

SmartProof¹

Joshua Goodman and Robert Rounthwaite
Microsoft Research MSN Safety Team
and
MSN Safety Team

Executive Summary

We describe SmartProof, an anti-spam challenge response system combining machine learning with computational challenges, human challenges and possibly micro-payments. Other techniques (Coordinated Spam Reduction Initiative, CSRI), described elsewhere, make sure mail from large, legitimate senders is delivered and prevent spoofing, while SmartProof uses challenge-response to make sure that mail from individuals and small businesses reaches the recipient. In combination with CSRI, the SmartProof approach will block essentially all spam, while still letting through all legitimate mail.

Overview

Monday, sometime in 2006, Indonesia, afternoon. A pornographer sends mail to Sunny Potter, Mike Stoff, and 100 million others. He expects to make \$1000 in profits from the 500 people who will respond.

Tuesday, Santa Clara, CA, morning. Sunny Potter sits down to read her email. As usual, 95% of her mail is spam – spam volume has been growing exponentially. She sees one message that looks like it might be legitimate, and opens it. She feels almost ill at the picture that she sees (from the Indonesian pornographer.) She wishes her company would upgrade their software to the latest anti-spam technology. Next, Sunny Potter goes through the business cards she collected from a trade show, noticing Mike Stoff's. She emails him the brochure about her product. A moment later, a message appears on her machine. "I'm sorry. You're message looked like it might be spam. Please [click here](#) to download and execute a program that will solve a 1 minute computational challenge; or please [click here](#) to solve a simple puzzle; or please [click here](#) to make a credit-card based micro-payment." Unfortunately, her machine is incredibly old and slow, and she's also paranoid about downloading programs, even trusted code. So, she clicks on the second link. A little puzzle appears – a word with dots and lines crossing it. She reads the word, and types it in. A message appears "Thanks very much; your message will be delivered."

¹ This document does not represent Microsoft's product plans. Both authors are now affiliated with the MSN Safety Team, but this document was primarily written before the MSN Safety team existed, while both were members of Microsoft Research's Machine Learning and Applied Statistics team. This document represents only one possible course of action.

Tuesday, Bellevue WA, 4:00 PM. Mike Stoff comes back from a 4 hour meeting. He sighs as he realizes that he has received over 30 messages. He looks them over hopefully – perhaps a few are spam that he can dispose of quickly with the delete key. The Indonesian pornographer’s mail has been either deleted at his server or placed in a junk folder, along with almost 500 other spam messages correctly identified by the SmartScreen filter, but two messages stick out – “FREE Software” and “FW: Great Credit Card Deal!!!” How did messages that blatant get through the SmartScreen spam filter, he wonders? He opens the messages. The first is from GreatDiscountSoftware.com – a company he has actually bought software from and requested sale information from. GreatDiscountSoftware uses the Coordinated Spam Reduction Initiative (CSRI), a technique for identifying large companies that do not spam. Of course their mail gets through. (CSRI is covered in other documents.) Next he reads the credit card message – it turns out that it is from his brother, who is, of course, on his list of friends. This really is a great credit card deal, and he marks it to look at later. He notices another message, “the information you requested,” from Sunny Potter. Mike doesn't even know that Sunny had been challenged – he just receives the message in his inbox. He decides he'll look at that one later. Another message is from an old friend, letting him know he changed jobs and has a new email address. Even though it's a new address, the SmartScreen filter didn't challenge that message, because it doesn't look like spam. Mike realizes that all 30 messages are real and he has lots of work to do.

Tuesday, Bellevue WA, 6:00 PM. Mike has finally finished reading his mail. As usual, not a single one was spam. In the earlier meeting, the decision had been made that Mike should contact Dave Green about a “lucrative business proposal” and Mike decides to send out that email now. Unfortunately, the mail Mike sends looks an awful lot like spam to Dave’s SmartScreen filter, which automatically sends a challenge to Mike. Mike’s SmartScreen filter recognizes the challenge, and, in the background, automatically starts processing the computational challenge. 1 minute later, Mike’s email client replies to Dave’s, and Dave’s spam filter moves the message to Dave’s inbox. Neither Mike nor Dave notices. Now it’s 6:30 and time for Mike to go home. Another day has gone by; Mike hasn’t seen a single piece of spam. Every message sent to him by a non-spammer has gotten through. Every message he has sent has gotten through. The most important application on his desktop, his email system, has worked flawlessly.

In our vision of the future, every Microsoft email application will be enabled with the technologies we described above. Several of the key pieces have already been deployed. The machine learning SmartScreen filter already ships in most Microsoft email products, or will soon. The Human Interactive Proofs are already deployed at Hotmail, helping prevent mass account signups. Microsoft Research has already prototyped the computational puzzles. In this document, we describe these pieces in more detail and how to combine them to form SmartProof.

The first piece is the machine learning filter to recognize spam that is already shipping on Hotmail, MSN client software, and Outlook, and is slated to ship with Exchange and elsewhere. The machine learning filter, similar to a neural network, is trained to

recognize spam messages, using the 100,000 volunteers in Microsoft's Feedback Loop. The machine learning filter can be present on the server only or the client only, or some combination. In a sever-wide implementation, messages the filter is extremely confident are spam can be deleted on the server, with others marked for automatic sorting or filtering on a client. In a client-only implementation, the suspicious messages are placed in the junk folder. In either case, there is also a safe-list of correspondents, based on the address book (or in some cases maintained separately) to reduce even further the chance of accidentally misclassifying a legitimate e-mail as spam.

The machine learning filters work well on their own, but work even better when combined with computational challenges, Turing-test type puzzles, and micro-payments to effectively eliminate the chance that a legitimate e-mail will be missed. In this combined approach, for messages that are suspected as spam, a challenge is sent. The message sender/challenge receiver has a choice of solving the computational challenge, a simple puzzle, or perhaps making a small payment. While computational challenges, Turing tests, and micro-payments have been proposed before, there are significant advantages to this combined approach. First, by using machine learning to only challenge suspicious email, we improve the spam filter by effectively eliminating the chance of a mistake leading to a missed message while also reducing the number of challenges that must be exchanged – thus reducing the chance that a legitimate user won't bother to answer a challenge. Second, by allowing a choice of challenges – human, computer, or money – we make it much more likely that a mail sender/challenge recipient will be able to answer the challenge. Finally, because we challenge messages, instead of deleting them, we can set the machine learning filter to be more aggressive about classifying mail as spam. Overall, this combination of approaches works much better than any one alone.

In this document, we first briefly describe the cost of spam. Next, we outline each of the four pieces of the combined approach – the machine learning component, the computational challenges, the Turing-test puzzles, and the micro-payments. Finally, we describe the architecture for the system.

The Cost of Spam

The cost of sending spam is extremely low, about .01 cents per message, while the burden spam places on users is huge and growing. Brightmail now estimates that over half the mail sent on the internet is spam. A recent Infoworld poll identified spam as the number 1 “information technology disaster of the past year.” A report by the Federal Trade Commission found that 66% of spam had false information somewhere in the message. A report by the Pew Internet and American Life Project found that 12% of users spend a half hour or more per day dealing with spam

A Machine Learning Approach

Microsoft has been working on solving the spam problem since around 1997. Much of our focus has been on building a machine learning-based spam detector. This detector,

SmartScreen, has now shipped in MSN, Outlook, and Hotmail. It works by examining the content of the messages, including spam-like words like “money” or “free”, and special features, such as mail being sent in the middle of the night. The filter also takes into account indicators of good mail. For instance, the presence of words such as “photos” or “going” are indicative of good mail. This information is combined using a neural-network-like learning technique.

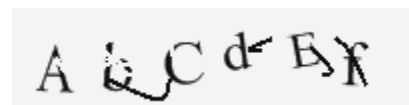
The filter is trained with Microsoft’s unique Feedback Loop technology, in which 100,000 hotmail users are polled daily, asked to classify one piece of mail as good or junk. This provides millions of high quality training messages for the filter. A unique aspect is the large, unbiased sample of good mail. Given the large number of different kinds of spam, it is sometimes easier to learn what good messages look like, and be suspicious of anything that is not close enough to this known good mail.

The filter is already quite effective at stopping spam, while maintaining a low false positive rate. We continue to improve the filtering technology, so far improving faster than the spammers. Among other improvements, we have found new special features that allow us to substantially improve over simple word-based technology alone.

Computational challenge, human tests and micro-payments

The computational challenge approach is based on work by Cynthia Dwork and Moni Naor (see <http://research.microsoft.com/research/sv/PennyBlack/junk1.pdf>). They describe functions that are hard to compute, but easy to check the computation of. Conceptually, these are a bit like jig-saw puzzles – it’s easy to make a jig-saw puzzle – rip a picture randomly. It’s hard to solve a jig-saw puzzle (I always give up!). And it’s easy to check that it has been solved correctly (do all of the pieces fit together and form the picture on the front of the box?) The actual puzzles they suggest are, of course, computational. The original versions are based on breaking simplified versions of cryptographic problems, and, more recently, Martin Abadi, Mike Burrows, Ted Wobber, Mark Manasse and Dan Simon have been working on related problems, using memory-bound functions, whose costs are more uniform across machines.

The human challenge component – a kind of simple Turing test – works by presenting a problem to the user. For instance, the user can be shown an image of a word that has been partially obscured or distorted: It may be very hard for a computer to recognize this word, but fairly easy for a human being.² We can use a human challenge as an option whenever a person will not, or cannot, solve a computational challenge. Patrice Simard, Rick Szeliski and Josh Benaloh of Microsoft Research have developed visual human challenges, which have been deployed on Passport/Hotmail to deter automatic account signups. Cormac Herley, Jasha Droppo and Joshua Goodman developed an



² This approach was originally suggested by Naor; see <http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps>

audio alternative challenge, involving recognizing a sequence of numbers and letters in the presence of noise. This challenge might be preferred by visually impaired users.

The micro-payment, or postage, approach, similar to that described in [The Road Ahead](#), is a final option. It gives yet another way to get past the spam filter, that can be used, for instance, by those with disabilities, or small business that may, when sending advertising to customers who have not safelisted them, hit a spam filter. The cost of the monetary challenge can be set so that it is slightly lower than the cost of solving the computational challenge. The money can be given to the recipient, the ISP, or a charity. A variation on the micro-payment approach allows the payment to be refunded if the message is not spam. Note that the micropayment approach is significantly harder to implement than the other two for technical and social reasons. Its deployment may be delayed until later.

Combining the approaches

The machine learning, computational challenge, Turing test, and micro-payment approaches work best when they are combined. In this combination approach, when a message arrives, the machine learning filter determines whether it might be spam. If it might be, then a challenge is sent back to the message sender. In particular, the message sender gets a message requesting that he solve his choice of challenges. If the sender is running an enabled client, the client automatically solves the computational challenge in the background or perhaps makes the micro-payment, and responds. If the client is not enabled, then the user must choose which challenge type to solve. Once a challenge is responded to, an additional message is sent, including the puzzle solution, and the message is moved to the receiver's inbox.³

Notice that this combination architecture has a number of advantages over the individual pieces. Compared to the machine learning filter alone, the combination allows more spam to be caught, because we can be more aggressive in calling mail junk, since legitimate senders can still get through. It simultaneously guarantees that mail from legitimate senders who are even slightly motivated will arrive. Compared to a human challenge, computational challenge or micro-payment alone, it is much easier to adopt. Since only mail that is spam-like is challenged, and then only for users not on a safe-list,

³ A variation on this main technique, using "ticket servers" has some additional advantages and disadvantages. In this version, outlined by Martin Abadi, Andrew Birrell, and Mike Burrows, <http://research.microsoft.com/research/sv/PennyBlack/demo/ticketserver.pdf>, the challenger requests a "ticket" from a ticket server. The ticket server responds with a ticket (and puts a hold on it so it cannot be reused). The user sees the message and either "releases" the ticket, or invalidates it if the message was spam. Mail senders can get tickets from ticket servers by solving computational challenges, Turing tests, or perhaps buying them. The ticket server approach – which is overall very similar to our server-less proposal – has some additional advantages. In particular, because a ticket is only finally spent when the message is detected as spam, the cost of a ticket can be much higher than in the server-less approach. This can increase the relative cost of sending spam to non-spam. Furthermore, tickets can be sent even without a challenge; this can reduce delays if the challenge would be sent by a client that is not always running. On the other hand, the ticket server approach is somewhat more complex, requiring either the creation of large ticket servers, or a system for federating ticket servers. It also requires a third party in what is otherwise a two party transaction. (We have suggested an approach in which ticket servers are used only for micro-payments; this substantially reduces the load on the ticket server, and makes it clear how to fund their construction and maintenance.)

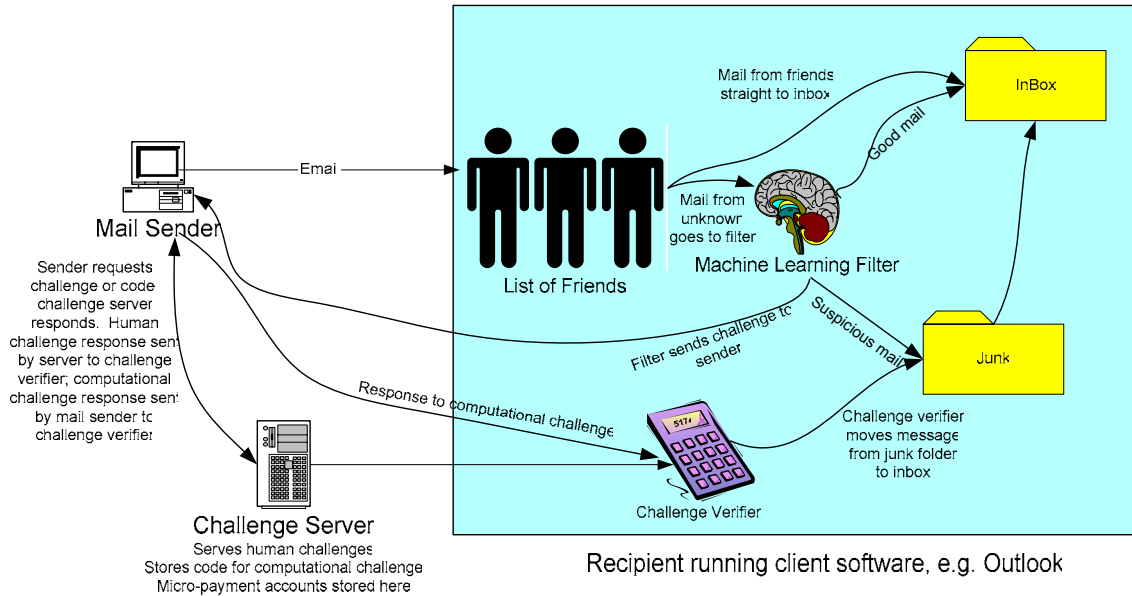
challenges to legitimate mail are rare. This means the filter can be installed, and there is little chance of annoying a legitimate user or friend who has changed addresses, or is sending mail to you for the first time. Compare this to, say, a pure computational-challenge based approach, where users not on a safe-list must always solve a computational challenge to send every message. This pure approach must have a relatively high cost of sending non-spam (or a relatively low cost for sending spam), since challenges are so common, as well as issues about how to achieve universal adoption. With the machine learning combination, because messages are challenged only rarely, we can keep the cost of these computational challenges high and the annoyance level low. In addition, we provide a natural adoption progression – senders who are using an unenabled client find value in adopting an enabled one, but are still able to send and receive mail with their current client, an important consideration given the slow adoption of new email clients.

Example architectures

Client-only architecture

There are a number of architectures that can be used to implement the SmartProof system. For instance, SmartProof can be implemented totally in a client program like Outlook. Alternatively, it can be implemented mostly in a server like Exchange, although cooperation from the client can be very helpful for some simple tasks like creating safe lists. We start by describing the simplest architecture, purely in the client (although ideally, there is a web server some place to serve Turing-test challenges, for downloading the computational challenge code, and for storing micro-payment accounts.) The architecture looks like the figure on the following page, where we leave out the server (e.g. Exchange) for simplicity.

In this architecture, a sender sends mail to the recipient. The recipient client checks a list of friends (the safe-list). If the sender is on the list of friends, the message goes to the Inbox. If not, the message is examined by the machine learning filter. If the filter determines that the message is unlikely to be spam, the message goes to the inbox; if not, the message goes to the junk folder, and a challenge is sent to the mail sender. The mail sender might solve the computational challenge, either automatically if he is running an enabled client, or manually by clicking on a link that downloads code to solve the challenge on his machine. Either way, he sends a special message back to the original recipient. The special message has a format indicating it is a challenge response; the challenge verifier verifies the correctness. The message may be retrieved from its temporary location, or may have been resent along with the response to the challenge. Either way, the recipient now gets the message in the inbox.

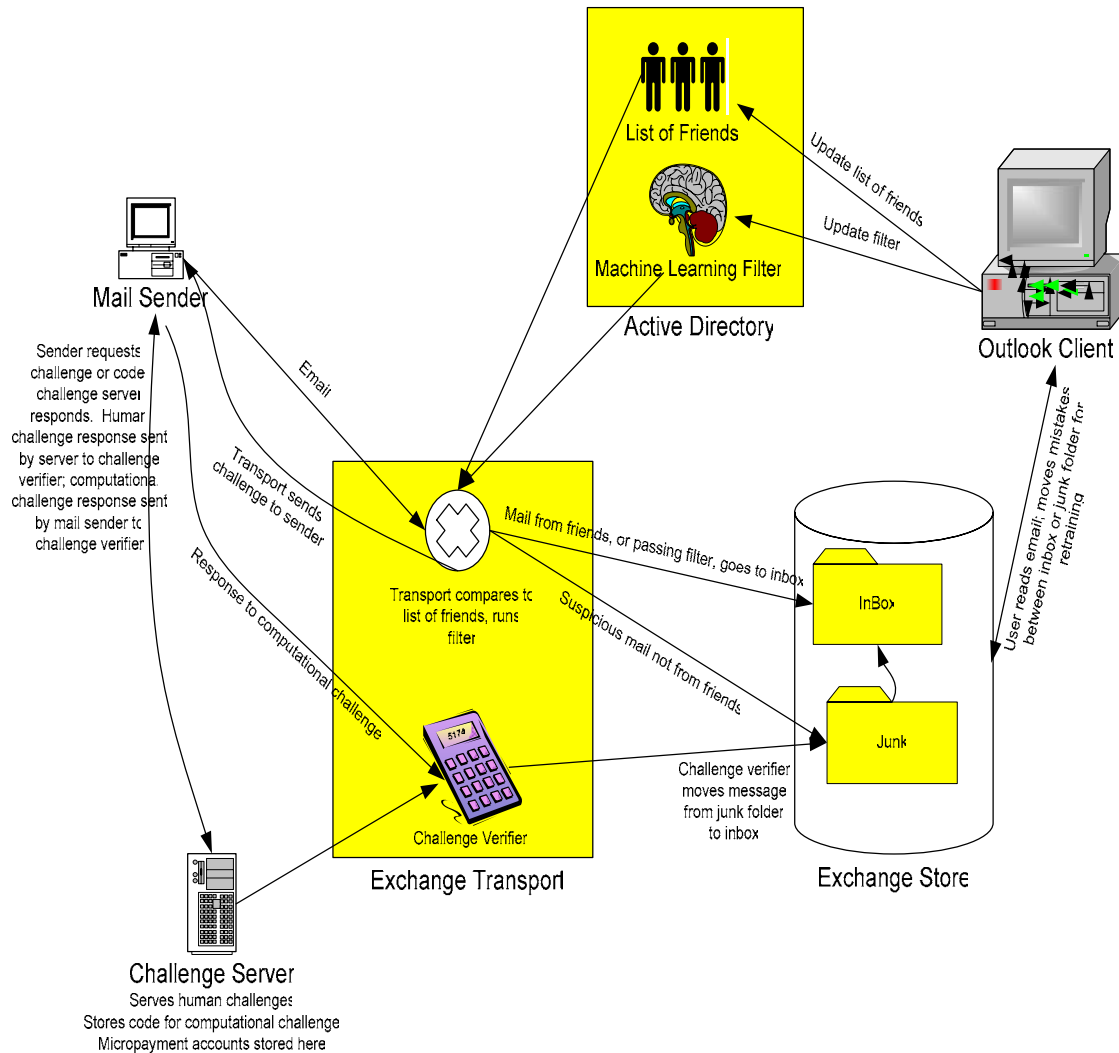


If the mail sender chooses to solve a human challenge, he brings up a web page on the challenge server. When he correctly responds to the challenge, the challenge server sends the response to the challenge verifier. For micro-payments, the challenge receiver sends account information and a signed permission to decrement one credit from that account to the challenge server, which either decrements the account, or places a one-credit hold on the account, and sends a signed permission to the challenge verifier. Note that messages from the challenge server to the challenge verifier or from the mail sender to the challenge verifier are simply delivered as normal email with special text in the headers or content; this text is detected and causes the mail to be recognized as a challenge response.

Server-centric architecture

In this server-centric architecture we assume that each user has a list of friends.. This list could be stored in the Active Directory, or perhaps in the Exchange Store, or elsewhere. When a message arrives at an Exchange Transport, the transport requests the list of friends. If the sender is on the friends list, the message is simply sent to the Inbox folder of the store. The message is then scored by the filter. If it passes, it is sent to the inbox. If not, the transport puts the message in the junk folder, and sends a challenge to the mail sender. The mail sender responds, by solving a computational challenge, a Turing test, or making a micro-payment. In one of these ways, a response is sent to the transport, and headers identify it as such a response, routing it to the challenge verifier, which causes the message to be moved from the junk folder to the inbox.

Clearly, the server-centric and client-centric architectures represent two ends of a continuum – a combined architecture is quite reasonable as well.



The server-centric architecture has a number of advantages compared to the client-only architecture. The first big advantage is that it potentially reduces the cost of spam to the recipient; storing and internally transporting spam, even if messages are never seen by actual users, can be expensive. A server-centric approach allows the spammiest messages to be immediately deleted, before they burden the client, and with substantially less burden on the server. This advantage is especially large for users with low speed connections. Another advantage is that challenges will happen much faster, especially if a user only occasionally runs their client. Also, having the code on the server will work better for users who access their account from a variety of clients, including web browsers and handhelds.

Note that this server-centric solution works best when there is substantial help from the client interface. For instance, it is still very useful to have a client-side interface for creating safe lists. If the filter is adaptive on a per-user basis, then it is also useful to have a “Delete as Junk” button on the client. Thus, even a filter running on the server benefits from coordinating with the client.

Conclusion

SmartScreen technology makes sure that most spam is caught, but may rarely catch legitimate messages. Given the high cost to users of missing good mail, it's important to make sure that every good message is delivered. Other documents (CSRI) describe a solution for large legitimate senders who agree to abide by standards of good email practices. This document describes the SmartProof system which ensures delivery of mail from individuals, small businesses, and others who for some reason do not use the CSRI approach.

Individual components of SmartProof have been described elsewhere by others. The unique aspect of this system is the combination of the pieces. By using machine learning, we minimize the number of irritating challenges. By offering users a choice of proof methods – computation, Turing-tests, or perhaps money, we make sure that every user has some means to solve the challenge. And by using computational challenges, which can be responded to automatically, we further ensure that the number of annoying challenges that actual people see is as small as possible. SmartProof also means that the SmartScreen filters can be used much more aggressively, making it nearly impossible for spammers to get their messages through. SmartProof, in combination with other technologies including CSRI and SmartScreen, can mean an end to spam as we know it.

Thanks to Elissa Murphy, Nina Kang, Ryan Hamlin, Manav Mishra, John Doerbrouk, John Deurbrouck, Derek Hazeur, Gopalakrishnan Seshadrinathan, Anthony Penta, Ryan Colvin, Dean Slawson, Dennis Adler, Chris Meek, David Heckerman, Eric Horvitz, Josh Benaloh, Arnold de Leon, Cynthia Dwork, Ted Wobber, Andrew Birrell, Dinei Florencio, Mike Burrows, Rick Holzli, Bryan Starbuck, Patrice Simard, Pablo Stern, Micah Rupersburg, John Mehr, Nathan Howell, Tim Dillon, Vivek Sharma, Dave Walsh, Andrew Birrell, Mike Burrow, Dan Crevier, DeAnne Dodson, David Cortright, John Platt, and many others for discussions on this topic. Joshua Goodman and Robert Rounthwaite did most of the analysis in this document, but the actual planning, testing, and implementation of SmartScreen and any future implementation of SmartProof were done by a huge team of people, including many of the aforementioned.

Rude Q&A

How will this be deployed? What about legacy or incompatible clients?

The SmartProof concept is very concerned with deployment. We've designed it in such a way that even if no one else in the world has SmartProof, it still works fine. Legitimate senders will only be challenged rarely: if they send spam-like mail to people who haven't safe-listed them. When people with legacy clients are challenged, they can answer human challenges, or make micro-payments, or manually download and run the code for solving a computational challenge. Of course, the SmartProof works better the more people who deploy it – those people can answer challenges automatically.

Why can't we just use the machine-learning filter by itself? If we expect the machine-learning filter to fail eventually, why ship it now? If the filter will not fail over time, why ship anything else?

Our current filter's default threshold is relatively conservative, so that all legitimate mail gets through, as do almost all "subscriptions" – legitimate mass mailings that users don't care much about. With such a conservative threshold, we can't block all spam. With the challenge-based approach, you can set the threshold more aggressively, because you know that if you ever block mail from a friend, they can respond to your challenge. That way, you can block essentially all spam. Also, there is some very small chance that with the threshold we've set, some legitimate message might not get through. For instance, one of the examples we know of was an article about spam with samples of spam in it. The challenge approach guarantees that every legitimate message from a person can get through: many users want that guarantee. As for whether the filter will fail, we hope not. We've set it up with hundreds of thousands of features, making it relatively hard for spammers to adapt to it. But spam is a war, and spammers will do what they can.

What about users with disabilities? They cannot solve the visual Turing-tests.

Users with disabilities can solve the computational challenges, or make the micro-payments, or, for the visually impaired, solve an audio challenge.

Won't SmartProof annoy my friends? How will my grandfather get through?

No. Your friends will only be annoyed if they send you spam-like mail, and only if you haven't safe-listed them yet and only if they are running legacy software that does not automatically answer challenges. If your grandfather is sending you spam-like mail, and cannot figure out how to answer the challenges, add him to your safe list.

How will companies like Amazon.com get through to their customers?

Companies have two options. One option is to sign up for one of our other initiatives, like CSRI. If they are unable or unwilling to do that, they still have many options using

our challenge-response system. If they send mail like receipts, they should have no problem, because the filter will not mark those as spam. If they want to send advertising to their customers, they will need to make sure their customers safe-list them. They can either email a request to be safe-listed (which will not look like spam), or they can include this request at purchase time. Alternatively, they can answer computational or monetary challenges.