Decentralized Proof-of-Effort Bonds for Spam Control

Edward N. Taylor
Department of Computer Science
St. Cloud State University
St. Cloud, Minnesota 56303
taed0701@stcloudstate.edu

Jayantha Herath
Department of Computer Science
St. Cloud State University
St. Cloud, Minnesota 56303
jherath@stcloudstate.edu

Abstract

At the heart of the spam problem is that its cost is wildly asymmetrical: negligible for senders, but, collectively, huge for recipients. Proof-of-effort systems address this disparity by requiring that senders demonstrate that a significant amount of work has gone into the sending of a particular message. Sender-at-risk systems require that senders post bonds warranting that the messages in question will not be considered unwelcome by recipients. Sender-at-risk systems usually require the involvement of a central authority. We demonstrate how melding features of sender-at-risk and proof-of-effort can be used to build a sender-at-risk system that minimizes or entirely eliminates the involvement of a central agency.

1 Introduction

Postal junk mail is a nuisance, but because it is expensive to send, does not threaten to bring the postal service to its knees. Advertisers consequently target recipients selectively. Spam, on the other hand, has been compared to junk mail printed on free paper sent postage due with no option to refuse.

Taking a cue from this observation, systems have been proposed and developed that attempt to shift the cost of email from the recipient to the sender. Naive proposals in which senders are taxed a few cents for every message have, for good reason, proved unacceptable.

Proof-of-effort, which we discuss in Section 2, is a system in which the sender of an email must demonstrate to the recipient that a certain quantum of effort has been expended in the sending of that particular message.

Sender-at-risk, discussed in Section 3, is a system in which the sender "bonds" her email with some item of value that will be forfeited if the recipient deems the email unwelcome. These systems depend on a central authority to oversee the bonding process.

We consider the involvement of a central authority to be undesirable for privacy and other reasons. In Section 4 we describe two types of proof-of-effort bonds that minimize or eliminate entirely the need for a central authority.

2 Proof-of-effort stamps

Proof-of-effort (POE) systems require email senders to prove to the intended recipient that a certain quantum of effort (machine or human) has gone into the sending of a particular message. The idea is that this will be easy for the legitimate sender, but impossible for the spammer, who is required to provide a separate POE for each and every of his millions of messages.

The best known such system is HashCash, proposed and developed by Adam Back in 1997 [1]. HashCash requires that the sender (actually, the sender's computer) calculate a "stamp", requiring a significant amount of CPU time, to be included in the email as an X-header. The work is all on the sender's side -- the recipient can trivially verify that the stamp is valid and that the required work has actually been performed.

An example of a HashCash stamp is:

1:20:090314:entaylor@drdmail.com::LB6Oq2updYu8pMrQ:0000005lCd

The colon-delimited components are

1) The HashCash version number.
2) The 'size' *n* of the stamp in bits (see below).
3) The date the stamp was created.
4) The resource (recipient's email address in this case) for which the stamp was created.
5) A reserved field

6) A large (base 64) random number, effectively a nonce. The resource plus the nonce uniquely identify this stamp.
7) A value calculated by the sender (see below).

The work is done in step seven. The sender must try different values until one is found that results in the SHA-1 hash of the entire stamp having the first *n* bits zero. This will take on average $2^{(n-1)}$ attempts. All the recipient has to do is to hash the stamp once to verify that its first *n* bits are zero. A 20-bit stamp takes an average of about 340 milliseconds on a midrange notebook machine. A 28-bit stamp takes about 87 seconds and a 36-bit stamp takes about 6 hours.

Laurie and Clayton [2], analyzing HashCash, argue that the amount of work required to discourage spammers places an unreasonable burden on legitimate senders. Using reasonable assumptions, and taking into account that spammers will harness botnets to aid in the calculation of POEs, they estimate that a 346 second stamp (~30 bits) would be required to reduce spam to 1% of all delivered email. They estimate that this would prevent activity by as many as 13% of legitimate email senders. A fifty second stamp would inconvenience up to 1% of legitimate senders. In this paper we will consider as viable a system that costs spammers at least 346 seconds, and legitimate senders at most 50 seconds, per email.

Implicit in Laurie and Clayton's calculations is the assumption that the same per-message cost will be borne by the legitimate sender as by the spammer. This assumption need not hold. Laurie and Clayton do not consider the effects of whitelisting, although whitelists are an integral part of some HashCash implementations [3]. It seems reasonable to estimate that at least 80% of legitimate email is exchanged between parties who have previously communicated. If these parties whitelist each other their communications need not require a POE. If we use a 346 second POE, this brings the cost per email to the legitimate sender down to 70 seconds, while the cost to the spammer remains at 346 seconds. Whitelisting alone could be sufficient to reduce the work of legitimate senders to a viable level. The relative amount of work required of legitimate senders can be reduced further, as we shall see below.

3 Sender-at-Risk

In a sender-at-risk system, the sender of an email “bonds” the email with an item of value that is subject to forfeit if the recipient deems the email abusive. In discussions of sender-at-risk the item of value is usually assumed to be a small amount of money.

3.1 The Attention Bond Mechanism

Loder, Alstyne, and Wash [4] have proposed a system in which senders post cash bonds on their outgoing messages, guaranteeing that the recipients will not find the messages abusive (this is known as the “Attention Bond Mechanism” or “ABM”). Participants maintain cash accounts which are used to fund these “attention bonds.” When a bonded message is sent, money moves from the sender's account into an escrow account, and can be claimed by the recipient if she finds the message unwelcome. Otherwise the bond can be explicitly refused by the recipient or allowed to expire, in which case the money returns from escrow to the sender's account.

Implementation of such a system presents many difficulties. If the ABM is to be adopted by thousands of ISPs, as it must be if it is to become “the” solution to the spam problem, clearly a centralized, trusted

micropayment infrastructure will be required. Although there are a number of companies offering micropayment services, it ts doubtful that any are prepared to handle the volume of transactions that would develop. Furthermore, such an entity would need to accept extraordinary risks. A current favorite spamming technique is to register a throwaway domain using a fraudulent credit card number, send a few hundred thousand messages, and abandon the account. Similarly, it is to be expected that spammers will make fraudulent deposits into their bond accounts, send as many spam messages as they can, and abandon the accounts. Most of the recipients will claim the bond, and to maintain credibility the agency will need to pay. Later, when the transaction is disallowed as fraudulent, the agency will be out real money. While credit card companies often protect merchants against fraudulent transactions, this protection applies only when tangible goods (which attention bonds are not) change hands.

There is also a serious privacy concern – the central agency would have knowledge of the existence, if not of the content, of every email sent using the system.

Another problem with the ABM as stated is the "adoption problem" – that it will be a terrible bother until it is universally adopted. Consider the cost to the sender, whose ISP has not adopted the ABM, of sending a message to a recipient whose ISP has adopted the ABM. How will this be handled? Will the recipient be required to create an escrow account on the spot, fill in forms, and supply his credit card number? The sender may well decide that his message isn't worth the effort.

To our knowledge, no cash-based ABM system is currently in use. Vanquish Labs (http://www.vanquish.com) has a patent on something very like the ABM, and has released at least two versions over the past few years. However, Vanquish, although still in the anti-spam business, appears to have abandoned its ABM products, and their website no longer mentions them.

3.2 Attention bonds using proofs-of-effort

Most of the objections to the Attention Bond Mechanism revolve around the problems inherent in implementing the payment system. If to fund bonds we use POEs rather than money, most of these objections can be made to disappear.

We need a reusable proof-of-effort. We must allow POE reuse while detecting and preventing double spending of the POE.

3.2.1 Microsoft's "Penny Black": a centralized system

The "Penny Black" project is a sender-at-risk project under development at Microsoft [5]. Microsoft does not specify the item of value backing the bond; various forms of proofs-of-work are mentioned as possibilities. Email is sent by obtaining a "ticket" from a central ticket server, and sending the ticket and the email together to the recipient, who verifies the ticket. The recipient can tell the ticket server to refund the ticket to the sender if she does not consider the message inappropriate.

Again, the involvement of a central agency having vast knowledge of private email traffic raises serious concerns. Clearly a decentralized system would be preferred if one could be devised.

4 Decentralized solutions using proof-of-effort bonds

In the following discussion, when we say “ISP” we mean an internet service provider that participates in the proof-of-effort system. By “rogue ISP” we mean a participating ISP that actively attempts to subvert the system.

If we wish to avoid the use of a centralized “ticket server” we have a problem: ISPs must be able to trust one another's POE bonds. We must not allow rogue ISPs to forge unlimited numbers of POE bonds to send spam.

We propose two solutions to this problem: one using CAPTCHAs as bonds, and the other using a modification of HashCash stamps.

4.1 CAPTCHAs as attention bonds

“CAPTCHA” is an acronym for “Completely Automatic Public Turing test for telling Computers and Humans Apart”. A CAPTCHA is usually given in the form of a sequence of distorted alphanumeric characters easily understood by a person but unrecognizable by a computer. Audio CAPTCHAs have been used to accommodate visually impaired users.

Figure 1: A CAPTCHA

CAPTCHAs have utility (they can be used to send email) and scarcity (they cannot be produced in unlimited quantity by a mere human) and can be used to fund attention bonds. Since we do not wish to involve a central agency in email transactions we must find a way to prevent rogue ISPs from forging solved CAPTCHAs. We do this by having a trusted authority create unsolved CAPTCHAs and supply them in bulk to ISPs.

Our CAPTCHAs have the following format:

1) domain for which the CAPTCHA was created (e.g. drdmail.com)
2) serial number
3) date of creation (e.g. 090315)
4) alphanumeric sequence represented by the CAPTCHA image (the CAPTCHA sequence)
5) encrypted data including the CAPTCHA sequence
6) the image file containing the visual representation of the CAPTCHA

The domain and serial number uniquely identify a CAPTCHA. Item 5, the encrypted CAPTCHA sequence, is generated by the following steps:

1) items 1-4 are concatenated
2) the result from step 1 is encrypted with the trusted authority's private key
3) the result from step 2 is XORed with the SHA-1 hash of the result from step 1

Unsolved CAPTCHAs are produced in bulk by a trusted authority and distributed to ISPs. These unsolved CAPTCHAs do not include item 4, the plaintext CAPTCHA sequence.

A solved CAPTCHA bond includes items 1-5 (item 4 having been supplied by the solver); once the CAPTCHA has been solved, the image is no longer required and is discarded.

Anyone in possession of the plaintext CAPTCHA sequence can recreate step 1, hash the result, XOR with the encrypted CAPTCHA solution (item 5), and decrypt the result using the trusted authority's public key. If the decryption matches the plaintext items 1-4, the CAPTCHA has been correctly solved. A rogue ISP has no way to produce a valid bond without obtaining the plaintext CAPTCHA sequence, which it cannot do other than by having someone solve the CAPTCHA manually.

Our CAPTCHA sequence consists of eight lowercase alphanumeric characters omitting the numerals '0' and '1' and the letters 'o' and 'l'. This leaves 32 characters, fitting nicely into five bits. The entire sequence space is 40 bits. A brute force attack would involve an average of 2^{39} attempts, each of which would involve an expensive RSA decryption, making a brute force attack is impractical.

This approach still requires the involvement of a trusted authority, but is less unacceptable than the Penny Black approach. It is not necessary to limit the trusted authority to a single agency; just as browsers can be configured to accept X.509 certificates from arbitrary authorities (Verisign, Thawte, etc.) email clients could be configured to accept CAPTCHA bonds produced by any number of authorities. The involvement of the trusted authority is limited to the production of unsolved CAPTCHAs; it is not involved in the exchange of any particular message and has no access to private data.

4.2 HashCash stamps as attention bonds

It is possible to devise a proof-of-effort attention bond that entirely eliminates the need for a trusted authority by making a few simple modifications to the HashCash proof-of-effort system.

HashCash stamps have not been considered reusable but could easily be made so. An example of a reusable HashCash stamp would be:

1:31:090314:entaylor@drdmail.com::r8ybU4m3Cy4H5o5y:00D+GaFcY

the format of which is just the same as before, although the meaning of some fields has changed. The bit strength of the stamp has been increased to 31 bits, which represents about 700 seconds on an average notebook. We will specify arbitrarily that this stamp will expire one year after its creation date, allowing for repeated reuse. The email address on which the stamp is calculated is the sender's rather

than the recipient's address, reflecting the fact that the stamp is owned by the sender and can be reused to send to multiple recipients.

An ISP receiving an email bonded by such a stamp will, after validating the hash, check in a local database that the stamp, identified by its sender address and nonce, is not currently in use and has not been revoked by a prior recipient. If the stamp is valid, the email will be delivered to the recipient's inbox.

4.3 Issues common to both HashCash and CAPTCHA bonds

If the recipient of the message finds it to be abusive or unwelcome, she can penalize the sender by revoking her POE. If the message is not deemed abusive the recipient can either explicitly refuse the POE or simply allow it to expire as it will in (again arbitrarily) two business days. If the sender is not notified of the revocation of her POE within this period, she can use that same POE to bond another message. (This would be a good time for the sender and recipient to whitelist each other.) POEs revoked by a recipient do not become the recipient's property; such would lead to the use of third party POEs that would complicate the identification of rogues.

The sender's ISP keeps track of the bonds owned by each of its users. The sender must create enough POEs to cover all messages she will send over a two-business-day period; this is the interval within which POEs can be revoked by recipients. When the two days have passed without notice of revocation, the sender may reuse the POE. The sender's ISP will notify the sender if she does not have a required POE available, and will give her the opportunity to produce one.

Each participating ISP will need to maintain a database to track revoked and in-use bonds. So that this database will not grow without limit, all POEs will expire in (arbitrarily) one year from the date of creation. Expiring POEs in this manner also allows for the certainty that hardware will continue to improve, and that larger bit strengths (in the case of HashCash bonds) or modifications of CAPTCHAs will be required as time passes.

The “adoption problem” is much less serious with POE bonds that with cash-based attention bonds. The sender of an email from outside the system can be directed to a web page where she can complete a CAPTCHA or where a Java applet can compute a HashCash stamp. Once the POE has been completed, it will reside on the recipient's ISP and will be transparently reused (unless revoked) whenever that same sender sends a message to any user on the recipient ISP. This is still something of a bother, but at least is much more tractable than was the case with cash bonds.

5 Attacks on decentralized POE bonding

5.1 Cross-domain misuse of POEs

To this point we have ensured that a POE that has been revoked on a recipient ISP cannot be reused on additional messages to users on that same ISP. However, a rogue ISP could use the same POE to send a single spam to a single recipient on each of every other ISP.

To prevent this type of abuse, in the case of HashCash bonds, each participating ISP is required to calculate a particularly gigantic (40 bits, about 4 days) domain-level stamp. This domain-level stamp

discourages domain-hopping by spammers. Each HashCash POE will now contain two stamps. An example might be:

1:31:090314:entaylor@drdmail.com::r8ybU4m3Cy4H5o5y:000D+G3FcY
1:40:090101:drdmail.com::2sXrkvp8adzf8sYw:0g6BPI8rc9

the first of which is the user-level stamp, and the second of which, the domain-level stamp, guarantees the good behavior of drdmail.com, and will be revoked if drdmail.com is found to be violating the rules.

In the case of CAPTCHA bonds, we consider the work involved in obtaining a batch of uncompleted CAPTCHAs from a trusted authority to be its domain-level "bond". We note that spamming by an ISP will reflect badly on the trusted authority that issued its CAPTCHAs. Therefore authorities will be careful to establish the bona fides of an ISP before issuing CAPTCHAs.

To detect multiple use of bonds, ISPs should share information on revoked bonds over a peer-to-peer network. No single POE should ever be found to have been revoked on more than one ISP. Such an event would indicate misconduct by the originating ISP.

Participants should notify each other on the occurrence of events that suggest spam activity with high probability. One example of such an event would be the receipt of a message from a previously unknown domain, the bond on which is revoked by the recipient. The receiving ISP should promptly communicate this fact to its neighbors. Should that bond be found to be in use or to have been revoked on one of the neighbors, the domain-level stamp of the offending ISP should be revoked, and this fact should be communicated over the network to all participating ISPs.

ISPs should also communicate information on revoked bonds at regular intervals. Notification of all participants of all revocations is not required; notification of a subset of participants of a subset of revocations should eventually lead to identifications of miscreants with high probability.

In the case of HashCash POEs, the size of the user-level stamp should be set high enough that revocations are rare, to minimize administrative message traffic between participating ISPs. A 700 second (31 bit) stamp, twice as large as the 346 second stamp that Laurie and Clayton calculate would bring spam down to 1% of delivered email, should be sufficient. If we take into account that 80% of email is sent between parties who will have whitelisted each other, and that 90% of the stamps on legitimate email will not be revoked, a 700 second stamp will cost the legitimate sender an easily manageable 14 seconds per email.

Let us consider a worst case situation in which a rogue ISP calculates 1000 user-level stamps and one domain-level stamp, then as quickly as possible sends 1000 spam messages to each of 1000 ISPs. The cost to the spammer is the four days (338,000 seconds) for his domain-level stamp plus 1000 user-level stamps at 700 seconds each. The cost per spam is then just over a second – clearly viable for the spammer.

However, there will be a first user on one of the target ISPs who opens the first spam and revokes its POE. This will trigger notification of neighbor ISPs, who will find that POEs are being multiply applied, which will trigger a broadcast notification that a rogue ISP has been detected, and that its domain-level POE should be revoked. Each target ISP can then delete messages received from the

rogue, in most cases before their users have seen these messages in their inboxes. It seems likely that only a very few spam messages will actually survive to be read by their intended recipients, and that the cost to the rogue per delivered message will be very large.

5.2 Theft of proofs-of-effort

An attacker who can observe messages in transit could "steal" proofs-of-effort. Then, pretending to be the owner of the POE, the attacker could send spam messages in the owner's name. In the case of HashCash style POEs, theft of domain-level stamps would have a particularly deadly effect – leading to the improper revocation of a legitimate ISP's domain level stamp.

To squelch this attack, we must provide for sender authentication. The Sender Policy Framework [6] and DomainKeys Identified Mail [7] would work well in this regard. The latter is especially attractive as it provides for public key retrieval via DNS, which will allow ISPs to digitally sign their POE bonds.

6 Proof of concept

An email system implementing CAPTCHA attention bonds has been implemented and is available for review on http://www.drdmail.com. Note that this site implements several unusual features that are not within the scope of this paper.

7 Conclusions

Claims that proofs-of-effort are not viable as a spam control measure may be premature. Allowing for proof-of-effort reuse by incorporating them into a bonding system increases the effort component for the spammer while reducing it for the legitimate sender. This can be implemented in a way that does not require the involvement of a central authority in any actual exchange of messages. While some hurdles remain, we believe that our proposals represent a few steps along the road to practicality for the sender-at-risk and proof-of-effort families of spam control measures.

8 References

[1] Back, Adam, "Hashcash – A Denial of Service Counter-Measure (5 years on)", 2002, http://hashcash.org/papers

[2] Laurie, Ben and Clayton, Richard, "'Proof of Work' Proves Not to Work", http://www.cl.cam.ac.uk/~rnc1/proofwork.pdf

[3] "Hashcash FAQ", http://hashcash.org/faq

[4] Loder, Thede; Van Alstyne, Marshall; Wash, Rick, "An Economic Response to Unsolicited Communication", 2006, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.7827

[5] "Microsoft Research: Projects: Penny Black", http://research.microsoft.com/en-

us/projects/pennyblack/

[6] “Sender Policy Framework: Project Overview”, http://openspf.org

[7] “DomainKeys Identified Mail (DKIM)”, http://www.dkim.org