from *SIAM News*, Volume 36, Number 9, November 2003

Math 1, Spam 0

Even as governments consider anti-spam legislation, computer security experts say that laws alone are unlikely to solve the problem. But hope has emerged in the form of two promising mathematically based measures for outwitting the spammers.

By Dana Mackenzie

Remember what it was like to open your e-mail folder day after day and see nothing but real messages, from real people? Say, 10 or 15 years ago—way back in the early days of the Internet era? We hardly realized how good we had it.

Nowadays, an estimated 40 to 50% of the e-mail traffic on the Internet is spam: unwanted solicitations for Viagra, get-rich-quick schemes, offers for low-interest mortgages, and the like. For many Internet users, checking e-mail has become a depressing ritual of hitting the delete button. The Internet's original "killer app" has simply become a killer—of time and of productivity.

Fortunately, the news is not all bad. This September, California passed the nation's toughest anti-spam law, and the U.S. Congress is also considering five different anti-spam bills. But spammers can be amazingly elusive, using bogus addresses or covering their tracks by forwarding their mail through unsuspecting third parties. For that reason, computer security experts don't expect laws alone to solve the problem.

If you can't catch spammers, maybe you can ruin their business model instead. That's the main idea behind two very different anti-spam measures that have emerged recently. Both methods outwit the spammers with mathematical tricks. Although we may never return to the days of no spam at all, perhaps we can at least hope for an "Ivory Snow" level of purity—an e-mail inbox that is 99 and 44/100 percent spam-free.

An Adaptive Solution

Most users' first line of defense against spam is a filter. Any commercial software contains filters: You can tell your computer to delete messages that contain words like "Viagra" (or worse). Then you sit back and chortle as the offending messages tumble into your wastebasket. But gradually, you start getting more and more messages that dodge the filter, with words like "V*I*A*G*R*A" or "V1agra." That is the big weakness of filtering rules: They are static, and spammers are not.

Over the past year, many anti-spam programmers have been working on filters that adapt to the spammers. These filters have to be trained, by being shown (say) 100 examples of spam and 100 valid messages. If 19 spams contain the word "Viagra" and only one valid message does, the filter will conclude that a message containing the word "Viagra" has a 95% chance of being spam. This calculation is justified by Bayes's theorem from statistics, a simple rule for quantifying your "degree of belief" in a statement. Bayes's theorem also gives you a way to update your degree of belief as new evidence comes in. Thus, if a lot of spammers start selling tulips, the filter will automatically increase the spam probability of the world "tulip." The spamminess of a whole message can be computed from the spam probabilities of the words in it. The spam filter rejects messages whose spam probability exceeds some threshold, such as 0.99.

The first trials of Bayesian spam filters, published in 1998, were disappointing. One group reported that their filter caught about 90% of spam, but also that it discarded about 1% of valid messages—an unacceptably high rate for most users. But Bayesian filters are getting better. This year, at a spam conference at the Massachusetts Institute of Technology, several speakers reported filtering rates above 99% and false-positive rates near zero. The improved filters pay more attention to the context and to groupings of words. For example, a filter written by Paul Graham, who organized the MIT conference, gives the word "free" a spam probability of only 0.65 if it occurs in the message text, but 0.9782 if it occurs in the header and 0.9999 if it is also capitalized. Most importantly, these numbers are not static: They evolve as the spam itself evolves, because the program learns from its mistakes.

Until recently, Bayesian filters were only a plaything of computer geeks. For that reason, spammers had little reason to fight back against them. However, the newest version of America Online's software package (AOL 9.0) includes a Bayesian filter, and so will the next release of Microsoft Office. As Bayesian filters become ubiquitous, spammers will have to look for ways to fool them. Will they succeed? "That kind of question is hard to answer," says Graham, "but if someone told me to find a way to beat a Bayesian filter, I don't think I could."

According to Bill Yerazunis of Mitsubishi Electronic Research Labs, a filter that catches 99.5% of spam could drive spammers out of business. He explains that advertisers typically pay a spammer \$200 to \$500 to send out a million e-mail messages. If they generate 100 responses, the advertiser has paid 2 to 5 cents per response. Direct mail, on the other hand, typically costs around \$5 per response. If filters wiped out 99.5% of spam, then the cost per response would go up by a factor of 200 and spam would no longer have an economic advantage over postal mail.

A Puzzling Solution

Back in the early 19th century, ordinary mail had a problem similar to the one now affecting the Internet: The relative costs to senders and recipients were out of whack. Today, spam costs almost nothing for a spammer to send, but a recipient who looks at the message and manually deletes it incurs a perceptible cost in lost time. (Sift through a hundred pieces of spam and you'll agree.) Similarly, before 1840, ordinary mail cost nothing to send, and the recipient paid for it. But that year, England introduced the world's first postage stamp, the "Penny Black," which transformed mail from an expensive luxury into a tool for the masses.

Inspired by this history, a group of computer scientists at Microsoft Research are working on the Penny Black Project; the aim is to develop an electronic analogue of postage. Real money (in the form of e-cash, perhaps, or a token that the sender would purchase) is unlikely to work for Internet postage, however—it would be too easy for the spammers to steal the tokens.

The Penny Black Project instead uses "proofs of work," a concept first introduced in 1992 by Cynthia Dwork and Moni Naor of the IBM Almaden Research Center. The idea is simple: "If I don't know you, you have to prove to me that you spent ten seconds of CPU time just for me, and just for this message," says Dwork, who now works at Microsoft Research. For legitimate senders, spending ten extra seconds to send an e-mail message is no problem. Most of the time, you spend more time than that simply composing the message. But for spammers, those ten seconds are the kiss of death. The one thing that no one can steal is more seconds than there are in a day. For a single computer, the CPU time available in a day amounts to 86,400 seconds; a spammer who wanted to put electronic postage on millions of messages would thus need hundreds of computers. Dwork is betting that most spammers cannot afford that kind of expense.

Spam costs almost nothing for a spammer to send, but a recipient who looks at the message and manually deletes it incurs a perceptible cost in lost time. (Sift through a hundred pieces of spam and you'll agree.)

It is far from obvious that such a "proof of work" is feasible. The token that the sender attaches to a message must involve a computation that is neither too easy (so there are no shortcuts for the spammer) nor too hard (you and I don't want to sit at our computers for hours to send one message). It needs to take about the same amount of time, whether the sender has the clunkiest Pentium 2 or the zippiest Pentium 4. And it has to be much easier for the recipient to verify that the computation has been done than it is for the sender to do it.

At this year's Crypto '03 conference in Santa Barbara, California, Dwork described a scheme that satisfies all of these requirements. The key is to make the computation involve lots of memory look-ups. (Dwork credits Michael Burrows of Microsoft Research for this idea.) The time a computer needs to fetch a piece of data from memory, about 2 microseconds, has very little to do with the kind of processor it has. In practice, Dwork's algorithm, developed with Naor and Andrew Goldberg of Microsoft Research, takes between 9 and 24 seconds on a very wide variety of machines and operating systems.

To run the proof-of-work algorithm, the sender's computer stores a large array of random numbers, *T*, in its memory. Each time it sends a message, it converts the message text, the date, the recipient's address, and an "attempt number" *k* into a string of 1's and 0's. It then applies a "hash function" that scrambles the string into a shorter string of 1's and 0's. The hash functions are designed to behave as "random oracles," meaning that their output is reproducible (which allows the recipient to verify the computation), yet even the tiniest change to the input data will change the output in a wild and unpredictable way. The new, shorter string produced by the hash function corresponds to a location in table *T*. The computer looks up that location, retrieves its contents, and then presents it to another hash function, which converts it into a new location to look up. It's very much like a treasure hunt, in which you keep fetching clues that tell you where the next clue is hidden.

After following 2048 clues, the computer gets to the end of the hunt and prints out a hashed version of the most recently accessed element of *T*. That is the stamp it attaches to the message to prove it has done the computation. But there's a catch—the stamp is valid only if the last 15 bits are 0's. If they're not, the computer has to add 1 to the attempt number *k* and start all over again.

Each part of this scheme is ingeniously designed to foil spammers. Because the initial data depends on the message, the date, and the recipient, the treasure hunt will start in a different place and take a different route every time. The treasure hunt has enough steps that it will require at least a few milliseconds for each round. And because the last 15 bits of the stamp have to be 0's, the sender's machine will (on average) have to go through 2^{15} (about 32,000) treasure hunts before it finds a valid stamp. The numbers work out so that the predicted amount of work for the sender's machine is about 10 seconds. The recipient's machine, on the other hand, needs to go through only the last of the treasure hunts—a task it can complete in a few milli-seconds.

As with any treasure hunt, the big question is whether shortcuts can be found. Dwork has good news and bad news. If the table *T* and the hash functions are *truly random*, she can prove that no shortcut exists. The bad news is that no one knows how to create a random oracle, only a reasonable facsimile. But this objection is more theoretical than practical. "I would love to have a proof that didn't use random oracles," Dwork says. "But the crypto and security community is very willing to accept them."

For people who don't trust Microsoft, an open-source electronic-postage system is also in the works. The program, called Camram (a rough acronym for "Campaign for Real E-mail"), is the work of a Massachusetts-based security engineer named Eric Johansson. At present, he says, it is not quite ready for widespread use—"It takes me four hours to install on a mail server, and I'm the one who knows what is going on!" But he hopes to have his system ready before Microsoft's. (Dwork does not know how soon proof-of-work systems might be incorporated into a Microsoft product.)

Unlike Bayesian filters, electronic postage can guarantee that a legitimate sender's message will get through. That's good business for the receiver, too. "I have a commercial client, a fruit and vegetable reseller, who absolutely cannot lose a message," says Johansson. "With a filter, he was constantly going into the spam trap and checking 400 messages a day," looking for false-positives. That is now a thing of the past.

According to Graham, the Bayesian filter expert, the big problem with electronic postage is a "chicken-and-egg problem." How can you rely on a program that inspects incoming messages for stamps if no one else is using them? One solution, which Johansson has programmed into Camram, is to send a "postage-due" notice to anyone who sends you an unstamped message. This would take the sender to a Web site where he could easily do the ten-second proof of work. (At the same time, the Web site would teach the sender about electronic postage. “It’s a seducible moment,” says Johansson.)

For people beleaguered by spam, the best news is that the two methods can team up. For example, the presence or absence of a stamp could be used as additional data for a Bayesian filter. Or if the Bayesian filter can’t decide whether a message is spam or not, it could return the message to the sender with a postage-due notice. The amount of work the sender has to do could even be tailored to the contents of the message. As Graham suggests, “If it says ‘Viagra,’ make them do the mother of all calculations.”

Dana Mackenzie writes from Santa Cruz, California.