Pawel Gburzynski* and Jacek Maitan†

Fighting the Spam Wars: A Remailer Approach with Restrictive Aliasing

To appear in ACM Transactions on Internet Technology

Abstract

We present an effective method of eliminating unsolicited electronic mail (the so-called *spam*) and discuss its publicly accessible prototype implementation. A subscriber to our system is able to obtain an unlimited number of aliases of his/her permanent (protected) E-Mail address to be handed out to parties willing to communicate with the subscriber. It is also possible to set up publishable aliases, which can be used by human correspondents to contact the subscriber, while being useless to harvesting robots and spammers. The validity of an alias can be easily restricted to a specific duration in time, a specific number of received messages, a specific population of senders, and/or in other ways. The system is fully compatible with the existing E-Mail infrastructure and can be immediately accessed via any standard E-Mail client software (MUA). It can be easily deployed at any institution or organization running its private E-Mail server (MTA) with a trivial modification to that server. Our system offers a simple method to salvage the existing population of E-Mail addresses while eliminating all spam aimed at them.

1 Introduction

One problem with the ubiquitous electronic mail is the ease with which senders of unsolicited junk mail, the so-called spammers, can exploit this medium. In contrast to traditional mass marketing, the cost of sending bulk E-Mail is virtually zero, regardless of the number of intended recipients [9]. Consequently, anybody can become a spammer, and even a token (or imaginary) return from this kind of marketing makes it attractive to many unscrupulous people. Ultimately, the cost of spamming is born by the vast majority of Internet users who are opposed to it, yet must put up with the ever-increasing volume of unsolicited E-Mail messages polluting their mailboxes.

Despite the efforts and beliefs of some crusaders, spam will not be eliminated (or even reduced) via legislative means. Even if declared illegal in the US [23], or any particular country, spam will continue to arrive from other places. Domestic spammers will promptly switch to foreign providers without a slightest disruption to their "business." Thus, the only viable ways of fighting spam must involve modifications to the mail handling systems, i.e., E-Mail servers (also called Mail Transport Agents or MTA'a) and (possibly) clients (also called Mail User Agents or MUA's). Owing to

*University of Alberta, Department of Computing Science, Edmonton, Alberta, CANADA T6G 2E8, pawelg@sfm.cs.ualberta.ca

†AppHome, Inc., 1301 Parkinson Ave., Palo Alto, CA 94301, jacekm@sfm.cs.ualberta.ca

the large infrastructure of already deployed MTA's and MUA'a, any immediate solution of this kind must be compatible with that infrastructure, i.e., it must assume the standard mail delivery protocol [15, 17].

Most solutions proposed along this line are based on the concept of mail filtering, whereby an incoming message is checked against a number of criteria which assess its legitimacy [1, 3, 5, 9, 13]. Filtering tools are popular because they are relatively simple and can be easily deployed by providers or individual subscribers without any change to the existing infrastructure of MTA's and MUA's. Another, considerably smaller and less popular class of methods aim at the elimination of address harvesting, i.e., the practice of collecting E-Mail addresses with the intention of using them for spamming [4, 9, 12]. Such solutions typically require the user to re-subscribe to a different E-Mail provider, e.g., to handle all unreliable contacts via a special channel. Some authors postulate drastic changes to the traditional concept of E-Mail service, e.g., introducing a fee for the right to deliver an unsolicited message [9, 10]. However, there seems to be no generally acknowledged, viable, reliable, and effective solution that would eliminate address harvesting without forcing the netizens to abandon or modify their existing E-Mail infrastructure.

Cognizant users often obfuscate their published addresses making them confusing to programs (i.e., harvesting robots), while preserving their legibility to humans. Notably, this conceptually simple trick has been recently put on a formal ground and turned into an interesting problem in artificial intelligence [6], which happens to be of some relevance to our proposed scheme (Section 4.6). Unfortunately, the most problematic of all situations are those when an address must be *de facto* published, i.e., revealed in a clear version, because it will be interpreted by a program. A typical example is entering an E-Mail address in a Web form supplied by an electronic merchant, i.e., for billing or order confirmation. With the proliferation of e-commerce and e-service, such scenarios are going to be more and more common. Needless to say, they will tend to become the primary source of address harvesting for unsolicited commercial mailing.

In this paper we outline a comprehensive solution to the problem of address harvesting and suggest ways of deploying it without revolutionizing, or even retiring, the present infrastructure. In a nutshell, we describe a *remailer* whose primary service consists in generating restrictive aliases pointing to the same permanent address of the subscriber. The permanent address need not be known by anybody except its owner, although it makes little difference if it has been heavily harvested and abused by spammers.

In the following discussion, we shall present our concept by describing the specific features available in a prototype Remailer accessible at `http://sfm.cs.ualberta.ca/remailer/`. If some of those features appear a bit arcane or unkempt, the reader should keep in mind that our present system does not purport to be a finished commercial product. Its role is to illustrate the possibilities that such a product may internally offer, possibly trimming them to the different levels of user expertise in a production deployment. From our point of view, it provides a fully usable and reliable platform for experiments, which let us identify problems, test solutions to those problems, try new features, and so on. As the

package is implemented in a highly flexible scripting environment, such experiments are easy to describe and monitor. The service is available without restrictions to account holders at the University of Alberta, and with some restrictions (aimed at limiting the volume of traffic) to the general public.

2 Principles of Operation

From the viewpoint of its role within the mail delivery system, the Remailer operates as an MTA extension, with the MTA itself conforming to the standard E-Mail delivery protocol, i.e., SMTP [15, 17]. The extension affects how the MTA interprets addresses of incoming and outgoing E-Mail and, formally, its operation can be described as aliasing, i.e., re-mapping of addresses. On its two ends, i.e., mail transport and client (user) interface, a Remailer-equipped MTA looks exactly as any other conformant MTA. This is illustrated in Figure 1, where the location of the Remailer extension is shown as the small shaded box. While the box does affect the information passed along the three outgoing arrows, it does not change the standard meaning and interpretation of this information within the global system. In particular, any standard user agent (MUA) can interface to the mailboxes handled by the Remailer using any popular mail delivery/retrieval technique, including direct storage in a local mailbox, POP/IMAP pull, forwarding to an address in a different domain, and, possibly, a superposition of any of those techniques with additional processing (like filtering or sorting). As we explain in the sequel, a subscriber to the Remailer system can have his/her real mailbox in any MTA domain, not necessarily equipped with its own Remailer instance.

The main function of the Remailer, as viewed by its subscriber, is easy (typically automatic) generation of limited-time, limited-accessibility, alternative E-Mail addresses. These addresses can be viewed as synonyms (aliases) for the permanent E-Mail address of the subscriber. They form the "username" component of the subscriber's E-Mail address within the domain serviced by the Remailer's MTA. The permanent address of the subscriber is never revealed by the Remailer. In principle, that address need not be known outside the Remailer by any party other than the subscriber, although this part is not critical from the viewpoint of spam elimination.[1] The subscriber can treat the permanent address as a user Id identifying him/her to the Remailer.

Aliases handled by the Remailer come in three flavors. Perhaps the most obvious (but not necessarily the most popular) way to create an alias is to use the Web interface to the Remailer. This way the subscriber can set up an arbitrary number of *regular aliases* with manually crafted configurations of attributes. Generally speaking, those attributes describe the acceptance criteria for an incoming message. A message addressed to the alias and meeting the acceptance criteria will be forwarded to the subscriber's permanent address. Otherwise, depending on the circumstances, the message may be dropped or bounced to the sender.

[1] In contrast to some other aliasing schemes, e.g. [12], the reliability of our Remailer does not rely on address secrecy.

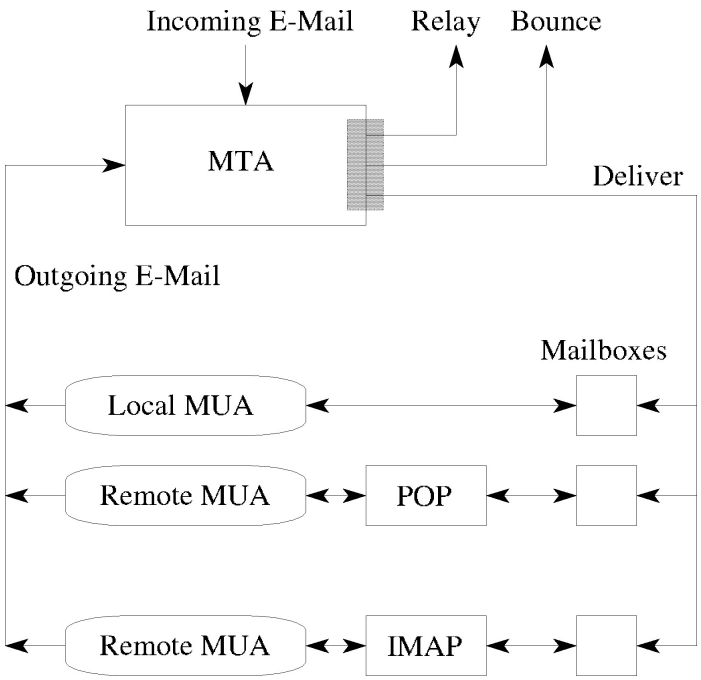

Figure 1: The Remailer's place in the mail handling system (the shaded box)

The attributes of a (regular) alias may indicate for how long the alias should be valid, how many messages can be received on the alias before it is invalidated, and who is allowed to use this alias for sending messages. They may also include a list of patterns applicable to the sender address, subject line, and message body. In particular, it makes perfect sense to create one alias per one sender. In a situation when the alias is exposed (e.g., by being inserted into a merchant's form on the Web), it can be made short-lived and thus useless for harvesting.

The need to create a regular alias arises seldom and usually involves a special occasion, e.g., a conference or other event requiring a point of contact with some idiosyncratic collection of acceptance criteria. In the vast majority of cases, new aliases are created automatically by the Remailer. Such aliases are dubbed *quick* and are functionally equivalent to regular aliases, the only difference being that their attributes are determined from predefined templates. Typically, a quick alias is "personalized" to the contact to whom it is presented as the subscriber's E-Mail address. This personalization means that (unless told otherwise) the Remailer will consistently use the same aliased identity of the subscriber in all correspondence with the same contact.[2]

The system offers a means for its subscriber to create and safely publish an E-Mail address to be used by any sender trying to reach the subscriber for the first time. An address of this type, called a *master alias*, can be published on a business card or on a Web page. A message arriving on a master alias is never forwarded to the subscriber but treated as a *query*, i.e., request for a quick alias of the subscriber personalized to the sender. In response to this request, the Remailer sets up a new quick alias restricted to the sender, and bounces the message. The returned message is preceded by a note explaining that the sender has to re-submit the message along with a response to a subscriber-defined challenge. Having received the correct response, the Remailer delivers the message to the subscriber and validates the new alias, which now becomes a legitimate address of the subscriber personalized to the new contact.

Like regular aliases, master aliases are created manually by the subscriber, but their population tends to remain small and stable. A master alias is not restrictive by itself: its role is to be as open as a traditional E-Mail address and accept E-Mail from any sender. The restrictive attributes of a master alias are used as a template for creating quick aliases in response to messages (queries) arriving on the master alias.

It makes sense to emphasize at this point that unlike other easily obtainable E-Mail addresses, e.g., ones assigned by Spamex [4], aliases created by the Remailer do not have to be viewed as second-rate disposable addresses intended for incidental contacts. This is because those aliases need not be terminated or abandoned when abused. Besides, they are set up in a way that for all practical purposes eliminates any possibility of abuse.

The Remailer absolves the subscriber from the burden of maintaining consistent identity with respect to a possibly large population of contacts, each knowing the

[2]Note that the word "contact" need not designate a single party but, possibly, a group of people (e.g., a set of addresses or a mailing list).

subscriber under a different alias. Notably, this is accomplished assuming no cooperation from the client software (MUA). When the subscriber replies to a message that has been received on an alias and forwarded to the subscriber's permanent address, the reply will automatically pass through the Remailer. The sender address of that reply will be changed to point to the proper alias of the subscriber before the message is forwarded to its recipient (or multiple recipients). If the subscriber initiates the E-Mail exchange, i.e., sends the first message, he/she can address that message to the Remailer itself, putting the recipient address (or addresses) in the subject field (along with the true intended subject of the message). In that case, the Remailer will locate the pertinent alias of the subscriber personalized to the receiving party and use that alias as the sender address of the forwarded message. This remailing mechanism makes it possible to specify a template for setting up a new quick alias, if no alias is already available. This template can be a master alias or a *quickcode*. The latter is a special object, similar to a master alias but not visible as an E-Mail address outside the Remailer, used exclusively for describing alias templates. This technique is especially handy for sending messages to casual and unreliable contacts, e.g., for e-commerce.

3 Organization of the Remailer

The Remailer has been programmed in Tcl [16] and installed under Linux. It operates as an add-on to a standard mail server (MTA). The present version of the package can be configured with sendmail [8] or qmail [21], as shown schematically in Figure 2. All E-Mail addresses (aliases) created by the Remailer for its subscribers necessarily fall into the domain handled by the MTA. For example, the MTA domain of our prototype installation is `sfm.cs.ualberta.ca`, and all addresses that can be assigned within that system are of the form *username*`@sfm.cs.ualberta.ca`. Thus, the Remailer allocates solely the *username* component, with the domain part being determined by the MTA.

Any MTA can be easily adapted to work with the Remailer with a minimal change to its standard functionality. The trivial modification[3] consists in directing to a special mailbox (set aside for the Remailer) all messages addressed to the MTA's domain that appear undeliverable because their *username* components cannot be resolved by the MTA. This way, the addition of the Remailer does not have to disturb the standard (traditional) E-Mail service. All that needs to be done to separate concerns is to guarantee that no message addressed to any of the aliases maintained by the Remailer, or to the Remailer itself, is ever deemed deliverable by the MTA. To this end, all the user names resolvable by the MTA have to be marked as "booked" in the Remailer's database, to make sure that the Remailer will never create its own aliases with those names. Additionally, a few reserved user names (e.g. `remailer`) must not occur as user names resolvable by the MTA within its domain.

[3]Notably, qmail does not even have to be modified, as the requisite behavior can be described in its configuration parameters.

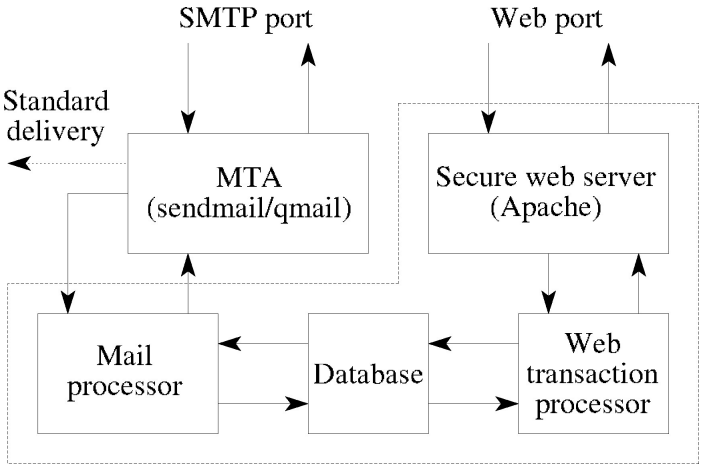

Figure 2: Remailer configuration

A message deposited by the MTA in the special mailbox, and classified as undeliverable by the MTA, may be addressed to one of the aliases maintained by the Remailer or, possibly, to the Remailer itself. If the Remailer finds out that the message has been in fact misdirected, it will bounce the message on the MTA's behalf.

The Remailer's database has been implemented using BerkeleyDB [20]. It stores records describing subscribers and their aliases, as well as some other information discussed in Section 5.2. The Web portion of the data flow in Figure 2 is straightforward: the dynamic Web forms provide a secure authenticated interface to the subscriber's record. Data passing through the mail processor is handled in several different manners, which will be described in detail in the following sections. Both the mail processor and the Web transaction processor are implemented in Tcl and share a significant amount of code.

4 Subscriber's Record

4.1 Signing up

To sign up (via the Web interface), a new subscriber has to specify his/her permanent E-Mail address, which will be used as the (globally unique) user Id, and a password. In response to this step, the Remailer sends to the provided permanent address a random secret code that must be entered by the subscriber once, at the first sign-on, to validate the new account. Non-validated accounts are unusable and they are discarded after 48 hours. This way nobody other than the actual owner of the permanent address can sign it up for service.

When signing up, the subscriber fills in some information constituting the so-called user profile. This profile can be edited and modified at any time, whenever

the subscriber logs on to the system. One element of the profile is the PIN code to be used as a simple means of authenticating remailing requests described in Section 5.1. Another (optional) item is the *filter cookie*. This is an arbitrary piece of text to be included with all messages sent or forwarded by the Remailer to the subscriber. If declared, the cookie string will be presented in a non-standard header (labeled `X-Filter-Cookie`), which will make it easily detectable by contextual filters guarding the subscriber's permanent address. Possible applications of the filter cookie are mentioned in Sections 6.2 and 6.3.

4.2 Regular aliases

Although regular aliases are not the most popular alias type (a subscriber can easily get away without setting up a single regular alias), it makes sense to discuss them first, as they well illustrate most of the functionality of our Remailer at a glance and provide a natural reference point for presenting all other features.

A regular alias is effectively equivalent to an E-Mail address within the MTA's domain. A message arriving at such an alias, and meeting the prescribed set of criteria, will be forwarded to the subscriber's permanent address.

A regular alias is created manually by the subscriber through a dynamic Web form. Its most important attribute is the name, i.e., the *username* component of the E-Mail address. The subscriber has three options: to let the Remailer generate the alias name automatically, to specify a prefix and let the Remailer complete the name, or to provide the complete exact alias name. Alias names generated by the Remailer are random, although, in contrast to [14], they are not impossible to memorize, which makes them only moderately burdensome for humans. For example, here are three sample random aliases generated by the Remailer: `qemtamek`, `wubcubol`, `cyssubib`.

With the last two options, the subscriber is able to enforce a specific form of the alias. In all cases, the Remailer makes sure that the name of the new alias is unique within the MTA domain. In particular, it will refuse to accept an exact name coinciding with the name of an already existing alias (or with a username booked by the MTA).

The subscriber can also specify his/her name to appear along with the aliased E-Mail address in the headers of outgoing messages. By default, this name is the same as the official name of the subscriber given at the time of sign-up.

The remaining attributes of an alias include the comment (which is an optional and functionally irrelevant piece of text, e.g., describing the alias's purpose), the expiration date, the message count, the acceptance flags, the fallback option, and the set of patterns. The *expiration date* indicates when the alias should expire. It can be specified as an explicit calendar date (in practically all sensible formats, e.g., `Oct 16`), as the number of days counting from the moment of creation, or as `infinite` (which is the default for a regular alias).

The *message count* puts a limit on the number of messages that can be received on the alias. Unless this attribute is `infinite` (which is the default), the specified number will be decremented by one after every message reception, and when it gets down to zero, the alias will become inactive and unusable. However, it will not be discarded until its expiration date, which makes it possible for the the subscriber

to re-activate the alias by resetting its message count. Also, explicitly setting the message count to zero is one way to disable an alias (temporarily) without discarding it.

The two *acceptance flags* indicate whether the subscriber is willing to accept on the alias executable attachments and/or HTML pages containing JavaScript. Some people are obsessed about the former, as they often carry viruses and worms. The reputation enjoyed by the latter is not much better: JavaScript in an E-Mail message is strongly indicative of abuse.

Normally, when a message arriving at an alias is rejected (because the alias has expired, run out of message count, or the message does not meet the alias's acceptance criteria described by the patterns), it is bounced to the sender along with an explanation. As an alternative, the alias can define the *fallback option*, which instructs the Remailer to treat such a message as if it arrived at the indicated master alias. Consequently, the sender will be challenged and given a chance to obtain a new alias of the subscriber. The primary purpose of the fallback option is to implement limited time periodically renewable aliases (see Section 5.3).

4.3 Pattern sets

The configuration of patterns is the most complicated attribute of an alias. Its role is to provide a powerful mechanism for advanced message classification, with the most obvious and most useful application being the restriction of the sender's domain. One can argue that with the Remailer's paradigm of restrictive aliases personalized to senders, there is no more use for contextual filtering of messages arriving on such aliases (except for sender verification). The pattern mechanism of the Remailer has undergone a drastic simplification from its previous (original) version,[4] and is likely to be simplified further in the next revision. Even an expert and obsessive user of our system has moderate need for manually created patterns.

The patterns of a regular (or quick) alias are grouped into three lists labeled *From*, *Subject*, and *Body*. The *From* patterns are applied to the sender address, the *Subject* patterns are matched against the subject line of the received message, and the *Body* patterns are sought within the message text. If a given pattern list is empty, the corresponding message component is not checked. Otherwise, the possibly multiple patterns in the list are viewed as an alternative of conditions. This means that to pass the test, the checked component must match *at least one* pattern from the list. Also, *at least one* of the checked components must pass the test before the message is considered acceptable. To put it succinctly into a single sentence, a message passes the pattern matching test, if any of its three components (the sender address, the subject line, the message body) matches at least one pattern in the corresponding list, or when all three lists are empty.

In all three types of patterns, the case of letters is ignored during matching. While the *Subject* and *Body* patterns have identical semantics, the *From* patterns are slightly different. A single *From* pattern is a string representing an E-Mail

[4]The previous version implemented four types of ranked patterns organized into six lists, including regular expressions and wildcards.

address or a class of E-Mail addresses. If the pattern looks like a full E-Mail address, e.g., `pawel@cs.ualberta.ca`, then it describes a single sender whose address should have almost the same appearance as the pattern. The word "almost" means that the Remailer admits some discrepancies. The first kind of discrepancy refers to to the domain components: they don't have to be identical, but they have to agree up to and including the first non-global domain counting from the right.

Example

The pattern `pawel@cs.ualberta.ca` will match the following addresses:

```
pawel@cs.ualberta.ca
pawel@ualberta.ca
pawel@phys.ualberta.ca
pawel@vpn.sheerness.cs.ualberta.ca
```

but not

```
pawel@shaw.ca
pawel@alberta.ca
piotr@cs.ualberta.ca
pawelg@ualberta.ca
```

This is because `ualberta` is the first "local" domain from the right. Consequently, `ualberta.ca` is deemed a sufficient signature of the sender's location.

The locality of a domain is assessed in a somewhat heuristic manner. The last domain is always considered global, and the second last domain is considered local unless it is one of the known global domains within a country domain, e.g., `org` in `imm.org.pl` is interpreted as global. Consequently, the pattern `henio@nri.org.pl` will not match the address `henio@imm.org.pl`.

The arguable advantage of adopting the fuzzy approach to domain matching is to account for the possible variability of the sender's host within the (local) domain of his/her organization. The second kind of discrepancy in address matching deals with the username component. Namely, if the username part of the sender address does not directly match the username part of the pattern, but it contains a dot, then it gets a second chance: it will still be considered a match, if the username component of the pattern matches the username segment of the address after the first dot.

Example

The pattern `gburzynski@ualberta.ca` will match the following addresses:

```
pawel.gburzynski@ualberta.ca
gburzynski@cs.ualberta.ca
mike.gburzynski@phys.ualberta.ca
```

The primary role of this feature is to account for the so-called *spices* in E-Mail addresses within a Remailer domain. As we shall see in Section 5.4, it is useful for making different Remailer subscribers (possibly in different domains) communicate with each other smoothly.

If the username component of a *From* pattern is empty, the pattern is matched exclusively to the domain of the sender address, i.e., it will accept any sender from the indicated domain. If the pattern starts with `@`, the domain part of the address must match the pattern string exactly; otherwise, it is enough if the pattern matches a trailing fragment of the address domain.

Example

The pattern `cs.ualberta.ca` matches the following addresses:

```
pawel@cs.ualberta.ca
mike@sheerness.cs.ualberta.ca
```

but not

```
pawel@phys.ualberta.ca
mike@ccs.ualberta.ca
```

A *Subject* or *Body* pattern is a list of space-separated words. Each word must be at least two characters long and may consists of any non-space characters. In order for the pattern to match the corresponding message component, all the words from the pattern must occur within the component in exactly the same order as in the pattern. They do not have to fall on any particular boundaries, e.g., they can be sub-words of longer words.

Examples

The following pattern:

```
quick jumps dog
```

will be matched by this fragment of text:

a **quick** brown fox **jumps** over the lazy **dog**

as well as by this one:

a dog **quick**er than the fox **jumps** over the **dog**house

but not by this:

```
a quick brown dog jumps over the lazy fox
```

Although all three pattern words occur in the text, the words `dog` and `jumps` appear in the wrong order.

For another illustration, consider the following list of *Body* patterns (which come from an actual correspondence with a car dealership):

```
Camry Corolla
Corolla Camry
Toyota
```

which will be matched by any text mentioning both `Camry` and `Corolla` (in any order) or containing at least one occurrence of `Toyota`.

For a slightly more complex scenario, consider the hypothetical case of an author willing to set up a special alias for correspondence regarding a paper submitted to a journal, e.g., *Transactions on Internet Technology*. In addition to restricting the longevity of the new alias (say, to one year and sixteen messages), the author may want to narrow down the context of the received messages, e.g., with the following set of patterns:

Type	**Pattern**
From	`editor@hisdomain.edu`
Subject	`toit`
Subject	`trans int tech`

According to them, to be deemed acceptable, a message has to arrive from the editor responsible for handling the paper (who is considered absolutely trustworthy as a sender) or, alternatively, it should mention the journal title (possibly in the abbreviated form) in its subject. The latter case may account for the situation where the paper has been passed to another editor who is now contacting the author as a previously unknown sender.

4.4 Quick aliases

The most significant difference between a quick alias and a regular alias is that the former is created automatically by the Remailer—always in response to a very specific demand. One consequence of this is that the subscriber has no influence on the name assigned to a quick alias. The restrictive attributes of a quick alias are set up based on a predefined template, which can be either a master alias or a quickcode.

Once created, a quick alias can be accessed by the subscriber (via the Web interface to the Remailer), viewed and possibly modified. Typically, however, the management of quick aliases is completely automatic and of no direct concern to the subscriber.

When a quick alias is created in response to a query arriving on a master alias, it is initially marked as "unvalidated." An alias in this state cannot receive (and forward to the subscriber) any messages, except for those that contain the correct reply to the challenge defined by the master alias. An unvalidated alias must be validated within 48 hours; otherwise it will be discarded regardless of its "official"

expiration attribute inherited from the master alias. Typically, an unvalidated quick alias becomes validated upon the first receipt of the correct response to the challenge. It can also be validated explicitly by the subscriber (see Section 5.6).

In addition to the standard set of restrictive attributes, which are the same as for a regular alias, a quick alias is equipped with two extra hidden attributes which cannot be modified. One of them, called the *sender association*, personalizes the quick alias to a sender or, possibly, to a group of senders. Note that in contrast to a regular alias (which is set up manually by the subscriber), a quick alias is created by the Remailer within the context of a specific E-Mail message. This can be an incoming message that has arrived on a master alias from a specific sender, or an outgoing message being sent by the subscriber to a specific destination. In the first case, the quick alias is personalized to the sender of the incoming message (the sender association is a single E-Mail address). In the second case, the alias is personalized to the recipient (or multiple recipients) of the outgoing message (and the sender association can be a group of E-Mail addresses).

The Remailer needs the sender association to consistently use the same existing quick alias when reaching the same contact. In particular, multiple queries arriving at a master alias from the same sender will not end up creating multiple quick aliases. Note that the sender association has no restrictive character and is independent of other attributes, e.g., patterns, possibly restricting the alias.

Another hidden attribute of a quick alias is the so-called *spice*, which is simply the name of the master alias that was used as the template for setting up the quick alias. For a quick alias created from a quickcode, the spice attribute is empty, unless the fallback option for the quickcode is effective (see Section 4.7), in which case the spice is set to the name of the fallback master alias. When the quick alias is used as the sender identity of the subscriber in outgoing E-Mail, the spice is appended to the alias name.

Example

Suppose that the Remailer's domain is `sfm.cs.ualberta.ca`, the name of the quick alias is `qemtamek` and that it has been created from the master alias named `gburzynski`. The sender address of the subscriber, as presented in the `From` header of an outgoing message, will be `qemtamek.gburzynski@sfm.cs.ualberta.ca`.

When the Remailer interprets the username component of a destination address within its E-Mail domain, it strips off the spice from the alias name. Thus, the spice does not affect the interpretation of the quick alias name as an E-Mail address, and can be viewed as a comment adding some descriptive content to the username component. This comment is interpreted by the Remailer in some circumstances, as discussed in Sections 5.3 and 5.4.

4.5 Master aliases

While regular and quick aliases are very similar in terms of their structure and semantics, master aliases are conceptually different. Although a master alias also represents an E-Mail address within the Remailer domain, it is never restrictive by

itself. Anybody and anything, including a spamming robot, can send a message to a master alias, and that message will be received by the Remailer. However, it will never be forwarded to the subscriber. While providing a globally accessible point of contact with the subscriber, the master alias acts as a check point validating the legitimacy of the sender.

Master aliases have no pattern sets. The remaining restrictive attributes of a master alias apply to the quick aliases created in response to *queries*, i.e., messages arriving on the master alias (as explained in Section 2). To this end, those attributes constitute an alias template.

The only way to create a master alias is to do it explicitly via the Web interface. The name of a master alias is assigned in the same manner as for a regular alias (Section 4.2), except that the *exact* option is now the default. Although there is no explicit limit on the number of master aliases owned by a single subscriber, this number is usually quite small and has no tendency to grow. In particular, it makes perfect sense to have a single master alias, which takes over the role of the traditional E-Mail address.

The following restrictive attributes of the master alias: the message count, the expiration date, and the acceptance flags, are directly inherited by all quick aliases created from the master alias. In contrast to the corresponding attribute of a regular alias, the expiration date can only be specified as the number of days from now, where "now" means the moment when the quick alias is created. Thus, it cannot be an explicit date, although it can be `infinite`. The fallback option is just a flag indicating whether the quick alias should have its fallback set at all, in which case it will be set to the master alias responsible for its creation.

One attribute specific to a master alias is the *sender option*, which can have one of three values: *One Sender*, *Sender's Domain*, and *All Senders*. It determines how a quick alias created from the master alias will assess the legitimacy of the sender of an incoming message. With *One Sender*, which is the default, the quick alias will only accept messages arriving from the single sender of the original query that arrived on the master alias. If the sender option is *Sender's Domain*, then any sender address from the E-Mail domain of the original sender of the query message will be deemed legitimate. Finally, *All Senders* disables sender verification altogether and makes the quick alias open to all senders.[5] The sender restriction is implemented by setting up an automatic *From* pattern (see Section 4.3).

The standard validation procedure for a quick alias created from a master alias is purely automatic: the quick alias becomes validated when it receives the first message containing the correct reply to the challenge. By setting the *user validation option* of the master alias, the subscriber can reserve the right to validate quick aliases explicitly.

4.6 The challenge

The challenge triggered by a master alias is simply a piece of text to be included by the sender in the subject line of a message sent to an unvalidated quick alias. This

[5]This last option is not recommended, unless the alias is going to be very short lived.

Figure 3: A sample magic phrase encoded as a CAPTCHA challenge (the original was in color)

piece of text, called the *magic phrase*, is an attribute of the master alias. When the original query message is bounced with the challenge, its `From` address is set to the quick alias. Thus, all the sender has to do is to copy the magic phrase into the subject line of the bounced query and hit the *Reply* button. The reply will be forwarded to the subscriber, and the alias will be automatically validated along the way.[6]

When presented to the sender as a challenge, the magic phrase is displayed as a randomly distorted JPEG image, to make it difficult to parse and decode by a program. The method used to transform the ASCII version of the magic phrase into the image has been borrowed from the CAPTCHA project [2]. Figure 3 shows a sample image with a magic phrase.

Notably, it is always safe to change the magic phrase without confusing any of the outstanding challenges. This is because the (current) magic phrase is internally copied to the quick alias when it is created. A message arriving at the unvalidated alias is thus matched against the magic phrase that was in effect at the exact time when the challenge was issued.

4.7 Quickcodes

A quickcode can be compared to a master alias, in that it represents a named template for creating quick aliases. However, in contrast to master aliases, quickcode names are not visible outside the subscriber's record.

The role of a quickcode is to provide a pure template, accessible only to the subscriber, for creating quick aliases needed for casual and short-lived contacts. A quick alias of this type can be created on the fly when the subscriber sends an outgoing message through the Remailer. Although in principle master aliases can be used for the same purpose, the templates provided by them are usually intended (and crafted) for more permanent contacts, while quickcodes provide means for setting up short-lived aliases primarily useful for e-commerce.

As a digression, we may note that a (quick) alias is spam-proof because of the following properties:

1. It is created for a specific contact and is not intentionally exposed by the

[6] Unless the user validation option is in effect for the alias (see Section 4.5).

subscriber outside the narrow circle of that contact. Thus, if the contact is reliable, the alias will never be harvested.

2. It is restricted to a single sender (or to a narrow set of senders). Consequently, even if a spammer tries to send a message to the alias, it will not get through, unless its headers specify the right sender address. This renders automated mass mailing impossible, even if the aliases can be harvested somehow.

3. It has a short life time. Thus, even if the alias is wide open to all senders, given to unreliable contacts, and eventually harvested, it will cease to exist before its abuse can commence.

Any single property from the above list is completely effective by itself. Typically, "serious" and reasonably reliable contacts will receive aliases with properties 1 and/or 2, e.g., obtained from master aliases or, occasionally, created by the subscriber manually (as regular aliases). With e-commerce contacts, it makes better sense to resort to property 3. This is because:

1. The subscriber would prefer not to bet on the reliability of the commercial contact, at least in most cases.

2. It may be difficult to restrict the population of senders of the new alias. The subscriber may be *bona fide* referred to another department or another manufacturer whose E-Mail address (or even domain) cannot be known in advance.

3. It makes sense to force the contact to be short-lived, because it is likely to be so anyway.

The list of attributes of a quickcode is similar to that of a master alias, except that there is no user validation option and no magic phrase. The default values of those attributes are set differently than for a master alias, to reflect the difference in typical applications of the two template types. Thus, a quick alias created from a default quickcode has no sender restriction, expires within one week, and is able to receive up to four messages.

The name of a quickcode is assigned explicitly by the subscriber: it can be arbitrarily short and it leaves no trace outside the subscriber's record. The fallback option of a quickcode, if selected, identifies one of the master aliases defined by the subscriber. Remarkably, the fallback option offers a safe way out, if the commercial contact turns out to be not-so-casual after all (see Section 5.3).

Two additional and specific attributes of a quickcode are *subject* and *body pattern flags*. Their role is described in Section 5.7.

5 Mail Processing

Every non-master alias provides the subscriber with an E-Mail address on which he/she can receive messages, as long as those messages meet the respective criteria, and as long as the alias does not expire or run out of message count. The subscriber can also use that address as his/her sender identity in outgoing messages.

5.1 Outgoing mail

Depending on the E-Mail client (MUA) used by the subscriber, it may be reasonably easy to maintain multiple sender identities corresponding to the population of aliases currently available to the subscriber. While doing this at the MUA's end might make some sense in the case of regular aliases (which are created explicitly by the subscriber), the postulate that the subscriber kept track of all quick aliases and used them consistently is hardly realistic, especially that those aliases are created in an automatic and essentially unpredictable manner.

The subscriber can easily avoid the burden of presenting a consistent sender identity to his/her multiple contacts by passing all outgoing E-Mail through the Remailer. Such a message should be addressed to `remailer` within the Remailer domain (e.g., `remailer@sfm.cs.ualberta.ca`) and its subject line should include a command identifying its actual recipient (or multiple recipients). The message must be sent from the official permanent address of the subscriber (identifying him/her as a legitimate user of the Remailer system).

In the simplest case, the subject command consists of two components: the PIN code (Section 4.1) followed by the recipient address. It must appear at the beginning of the subject line and be encapsulated in `+...+`, e.g.,

```
+a674 mary@someplace.net+
```

The rest of the subject line may include the actual subject of the message to be passed to the recipient. The command string will be removed from the subject line before the message is forwarded.

Note that the PIN code is essential. Without it, anybody knowing the subscriber's permanent address could play the same trick, which could easily lead to abuse. Having received a command as in the above example, the Remailer performs the following actions:

1. If the indicated recipient already has a non-expired personalized quick alias, that alias will be used as the sender identity, and the message will be forwarded to the recipient.

2. If the recipient already has a personalized quick alias, but that alias has expired or run out of message count, the restrictive parameters of that alias will be reset based on the template from which the alias was originally created. Then the message will be forwarded to the recipient using the existing alias as the sender address.

3. If the recipient has no alias, a new alias will be created and personalized to the recipient using the first (default) master alias of the subscriber as the template. Then the message will be forwarded to the recipient with the new alias used as the sender address.

The subject command may specify multiple recipient addresses. In that case, the Remailer will find (or create) a quick alias personalized to the entire group of

recipients. The message will be forwarded to all recipients, with the first address put into the `To` header and the remaining addresses used as `Cc` targets.

The subject command of a remailed message may include an alias/quickcode name following the PIN and preceding the (first) recipient address, e.g.,

```
+a674 aname mary@someplace.net john@somewhere.else.org+
```

If the name identifies a quickcode or a master alias,[7] the specified quickcode/alias is used as the template to create the new quick alias, but only if no quick alias personalized to the recipient(s) already exists. But if the name points to a regular or quick alias, that alias is forcibly used as the sender address of the forwarded message, ignoring its personalization and possible restrictions.

5.2 Replying to incoming E-Mail

Whenever an E-Mail message arrives at a non-master alias and is deemed acceptable, the Remailer reorganizes its headers before forwarding it to the subscriber's permanent address. When it reaches the subscriber, the message appears as if it were sent by the Remailer itself, from an address that consists of the name of the receiving alias concatenated with the so-called *message tag*. The address of its true sender, as well as the addresses of its possible other recipients, are removed from the standard headers (`From`, `Reply-To`, `Cc`), and presented in two nonstandard headers: `X-Originally-From` and `X-Originally-Cc`. Also, to make it more conspicuous, the original sender address is included as a comment along with the substituted sender address. The purpose of those tweaks is to make sure that when the subscriber replies to the message, the response will automatically pass through the Remailer instead of being sent directly to any of the addresses mentioned in the original headers. This way, the subscriber will not accidentally reveal his/her permanent E-Mail address (typically exposed by the MUA), and all the other parties involved in the correspondence will have a consistent perception of the subscriber's aliased identity.

Example

Suppose that a message with the following headers is received by the Remailer:

```
...
From: "Kris Kelvin" <k_kelvin@excite.com>
Subject: I am a sample message
To: "Pawel Gburzynski" <qemtamek.gburzynski@sfm.cs.ualberta.ca>
Cc: "John Smith" <jsmith@somewhere.com>, "Susan Doe" <susan@cs.ualberta.ca>
...
```

where `qemtamek` is a quick alias and `gburzynski` is its spice.[8] When the message is delivered to the alias's owner (`pawel@cs.ualberta.ca`), its headers will be changed to something like this:

[7] If the subscriber has a quickcode with the same name as an alias, the quickcode takes precedence.

[8] As explained in Section 4.4, it identifies the master alias from which the (quick) alias `qemtamek` was created.

```
...
From: "k_kelvin@excite.com" <qemtamek_horrysyszixvuqgs6@sfm.cs.ualberta.ca>
Subject: I am a sample message
To: <pawel@cs.ualberta.ca>
Cc: "jsmith@somewhere.com, susan@cs.ualberta.ca"
    <qemtamek__horrysyszixvuqgs6@sfm.cs.ualberta.ca>
...
X-Originally-From: <k_kelvin@excite.com>
X-Originally-Cc: <jsmith@somewhere.com>, <susan@cs.ualberta.ca>
...
```

Note that when the recipient replies to that message, the response will be addressed to `qemtamek_horrysyszixvuqgs6@sfm.cs.ualberta.ca` and, if the reply is to "all," carbon-copied to `qemtamek__horrysyszixvuqgs6@sfm.cs.ualberta.ca`, with both copies being intercepted by the Remailer. Based on the alias identifier (`qemtamek`) and the message tag (`horrysyszixvuqgs6`), the Remailer will address the response to the proper recipients and also remove the permanent address of the sender replacing it with the alias.

To implement this transparent processing, the Remailer saves (for a limited time) some extracts from the headers of all messages that have arrived at aliases and have been forwarded to their owners. In the present version of the system, those extracts are stored for 180 days, which is the limit on the amount of time within which the subscriber can consistently reply to a message via the Remailer. After that time, such a reply is going to bounce (with a pertinent explanation of the reason). Note, however, that all information regarding the actual sender and other recipients is still available in the headers of the originally received message, so it can always be extracted and processed manually.

As a side effect of the feature discussed above, the subscriber can easily tell which particular alias was responsible for delivering every message reaching him/her through the Remailer.

5.3 Fallback

The Remailer defines two levels of fallback processing for a message that has arrived at a regular or quick alias and is being rejected. The first level is triggered if the following three conditions hold simultaneously:

1. The alias selects the fallback option, i.e., it specifies a fallback master alias.
2. The reason for rejection is that the alias has expired or run out of message count.
3. The message would be otherwise acceptable.

In such a case, the rejected message is treated as a query that has arrived at the fallback master alias. This type of fallback processing can be viewed as a renewal of the old contact. If the sender correctly replies to the new challenge, the old alias will be re-validated and made usable again.

An alias that specifies a fallback master alias is never deleted, even if it has expired, for as long as its fallback option remains set and the master alias is not removed by the subscriber. This way, the expired alias remains available for renewal to its previous legitimate contact.

If the first level fallback is not applicable, it means that the quick alias to which the message is addressed either does not exist or is not usable by the sender. Consequently, that alias cannot be recycled for the present contact. But if the destination address is spiced, and the spice identifies a valid master alias owned by the subscriber, the rejected message is still treated as a query that has arrived at the master alias.[9]

The primary difference between the two fallback levels is that the second level can be applied in a situation when the alias to which the message is addressed no longer exists. The presence of spice in the destination address, indicates that the sender knows the master alias of the subscriber. Thus, as a shortcut, the Remailer immediately gives the sender a chance to respond to the challenge associated with the master alias and thus establish a contact with the subscriber.

5.4 Remailer-Remailer interoperability

The automated processing, especially the challenge-response protocol used by the Remailer, may raise concerns regarding the interoperation of multiple Remailer systems. Anyone who has witnessed two sophisticated answering/callback machines accidentally involved in a "conversation" will quickly agree that there are few more convincing illustrations of the futility of easy automation.

One can easily see that two Remailer systems talking to each other cannot be caught in a loop, as long as the message exchange, including bounces, follows the rules of SMTP. This is because the Remailer itself never originates any E-Mail message and, in particular, all queries addressed to master aliases must be initiated by human subscribers (unless they originate outside the Remailer system).

The worst-case automated Remailer-Remailer exchange will occur when a subscriber uses a master alias as the return address in a message addressed to another master alias.[10] The destination Remailer will respond with a challenge (arriving from an unvalidated quick alias) which the source Remailer will interpret as a query. The challenge sent by the source Remailer will not be recognized as a valid response at the destination and it will bounce. The source Remailer will attempt to bounce the received reply one more time, and this is where the exchange will end. This is because the second bounce will have no envelope sender [15] and it will be dropped by the source Remailer's MTA.

A more relevant problem, that actually requires some attention, involves two subscribers setting up their first contact through aliases and remailing requests. Consider two subscribers named A and B located within the same Remailer domain or in two different domains. Suppose that each of them has set up a master alias, say, `mastera` and `masterb`, and made it his/her published E-Mail address. Now, suppose that subscriber A wants to send a message to subscriber B, but he/she does

[9] The preamble explaining the challenge to the sender differs slightly in the two scenarios.

[10] Note that this requires a malicious intention on the subscriber's part.

not yet have a personalized (quick) alias of B. Thus, A sends the message through the Remailer specifying `masterb` as the destination address and using `mastera` as a template to create a quick alias of A personalized to B. Let the name of that quick alias be `qumfovud`. When the message sent by A reaches the destination, the Remailer will create a quick alias of B (say `gabdubid`) personalized to A and bounce the message to A along with the challenge. When the challenge arrives at `qumfovud`, its sender is `gabdubid`. At first sight, this is unfortunate because `qumfovud` has been personalized to the master alias of B, i.e., `masterb`. In particular, with the default setting of the sender option of `mastera`, `qumfovud` is guarded by a *From* pattern that identifies `masterb` as the only legitimate sender. Thus, it appears that the challenge will be rejected and A will never see it!

This is where alias spicing comes to the rescue. To account for such scenarios, the Remailer spices a quick alias created from a master alias with the name of the master alias. In the discussed case, the challenge message arriving from B will have `gabdubid.masterb` as the username component of its sender address. According to the way address patterns are matched (see Section 4.3), this component will match `masterb`, as required to make the challenge message acceptable by the quick alias at A.

5.5 Abuse prevention

The Remailer has been devised to eliminate spam, which is considered these days to be the most serious type of abuse suffered by the public E-Mail system. Owing to the fact that E-Mail headers in SMTP are little more than decorations with no authoritative content, any E-Mail system is potentially open for other types of abuse, including identity theft and misuse of service, which are closely related to spamming.

Fighting identity theft within the framework of SMTP is hopeless because the contents of the `From` headers are never authenticated, and the sender can put absolutely anything into that field with no impact on the E-Mail delivery protocol. Within the Remailer system, the sender address of a remailed (outgoing) message is used to identify the subscriber. As it can be trivially faked, other measures must be taken to make sure that only the actual subscriber can receive the service.

Those measures come as the PIN code that must appear within the subject line of a remailed message. While the solution is not perfect—because the PIN code is potentially exposed on its way to the Remailer—it is the only type of solution that can be adopted within the existing framework of MUA's and MTA'a. The PIN code can be changed (e.g., periodically) to reduce the likelihood of its harmful exposure.

Let us note, however, that the remailing protocol, as introduced so far, leaves room for another type of abuse. If the subject line command contains an error, the Remailer should notify the subscriber about the mishap and give him/her a chance to correct the problem and resubmit the message. Imagine an intruder sending a remailing request on behalf of a legitimate subscriber and specifying an invalid PIN code. Such a request would bounce to the subscriber with an error message (invalid PIN). This way the intruder might be able to deliver spam to the subscriber's mailbox.

To eliminate this trapdoor, the Remailer will not bounce to the subscriber a remailed message whose PIN code in the subject line is incorrect. As the subscriber may be concerned that some fatal problems are potentially left undiagnosed, the Remailer confirms every successfully remailed message by sending a note back to the subscriber. This default behavior can be switched off with a flag in the subscriber's profile record. Note that regardless of the setting of this flag, the subscriber will be explicitly informed about any problems that have occurred after the PIN code was successfully verified.

One more possibility of abuse stems from the way the Remailer handles replies to messages that arrived on aliases and have been forwarded to the subscriber (Section 5.2). Owing to the fact that the header extract records needed to consistently remail those replies are stored for a limited time, it is conceivable that a reply will bounce to the subscriber—because the header extract identified by the message tag has expired and is no longer available. Again, an intruder could exploit this feature by faking a subscriber's reply using a known alias of the subscriber concatenated with a random message tag.

To make sure that such a message is never bounced to the subscriber, the Remailer signs message tags with a secret key stored within the subscriber's record. This key is generated at random when the subscriber signs in to the system. A message tag whose signature is invalid does not trigger a bounce: it is deemed fake and dropped.

5.6 Validating quick aliases

A quick alias acquired by a new contact via a master alias query (Section 4.4) must be validated to become permanent and fully usable. By default, the validation is automatic and occurs when the alias receives the first message whose subject line contains the correct magic phrase. The subscriber may choose to switch off the automatic validation by selecting the *user validation option* of the master alias.

Regardless of the setting of that option, the subscriber can always validate a quick alias manually, either by using the Web interface to the Remailer (and explicitly modifying the alias's record), or by sending a message through the alias (Section 5.1) and inserting the string `+V` or `V+` into its subject line. The Remailer will remove this string before it forwards the message to the other party. The latter operation is natural in combination with the user validation option. In such a case, the (validating) message sent by the subscriber through the unvalidated alias is a reply to the first message received from the new contact.

5.7 Transient patterns

A remailing request that specifies a quickcode (Section 5.1) offers one option that is not available for an (implied or explicit) alias. In addition to (or instead of) the sender restriction (described in the quickcode), the subscriber may specify subject and/or body restrictions referring to the expected contents of the subject field or the message text expected to arrive from the new contact. This may be useful in a truly desperate situation when:

- it is difficult to sensibly restrict the sender
- the quick alias cannot be short-lived (its longevity is likely to make it susceptible for abuse)
- a legitimate message arriving at the alias is expected to mention a distinct keyword or phrase in the subject or message body

In such a case, the subscriber may use a quickcode that has no sender restriction but instead sets the *subject* and/or *body pattern flag*. This makes it possible to specify simple patterns in the subject line and/or text of the remailed message, depending on which flag(s) have been selected in the quickcode.

Example

The simplest case of a subject/body pattern involves a single characteristic keyword. For example, the subject line of a remailed message may look like this:

```
+a674 qc sales@robmeblind.com+Need a quote on +RSX8922
```

where `RSX8922` identifies a piece of equipment that the subscriber would like to purchase. By preceding a single word in the subject line with `+` (which is removed before the message is forwarded to its recipient), the subscriber turns this word into a *Subject* pattern that will be associated with the new quick alias. To be considered acceptable, a message arriving at the alias will have to include the selected word in its subject.

The general syntax of a subject restriction involves two special characters, `+` and `&`, which can precede words, with a word being defined as any sequence of characters from this set: <`a-z`, `A-Z`, `0-9`, `_`>. A word preceded with `&` is appended to the previous pattern (as a separate word), while a word preceded with `+` starts a new pattern.

Example

The following sequence:

```
+Toyota &Corolla for +Johann Sebastian &Bach (&Baroque)
```

defines two patterns: `toyota corolla` and `johann bach baroque`.

Recall from Section 4.3 that if the subject line includes multiple patterns (like in the above example), it is sufficient if at least one of them matches the subject line of a received message to deem it acceptable.

A similar technique is used to mark patterns within the message body, except that the sequences `++` and `&&` are used instead of `+` and `&`. This is because `+` and `&` are not unlikely to occur in the message text as regular characters, which could easily lead to a confusion. Note that for the subject/body patterns to be available at all, the corresponding options of the quickcode must be selected. Thus, this exotic feature is not available by default and cannot be accidentally misused by an unexperienced subscriber.

5.8 Caveats

The Remailer makes it is easy to maintain multiple quickcodes with different types of restrictions, e.g., intended for different types of (casual) contacts. A more obsessive subscriber with a taste for intricate solutions may be inclined to take advantage of this possibility to create an elaborate structure of aliases guarded by subject and/or body patterns. We would like to point out that these tools are not intended for routine usage and they may affect the reliability of communication, i.e., block some messages that should be accepted.

Note that while there is no danger in limiting the time duration of a commercial contact, there is always a risk of missing some important message, if the alias on which the message is expected to arrive is guarded by subject/body patterns. This risk is amplified if the sender happens to be a program, as is typical in many e-commerce scenarios. Thus, a short-lived wide-open alias is the recommended solution in such circumstances. By furnishing this alias with a fallback option (Section 5.3), the subscriber makes it available for long-term human contacts, while limiting the duration of its exposure to automatic mailers. Even if the quick alias itself is subsequently deleted by the subscriber and forgotten by the Remailer, the presence of spice (Section 4.4) makes the human contact possible for as long as the fallback master alias is still there.

Another controversial class of communication scenarios, that may drive the subscriber towards using subject/body patterns, is mailing lists. The problem here is that unmoderated mailing lists tend to be spammed. Consequently, an alias set up to accept E-Mail arriving from a mailing list (viewed as a single sender by the alias) will let through all the spam sent to the list. One should realize, however, that formally a subscription to a mailing list implies a willingness to accept all traffic sent by the list owner. The problem is thus not with the Remailer but rather with the list, i.e., its not being secured against spam.

6 Salvaging the Existing Infrastructure

The primary concern of an institution or organization contemplating a transition to the new E-Mail paradigm represented by the Remailer will be the fate of the existing infrastructure of E-Mail addresses that have been heavily harvested and put onto numerous lists sold to unscrupulous spammers. At first sight, there seems to be no alternative to scrapping them and starting the game from scratch. Fortunately, the open-ended character of the Remailer offers solutions to this problem.

6.1 Same E-Mail domain

If the Remailer is installed within the E-Mail domain of an existing address infrastructure, the old addresses can be re-declared as master aliases. This will let them retain their official and traditional publishable status, while freeing them completely of unsolicited E-Mail, no matter how heavily they have been abused in the past. Owing to the fact that the Remailer's MTA need not give up its traditional duties, this solution can be adopted gradually, as the users become convinced that they really

want to switch to the new type of service. Those of them who, for one reason or another, will be reluctant to subscribe to the Remailer will be able to continue using their old addresses exactly as before.

6.2 Different E-Mail domains

Even if the Remailer is installed in a different E-Mail domain, the old (possibly harvested) addresses can still be used as permanent addresses for the Remailer. To de-spam them, the users can deploy aggressive filters, e.g., blocking all incoming messages except for the ones arriving from the Remailer (which are easy to identify through *filter cookies*—see Section 4.1). In a slightly more refined setup, the filters may additionally let through all local E-Mail, i.e., originating and entirely passed within the domain of the local organization, which property is usually easily verifiable via a superficial examination of message headers [1]. This is how one of us (PG) handles all E-Mail, personal and business alike, enjoying life without spam.

6.3 Refiltering through the Remailer

Some MUA's, notably Microsoft Outlook™ can take advantage of the Remailer's *refiltering* feature to completely de-spam an old E-Mail address while retaining its traditional and official role. This solution works with the Remailer installed in any domain, not necessarily within the domain of the old address. The necessary prerequisite on the MUA's side is the ability to filter messages based on keywords detected in the headers and, conditionally, forward them to a specified E-Mail address in a way that preserves the essential information from the original headers. In a nutshell, the procedure is carried out as follows:

1. The MUA receives an E-Mail message. If the headers of that message include the filter cookie identifying the message as arriving from the Remailer, the message is delivered to the subscriber's mailbox.

2. Otherwise, the message is sent to the Remailer for *refiltering*. The Remailer treats the message as if it arrived on a master alias indicated by the subscriber.

The refiltering address to which the message is forwarded in step 2 must have the following format:

masteralias`_f_`*PIN*`@`*remailer.domain*

where *masteralias* is the name of the master alias to be equivalenced with the salvaged address, and *PIN* is the subscriber's PIN code. The master alias implicitly identifies the subscriber, while the PIN is used for authentication. For example, a specific refiltering address may be `gburzynski_f_a674@sfm.cs.ualberta.ca`.

With refiltering, the old E-Mail address is functionally turned into a master alias within the Remailer's domain. Any E-Mail message arriving at the old address, that has not been forwarded by the Remailer, is treated *exactly* as a query received on the selected master alias. Thus, parties trying to reach the subscriber on the old address will be assigned personalized aliases for reliable spam-free communication.

7 Demands on User Expertise

When the Remailer is introduced with all its options and features, as in the present paper, it may appear intimidating and discouraging to a non-expert user. Terms like pattern sets, transient patterns, fallback option, sender associations, are not indicative of simplicity or user-friendliness. However, the purpose of the present paper is not to introduce the Remailer to a non-expert user but to explain its functionality and the underlying mechanisms to an audience of experts. Moreover, we admit that some of the more arcane features of the Remailer are of disputable advantage even to expert users. They exist because the present version of the Remailer is still a prototype and, at the current stage of its development, it is safer to leave an unneeded feature (which may be removed or trimmed down in the next version) than to miss some functionality that may turn out to be important and useful after a more thorough study.

7.1 Quick start

In fact, the Remailer turns out to be remarkably simple and immediately accessible to a non-expert user for this reason: the most useful and relevant parts of its functionality are available by default. Consequently, a casual user need not understand the meaning of any non-obvious features, or even realize their existence. For a quick start, a new subscriber has to:

1. Declare a PIN code.
2. Create a single master alias. This operation boils down to inventing a name for the alias and a magic phrase. There is no need to touch any of the default attributes of the master alias.

Following this rudimentary setup, the subscriber is immediately able to receive spam-less E-Mail and respond to it through the Remailer without any extra tricks or knowledge. To be able to initiate E-Mail exchange through the Remailer, the subscriber has to understand the concept of the subject line command (Section 5.1) in its most basic variant. Over 95% of all E-Mail sent/received by the authors requires no more familiarity with the system.

The next step, which will cover 4 of the remaining 5%, is the creation of a single quickcode to be used for intermittent casual contacts. Again, with the default setting of quickcode attributes (crafted exactly for this type of usage), the effort on the subscriber's part reduces to selecting a name for the quickcode. Additionally, the subscriber has to extend his/her understanding of the subject line command, to be able to insert the quickcode name between the PIN code and the recipient address.

In the next version of the Remailer, the quick start procedure can be simplified and provided as a one-click operation, which will create a single master together with a single quickcode. While the subscriber will still have to assign (or at least agree upon) the name of the master alias, the quickcode name can be generated automatically by the Remailer.

7.2 Possible assistance from the MUA

With the Remailer keeping track of the multiple identities of its subscribers, the only operation that incurs some extra effort at the MUA is remailing (Section 5.1). For this operation, the subscriber must (manually) create a subject line command consisting of the PIN code, the optional alias/quickcode identifier, and one or more recipient addresses. A trivial extension of the GUI provided by the MUA could assist the subscriber with this operation rendering it as simple and natural as hitting the *Send* button. This extension could consist of a few extra *Send* buttons (two buttons would satisfy most demands) that the subscriber could bind to aliases and quickcodes. By hitting one of those special buttons, the subscriber would be able to remail the message without having to manually enter the subject line command, which would be set automatically by the MUA.

Example

A sample collection of *Send* buttons might look as follows:

Send	This is the standard *Send* button. When this button is pressed, the message is expedited in the traditional way.
*Send **	When this button is pressed, the message is remailed using the default (first) master alias (the subject line command specifies no alias).
Send ec	With this button, the message is remailed using the quickcode named `ec`.

In response to a non-standard *Send* button, the MUA would create the pertinent subject line command, prepend it to the message subject specified by the subscriber, and address the message to the Remailer.

8 Related Work

The idea of channels, proposed by Hall [12], is somewhat reminiscent of our solution, in that it introduces multiple versions of the recipient address (channels) personalized to different senders or generated for different occasions. However, in contrast to our approach, the security of a channel (viewed as its resistance to abuse) is primarily in its secrecy. Additionally, to manage the channels, the user needs a Personalized Channel Agent collaborating with the MUA, whose list of responsibilities includes the consistent presentation of the user's identity (e.g., the "Cc problem" mentioned by Hall). Consequently, it is difficult to deploy that solution without seriously modifying the existing infrastructure. While the motto of the channels approach (see [12]) is "If people don't know your address, they cannot send you E-Mail," the idea of our solution can be succinctly stressed as "Programs cannot send you E-Mail, even if they know your address."

A publicly available commercial service similar to our Remailer is available as Spamex [4]. Spamex subscribers are able to acquire disposable addresses that can

be easily revoked (terminated) when abused. However, there is no automatic way to restrict them in time or to confine them to a narrow population of senders. Besides, Spamex offers no tool similar to our master aliases (Section 4.5) that could be used as safe publishable points of contacts for its subscribers. Although Spamex service is useful in many circumstances (for incidental short-lived contacts with untrusted parties), it cannot be reliably used for more serious and long-lived communication, as the only remedy to abuse detected by the subscriber is alias revocation.

Spamex implements transparent replies through aliases using a method somewhat similar to the one described in Section 5.2, but it does not handle group replies, i.e., those addressed to the 'Cc' recipients. A group reply through Spamex is going to reveal to those recipients the permanent address of the subscriber, while hiding the alias. This is because Spamex only archives information about the senders of the re-mailed messages, its official purpose being the possibility to track down abuse.

A solution aimed at associating attributes (policies) with E-Mail aliases has been recently proposed in [14]. With that approach, the attributes are encoded in the alias itself, which is then encrypted to make the policy tamper-resistant. The primary advantage of this scheme is the state-less and memory-less processing at the MTA, which requires no database to keep track of the aliases and their attributes. However, there is no way to use a system based on state-less aliasing without a special (aware) E-Mail client (MUA) that either generates the aliases by itself or knows how to reply to a message arriving on an alias. This makes it impossible to switch to the new system without replacing critical elements of the existing infrastructure. A state-less system is also unable to provide the type of service offered by master aliases in our Remailer. Finally, the state-less approach limits the population and types of alias attributes and makes it impossible to modify them once the alias has been created and handed out. In particular, dynamic attributes, (like the message count in our system), cannot be accommodated by this method.

In our opinion, the most recent attempts to employ sophisticated filtering techniques for identifying and eliminating spam [13, 18] are bound to fail. We claim that spam filtering is fundamentally flawed as a concept because spamming is in the intention and *modus operandi* of the sender rather than in the message content. Consider the following message:

```
Dear Son:

We have arrived home safe and sound. You may want
to know that your Junktronix camera worked great,
and we are ready to share with you the magnificent
pictures that we took on our way.

See you soon,

Mom
```

Even a human viewer cannot say whether the message is legitimate or not without knowing the circumstances of its creation and transmission. And if a sophisticated

filter can somehow spot the brand name of a product in this completely innocuous message and, in its desperation, use it as a hint, the spammer can always resort to sending the entire message (or merely its sensitive fragment) in a graphic attachment. Notably, graphic attachments are considerably easier to disturb randomly (personalize) without affecting the message content. This simple technique defeats filters driven by databases of previously sighted spam, e.g., recognized as spam by humans [9].

In the light of the above observations, the comparative studies of spam filters trained on and benchmarked against various samples of unsolicited E-Mail collected over the past [7, 11, 19] cannot be viewed as more than an abstract academic exercise in computational linguistics. This is because the form and content of spam have an essentially unlimited flexibility to evolve in response to advances in filtering techniques. We have to conclude that the only reason why spam filtering has been effective at all (at the scale at which it has been deployed), is that spammers have yet to perceive it as a serious disturbance to their practice.

9 Final Remarks

The amount of unwanted junk E-Mail in the Internet has been increasing steadily since the network went global, and now seems to be approaching a critical point after which a drastic solution will have to be adopted. Our proposed paradigm offers one possible remedy to this problem.

One of our most serious concerns was the feasibility of smooth deployment of the new service and its seamless coexistence with the legacy infrastructure. Our experience with the prototype implementation of the Remailer strongly suggests that despite its shortcomings (typical of a prototype), the system is viable, reliable, effective, and safe. We would like to mention that the present version of the system is a significant revision of its first implementation, and it brings about several simplifications of the user interface which have been arrived at after consultation with its users. While some assistance from a Remailer-aware MUA could still be useful in simplifying the subscriber's interaction with the system, the present service is quite acceptable and practically ready for public deployment.

We do not share the worries expressed in [14] that the database of a state-based aliasing system will have a tendency to "grow without limits." Even if the database of the Remailer has some natural tendency to grow as new users subscribe to the service, that tendency is considerably less pronounced than in a typical database of a public E-Mail service, which tends to be heavily polluted with throw-away mailboxes. Although the Remailer can easily generate large numbers of aliases, practically all those aliases are short-lived and disappear after a while without a trace. On the other hand, an alias explicitly set up as long-lived is likely to be treated as a serious E-Mail address not to be discarded, abandoned, or forgotten. This is especially true if the Remailer service is deployed in a serious environment, e.g., at a scale of one institution, rather than being offered as a cheap and "spamvertised" source of disposable aliases for everybody.

Our rough and pessimistic estimate indicates that the stable amount of database

space per subscriber, under a heavy usage pattern, is much less than 0.5MB. With the contemporary pricing of disk storage, this amount is hardly prohibitive. Its bulk (more than 90%) is used to accommodate the archived header extracts needed to implement transparent replies from unaware MUA's (Section 5.2). As the system will evolve towards more collaborative MUA software, the need for this space will be reduced and eventually eliminated. By that time, of course, disk storage will become even cheaper than it is today.

The general idea of a challenge-response protocol for establishing the first contact with an E-Mail recipient is sometimes criticized as cumbersome, unfriendly, or impolite. Interestingly, one of us (PG) has had a chance to observe how the attitudes of people towards filter challenges evolve with time, or rather with the amount of spam that those people are forced to dig through every day. A few years ago, a message bounced with a challenge would occasionally meet with an objection from a mildly upset sender.[11] These days, instead of objections, we are receiving words of appreciation and requests for our spam prevention tools. To put it in the right perspective, there is nothing wrong about a politely worded challenge after which the correspondence becomes noiseless, spam-less, and smooth.

One should also note that the amount of extra traffic caused by the challenge-response protocol is going to be a completely negligible fraction of the total E-Mail traffic, even after all spam has been accounted for and eliminated. The protocol involves at most one challenge-response per each new alias, and a new alias acquired this way (in contrast to a quick alias created by the subscriber) is likely to be used for sending a nontrivial number of messages, possibly including lengthy attachments. Although no meaningful statistics are available, we can safely speculate that the average percentage contribution of the challenge-response exchange to the total traffic passing through the alias is likely to be below 1%.

The proliferation of spam on the Internet has brought us a challenge, which we originally interpreted as a need to devise better filters in response to new spamming tricks and gimmicks. This is difficult and unfair. Ultimately, to carry out its duties without a mistake, a spam filter must be able to understand not only the message, but (as we argued in the previous section) also the intentions of its sender. Such a filter will be able to pass the Turing test [22] with flying colors. Consequently, it makes much better sense to reverse the challenge. Even if the spammers finally manage to create an automated responder to CAPTCHA challenges [6], i.e., one capable of passing the Turing test, that program will be necessarily intelligent enough to find itself a more creative, productive, and gratifying activity.

Acknowledgments

The authors wish to thank Carey Williamson for a number of constructive suggestions that have greatly improved the quality of this paper, as well as the anonymous reviewer number 2 whose criticism and skepticism have stimulated many recent enhancements to the system.

[11] The single anecdotal really harsh comment was triggered when a letter inviting a job applicant for an interview bounced to his prospective employer.

References

[1] `http://sheerness.cs.ualberta.ca/RabidFire/`.

[2] `http://www.captcha.net/`.

[3] `http://www.deersoft.com/`.

[4] `http://www.spamex.com/`.

[5] `http://www.sunbelt-software.com/`.

[6] L. Ahn, M. Blum, and L. J. Hopper, N. Captcha: Using hard AI problems for security. In *Proceedings of EUROCRYPT'03*, Warsaw, Poland, May 2003.

[7] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM Press, 2000.

[8] B. Costales and E. Allman. *Sendmail*. O'Reilly and Associates, 2002.

[9] L. Cranor and B. LaMacchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.

[10] S. Fahlman. Selling interrupt rights: A way to control unwanted E-Mail and telephone calls. *IBM Systems Journal*, 41(4):759–766, 2002.

[11] J. M. Gómez Hidalgo, E. Puertas Sanz, and M. M. López. Evaluating cost-sensitive unsolicited bulk E-Mail categorization. In *Proceedings of JADT-02, 6th International Conference on the Statistical Analysis of Textual Data*, Madrid, ES, 2002.

[12] R. Hall. How to avoid unwanted E-Mail. *Communications of the ACM*, 41(3):88–95, 1998.

[13] R. Hindle. An introduction to the Spambayes Project. *Linux Journal*, 2003(107), 2003.

[14] J. Ioannidis. Fighting spam by encapsulating policy in E-Mail addresses. In *Proceedings of NDSS'03*, San Diego, CA, Feb. 2003.

[15] J. Klensin. Simple Mail Transfer Protocol. Request for comments 2821, Internet Engineering Task Force, 2001.

[16] J. Ousterhout. *Tcl and the Tk toolkit*. Addison-Wesley, 2001.

[17] J. Postel. Simple Mail Transfer Protocol. Request for comments 821, Internet Engineering Task Force, 1982.

[18] G. Robinson. A statistical approach to the spam problem. *Linux Journal*, 2003(107):3, 2003.

[19] J. Schneider. A comparison of event models for naive bayes anti-spam E-Mail filtering. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, 2003.

[20] Sleepycat Software, editor. *Berkeley DB*. Que, 2001.

[21] D. Still. *The qmail handbook*. Apress LP, 2001.

[22] A. Turing. Computing machinery and intelligence. *Mind*, 49:433–460, 1950.

[23] A. Weiss. Ending spam's free ride. *netWorker*, 7(2):18–24, 2003.