An Automation of Mail Channels

Nicholas M. Boers and Pawel Gburzynski
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8
E-mail: {boers,pawel}@cs.ualberta.ca

***Abstract*— Mail channels allow an electronic mail (e-mail) user to have multiple points of contact, each with a potentially different policy. For example, a user may have two channels, one for personal use and one for business use. The former channel's policy may accept all senders and the latter channel's may restrict senders to those within a given company. By extending this example, an e-mail user can have a channel unique to each contact. Each of these channels may have a policy that limits its use to the particular contact. Traditionally, this scenario requires substantial administrative overhead, making it impractical. The system proposed here, the Spam Free Mail (SFM) service,[1] automates the creation of mail channels. SFM's restrictive mail channels effectively eliminate a considerable part of e-mail abuse, such as spam and phishing.**

I. Introduction

The term *channel* can describe a medium through which information travels. For example, radio and television have multiple channels (i.e., stations). Consumers isolate information of interest by selecting appropriate stations. A simple extension of the same concept creates channels for electronic mail (e-mail). They are known as *mail channels*.

Many of the existing mail channel implementations [1]–[4] require substantial administrative overhead. Users must create channels manually. The system proposed here supports the automatic creation of mail channels in two instances. Users implicitly create them when they send mail to new correspondents. External users initiate their creation when they first send mail to users of our system. We use Completely Automated Public Turing Tests to Tell Computers and Humans Apart (CAPTCHAs), also known as Human Interaction Proofs (HIPs), to limit their creation. Consequently, our system prevents much e-mail abuse.

Section II describes key components in the evolution of mail channels. Given that background information, Section III introduces the Spam Free e-Mail (SFM) service, our solution to mail channels. Section IV describes related work and its similarities with SFM. In Section V, we indicate our direction for future work. Finally, Section VI provides a summary of this paper.

II. Previous Work

In 1982, Denning [5] described a "tidal wave of electronic junk mail" entering his mailbox. To solve the problem, he envisioned multiple e-mail paths (or mail channels), each for a specific type of e-mail. The mail system would always deliver messages arriving on some paths (e.g., urgent, certified, and personal). It would filter messages arriving on all other paths. He observed that existing organizations already meet these requirements for (non-electronic) communications paths. He suggested that the research community should study these traditional paths when developing a solution to the junk mail problem.

Borenstein and Thyberg [1] address the creation of a mail channel in the context of the Andrew Messaging System (AMS). The system received a large number of e-mails asking for end-user support, prompting the creation of the Advisor service to help manage the volume. An early version of 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 channeled 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.

Hall [2], [3] extends the mail channel idea to prevent unwanted e-mail. In his system, an address's form is `user-extension-@host`. Extensions are pseudo-random text (cryptographically secure BlumBlumShub [6]) 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 an incoming or outgoing message uses an appropriate extension). His system uses three channel classes (i.e., policies): (1) send only, (2) private, and (3) public, with the possibility for additional classes. Send only channels support only outgoing e-mail, as their name suggests. Private channels maintain, and only accept mail from, a set of known correspondents. Public channels are accessible to anyone, much like a traditional e-mail address.

We see three fundamental weaknesses in Hall's system. First, users must manage channels through the PCA's administrative interface. The manual opening, closing, creating, deleting, or switching of channels can be time consuming, particularly for a large number of channels. Second, messages rejected on a channel (e.g., an unauthorized sender on a private channel) result in a bounced "no permission" message. Suppose User_a uses Hall's system and regularly communicates

[1] See http://sfm.cs.ualberta.ca.

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. Third, 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.

Single-Purpose Addresses (SPAs) [4] also manipulate an address extension to create a mail channel. Their presented syntax is similar to the AMS, `user+`*`extension`*`@host`, but can be changed easily on a per-user basis. Instead of storing extensions in a database [2], [3] that maps them to a policy, SPAs encode the policy in the extension. This novel approach removes the dependency on a database to link channels with policies. To prevent tampering and keep the policies private, the system encrypts the extensions.

We see several issues with SPAs. First, users of SPAs must manually create them. Like the previous system, the administrative burden is on the SPA user. Second, this scheme is not for general use (i.e., "no one is expected to type in such addresses" [4]). The addresses' lengths make them appropriate for automated systems and not business contacts, family members, or friends. Third, the SMTP [7] standard imposes a 64-character maximum length on an address's user name (or local-part) portion. This maximum bounds the expressiveness of the policies.

III. The Spam Free Mail (SFM) Service

The Spam Free e-Mail (SFM) service represents an evolution in mail channels. It addresses several of the deficiencies in previous systems and generally improves the usability. [8], [9] describe early prototypes of this system. This paper presents the many changes in recent versions, while assuming no prior knowledge of SFM.

A. New Users

SFM supports two account types. The first, a *forwarding* account, redirects mail accepted by a mail channel to a permanent e-mail address. Users specify this e-mail address when they subscribe. The second, a *hosted* account, stores mail accepted by a mail channel in a mailbox hosted on the SFM server. With this account type, users need not have a pre-existing e-mail address.

For both account types, the SFM users are "virtual" from the viewpoint of the SFM host. This means that they do not have actual user accounts on the SFM server, and thus would not be normally recognized as legitimate recipients by a traditional e-mail delivery system.

New users of the *forwarding* variety subscribe to the system through a Web-based interface. SFM implements this interface in a Tcl [10] script (see the *https* component in Figure 1), executed by the Apache [11] web server. This Tcl script is the first of the three primary components of any SFM installation.

When a user creates an account, SFM stores the user information in a BerkeleyDB [12] database. At the same time, SFM (1) schedules the account for deletion and (2) sends a secret code to the subscriber's permanent e-mail address. All new users must verify their accounts (with the secret code) to prevent their deletion.

In the current version of SFM, new users of the *hosted* variety cannot subscribe through the Web-based interface. Instead, administrators must manually create their hosted accounts.

B. Overview

After subscribing to the service, users first create one or more new e-mail addresses (masters). They never receive a message sent to a master; these addresses can be openly published. Messages arriving at a master receive an automated reply from the server, asking the sender to prove their humanity. Specifically, the reply challenges the sender to resend the message to a new mail channel (an e-mail address known here as an alias). We will use the terms *mail channel* and *alias* interchangeably. When the alias receives its first message, it becomes personalized to the sender. For a user-specified duration, it will accept e-mail from all addresses, adding them to its list of senders (personalization list). After this duration, it closes and then only accepts messages from senders in its personalization list. If an alias is unpersonalized at the time that it closes, the alias is deleted. Note that aliases need not close automatically (i.e., they can have an infinite open time).

Users can manually create aliases tailored to specific senders in SFM's extensive Web-based user interface. This ability allows for the reception of mass-mailed e-mail (e.g., mailing lists and solicited advertisements).

C. Incoming Mail

Incoming mail first arrives at the `qmail` [13] message transfer agent (MTA). For local users (i.e., users with true accounts on the machine), `qmail` delivers the mail to their mailboxes. For unknown users (remember that all SFM users are virtual and hence unknown), `qmail` pipes the mail into a Tcl script (see the *filter* component in Figure 1). This Tcl script is the second of the three primary components of any SFM installation.

The Tcl script first analyzes the e-mail's recipient address. SFM addresses use the syntax *`extension`*`.user@host` (or *`alias`*`.master@host`). The analysis results in one of several possible actions (summarized in Table I). If the master is not found, the system bounces the message and specifies "no such user" as the cause. This action is consistent with the behavior of existing mail servers. Upon finding the master, the alias may or may not exist for that particular master. If the alias does not exist, the system bounces the message, including an alias (new or existing) with the message. The bounce uses an existing alias if the sender already acquired a valid alias for the master. If the alias exists, then further analysis is required.

For messages addressed to a known master and alias, the SFM server must perform further analysis (Table II). Associated with each alias is a personalization list (a list of

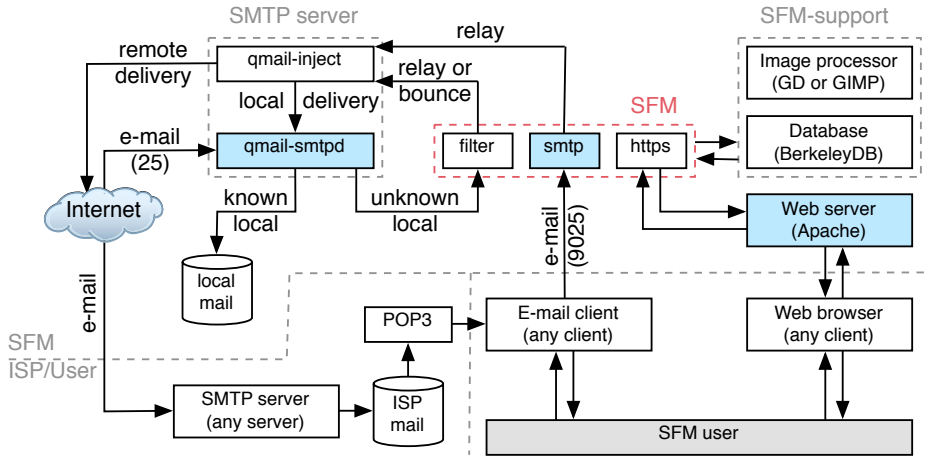

Fig. 1. The SFM system from the viewpoint of a *forwarding* account.

TABLE I

POSSIBLE ACTIONS DEPENDING ON WHETHER SFM FINDS THE MASTER AND ALIAS THAT IT PARSES FROM AN E-MAIL ADDRESS.

Master	Alias	Action
not found	N/A	reject: bounce no such user
found	not found	reject: bounce new alias
found	found	*further analysis*

TABLE II

FURTHER ANALYSIS (FROM TABLE I) WHEN A MASTER AND ALIAS ARE FOUND. THIS TABLE DESCRIBES POSSIBLE ACTIONS GIVEN THE STATE OF AN ALIAS. LET S BE THE SENDER ADDRESS, L_P BE THE SET OF PERSONALIZATION ADDRESSES, AND L_B BE THE SET OF BLOCKED ADDRESSES.

$S \in L_P$	$S \in L_B$	State	Action
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

known senders) and a blocked list (a list of blocked senders). Furthermore, the alias can be in one of two states: *open*, accepting messages from new senders, or *closed*, accepting only messages from known senders. Rejected messages always receive a bounce from the SFM server. This contrasts with filtering techniques, where rejected messages are often deleted. The bounce includes an alias whenever possible to ensure that human contacts can always reach SFM subscribers.

With the above framework, it may seem that SFM does not support automated mailers (e.g., solicited advertisements and mailing lists). These mailers could not answer the challenge (i.e., extract the alias from the bounce and resend to that alias). To solve this problem, SFM users can manually create new aliases (mail channels) using the Web interface. By creating an open alias, any human (or any machine) can send e-mail to it without hassle. If at any point one of these open channels becomes subject to abuse, it can simply be closed. By the time that this abuse occurs, the alias should be sufficiently personalized to the intended senders.

D. Outgoing Mail

When an SFM user sends mail, they do so using the SFM Simple Mail Transport Protocol (SMTP) [7] mail server (see the *smtp* component in Figure 1) This Tcl script is the third and final primary component of any SFM installation.

The current system uses `xinetd` [14] to monitor a non-standard port (e.g., 9025) for incoming connections. Upon receiving a connection, `xinetd` passes the connection to the SFM SMTP mail server. When the SMTP server receives the message, it performs several steps. It first looks at the recipient addresses to determine whether a suitable existing mail channel exists. If it finds a suitable channel, it uses it for sending the message. It replaces the sender address in the message with the appropriate channel.

If an appropriate channel cannot be found, the SFM server generates a new channel automatically. It similarly replaces the sender address with the new channel. This technique of automatic channel creation may result in multiple aliases for a given contact. However, the contact can communicate on any one of them, so the additional aliases have no impact on the reliability of correspondence.

E. Restricting the automatic creation of channels

Upon rejecting a message, the incoming mail server automatically creates a new mail channel and communicates it to the user in a bounce. If this alias were communicated in plaintext, it would be trivial for an abuser to automatically extract it. E-mail abusers could abuse our system and automatically send messages to our subscribers.

Fig. 2. In this SFM CAPTCHA image, the initial eight letters are distorted to make optical character recognition difficult.

The SFM bounce has two goals: (1) communicate the mail channel to the user and (2) ensure only a human can decipher it. Completely Automated Public Turing Tests to Tell Computers and Humans Apart (CAPTCHAs) or Human Interaction Proofs (HIPs) provide a solution. The CAPTCHA project [15] converts plain-text into a special graphic. Properties of the graphic, such as distortion, make optical character recognition difficult.

Researchers have developed several variations of these tests [16]. Some of these variants are easy to break [17]–[20]. In fact, some of the variants are better solved by computers than by humans [21]. The particular one used in the current SFM implementation (Figure 2) is not excessively difficult to break. At this time, however, no one has exploited the weaknesses in the SFM CAPTCHAs to abuse SFM users. The CAPTCHA component of SFM, which constitutes a well-isolated module, can be replaced easily with a more secure variant if a need arises.

Using a visual challenge (i.e., CAPTCHA images) creates an accessibility issue. For example, a visually-impaired e-mail user could never answer even a simple visual CAPTCHA. Researchers are developing aural challenges [22]. Using such a challenge, the computer could read the alias to the user. The sound clip would include background noise to increase the difficultly of automatic recognition. As these aural systems improve, a future version of SFM will include this support.

F. Streamlining regular actions

Users, especially power users dealing with many open contacts that are likely to lead to abuse, may find themselves regularly performing certain actions, such as closing open aliases. To make common actions more efficient, SFM optionally appends a few HTML shortcuts (i.e., links) to forwarded message bodies. Users can enable this feature in their user profile, accessible through the Web-based user interface. When appending links, SFM converts accepted messages into the `multipart/mixed` MIME-type. The first part is the original message and an HTML table of links constitutes the second part.

The links can

- log a user into the SFM Web-based user interface,
- view the alias that the message arrived on,
- view the master that the message arrived on,
- close the alias to prevent further personalization,
- block the sender from using the alias, and
- close the alias and block sender from using the alias.

When a user selects a link, an SFM log in screen appears. The screen displays only the action and a prompt for the password. We paid particular attention to privacy in these links. Although the permanent e-mail address acts as the SFM user name (for forwarding accounts), we do not expose that address in the links or on the log-in screen. Additionally, we do not expose arguments to the action (e.g., the address to block if blocking the sender). All of these details are encrypted or otherwise hidden. After submitting the correct password, the site displays a summary for the action, if appropriate.

To further protect the privacy of users, the SFM SMTP server attempts to remove these links when processing outgoing mail. The link panel is delimited by special tags that appear in two parts within the HTML. First, a tag appears as part of a `div` tag that surrounds an HTML table. This tag is invisible to the user. Second, they appear as white text at the immediate start and immediate end of the HTML table. When viewing the message in its HTML rendered form, these tags are also invisible. The tags appear in two forms for a reason. When a mail client forwards/replies to the e-mail, it may include both an HTML and plain-text version. The tag in the `div` suffices to remove the link table from the HTML variant. The tag in white, which becomes visible in the plain-text variant, suffices to remove that variant.

G. SFM-to-SFM interoperability

Two SFM users experience no trouble establishing communication under the SFM framework. Consider the sequence of events. Suppose User_A sends a message to User_B's master, attempting to establish first contact. This message will leave User_A on a new (open) mail channel. When User_B's master receives the message, it will bounce a challenge to this new mail channel belonging to User_A. The alias will accept the message and deliver it to User_A, who can then resend the message to the new alias presented in the bounce (which is also an open alias).

H. Reclaiming an abused permanent e-mail address

For forwarding accounts, abuse of a permanent e-mail address might concern some users. SFM supports a scheme, which we call *refiltering*, whereby e-mail arriving at the old permanent address of the subscriber can be handled by our server and thus made free of abuse. This feature allows users to retain the traditional and official status of an old known point of contact.

When the server forwards a message to a permanent e-mail address, it can include a user-defined signature (a *filter cookie*) in the message headers. If a user's e-mail client receives a message with this tag, it knows that the message passed through the SFM server. If the tag does not exist, the client can forward it to one of the user's masters. Upon arriving at a master, the SFM server will then process it by sending out a challenge.

Most e-mail clients can take advantage of the refiltering features of SFM to reclaim old abused e-mail addresses. This solution works with the SFM server installed in any domain, not necessary within the domain of the old address. The necessary prerequisite on the client's side is the ability to filter messages based on keywords detected in headers and, conditionally, forward them to a special e-mail address in a

way that preserves the essential information from the original headers.

I. Demands on user expertise

While SFM offers many features to its power user, it does not intimidate a typical e-mail recipient who does not want to learn a new paradigm or change the old easy way of dispatching and receiving messages. It took us a while to arrive at the present version of the interfaces, as well as the exact semantics of channels. For illustration, a previous prototype version of SFM [9] lacked its own specific SMTP service for outgoing e-mail. To make sure that his/her identity in outgoing e-mail was consistent with the personalization of aliases, the subscriber had to send messages to a single address within the SFM domain, while passing the recipient information via special sequences in subject lines. That wasn't very friendly: the system, although useful to experts, was criticized as being cumbersome to an average user. Another and even more serious inconvenience with the old system was a large collection of arcane attributes associated with aliases and masters. The apparent need for those attributes, including patterns matched to the subject line and message body, was dictated by our efforts to account for the different types of casual contacts in a way that would make them as reliable as possible. That was before the invention of *lazy personalization*, i.e., the idea of leaving the channel open for a prescribed amount of time and letting it personalize itself. This idea simplified things enormously, while making all legitimate contacts highly reliable. The configuration of alias attributes was reduced to a trivial number of easily understood items. As viewed by a non-expert user, the system became transparent and maintenance-free.

To a casual user, most of the functionality of SFM is available through a simple and practically automatic setup involving a single master. The entire effort boils down to a few rather obvious steps involved in signing up to the service and modifying the subscriber's sender identity in the e-mail client. Following this step, the operation of sending and receiving e-mail works exactly as with a traditional mailer. The user need not worry about the web interface to SFM as long as his/her contacts are "typical." More than half of all users never bother to create aliases by hand or modify their attributes via the web interface. The system automatically allocates channels to their new contacts and renders those channels transparent to its subscribers.

A useful and simple enhancement of this simplistic (albeit effective) setup is the addition of a second master—intended for short-lived commercial contacts. SFM offers a simple (one-click) method of acquiring instant channels for various occasions, e.g., to be inserted in a web form of an electronic merchant.

IV. Related Work

Gabber et al. [23] describe a mail system that also supports the automatic generation of mail channels. Here we highlight some of the differences between their proposed system and our proposed and implemented system.

In their proposal, a user must complete a handshake before receiving a mail channel. As described, this handshake involves

1) User_a sending a message to User_b,
2) User_b requesting that User_a should incur a "cost",
3) User_a incurring a "cost" and sending proof to User_b,
4) User_b creating a mail channel and sending it to User_a, and
5) User_a re-sending the message using the new channel.

They describe the desirable properties of the "cost" and even suggest a possible function (but fail to discuss a possible implementation of it). They indicate that the cost must be unacceptable to a spammer while remaining reasonable to genuine contacts. In contrast, SFM creates a mail channel immediately, resulting in a three-step handshake. By presenting the mail channel in a CAPTCHA image, SFM essentially combines the cost with the announcement of the mail channel.

The Gabber et al. proposal suggests that users must employ an HTTP proxy and server to use their system. The user will use both of these when they need to generate mail channels (i.e., when registering on web sites). Since SFM is server-based rather than client-based, SFM users only need a Web client (e.g.., Internet Explorer, Mozilla, or even `lynx`). We believe that our requirements are more reasonable for the average e-mail user.

Gabber et al. describe a fairly complete framework, but since they did not implement the system, it lacks many details. SFM includes many additional features that evolved through actual use (e.g., additional attributes for aliases and the appending of links to messages).

Regarding the elimination of e-mail abuse, notably spam, the most popular solutions proposed in the literature are based on various flavors of contextual filtering [24]. Despite considerable efforts aimed at refining Bayesian filters [25], [26] and similar approaches based on AI techniques [27], which many people still see as an effective remedy for spam, all these solutions have proved essentially futile. Although modern spam filters do exhibit better accuracy rates than early solutions, their primary disadvantage is in the possibility of false positives, i.e., rejecting a legitimate and possibly important message. This is why they meet with reluctance in the corporate world, where such a mishap can be disastrous. As one of us argues at length in [9], contextual filtering of spam is flawed as a concept, and the mail channels approach offers a much better path to eliminating this plague.

V. Future Work

Future work on SFM may proceed in two directions. First, we will concentrate on the existing SFM system. When aural HIPs become sufficiently mature, we will add them to SFM. To ease people's transition to SFM, we may give users the ability to provide a list of valid senders on a per-master basis. Essentially, this ability will provide a white-list for each master and reduce the number of challenges after switching to

SFM. Second, we will look into the possibility of integrating mail channels with a spam filter. Our intention is to explore the possibility of eliminating false positives by focusing on those contextual aspects of a message that cannot possibly misqualify a legitimate message for a spam. While filters based on this assumption would perform poorly when working alone, there may be a way to incorporating them into SFM that will benefit our system.

VI. Conclusion

We presented a system for automating the use of mail channels. Whether writing to or receiving a first message from a new contact, SFM absorbs the complexity of re-mapping the identity of its subscriber and protecting his/her mailbox from abuse. A working production-grade system has demonstrated the effectiveness of our approach. It has completely eliminated spam from the mailboxes of its subscribers without compromising the reliability of their legitimate contacts.

The general idea of a challenge-response protocol for establishing the first contact with an e-mail recipient is often criticized as cumbersome, unfriendly, unreliable, or impolite. Some opponents of the challenge-response paradigm [28] even object to the extra traffic incurred by the rejected messages arguing that such schemes will "entangle users into bounces." Our experience does not support these worries. The protocol involves at most one challenge-response per each new channel, which is then likely used for sending a nontrivial number of messages, possibly including lengthy attachments. Of course, the amount of e-mail traffic on the network will decrease drastically when abuse becomes futile and pointless.

Our arguments in defense of the challenge-response paradigm can be concluded with the observation that, perhaps, they are not needed at all. With the proper usage patterns the number of actual challenge cases may turn out to be completely insignificant. Moreover, those users of SFM who dislike the challenge-response component can ignore it altogether and use the system for allocating pure (permanently open) mail channels. This approach is reactive: the channel can be abused, but its abuse is trivially easy to rectify when it occurs (Section III-F).

One more thing to note is that SFM requires practically no cooperation from the e-mail client, beyond some standard and popular features available with practically all contemporary mailers. The "extra buttons," that may be of interest to power users, are provided by the optional HTML links in a delivered message.

References

[1] 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*. London, UK: Academic Press Ltd., 1991, pp. 99–132.

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

[3] ——, "How to avoid unwanted email," *Commun. ACM*, vol. 41, no. 3, pp. 88–95, 1998.

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

[5] P. J. Denning, "ACM president's letter: electronic junk," *Commun. ACM*, vol. 25, no. 3, pp. 163–165, 1982.

[6] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo random number generator," *SIAM J. Comput.*, vol. 15, no. 2, pp. 364–383, 1986.

[7] E. J. Kensin, "Request for comments 2821, simple mail transport protocol," April 2001.

[8] P. Gburzynski and J. Maitan, "A comprehensive approach to eliminating spam," in *Proceedings of EUROMEDIA'03*, Plymouth, UK, 2003.

[9] ——, "Fighting the spam wars: A remailer approach with restrictive aliasing," *ACM Trans. Inter. Tech.*, vol. 4, no. 1, pp. 1–30, 2004.

[10] "Tcl developer site," Web site, 2005, http://www.tcl.tk/.

[11] "Welcome! - the Apache HTTP server project," Web site, 2005, http://httpd.apache.org/.

[12] "Sleepycat software: Berkeley DB database, native XML database, native Java database," Web site, 2005, http://www.sleepycat.com/.

[13] "qmail: Second most popular MTA on the Internet," Web site, 2005, http://www.qmail.org.

[14] "xinetd," Web site, 2005, http://xinetd.org/.

[15] "The CAPTCHA project," Web site, 2005, http://www.captcha.net/.

[16] C. Pope and K. Kaur, "Is it human or computer? Defending e-commerce with Captchas," *IT Professional*, vol. 7, no. 2, pp. 43–49, 2005.

[17] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, "Designing human friendly human interaction proofs (HIPs)," in *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM Press, 2005, pp. 711–720.

[18] K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (HIPs)," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 265–272.

[19] 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*, vol. 2, 2004, pp. II–23–II–28.

[20] 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*, vol. 1, 2003, pp. I–134–I–141.

[21] K. Chellapilla, K. Larson, P. Simard, and M. 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*, Jul 2005.

[22] T.-Y. Chan, "Using a test-to-speech synthesizer to generate a reverse Turing test," in *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, 2003, pp. 226–232.

[23] E. Gabber, M. Jakobsson, Y. Matias, and A. J. Mayer, "Curbing junk e-mail via secure classification," in *FC '98: Proceedings of the Second International Conference on Financial Cryptography*. London, UK: Springer-Verlag, 1998, pp. 198–213.

[24] J. Goodman, D. Heckerman, and R. Rounthwaite, "Stopping spam," *Scientific American*, vol. 292, no. 4, pp. 42–49, Apr. 2005.

[25] G. Robinson, "A statistical approach to the spam problem," *Linux Journal*, vol. 2003, no. 107, pp. 58–64, 2003.

[26] T. A. Meyer and B. Whateley, "Spambayes: Effective open-source, bayesian-based, email classification system," in *Proceedings of CEAS*, Mountain View, CA, July 2004.

[27] T. Oda and T. White, "Developing an immunity to spam," in *Lecture Notes in Computer Science*. Springer-Verlag, 2003, vol. 2723, pp. 231–242.

[28] L. Weinstein, "Spam wars," *Communications of the ACM*, vol. 46, no. 8, p. 136, 2003.