure operates on an email:

- The email is ham and the filter correctly identifies the email as a genuine mail.
- The email is spam and the filter correctly identifies it as such.
- The email is not spam and the filter wrongly identifies the email as spam. This is called a false positive.
- The email is spam and the filter wrongly identifies the email as a genuine mail. This is called a false negative.

The two first items are the ones that we want to make happen. The two last items are the outcomes we do not want. To measure these we define false positive (FPR) and false negative ratios (FNR) as follows:

$$\mathsf{FPR} = \frac{\mathit{Emails\ wrongly\ identified\ as\ spam}}{\mathit{Total\ emails}}$$

$$\mathsf{FNR} = \frac{\mathit{Emails\ wrongly\ identified\ as\ ham}}{\mathit{Total\ emails}}$$

Measuring FNR and FPR can be difficult in a real life situation [21] and there is trade-off between false positives and false negatives. A lower false negative will in most cases give a higher false positive. A spam filter should have as low false positive ratio as possible. This is because some spams getting through the spam filters are better than loosing genuine emails.

3 Literature Review

This chapter will give an overview of the literature which is relevant to this thesis. The reason for spam, problems and various anti spam methods with tests will be the main parts of this chapter. Content based methods are the most covered and will therefor be the biggest part of the testing section.

3.1 Spam Existence and Problems

In [6], an overview of the spam problem is presented. Sending hundred thousand email messages can cost under $200 and obtaining a million email addresses can cost under $100. With such low expenses, spammers can recoup their costs even if only a tiny fraction of the email messages they send out result in sales. Spam can also be used by people to cheaply spread the word about religion, recruit members for supremacist organizations or explain their latest theories about physics and happiness [17].

Spam and the social-technical gap is covered in [36]. It is claimed that the email system lacks the properties we have outside the technical world and the article proposes bridging the gap between society and technology by applying social concepts to technology design. Spam illustrates what happens when technology ignores fair communication. These ideas are probably good, but this thesis will focus on improving existing technology and systems.

Spam places a considerable burden on ISPs systems, thus cost the ISP and the customers a considerable amount of money. The cost of receiving a single piece of bulk email is minimal, but the cost of receiving many messages can be considerable. Laws that would restrict sending of spam has a growing support in the US, probably because of the resources users of email spend on dealing with spam. The conclusion of [6] is to pursue existing laws against spammers and study the impact of this action.

Getting an overview of the spam traffic may be necessary when deciding new methods which can be implemented and tested. Characterizing the spam traffic is done in [13]. According to the article, key workload aspects where spam traffic significantly deviates from traditional non-spam traffic are email arrival process, email sizes, number of recipients per email, popularity and temporal locality among recipients.

3.2 Testing of Technical Anti Spam Methods

Jung and Sit [18] did an experiment over time to see how SMTP connections and real time DNS black lists changed over a longer period of time. Table 1 shows SMTP connection attempts into MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) which came from black listed hosts. One column has data from December 2000 and one column has data from 2004. This experiment also records port scans on the SMTP port, thus may not be accurate.

3.3 Testing of Content Based Anti Spam Methods

Oda and White looks at application of the artificial immune system model to protect email users effectively from spam in [26]. An immune system's main goal is to distinguish

Black list	SMTP connection attempts	
	Dec 2000	Feb 2004
cbl.abuseat.org	0	1401
list.dsbl.org	5	7624
opm.blitzed.org	0	122
ipwhois.rfc-ignorant.org	25	2030
dnsbl.sorbs.net	3	8529
bl.spamcop.net	0	496
sbl.spamhaus.org	2	1123
Total unique hosts black listed	34	11521

Table 1: SMTP connection attempts which came from a black listed host.

Classifier	P(%)	R(%)	F
Centroid-based	0.79	0.88	0.83
Bayesian-based	0.73	0.89	0.80

Table 2: Results from centroid-based compared to Bayesian-based algorithms.

between self and potentially dangerous non-self elements. In a biological system, these non-self elements include bacteria and viruses. In a spam immune system, the goal is to distinguish legitimate messages from spam. A problem for a spam immune system is that self elements change over time unlike a biological immune system. For example if an English speaking person learns Japanese and begins communicating in that language, the system must adapt to avoid an auto-immune reaction. The resulting system [26] classifies the messages with similar accuracy to other spam filters, but uses fewer detectors to do so, making it an attractive solution for circumstances where processing time is at a premium. Correct classification was 99% of non-spam and 70% of spam with a default threshold. The article authors thinks that this system, like Bayesian systems, should be used with personal mail for one user and systems where Bayesian algorithms are in use but uses too much resources.

Soonthornphisaj, Chaikulseriwat and Tang-On researched the a centroid-based classification approach to anti-spam filtering in [35]. The following definitions are needed to understand their results:

$$P = \frac{no.\ of\ correctly\ predicted\ positive\ examples}{no.\ of\ predicted\ positive\ examples}$$

$$R = \frac{no.\ of\ correctly\ predicted\ positive\ examples}{no.\ of\ all\ positive\ examples}$$

$$F = \frac{2PR}{P + R}$$

The results comparing the centroid-based classification approach against the common Bayesian algorithm are shown in Table 2. As shown the centroid based algorithm got 83% correctness against the Bayesian's 80%.

Another new content based spam filtering approach is presented in [25] by O'Brien and Vogel. It compares the new content based, which is called Chi-squared, with a Bayesian related algorithm. The findings of the authors indicates that the Chi-sqaured approach gives better results than the Bayesian algorithm. Unfortunately, the sum of spam and ham that was used in the experiment was only 67, a too low number to con-

	RBLs	Phrase Matching	Heuristics (SA)	Statistics
Accuracy	0-60%	80%	95%	99%+
False Positives	10%	2%	0.5%	0.1%

Table 3: Results from measurements done by Sergeant

Package	Messages	% F. P.	% F. N.	% Errors
Sophos PureMessage (%50 filter)	1187	1.3	0.7	1.9
iHateSpam	1122	0.6	2.0	2.7
Eudora 6 (Windows)	8770	3.2	0.9	4.1
Eudora 6 (Macintosh)	2237	2.1	2.0	4.1
Outlook 2003	1456	2.2	2.2	4.4
Sophos PureMessage (%40 filter)	3618	2.6	2.2	4.8
Macintosh Mail 10.3	933	0.8	6.0	6.8
Macintosh Mail 10.2.6	1058	0.9	7.6	8.4
SpamBayes	1238	1.1	8.3	9.4
Norton AntiSpam	1024	0.4	9.9	10.3
SAproxy	118	0.0	12.3	12.3
MailFrontier (Matador)	421	22.3	0.5	22.8

Table 4: Results from the anti-spam software experiment conducted by Iowa State University.

clude that this method is better. The suggestions for future work also mention a larger test corpus.

Sergeant works in a email hosting company and has measured various anti spam methods in [33]. His results are presented in Figure 3.

A density-based spam detector is presented in [39]. It analyzes document space density to detect spam. Then it uses the same method as DCC described in section 2.3.6 to flag messages as spam, looking at how many times a message has been seen by email servers before. Characteristics of this method are:

- High processing speed: On a Pentium 4, 2.4Ghz the density-based method can handle over 13000 emails per second, thus it can be ran on a large email server.
- Maintenance free: Like for example the Bayesian methods, this method requires no manual database updates.
- Claimed 98% recall rate and 100% precision. This however requires white lists of all solicited email sources.
- Privacy protection: Since only hashes of the email are stored and the system is automatic, the privacy of the users are protected.

Balvanz, Paulsen and Struss [3] supervised testing and evaluation in order to recommend several desktop anti-spam software products for use at the Iowa State University (ISU). Volunteer testers used the product for a period of two weeks and recorded their results which is shown in Table 4. Since different volunteers has different email corpora, the results may not be comparable. Software packages that were chosen after evaluating the experiment results were Outlook, Eudora, Macintosh Mail and Sophos PureMessage.

Zhang, Zhu and Yao do a thorough testing of various statistical methods in [40].

Methods include naive bayes, maximum entropy model, memory based learning, support vector machine and adaboost. They use publicly available ham and spam corpora, both English and Chinese. One interesting finding of their work was that both header and body in an email is important when using statistical methods, previous work had only looked at the body. Using the whole email yielded comparable or better results than just using the body.

3.4 Summary

Many articles and papers suggests ways to do content based spam filtering [8] [5] [25] [10] [9] [26] [35] [27]. Many require a training period and maintenance by a user or administrator. Testing of content based filters is often done on a limited spam and ham corpus [22] [1] [40], no more than a couple of thousand emails. And the testing of non-content based methods requiring little or no maintenance are few [18]. There are not many experiments comparing many different anti spam methods on the same email corpus taken from a production system over a longer period of time.

4 Experiment

In order to find out how effective spam filters and countermeasures work we have done an experiment in a real life situation. Thus the results will be more valid than an experiment in a simulated environment. To detect if it is possible to reduce spam in a secure way, by using filters and countermeasures at the server level, the experiment ran on an email server.

Gjøvik University College [12] (GUC) allowed us to run an experiment on their incoming email server. This server handles email for about 1600 students and 200 employees. Every email coming from an external source to GUC will pass the experiment setup. There are no other spam filters or countermeasures between this email server and the Internet in the experiment period.

4.1 Design

This section will describe the design of the experiment and the technical implementation of the experiment. We have decided to run an experiment that is as much automated as possible because of time and resource limitations. Anti spam solutions like DSPAM requires user input over time and is therefor left out of the experiment. Suggestions for an experiment that involves the users more is described in Chapter 8. How to get a copy of the experiment documentation and configuration is describe in Appendix A.

4.1.1 Overview

To compare the various methods of reducing spam, the experiment data must be as equal as possible for the filters and countermeasures. This will produce more valid results. To accomplish this all methods are ran on all email that are received on the experiment server.

An exception to this is greylisting. Greylisting makes the design of the experiment more complex. To measure greylisting effectiveness, triplets that are not acceptable according to the database must be rejected in order to track success rate for greylisting. And at the same time the email that the host tried to deliver must be fed to the other filters and countermeasures to track success rate for the other methods. Thus a greylist rejection always happens after the whole email is transfered and waiting to be acknowledged, a change from the reference greylisting implementation. It also means that when an email is accepted by the greylisting for the first time or when it is rejected by the greylisting for the second or more times, it must not be fed to any other filter or countermeasure because the results have already been recorded for those filters and countermeasures.

This gives another problem when a user tries to send two emails to the same recipient from the same SMTP relay server. A solution can be to minimize the greylist block period in order to minimize the probability for this.

All emails in the testing period will be delivered to the users no matter what the filters think of their spam status. An exception to this is when greylisting denies an email, these emails will be delayed. A header will also be added to every email that is delivered to give the users the ability to report the spam status of the message they receive. Some

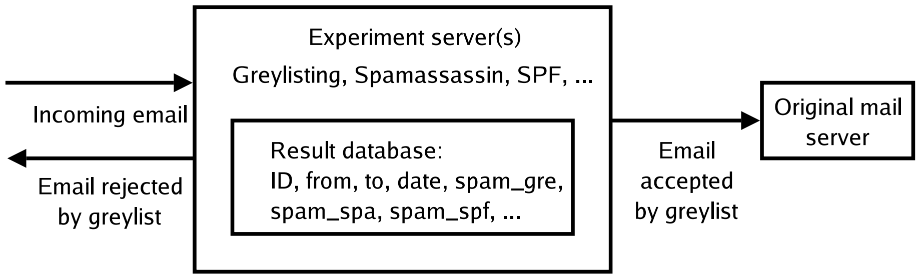
Experiment server(s)
Greylisting, Spamassassin, SPF, ...
Incoming email
Email rejected by greylist
Result database:
ID, from, to, date, spam_gre, spam_spa, spam_spf, ...
Email accepted by greylist
Original mail server

Figure 2: The route of an incoming email from a relay SMTP server, through the experiment setup and to a local mailbox.

selected employees and students will provide feedback. Figure 2 shows the experiment overview.

An unique ID must be assigned to every email once it enters the system. This ID and the current time and date will be stored in a result database. For the greylisting filter, the envelope sender and receiver address must also be stored when the email is rejected. This is because when the email enters the system again and is accepted it is identified as the rejected one and the greylist status of that email is changed from spam to ham. Further accepted emails with the same envelope sender and receiver will produce a new row and thus a new ID and greylist status of ham in the results database. The various filters which operate on the email must also store their findings, spam or ham, for every email. The database with the results will then be analyzed when the data harvesting is finished.

4.1.2 Technical Implementation

GUC's Information Technology department wanted an experiment design that separated the email delivery part to users as much as possible from the rest of the experiment. This reduces the chance of an interrupt in the email delivery to employees and students at GUC.

A two-server solution satisfies this demand. Greylisting needs to be placed in the real email path for it to work as intended. Thus accepting email from the Internet, run it through greylisting and, depending on the greylisting result, relay it to GUC's original incoming email server, are put on the first server. The greylist modification, identification and greylist status described in subsection 4.1.1 are also implemented on this server. Email that should be tested by the other anti spam methods, decided by the greylist modification, are copied, marked with it's unique ID in an added header and sent to a spamtest mailbox on the second server.

The second server will run the other anti spam methods in the experiment on email sent to the spamtest mailbox: Spamassassin, Razor, DCC, RBLs, SPF, DomainKeys and Bogofilter. The result of every anti spam method is then inserted in the result database in the row for the email's unique ID found in the header. The second server also runs the result database server. The two-server solution is visualized in Figure 3

Both the experiment servers run GNU/Linux as the operating system and the widely used Sendmail as the email server. The Sendmail running on the second server was configured to always queue incoming mail and run the queue when the server load was low. This is because running the anti spam methods other than greylisting is not important to run in real time. Thus the result database, which is needed by greylisting, has priority.

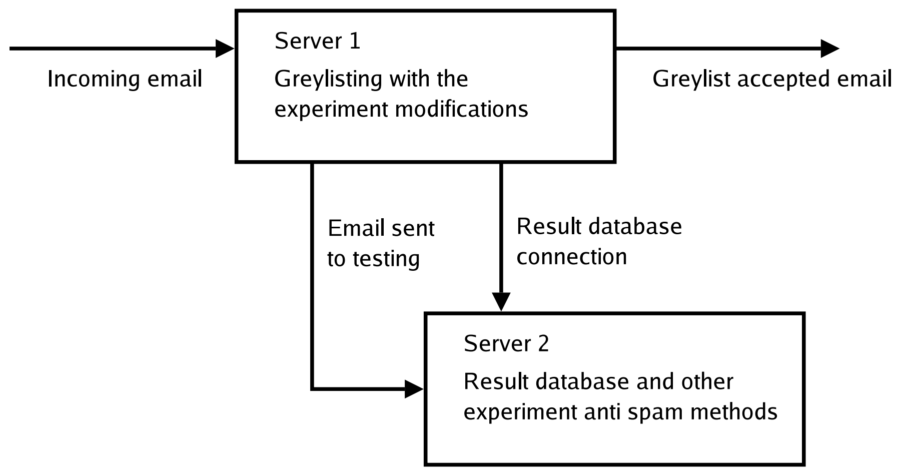
Incoming email
Server 1
Greylisting with the experiment modifications
Greylist accepted email
Email sent to testing
Result database connection
Server 2
Result database and other experiment anti spam methods

Figure 3: The two-server solution. The server named spam is critical for email delivery to users at GUC. The other server named ham does most of the experiment work.

The greylist reference implementation with modifications to fit this experiment runs on the first server.

PostgreSQL, an advanced Object-Relational database management system (DBMS), was chosen as the result database DBMS for its flexibility and scalability on the second server. The database model has the generic and greylisting columns described in subsection 4.1.1. It also has a column with a suitable data type for the return value of the other anti spam methods. By doing it this way the spam threshold can be tweaked for methods that return more than simply a boolean with spam or ham when the data is analyzed. The column name and possible values that can be analyzed in the result database are shown in Table 5.

Bogofilter uses a Bayesian algorithm to detect spam. It is therefor necessary to train it with both ham and spam before it can give a valid result. We used freely available ham and spam corpora in this experiment. This will not give a good result since each user using Bogofilter should train it with email in their own inbox. But it was the only way we could include it in the experiment because of time and resource limitations.

4.2 Results

This section describes the results from the experiment. These results are generated from a script, operating on the result database, which is included in the experiment documentation and configuration. See Appendix A for more information.

Columns in the database that affects privacy, relay_id, mail_from and rcpt_to, are removed from the database before the data is analyzed in accordance with the Norwegian Personal Information Law, Personopplysningsloven [20].

4.2.1 Overview

Some of the anti spam methods returns a true or false for spam or ham. Other, like Spamassassin, returns a score. For those methods that do not return true or false, the default configuration will be used when presenting the overview results. Table 6 shows an overview and Figure 4 visualizes the detected spam ratio of the various methods on the 342,181 messages that passed the experiment setup in the one month it was in

Column name	Possible values
id	unique ID
date	when email first seen
relay_ip	IP address email came from
status_human	spam, ham, dont know
status_greylist	spam, ham, white-listed
status_spamassassin	decimal number
status_rbl_spamcop	empty, in list
status_rbl_spamhaus	empty, in list
status_rbl_ordb	empty, in list
status_rbl_njabl	empty, in list
status_rbl_sorbs	empty, in list
status_rbl_dsbl_list	empty, in list
status_rbl_dsbl_multihop	empty, in list
status_rbl_dsbl_unconfirmed	empty, in list
status_spf	fail, pass, . . .
status_domainkeys	bad, good, . . .
status_razor	spam, ham
status_dcc	Message count
status_bogofilter	spam, ham

Table 5: Column names and possible values in the result database. The notation . . . indicates other possible values of spam or ham.

Anti spam method	Detected spam percentage
Razor	74.6%
Greylist	71.3%
All RBLs combined	69.5%
Spamassassin	66.8%
DCC	58.7%
Bogofilter	31.7%
SPF	3.1%
DomainKeys	0.1%

Table 6: Anti spam methods and their detected spam percentage in the experiment.

operation.

4.2.2 Anti Spam Methods Compared

Comparing the results of the various methods gives valuable information about how methods can be combined to give a more accurate detection rate. Bogofilter, SPF and DomainKeys all had such a small spam detection ratio that they are left out of the comparison. Look at Chapter 5 for a discussion of their results.

For the anti spam methods that marked a significant amount of spam, over 50%, they all agreed that 38.7% of percent of the messages are spam.

Comparing different methods with Spamassassin at different score thresholds are done in Figure 5. Here we see that the amount of messages all methods agreed are spam, are between 115825 and 139629 depending on the Spamassassin score threshold. This is between 33.8% and 40.8% of the messages. Also worth noticing, real time black lists and Spamassassin agrees on the most messages. This line doesn't lower itself as much as the others when the score is close to 10.

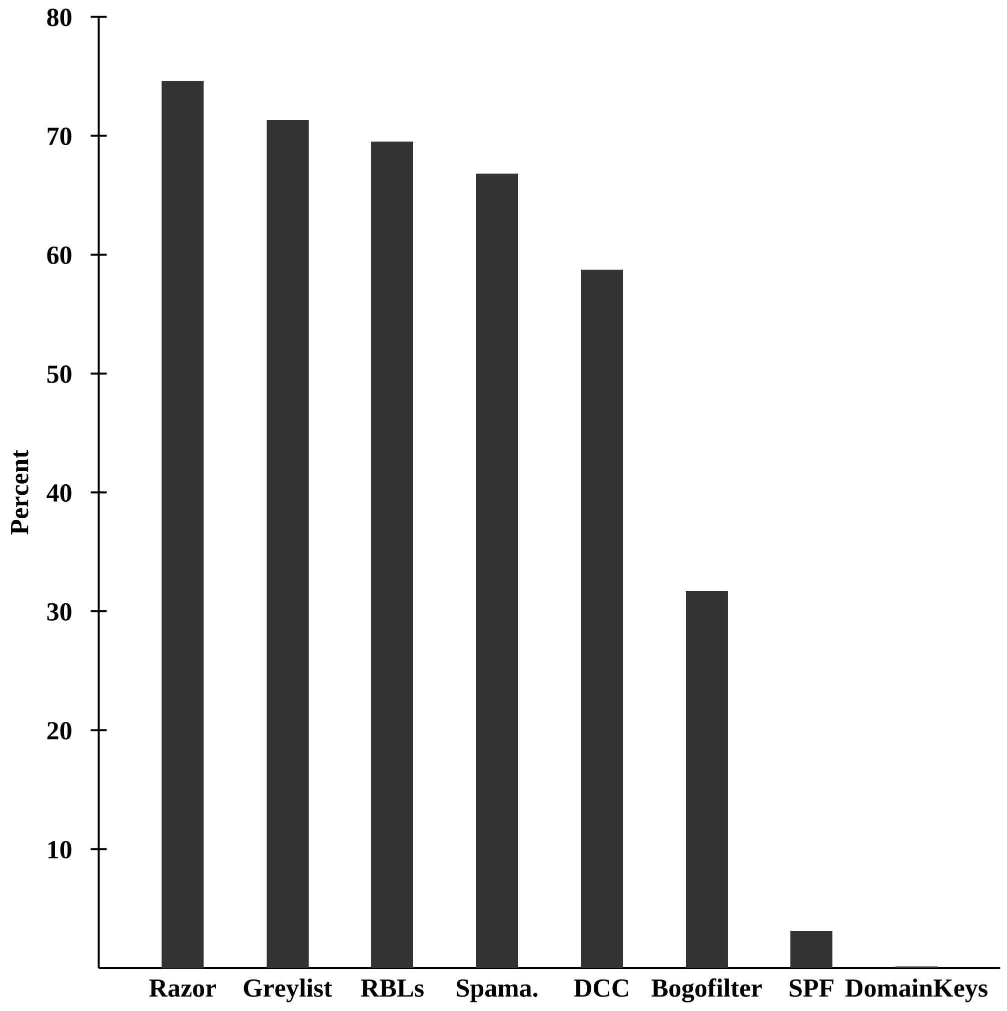

Figure 4: Anti spam methods and their detected spam percentage in the experiment.

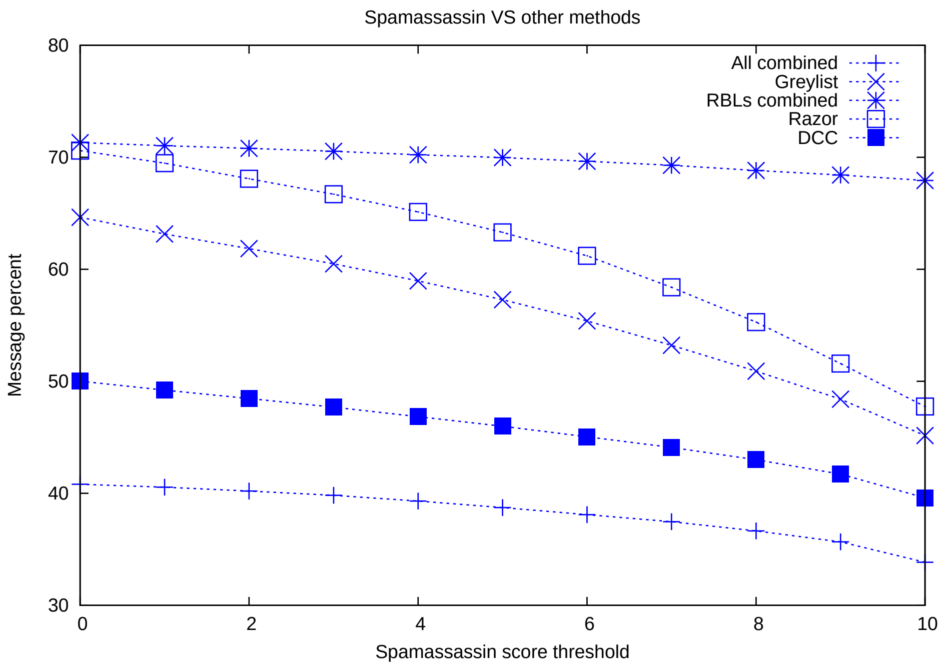

Figure 5: A graph showing how many messages Spamassassin and other anti spam methods agree on what is spam with different Spamassassin score thresholds.

Anti spam method	FNR	FPR
Razor	0.259	0.025
All RBLs combined	0.246	0.031
Spamassassin	0.573	0.015
DCC	0.553	0.288
Bogofilter	0.041	0.118
SPF	0.987	0.000
DomainKeys	1.000	0.000

Table 7: Anti spam methods compared with user feedback.

Figure 6 shows the amount of methods that marked messages as spam. The graph shows that requiring more methods before flagging a message as spam will mark fewer messages.

4.2.3 Anti Spam Methods Compared with User Feedback

User feedback on the messages are done after the messages are accepted by greylisting. 0.7% of the messages that can have user feedback has it. Table 7 shows and Figure 7 visualizes false positive and false negative ratios based on user feedback.

4.2.4 Anti Spam Methods Compared with Greylisting

Greylisting [16] claims to have zero percent false positives. This is true if all email servers where standard compliant. Figure 8 shows the amount of messages greylisting marked as spam compared with the messages that other methods marked as spam. Over two thirds of the messages were spam according to greylisting and Spamassassin, RBLs or Razor.

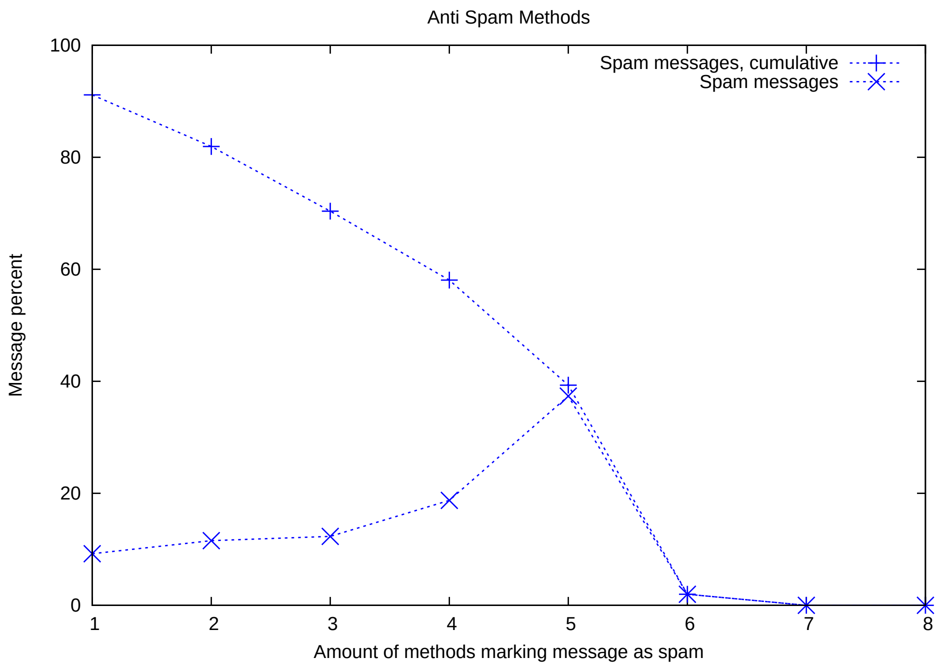

Figure 6: A graph showing the amount of methods that marked messages as spam.

4.2.5 Anti Spam Methods Compared with Razor

Razor [31] detects spam by user feedback. If the users using Razor never did any wrong when reporting spam messages and the algorithms for detecting similar messages where perfect, it would have zero false negatives. Figure 9 shows the amount of messages razor marked as spam compared with the messages that other methods marked as spam. RBLs, Spamassassin or greylisting and Razor marked around two thirds of the messages as spam. Spamassassin agrees on more messages with Razor than greylisting. This could be because spammers that are advanced enough to evade greylisting also create their spam in a way that make it harder for Spamassassin to detect.

4.2.6 Real Time Black Lists Compared

The experiment included 8 real time black lists. Figure 10 shows how many messages that where marked by different amounts of black lists. It also shows how many messages that would be marked as spam when requiring hits from a specific or higher number of black lists. This is also shown when greylisting agrees with the black lists.

Table 8 shows how many percent each real time black list marked as spam. A total of 340,841 messages was checked against the RBLs. Since the graph in Figure 10 shows a higher number of spams detected when requiring one method than the black list that marked the most messages in Table 8, the black lists that mark the most messages as spam do not have exactly the same records in their database.

4.3 Validity and Reliability

Validity is defined as ”the extent to which any measuring instrument measures what it is intended to measure” [4]. For a large email system that relays email more servers before it reaches it's recipient, the validity should be good. The validity of this experiment in

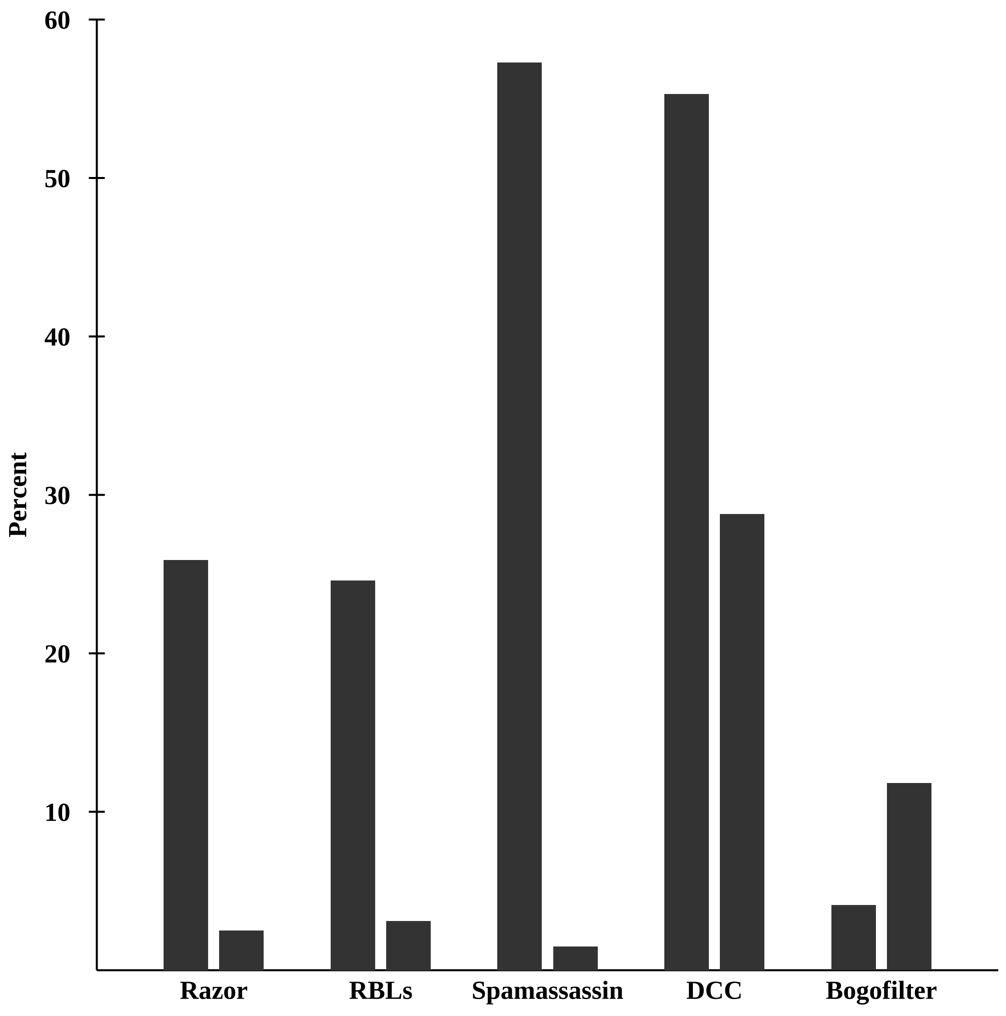

Figure 7: Anti spam methods compared with user feedback. False positive and false negative ratios for the various methods.

RBL	Spam messages	Spam percent
Spamcop	190103	55.8%
Sorbs	139971	41.1%
DSBL, unconfirmed	109849	32.2%
DSBL, list	108909	32.0%
Njabl	40049	11.8%
Spamhaus	16623	4.9%
DSBL, multihop	850	0.2%
Ordb	388	0.1%

Table 8: Spam percentage detected by the real time black lists in the experiment.

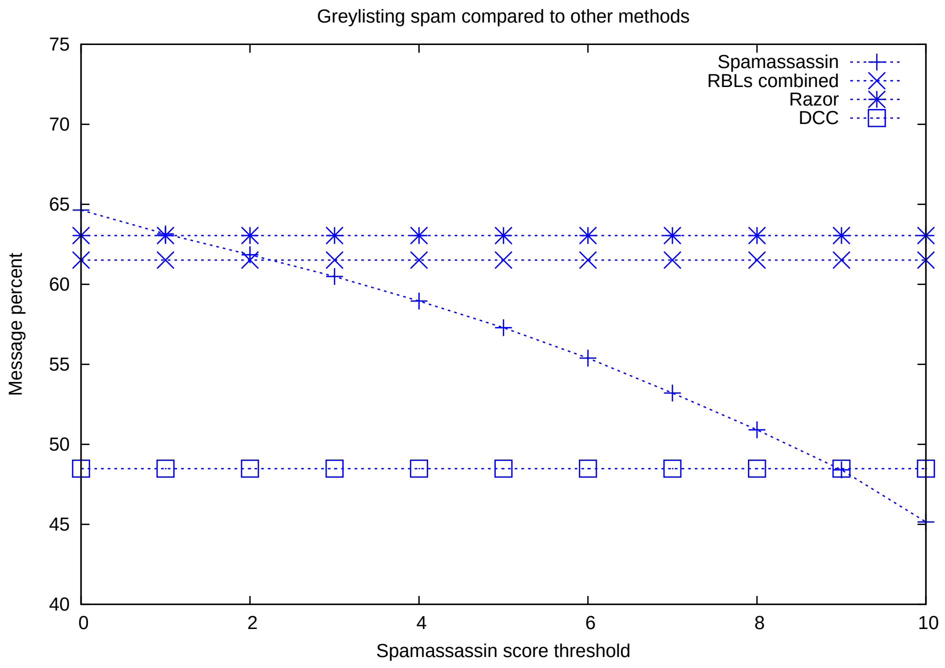

Figure 8: A graph showing the amount messages marked as spam with greylisting compared with the other methods.

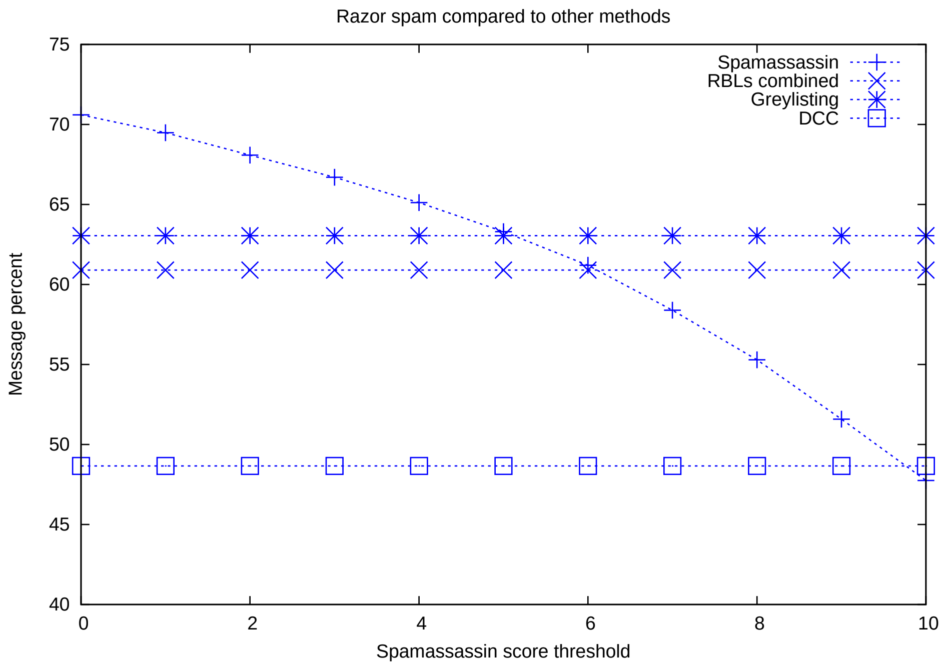

Figure 9: A graph showing the amount messages marked as spam with razor compared with the other methods.

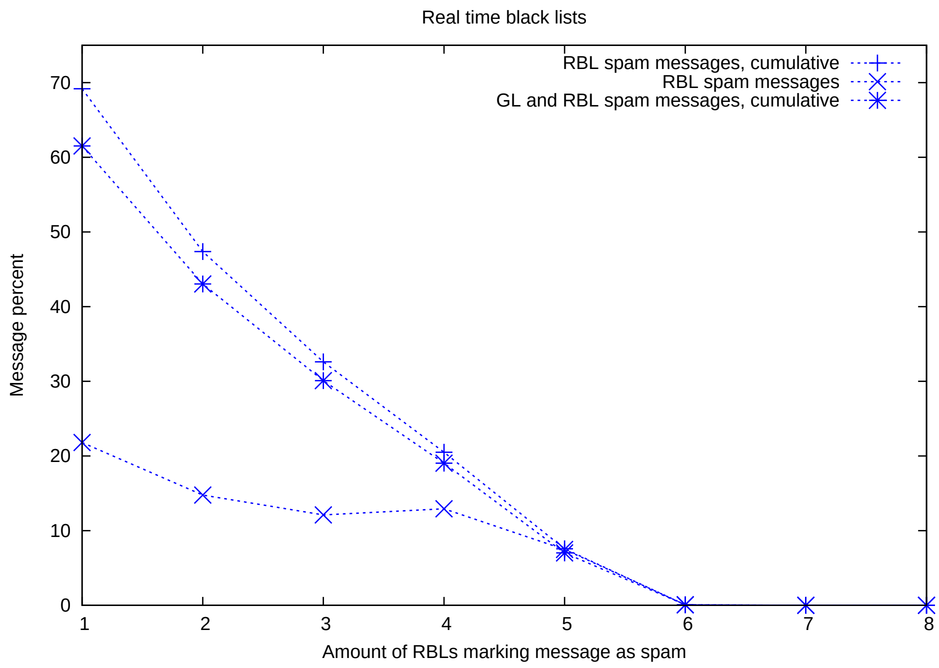

Figure 10: A graph showing the amount black lists that marked messages as spam.

relation to reducing spam in user's mailbox is reduced since some spammers tries random addresses and the first experiment server didn't have access to valid addresses. Bugs in the experiment software can also cause reduced internal validity. Software modifications related to the experiment are likely to have more bugs than the original software since fewer people have audited the modifications.

Reliability is defined as "the extent to which an experiment, test or any measuring procedure yields the same results on repeated trials" [4]. The same experiment can be performed by anyone with enough knowledge and access to the experiment software and configuration. But the email that passed the experiment servers can never be used again for obvious reasons. Nevertheless, people with access to an equal environment like GUC could perform the experiment and, if it's within a reasonable time of this writing, get similar results. As time passes, spammers and the anti spam methods will develop and the contents of a new experiment must reflect this.

5 Discussion

This chapter will discuss the results of the experiment, give possible explanations of the results and comment on what may happen in the future.

5.1 Sender Authentication Standards

DomainKeys and SPF had a spam ratio of 0.1% and 3.1%. There are many possible reasons for this low result. First, DomainKeys and SPF are not yet Internet standards. Therefor, few Internet email administrators may have implemented them. There is also a possibility that the filters marked ham as spam because of bad user configuration since the proposed standards are a relatively new invention. The last reason for this low spam ratio is that spammers are clever and avoids getting caught by these methods. However when looking at all the spam that is caught by other methods this seems like an unlikely reason.

Sender authentication standards will probably stop email with fake addresses and content in the future. This will help stop fraud and phising attacks. These standards will however have low impact on other types of spam as spammers can easily circumvent them.

5.2 Methods Based on Bayesian Algorithms

Bogofilter detected a spam ratio of 31.7% in the experiment. This is low compared with the other methods. The reason for this seems that Bogofilter was trained with spam and ham email boxes that are freely available taken from for example mailinglists. The documentation claims that best results are achieved by training the filter with the email that a user receives. Articles refereed to in the literature study confirms that better results are possible [35] [33].

Content filtering and improvements to the Bayesian and similar algorithms will probably survive in the future as spammers need to get their message through to their targets. Technical anti spam methods can in many cases be circumvented, but the email with it's spam message can be recognized as spam with the right content filtering.

5.3 Ruled Based Anti Spam Solutions

Spamassassin had a spam detection ratio of 66.8%. This is a high percentage considering only the static rules of Spamassassin was used in the experiment. It should be easy for spammers to make sure that a static anti spam solutions like Spamassassin would not mark the spam as spam.

In Figure 5, the methods compared with Spamassassin agrees on a lower spam percentage as the threshold is increased. Real time black lists and Spamassassin however gives only a few thousand messages difference with a threshold difference of 10. One reason for this may be that spammers that are added to the black lists are less sophisticated and doesn't check their spam against Spamassassin.

Spamassassin has the ability to include other methods like Razor, real time black lists and Bayesian filtering when testing for spam. This will make it also work in the future since static rules by themselves should be easy bypass. Maintaining and updating the static rules of Spamassassin is a resource demanding task for the developers and therefor Spamassassin can be more inaccurate in the future.

5.4 Distributed Anti Spam Solutions

DCC and Razor had a spam detection ratio of 58.7% and 74.6%. Since DCC has no other spam indicator than how many times a message has been seen, it is very inaccurate when used alone. Razor which is based on user feedback has a result which is similar to greylist and real time black lists. Given that the algorithm used for detection similar messages and that the user feedback is valid, Razor should give a low false positive ratio. With the assumption above, the system must be very popular since it catches a relatively high number of spam email.

Razor should also be ready for the future. Validating user feedback and improving the fuzzy hash are possible improvements. But looking at the user feedback, with a very low false positive rate, it seems that Razor is an acceptable solution today.

5.5 Greylisting

Greylisting had a spam detection ratio of 71.3%. If every SMTP server followed the standard, the false positive ratio would be 0%. The GUC's IT department didn't receive any complaints of missing email in the experiment period due to greylisting. However the probability of non standard SMTP servers existing on the Internet is high so greylisting will probably have a low false positive ratio. Also, there were no complaints of delayed email, thus greylisting has few negative effects.

In the future, spammers can use smarter SMTP sending software against greylisting, but that will raise the cost of sending spam. To raise the cost even further, the time before accepting a triplet can be increased. The effectiveness of greylisting depends on how much it will be used on the Internet and the cost of spammers to circumvent it. If every SMTP server used greylisting spammers would have to try go around it, but if few use it spammers will probably use the same methods for sending spam as today.

5.6 Real Time Black Lists

When requiring hit from one of the black list, 69.5% of the email would be marked as spam. There are different criteria for adding IP addresses on different black lists. Some check for insecure servers and some adds known spam sources. Therefor users sending genuine email through an insecure system will often see that their email will be rejected or just deleted. Comparing the user feedback with the black lists, black lists give a false positive ratio of 3.1%. This is a twice as much as the best, Spamassassin with 1.5%.

The Figure 10 shows that requiring hits from more black lists decreases the amount of spam marked. The RBL which marked the most messages as spam marked 190,103 messages. Therefor the RBLs are not marking the same messages since requiring hit from only one RBL equals 236,717 messages. Requiring hit from two or more lists are probably more secure, but decreases the number of detected spam messages to 162,133.

In the future, real time black lists can still be effective. But a threat to it's effectiveness is that spammers hijack insecure hosts and use them to relay spam. When they are black listed, they find new machines to relay from. Also when IP version 6 is commonly used on the Internet the address space will be much bigger, thus spammers may be able to change the addresses they relay spam from more often.

5.7 User feedback and False Positives

When excluding sender authentication methods, the methods that had the lowest false positive ratio were Razor, RBLs and Spamassassin. The possibility that a user reported a wrong spam status for an email is high, thus the actual numbers can be close to 0%. It is poosible to run all these methods in an anti spam solution to make it more secure.

5.8 Real Spam Ratio

The highest graph in Figure 6 shows how many messages that would be marked as spam if requiring different numbers of methods which agree that the message is spam. When requiring only one method, 311,869 out of 342,181 messages would be marked as spam. This is 91.1% of the messages, a relatively high number which probably has included many false positives due to including methods with a high false positive ratio.

Another interesting thing to observe in Figure 6 is that requiring hits from 5 methods and only 5 would mark the most messages, a total of 127,811 messages or 37.4%, as spam. When requiring such a high number of methods before deciding if a message is spam, it should give a low false positive ratio.

6 Suggested Anti Spam Solution

This chapter will suggest some anti spam solutions based on the results of this report. The suggestions will be categorized into solutions according to the risk of the solution marking a ham email as spam email. Marking the most spam email as spam is also taken into consideration. Resources needed for the various solutions are also commented.

6.1 Low Risk Solution

A solution including greylisting in combination with black lists seems appropriate for this risk level. If the sender SMTP server is black listed, the recipient server can greylist the email. This will result in fast delivery for unknown sources that are not black listed. Sources which are black listed by mistake will get their email delivered as long as they are running an SMTP service which follows the standard, even though it takes a little more time. This configuration will stop around two thirds of email traffic according to the experiment. And it will have little or none negative effects for the users shown by the experiment which ran greylisting on every email. It is also a solution that requires little resources compared with content based methods.

6.2 Medium Risk Solution

To detect more spam than the low risk solution, requiring hit from both Razor and Spamassassin before deciding if a message is spam will mark ten percent of the remaining traffic from the low risk solution as spam. These two methods are different in the way they work and the user feedback in the experiment showed a low false positive ratio for both methods. Therefor the combination should be safe to use for w medium risk solution. False positives from this solution should be under 1% according to the experiment. Adding this solution to the low risk solution will be more resource demanding since it operates on the email content.

6.3 High Risk Solution

For a high risk solution two or more of the methods which catches a lot of spam can be used. Spamassassin, Razor or a combination of the real time black lists are good candidates. This approach will easily mark 75% of the emails as spam, but may also give over 1% false positives. Methods which analyze the contents of the email will also be resource demanding compared to the others.

7 Conclusion

We have studied different anti spam solutions and the work that others have done regarding anti spam solutions. We have also designed, implemented and ran an experiment in a production environment to look at how various anti spam methods perform and compares to each other on real data. The results have been analyzed and presented. Suggestions for anti spam solutions based on the work is also presented.

The main methods in the experiment detected from 37% to 75% spam and had a low false positive ratio compared with user feedback. Requiring several methods before deciding that an email is spam still gives a high spam percentage. This confirms what the literature claims regarding spam traffic on the Internet. Also, some methods claiming to have a low false positive ratio does. It is impossible to conclude that any method have a zero false negative ratio though.

The future will probably give an even higher spam percentage ratio. For every new anti spam solution, the spammers adapt and find ways to circumvent it. It is important that new and existing methods has the lowest false positive ratio as possible so the users can trust the email system and it can continue to be a choice for communication in the future.

8 Future Work

To further enhance the experiment and provide more accurate data, the user can provide more feedback. The following are possible ways to do this:

- Run a training period before the experiment starts where users train the Bayesian filters with spam and ham they receive.
- Email denied by greylisting can be stored and made available for every email user so they can provide feedback to these emails also.
- Look at how to improve user feedback to get more accurate numbers of false positives and false negatives.

The experiment can also be ran in different organizations and repeated after a period of time to see if there are differences in the spam amount and spam content that are received.

The resources needed for each anti spam method can also be recorded while running the experiment. The results will show which anti spam methods that requires the least resources while providing good results in marking spam messages as spam and ham messages as ham.

Bibliography

[1] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM Press, 2000.

[2] S. Atkins. Size and cost of the problem. In *Proceedings of the Fifty-sixth Internet Engineering Task Force (IETF) Meeting*. SpamCon Foundation, March 16-21 2003.

[3] Jeff Balvanz, Don Paulsen, and Joe Struss. Spam software evaluation, training, and support: fighting back to reclaim the email inbox. In *SIGUCCS '04: Proceedings of the 32nd annual ACM SIGUCCS conference on User services*, pages 385–387. ACM Press, 2004.

[4] Edward G. Carmines and Richard A. Zeller. *Sage University Paper*, volume 17 of *Quantitative Applications in the Social Sciences*. Sage Publications, Inc., 1979.

[5] S. Chhabra, W.S. Yerazunis, and C. Siefkes. Spam filtering using a markov random field model with variable weighting schemas. In *Proceedings. Fourth IEEE International Conference on Data Mining*, pages 347–350, 2004.

[6] Lorrie Faith Cranor and Brian A. LaMacchia. Spam! *Commun. ACM*, 41(8):74–83, 1998.

[7] John W. Creswell. *Research Design — Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications, Inc., 2nd edition, 2003.

[8] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, New York, NY, USA, 2004. ACM Press.

[9] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. An open digest-based technique for spam detection. In *Proceedings of the 2004 International Workshop on Security in Parallel and Distributed Systems*, 2004.

[10] P. Deepak and Sandeep Parameswaran. Spam filtering using spam mail communities. In *Proceedings. The 2005 Symposium on Applications and the Internet*, pages 377–383, 2005.

[11] Pawel Gburzynski and Jacek Maitan. Fighting the spam wars: A remailer approach with restrictive aliasing. *ACM Trans. Inter. Tech.*, 4(1):1–30, 2004.

[12] Gjøvik University College. `http://www.hig.no/` (Visited January 2005).

[13] Luiz Henrique Gomes, Cristiano Cazita, Jussara M. Almeida, Virg&#237;lio Almeida, and Jr. Wagner Meira. Characterizing a spam traffic. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 356–369. ACM Press, 2004.

[14] Paul Graham. A plan for spam. World Wide Web, 2002. `http://www.paulgraham.com/spam.html` (Visited January 2005).

[15] Paul Graham. Better bayesian filtering. In *Spam Conference at Cambridge MA*, 2003.

[16] Evan Harris. The next step in the spam control war: Greylisting. World Wide Web, 2003. `http://projects.puremagic.com/greylisting/whitepaper.html` (Visited January 2005).

[17] K.C. Ivey. Spam: The plague of junk e-mail. *Computer Applications in Power*, 11:15–16, April 1998. Issue: 2.

[18] Jaeyeon Jung and Emil Sit. An empirical study of spam traffic and the use of DNS black lists. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 370–375. ACM Press, 2004.

[19] Editor J. Klensin. RFC2821: Simple Mail Transfer Protocol. Technical report, AT&T Laboratories, April 2001.

[20] Lov om behandling av personopplysninger. `http://www.lovdata.no/all/nl-20000414-031.html`.

[21] S. Mane, J. Srivastava, San-Yin Hwang, and J. Vayghan. Estimation of false negatives in classification. In *Proceedings. Fourth IEEE International Conference on Data Mining*, pages 475–478, 2004.

[22] R. Matsumoto, Du Zhang, and Meiliu Lu. Some empirical results on two spam detection methods. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration*, pages 198–203, 2004.

[23] Brian McWilliams. *Spam Kings*. O'Reilly Media, Inc., 2004.

[24] Monty Python's Flying Circus. *Just the Words*, volume 2, chapter 25, pages 27–28. Methuen Publishing Ltd, 1989.

[25] Cormac O'Brien and Carl Vogel. Spam filters: bayes vs. chi-squared; letters vs. words. In *ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies*, pages 291–296. Trinity College Dublin, 2003.

[26] T. Oda and T. White. Increasing the accuracy of a spam-detecting artificial immune system. *The 2003 Congress on Evolutionary Computation*, 1:390–396, December 2003. 8-12 Dec. 2003.

[27] L. Pelletier, J. Almhana, and V. Choulakian. Adaptive filtering of spam. In *Proceedings. Second Annual Conference on Communication Networks and Services Research*, pages 218–224, 2004.

[28] S.L. Pfleeger and G. Bloom. Canning spam: Proposed solutions to unwanted email. *IEEE Security & Privacy*, 3(2):40–47, Mar-Apr 2005.

[29] Jeffrey Posluns, editor. *Inside the Spam Cartel*. Syngress Publishing, Inc., 2004.

[30] Jon Postel. RFC706: On the Junk Mail Problem. Technical report, Network Working Group, November 1975.

[31] Vipul Ved Prakash. Razor: spam should not be propagated beyond necessity. World Wide Web, 2005. `http://razor.sourceforge.net` (Visited January 2005).

[32] Editor P. Resnick. RFC2822: Internet Message Format. Technical report, QUALCOMM Incorporated, April 2001.

[33] Matt Sergeant. Internet-level spam detection and spamassassin 2.50. In *Spam Conference at Cambridge MA*, 2003.

[34] Rhyolite Software. Distributed checksum clearinghouse. World Wide Web, 2005. `http://www.rhyolite.com/anti-spam/dcc/` (Visited January 2005).

[35] N. Soonthornphisaj, K. Chaikulseriwat, and Piyanan Tang-On. Anti-spam filtering: A centroid-based classification approach. *2002 6th International Conference on Signal Processing*, 2:1096–1099, August 2002. 26-30 Aug. 2002.

[36] Brian Whitworth and Elizabeth Whitworth. Spam and the social-technical gap. *Computer*, 37(10):38–45, 2004.

[37] Meng Weng Wong. Spf overview. *Linux J.*, 2004(120):2, 2004.

[38] Yahoo! Inc. Domainkeys: Proving and protecting email sender identity. World Wide Web, 2005. `http://antispam.yahoo.com/domainkeys` (Visited January 2005).

[39] Kenichi Yoshida, Fuminori Adachi, Takashi Washio, Hiroshi Motoda, Teruaki Homma, Akihiro Nakashima, Hiromitsu Fujikawa, and Katsuyuki Yamazaki. Density-based spam detector. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 486–493. ACM Press, 2004.

[40] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.

A Obtaining The Experiment Software

The experiment software or a more technical overview of the experiment can be obtained by sending an email to the thesis author at anders@wiehe.org. The software may not be maintained therefor it may not be working without modifications or rewriting in the future.