

Bayesian Classification of Unsolicited E-Mail

Michael J. Fromberger¹
Michael.J.Fromberger@Dartmouth.EDU

Version 1.6—April 3, 2004

In this article, I examine the idea of using naïve Bayesian probabilistic techniques to recognize and filter unsolicited commercial e-mail (also known as “spam”). This work is based upon the ideas described by Paul Graham in his essay entitled “A Plan for Spam.”[1]

My purpose is to explicitly formalize the mathematics behind this idea, and further, to explore the possibilities of implementing the technique as part of a more general e-mail filtering system.

Definitions

Let Σ be a finite alphabet of symbols. Define a **word** as any finite sequence $w \in \Sigma^*$.

A **lexicon** is any countable subset of Σ^* .

Given a lexicon W , a **message** m over W is any subset of $W \times \mathbb{N}$ with the following two properties:

1. For any w , there is at most one x such that $(w, x) \in m$.
2. For each $(w, x) \in m$, $x > 0$.

Given a word w and a message m , we say that w **occurs in** m if there exists some x for which $(w, x) \in m$. Equivalently, if w occurs in m , we say that m **contains** w .

The **frequency** of a word w in a message m is defined as the the number of times w occurs in m , or 0 if w does not occur in m . Formally, we denote this quantity by:

$$\text{freq}(w, m) = \begin{cases} x & \text{if } (w, x) \in m \\ 0 & \text{otherwise} \end{cases}$$

The **vocabulary** of a message m , denoted $V(m)$ is the set of distinct words that occur in m , or in other words:

$$V(m) = \{w \in \Sigma^* \mid \exists x : (w, x) \in m\}$$

The **length** of a message m , denoted $\text{len}(m)$, is the total number of occurrences of all the words that occur in m . Formally, $\text{len}(m) = \sum_{(w,x) \in m} x$

¹Copyright © 2003–2004 Michael J. Fromberger, All Rights Reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this document, to use, copy, merge, publish, and distribute the document without limitation, subject to the following condition: This copyright notice and this permission shall be included in all copies or substantial portions of the document.

A **corpus** is a set of messages over some lexicon. We extend the notion of word frequency to a corpus in the natural way. Given a word w and a corpus C , $\text{freq}(w, C) = \sum_{m \in C} \text{freq}(w, m)$. We say that w occurs in C if $\text{freq}(w, C) \neq 0$.

Similarly, we extend the definitions of vocabulary and length to a corpus in the obvious ways. Given a corpus C , $V(C) = \bigcup_{m \in C} V(m)$ and $\text{len}(C) = \sum_{m \in C} \text{len}(m)$.

Analysis

Our ideal goal would be to design an algorithm A that can perfectly distinguish spam messages from good ones. In other words, given any message m , evaluating $A(m)$ yields *true* if m is a “spam” message, and *false* otherwise.

Unfortunately, short of creating an artificial intelligence, we do not yet know how to achieve this goal perfectly. However, the insight of Paul Graham’s paper is that a human is a good *oracle* for this problem. Given any message m , I can perfectly categorize m as either “good” or “spam,” according to my own criteria. This suggests the possibility of applying simple machine learning techniques to the problem, using my own judgment as “expert testimony.”

For purposes of analysis, therefore, let us model our own judgement about what is spam as a predicate **spam**() capable of perfectly distinguishing spam e-mail from good e-mail. In other words, given any message m , **spam**(m) is true if m is “spam,” and \neg **spam**(m) is true if m is good. For an algorithm A to be a perfect spam-detector, would require that $A(m) = \mathbf{spam}(m)$ for all m . However, we might be content with an algorithm A that has the following properties: For any message m ,

1. $A(m)$ returns either *true* or *false*.
2. With high probability, $A(m) = \mathbf{spam}(m)$.

Let us first consider the problem of classifying messages for which we already know the correct classification. Let’s assume we have a corpus C of messages, and that for each $m \in C$, we know the truth-value of **spam**(m). Let $S = \{m \in C \mid \mathbf{spam}(m)\}$, and let $G = (C \setminus S) = \{m \in C \mid \neg \mathbf{spam}(m)\}$. In other words, S is the set of all the spam messages, and G is the set of all the good messages.

For the sake of intuition, suppose we throw all the words that make up the messages of C into a bag, shake up the bag, and then reach in and draw out a word w at random. What is the probability that the message w originally came from, say m , was spam? Let us denote this probability as $P(m \in S \mid w)$, the probability that m is spam given that we observe w .

Using Bayes’s rule, we can express this probability as follows:

$$P(m \in S \mid w) = \frac{P(w \mid m \in S) \cdot P(m \in S)}{P(w)} \tag{1}$$

To evaluate this equation, we need simply determine the values of each of the probabilities on the right-hand side.

First, let’s compute the probability of choosing our particular w , out of all the words in the corpus. That is simply the number of possible ways of choosing w , out of all the possible ways of choosing

any word from $V(C)$:

$$P(w) = \frac{\text{freq}(w, C)}{\text{len}(C)} \quad (2)$$

Now suppose we have a bag containing only the words from the messages in S . What is the probability that, upon reaching into the bag, we would draw out w ? It is the number of times w occurs in S , out of the total length of S :

$$P(w | m \in S) = \frac{\text{freq}(w, S)}{\text{len}(S)} \quad (3)$$

Finally, if we were to just reach into a bag of all the words in the corpus, and draw one out at random, what is the probability that we pick one of the words that was contributed by a spam message? It is the number of ways of choosing an arbitrary word from S , out of the number of ways of choosing a word from C , or:

$$P(m \in S) = \frac{\text{len}(S)}{\text{len}(C)} \quad (4)$$

Substituting these quantities into (1) yields:

$$\begin{aligned} P(m \in S | w) &= P(w | m \in S) \cdot P(m \in S) \cdot \frac{1}{P(w)} \\ &= \frac{\text{freq}(w, S)}{\text{len}(S)} \cdot \frac{\text{len}(S)}{\text{len}(C)} \cdot \frac{\text{len}(C)}{\text{freq}(w, C)} \\ &= \frac{\text{freq}(w, S)}{\text{freq}(w, C)} \end{aligned} \quad (5)$$

This equation allows us to specify the probability that m is spam given that we have observed a single word w . Most messages consist of multiple words, however—so for an arbitrary message $m \in C$, we should extend equation (5) to compute the probability that $m \in S$ given the entire vocabulary of m . If $V(m) = \{w_1, w_2, \dots, w_n\}$, then, making the naïve assumption that words in m occur with independent probabilities,

$$P(m \in S | V(m)) = \frac{\prod_{i=1}^n P(m \in S | w_i)}{\prod_{i=1}^n P(m \in S | w_i) + \prod_{i=1}^n (1 - P(m \in S | w_i))} \quad (6)$$

Provided the messages in S and G are of a sufficiently different character, it is reasonable to expect that for any $m \in S$, $P(m \in S | V(m)) > 0.5$ and for any $m \in G$, $P(m \in S | V(m)) < 0.5$. Indeed, the more dissimilar the two groups are, the wider this gap should be. Conversely, if the two are very close to 0.5, it is reasonable to argue that the messages in S and G are not sufficiently well differentiated to form a good training corpus.

Now, we will make a bold assumption:

If C is sufficiently large and diverse, then given a novel message $m \notin C$, $P(m \in S | V(m) \cap V(C)) \approx 1$ if and only if **spam**(m).

In other words, I'm making the assumption that if I have a large and “interesting” enough training corpus, the distribution of $V(C)$ will produce a good model of the abstract source of the messages I will receive in the future, and that I can use this model to classify unknown messages fairly accurately. This is a big assumption! But, in practise, it appears to be a good one.

Given this assumption, we can use Equation (6) as an estimate of the probability that a new message m is also spam. To construct an algorithm for classifying messages, choose a value $0.5 < t < 1$ called the **threshold value**, and use the following procedure:

```

SPAM( $m$ ) is:
  let  $p := P(m \in S \mid V(m))$ :
    if  $p > t$ :
      return true.
    else:
      return false.
end.

```

Implementation Considerations

Before the above algorithm can be used effectively to classify new messages (*i.e.*, ones not occurring in the training corpus), there are a couple of minor technical problems which need to be resolved.

The first question we need to address is what to do about words which occur in a new message, but which were not part of the vocabulary of the training corpus. Given a new message m , let us say that a word $w \in V(m)$ is **novel** relative to C if $w \notin V(C)$. Since $w \notin V(C)$, we cannot use equation (5) to compute a probability estimate, so we must adopt some other approach.

One simple strategy would be to assign novel words a fixed probability estimate $0 \leq q \leq 1$. Then, we could restate (5) as:

$$P(m \in S \mid w) = \begin{cases} \frac{\text{freq}(w,S)}{\text{freq}(w,C)} & \text{if } w \in V(C) \\ q & \text{otherwise} \end{cases} \quad (7)$$

Let us call q the **novelty bias**. If $q < 0.5$, novel words decrease the estimated spam probability of the message, and we will say that our estimator is **forgiving**. If $q > 0.5$, novel words increase the estimate, and we will say that our estimator is **suspicious**. When $q = 0.5$, we will say that the estimator is **neutral**.

A second problem is what to do with words which occur in the vocabulary of spam messages, but not in the vocabulary of good messages, or vice versa. On the surface, this appears harmless—even helpful. After all, suppose there is some $w_s \in V(m)$ where $w_s \in V(S)$ but $w_s \notin V(G)$. Then by equation (7), $P(m \in S \mid w_s) = 1$. That is, w_s is an excellent predictor of spam.

Here is the trouble: If such a w_s exists, then the $\prod_{i=1}^n (1 - P(m \in S \mid w_i))$ term in the denominator of equation (6) will go to zero, and the spam probability 1 will be computed for m , as desired. But now suppose there is also (simultaneously) a $w_g \in V(m)$ such that $w_g \in V(G)$ but $w_g \notin V(S)$. By the same reasoning, the other term in the denominator of equation (6) goes to zero, and the result is no longer well-defined.

A simple solution in this case is to re-define (7) so that the probability estimate is never exactly 1 or exactly 0. Choosing a small positive constant ϵ as an adjustment value, we can extend (7) as follows:

$$P(m \in S \mid w) = \begin{cases} q & \text{if } w \notin V(C) \\ 0 + \epsilon & \text{if } w \notin V(S) \\ 1 - \epsilon & \text{if } w \notin V(G) \\ \frac{\text{freq}(w,S)}{\text{freq}(w,C)} & \text{otherwise} \end{cases} \quad (8)$$

Paul Graham suggested this strategy, and for his trial implementation, he chose $\epsilon = 0.01$. This value seems to work well in practise.

Dealing with Errors

Another significant problem is how to deal with incorrect answers. Since the correctness of this algorithm is dependent upon a static model of an unknown distribution, it is almost inevitable that there will be at least some errors. These will fall into two general categories:

1. **False positives:** The algorithm classifies m as spam ($A(m) = true$), but $\mathbf{spam}(m) = false$.
2. **False negatives:** The algorithm classifies m as good ($A(m) = false$), but $\mathbf{spam}(m) = true$.

From a theoretical perspective, there is no reason to distinguish between these two classes of error—they are errors, and that is all that matters. The difference to a human user of the system, however, is substantial. *False negatives* are annoying, but mostly harmless: Obviously, you don't want to receive spam in your mailbox, but if the system does miss one now and again, it will not cause significant difficulty.

False positives, on the other hand, are much more serious. If the algorithm classifies an important message from a family member or business associate as spam, you might not see that message in a timely fashion (or at all, depending upon the disposition of your filters). Clearly, a false positive is a much more dangerous error, in this problem domain, than a false negative.

Before we can address this problem, let us first consider how to detect errors of classification. False negatives are easy to spot, because they show up in your mailbox along with your good messages, and you open them to discover they are actually spam. False positives, on the other hand, might go undetected for some time. If spam is discarded outright, you will never see a misclassified message at all. Even if your filtering scheme causes spam to be collected in a separate mailbox for manual review, you may not discover the error until the next time you perform that review. Even then, you might overlook the misclassified message in a cursory review, especially if it is buried amid a lot of correctly-classified spam.

Obviously, this is a potentially difficult problem, and some of the solutions lie outside the ambit of this paper. For those who are interested in exploring this question further, Eric Kidd has written an excellent article on “Bayesian whitelisting,” [2] which offers a very promising solution. Here is a brief synopsis of the intuition behind his article:

In practise, a common source of false positives occurs when someone whose messages you want to keep—for instance, a friend or relative or business associate—sends you a message which would otherwise be considered spam. The algorithm described here does not take the sender into consideration; it only examines the content of the message. This suggests that one productive strategy might be to turn the problem around—instead of asking, “Is this message likely to be spam?” you ask, “Is this message likely to be good?”

Armed with this insight, we need only make a small adjustment to Equation (1) to construct such a filter for good messages:

$$P(m \in G | w) = \frac{P(w | m \in G) \cdot P(m \in G)}{P(w)} \tag{9}$$

Following through the same computations as we did for spam, we get the practical equation:

$$P(m \in G \mid w) = \begin{cases} q & \text{if } w \notin V(C) \\ 0 + \epsilon & \text{if } w \notin V(G) \\ 1 - \epsilon & \text{if } w \notin V(S) \\ \frac{\text{freq}(w,G)}{\text{freq}(w,C)} & \text{otherwise} \end{cases} \quad (10)$$

As before, this can be expanded to the general case:

$$P(m \in G \mid V(m)) = \frac{\prod_{i=1}^n P(m \in G \mid w_i)}{\prod_{i=1}^n P(m \in G \mid w_i) + \prod_{i=1}^n (1 - P(m \in G \mid w_i))} \quad (11)$$

From this, we build the simple Bayesian “good filter” using a simple twist on the original spam filter procedure:

```

GOOD(m) is:
  let p := P(m ∈ G | V(m)):
    if p > t:
      return true.
    else:
      return false.
end.

```

In fact, you can combine both approaches to classify a message as “good”, “neutral”, or “spam”, as in:

```

CLASSIFY(m) is:
  if GOOD(m) = true:
    return good.
  else if SPAM(m) = true:
    return spam.
  else:
    return neutral.
end.

```

This combined approach might be reasonably expected to reduce the rate of false positives. Even so, there is still the possibility of errors. For the moment, therefore, let’s suppose you have already *located* a misclassified message *m*, and now wish to have the system “learn” from its error. The simple solution is to add *m* to the training corpus, yielding a new corpus $C' = C \cup \{m\}$, with the correct classification supplied by the human oracle, and recompute all the probability estimates.

It is also possible to effect a tradeoff between errors and accuracy, by adjusting the parameters of the algorithm:

Parameter	Adjustment	False +	False -	Recommended
threshold (<i>t</i>)	raise	decrease	increase	<i>t</i> = 0.9
	lower	increase	decrease	
novelty bias (<i>q</i>)	more forgiving	decrease	increase	<i>q</i> = 0.4
	neutral	<i>no effect</i>	<i>no effect</i>	
	more suspicious	increase	decrease	

In practise, not all the words of a new message are “interesting.” For instance, those $w \in V(m)$ for which $P(m \in S \mid w) \approx \frac{1}{2}$ do not contribute significantly toward a meaningful classification of the message. To save a bit of computation, an implementation might consider only those w_i for which $|P(m \in S \mid w_i) - 0.5|$ is as large as possible. We might, for instance, examine only the k greatest order statistics of w_i when ordered by $P(m \in S \mid w_i)$. Let us call k the **interest threshold**.

The interest threshold should be high enough that you are considering a representative sample of the message’s interesting words—but not so many that you wind up wasting time examining lots of words whose contribution to the classification of the message is minimal. In practical tests, it seems that a value in the range $15 \leq k \leq 25$ works well.

Pragmatic Considerations

In order for the system to “learn,” we need to periodically adjust the training corpus and recompute the probability estimates. This suggests that we need to retain the entire original corpus. And yet, it would be terribly inefficient to recompute the spam probabilities for the entire corpus for each message classified. Fortunately, it is possible to store the relevant information in tabular format, so that the original corpus does not need to be retained.

Given a corpus C over some lexicon W , let us define a **system state** T as a subset of $(W \times \mathbb{N} \times \mathbb{N})$ so that:

$$T = \bigcup_{w \in V(C)} \{(w, x, y) \mid x = \text{freq}(w, G), y = \text{freq}(w, S)\}$$

For any given $w \in V(C)$, let $T(w)$ denote the unique $(w, x, y) \in T$.

Because S and G are disjoint by definition, we can take advantage of the relation $\text{freq}(w, C) = \text{freq}(w, S) + \text{freq}(w, G)$, for any w . This allows us to restate the last case of equations (8) and (10) as:

$$P(m \in S \mid w) = \frac{\text{freq}(w, S)}{\text{freq}(w, S) + \text{freq}(w, G)} \quad (12)$$

$$P(m \in G \mid w) = \frac{\text{freq}(w, G)}{\text{freq}(w, S) + \text{freq}(w, G)} \quad (13)$$

Now, if we are given a new message $m \notin C$ to add to the corpus, we need only compute a new system state T' to reflect this new information. We can do this as follows: Let

$$T_m = \bigcup_{w \in V(m)} \left\{ \begin{array}{l} T(w) \\ (w, 0, 0) \end{array} \middle| \begin{array}{l} \text{if } w \in V(C) \\ \text{otherwise} \end{array} \right\}$$

Then, let:

$$T'_m = \bigcup_{(w, x, y) \in T_m} \left\{ \begin{array}{l} (w, x, y + \text{freq}(w, m)) \\ (w, x + \text{freq}(w, m), y) \end{array} \middle| \begin{array}{l} \text{if } \text{spam}(m) \\ \text{otherwise} \end{array} \right\}$$

We can now define $T' = (T \setminus T_m) \cup T'_m$. Informally, we have added all the words from the new message into the system state, either adding their frequencies to existing entries, or creating new entries for the novel words. Notice that the construction depends on the oracle’s correct classification of m .

The implication of this construction is that it is not necessary to maintain the corpus in its original structure for the algorithm to work correctly—only the words, and their frequencies, are required. It is even possible to remove messages from the corpus, by a similar construction which replaces addition with subtraction.

Alternate Probability Measures

Equations (8) and (10) give one reasonable definition of spam/good probability based on the word frequencies throughout the entire training corpus. However, it is not the only possible formulation we can use. Instead of looking at the overall frequency of w in the corpus, one attractive option is to look at the expected number of occurrences of w in each message. To do this, let us define the **density** of w in corpus C as:

$$\Delta(w, C) = \frac{\text{freq}(w, C)}{|C|} \quad (14)$$

To understand why this measure might be useful, consider the following scenario. You have a training corpus for which $|S| \gg |G|$. Suppose there is a common word w which occurs with about equal frequency in both good and spam messages (for instance, suppose $w = \text{‘the’}$ in a corpus of English messages). However, because there are many more spam messages than good ones, $\text{freq}(w, S) \gg \text{freq}(w, G)$, which will make w appear to be a strong indicator of spam, when in fact it is mostly uninteresting.

Conversely, suppose $|S| \ll |G|$, and w is a word that occurs very frequently in spam messages, but very infrequently in good messages. Here, because there are many more good messages than bad ones, it may turn out that $\text{freq}(w, S) \approx \text{freq}(w, G)$, and so we will find that $P(m \in S \mid w) \approx 0.5$, making w appear uninteresting—when, in fact, it is probably a good predictor of spam.

These examples suggest that we could obtain some benefit by *normalizing* each frequency, dividing it by its cardinality in the corpus, rather than using its magnitude directly. This normalization weights each w toward the side in which it has higher frequency *per message*, rather than where it is more frequent overall. Using this measure, we can restate the general case of (8) and (10) to use the density function rather than frequency, giving:

$$P(m \in S \mid w) = \frac{\Delta(w, S)}{\Delta(w, S) + \Delta(w, G)} \quad (15)$$

$$P(m \in G \mid w) = \frac{\Delta(w, G)}{\Delta(w, S) + \Delta(w, G)} \quad (16)$$

As before, we must deal with novel words and words which occur only in S or only in G , but we can use the same solutions here as we did for the original statement of (8) and (10).

Here is an example that illustrates the difference between this new probability measure (15) and the original system using equation (8). Suppose that

$$\text{freq}(w, S) = 5, \text{freq}(w, G) = 5, |S| = 5, \text{ and } |G| = 100$$

Then, equations (8) and (10) give

$$P(m \in S | w) = P(m \in G | w) = \frac{5}{5+5} = 0.5,$$

while equations (15) and (16) give

$$P(m \in S | w) = \frac{1}{1+0.05} = 0.9524 \text{ and}$$

$$P(m \in G | w) = \frac{0.05}{1+0.05} = 0.0476.$$

In the first case, we would almost certainly ignore w since it gives us no real information about the message; in the second case, we would take w as a strong indicator of spam or a very weak indicator of goodness. So, while these two measures are both interesting, they are not equivalent.

Filtering in Practise

Provided you have a sufficiently large and diverse training corpus, the probabilistic strategies described here appear to perform well at classifying messages. However, the possibility that you may get false positives, even if they occur with low probability, is a very serious consideration.

To safeguard against this possibility, Bayesian filtering may be combined with other filtering strategies, for example:

1. **Blacklisting:** A *blacklist* is a filter that explicitly discards messages based on certain structural properties. For instance, you might blacklist all e-mail from a certain user who always sends you spam. Or, you could reject any message which was sent through a site known to have an open relay. Or, you might discard any message which does not contain your own address in one of the standard recipient fields.²
2. **Whitelisting:** By analogy to a “blacklist,” a *whitelist* is a filter that explicitly selects messages to be preserved, rather than discarded. A typical whitelist for e-mail would preserve all e-mail sent from the addresses of your friends and colleagues, even if they might otherwise appear to be spam.
3. **Collating:** By organizing your messages into various categories, you can refine the environment in which other filters are applied. For instance, you might want to filter e-mail sent to various mailing lists into separate mailboxes. Often, a message you would consider spam if you received it as personal e-mail, would be perfectly acceptable as mailing list traffic.

By collating messages before applying other filters, you can take advantage of different data sets to improve the results for Bayesian or other types of filtering.

In addition to deciding what *types* of filters you will use, you will also need to decide what *order* they will be applied in. Ordering has a profound effect on the quality of the results. For instance, if your blacklist runs before your whitelist, you would get different results if your best friend sent you a note through an open relay, than if the order were reversed. More subtly, the effect of a false positive from a Bayesian filter is much more serious if that filter is run *prior* to your whitelist, than when it is the other way around.

Of course, both blacklists and whitelists represent fairly radical approaches to filtering. You could, instead, create a kind of “graylist,” a filter which adds or detracts “points” from a message based on various positive or negative attributes the message has. A message enters the graylist with a score of zero, and leaves with some positive or negative number of points. A message which is sufficiently negative gets blacklisted; one with enough good karma gets whitelisted. Messages in the middle can be passed on to other filters for further consideration.

An approach I have found useful is the following:

1. Use a Bayesian whitelisting filter to mark messages which definitely should be kept.
2. Among the remaining messages, use a Bayesian blacklisting filter to mark messages which definitely should be considered spam.

²This latter must be used with caution, especially if you subscribe to mailing lists.

Alternatives to Filtering

One negative aspect of filtering is that it puts the computational burden on the *recipient* of the unwanted e-mail, rather than on the sender where it belongs. Spammers often send thousands of messages at a time, and the cost of doing so—both monetary and computational—is negligible, at the source. If each recipient has to run an elaborate filter to discard this unsolicited garbage, a much greater combined cost has been incurred by the recipients than by the sender.

In the long run, I suspect we may need to work with our legislature to come up with some kind of legal means for controlling the proliferation of bulk commercial e-mail, just as we have done within the traditional hard-copy postal service. This is a very difficult problem from a legislative perspective, given the open and unregulated nature of the Internet. Because most of the obvious legislative “solutions” to the spam problem are likely to undesirably restrict the freedom of its use, I feel it is incumbent upon the marketplace to put some negative pressure on the producers of this nuisance.

One attractive solution, called “Hashcash,”[3] was proposed by Adam Back based on the work of Cynthia Dwork and Moni Naor.[4] The basic idea is to require the sender of a message to provide proof that (s)he expended a certain amount of computational effort prior to sending that message.

The implementation is fairly elegant: Let H be a one-way hash function so that, given x , it is easy to compute $H(x)$, but very difficult to find $y \neq x$ so that $H(y) = H(x)$. Practical examples of this sort of function include the MD5 and SHA1 algorithms.³

To send a message, the sender must generate k bits of “postage” for each recipient, where k is determined by the receiver. To do this, the sender composes her message m , and then computes $h = H(m)$. Now, by simple exhaustive search, she finds a value y for which the first k bits of $H(y)$ are exactly equal to the first k bits of h . She then sends $\langle m, y \rangle$ off to the receiver.

To verify the postage, the receiver computes $H(m)$ and $H(y)$, and checks to make sure that their first k bits are identical. If they are, he accepts the message, otherwise, he rejects it. If he accepted it, he would also store a copy of y in a database, so that the sender couldn’t use it again to send the same message to somebody else on the same server. If this scheme were implemented on the primary mail exchangers for each domain, the the number of times a spammer could re-use a particular hash-stamp would be quite limited.

The amount of work done by the sender to compute the hash is many times greater than the amount of work done by the recipient to verify it. Furthermore, the recipient can raise the bar simply by choosing a higher k , and the work of verifying postage remains constant. The value of k should be chosen so that the postage token takes a second or two to compute (as computers get faster, k gets raised, just like regular postage). For individuals sending a few dozen messages in the course of a day, this is no big deal. But for the spammer, trying to send out thousands of messages in one inexpensive transaction, it is fatal.

If this scheme could be smoothly integrated into mail servers, it has the potential to greatly increase the cost of sending out bulk e-mail, without putting any undue burden on normal e-mail users. Since the main attraction of bulk e-mail is its low cost, this is a good market-driven solution for the problem.

³These have not been proven to be one-way functions, but they are sufficiently difficult to be adequate for this purpose.

References

1. Paul Graham, “A Plan for Spam”:
<http://www.paulgraham.com/spam.html>, August 2002.
2. Eric M. Kidd, “Bayesian Whitelisting: Finding the Good Mail Among the Spam”:
<http://www.randomhacks.net/stories/bayesian-whitelisting.html>
3. Adam Back, “Hashcash: A Denial-of-Service Counter-Measure”:
<http://www.cypherspace.org/hashcash/hashcash.pdf>, August 2002.
See also: <http://www.cypherspace.org/hashcash/>
4. Cynthia Dwork and Moni Naor, “Pricing via Processing, or Combatting Junk Mail” in *Advances in Cryptology—CRYPTO '92*, Proceedings. August 1992.
Reprinted in *Lecture Notes in Computer Science*, Vol. 740/1993, pg. 139.
Heidelberg: Springer-Verlag (ISSN 0302-9743).