University of Alberta

An Automation of Mail Channels to Eliminate Junk E-Mail

by

Nicholas M. Boers

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2006

Your file Votre référence
ISBN: 0-494-13796-7
Our file Notre référence
ISBN: 0-494-13796-7

To my grandparents, parents, sister, and Xiye.

Abstract

Unsolicited bulk e-mail (spam) threatens e-mail's effectiveness for communication. Existing techniques that aim to reduce or prevent it decrease e-mail's reliability or place unnecessary restrictions on e-mail's use. This thesis describes a system, the Spam Free e-Mail (SFM) service, that effectively eliminates spam without such drawbacks.

SFM combines challenge/response techniques with mail channels, allowing it to eliminate spam and support *solicited* bulk e-mail. Its subscribers initially create one or more e-mail addresses that they can openly share and publish. Messages sent to them never reach the subscriber; instead, they solicit an automatic response from the system. The response communicates a new e-mail address unique to the sender and prompts the sender to resend the original message to it. To solicit a bulk mailing, subscribers can manually create one of these alternative addresses unique to the bulk mailer.

Acknowledgements

I would like to thank my supervisors, Dr. Pawel Gburzynski and Dr. Ioanis Nikolaidis, for their support, encouragement, and feedback. Pawel conceived the Spam Free e-Mail service and I am grateful for his suggestion to expand it into a thesis topic. Ioanis also provided valuable feedback into an early draft of this thesis. Both made themselves available whenever I needed them.

Dr. Jon Schnute, a valued friend and mentor, helped prepare me for graduate studies. After I started, he continued to offer his support and encouragement on numerous occasions. Dr. Jim Uhl and Dr. Peter Walsh, both professors at Malaspina University-College, first encouraged me to pursue graduate studies. They provided me with great inspiration.

Finally, I would like to thank my family. My grandparents, parents, sister, and Xiye provided constant love, encouragement, and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Millions of Internet users communicate using electronic mail (e-mail). It represents a fundamental use of the Internet as supported by surveys of Canadian and American Internet users alike [53, 54, 47]. With the rapid growth of the Internet, it now reaches into over half of Canadian homes [53], three quarters of Canadian private-sector enterprises, and almost all of Canadian public-sector enterprises [54].

Anyone with access to the Internet can obtain one or more e-mail addresses. Internet service providers (ISPs) often provide their subscribers with at least one. Other companies, known as e-mail service providers (ESPs), specialize in offering e-mail accounts. Some ESPs even offer accounts at no charge, paying for them with advertisements.

E-mail can provide many advantages over traditional mail:

- *speed*, where messages often transfer from source to destination in mere seconds;
- *cost*, where messages have negligible cost to both sender and receiver;
- *reliability*, where messages are rarely lost, and delayed/failed deliveries often result in feedback; and
- *privacy*, where technologies can encrypt messages and ensure private communication.

These advantages make it an appealing alternative to traditional mail and have led to its mass adoption. Of them, its cost is arguably a double-edged sword; it allows for one of e-mail's primary threats.

Section 1.1 introduces this primary threat to e-mail. The techniques introduced in Section 1.2 fail to solve the problem and consequently motivate our work. With that motivation, Section 1.3 outlines our contribution. Finally, Section 1.4 overviews the remainder of this thesis.

1.1 Problem Statement

Businesses have long used traditional mail for marketing. In doing so, two of the fundamental costs are (a) physical production (e.g., printing) and (b) distribution (i.e., postage). These costs have helped keep the quantity of traditional junk mail in check.

E-mail carries no physical production and little distribution cost. A computer transparently replicates a single message *while* distributing it, and the primary difference between sending one and a million messages is time. Since software automates the entire distribution, this time costs relatively little. The process only requires human-intervention to set it up.

When sending bulk e-mail is so inexpensive, low response rates can generate a profit for advertisers and scam artists. A 2002 estimate suggests a response rate as low as 0.005% [15] or 50 per million. At the same time, a marketer could purchase 70 million e-mail addresses on compact disc for US$150 [15]. With a relatively small investment, a bulk mailer could potentially obtain 3500 customers.

Low costs combined with ample response rates have led to a proliferation of bulk mailings. Messages come in many forms: advertisements for goods and services and scams to name a couple. Current estimates suggest that around 70% of all e-mail is junk.[1]

The costs born by the recipients of this e-mail exceed those on the senders. Receiving ISPs and ESPs must

- receive the e-mail using their own bandwidth,
- store it until their subscribers (i.e., the final recipients) retrieve it, and
- relay it to their subscribers.

The final recipients then pay the highest cost; they waste their valuable time discerning it from non-junk. Unlike with traditional mail, technology can help solve this problem.

1.2 Motivation

While filtering traditional junk mail requires a human, computers can automatically filter junk e-mail. Many techniques have come from academia and industry. Their effectiveness ranges from simply reducing its volume to completely eliminating it. At the same time, they all have a variety of weaknesses, some of which threaten to cause more harm than good.

Blacklisting blocks specified senders by e-mail address, domain, or IP address. Blocking individual *e-mail addresses* worked many years ago, when senders reused

[1] http://www.messagelabs.com/

them. Now, they rarely reuse them and this technique proves useless. Blocking entire *domains* or *IP addresses* of notorious senders can still help. However, bulk mailers may send using a bona fide ISP or ESP and subsequently cause its inclusion in a blacklist. As a result, blacklisting risks blocking legitimate senders.

Whitelisting only admits mail from certain senders, commonly specified by e-mail address or domain. Establishing new contacts requires communication through another medium (e.g., traditional mail, in person, or telephone). This technique has limited support for solicited bulk mail and is criticized for being too effective when used alone. A user seldom knows the sender's address for a solicited bulk message a priori. The creation and management of such lists also places a burden on their user.

Challenge/response enhances whitelisting by allowing senders to add themselves to the list. These systems initially block messages from unrecognized senders; they reply to them with a challenge. When the sender completes the steps outlined in the challenge, the system adds the sender to the whitelist. They use challenges trivial for humans and difficult for bulk mailers. Although an improvement over whitelisting, this technique still lacks support for automatic bulk mailings.

Techniques based on filtering are the most popular, and all of these techniques decrease e-mail's reliability. Rule-based filters analyze the content of a message to determine whether it is junk. Although certain content may *suggest* that a message is junk, a definitive decision is impossible and always prone to error. Collaboration-based filters use people to hand-classify messages; once a specific number of users identify a message as junk, other users no longer receive it. Unfortunately, different people have different beliefs about what constitutes junk, and this technique is also prone to error.

Chapter 2 discusses these techniques and several others that attempt to solve the problem. In summary, all of the commonly implemented techniques have weaknesses, and they motivate the development of a new technique that more effectively eliminates junk e-mail.

1.3 Design Goals and Features

This thesis presents a technique to completely eliminate junk e-mail without imposing restrictions on the use of e-mail or destroying its reliability. This solution follows from several observations.

Many of the existing techniques judge e-mail as it enters the in-box. They decide its fate solely on its content, both headers and body, without context. Existing techniques fail because nothing in the content definitively identifies junk e-mail. A serious solution to the problem must account for more than just content; it must begin before the message arrives. Although not initially obvious, junk e-mail does have distinguishing

characteristics, and solutions must exploit them.

Challenge/response techniques account for a non-content characteristic of junk e-mail. They eliminate junk e-mail based on the premise that it all originates from automatic bulk mailings. In a bulk mailing, the sender does not care about each individual instance of the message. A single undelivered message has no effect on the overall success of a bulk mailing. This contrasts with an individual mailing by a human sender. In the human instance, someone spent time and effort composing and sending the message. This investment of valuable time creates a desire to see it delivered. If it goes undelivered, the attempt at communication fails. The difference between the determination of a bulk mailer and an individual human sender is why challenge/response techniques work. When these systems send a challenge, the human recipient will usually answer it while the bulk mailer will not. The fundamental problem with challenge/response is that although junk mail always originates from bulk mailings, bulk mailers also produce *solicited* bulk mail.

When challenge/response systems insist on challenging all senders, they essentially prevent solicited bulk mail. Both solicited bulk mail and junk mail are very similar: they both originate from bulk mailings and neither sender cares about the delivery of individual instances. The fundamental difference is in how they obtain the destination e-mail address.

E-mail addresses have always been a publishable point of contact for e-mail users, the type of information that they publish on their business cards and web sites. In these cases, the addresses serve to solicit *human* contact; however, producers of junk e-mail also collect (i.e., harvest) them. When a user solicits a bulk mailing, the user explicitly provides the sender with an address. This explicit invitation is an ideal opportunity to differentiate solicited from unsolicited bulk e-mail.

The system documented in this thesis, the Spam Free e-Mail (SFM) service, works as follows. Its users have one or more e-mail addresses that they can openly publish, e.g., on business cards and web sites. SFM blocks all messages sent to these addresses since any message could be unsolicited bulk e-mail. At the same time as it blocks the mail, it

1. creates a unique e-mail address intended solely for the sender of the blocked message and
2. communicates it to that sender in a reply to the original message.

If the sender is human and cares about the delivery of the message, the sender can resend it to the new e-mail address. We present the e-mail address in a form trivial for humans and difficult for bulk mailers to decipher. Since current bulk mailers cannot

answer the challenge, this technique effectively blocks them from reaching the users on published addresses.

As previously mentioned, e-mail users sometimes solicit advertisements, sign up for mailing lists, or otherwise request bulk e-mail. To support solicited bulk mailings, users can manually create mail channels (i.e., new e-mail addresses) suitable for each solicited bulk mailer. When soliciting the bulk mailing, they can provide the unique mail channel. Since these addresses are not published, they are not vulnerable to harvesting and thus unlikely to receive abuse. If a channel becomes abused (e.g., an untrustworthy company sells it), a user can enable a whitelist, limiting it to only known and acceptable e-mail addresses.

1.4 Overview

Five chapters follow this brief introduction. Chapter 2 provides some history on e-mail use, its primary threats, some of the techniques used to fight those threats, and a definition of spam. Our solution builds on two existing technologies, mail channels and challenge/response, and Chapter 3 describes key events in each one's history. Chapter 4 introduces our solution to the spam threat. We have named our system Spam Free e-Mail, or simply SFM, to reflect its ability to completely eliminate spam. With a firm grasp of how SFM works, Chapter 5 discusses various aspects of our system. Finally, Chapter 6 provides a summary of the problem and our solution.

Chapter 2

Background

Electronic mail (e-mail) began anywhere from 1832 to the 1960s, depending on one's definition. If e-mail is simply the communication of messages electromagnetically, it predates computers with the invention of the telegraph in 1832 [16]. E-mail in a modern sense developed much later with the advent of timesharing systems in the 1960s. On these systems, users had unique usernames to which others could address mail [16]. For example, the Tele-Computer Center, operated by the Westinghouse Electric Corporation in the mid-1960s, supported primitive e-mail [48]. The Center could communicate with 325 company locations, including field sales offices, manufacturing divisions, and warehouse shipping points. A user at one location could send an electronic message to any other location on the network. The Center would route the messages and store them temporarily if the recipient was offline.

Large-scale computer networks started to develop around 1970 (e.g., the ARPANET). On the ARPANET, e-mail was one of four primary applications, along with news, remote login, and file transfer. In the first primitive e-mail systems, users would simply pass around text files using file transfer protocols. Each message resided in its own file, and by convention, the recipient's address always appeared as its first line [55].

These primitive e-mail systems received several complaints [55]. Due to the use of file transfer protocols, (a) sending e-mail to multiple recipients was inconvenient, (b) composing mail was not user-friendly as it involved both a text editor and file transfer program, and (c) the sender never knew whether a message arrived. By simply using text files as messages, (a) the lack of an internal message structure complicated computer processing and (b) messages were unable to support a mixture of text, graphics, and audio. These limitations led to proposals for more elaborate e-mail systems.

Two primary proposals came forward in an attempt to standardize e-mail. In 1982, the Internet Engineering Task Force (IETF) created the first proposal. It came as two Request for Comments (RFC) documents: RFC 821, Simple Mail Transfer Protocol [49] and RFC 822, Standard for the Format of ARPA Internet Text Messages [9]. In

1984, the International Telegraph and Telephone Consultative Committee (CCITT), now known as the ITU Telecommunication Standardization Sector (ITU-T), issued the second proposal. They issued recommendation X.400 (Message Handling Services: Message Handling System and Service Overview).

Given the simplicity of RFC 821/822, most organizations went that route [55]. The X.400 proposal was criticized for its complexity: "X.400 was so poorly designed and so complex that nobody could implement it well" [55] and its complexity made it "difficult and time consuming to implement; hence, products based on it tend[ed] to be large, expensive, and difficult to deploy" [28]. In the end, e-mail systems based on the RFCs see widespread use today, while X.400 sees little use.

RFC 821 and 822 provided the groundwork for the current e-mail system. These documents specify the protocol for moving messages between computers and the format of the messages themselves, respectively. Now, new versions of each standard (RFC 2821 [35] and 2822 [50], respectively) obsolete the originals. An abundance of additional standards enrich modern e-mail with new features and functionality.

Section 2.1 describes the growth of e-mail and its significance in modern life. Given that popularity, both advertisers and scam artists exploit it, as described in Section 2.2. To counter these threats, researchers from both academia and industry propose many solutions, with some introduced in Sections 2.3 to 2.5. Finally, we present some of the many definitions of the term *spam* in Section 2.6.

2.1 Significant Usage

Many Canadians use the Internet both at home and work (Table 2.1). A 2003 survey suggests that 54.5% of Canadian households use the Internet at home; of those, over 95% use e-mail. E-mail also sees popularity in both private and public sector Canadian enterprises. In the private sector, a 2004 survey shows that 81.62% of enterprises use the Internet, with over 93% of those using e-mail. In the public sector, the numbers are even greater at 99.89% and 100%, respectively. The United States of America has an estimated 161 million Internet users (2004) [33]. Surveys suggest that around 86% (138 million) of those Internet users send and receive e-mail at least occasionally [47]. In North America, these surveys show that e-mail is a fundamental use of the Internet.

To the best of our knowledge, no research explores a possible difference between worldwide e-mail usage and North American usage. With an estimated 841 million Internet users worldwide (2004) [33], we can speculate that the global number of e-mail users exceeds 700 million.

Table 2.1: Internet and e-mail use in Canadian homes and enterprises [53, 54].

Where	Use	1997	1998	1999	2000	2001	2002	2003	2004
home	Internet	16.0	22.6	28.7	40.1	48.7	51.4	54.5	N/A
	e-mail	13.3	19.3	26.3	37.4	46.1	48.9	52.1	N/A
	%	83.1	85.4	91.6	93.3	94.7	95.1	95.6	N/A
private-sector enterprise	Internet	N/A	N/A	N/A	63.40	70.84	75.67	78.27	81.62
	e-mail	N/A	N/A	N/A	60.46	66.03	71.22	73.89	76.60
	%	N/A	N/A	N/A	95.36	93.21	94.12	94.40	93.85
public-sector enterprise	Internet	N/A	N/A	N/A	99.16	99.71	99.59	100.00	99.89
	e-mail	N/A	N/A	N/A	98.98	99.71	99.60	99.81	99.89
	%	N/A	N/A	N/A	99.82	100.00	100.01	99.81	100.00

2.2 Threats

Many e-mail users are already familiar with one of e-mail's primary threats – *spam*. Although they recognize the term and innately know what it means, they often have difficulty articulating a generally agreeable definition. Many definitions exist, but we will simply define it as unsolicited bulk e-mail (UBE) for the time being. Section 2.6 discusses this simple definition as well as some expanded, specific definitions.

As the popularity of e-mail grew, so did its marketing potential. In 1978, an employee of Digital Equipment Corporation (DEC) sent what many believe was the first spam [61]. Gary Thuerk of DEC sent an e-mail addressed to over 320 people, inviting them to attend a product presentation. This e-mail initiated a series of discussions and flames.

Charity and money-making schemes occasionally appeared over the next several years with increasing frequency. People often responded to them with a series of flames [61]. The first chain letters occurred in February 1982 and August 1985; both resulted in flames.

The amount of spam increased in the 1990s with the commercialization of the Internet. Some now notorious spam and spammers appeared in 1994 such as the Jesus spam, Canter and Siegel spam, and Jeff Slaton. As a result, spam started to grow exponentially.

In 1995, the first commercial spamware (Floodgate) became available. In the same year, the first known public list of around 2,000,000 e-mail addresses went on the market. With the available software and addresses, spam became much easier to send. As a result, its popularity continued to grow.

It is difficult to measure the current volume of spam, which leads to a wide range of estimates. John Graham-Cumming performed an end-user survey from November 23, 2004 to December 30, 2004 [22]. His results indicate that 98.5% of e-mail users receive spam, where the average user receives 318 spams per week. In total, the results sug-

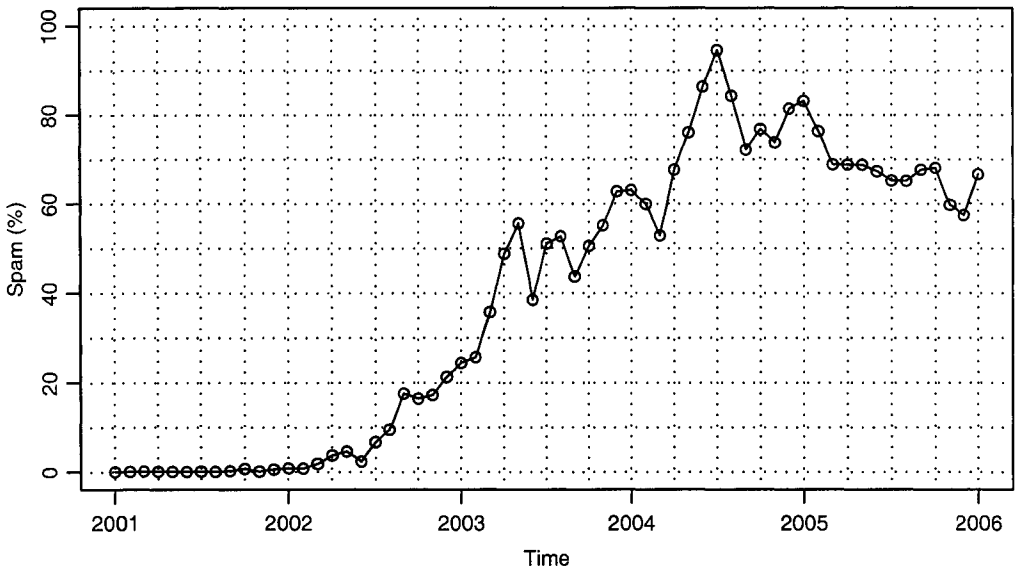

Figure 2.1: Proportion of all e-mail detected as spam by MessageLabs.

Table 2.2: FTC categorization of spams that contained scams [14].

Category	**Description**
chain letters	involve money or other valuables and promise big returns
work-at-home schemes	promise the ability to work from home
weight loss claims	promote programs or products for easy or effortless long-term weight loss
credit repair offers	offer to erase negative information from credit reports
advance fee loan scams	offer loans, regardless of credit history
adult entertainment	claim to offer free adult content in exchange for downloading a *viewer* or *dialer* program

gest that 77% of e-mail is spam. Message Labs,[1] self-proclaimed the world's leading provider of messaging security and management services to business, regularly publishes data on the proliferation of spam. Figure 2.1 shows the proportion of spam found in all e-mail as calculated by MessageLabs. Their estimate for November/December 2004 is at least 10% lower than John Graham-Cumming's survey results.

Spam comes in various forms and can be categorized in different ways. In April 2002, the FTC made an effort to classify spam containing scams [14]. They created six categories (Table 2.2). For a more general classification of spam, see the ten categories in Table 2.3.

When looking at spam on a global scale, one can categorize it based on its market. Microsoft researchers [29] suggest three categories:

[1]http://www.messagelabs.com/

Table 2.3: A categorization of spam with the proportion that each type represents [30].

Category	Definition	%
porn/sex non-graphic	enhancers with sexual connotation, links to porn	34%
financial	refinancing, get out of debt, financial advice	13%
dubious products	pirated software, diplomas, etc.	10%
Rx and herbal	cheap drugs or herbal supplements	10%
porn/sex graphic	anything that contains pornographic images	7%
newsletters	any newsletter that is not selling something	6%
scams	get rich quick, phishing scams, etc.	6%
insurance	health, dental, life, home, and auto insurance	4%
travel and casino	selling airline tickets, hotel reservations, rental cars; Internet casino sites; other gaming sites	3%
other spam	everything else that appears to be spam	8%

1. domestic: products such as financial services, insurance, government grant programs, and items deemed too expensive to ship internationally that require a domestic presence to sell;
2. semi-domestic: small products such as Viagra and other medical products, college diplomas, magazines, and more that are small enough to ship to/from nearby countries;
3. international: products or services such as pornographic web sites, software, and scams that require no physical shipping nor domestic presence.

The authors describe that legislation can potentially impact domestic spam, but will have no impact on international spam in the semi-domestic and international categories.

2.2.1 Phishing and Identity Theft

Starting in 2001, the *phishing* subtype of spam developed. These e-mail messages purport to originate at a legitimate company, often a well-known online or financial institution. They convey a message of urgency, such as the immediate need for the recipient to verify a bank account. They include a link that points to a replica of a commercial web site, which prompts the user for a bank card number, credit card number, or username and the associated password or PIN.

One of the very first attempts at phishing targeted `paypal.com` users [1]. Claiming that Paypal had lost their user information, the e-mail directed users to a look-alike site that asked visitors to enter their account and credit card information. Subsequently, the fraudsters used the collected account information to steal Paypal users' money.

Phishing schemes have two victims. First, individual users who are tricked into divulging personal information often lose money. Gartner, Inc. estimates that 57 mil-

lion U.S. adults had received phishing e-mails as of April 2004 [39]. Of those people, roughly 3% (1.78 million) remembered giving phishers sensitive financial or personal information, of which roughly half subsequently fell victim to identify theft. Second, reputable companies lose credibility online and an ability to electronically contact their customers. Their customers become very cautious about giving out any information online.

2.3 Fundamental Techniques

Spam threatens the usefulness of e-mail for daily communication. In response to its growing attack, researchers are on the defensive with proposals that they hope can save e-mail. Their complexity has grown over time. While simple solutions defeated early abuse, an arms race has developed between spammers and those aiming to prevent it. Now, a variety of approaches tackle the problem, but none effectively solve it.

Several fundamental techniques evolved in the battle on spam. While blacklisting was one of the first, a spam fighter's arsenal now contains whitelisting, challenge/response, content filtering, and collaborative filtering. The effectiveness of these techniques varies.

2.3.1 Blacklisting

With blacklisting, a client, server, or even router explicitly blocks specific senders based on their e-mail address, domain, or IP address. When applied at the user level, clients often block e-mail addresses. At the server or ISP level, servers and routers are more likely to block domains and IP addresses. ISPs that choose not to maintain their own blacklist can subscribe to an externally-maintained list. For example, the Spamhaus Project publishes the Spamhaus Block List,[2] a database of IP addresses belonging to spammers, spam gangs, and spam support services. As of February 21, 2006, they claimed that their list was protecting 415,400,895 mailboxes.

To effectively block individual e-mail addresses, spam must arrive from known senders (i.e., reused e-mail addresses). Although true years ago, spammers no longer reuse e-mail addresses, so blacklisting them is futile. Blocking domains and IP addresses assumes that all of the senders at a source produce spam. While true in some cases, such as when spammers purchase a domain for the purpose of spamming, they may also share a source with legitimate users. In this case, legitimate users suffer from unreliable e-mail while their ISP/domain appears in a blacklist, and depending on the list, it may take substantial time and effort to have the domain removed. The

[2] http://www.spamhaus.org/sbl/index.lasso

same delay occurs for adding domains to the blacklist; during the delay between the initial spamming and subsequent blocking many spams may leave the domain.

Blacklists see little use among end-users, who often opt for other techniques. Some ISPs continue to use them, and by blocking domains and IP addresses, they risk blocking legitimate e-mail.

2.3.2 Whitelisting

Users and specially-designed servers can implement whitelisting, a technique similar to blacklisting. Its users maintain a list of acceptable e-mail addresses and domains, possibly beginning with the existing entries in their address book. At the user level, this list likely resides in the e-mail client; at the server level, users may modify the list through a web-based user interface. When receiving mail, their mail client or server only accepts messages from senders on the whitelist and rejects all others.

For whitelisting to work, e-mail users must establish new contacts through another channel, such as in person, traditional mail, or by telephone. This technique has limited support for solicited advertisements, since users never know the sender's address, let alone even the domain, definitively before a message arrives. A message's `From:` header (i.e., the sender address) is also trivial to forge. A determined spammer can simply forge common e-mail addresses (e.g., `info@amazon.ca`) in an attempt to defeat the whitelist.

While blacklisting can prove ineffective, whitelisting is criticized for being too effective. Without extending the system, new contacts cannot establish communication with a user strictly through e-mail. This limitation makes the system only suitable for users with a small number of static contacts.

Challenge/response techniques (Section 2.3.3) can enhance this simple whitelisting technique. The modified version allows human users to establish contact strictly through e-mail.

2.3.3 Challenge/Response

Most spam originates from completely automated mailings. The spammer utilizes a specially-designed program, combined with a list of e-mail addresses, to produce copious amounts of junk. Challenge/response techniques prevent these messages from reaching users' in-boxes; they force unrecognized senders to prove that they are human. A user's client or mail server maintains a list of known senders, much like whitelists. When a message arrives from an unknown sender, the server does not deliver it to the recipient. Instead, the server replies to the message with a challenge for the sender. Only when the sender correctly answers the challenge does the sender

get added to the whitelist and the message delivered.

Challenge/response systems generally make two critical assumptions. First, they implicitly assume that all legitimate senders are human. Second, they assume that unknown senders care enough about the delivery of their message to answer the challenge. Unfortunately, legitimate e-mail senders are not always human. Loder et al. [40] point out that some challenge/response systems lock out potentially useful automated correspondence (e.g., account updates from online retailers). Depending on the exact challenge/response system deployed, it may take precautions to prevent such problems. As for the second assumption, human senders spend valuable time writing a message; this investment creates a desire to see it delivered, and they are likely to answer the challenge.

Some anti-spam advocates discourage the use of challenge/response for a variety of reasons. Gansterer et al. [18] describe some potential limitations of challenge/response techniques:

- two users of challenge/response systems cannot initiate communication with each other,
- automated systems or mailing lists cannot respond to a challenge,
- character recognition or pattern matching challenges are easy to bypass, and
- spammers may forge the e-mail address of a legitimate sender.

In a strictly challenge/response system, the first criticism is possible. Even if the server adds the recipient of outgoing mail to a whitelist, a challenge from the recipient could come from a different address, possibly from a different domain, and then be blocked. The second criticism states exactly why challenge/response is, to an extent, successful. If automated systems could respond to a challenge, then this technique could not prevent spam. A problem occurs with *solicited* automatic mailings because it is difficult to know the sender's address in advance without their cooperation. In response to the third point, although certain character recognition challenges are easy to solve [5, 7, 43, 44], suitably difficult challenges exist [6]. Finally, a spammer's ability to forge e-mail addresses requires knowledge of a whitelisted e-mail address. Some online retailers will appear in many whitelists (e.g., `info@amazon.ca`) and forging their addresses is trivial. With a strictly challenge/response-based system, address forgery is a threat.

Other anti-spam advocates even claim that challenge/response techniques "threaten to tangle users' email and legitimate Internet mailing lists into knots, while actually increasing the flow of spam-related traffic" [56]. Depending on the systems involved, such concerns may be legitimate.

2.3.4 Content Filtering

Researchers have suggested different types of content filters. These filters essentially analyze a message, and based on certain rules, decide whether a message is spam. A message detected as spam may be marked (e.g., by adding keywords to the subject) or deleted automatically. Simply adding keywords to the subject does little to fight spam; users must still manually delete the message. To truly combat spam, the filter should delete the messages that it determines to be spam.

The underlying assumption in filtering is that characteristics inherent to spam differentiate it from non-spam. Unfortunately, this is not always the case and content filters are prone to errors.

In spam filtering, two types of errors occur: false positives and false negatives. A false positive occurs when a filter classifies a legitimate message as spam. A false negative occurs when a filter misses spam, classifying it as legitimate e-mail. Of the two cases, false positive errors are worse. In this case, neither the sender nor receiver knows that the spam filter rejected the message (assuming the automatic deletion of messages categorized as spam).

The first primitive filters, heuristic filters, searched incoming messages for keywords in the message body and headers. The presence of certain keywords would indicate spam. As this filtering became more common, e-mail abusers responded with obfuscating words in their messages (e.g., Viagra might become V1@gr@). The filter maintainer would then need to revise its rules to account for the new characteristics of spam. With all of the tricks that spammers now employ, the maintainer could never keep up and these early filters are essentially useless.

Starting in 2002, a new breed of filters, those based on language-classification, replaced the early heuristic filters. Language-classification allows computers to write their own rules using machine learning. Users provide them with examples of both legitimate and spam messages, and using those messages, they train themselves. Consequently, they offer greater accuracy than heuristic filters. Zdziarski [61] states that "the very lowest level of accuracy that should be expected from any language classifier is around 99.5 percent." That said, the possibility of a false positive for an important e-mail could have dire consequences.

Language-classification filters are the most commonly used tool to fight spam. Both users and ISPs use them. Unfortunately, the constant threat of false positives reduces the reliability of the e-mail system.

2.3.5 Collaborative Filtering

Collaborative filtering relies on a community to detect spam. When a message arrives at a user, the mail client computes a special message digest for the message. Minor differences between messages produce a similar digest. The mail client compares the computed digest to a database of known spam before displaying the message to a user. Messages with a digest in the database may simply be deleted. If the digest does not exist in the database, the user receives the message. If the user decides that the message is spam, the user can submit its digest to the database, and when enough people submit a similar digest, other users need not view it.

The effectiveness of collaborative filtering is based on several critical assumptions:

1. spam is sent in nearly identical copies to many recipients,
2. suitable algorithms can detect these nearly identical copies, and
3. people have the same beliefs about what constitutes spam.

The first assumption generally holds true: for one spam mailing, a spammer will send nearly identical messages to possibly millions of recipients. With some digest functions, such as the MD5 message-digest algorithm [51], a small change in the input produces a large change in the resulting digest. These digest functions are not suitable for collaborative filters. Instead, researchers have proposed alternative functions where minor changes in the input do not significantly alter the output (e.g., [10]). These alternative functions meet the second assumption, at least to an extent. A fundamental problem with collaborative filters lies in the third assumption; different people classify spam differently. Said in another way, "one person's garbage is another's gold; your neighbors end up deciding what you read" [40].

2.4 Support Techniques

Section 2.3 presented some of the fundamental techniques in the fight against spam. Some other techniques, less specific to spam, can complement these fundamental techniques.

2.4.1 Throttling

Throttling involves limiting the bandwidth available to a single sender (network or host) communicating with the ISP's Simple Mail Transfer Protocol (SMTP) server. For inbound traffic, it reduces the ISP resources used by incoming bulk mail. For outbound traffic, it helps ISPs detect and prevent bulk mail from originating within their network.

One can use throttling or *tarpitting* techniques at the SMTP mail server or as a proxy to it. Many ISPs currently use throttling for both inbound and outbound traffic [61]. For some SMTP servers, such as postfix,[3] throttling is simply a configuration option. For others, throttling is available as a patch.[4] For those that do not support it, one can use a program like TarProxy.[5]

Throttling does not aim to prevent spam, but it may reduce it. Slowing a spammer's connection increases a spammer's resources (time and processing). At some point, it no longer becomes practical for a spammer to use a throttled SMTP server and that spammer will look elsewhere.

2.4.2 Mail Channels/Aliases

Many alias systems simplify the process where a single e-mail user obtains and manages multiple e-mail addresses. In terms of sharing e-mail addresses, individuals and companies have varying risk levels. For example, family members are unlikely to share an address with spammers whereas an untrustworthy company may. For that reason, one may want to use disposable e-mail addresses for risky contacts. If and when these disposable addresses become overrun with spam, a user can simply delete them.

Several companies currently specialize in providing this type of service. For example, mailexpire allows users to create free e-mail aliases.[6] Upon creation, users select a lifespan (12 or 24 hours, 1 or 2 weeks, or 1 or 3 months), but always maintain the ability to delete the alias at any time.

2.4.3 Sender Authentication

Current e-mail standards allow any Internet user to mail any e-mail user. Users without access to an ISP-based SMTP server can simply run one on their own computer. Since the sender does not need to use a valid address or send through an ISP-level mail server, no accountability exists between the Internet user and the e-mail recipient.

Several proposals aim to make senders more accountable [11, 58, 41]. Of these proposals, the primary two are the Sender ID Framework (SIDF) and Domain Keys Identified Mail (DKIM) [37]. The proposals provide mechanisms for verifying that e-mail originates from a recognized mail server. None of the proposals can fight spam alone; they only offer a method by which a sender can be verified.

[3] http://www.postfix.org/
[4] http://spamthrottle.qmail.ca/ and http://www.palomine.net/qmail/tarpit.html
[5] http://www.martiansoftware.com/tarproxy/
[6] http://www.mailexpire.com/

As e-mail traverses SMTP servers, each server prefixes trace information to the e-mail's headers. By the time that a message reaches the recipient's ISP, the headers show the path that the e-mail followed.

With SIDF, a recipient's ISP can verify whether a sender spoofed an e-mail's source. SIDF adds new entries to a server's DNS record. These entries describe the IP addresses from which legitimate mail can originate. To verify the sender, the receiver compares the originating IP address with the server's DNS record to determine whether the message passed through the listed server.

DKIM approaches the problem using cryptographic-based e-mail authentication. The system works using public-key cryptography. DKIM also adds new entries to the DNS record; in this case, the entry is the public key for the mail server. The sender's server (a) computes a secure hash (SHA-1 by default) of the message, (b) encrypts that hash value using the server's private key, (c) encodes that encrypted value in base-64, and (d) appends the resulting string to a new header field named "DomainKey-Signature". The receiver's server (a) performs a DNS lookup for the sender's server to obtain the sender's public key, (b) decodes the encrypted "DomainKey-Signature" string, (c) decrypts that decoded string, (d) computes a hash for the message, and (e) compares the two hashes. Matching signatures confirm that the mail originated at the purported domain and no one tampered with it.

In summary, both SIDF and DKIM verify that the e-mail originated at the claimed server. DKIM has an additional feature of detecting whether an intermediate server tampered with a message.

Many ISPs have implemented SIDF and DKIM. A study of SIDF adoption indicates that spammers are among the technology's biggest adopters [37]. Other techniques must be employed if either technology will help stop spam.

2.4.4 Authenticated SMTP

As originally defined in RFC 821 [49], the Simple Mail Transfer Protocol (SMTP) makes no secure distinction between authorized and unauthorized users. Authenticated SMTP allows service providers to limit access to their SMTP mail servers. Before being able to send a message, users authenticate themselves. Several possible authentication standards now exist, some as simple as requiring a username and password. If these credentials authenticate the user as a subscriber, then the user is allowed to use the server.

Many servers implement authenticated SMTP now. Before this technique became commonly used, spammers would find open SMTP servers (i.e., SMTP servers requiring no authentication). Using them, they could send their spam through another's

server. With authenticated SMTP, they need to either subscribe to the server or gain unauthorized access to a user's account.

2.5 Other Opportunities

All of the previous techniques use current technology to reduce or eliminate spam. Some other opportunities exist, both technological (micropayments) and not (education and legislation). This section explores these alternative approaches.

2.5.1 Micropayments

Micropayments represent a radical solution to the spam problem, one that requires fundamental changes to the foundations of e-mail. They recognize that the current cost of sending spam is low; spammers can generate a profit with a very low response rate. They aim to increase the cost of sending spam, while having minimal impact on legitimate e-mail messages.

In one proposed scheme, senders post a bond when establishing first contact with an e-mail recipient [40]. If the bond's value exceeds the minimum that the recipient accepts, the recipient will receive the message. After viewing the message, the recipient chooses whether or not to capture the bond based on the value of the message. For e-mail of high value, the receiver will not collect on the bond and instead return its value to the sender. For e-mail perceived as junk, the receiver collects on the bond as compensation for the time spent reading the message.

In another proposed scheme, Zmail, all e-mail senders attach virtual postage to their outgoing e-mail [36]. Unlike with traditional (non-electronic) mail, where postage supports the delivery, this e-postage goes to the recipient. The e-postage is sufficiently small to have little impact on non-bulk e-mail, while still deterring bulk mailers. The authors describe their system as a zero-sum e-mail protocol. Users collect postage on receiving and pay postage on sending; users with a send-to-receive ratio close to one-to-one experience little income and cost.

Both of these protocols require one or more intermediaries to support the fiscal requirements. Without significant infrastructure, their chances of adoption are small.

The concept of micropayments may make significant headway if a recent plan by AOL and Yahoo comes to fruition. Both companies partnered with Goodmail Systems, Inc. to take advantage of that company's e-mail certification and sender authentication technology.[7] Using CertifiedEmail technology, AOL and Yahoo plan to guarantee the delivery of certified messages for a cost [26]. AOL announced plans to charge one-quarter to one cent per message; in exchange, these messages will bypass its volume

[7] http://www.goodmail.com/

and content filters. Companies wishing to send certified message must first undergo the Goodmail accreditation process. The accreditation processing fee is US$199 before July 31, 2006, and US$399 afterwards.

2.5.2 Education

Spam will persist as long as its senders turn a profit. Estimates suggest that a spam mailing achieves at least a 0.005% response rate [15]. The cost of a mailing is relatively low, allowing this response rate to easily generate a profit. Technological solutions help reduce the response rate by blocking spam before it reaches someone capable of ordering from it. For spam that reaches people, education can further help to reduce the response rate.

If education can sufficiently lower the response rate, spammers will begin to lose money. Subsequently, the quantity of spam will decrease. ISPs, along with governments and the media at large, can help educate e-mail users. These authorities can stress the importance of ignoring spam, rather than ordering from it.

2.5.3 Legislation

To efficiently use legislation, targeting the largest spam producers will have the greatest impact. One organization, the Spamhaus Project, aims to help. Their "About Us" page describes them:[8]

> Spamhaus is an international non-profit organization whose mission is to track the Internet's Spam Gangs, to provide dependable real time anti-spam protection for Internet networks, to work with Law Enforcement Agencies to identify and pursue spammers worldwide, and to lobby governments for effective anti-spam legislation.

They state that 200 known spam operations are responsible for 80% of e-mail's spam. Furthermore, they publish a Register of Known Spam Operations (ROKSO), containing spammers terminated from a minimum of three ISPs for Acceptable Use Policy (AUP) violations. At the time of writing, this list contains 168 entries, 14 from Canada.[9]

International borders are a roadblock to the legislative approach to fighting spam. The Internet carries spam freely across borders, while legislation's jurisdiction usually stops at the border. Without international cooperation, a country can only use legislation effectively against spammers within the country.

Although Canada does not have laws that explicitly target spam, much spam is already illegal under a variety of Canadian laws. For example,

[8] http://www.spamhaus.org/organization/index.html
[9] http://www.spamhaus.org/rokso/index.lasso

- The Personal Information Protection and Electronic Documents Act (PIPEDA) [21]:
  PIPEDA defines personal information as "information about an identifiable individual, but does not include the name, title or business address or telephone number of an employee of an organization," a broad enough definition to include a personal e-mail address. For commercial activities, which includes much spam, PIPEDA protects collection, use, and disclosure of personal information. When spammers sell e-mail addresses, they *disclose* protected personal information without permission. When spammers send spam, they *use* protected personal information without permission. Hence, both of these activities are illegal under PIPEDA.
- The Competition Act [19]:
  This act includes provisions for deceptive marketing practices, something common in much spam.
- The Criminal Code of Canada [20]:
  Many types of spam are already illegal under this act [31]. For example, spams requesting fees in exchange for lottery prize winnings and the Nigerian/West-African money transfer scams are fraud under Part X (Fraudulent Transactions Relating to Contracts and Trade). Relaying spam through open SMTP servers without permission may constitute unauthorized use of a computer in Part IX (Offences Against Rights of Property).

As of May 2005, three spam complaints were settled under PIPEDA and one under the Competition Act [32].

While much spam is illegal, enforcement of existing laws remains difficult. When the Canadian Task Force on Spam issued its final report in May 2005 [32], it identified key challenges for enforcement:

1. limited resources and competing priorities;
2. frequent lack of the specialized technical expertise required to track down, investigate, and prosecute spammers; and
3. existing enforcement powers have not yet been used, and the legislative tools to attack particular elements of spam are either too uncertain in their application or simply missing.

The task force strongly believes in the need to strengthen the enforcement process.

The task force identified two gaps in the current legislation [32]. First, enforcement agencies should be sufficiently certain of a victory when prosecuting spammers. The current laws do not provide that certainty for many of the methods and means used by spammers. When using the current laws, many spamming and spam-related activities

Table 2.4: Implied definitions for *spam* particular to the technique for eliminating it.

Technique	Definition
blacklisting	Spam is a message sent from a known sender whose previous message caused the receiver to explicitly deny access to that sender.
whitelisting	Spam is a message sent from an unknown sender.
challenge/response	Spam is a message with no human contact at the sending end who would be interested in the fate of its individual instances.
heuristic filters	Spam is a message whose textual component includes words or phrases anticipated to occur in spam.
language-classification filters	Spam is a message fitting certain patterns determined by a reasonably large corpora of messages collectively categorized as spam by human recipients.
collaborative filters	Spam is a message sent in (nearly) identical copies to a significantly large number of different recipients who collectively classify it as spam.
micropayments	Spam is a bulk mailing where, on average, the value of each individual instance is less than the micro-expense of sending the message.

fall outside their boundaries. Second, the task force does not believe that existing laws provide an effective deterrent effect. They believe that stronger penalties will deter serious spammers.

2.6 Defining Spam

Many definitions exist for spam. Some definitions for spam include "unwanted e-mail," "unsolicited commercial e-mail," "junk e-mail," and "unsolicited bulk e-mail" [61]. These definitions are often used informally. The Oxford English Dictionary (Online) defines spam as "to flood (a network, esp. the Internet, a newsgroup, or individuals) with a large number of unsolicited postings, or multiple copies of the same posting. Also *intr.*: to send large numbers of unsolicited messages or advertisements" [46]. Spamhaus provides a more technical definition,[10] stating that an electronic message is spam if (a) "the recipient's personal identity and context are irrelevant because the message is equally applicable to many other potential recipients," and (b) "the recipient has not verifiably granted deliberate, explicit, and still-revocable permission for it to be sent." Researchers, on the other hand, implicitly define spam differently depending on their technique for eliminating it. Table 2.4 suggests some possible definitions for each of the fundamental techniques to eliminate spam.

Our technique takes advantage of several of the existing techniques: whitelisting, challenge/response, and aliasing. Our definition for spam begins with that of chal-

[10] http://www.spamhaus.org/definition.html

lenge/response, "a message with no human contact at the sending end who would be interested in the fate of its individual instances," and then adds "unless explicitly solicited by the recipient." We create a flexible system with provisions for solicited advertisements and other automated mailings. Unlike strict whitelisting, we acknowledge that people often want to initiate first contact through e-mail; some unknown senders write legitimate e-mail. Finally, our aliases should not be thought of as disposable, as with strictly alias-based systems.

Chapter 3

Prior Work

The work in the next chapter builds on two topics: challenge/response techniques and mail channels. Both are research areas in their own right and worthy of a detailed explanation. This chapter introduces them and summarizes key points in their development, easing the transition into our work.

3.1 Challenge/Response Techniques: Theory to Implementation

No definitive resource describes the history of challenge/response, particularly before its use with e-mail. This section pieces together a brief history of its development, beginning with an initial idea and ending with its implementation in systems that eliminate junk e-mail.

3.1.1 Initial Idea

Challenge/response is an idea similar to passwords. With passwords, a prompt asks for secret information, something known by one or more individuals. By answering the prompt, an individual proves their inclusion in that group. With challenge/response, a prompt asks a question that most humans can answer. A correct response proves that a human answered the challenge.

3.1.2 Pricing via Processing

In August 1992 at the 12th Annual International Cryptology Conference (CRYPTO'92), Cynthia Dwork and Moni Naor [13] introduced a system to help combat junk e-mail. In their system, a user's computer must answer a challenge when sending e-mail; more specifically, sending a message requires that the sender's computer first solve a moderate computational problem. Their work refers to this problem as a *pricing function*. The pricing function's result is specific to the message content, the current time, and

the destination address to prevent reuse of the solution. A user's client attaches the solution to each message that it sends.

All users of the system use the same pricing function. A *pricing authority* determines the pricing function, and all users agree to obey that pricing authority.

The pricing function has a shortcut, a concept similar to a trapdoor one-way permutation [13]. Given the shortcut, the pricing function is easy to evaluate. In addition to the pricing authority, the system proposes that a number of trusted agents know this shortcut function. Using the shortcut, both the pricing authority and these agents could send bulk e-mail (for a fee) without incurring the computation cost of the challenge. This fee could go towards the authority for providing the service as well as to the recipients of the message. If the shortcut becomes public knowledge, the pricing authority can simply replace the pricing function.

For users who frequently correspond, each user can have a "frequent correspondent list," something like a whitelist. Messages from people on this list would not require a solution to the challenge. Using this list, the system can support mailing lists and solicited bulk mailings without requiring those senders to pay a fee.

3.1.3 Three-Way Handshake

On September 13, 1996, Naor follows up on the idea of pricing via processing and introduces the three-step system now used in many modern challenge/response systems to prevent junk e-mail [45]. The paper describes four desirable properties for the Turing test (i.e., the challenge):

1. generating many instances of the problem, together with unambiguous solutions, is easy;
2. solving a given instance is effortless for a human, yielding few errors;
3. computing a solution to an instance of the problem fails a non-negligible fraction of the time, even with knowledge of the instance generation method; and
4. displaying an instance specification is brief (i.e., little communication bandwidth and screen area).

Further to these properties, the paper lists a few types of problems including recognizing genders, understanding facial expressions, locating body parts, deciding whether a person is nude, understanding naïve drawings, understanding handwriting, recognizing speech, filling in blanks/words, and disambiguating text.

In addition to providing much groundwork for types of challenges, the paper explores the use of challenge/response to fight junk e-mail. It describes the scenario where (a) a sender sends an e-mail, (b) the receiver responds with a challenge, and

(c) only when the sender correctly answers the challenge does the receiver actually see the message. Compared to the initial Dwork and Naor [13] proposal, this system uses the exchange of three e-mail messages, rather than one, between the sender and receiver. At the same time, its implementation no longer requires the cooperation of all e-mail users.

3.1.4 Implementations

The first implemented and released system to follow Naor's paper was the NAGS (Netizens Against Gratuitous Spamming) Spam Filter on September 15, 1996.[1] In a newsgroup posting on September 27, 1996,[2] the author of the filter (Ian Leicht) describes some of its features [38]:

1. blacklist capability: "[you] configure the script with a list of sites that you don't want to hear from";
2. whitelist capability: "[you] can also specify exceptions (via regular expressions) of people that you do want to allow from the site"; and
3. *challenge/response functionality*: "if [the senders] include a special message in the subject that the reject letter tells them about they can also get word to you."

With the third point, NAGS implemented the three-way challenge/response technique described by Naor [45]. In this particular instance, a determined spammer could easily overcome the challenge, but at the same time, it provided a start for subsequent systems to follow.

Soon after NAGS included the challenge/response capability, the majordomo mailing list software also added it.[3] Starting with version 1.94, released in October 1996, majordomo added a subscription confirmation feature to fight spam. This addition was part of several aimed at preventing spam. When using this feature, "a user must send back a one-time key for a subscription request to succeed" [57].

In 1997, two further systems supported challenge/response. In the winter/early spring after the NAGS release, Pawel Gburzynski developed his RabidFire filter, also capable of challenge/response. The RabidFire filter is still available online.[4] Soon after that in June 1997, Brad Templeton developed a (never-released) challenge/response filter known as Viking-12.

At this point, challenge/response techniques to prevent junk e-mail were well underway. Several companies have since tried to commercialize on the challenge/response

[1] http://www.nags.org/
[2] news.admin.net-abuse.misc
[3] http://www.greatcircle.com/majordomo/
[4] http://sheerness.cs.ualberta.ca/ pawel/RabidFire/

paradigm, e.g., MailBlocks Inc., Spam Arrest, LLC,[5] and VanquishLabs.[6] The former two companies became involved in a legal dispute when MailBlocks tried to enforce two patents [27, 8] under its control.

3.1.5 Patents

MailBlocks, Inc. owns two patents [27, 8] that pertain to the above challenge/response techniques. The abstract for patent #6,199,102, titled "Method and System for Filtering Electronic Messages," filed on August 26, 1997, and issued on March 6, 2001, states:

> The present invention provides a system and method for filtering unsolicited electronic commercial messages. A system and method according to the present invention for screening out unsolicited commercial messages comprises the steps of receiving a message from a sender, sending a challenge back to the sender, receiving a response to the challenge, and determining if the response is a proper response.

The abstract for patent #6,112,227, titled "Filter-In Method for Reducing Junk E-Mail," filed on August 6, 1998, and issued on August 29, 2000, states:

> A method is provided for preventing the delivery of unwanted electronic mail messages to the destination client. An original electronic mail message is first received from a source client at a destination server. Next, a reply electronic mail message is sent from the destination server to the source client requesting the source client to complete a registration process to register the source client's electronic mail address with the destination server. The original electronic mail message is only sent from the destination server to the destination client when the source client properly registers the source client's electronic mail address.

Both of these patents describe, to some extent, the many previously-mentioned challenge/response systems. Christopher Alan Cobb filed the first patent application almost a year after the first publicly available software to implement its ideas.

In 2003, MailBlocks started enforcing these patents. Using them, it filed several suits against competing companies (e.g., Spam Arrest, LLC and EarthLink, Inc.[7]). The suit against Spam Arrest was eventually settled amicably, as described in a press release by Spam Arrest. The terms of that settlement remain confidential. After much

[5] http://www.spamarrest.com/
[6] http://www.vanquish.com/products/products_personal_antispam.shtml
[7] http://www.earthlink.net/

searching about the suit against EarthLink, we cannot find any information as to its current status.

3.2 Mail Channels: Theory to Implementation

Within published research, one can follow some of the fundamental steps in the creation of mail channels. Like challenge/response, they began with an idea, and they are now implemented in commercial systems.

3.2.1 Initial Idea

In 1982, Peter J. Denning [12], then President of the Association for Computing Machinery (ACM), described "the receiver's plight" or the "constant barrage of information" that he experienced. While part of this incoming information was traditional (e.g., regular mail and telephone calls), a substantial portion was electronic mail. His e-mail in-box received a "riptide of normal business mail" and a "tidal wave of electronic junk mail." To solve the problem, he envisioned multiple e-mail paths (or mail channels), each for a specific type of e-mail. In addition to a general path, specific paths might include *urgent*, *certified*, and *personal*. The mail system would always deliver messages arriving on specific paths, while filtering messages on others. He observed that existing organizations already meet these requirements for (non-electronic) communications. He suggested that the research community study these traditional paths when developing a solution to junk e-mail.

3.2.2 Andrew Messaging System

Although Denning introduced the idea of a mail channel, he made no attempt to implement it. A 1991 paper by Borenstein and Thyberg [3] describes possibly the first implementation while explaining the Andrew Messaging System (AMS).

The Andrew project's goal was to build a realistic prototype of a university-wide distributed computing environment. It consisted of three main parts: the Andrew File System, the Andrew Toolkit, and finally the Andrew Messaging System (AMS). Carnegie Mellon University introduced the first public Andrew workstation lab in early 1986, before the creation of AMS. Soon after, when Andrew accounts were made generally available, the support e-mail address (Advisor) became overwhelmed with mail. In late 1986, the first release of AMS appeared on campus. The Advisor *service*, built using AMS, aimed to help balance this volume across a team of support staff.

An early version, Advisor II, searched a message's subject for keywords in an attempt to channel e-mail to the appropriate support staff. Unfortunately, users rarely included appropriate keywords and messages were seldom channelled correctly. A later

version, Advisor III, replaced subject keywords with extensions to the recipient e-mail address. An address's form became `user+extension`, where *extension* specified a mail channel.

3.2.3 Hall

Recall that Denning [12] described the idea of channelled communication and that Borenstein and Thyberg [3] implemented a method for creating such channels. To the best of our knowledge, Hall [23, 24] was the first to implement mail channels for preventing spam. His system's syntax differs from AMS while still modifying the recipient's address; an address's form is `user-extension-@host`. Extensions are pseudo-random text (cryptographically secure BlumBlumShub [2]) to reduce the probability of guessing an open channel. To manage the complexity of many channels, he developed a Personal Channel Agent (PCA), which essentially acts as an e-mail proxy. Sitting between the user's mail client and the mail server, the PCA's primary function is rewriting addresses (i.e., ensuring that incoming and outgoing messages use appropriate extensions). Recognizing that users have different types of contacts, his system introduced three channel classes (i.e., policies). For contacts that a user only sends to and never receives from, he introduced a *send only* channel. For communication with a particular contact, a *private* channel maintains, and only accepts mail from, a list of known correspondents. Finally, he created *public* channels to be accessible to anyone, much like a traditional e-mail address.

The PCA includes an administrative interface to manage channels. It provides the functionality to manually open, close, create, delete, or switch channels. No component in the system automates the creation of mail channels. Without automation, the need to manually perform this action can be time consuming, particularly for a large number of channels.

When the system rejects a message on a channel (e.g., an unauthorized sender on a private channel), it bounces a "no permission" message to the sender. Suppose User_a uses Hall's system and regularly communicates with User_b through a private channel. User_b has a friend, User_c, who wishes to communicate with User_a. User_b gives User_c the private address for User_a. Unfortunately, this address is useless to User_c since all messages sent to the address will result in a "no permission" message. Hall's system introduces some complications when contacts wish to exchange addresses.

The system depends on the secrecy of the public (open) channels. When an abuser compromises a public channel, switching channels involves contacting all of the legitimate senders on the compromised channel. The system attempts to automate channel switching, but nevertheless hassles a user's contacts.

3.2.4 Single-Purpose Addresses

Hall's extensions are essentially keys into a database; upon receiving a message, the server can locate the policy using the extension. Later work by John Ioannidis introduced a new technique of encapsulating policies within the extensions themselves. By doing so, he removed the dependency on a database.

Single-Purpose Addresses (SPAs) [34] use the same syntax as AMS. However, it can be changed easily on a per user basis by modifying a single text file. To prevent tampering and keep the policies private, the system encrypts the extensions.

The SMTP [35] standard imposes a 64-character maximum length on an address's username (or local-part) portion. This maximum bounds the expressiveness of the policies since they become part of the local-part portion of the address. In the current implementation, the SPAs contain an expiration date and a sub-string of the sender's domain name.

The SPA implementation requires that users manually create each new address. To create an address, a user must run a script (`spagen`), passing it arguments for the expiration date and domain name. The program formats those arguments appropriately, encrypts them, and then converts them to base-32. Manually creating addresses places a burden on the SPA user, similar to the burden in Hall's system.

Their choice to use base-32 rather than base-64 was based on an error. They supported their choice of base-32 stating that "email addresses are case-insensitive." While true for the domain portion, the local-part portion is case-sensitive. As far back as the now obsolete RFC 821 [49] states that for "some hosts the user name is case sensitive, and SMTP implementations must take case [sic] to preserve the case of user names as they appear in mailbox arguments. Host names are not case sensitive." The current RFC 2821 [35] states more clearly that the "local-part of a mailbox MUST BE treated as case sensitive." Using base-64 instead of base-32 would reduce the extension from 26 to 22 characters.

With a 26-character extension, SPAs have only a limited audience. The description of SPAs states that "no one is expected to type in such addresses" [34]. Thus, we can conclude that this scheme is not for general use. The addresses' lengths make them appropriate for automated systems and not business contacts, family members, or friends.

The fact that SPAs include an encoded expiration date can also cause problems. A contact may initially appear very short term, but later become longer term. In this case, a new e-mail address must be issued if the SPA user wishes to maintain contact.

3.2.5 Implementations

The two original implementations by Hall and Ioannidis no longer appear to be in use. Hall's channel technology is currently used in ZoEmail,[8] a company for which he is a consultant and adviser. No known companies currently make use of Ioannidis's proposal.

Several companies currently offer rudimentary mail channels in the form of aliases (e.g., Spamex[9] and Sneakemail[10]). Both companies allow users to manually create aliases through a web interface; when e-mail arrives at an alias, the server forwards it to a permanent e-mail address. They both rewrite the `From:` header field so that replies will pass through the same service and it can replace the permanent address with the alias.

3.2.6 Patents

AT&T Corp., Robert J. Hall's employer, owns a patent [25] that describes Hall's system. The abstract for patent #5,930,479, titled "Communications Addressing System" states:

> A system and method is provided for sending and receiving authorized messages from a sender to a recipient in a network. The method and system makes use of a channelled address to send the message from the sender to the recipient. The channelled address comprises a common address portion that indicates the identity of the recipient in the network and a channel identifier portion for verifying that the message is authorized for delivery to the recipient.

Unlike the two challenge/response patents, AT&T has not publicly enforced this patent.

3.3 Hybrid Solutions

The Spam Free e-Mail (SFM) service, the motivation for this thesis, uses a hybrid approach: it uses challenge/response techniques to automate the creation of mail channels. Prior work by Eran Gabber et al. [17] uses a similar approach; however, they presented merely the idea and not an implementation. Given the similarity between their work and ours, we will describe their idea in moderate detail before introducing our work in the next chapter.

In traditional e-mail use, a user has only a handful of e-mail addresses. For example, two e-mail addresses allow a user to separate work-related and personal e-mail. If

[8] http://www.zoemail.com/
[9] http://www.spamex.com/
[10] http://www.sneakemail.com/

one of these addresses falls into the wrong hands, it may become susceptible to abuse. If many contacts use the address, simply changing addresses is not always a viable option. Gabber et al. introduce a system where each e-mail user has many addresses, the now familiar concept of mail channels.

In their proposal, each user has a *core* address, similar in appearance to a traditional e-mail address (e.g., `user@host`). Users never receive messages sent to their core address. Instead, each user has multiple extended addresses, each derived from the core address that use the same syntax as the extensions in AMS. Messages arriving at an extended address *may* be delivered to the user depending on several conditions.

When a message arrives at an extended address, it must meet certain conditions before it is delivered to the user. Extended addresses can be (1) valid, (2) bound, or (3) revoked. Valid extended addresses will receive e-mail from any sender; no senders are rejected. Bound ones will only accept e-mail from recognized senders; unrecognized senders are rejected. Finally, revoked addresses do not accept mail; all senders are rejected.

When senders write to a core address, they can automatically obtain an extended address by completing a handshake protocol. Gabber et al. describe this handshake in general terms: (1) User_A sends a message to User_B, (2) User_B requests that User_A incur a "cost," (3) User_A incurs a "cost" and sends proof to User_B, (4) User_B creates a extended address and sends it to User_A, and (5) User_A resends the message using the new extended address. They further describe some desirable properties for this "cost," but they fail to discuss a possible implementation of it.

A user of the system can implicitly or explicitly generate new extended addresses. When a user writes to a new contact, the system automatically generates an extended address for the recipient (a single person or a group of people). Users can also manually generate extended addresses using a web-based interface. These manually generated addresses are appropriate for registering at web sites, subscribing to mailing lists, or other cases where the sender cannot complete the handshake process.

Chapter 4

Spam Free e-Mail (SFM)

The Spam Free e-Mail (SFM) service solves weaknesses of both the mail channel and challenge/response paradigms when those techniques are used alone. Mail channels generally have a high administration cost as users must manually create mail channels. Using challenge/response techniques to prevent abuse, SFM automatically creates mail channels. A strictly challenge/response-based system will not accept solicited automatic mailings as the sender cannot answer the challenge. Using mail channels, users can provide unchallenged paths to ensure the arrival of solicited bulk mailings. SFM combines these complementary technologies to produce a system more flexible than either one alone.

To gently introduce the system, we first present a user's perspective. After this brief introduction, the remainder of this chapter describes the technical details behind SFM.

4.1 User's Perspective

SFM is backwards compatible with the existing e-mail infrastructure. Users of the system are (a) subscribers and (b) non-subscribers trying to contact subscribers. In both cases, these users see different aspects of the system. For that reason, we describe each user type in turn.

4.1.1 SFM Subscribers

New users to SFM must first subscribe. After subscribing, they must configure our service and their own e-mail client; both require only a few steps, after which the service is ready to use. In many cases, SFM will then operate transparent to the user.

Subscribing

Subscribers first encounter SFM's web-based user interface. This interface supports many web browsers, e.g., Internet Explorer, Safari, Mozilla, Firefox, and even `Lynx`.

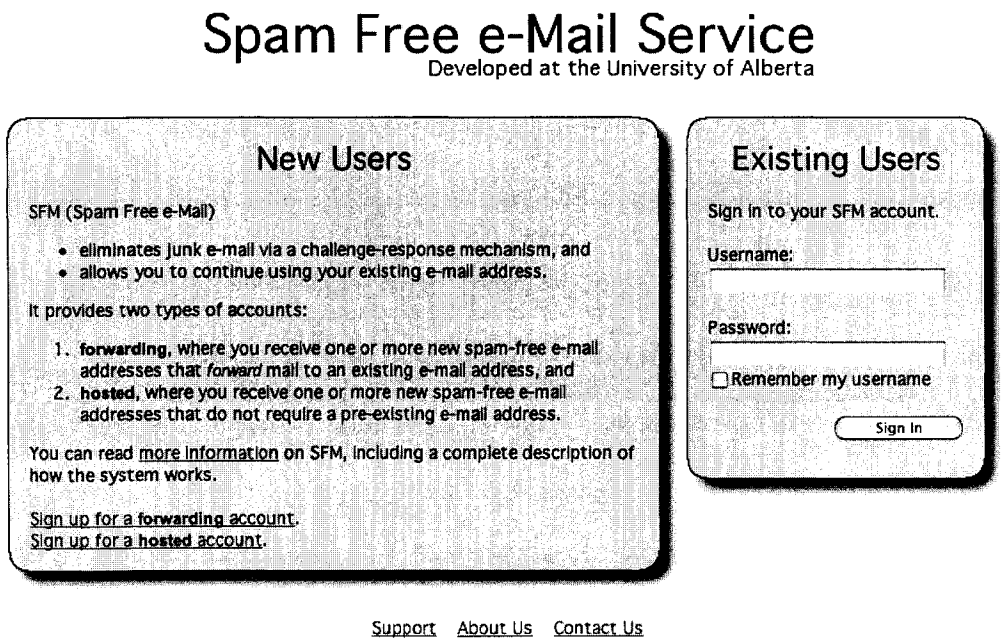

Figure 4.1: The SFM welcome page as viewed in Apple's Safari version 2.0.2.

By directing their web browser to the server's address users can (a) discover and learn about the service, (b) subscribe to the service, and (c) log in to an existing account.[1] Figures 4.1 and 4.2 show the welcome page in graphical and text-only web browsers, respectively.

After arriving at the welcome page, new users can create an account. The system supports two account types: *forwarding* and *hosted*.

Forwarding accounts require an existing e-mail address. When users subscribe, they provide their existing, permanent address, which later becomes their username. When SFM accepts a message destined for this type of user, the server forwards it to this address. Users of this type can subscribe through our web site (Figure 4.3). The subscription process collects minimal information about the user (Table 4.1).

After submitting the form to request a forwarding account (Figure 4.3), SFM verifies the forwarding address. It sends a *secret code* to this address. When users first attempt to log in to perform the initial setup, SFM requests this code.

Hosted accounts do not require an existing e-mail address. Instead, the subscriber receives a new e-mail address at the SFM server's domain. When SFM accepts a message for a hosted user, it deposits it in the user's local mailbox. The user can later retrieve these messages using any standard e-mail client. Unlike forwarding accounts, people cannot subscribe to hosted accounts online. Instead, the current SFM version requires that an administrator create these accounts. These accounts have the same

[1] https://sfm.cs.ualberta.ca/

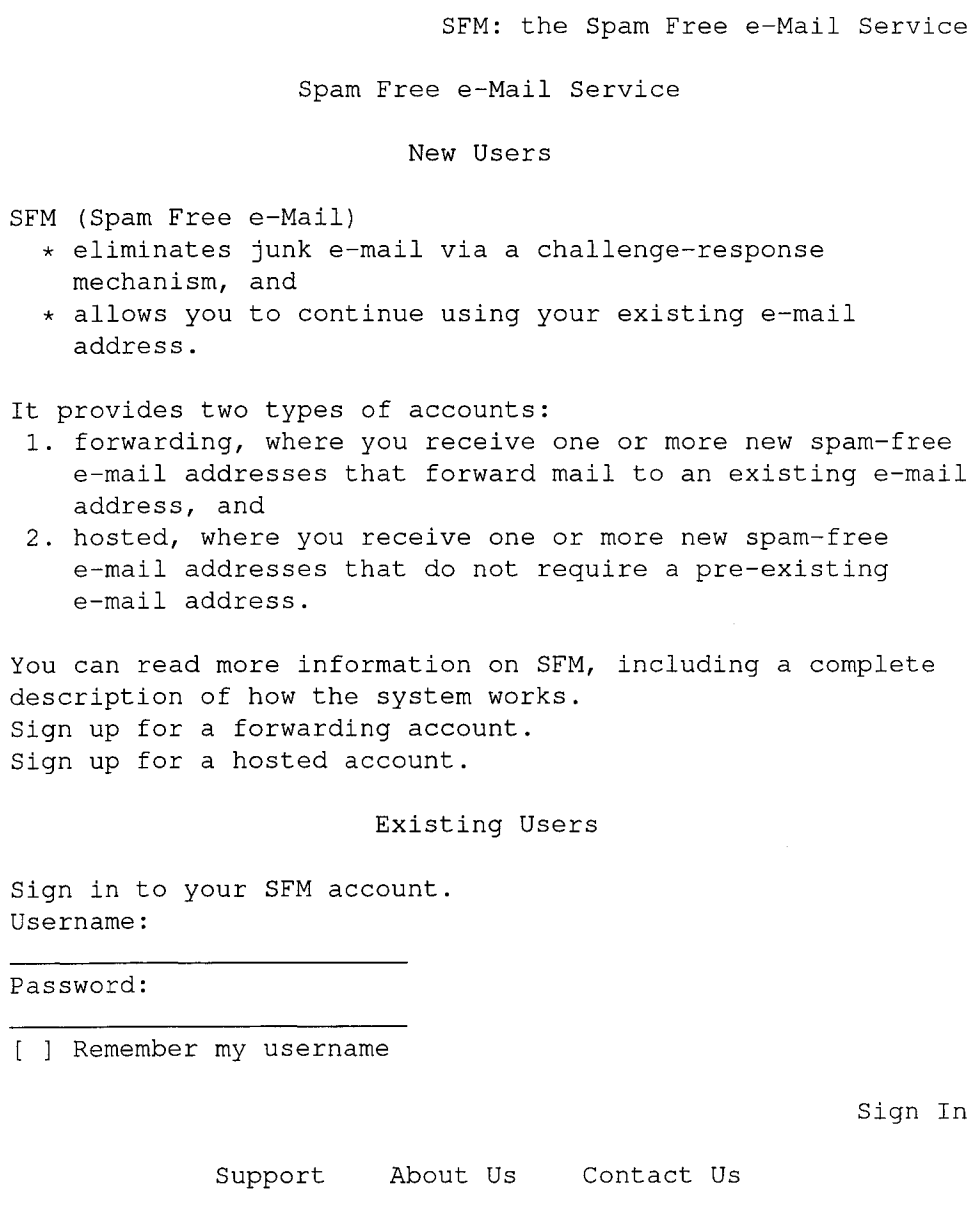

Figure 4.2: The SFM welcome page as viewed in Lynx version 2.8.4rel.1.

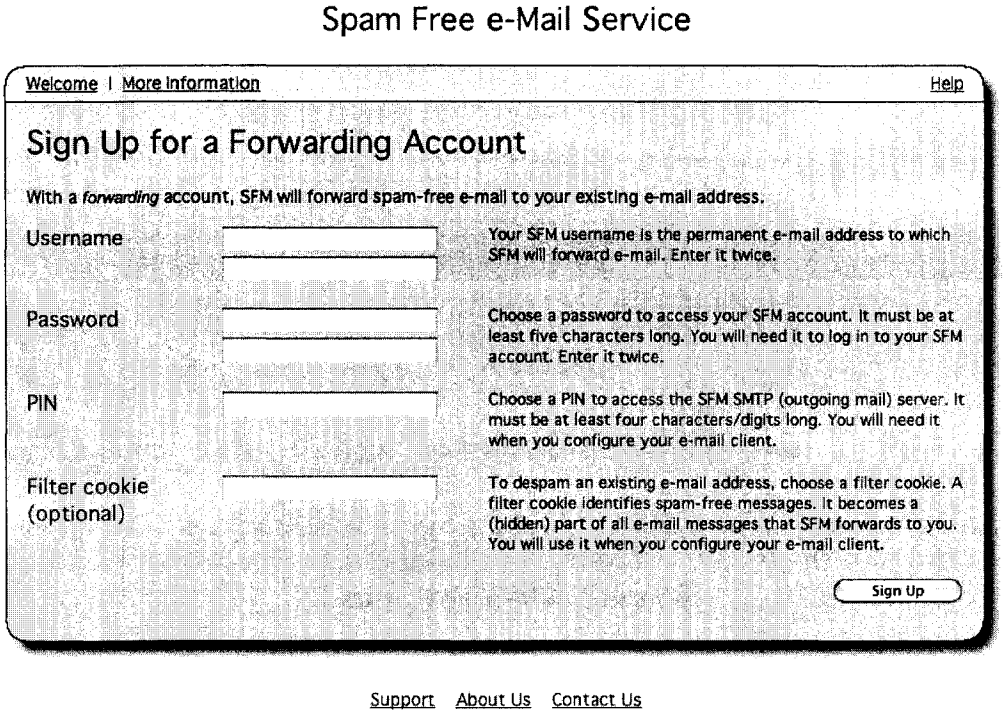

Figure 4.3: The SFM sign-up page as viewed in Apple's Safari version 2.0.2.

Table 4.1: Information collected about new users when they subscribe to our service.

Item	Description
Username	The username (i.e., a permanent e-mail address) is the address to which SFM forwards accepted mail.
Password	The password allows users to log in to their account through the web site. After the initial set up, users need not regularly log in to their account.
PIN	The PIN authenticates users for the SFM outgoing mail (SMTP) server.
Filter cookie	The filter cookie differentiates SFM-processed from unprocessed e-mail at the permanent e-mail address. It becomes a (hidden) part of each message that SFM forwards in an SFM-specific `X-Filter-Cookie` header field. Using it, SFM can despam an existing e-mail address.

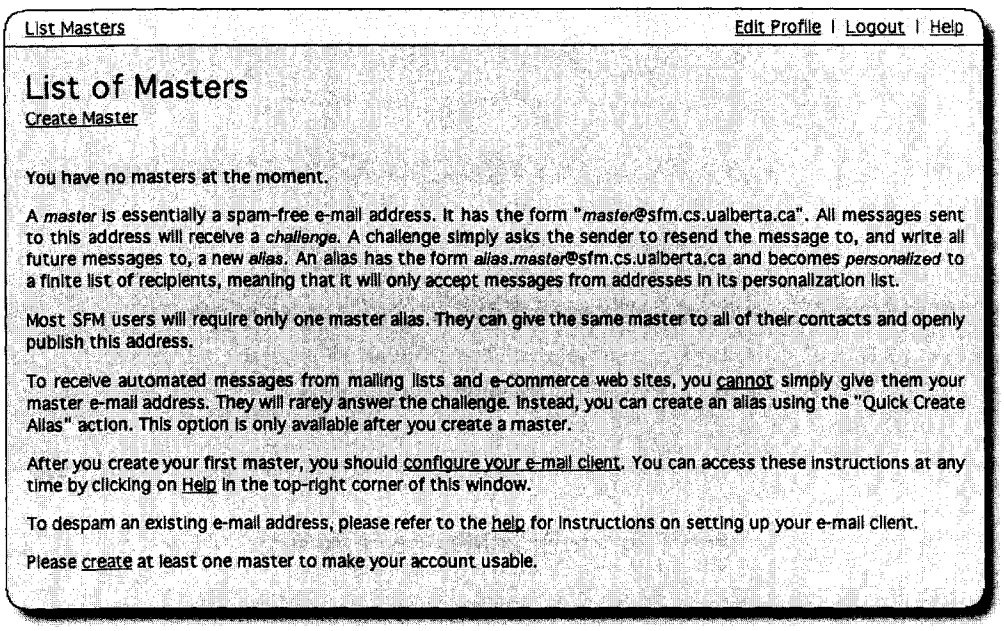

Figure 4.4: The main page after first signing in to a new account as viewed in Apple's Safari version 2.0.2.

attributes as forwarding accounts (Table 4.1), but the filter cookie is unnecessary.

Initial Setup

After creating a new account, users must configure it. After first logging in to the web-based user interface, the system asks new users to create one or more new e-mail addresses that we call *masters* (Figure 4.4). These addresses are publishable and immune to much e-mail abuse. Users never receive messages sent to their masters. Instead, SFM replies to these messages on behalf of the subscriber. The reply asks the sender to prove their humanity by resending their original message to a new e-mail address that we call an *alias*. This alias is a mail channel automatically created by SFM in response to a message arriving at a master. When the sender answers the challenge and resends the message to the alias, SFM delivers it to the user.

Recall that users must create one or more masters. Each master has several attributes: master name, display name, comment, expiration, and open time (Table 4.2). The SFM subscriber specifies these attributes when creating the master and can modify everything but the master name at a later date.

After users create their first master, they must update their e-mail client's configuration. Our system supports many e-mail clients (e.g., Outlook, Apple Mail, Thunder-

Table 4.2: Attributes of a *master*.

Attribute	Description
Master name	A master address looks like a traditional e-mail address. The master name is the username portion of the address, as in *master@host*.
Display name	The Internet message format standard [50] allows a user-friendly name to accompany each e-mail address. This name is known as the *display name*, and e-mail clients often display it instead of the e-mail address. Generally, it is the full individual or company name. If unspecified, SFM uses the display name from the user's profile.
Comment	The optional comment is text associated with a master. Comments may remind the subscriber of the master's purpose.
Expiration	This attribute applies to aliases derived from the master (discussed later). Acceptable values include a number of days or the string "infinite".
Open time	This attribute applies to aliases derived from the master (discussed later). Acceptable values include a number of days or the string "infinite".

Table 4.3: General configuration changes required for forwarding accounts.

Setting	Original value	New value
E-mail address	*username*`@`*host*	*username*`+`***PIN***`@`*host*
Outgoing mail server		
Address	ISP SMTP server	SFM SMTP server
Port	25 (default)	9025 (default)
Secure Sockets Layer (SSL)	varies	disabled
Authentication method	varies	none

bird, and even `Pine`).

When users obtain a forwarding account, they make only minor changes to their existing e-mail client configuration. In particular, they (a) change the outgoing mail server settings: address, port, Secure Sockets Layer (SSL), and authentication method and (b) append their PIN to their e-mail address. All of the incoming mail settings remain the same, since SFM forwards all accepted mail to this existing e-mail address. The SFM mail server uses the PIN in the e-mail address to authenticate senders. Table 4.3 summarizes the changes required to use SFM.

To describe the configuration changes concretely, consider Table 4.4. Notice that the only changes occur with the outgoing mail server and the e-mail address.

When users obtain a hosted account, they must configure their e-mail client. In particular, they (a) create a new mail account in the e-mail client, (b) set the outgoing mail server settings: address, port, Secure Sockets Layer (SSL), and authentication method, as required for the SFM SMTP server, (c) set the e-mail address to include their PIN, and (d) set the incoming mail server settings: address, port, SSL, and au-

Table 4.4: Concrete example of the configuration changes required for forwarding accounts. The outgoing SFM mail server is at sfm.cs.ualberta and uses port 9025. Nick's PIN is 1234.

Setting	Original value	New value
Full name	Nicholas Boers	Nicholas Boers
E-mail address	nboers@ualberta.ca	nboers**+1234**@ualberta.ca
Incoming mail server		
Address	pop.srv.ualberta.ca	pop.srv.ualberta.ca
Port	995	995
SSL	enabled	enabled
Authentication method	password	password
Username	nboers	nboers
Password	*****	*****
Outgoing mail server		
Address	smtp.srv.ualberta.ca	**sfm.cs.ualberta.ca**
Port	587	**9025**
SSL	enabled	**disabled**
Authentication method	password	**none**

Table 4.5: General configuration settings required for hosted accounts.

Setting	Value
Full name	User's name
E-mail address	*username+PIN@SFM host*
Incoming mail server	
Address	SFM POP3 server
Port	9110 (default)
SSL	disabled
Authentication method	password
Username	*username*
Password	*****
Outgoing mail server	
Address	SFM SMTP server
Port	9025 (default)
SSL	disabled
Authentication method	none

thentication method, as required for the SFM POP3 server. Table 4.5 summarizes the configuration.

After making the appropriate changes to the SFM configuration, the system is ready to use.

Using the System: Masters and Aliases

Recall that Section 4.1.1 introduced *masters* and *aliases*. Masters are publishable addresses, immune to much e-mail abuse. SFM automatically creates aliases (mail channels) in several instances, such as when it receives mail on a master.

When SFM automatically creates an alias, it uses the master as a template. Recall that masters have several attributes (Table 4.2). Aliases share some of them (i.e., the

Table 4.6: Attributes of an *alias*.

Attribute	Description
Alias name	An extension to the username portion of an e-mail address, which will have the form *alias.master@host*. The alias and master name combination is a mail channel.
Master name	The username portion of the new e-mail address, which will have the form *master@host*.
Display name	The name displayed along with this new e-mail address. Generally, the display name is the full name of an individual or a company name. If unspecified, SFM uses the display name that is part of the profile.
Comment	The optional comment is text associated with an alias. Comments might remind the subscriber of the alias's purpose.
Expiration	Acceptable values include a date and the string "infinite". SFM will delete the alias after this date.
Open time	Acceptable values include a date and the strings "closed" and "infinite". If a date, SFM will change this value to "closed" after this date. If "infinite", SFM will never automatically close the alias.
Personalization	E-mail addresses and domains accepted by this alias. SFM populates this list while an alias is open and rejects senders not on it when an alias is closed.
Blocked senders	E-mail addresses and domains rejected by this alias.

display name, expiration, and open time). New aliases inherit these shared attributes from the master and introduce new attributes of their own. Table 4.6 summarizes all of an alias's attributes. In addition to automatically creating aliases for incoming mail, SFM also creates them for outgoing mail.

Users send mail using the custom SFM SMTP server, which manages much complexity for the user. It helps ensure that SFM users always appear consistent to their contacts, by automatically selecting an appropriate alias for the message's recipient. If no such alias can be found, it creates a new alias automatically.

When a user sends a message to a new contact (or group of contacts), SFM creates an alias. Unless otherwise specified, it derives this alias from the first master in a user's master list. Users can specify an alternative master by way of their e-mail client's configuration. From Table 4.3, recall that user's append their PIN to the username: *username*`+`***PIN***`@`*host*. Users can also append a master name to send outgoing mail on a particular master: *username*`+`***PIN+master***`@`*host*.

Generally, SFM creates new, random aliases. It attempts to make their names pronounceable, alternating certain consonants and vowels. SFM creates alias names using the pattern

```
cvccvcvc,
```

where `c` is in the consonant set `bcdfhjklmnprstvwxz` and `v` is in the vowel set `aeiuy`.

Using this length and alphabet yields 236,196,000 aliases per user. An extensive user of SFM may have around 1,000 aliases, of which only some are open.

If users wish to override the random alias names, they can specify a name when sending a message to a new contact. SFM recognizes alias names in the subject line, given that they follow specific rules. First, two plus signs (++) must introduce the alias. Second, the suggested alias name must immediately follow those signs, without separating whitespace. This name must meet all of the standard naming rules, including a minimum length of five characters. Finally, if other characters follow the alias name, whitespace must separate subsequent characters.

For example, consider the following three subject lines:

> Greetings from Edmonton ++bobsmith
> ++bobsmith Greetings from Edmonton
> Greetings ++bobsmith from Edmonton

In all three cases, the recipient sees the subject

> Greetings from Edmonton

and the SFM server uses the alias name *bobsmith* (if possible).

Suppose that an alias already exists for a primary message recipient (i.e., a user listed in the "To" field) but not the secondary recipients (i.e., those listed in the "CC" field). If SFM does not find an alias personalized to all recipients, it creates a new alias by default. A user can override this behavior by simply including two plus signs (i.e., ++), preceded and followed by whitespace, in the subject. In this case, SFM will (1) find an open alias appropriate to a primary recipient and (2) use it, personalizing it to the secondary recipients. If SFM does not find a suitable alias, it will create a new one.

If SFM users always want to use the same alias when contacting a group of people, they can do that as well. Recall how users could specify a specific master in their e-mail client configuration: *username*`+PIN+master@`*host*. By replacing master with alias, they can specify a particular alias as well. Their e-mail client's e-mail address will become *username*`+PIN+alias@`*host*.

Using the System: The Web-Based User Interface

After connecting to the web-based user interface, users have several choices. The initial control panel includes the options: List Masters, Edit Profile, Logout, and Help.

"List Masters" provides an overview of a user's masters. It lists each master (e.g., bobsmith) along with its full address (e.g., bobsmith@sfm.cs.ualberta.ca). Users can also perform several actions:

- The "Create Master" link allows users to create new masters.

- By selecting a master, users can modify its attributes. Users can modify all of the attributes in Table 4.2, with the exception of the master name. To change a master's name, users must delete and recreate a master using the new name. Doing so invalidates all of the aliases derived from the master.
- For a particular master, users can quickly create a new alias using the "Quick Create Alias" link. SFM randomly generates the new alias's name. As such, they are suitable for online forms.
- Finally, users can place check marks beside masters and then delete them.

In summary, the "List Masters" page allows users to create, modify, and delete masters. As well, it supports the quick creation of aliases based on a particular master.

"Edit Profile" allows users to modify much of the information that they supplied when subscribing (Table 4.1), with the exception of their username. Text boxes provide easy access to this information. This section is also a gateway to additional functionality:

- Users can enable a challenge log. Recall that SFM sends challenges to rejected messages. This feature instructs SFM to keep a copy of all messages that initiate a challenge. To enable it, users must select the length of the challenge log (1 week, 2 weeks, 1 month, 2 months). After a message's age exceeds the maximum age of the challenge log, SFM deletes it. Enabling this option creates a new "Challenge Log" option in the control panel's main menu.
- Users can ask SFM to deliver local messages unconditionally. Messages arriving to a master or alias from a local server will be forwarded to the user's permanent address without further inspection. The system administrator declares those servers that are local.
- After enabling "Show the List Aliases link", SFM will display the link "List Aliases" in the control panel's main menu. When users select this link, they can view a complete list of their aliases.
- The "Append hyperlinks to forwarded messages" option instructs SFM to append SFM-specific hyperlinks to messages that SFM forwards to permanent addresses. These links allow users to quickly edit their profile, view the master or alias that the message arrived on, and so on. When users reply to a message containing these links, the SFM server will attempt to remove them from both the plaintext and HTML variant of the message, if each variant exists.
- Finally, users can enable the "Show the Legacy Patterns link". This link appears in the control panel's main menu under "Legacy Patterns". SFM provides this feature only to ease the transition to SFM from old filters driven by patterns that describe legitimate e-mail (like RabidFire and procmail).

In summary, the "Edit Profile" page allows user to change many of their profile's attributes as well as enable optional functionality.

The "Logout" and "Help" links in the SFM control panel behave just as expected. The former closes a session to securely log out of the SFM web-based user interface. The latter describes much of the web interface as well as configuring an e-mail client.

When enabled, "List Aliases" provides an overview of a user's aliases. It lists each alias, the alias's associated master, the first entry in its personalization list, the first entry in its blocked senders list, its expiration date, and finally its closing date. Given that the alias list will grow over time and become rather long, it provides filter functionality. Users can filter by: master name, alias name (partial or exact), personalization (partial or exact), and finally alias state. Users can also perform several actions:

- The "Create Aliases" link allows users to create new aliases.
- By selecting an alias, users can modify its attributes. Users can modify all of the attributes in Table 4.6, with the exception of the alias name. To change an alias's name, users must delete and recreate an alias using the new name.
- Finally, users can place check marks beside aliases and then delete them.

In summary, the "List Aliases" page allows users to create, modify, and delete aliases.

When enabled, the "Challenge Log" lists all messages arriving at SFM that resulted in a challenge. It lists each message with the associated date and time, sender, subject, and size. Since this list can grow rather large, SFM provides filter functionality. In particular, users can filter by the master name, as well as patterns. For patterns, users can specify one of sender, subject, headers, and body. Users can also perform several actions:

- By selecting a message's "View" link, users can view the message that initiated the challenge.
- After placing check marks beside messages, users can (a) remove, (b) deliver, or (c) deliver and remove the indicated messages.

In summary, the "Challenge Log" provides a list of all messages that initiated challenges. Users can view these messages as well as deliver and remove them.

When enabled, the "Legacy Patterns" page lists all of a user's *legacy patterns*. A legacy pattern is a simple rule that supersedes SFM's efforts to identify legitimate e-mail. By default, this feature is disabled and is provided only to ease the transition to SFM from old filters driven by patterns that describe legitimate e-mail (like RabidFire and procmail). This feature is not intended as a permanent solution. Users are advised not to view *legacy* patterns as a way to increase the reliability of their future contacts. Instead, they offer a quick way to immediately accommodate into SFM setups

whose contacts that have been already established and declared as legitimate for some old filters. SFM cannot assume responsibility for protecting those users whose legacy patterns are overly permissive.

The legacy patterns page includes two text boxes. One text box lists rules that apply to headers and the other to the body. Each rule is a regular expression, and each regular expression occupies one line in the text box. The expression syntax corresponds to the Tcl variety. For example, consider the following examples of header rules.

```
From:.*boers@cs\.ualberta\.ca
Subject:.*NSERC
Subject:.*CMPUT
Received: .*from [^ ]*microsoft.com
```

The first rule permits all messages from the specified sender, in this case any address ending with `boers@cs.ualberta.ca`. The next two rules match a message's `Subject` header. The final rule matches the `Received` headers, those prefixed by mail servers.

Note that the following characters: `.][\)(*+^$` are special. The case of letters is ignored during matching. Initial and trailing blanks are ignored. For sanity reasons, body patterns are only sought within the first 16KB of the message text.

When users disable legacy patterns in their profiles, SFM disables all patterns while leaving them in the database. Thus, if they subsequently reactive them, they will see their old collection of rules.

4.1.2 SFM Non-Subscribers

Recall that SFM has two types of users: (a) subscribers and (b) non-subscribers, trying to contact subscribers. The previous subsection described a subscriber's perspective. This subsection describes a non-subscriber's perspective.

Our system is completely backwards compatible with the existing e-mail infrastructure and depending on the situation, may be invisible. Non-subscribers use the service whenever they write to subscribers. Both master and alias addresses look no different than traditional e-mail addresses.

If a subscriber initiates communication with a new contact, the service generates a new alias for the outgoing message on-the-fly. The subscriber simply writes the message as with traditional e-mail. The receiver need only reply to the message, with no knowledge of SFM, and the subscriber will receive the reply. Figure 4.5(a) followed by (d) reflects this scenario.

If the non-subscriber initiates the communication, the service becomes visible, i.e., Figure 4.5(b). The non-subscriber begins by sending a message to a master address. Recall that subscribers do not receive messages sent to their master addresses; instead,

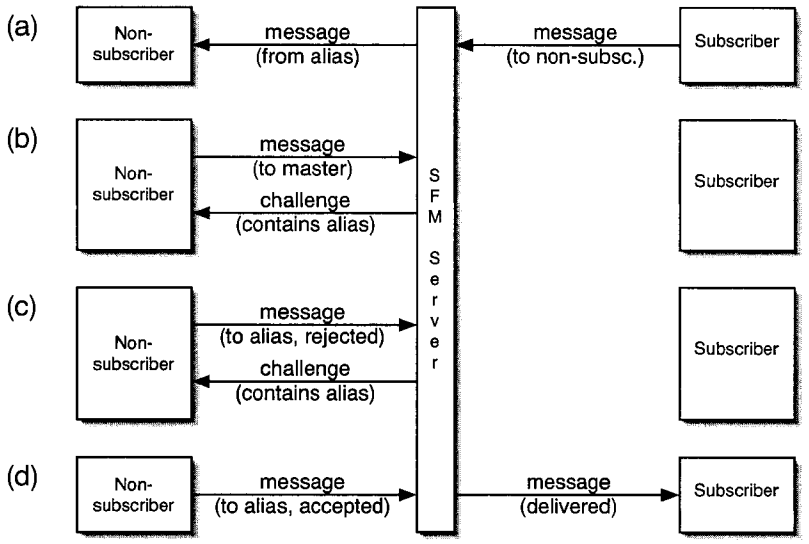

Figure 4.5: The protocol for communicating with an SFM subscriber. In (a), the subscriber initiates the communication and the system is transparent in further communication. In (b), the non-subscriber initiates the communication by writing to a master. In (c), the non-subscriber initiates the communication (or attempts to continue communication) on an invalid alias. Finally, in (d), the non-subscriber communicates on a valid alias and the system is again transparent.

the service replies on their behalf. Soon after sending the message, a reply arrives from the subscriber's master address. Instructions in the message ask the sender to resend their message to an alias. When the user sends mail to the alias, communication proceeds as in Figure 4.5(d).

Several situations can give rise to the scenario in Figure 4.5(c). First, suppose that a subscriber deletes a working alias. Instead of simply rejecting the message, the service challenges the sender with a new alias. Second, suppose that the sender is blocked on the alias. An alias's blocked senders list is not a blacklist for the SFM subscriber, but merely for the one alias. In this case, the service challenges the sender with a new alias. Third, suppose that the alias is closed and not personalized to the sender's address. This situation could occur if the sender received the alias from a third party, where the alias is personalized to that third party. Here, the service challenges the sender with a new alias. In all of these cases, the sender receives a new alias. With the new alias, the sender can contact the subscriber as in Figure 4.5(d).

Once a sender has a valid alias for a subscriber, the service becomes invisible, i.e., Figure 4.5(d). Communication continues with no interruptions and no unnecessary challenges. In most cases, aliases are long-term points of contact rather than simply disposable addresses.

SFM also supports the scenario where a subscriber wishes to make aliases available

through a web site. For example, a web site's "Support" link may generate an alias for the non-subscriber to use. By allowing non-subscribers to obtain aliases in this manner, it avoids the challenge/response mechanism for new contacts.

The preceding description of alias acquisition (i.e., through a challenge e-mail and through a web site) does not discuss the presentation of the alias. Chapter 5 will discuss the opportunities and challenges for such presentation.

4.1.3 SFM-SFM Interaction

The previous section considered a non-subscriber establishing contact with a subscriber. This section considers a subscriber establishing contact with another subscriber. The events here are quite similar to those previously described.

In Figure 4.6(a), subscriber A attempts to contact subscriber B through a master address (B_M). User A's SFM server automatically creates an alias for the outgoing message (A_B). User B's SFM server rejects the message and replies to user A's newly created alias (A_B) with a challenge that includes a new alias (B_A). User A's server accepts the challenge, since the alias is open and delivers it to user A. After viewing the message and identifying user B's alias, user A resends the message to this alias (B_A). User A's SFM server automatically uses the existing alias (A_B) for the outgoing message since it recognizes that the alias B_A originated from the master B_M. User B's server accepts the message, since the alias (B_A) is open, and delivers it to user B.

In Figure 4.6(b), similar events take place when subscriber A attempts to contact subscriber B on a closed alias (B_C) not personalized to subscriber A. Note that user A's SFM server always uses the alias A_B when communicating with user B. Both aliases (B_C and B_A) apparently originate from the same master B_M, so it does not create a new alias.

As described, SFM experiences no issues with SFM-SFM interaction. The steps in subscriber-to-subscriber communication are very similar to those between non-subscribers and subscribers.

4.1.4 Common Usage Scenarios

With traditional e-mail, people may use more than one e-mail address. An e-mail user may have one e-mail address for trusted contacts and another for untrusted contacts. SFM simplifies this situation; the user only requires one master for both types of contacts. In another situation, an e-mail user may have one address for personal use and another for business use. In this case, SFM allows this user to have two masters, one for each type of correspondence.

Master e-mail addresses can be openly published and given to individuals. Only

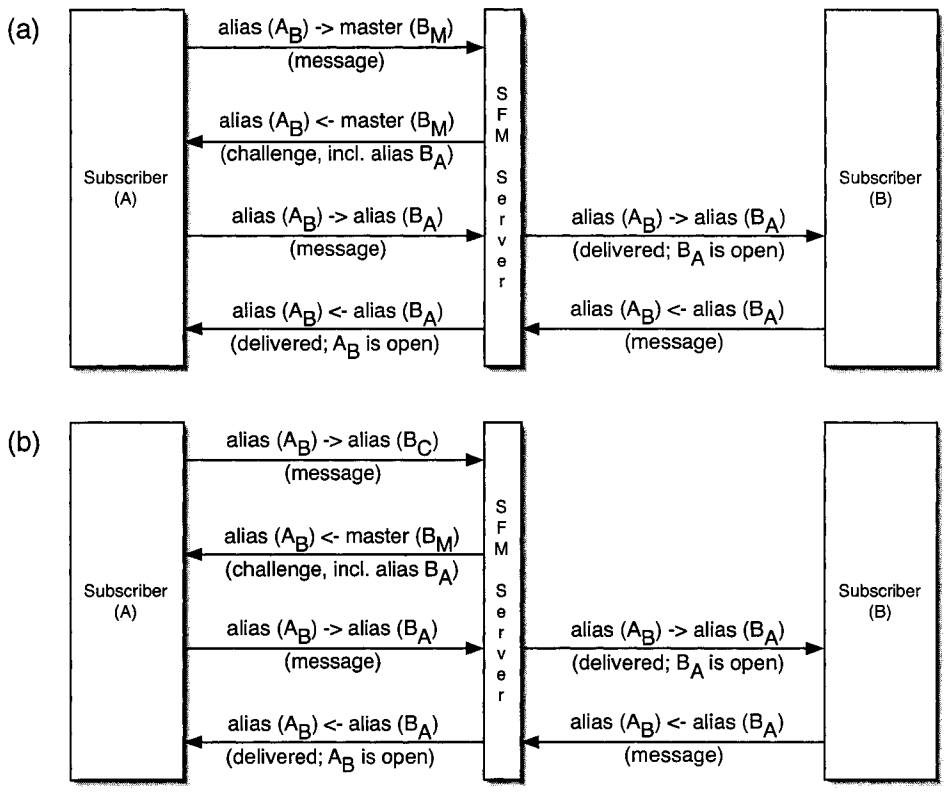

Figure 4.6: The event progression when a subscriber initiates communication with a subscriber. In (a), the subscriber sends the first message to a master. In (b), the subscriber sends the first message to an alias that rejects the sender.

humans can use them, as only humans are able to answer the challenge. Automatic mailers, on the other hand, are unable to answer the challenge and cannot abuse the address.

Even though SFM provides superior protection against automatic mailers, it provides enough flexibility for its users to solicit automatic mailings. For example, users may want these mailings when they make online purchases, subscribe to solicited advertisements, and subscribe to mailing lists. In these instances, a user can manually create aliases using SFM's extensive web-based user interface. Anyone (human or machine) that sends to one of these manually created aliases will not need to answer a challenge.

If a user makes substantial use of mailing lists, the user may benefit from modifying their e-mail client's configuration. In an e-mail client that supports multiple identities or accounts, a user can set up a new one for the mailing list. Recall from Section 4.1.1 that a user can specify an alias in their e-mail address when configuring their e-mail client. After specifying such an e-mail address, all outgoing mail using that account will use the specified alias. Making this type of configuration change will ensure that a particular alias is always used with a particular mailing list.

SFM users can effectively use our system with a single master. When they solicit an automated mailing, they can manually create an alias specific to the sender. The web interface is accessible from any web browser, so whether at home, in the office, or on the road, users can create new aliases as they are required.

This section described SFM from a user's perspective, a natural way to introduce a system of this complexity. With that high-level knowledge, the next sections delve into the technical details behind the system.

4.2 Requirements

The Spam Free e-Mail (SFM) service is available in a *tarball*, an archive produced by the UNIX `tar` utility. This file contains complete installation instructions,[2] including the list of requirements presented here. This section does not simply reproduce those instructions, but instead aims to provide some insight into them. The following discussion is independent from those installation instructions.

SFM consists of three primary components and many auxiliary files. None of the components require `root` or even regular user access. We strongly recommend that users create a new account, one capable of login (to set things up) but not a regular user account. For the remainder of this section, we will assume that this account's username

[2] `SFM/README.html`

is `sfmowner`. Furthermore, we will assume that the SFM tarball is extracted in this user's home directory.

In addition to what we include in the SFM tarball, the service depends on several preexisting software packages. Each software package meets a particular requirement of the system:

1. displaying our extensive web-based user interface,
2. storing subscriber data,
3. transferring e-mail to and from remote hosts,
4. managing network connections to third-party software and our own custom software,
5. processing images, a technique that we use to display aliases, and
6. bringing everything together with an environment for our own code.

An awareness of these dependencies and how we fill them will aid in our later description of SFM's three primary components.

4.2.1 Web-Based User Interface

As described in Section 4.1.1, our service includes a substantial and essential web-based component. At the web site, users can learn about our service, subscribe to it, and manage their accounts.

We have thoroughly tested SFM with the Apache web server and recommend it for SFM deployments. Developers first released Apache to the public in April 1995. After over ten years of development, the Apache web server is now the most popular one on the Internet. It is freely available as open-source software and provides a stable, mature platform for our work.

Obtaining Apache

The Apache web server is freely available at the Apache web site.[3] SFM requires version 2.0.x or later, and it must include the ability to handle Secure Sockets Layer (SSL) connections.

Configuring Apache

The SFM web site consists of both static and dynamic content. The site's static content includes information about the service. We reduce redundancy in it by using Server Side Includes (SSI). As an SSI-aware web server parses an HTML page, it looks for specially formatted comments in the HTML code. These comments instruct the server

[3] http://httpd.apache.org/

to pause parsing and complete an action before continuing. In the case of SFM, they ask the server to execute Common Gateway Interface (CGI) scripts that generate the headers and footers of static pages. By using SSI, each HTML file simply contains a comment at the start, the content for that page, and a comment at the end. At the cost of increased load on the web server, SSI allows for more efficient maintenance. The site's dynamic content appears when users try to create a new account or manage their existing account. An extensive CGI script and its dependent auxiliary files generate all of the site's dynamic content.

The Apache documentation describes its installation and setup. The following configuration instructions describe SFM-specific changes to an existing, running Apache server.

The first changes occur in the Apache configuration file.[4] To update this file for SFM, complete the following steps.

1. We must first know from where Apache loads documents. Locate and make note of the directory set as the `DocumentRoot`. For example, in the directive

   ```
   DocumentRoot "/var/www/html"
   ```

   the `DocumentRoot` is `/var/www/html`.
2. With knowledge of the `DocumentRoot`, we need to ensure that Apache has the ability to process SSI, follow symbolic links, and execute CGI scripts. Locate the subsection that contains settings for the `DocumentRoot` directory. Carrying on from the previous example, the text

   ```
   <Directory "/var/www/html">
   ```

   introduces this subsection. Within the subsection, locate the `Options` directive. The required options include (a) `Includes` to enable SSI, (b) `FollowSymLinks` so that Apache can follow a symbolic link into the SFM directory, and (c) `ExecCGI` so that Apache can execute the CGI script that generates the site's dynamic content; add any missing options. SFM includes a custom 404 (page not found) error page that can be set in the Apache configuration. Within the same subsection, add the line

   ```
   ErrorDocument 404 /error404.shtml
   ```

   to enable the custom error page.
3. When a user asks Apache to load a directory, Apache prefers to serve an index page; the Apache configuration lists possible index pages. The default configuration does not list the SFM index page, so we need to add it. Locate the `DirectoryIndex` directive. The standard extension for files that use SSI is `.shtml`, and our `index` file uses this extension. Add the filename `index.shtml`

[4] On our system, this file is `/etc/apache/httpd.conf`.

to the list so that Apache can load our index file.

4. The default Apache configuration does not treat our scripts as CGI scripts because it does not recognize their extension. We need to add the extension to Apache's list of valid CGI extensions. Locate the `AddHandler cgi-script` directive. The SFM scripts use the extension `.aph`, so add this extension to this directive so that Apache can execute SFM's dynamic content.

With these changes to the Apache configuration file, Apache should support SFM.

The SFM distribution contains two sets of web documents: `./ADMIN/HTDOCS` and `./htdocs`. The former is for development and the latter for deployment. When setting up SFM, users often create a symbolic link from Apache's `DocumentRoot` directory (previously noted) to the SFM *deployment* directory. For example, suppose that the `DocumentRoot` directory is `/var/www/html` and the SFM *deployment* directory is `~sfmowner/htdocs`. The command

```
ln -s ~sfmowner/htdocs /var/www/html/sfm
```

will create such a symbolic link.

For dedicated SFM servers, creating two more symbolic links will complete the configuration. From the `/var/www/html` directory, the commands

```
ln -s sfm/index.shtml index.shtml
ln -s sfm/error404.shtml error404.shtml
```

will create symbolic links for the index page and custom 404 error page, respectively.

4.2.2 Data Storage

Starting from when users subscribe, SFM stores user data. Initially it stores data such as a user's permanent e-mail address and password, and when users start to configure and use the service, it stores items such as their masters and aliases. All of the data that SFM stores are key/value pairs.

To store these data, we look to Sleepycat Software and Berkeley DB. Like Apache, Berkeley DB has a long history after being first broadly distributed in 1992. Since its initial release, its popularity has grown to where it is now the most widely used open source developer database in world. It is freely available for open-source projects and provides a stable, mature platform for our work.

Obtaining Berkeley DB

Users of Berkeley DB can choose between two licences: their open source or commercial license. The open source license permits use of Berkeley DB in open source projects such as SFM. As such, SFM users can freely obtain Berkeley DB from their web site.[5]

[5] http://www.sleepycat.com/

SFM requires version 4.2 or newer and its included Tcl API.

Configuring Berkeley DB

SFM requires no SFM-specific Berkeley DB configuration settings.

4.2.3 Message Transfer

Our service requires a message transfer agent (MTA) to communicate with remote mail servers. Whenever someone writes to an SFM subscriber, this server receives the incoming mail before passing it to SFM for processing. When SFM subscribers send mail, this MTA manages the actual transmission to remote hosts.

Many mail servers exist, with one of the most common and mature being sendmail.[6] The first version of SFM used sendmail, but configuring it was difficult. To simplify the setup process, the current version of SFM uses qmail,[7] a more friendly alternative to sendmail.

Obtaining qmail

The qmail MTA is available in source code form for a variety of operating systems. Note that qmail is not *open source*, since its author (D. J. Bernstein) prohibits its distribution when modified. SFM users can freely obtain it from its web site.[8] SFM requires version 1.03 or newer.

Configuring qmail

The qmail documentation describes its installation and setup. The following configuration instructions describe SFM-specific changes to an existing, running qmail server.

Subscribers to our service, both the forwarding and hosted variety, are virtual in terms of the operating system. They do not have accounts on the system; instead, they are merely entries in a database and files in the file system. Normally when a message for a nonexistent user arrives, the mail server bounces a "user does not exist" message. Given our situation, where users are virtual, we modify this behavior.

A system administrator can easily modify how qmail delivers messages. By default, qmail delivers incoming mail to the `Mailbox` file in a user's home directory (i.e., `~/Mailbox`). Files in the `alias` user's home directory (i.e., `~alias`), a user created during the qmail installation, can override this default behavior. Each file named `.qmail-`*x* indicates that e-mail arriving for user *x* should receive special handling. These files can contain a list of delivery instructions, one per line. If a file exists but

[6] http://www.sendmail.org/
[7] http://www.qmail.org/
[8] http://www.qmail.org/

is empty, qmail will follow the default delivery instructions; in this case, it will deliver the message to the file ~alias/Mailbox.

In our installation, we create three empty files in ~alias/:

- .qmail-root for mail addressed to root,
- .qmail-postmaster for mail addressed to postmaster, and
- .qmail-mailer-daemon for mail addressed to mailer-daemon.

The Mailbox file in ~alias/ is rotated by logrotate and symbolically linked to /root/Mailbox, so that all e-mail arriving to the three users (root, postmaster, and mailer-daemon) is effectively received by root and kept in root's Mailbox for a limited time.

In the ~alias/ directory, we also modify the file .qmail-default. This file describes the handling of all e-mail messages addressed to formally nonexistent users (ones not in the password file and not otherwise explicitly aliased) and contains the single line

```
| forward sfmowner@localhost
```

which tells qmail to forward all apparently undeliverable messages to sfmowner.

The sfmowner home directory contains a file .qmail-default. This file provides a list of delivery instructions for all mail arriving at the sfmowner account. In our setup, this file contains

```
| preline tee -a incoming | /home/sfmowner/Mailer/filter
```

to pipe all incoming mail into the SFM *filter* component (discussed in Section 4.3.3).

The *filter* component determines whether the incoming message is addressed to an SFM-subscriber or a truly nonexistent user. In the case of an SFM-subscriber, this script processes the message. In the case of a nonexistent user, this script initiates a bounce, i.e., a "user does not exist" message.

4.2.4 Connection Management

The term *daemon* refers to a process that runs in the background. They generally run as long as the system is running; they are started on startup and stopped on shutdown. The SFM server does not run as a daemon; instead, it runs only when required for an incoming connection.

Since SFM does not run all of the time, it needs another program to constantly monitor its port for incoming connections. We use the program xinetd to watch for incoming connections. xinetd is an Internet services daemon (or super-server); it listens for incoming connections on selected ports. When it detects an incoming connection, it passes the connection to the appropriate program, as determined by its configuration files.

Obtaining xinetd

The program xinetd is common on many UNIX-like operating systems. If the system does not already have it installed, you can obtain it from their web site.[9]

Configuring xinetd

After obtaining xinetd or ensuring that it exists, configure it for SFM. The configuration file `xinetd.conf` is generally located inside `/etc/`. Depending on the system's configuration, the services may not be listed in this configuration file. Instead, they may reside in a separate directory, where each file specifies one or more services. To determine whether this is the case, look in the file `xinetd.conf`. If the directive `includedir` exists, the configuration may load files from another directory.

If the system uses one main configuration file, open it. Otherwise, create a new configuration file named `sfmsmtp` in the appropriate directory and open it. Inside the file, enter the commands that tell xinetd how to handle incoming connections for the custom SMTP server. For our SFM systems, this file contains:

```
service sfmsmtp
{
        type            = UNLISTED
        port            = 9025
        socket_type     = stream
        wait            = no
        user            = sfmowner
        server          = /home/sfmowner/Mailer/smtp
        disable         = no
}
```

The port, user, and server may vary depending on the setup.

For servers that handle the *hosted* account type, configure xinetd to load the POP3 server. On systems where xinetd uses directories, create and open a new file named `sfmipop`. For our SFM systems, this file contains:

```
service sfmipop
{
        type            = UNLISTED
        port            = 9110
        socket_type     = stream
        wait            = no
        user            = sfmowner
        server          = /home/sfmowner/Mailer/pop
        server_args     = -Y
        disable         = no
}
```

The port, user, and server may vary depending on the setup.

Finally, add entries for qmail so that it can accept incoming SMTP connections on port 25 (the standard port). On systems where xinetd uses directories, create and open a new file named `qmail-smtp`. For our SFM systems, this file contains:

[9] http://www.xinetd.org/

```
service smtp
{
        socket_type = stream
        wait = no
        user = qmaild
        id = qmail-external
        interface = meander.cs.ualberta.ca
        server = /var/qmail/bin/tcp-env
        server_args = -R /var/qmail/bin/qmail-smtpd
        disable = no
}

service smtp
{
        protocol = tcp
        wait = no
        user = qmaild
        id = qmail-internal
        interface = 127.0.0.1
        env = RELAYCLIENT=
        server = /var/qmail/bin/tcp-env
        server_args = -R /var/qmail/bin/qmail-smtpd
        disable = no
}
```

These settings may vary depending on the installation.

4.2.5 Image Processing

In previous sections and chapters, only vague descriptions have described SFM's challenges. They have described that SFM communicates aliases to users, but have not described that communication. The current implementation of the service requires an image processing program to create images that SFM uses in the challenge. Chapter 5 provides further details.

For image processing, SFM can use the GD Graphics Library[10] or the GNU Image Manipulation Program (GIMP).[11] The installation of the GD Graphic Library is much easier, and for that reason, we recommend GD.

Obtaining the GD Graphics Library

The GD library is available from their web site.[12]

Configuring the GD Graphics Library

The GD graphics library documentation describes its installation and setup. SFM requires no SFM-specific GD Graphics Library configuration settings.

SFM includes a script to check whether image generation works. For this, the subdirectory `SFM/Mailer/CAPTCHA` contains a script `testme_gd.sh`. This script assumes

[10] http://www.boutell.com/gd/
[11] http://www.gimp.org/
[12] http://www.boutell.com/gd/

Figure 4.7: The file `image.jpg` produced by command `testme_gd.sh`.

that `~sfmowner` is the home directory of the SFM user; if this is not the case, edit the script. A successful test will produce the file `image.jpg` containing the text

```
captcha.works@fine.on.this.server
```

encoded in the JPEG image (Figure 4.7).

4.2.6 An Environment for SFM

The primary components in SFM are three extensive scripts. We have already introduced these three scripts:

1. the web site's dynamic content is handled by a CGI script,
2. the message transfer agent passes messages to another program for processing, and
3. xinetd passes incoming connections to a custom SMTP server.

We have implemented all three of these applications as Tcl scripts.[13]

All three of these scripts are written in the Tcl language. The Tcl interpreter is our final dependency for using SFM; it requires Tcl version 8.4 or later.

Obtaining Tcl

Tcl is available for download, for a wide-variety of platforms, from their web site.[14]

Configuring Tcl

Tcl requires no SFM-specific configuration.

4.3 Components

The service consists of three extensive Tcl scripts.

The first, *https*, provides the dynamic content on the web-based user interface. When users log in to their account, they do so using this script. The script loads various forms as necessary to provide users with the information that they seek.

The second, *smtp*, provides an SMTP server for subscribers' outgoing mail. The super-server, xinetd, passes connections to this script. The script is responsible for

[13] http://www.tcl.tk/
[14] http://www.tcl.tk/

automatically creating aliases for new contacts and selecting appropriate aliases for existing contacts. After reading and processing the message from the user, this script redirects the message to qmail for its actual delivery.

The final component, *filter*, provides for the filtering of incoming e-mail. When the qmail message transfer agent cannot locate a user, it redirects the message to this script. The script determines whether the message is for a subscriber and appropriately processes the message.

The following sections investigate the intricacies of each script.

4.3.1 *https*

The *https* component provides the dynamic content on the web-based user interface. When using this dynamic content, users always communicate with a single Tcl script.

Depending on its arguments, this Tcl script displays one of several forms. Each form resides in its own file, each with a single function `page`. After the main Tcl script performs the necessary preprocessing, it simply executes the `page` function in the appropriate form file.

With HTML forms, two methods can communicate form values (arguments) to a Tcl script [59]: the `get` and `post` methods. The main difference between these methods is the way that they transfer the arguments. With the `get` method, the client appends the set form data to the universal resource identifier (URI). With the `post` method, it sends the data separately. The HTML 4 specification [59] recommends using the `get` method when the form is idempotent (i.e., causes no side-effects). We follow this suggestion in our interface, and our Tcl script recognizes both methods of form submission.

Our Tcl script generally works as follows. It first determines whether the client called it with a `get` or `post` method.

If called with the `get` method, the URI must have a standard form. It can be:

- 1 argument of length 1: the single argument corresponds to a form that requires no authentication. An argument of `0` indicates the new user form, `1` the login form, and `2` the account verification form.
- 1 argument of length greater than 1: if the argument contains a master name, then an alias acquisition form appears. If the argument contains an encrypted alias and master name (i.e., *`encrypted.master`*), then it displays the image of that alias. These forms also require no authentication.
- 2 or 3 arguments indicate an action from the link panel (i.e., a link from an e-mail message). The form `action` also requires no authentication.
- 4 or more arguments indicate the usual case where the arguments are formatted like: user, session tag, counter, and page followed by zero or more arguments.

All of these forms require authentication (a preexisting, open session). The client uses the user and session tag combination to authenticate the user. SFM generates a new session tag each time a user successfully logs in using his/her username and password. When users log off, the session tag expires, and they must log in again to generate a new one.

If called with the `post` method, some necessary fields must exist in the form. We use essentially two types of forms: those that require authentication and those that do not. Depending on the type of form, SFM requires different form fields.

For unauthenticated forms submitted with the `post` method, the field `ftype`, which specifies the name of the form, must exist. Depending on the form being run, other fields may also be necessary. For example, the login screen is an unauthorized form, since when a user submits it, he/she has not yet been authenticated. This form includes some additional fields: the user's username and password.

For authenticated forms submitted with the `post` method, several fields are required. As well as the form name (`ftype`), the system requires the username (`username`) and the session ID (`stag`). Before executing an authenticated form, the system loads the user's record and verifies that the submitted session ID matches the stored one. Remember that the session ID is generated randomly each time the user connects and cleared each time the user logs out.

The current version of SFM consists of five unauthorized forms and eleven session forms. The unauthorized forms are (1) `newuser` for subscribing to the service, (2) `login` for logging into existing accounts, (3) `verify` for verifying a newly created account, (4) `aliacq` for acquiring an alias for an existing subscriber, and (5) `action` for completing an action, linked to by e-mail. The authorized forms include

1. `root` for listing masters,
2. `newal` for quickly creating aliases,
3. `list` for listing aliases,
4. `alias` for creating and viewing masters and aliases,
5. `chlog` for listing the messages in the challenge log,
6. `chmes` for viewing a message in the challenge log,
7. `legpt` for viewing and modifying legacy patterns,
8. `admin` for performing various administration tasks,
9. `adrec` for editing records in administration mode,
10. `prof` for modifying user profiles, and
11. `logout` for logging out of an existing session.

These forms, both unauthorized and authorized, constitute all of the dynamic content

of the web site.

In the previous list, notice the two administration forms (`admin` and `adrec`). In a user's profile, a flag indicates whether a user is an administrator. If the flag exists, then a menu option appears to access the main administrator form `admin`. When performing any tasks involving the two forms, the administrator flag is always verified first.

4.3.2 *smtp*

The custom SMTP server provides an interface for outgoing mail. A single Tcl script implements it, and it shares much code with the web-based user interface.

Since xinetd manages incoming connections, the script simply reads from `stdin` and writes to `stdout`. It implements a minimalist SMTP server that nearly meets the SMTP standard [35]. With the exception of the `vrfy` command, it supports the minimal set of commands: `ehlo`, `helo`, `rset`, `quit`, `noop`, `mail`, `rcpt`, and `data`. These supported commands suffice for it to work with all major e-mail clients. Other SMTP servers never communicate with our SMTP server: ours serves only our users for outgoing mail, and thus only communicates with those user's e-mail clients.

After xinetd calls the script, it initializes its use of standard input and output. Immediately after that, it prints out the welcome banner and enters an infinite loop where it accepts commands from the e-mail client.

Recall that SFM subscribers include a PIN in their e-mail address. This inclusion allows SFM to authenticate users early in the SMTP connection. In a transaction, the `from` command follows the initial `helo` or `ehlo`. As part of the `from` command, the e-mail client sends the sender's e-mail address (including PIN and optional master/alias name). SFM identifies the username, PIN, and master/alias name within this e-mail address. For forwarding accounts, the username is the e-mail address without the PIN and optional master/alias name. For hosted accounts, the address's domain is the local SFM server and the username is simply the e-mail address minus the PIN, optional master/alias name, and domain.

After parsing the sender's e-mail address, SFM can compare the parsed PIN with the one stored in the user's profile. If they match, then communication can continue. Otherwise, the server returns an "unauthorized" error message and disconnects the client. After the `from` command, users proceed with the `rcpt` and `data` commands.

After the SFM SMTP server accepts a message, further processing occurs before the message reaches qmail for sending. The SMTP server rewrites the sender address to include an appropriate alias. It first uses all of the message's recipients to search for an existing alias. If an alias personalized to all of the recipients exists, it uses that alias.

Otherwise, it creates a new alias and personalizes it to the recipients. After rewriting the sender address, it stores the message ID of the outgoing message in a database. With the message ID, it can later detect bounces to the message. If the user has the "append hyperlinks" functionality enabled, it will attempt to remove the hyperlink panel (in case the message is a reply or forward). Finally, it passes the message to qmail for sending.

4.3.3 *filter*

While the *smtp* component handles outgoing mail, the *filter* component handles incoming mail. After qmail accepts a message, it first determines whether its recipient is a local user. Recall that all SFM subscribers are virtual and not local. For local users, it deposits the message into their mailbox. For non-local users, it pipes the message into a script, which is the third component of the SFM system. The *filter* script examines incoming messages and determines whether to deliver them to subscribers.

Before processing can begin, *filter* reads the message from standard input. It stores the headers into a Tcl data structure and copies the message body into a temporary file. For all incoming messages, it also adds an entry to a log file. With the headers and body stored, it can begin processing the message.

It first determines whether the message is a *refiltering* request. We call our technique for recovering an abused permanent e-mail address refiltering. Recall from Table 4.1 that a user's profile contains a filter cookie, a header field that distinguishes SFM-processed e-mail. In the absence of a filter cookie, a user's mail client can forward an incoming message to a master address. When SFM receives such a message, it modifies the content altered in the act of forwarding and then processes the message as if sent directly to a master.

When a message is not a refiltering request, SFM validates the master and alias. If the master cannot be found, it bounces an "unknown user" error message. If the alias does not exist for the specified master, it ignores the missing alias and treats the message as if addressed to the master.

The *filter* script now makes a few more checks. If the message lacks a `From:` header, it attempts to bounce the message to the envelope sender. Next, it looks at legacy patterns; if the legacy patterns accept the message, SFM delivers it to the user with no further processing. When the legacy patterns do not match, further processing depends on whether the sender addressed the message to a master or alias.

For messages addressed to masters, processing consists of several steps.

1. If the user enabled the unconditional delivery of local messages and the sender is local, SFM delivers the message with no further checks.

2. If the sender's address matches the subscriber's forwarding e-mail address, it logs and drops the message.[15]
3. If the sender is in a list of addresses to which SFM should never send bounces, it drops the message. The system administrator initializes and maintains these lists across several text files in `Mailer/CONTROL`.
4. If the envelope sender does not exist, it drops the message. When SFM sends a bounce, it uses the envelope sender rather than the sender address in the message headers. Without this address, SFM cannot send a challenge.
5. If it finds an alias personalized to any of the sender addresses, it uses it. Otherwise, it creates a new alias.
6. If a user has enabled the challenge log, it records the incoming message in it.
7. Using the master as the sender address, it sends a challenge to the user. This challenge includes the alias encoded in an image.

For messages addressed to aliases, processing proceeds quite differently.

1. Recall that SFM stores the message IDs of outgoing messages. If a recognized message ID exists in the message (i.e., it is a bounce), SFM delivers the message. For messages with a non-empty and non-self envelope sender, it also updates the alias personalization.
2. If the user enabled the unconditional delivery of local messages and the sender is local, SFM delivers the message with no further checks.
3. If the sender's address matches the subscriber's forwarding address, it logs and drops the message.[16]
4. If the alias will accept the message (acceptance depends on alias state, the personalization list, and the blocked list), SFM delivers the message. Table 4.7 describes the possible cases and outcomes for acceptance. Otherwise, it treats the message as if the sender addressed it to the master.

If the user has the "append hyperlinks" functionality enabled, *filter* will append the hyperlink panel before delivering the message.

4.4 Summary of the Spam Free e-Mail Service

Sections 4.1 to 4.3 progressively describe the Spam Free e-Mail (SFM) service, starting with a high-level user's perspective and finishing with a description of its three primary

[15]Upon dropping the first message, SFM sends a message to the subscriber explaining this behavior. If it did not drop these messages, anybody knowing a subscriber's permanent address and masters or aliases could use this simple trick to deliver spam to a subscriber's mailbox.

[16]Just as with messages to masters, it sends a message to the subscriber explaining this behavior at the first instance.

Table 4.7: Several factors determine whether SFM accepts a message for a subscriber. Given S (sender address), L_P (set of personalization addresses), L_B (set of blocked addresses), and the alias's state, this table describes possible outcomes.

$S \in L_P$	$S \in L_B$	State	Outcome
false	false	closed	reject: bounce: new alias
false	false	open	accept: add S to L_P
false	true	closed	reject: bounce: new alias
false	true	open	reject: bounce: new alias
true	false	closed	accept
true	false	open	accept
true	true	closed	reject: bounce: new alias
true	true	open	reject: bounce: new alias

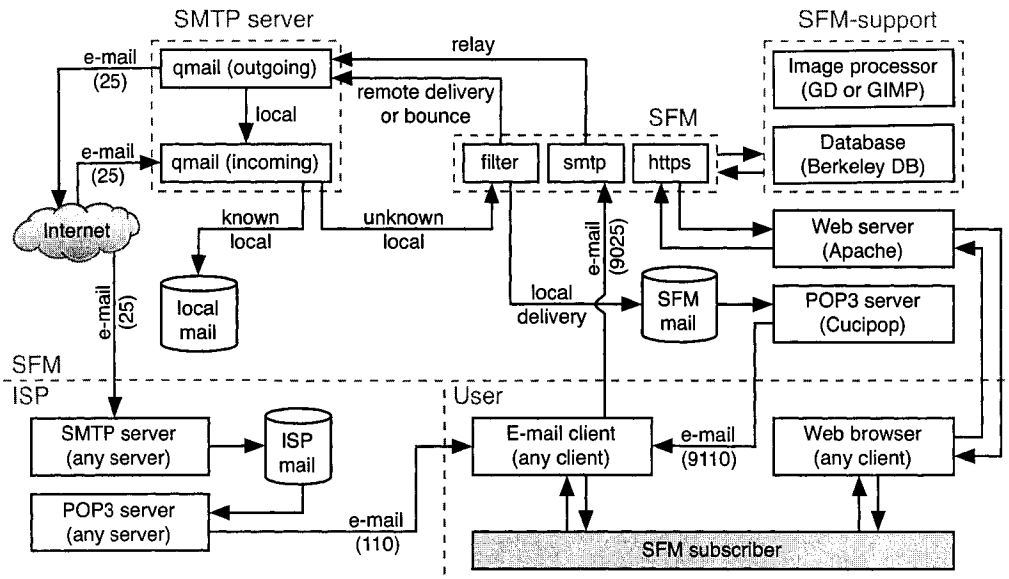

Figure 4.8: Summary of the SFM server setup.

components. This section presents a quick summary of the server setup and should be read while referencing Figure 4.8.

With a standard web browser, subscribers can connect to SFM's web site (Figure 4.8, *Web server*). The Apache web server serves both static and dynamic content. The static content introduces the service and consists of several files that use Server Side Includes (SSI) to reduce redundancy. A Tcl script, executed through the Common Gateway Interface (CGI), creates the site's dynamic content (Figure 4.8, *https)*. This script uses a collection of forms to display information to the subscriber, using the SFM-support components as necessary (Figure 4.8, *Database* and *Image processor*). The dynamic content allows a user to manage master and alias addresses, view messages that initiated challenges, and more.

When users send e-mail, they use a custom SMTP server written for SFM (Fig-

ure 4.8, *smtp*). This server hides much of SFM's complexity from the subscribers. Its tasks include (a) verifying that the sender is authorized to use the server, (b) locating an appropriate alias for the recipient, and if one does not exist, creating a new one, (c) rewriting the sender's address to make it consistent with the alias, (d) saving the message ID for later detection of bounces, and finally (e) attempting to remove the link panel if the user enabled it in his/her profile. After performing these tasks, the *smtp* component passes the message to qmail (Figure 4.8, *qmail (outgoing)*) for delivery.

For both forwarding and hosted accounts, e-mail enters *qmail (incoming)* from the *Internet*. qmail first checks whether the recipient has a valid user account on the system. When it finds a valid account, it deposits it in the user's `~/Mailbox` file. For all unknown recipients, qmail forwards the message to the user `sfmowner@localhost`. When `sfmowner` receives it, qmail automatically pipes it to the *filter* script for further processing.

For messages addressed to unknown users (i.e., not SFM subscribers), *filter* initiates a bounce (i.e., no such user). It generates a bounce message and passes it to *qmail (outgoing)* for delivery.

Several situations may result in a challenge, i.e., messages to (a) a known master but unknown alias, (b) a closed alias not personalized to the sender, and (c) an alias where the sender is blocked. In these cases, *filter* creates a new alias, and using the *Image processor*, it generates a CAPTCHA image that will communicate the alias to the sender. It creates and passes the challenge to *qmail (outgoing)* for delivery.

The *filter* accepts messages from unblocked senders when (a) an alias is open or (b) an alias is personalized to the sender. If an alias is open but not personalized to the sender, *filter* adds the personalization. Its next action depends on whether the receiving account is forwarding or hosted. For forwarding accounts, it forwards the message to the subscriber's permanent address using *qmail (outgoing)*. For hosted accounts, it deposits the message into a mailbox for the user (*SFM mail*). In both cases, it attaches the link panel before delivery when a user has enabled that option.

Users can check for incoming mail in a standard e-mail client whether their account is forwarding or hosted. For forwarding accounts, checking mail continues as before SFM. For hosted accounts, the *E-mail client* connects to the *POP3 server (Cucipop)* bundled with SFM. The user authenticates with the username and password associated with his/her account, and *Cucipop* retrieves the mail from *SFM mail*.

Chapter 5

Discussion

This chapter discusses various aspects of SFM. It begins by addressing some design decisions, followed by some security considerations. Our work remains a solution to eliminating spam insofar as its challenge mechanism (i.e., a CAPTCHA image) remains effective. Section 5.3 discusses some of the research surrounding this and other challenge techniques. Finally, Section 5.4 describes its storage requirements.

5.1 Design Decisions

Many design decisions culminate in the Spam Free e-Mail (SFM) service. In this section, we address some of the most recent modifications and additions in the system.

5.1.1 User Interface

The SFM web-based user interface is an integral component of the service. Using it, subscribers create and subsequently manage their accounts. Efficient use of the system partially depends on their ability to easily navigate and access relevant features in this interface.

In the middle of September 2005, the service went live with a renovated interface. It addressed several weaknesses that we identified with the previous interface. The previous interface

- provided indirect access to the log in form,
- required the retention of usernames,
- used low contrast images,
- included two modes of operation, basic and advanced, and
- displayed content using pop-up windows.

By solving these weaknesses, the new interface provides for increased efficiency, greater privacy, and increased accessibility.

In the previous interface, subscribers had to visit two web pages to log in to their accounts. After visiting the main welcome page, they had to select the "Login" link to access a separate login page. On that login page, they could enter their username and password, submit the login form, and gain access to their account. This behavior is inconsistent with the common-practices of other similar sites[1] and consequently violates the *consistency* golden rule of interface design [52]. The new interface reduces this two-step process to a single step. The login form (i.e., username and password boxes) appears on the main page to simplify the login process and make the site more consistent with other similar sites.

The previous interface automatically remembered the last-entered username. On a private computer, remembering the username is very convenient. When on a public computer, automatically remembering this information could expose a user's permanent e-mail address to the next computer user. User interfaces can include mechanisms to protect privacy [52], and the new interface adds a checkbox titled "Remember my username". It allows users to choose whether or not the site remembers their username based on their situation (i.e., public or private computer).

Low-contrast images in the previous interface reduced the site's accessibility to visually-impaired users. For these users, some of the headings and buttons could be very difficult to read. These images have been subsequently replaced with highly contrasting text (i.e., dark blue text on a light blue background). The W3C Web Content Accessibility Guidelines specify criteria for making foreground information easily distinguishable from the background [60]. The dark blue text on the light blue background meets the criteria for this guideline's highest conformance level. To add emphasis to titles, that text appears in a larger text size.

As previously mentioned, SFM ran in two modes: a basic and an advanced mode. Switching from the basic to advanced mode introduced dramatic changes in the interface with no accompanying explanation. Borenstein and Thyberg describe this behavior, stating that programmers and end users often believe that "a fundamental trade-off exists between easy-to-use, novice-oriented programs... and very powerful and customizable expert-oriented programs" [3]. They suggest three requirements for a user interface that supports both novice and expert users at the same time [3]:

1. a default interface tailored to novices,
2. a set of features and options not immediately visible or enabled for new users, and
3. a smooth, obvious, and easy-to-use mechanism for new users to gradually learn

[1]Many popular web-based e-mail sites (e.g., Google's Gmail, MSN's Hotmail, Yahoo! Mail) display the login form on their opening page.

these additional features and options.

The new SFM interface attempts to implement these three requirements. The initial interface includes only four immediate sections: "List Masters", "Edit Profile", "Logout", and "Help". These sections suffice for novice users. It provides additional features in the "Edit Profile" section, with each option initially disabled. A brief description accompanies each additional option, and as users enable them, new sections appear in the primary menu bar. This interface allows users to find a middle ground in terms of functionality, rather than having either the novice or expert extreme.

The previous interface used many pop-windows once subscribers signed into their accounts. Pop-up windows are popular for advertisements; due to their popularity, some web browsers and web browser add-ons indiscriminately block them. On these systems, users had to disable their pop-up blocker before accessing the web interface. Without doing so, the site would appear to be broken, and depending on the blocker, it may not provide obvious feedback to the user that it is blocking pop-up windows. The numerous pop-up windows also created an accessibility problem for visually-impaired users. These users rely on screen readers, software to read the contents of their screen. In the presence of numerous pop-up windows, these users could easily become disoriented. The new interface eliminates most pop-up windows; instead, users navigate using a menu bar at the top of the page. The interface still uses pop-up windows in two instances:

1. viewing the help, where it is desirable to view both the content in question and the explanation of that content at the same time, and
2. viewing messages in the challenge log, where displaying them in a separate window allows users to maintain marked (i.e., checked) entries in the challenge log.

In these two instances, pop-up windows appear to increase usability, rather than detract from it.

5.1.2 Link Panel

At any given time, subscribers have many open aliases. Any open alias is susceptible to abuse as it will accept e-mail from any sender. In most cases, they never see abuse. However, in cases where they are given to untrustworthy individuals, abuse is certainly possible. Before the advent of the link panel, subscribers could recover an abused alias in several steps:

1. connect to the SFM web site,
2. go to the "Login" page,
3. log in to their account,

4. go to the list of aliases,
5. find the abused alias, and finally
6. remove the personalization to the abuser and close the alias.

Recovering an alias from abuse required several steps before the addition of the link panel.

With the link panel, recovering an abused alias is trivial. Users must first enable it in their profile via the "Edit Profile" page. After doing so, the SFM *filter* script appends the link panel to all delivered messages. To append the link panel, it modifies the original message. In particular, it

1. creates a new message using the MIME-type `multipart/mixed` with two parts,
2. copies the original message into the first part, using the message's original MIME-type for this part, and
3. generates an HTML document for the second part and uses the MIME-type `text/html`.

The HTML portion contains the link panel within an HTML table.

Although the initial purpose of the link panel was recovering abused aliases, its addition to a message allows for additional options. The link panel contains links to

1. log a user into the SFM web-based user interface,
2. view the alias that the message arrived on,
3. view the master that the message arrived on,
4. close the alias to prevent further personalization,
5. block the sender from using the alias, and
6. close the alias and block the sender from using the alias.

Activating one of these links displays a log in screen on the SFM web site. After subscribers enter their password, SFM completes the requested action.

Before implementing the link panel, we considered alternative options. One option involved building a plug-in for one or more e-mail clients. Using the plug-in, users could simply click on a button to perform an action on the selected message. This technique has the advantage of not modifying the message, but at the same time, it limits the functionality to only select e-mail clients. The link panel works in most e-mail clients, and for that reason, it was the chosen solution.

Several security issues could arise with the inclusion of the link panel. The discussion in Section 5.2.2 addresses these potential issues.

5.1.3 Blocked Senders List

Implementing a recover alias option in the link panel may initially seem trivial. To recover an abused alias, such an option might remove the offending personalization

and close the alias. Unfortunately, this technique fails because SFM simplifies personalization lists.

Immediately after updating a personalization list, SFM attempts to remove *equivalent* addresses and domains to simplify it. Equivalent addresses have the same master and top two components in the host address. For example, the following addresses are equivalent:

```
alice.smith@ualberta.ca
bob.smith@ualberta.ca
bob.smith@cs.ualberta.ca
```

Given equivalent addresses, SFM stores only the shortest one. In the previous example, SFM stores only `bob.smith@ualberta.ca`. Equivalent domains overlap when one domain is a sub-domain of another. All domains must have at least two parts. For example, the following domains are equivalent:

```
ualberta.ca
cs.ualberta.ca
engineering.ualberta.ca
```

In this case, SFM maintains the most-general and shortest one: `ualberta.ca`.

Given the simplification of the personalization list, simply removing the sender does not work. One or more other addresses may be equivalent to the sender, and removing the sender essentially removes all of these equivalent addresses as well. Our solution is the introduction of a new list, the blocked senders list. This list does not use simplification (other than exact duplicates). Furthermore, the blocked sender list supersedes the personalization list.

Blocking a sender simply prevents a message from being accepted by an alias if the sender matches an address in the blocked senders list. This feature is not intended to globally blacklist a sender; instead, it simply blocks a specified sender from a specified alias. When users write to aliases where they are blocked, SFM's behavior is the same as writing a closed alias not personalized to the sender – SFM sends a challenge.

5.2 Security

Security has many facets, and in a system as large as SFM, it encompasses many areas. This section focuses on several select topics.

5.2.1 General

When properly set up on a server, SFM poses little threat to the server itself. All SFM components run as a non-root user (i.e., `sfmowner`). If a severe bug is discovered in an SFM component, it will most-likely effect only the files (scripts, databases, mail, etc.)

stored in the SFM user's home directory. The SFM code base contains over 9500 lines of Tcl code with comments and blank lines removed. A code base of this size presents a real possibility for bugs.

5.2.2 Link Panel

A link panel can contain up to six links. The six links follow a standard format:

script_address?plaintext_argument&encrypted_argument

The *script_address* is the Uniform Resource Identifier (URI) for the *https* Tcl script. The plaintext argument contains apparently random information. The system encrypts *encrypted_argument* on a user-by-user basis using a key stored in each user's profile. Decrypting it requires access to this key. The plaintext argument contains a random user identifier, different from the username, that maps directly to the username.

With the introduction of the link panel, we decided that the server should attempt to remove the links from outgoing e-mail. When users reply to a message, that reply is often sent as plaintext, HTML, or both, depending on the configuration of the e-mail client. Consequently, both must be removed from the message. To make this happen, SFM marks the link panel within the message.

The source code for the link panel looks like

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title></title>
  </head>
  <body>
    <div id=MARK>
      <center>
        <table width="500" border="1" cellpadding="5" cellspacing="0">
          <tr>
            <td>
              <font size="1" color="#FFFFFF">MARK<--keep</font><br>
              ...
              <font size="1" color="#FFFFFF">keep-->MARK</font>
            </td>
          </tr>
        </table>
      </center>
    </div>
  </body>
</html>
```

where MARK is the result of encrypting the user's secret code with that secret code. The ... indicates where the actual content of the link panel appears. The MARK in the tag suffices to remove the panel from the HTML variant. Two additional MARKs appear inside <font> tags. In HTML-rendered form, these font tags hide the MARKs

by displaying them in white text. In the plaintext version, they suffice to remove the link panel from that variant.

Each link in the link panel is an HTTP request. Most attacks on web applications pass malicious input within such a request [42]. As described in [42], the top threats include

- code injection: running arbitrary code,
- session hijacking: capturing an authentication token (e.g., the session ID) and then taking control of another user's session,
- identity spoofing: assuming the identity of a legitimate user,
- parameter manipulation: modifying the arguments in a query string,
- network eavesdropping: capturing data sent between the web client and server, and
- information disclosure: trying to cause an exception of the web server in the hope that it will expose information useful in an attack.

SFM takes precautions to help prevent these attacks. To address code injection, SFM (a) immediately validates a link's arguments and (b) does not output any of them without encoding them first. Session hijacking does not apply to the link panel, since no session exists at this point. With respect to identity spoofing, all data transfer involving passwords occurs over encrypted (SSL) connections, reducing the chance that an attacker will obtain a user's password. Manipulating the two parameters is made difficult by (a) using the first parameter only to access the username and (b) encrypting the second parameter. Even if an attacker managed to manipulate these parameters, little is gained since the server still requires a password before granting access to an account. To counter network eavesdropping, connections involving the link panel occur over an encrypted (SSL) connection. To address information disclosure, SFM generates generic error pages when it encounters an error while processing links.

Although links are not easily vulnerable to network eavesdropping, they may be exposed in two instances. When a user replies to or forwards a message containing the link panel, SFM cannot guarantee its removal. The current version attempts to remove both plaintext and HTML variants of the link panel, but if the sender alters it, it may not be automatically removed. When a user activates a URL on a public computer, that URL may remain in the web browser's URL history. In both instances, the links contain no private user information (e.g., username or e-mail addresses) to protect a user's privacy.

5.2.3 Preventing Abuse of *smtp*

In the implementation of our outgoing mail (SMTP) server, we take precautions to prevent abuse while maximizing the number of compatible e-mail clients. Our SMTP server does require explicit authentication. Instead, it uses a less obvious method that is more compatible with e-mail clients. When users configure their client, they add an SFM-specific PIN to their e-mail address. Our custom SMTP server rewrites their e-mail address and removes this PIN when it forwards the mail.

Furthermore, our SMTP server prevents abuse from new, unverified users. If a user creates an account but fails to verify it, our SMTP server will not accept his/her messages. The simple reason behind this is that a user must have a master for the server to accept messages. New unverified users have not yet created their first master.

If enabled in the SFM configuration, the SFM server will allow untrusted users to create accounts. These users are outside of the trusted domain. For these users, the system imposes restrictions on the number of outgoing messages. For example, they may only be allowed to send 10 messages per day; such a restriction makes our accounts useless to spammers.

To improve our existing system, we could require SSL user authentication. Unfortunately, this reduces the number of compatible e-mail clients while complicating the technical details of our system.

5.3 Challenge/Response Mechanisms

When SFM rejects a message, it replies to it and asks the sender to resend to a new e-mail address (a mail channel). The ability to accurately distinguish human from computer senders is paramount in this automation of mail channels. Several techniques can communicate a mail channel in an e-mail message.

An obvious method uses plaintext in the message body. With a sufficiently small user-base, spammers would see little benefit from defeating even a simple system. However, widespread use would give spammers greater incentive and eventually a plaintext system would no longer protect its subscribers from abuse. With this technique, spammers could automatically scan the message body, identify the alias, and resend their spam.

Instead of communicating the alias in the body, a system could use an attachment, writing the plaintext mail channel to an image. This adds an additional level of sophistication to the challenge; spammers could no longer simply scan the plaintext body. In this case, a popular system would cause spammers to use Optical Character Recognition (OCR) technology to decipher the mail channel. Existing OCR techniques are very

Figure 5.1: An example CAPTCHA image used by SFM.

accurate at recognizing plaintext in images.

The technique used to communicate mail channels requires more sophistication. It must be capable of differentiating humans from computers.

5.3.1 Communicating Mail Channels Securely

SFM must challenge senders in a way that humans can identify the mail channel but computers cannot. The CAPTCHA project provides a solution.[2] Although computers are good at solving plaintext encoded in images, their accuracy dramatically decreases with suitably obfuscated text.

A strong CAPTCHA image incorporates two key problems: segmentation and recognition [5]. The former involves locating characters within an image in the correct order and the latter involves identifying individual characters. Of these two problems, segmentation is more difficult for computers.

Results in [7] indicate that the difficulty of the segmentation problem determines the overall difficulty of the challenge. After solving segmentation, computers can use machine learning to solve single-character recognition problems with greater accuracy than humans [5]. An effective CAPTCHA image contains a difficult segmentation problem.

The current version of SFM uses an early technique to generate its images (e.g., Figure 5.1). As seen in the image, solving this segmentation problem is trivial; the characters appear predictably and are well-isolated. Although suitable several years ago, this technique of encoding images no longer reliably differentiates between humans and computers.

Although SFM's CAPTCHAs are weak, its user base is currently small. As such, no attacker is likely to exploit the weaknesses of these images at present. Even so, a future version of SFM should include improved CAPTCHA images. The SFM component that produces the current challenge is isolated and can be easily changed.

[2]CAPTCHA is an abbreviation for completely automated public Turing test to tell computers and humans apart.

5.3.2 Accessibility

Using images that computers cannot decode introduces an accessibility issue. Visually-impaired users often rely on screen readers to navigate their computer. When receiving a challenge, these screen readers are unable to decipher it. Without taking special precautions, visually-impaired users could never send messages to SFM subscribers.

Researchers are working on an aural equivalent to the visual CAPTCHA [4]. Preliminary results suggest that these audio-based CAPTCHAs are not yet ready for deployment. When such a test matures, the computer can read the alias to the user while using distortion to prevent automatic recognition.

5.3.3 Internationalization

Another issue arises with respect to internationalization. SFM sends the current challenges in English; for foreigners who do not understand English, this may create a problem. For the system to work internationally, it needs to support multiple languages.

The system could select one or more challenge languages based on (a) the content of the message, (b) the originating e-mail address and/or IP address, or (c) settings in the SFM subscriber's profile. Of these options, the third will likely prove most reliable. An SFM subscriber will know only a few languages. People corresponding with that recipient will likely know one of the same languages. Subscribers could select one or more languages in which SFM should send challenges.

5.4 Storage Requirements

A publicly-accessible server hosts a stable version of the Spam Free e-Mail (SFM) service.[3] It went online in 2003, and it now has 120 registered users. Data extracted from this server provide the foundation for calculations in this section.

The service's 120 registered accounts use it to varying extents. We group the accounts into four distinct categories:

- suspicious: accounts that show unusual usage reminiscent of experiments rather than real use,
- active: accounts with at least one message sent or received between February 5, 2006 and March 7, 2006 (a 30-day period) and an alias created before February 5, 2006,[4]
- inactive: accounts with a master, but not classified as *active*, and

[3] http://sfm.cs.ualberta.ca/

[4] Challenges sent by SFM automatically are insufficient to count as a message sent.

Table 5.1: SFM uses several databases to store user information. Some of them exist globally, where all subscribers share a single database. Others exist per user, where each user's directory contains several databases.

Database	Scope	One record per...
CLEANER	global	varies
MASTERS	global	master
RANDOMIX	global	user
USERS	global	user
ALIASES	user	alias
BLKIX	user	blocked sender
MIDS	user	outgoing message
SNDIX	user	known sender

- useless: accounts that exist but have no masters.

Two users show unusual usage and are classified as *suspicious*. One of them created 995 aliases in a single session of 28 minutes and 41 seconds. The other created 700 aliases divided across three distinct sessions. Nine users belong to the *active* category, and 86 users appear *inactive*. The remaining 23 users have accounts with no masters, making their accounts *useless*.

SFM stores data in Berkeley DB hash databases. In SFM, these databases have two scopes: user and global. The user databases reside in a user's directory and only contain records for that user. The global databases contain records for all users in the system. Table 5.1 lists the databases and their scope along with what constitutes a record.

Two independent methods can estimate the storage requirements for a user. The first uses database file sizes, as observed in the public server. The second uses a method outlined in the Berkeley DB Reference Guide. This section explores both methods and shows how their results differ. In doing so, it also provides some insight into users' data.

5.4.1 Using File Sizes

For a given database, the UNIX command `ls` displays its size and the Berkeley DB command `db_stat` summarizes its statistics, which include the number of records. Its size may be greater than its disk usage; Berkeley DB allocates hash databases as sparse files. To allow for consistent comparisons between the two techniques, this section reports file size rather than disk usage. Berkeley DB databases only grow, so the measured sizes reflect the maximum historic size of the database. After deleting records, a hash database does not shrink; instead, Berkeley DB reuses the free space whenever possible.

Determining each users' contribution involves a simple division operation after tak-

ing some precautions. With the user databases, calculations only include *active* users. With global databases, the precautions vary case by case. With one exception, these calculations include all overheads, some of which are initial rather than marginal. In a larger system, more users share the initial costs and their per user effect will be lower.

5.4.2 Using Formulas

The Berkeley DB Reference Guide presents a method for roughly estimating database size based on data size.[5] Its formulas do not account for (a) an apparent minimum database size and (b) pages being allocated in power-of-two chunks. When presented here, the original formulas are expanded to account for both.

Berkeley DB stores data in pages (or blocks). If an application does not explicitly set the page size, Berkeley DB chooses it based on the filesystem I/O block size. SFM does not explicitly set the page size (b_p). When Berkeley DB uses a page, 26 bytes of it are overhead. The page size less this overhead results in the useful bytes per page (b_u). When storing records, Berkeley DB uses a further overhead of 6 bytes per key/data pair (b_o).

Let b_d be the bytes of data, b_o be the bytes of overhead per key/data pair, b_p be the bytes per page, n_p be the number of pages, b_r be the bytes in a record (mean), n_r be the number of records, and b_u be the bytes of useful space per page. The estimated number of bytes required to store the database (b) is

$$\begin{aligned} b_d &= n_r \times (b_r + b_o) \\ n_p &= \begin{cases} \frac{b_d}{b_u} & b_d > 2 \times b_u; \\ 2 & \text{otherwise.} \end{cases} \\ b &= b_p + 2^{\lceil \log_2 n_p \rceil} \times b_p \end{aligned} \tag{5.1}$$

On the SFM production server, the page size (b_p) is 4096 bytes, making the useful bytes per page (b_u) be 4070 bytes.

According to the Berkeley DB Reference Guide, these formulas do not consider several important factors. They do not account for overflow records, wildly variable key and data sizes, and changes in the page-fill factor over time. Furthermore, they assume that the hash function evenly distributes keys among hash buckets. Finally, sparse hash tables may result from overflow pages and duplicate pages being allocated only at specific points in the file (required for contiguous allocation).

5.4.3 Findings

Table 5.2 highlights data for the user databases that belong to SFM's nine *active* users. Before collecting these data, we ran the command `reindex` in SFM's administrator

[5] http://www.sleepycat.com/docs/ref/am_misc/diskspace.html

Table 5.2: File and record sizes, in bytes, for the four user-specific databases belong to SFM's nine active users.

		ALIASES	BLKIX	MIDS	SNDIX
	Files	9	9	9	9
	Users	9	9	9	9
Files	Min.	12,288	12,288	12,288	12,288
	25 %ile	77,824	12,288	12,288	12,288
	50 %ile	155,648	12,288	12,288	49,152
	75 %ile	487,424	12,288	12,288	176,128
	Max.	1,241,088	12,288	77,824	339,968
	Mean	323,129	12,288	19,570	103,765
	File size	2,908,160	110,592	176,128	933,888
	Per user	323,129	12,288	19,570	103,765

		ALIASES	BLKIX	MIDS	SNDIX
	Records	4462	2	37	7305
	Users	9	9	9	9
Records	Min.	94	33	52	11
	25 %ile	128	33	54	26
	50 %ile	145	35	54	35
	75 %ile	170	37	54	54
	Max.	82,574	37	56	1861
	Mean	240	35	54	58
	Total data	1,069,023	70	1,997	421,839
	Est. size	1,216,512	110,592	110,592	626,688
	Per user	135,168	12,288	12,288	69,632

interface to rebuild the BLKIX and SNDIX local databases. Only two active users blocked a sender, resulting in only two copies of the BLKIX database. However, as soon as a user receives a message, SFM creates an empty BLKIX database; for that reason, we assume a BLKIX database for each active user.

The file size data show that the ALIASES and SNDIX user databases are the most significant, accounting for roughly 93% of the data. Both of these databases contain a significant number of records. The ALIASES database contains a record for each of a user's aliases, while the SNDIX database contains a record for each of a user's contacts.

When looking at record sizes, the maximum record sizes for ALIASES and SNDIX are 344 and 34 times their mean, respectively. ALIASES records can have a large data portion when a personalization list contains many senders. This situation occurs when an open alias regularly receives mail from different e-mail addresses. Some of our subscribers receive regular newsletters, and the sender uses a different e-mail address with each mailing. SNDIX records can have a large data portion when a subscriber's contact sees many aliases. Consider an SFM subscriber who regularly sends mail to different groups of people, and one contact is a member of most of those groups. Since SFM creates a new alias for each unique group, this contact will receive many aliases.

Comparing the "per user" rows in Table 5.2 for both the file and record method show

Table 5.3: Both file sizes (bytes) and record sizes (bytes) for the four global databases.

		CLEANER	MASTERS	RANDOMIX	USERS
	Files	1	1	1	1
	Users	9	97	120	120
Files	File size	90,112	24,576	12,288	61,440
	Per user	10,012	253	102	512

		CLEANER	MASTERS	RANDOMIX	USERS
	Records	351	120	4	120
	Users	9	97	120	120
Records	Min.	24	26	33	79
	25 %ile	102	51	35	107
	50 %ile	102	56	36	119
	75 %ile	102	64	37	142
	Max.	183	107	37	15641
	Mean	95	58	36	258
	Total data	33,391	6,990	142	30,901
	Est. size	69,632	12,288	12,288	36,864
	Per user	7,737	127	102	307

that the two methods differ significantly. For all of the databases, the formula approach provides estimates that are less than or equal to the file size approach. The greatest difference occurs with the ALIASES databases, where the actual sizes are roughly 2.4 times larger than those predicted by the formula. The formula's predictions are closer to the median than the mean, suggesting that in a sense they may still reflect the average user. A variety of reasons may explain the lower formula estimates in this data set (outlined in Section 5.4.2). In addition to those observations, the formulas use the current database content whereas the file sizes of ALIASES and MIDS reflect their historic maximum size.

Table 5.3 highlights data from SFM's global databases. Before collecting these data, we ran the command `reindex` in SFM's administrator interface to rebuild the CLEANER and RANDOMIX databases. The CLEANER database was then recreated using only active users. The MASTERS database contained a static number of placeholders to prevent users from choosing masters that already have an account on the system. We removed these placeholders since their data length of one byte unrealistically skewed the results. Only 97 of SFM's users created masters, so for that database, the calculations do not consider *useless* accounts. The status of an account (i.e., suspicious, active, inactive, or useless) has negligible impact on the RANDOMIX and USERS databases; for that reason, calculations using these databases consider all users.

The number of RANDOMIX records is relatively small when compared with the other databases. RANDOMIX records only exist for users who enable the link panel; few users have enabled it, so few records exist in this database. The maximum record

size in the USERS database is approximately 61 times the mean. SFM stores a user's legacy patterns in the USERS database; users with many legacy patterns have large USERS records.

Comparing the "per user" rows in Table 5.3 for both the file and record method show that these values differ less significantly than with the user databases. The largest difference occurs with the MASTERS database, where each user's contribution is roughly twice the estimated value. For all of the databases, the formula-based estimate is less than or equal to the file size approach. These differences are possibly explained by the same observations as with the user databases.

In summary, these data can quantify the storage requirements for an SFM subscriber. The formulas produce a per user estimate of 237,649 bytes. Using actual file sizes produces a much larger estimate of 469,631 bytes.

These measurements do not include space used by the "Challenge Log". Only four of the active users currently use the challenge log, where its average size is 447,488 bytes. This mean suggests that the challenge log is a significant contributor to the storage requirements. Unfortunately, insufficient data exists to accurately estimate its size for an average user.

Chapter 6

Conclusion

This thesis described an automation of mail channels in the context of the Spam Free e-Mail (SFM) service, an effective solution to preventing unsolicited bulk e-mail (spam). The system automatically creates new e-mail addresses (mail channels) for mail arriving on published e-mail addresses. It announces them to senders using CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) images. This technique capitalizes on a fundamental difference between human and computer senders. Human senders spend time composing their messages, and this investment makes them care about the delivery. They will more often than not take the extra time to resend their message to the address encoded in the image. On the other hand, automated senders are unable to respond to the challenge and their messages never reach SFM subscribers.

Although the system effectively blocks all bulk mail, it still provides the flexibility for users to solicit bulk mail and sign up for mailing lists. Using its extensive web-based user interface, users can manually create mail channels. By giving them to a bulk mailer, messages can arrive unchallenged.

6.1 Contribution

This thesis contributes:

- a summary of the fundamental techniques, as well as supporting techniques, that help prevent e-mail abuse,
- a history of both challenge/response techniques and the concept of mail channels,
- a detailed description of the Spam Free e-Mail (SFM) service from both a user and technical perspective, and
- a discussion of various design decisions, security issues, challenge/response issues, and an estimate of SFM's storage requirements.

6.2 Future Work

Even though SFM is a fully-functional, production quality system, some opportunities exist for further work. These opportunities fall into two areas: research and implementation. Research opportunities may benefit other challenge/response and mail channel systems. The current SFM system can immediately benefit from the other opportunities.

6.2.1 Research

The research opportunities for future work include the development of improved CAPTCHA techniques. Eventually machines will overcome the current techniques, and if a new CAPTCHA technique cannot replace them, this automation of mail channels will fail. Researchers must proactively identify and produce new CAPTCHA techniques.

The idea of challenge/response is new to many Internet users. Researchers could study their reaction to these techniques when used with e-mail. For example, some rumors suggest that people may (a) be offended when they receive a challenge or (b) ignore challenges. Further research could determine the public's response to these techniques and possibly provide a new direction for their development.

An automation of mail channels, such as that used in SFM, could be combined with a filter. The system could accept messages on a published address only when a filter accepts them. Otherwise, it could send a challenge with a new mail channel. Messages arriving on mail channels could bypass the filter. While such a system would no longer eliminate spam entirely, it would improve on existing filtering techniques and challenge senders less than the current implementation.

6.2.2 Implementation

As mentioned in Section 5.3, the use of visual CAPTCHA images creates an accessibility issue. Visually-impaired users are unable to answer challenges and thus communicate with SFM users. A future version of SFM could incorporate audio CAPTCHA techniques, even if they are not yet fully developed.

From a security standpoint, the current version of SFM stores plaintext passwords in a database. If this database were ever compromised and the passwords stolen, every user account would be accessible. Instead of storing plaintext passwords, a future version of SFM should store a message-digest version of the password. Each password would be passed through a message-digest algorithm, such as SHA, before storage.

Bibliography

[1] Steven M. Bellovin. Spamming, phishing, authentication, and privacy. *Commun. ACM*, 47(12):144, 2004.

[2] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[3] N. S. Borenstein and C. A. Thyberg. Power, ease of use and cooperative work in a practical multimedia message system. In *Computer-Supported Cooperative Work and Groupware*, pages 99–132. Academic Press Ltd., London, UK, 1991.

[4] Tsz-Yan Chan. Using a text-to-speech synthesizer to generate a reverse Turing test. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 226–232, 2003.

[5] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In *Proceedings of the Third Conference on E-Mail and Anti-Spam*, July 2005.

[6] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (HIPs). In *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 711–720, New York, NY, USA, 2005. ACM Press.

[7] Kumar Chellapilla and Patrice Y. Simard. Using machine learning to break visual human interaction proofs (HIPs). In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 265–272. MIT Press, Cambridge, MA, 2005.

[8] Christopher Alan Cobb. Method and system for filtering electronic messages. US Patent #6,119,102, March 6 2001. Filed August 26 1997.

[9] David H. Crocker. Standard for the format of ARPA Internet text messages. Request for Comments 822, August 1982.

[10] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. An open digest-based technique for spam detection. In *Proc. of the 2004 International Workshop on Security in Parallel and Distributed Systems*, San Francisco, CA USA, September 15–17 2004.

[11] Mark Delany. *Domain-based Email authentication using public-keys advertised in the DNS (DomainKeys)*. Yahoo! Inc, September 2005. Internet Draft.

[12] Peter J. Denning. ACM president's letter: electronic junk. *Commun. ACM*, 25(3):163–165, 1982.

[13] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147, London, UK, 1993. Springer-Verlag.

[14] Federal Trade Commission. You've got spam: How to "can" unwanted email, April 2002. http://www.ftc.gov/bcp/conline/pubs/online/inbox.htm (accessed Feb. 13, 2006).

[15] Mike France. Commentary: Needed now: Laws to can spam. *Business Week*, (3802):100, October 7 2002.

[16] William O. H. Freund. Bringing email services to the desktop. In *SIGUCCS '92: Proceedings of the 20th Annual ACM SIGUCCS Conference on User Services*, pages 101–103, New York, NY, USA, 1992. ACM Press.

[17] Eran Gabber, Markus Jakobsson, Yossi Matias, and Alain J. Mayer. Curbing junk e-mail via secure classification. In *FC '98: Proceedings of the Second International Conference on Financial Cryptography*, pages 198–213, London, UK, 1998. Springer-Verlag.

[18] W. Gansterer, M. Ilger, P. Lechner, R. Neumayer, and J. Strauss. Anti-spam methods - state of the art. Technical report, University of Vienna, 2005. Institute of Distributed and Multimedia Systems, Faculty of Computer Science.

[19] Government of Canada. Competition Act, 1985. http://laws.justice.gc.ca/en/C-34/ (accessed Feb. 13, 2006).

[20] Government of Canada. Criminal Code, 1985. http://laws.justice.gc.ca/en/C-46/ (accessed Feb. 13, 2006).

[21] Government of Canada. Personal Information Protection and Electronic Documents Act, 2000. http://laws.justice.gc.ca/en/P-8.6/ (accessed Feb. 13, 2006).

[22] John Graham-Cumming. People and spam. MIT Spam Conference, presentation slides, January 21 2005. http://www.jgc.org/pdf/spamconf2005.pdf.

[23] Robert J. Hall. Channels: Avoiding unwanted electronic mail. In *DIMACS '96: Proceedings of the 1996 DIMACS Symposium on Network Threats*. American Mathematical Society, 1997.

[24] Robert J. Hall. How to avoid unwanted email. *Commun. ACM*, 41(3):88–95, 1998.

[25] Robert J. Hall. Communications addressing system. US Patent #5,930,479, July 27 1999. Filed October 21 1996.

[26] Saul Hansell. Postage due, with special delivery, for companies sending e-mail to AOL and Yahoo. *New York Times*, Late Edition, East Coast:1.25, February 5, 2006.

[27] Jeffrey Nelson Heiner. Filter-in method for reducing junk e-mail. US Patent #6,112,227, August 29 2000. Filed August 6 1998.

[28] Lawrence Hughes. *Internet e-mail: Protocols, Standards, and Implementation*. Artech House, Inc., Norwood, MA, USA, 1998.

[29] G. Hulten, J. Goodman, and R. Rounthwaite. Filtering spam e-mail on a global scale. In *WWW Alt. '04: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pages 366–367, New York, NY, USA, 2004. ACM Press.

[30] Geoff Hulten, Anthony Penta, Gopalakrishnan Seshadrinathan, and Manav Mishra. Trends in spam products and methods. In *Proceedings of CEAS*, Mountain View, CA, July 2004.

[31] Industry Canada. An anti-spam action plan for Canada, May 2004. http://e-com.ic.gc.ca/epic/internet/inecic-ceac.nsf/en/h_gv00246e.html (accessed Feb. 13, 2006).

[32] Industry Canada. Stopping spam: Creating a stronger, safer Internet, May 2005. http://e-com.ic.gc.ca/epic/internet/inecic-ceac.nsf/en/h_gv00317e.html (accessed Feb. 18, 2006).

[33] International Telecommunications Union. Internet indicators: Hosts, users and number of PCs. Data set, http://www.itu.int/ITU-D/ict/statistics/, 2004.

[34] John Ioannidis. Fighting spam by encapsulating policy in email addresses. In *NDSS '03: Proceedings of the Network and Distributed System Security Symposium*, 2003.

[35] J. Klensin. Simple mail transfer protocol. Request for Comments 2821, April 2001.

[36] Benjamin J. Kuipers, Alex X. Liu, Aashin Gautam, and Mohamed G. Gouda. Zmail: Zero-sum free market control of spam. In *ICDCSW '05: Proceedings of the Fourth International Workshop on Assurance in Distributed Systems and Networks (ADSN) (ICDCSW'05)*, pages 20–26, Washington, DC, USA, 2005. IEEE Computer Society.

[37] George Lawton. E-mail authentication is here, but has it arrived yet? *Computer*, 38(11):17–19, 2005.

[38] Ian Leicht. Again: Beta testers wanted for UCE mail filter. news.admin.net-abuse.misc, September 27 1996.

[39] Avivah Litan. Phishing attack victims likely targets for identity theft. Technical Report FT-22-8873, Gartner, Inc., May 4 2004.

[40] Thede Loder, Marshall Van Alstyne, and Rick Wash. An economic answer to unsolicited communication. In *EC '04: Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 40–50, New York, NY, USA, 2004. ACM Press.

[41] J. Lyon and M. Wong. *Sender ID: Authenticating E-mail.* Microsoft Corp and pobox.com, May 2005. Internet Draft.

[42] J.D. Meier, Alex Mackman, Srinath Vasireddy, Michael Dunner, Ray Escamilla, and Anandha Murukan. *Improving Web Application Security: Threats and Countermeasures.* Microsoft Corporation, June 2003.

[43] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–134–I–141, 2003.

[44] G. Moy, N. Jones, C. Harkless, and R. Potter. Distortion estimation techniques in solving visual CAPTCHAs. In *CVPR '04: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–23–II–28, 2004.

[45] Moni Naor. Verification of a human in the loop or identification via the Turing test. http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps, September 13 1996.

[46] Oxford University Press. Oxford English dictionary: The definitive record of the English language. http://www.oed.com/ (accessed Feb. 18, 2006).

[47] Pew Internet and American Life Project. January 2005 tracking survey. Data set, http://www.pewinternet.org/datasets.asp, January 2005.

[48] Steven D. Popell. *Computer Time-Sharing: Dynamic Information Handling for Business.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1966.

[49] Jonathan B. Postel. Simple mail transfer protocol. Request for Comments 821, August 1982.

[50] P. Resnick. Internet message format. Request for Comments 2822, April 2001.

[51] R. Rivest. The MD5 message-digest algorithm. Request for Comments 1321, April 1992.

[52] Ben Schneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition).* Addison Wesley, March 2004.

[53] Statistics Canada. Household Internet use survey. CANSIM II, July 8 2004. IMDB #4432, tables 3580003 (series V969799) & 3580006 (series V969883).

[54] Statistics Canada. Survey of electronic commerce and technology. CANSIM II, April 16 2004. IMDB #4225, table 3580007 (series V2649704, V2649803, V2650219 & V2650318).

[55] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4 edition, 2003.

[56] L. Weinstein. Spam wars. *Communications of the ACM*, 46(8):136, 2003.

[57] Chan Wilson. Announcing majordomo 1.94! listserv, October 1996. Majordomo-Announce@greatcircle.com.

[58] M. Wong and W. Schlitt. *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-mail*. Network Working Group, 1 edition, Jun 2005. Internet Draft.

[59] World Wide Web Consortium. HTML 4.01 specification. W3C Recommendation, December 1999. http://www.w3.org/TR/REC-html40/ (accessed Nov. 17, 2005).

[60] World Wide Web Consortium. Web content accessibility guidelines 2.0. W3C Working Draft, November 23 2005. http://www.w3.org/TR/WCAG20/ (accessed Apr. 17, 2006).

[61] Jonathan A. Zdziarski. *Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification*. No Starch Press, San Francisco, CA, USA, 1 edition, 2005.