CAMRAM

A practical sender-pays antispam system

31-Aug-2003

http://www.camram.org
http://camram.sourceforge.net

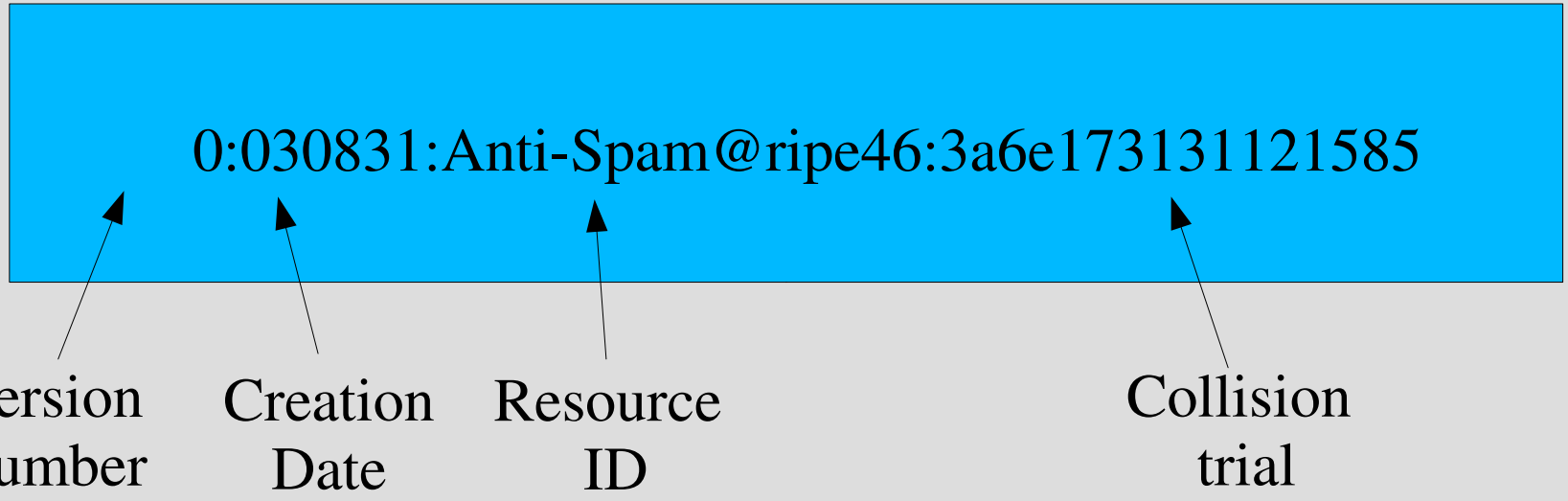
What is camram

- Practical example of sender-pays antispam system
- Grew out of hashcash
- Motivated by failures in other anti-spam technologies

Why use camram

- Preserves self introductions
- True sender pays
- Minimal receiver costs
- Low cost to deploy
  - Scales individual to enterprise
  - Incrementally deployable
  - Under local control
  - Minimizes misfiltered e-mail
  - Almost invisible to end-user
- Plays nice by itself

Stamp mechanics



http://hashcash.org/draft-hashcash.txt

Payment system differences

- Why hashcash works for payments...
  - Availability
  - Decentralized
  - Local minting
  - Ease of use
- ... and why money doesn't
  - No practical system
  - Infrastructure doesn't scale
  - Risk of theft by worm or e-mail virus
  - Additional local, state, national taxes
  - Regulation by International Postal System

Sender-pays mechanisms

- Embedded stamps
- Postage due notice (PDN)

Postage due

http://harvee.org/camram/pdr.cgi?alg=1.020&to=e.johansson@harvee.org&cookie=4a7f078fe1c5e1df

Just click on the above URL and your message will be released for
delivery to me. In the future, you can avoid getting these messages in one
of two ways: 1) wait for message recipient to send mailto you, 2) start using
the camram antispam system. See http://www.camram.org for more information.

Below is a fragment of your message. Please read it carefully to make
sure you are generating postage for your mail and your mail only.

From: esj@harvee.org
To : e.johansson@harvee.org
Subject: test

testing message

Differences from challenge-response systems

- Ability to set your own postage rates
- Ability to handle rate difference PDN's automatically
- Handicapped friendly (no problems for blind, mobility, or cognitive handicaps)
- No dependence on central infrastructure
- No deadlock conditions

Getting to a sender-pays world

- Satisfies criteria for successful change
- Provides benefits to early adopters
- Increasing benefits as the number of users increases

“But what about mailing lists” and other strawmen

- Mailing lists
- Spammers' increasing horsepower
- Hardware acceleration
- Legitimate commercial mass mail

Project status

- 0.1 Released last spring
- 0.2 Almost complete
- 0.3 First client-side release November/December 03

Project goals

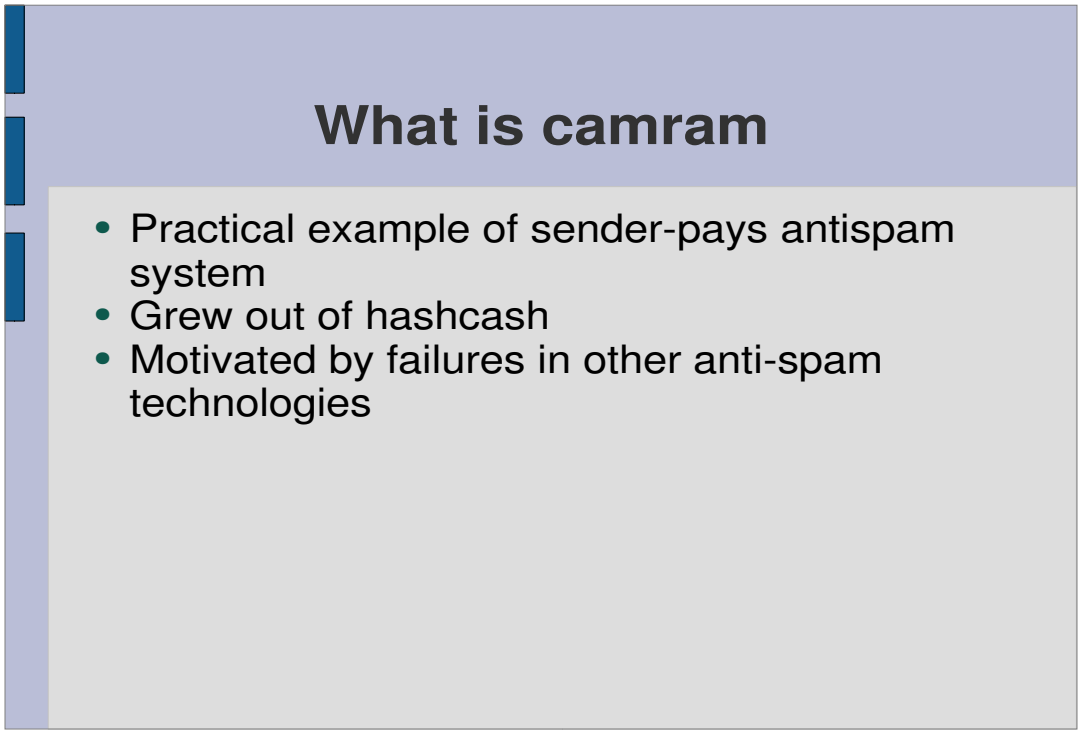
- White list by public key
- Opportunistic signatures
- Opportunistic encryption
- Multiple stamp definitions

CAMRAM

A practical sender-pays antispam system

31-Aug-2003

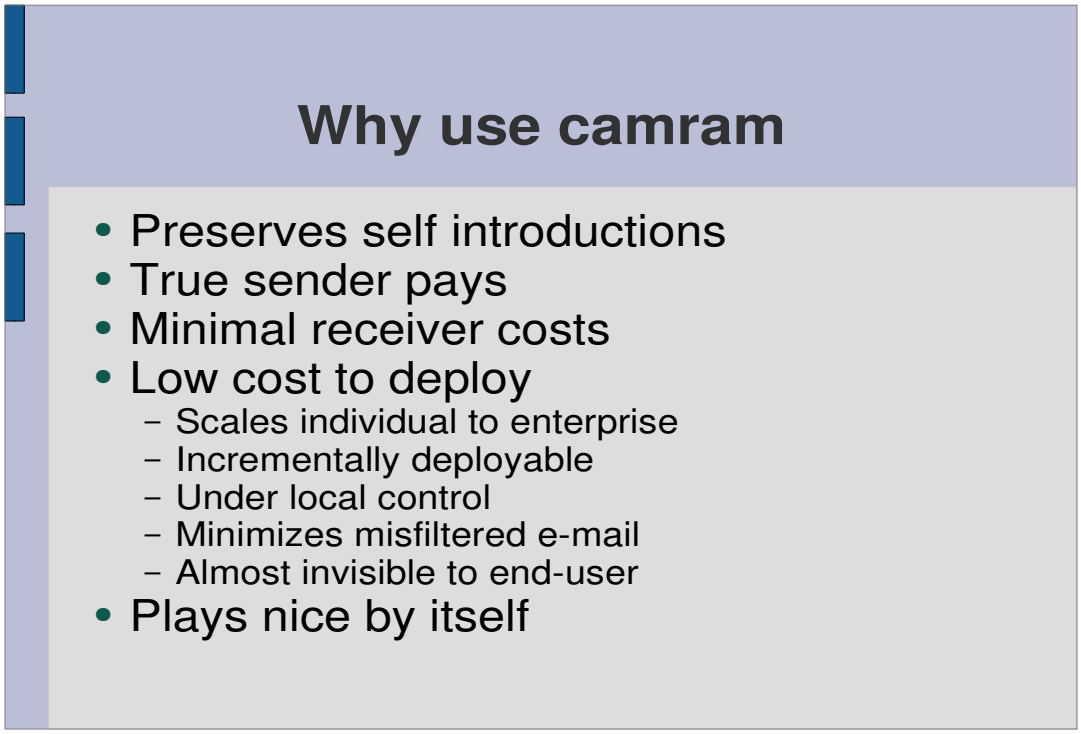
http://www.camram.org
http://camram.sourceforge.net



The Camram project grew out of Adam Back's work with proof-of-work postage mechanisms known as hashcash. Since there were no working examples and a lot of opinions both pro and con, a few people started conducting experiments with sender-pays systems and that grew into the Camram project. Today, we now have a practical sender-pays antispam system that is protecting a few sites.

Another motivator for the project was the significant error rates of antispam filters circa 2001. The migration to Bayesian filters has improved the error rates somewhat. But they are still too high, and most anti-spam systems based on Bayesian filtering have no recovery mechanism more elegant than manual scanning of the spamtrap. One of the goals of our experiments was to see if a sender-pays mechanism would permit reduction of the error rates and possibly provide an automatic mechanism for detecting spam.

Starting from a base of sender-pays mechanisms, Camram has incorporated additional antispam features to reduce the error rates below those of current antispam techniques and to provide a smooth transition between the epochs of no-stamp and all-stamp e-mail.

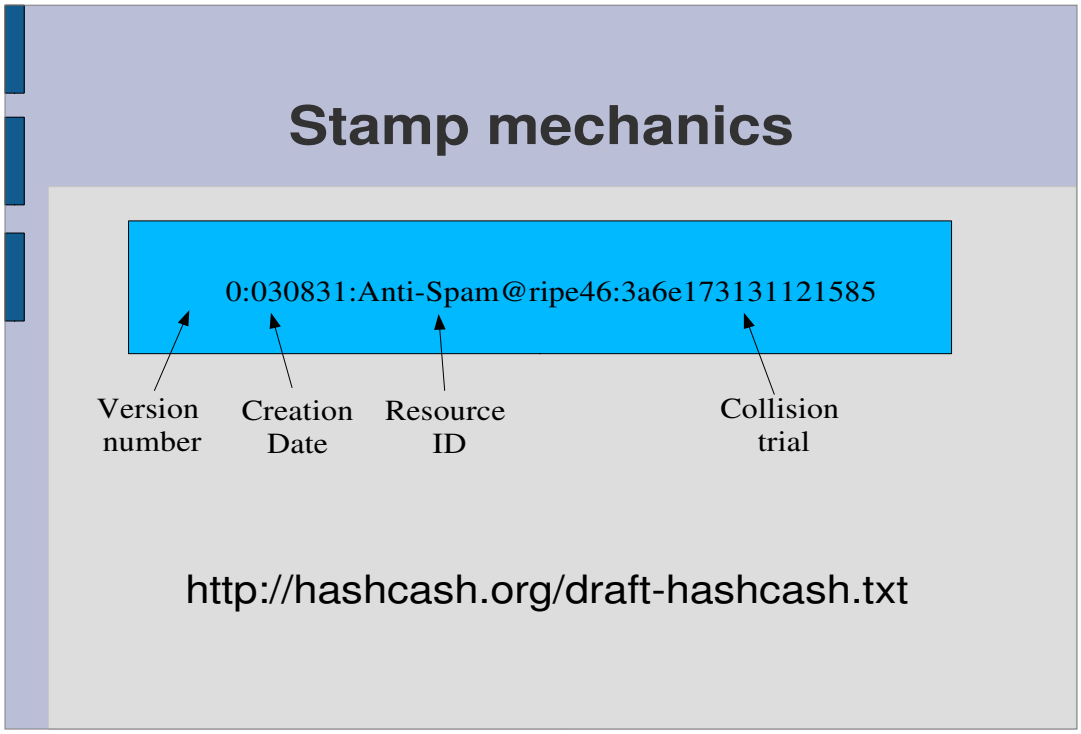


Why use Camram instead of staying with the status quo or going with some other solution? The initial reason was the somewhat abstract "it puts the cost of spam on the spammer." What we mean by this is that the spammer must spend all the cycles to generate stamps if they're going to spam. The theory is that if you make spam economically unviable, spammers will find some other line of work.

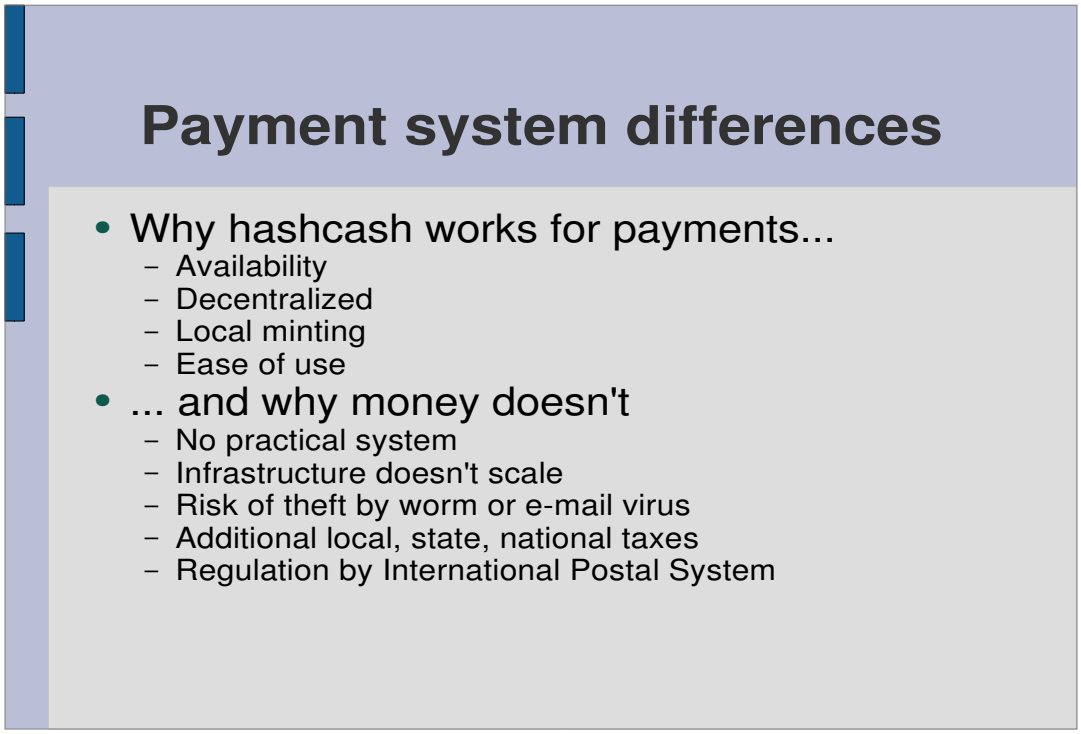
All other proposed spam solutions increase the cost of e-mail to the receiver in terms of reliability, speed, and/or bureaucratic overhead. Also in contrast to most of the other solutions, camram preserves many good features of today's e-mail while imposing a verifiable cost on the sender.

In a more practical sense, Camram is a lower-cost solution to deploy because of our attention to the considerations in the bullet points outlined on the screen.

Deployment growth will follow an epidemiological (viral) model, although Camram won't spread as fast as the latest viruses/worms. I think the model of rough consensus/running code will do more to move this concept along than anything else. Other steps to speed adoption would be the development of easy-to-use plug-ins for Outlook and Eudora, and enterprise-level filters for sendmail and postfix.



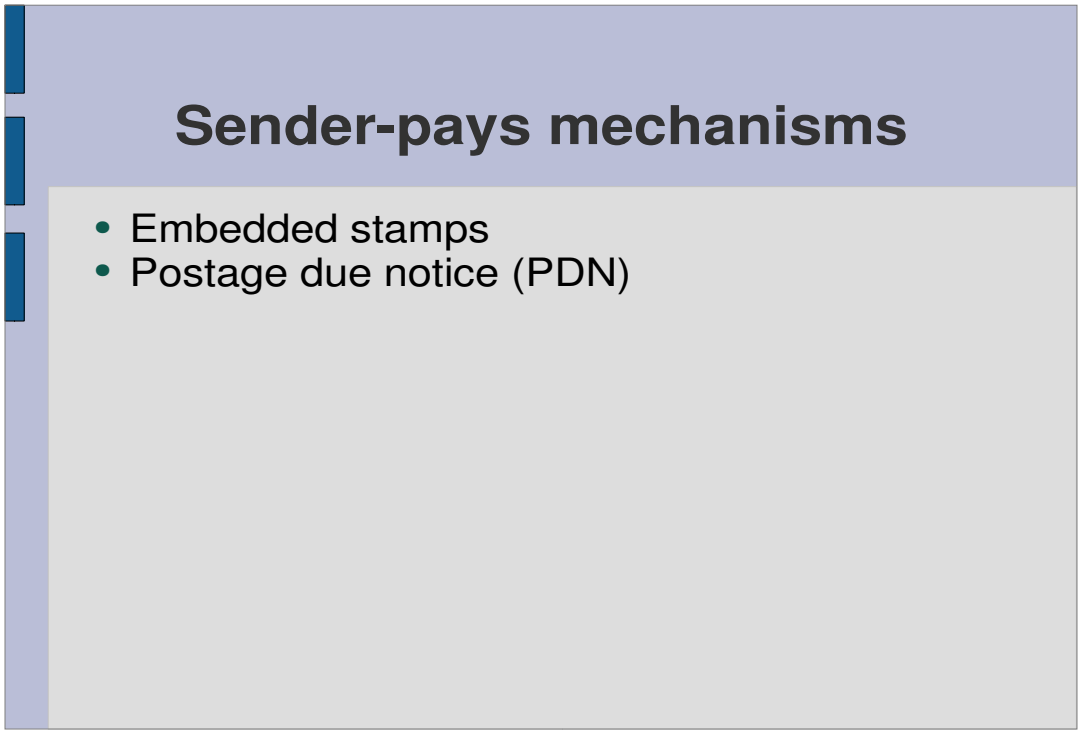
One of the things I find helps folks understand hashcash/Camram stamps is to see what one looks like. Stamps are based on N-bit hash collisions. The slide shows a 22-bit collision stamp. This stamp was a revision 0 stamp created on the 31st August 2003; the stamp recipient is the antispam working group at ripe 46; and the last field is the remaining data necessary to create a 22-bit collision. I won't go into the full details about stamp creation. Please see the draft RFC Adam Back has started for details about the stamp, its definition, and how it is calculated.



We should probably explain why hashcash was chosen as our unit of value. The primary reason is that it was available. There were no other digital units of value we could play with our experiments. It turns out this was a happy accident because the more you look into using digital currencies for stamps, the more significant problems appear. Specifically, double-spending checks become the major Achilles' heel and exploitable hole for spammers. There are also some additional risk of worm based theft of postage, additional taxes, co-option by incorporation into the international postal system.

In contrast hashcash stamps are self-minted and are bound specifically to a recipient. Double-spending checks are strictly the responsibility of the message recipient and therefore have no scaling problems. Since stamps have no monetary value, risk of theft, taxation, or regulation is low.

There is one worm/virus based risk: sobig.f or some equivalent beast could calculate stamps for a spammer. It would be theoretically possible to capture enough machines to satisfy spammer demands for postage. This kind of attack is unlikely to succeed for long -- machines would become unusable because of the computational and thermal load and from spam recipient's increasing their postage requirements.



There are two methods of paying for postage: either the stamp is embedded in the message, or the payment arrives after the message has been received but not delivered. Embedded stamps are the preferred form of postage. The sender generates a stamp and attaches at to the message before sending. This franking operation is performed without any user intervention and isusually invisible to the end-user except for (e.g.) 20 seconds of high CPU utilization.

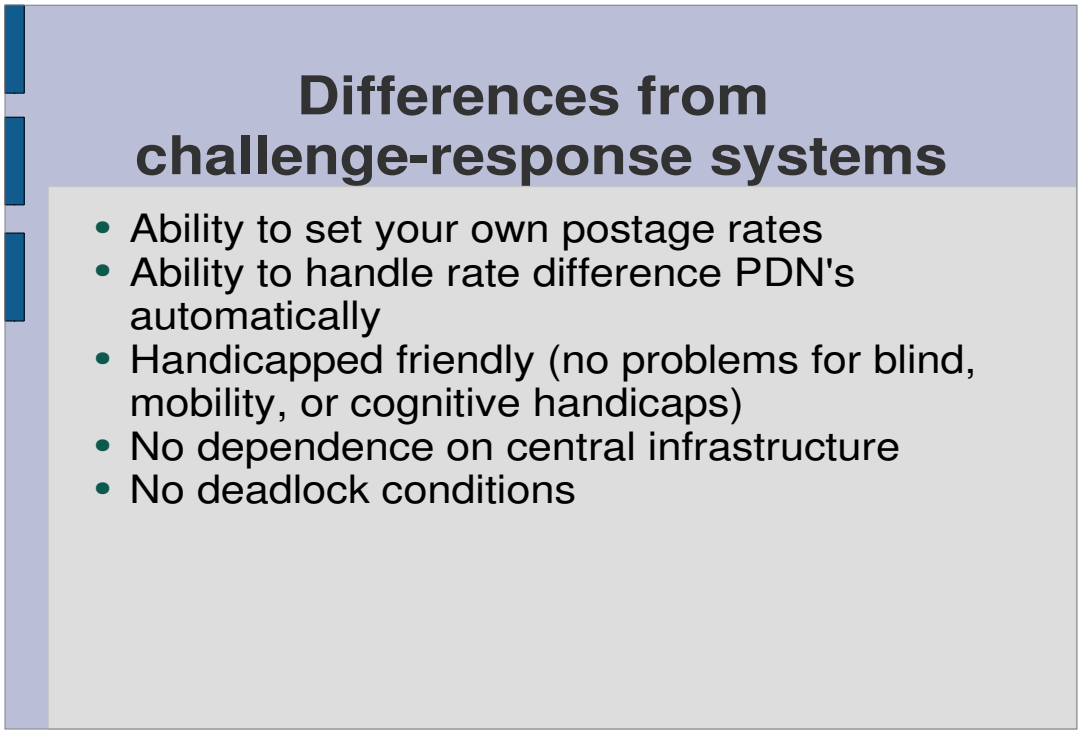
Embedded stamps are preferred because it's the most efficient mechanism for postage based e-mail. This approach generates no additional traffic; there is no work required of the recipient: the sender creates a message, stamps it, sends it, and it gets delivered.

Postage-due payments are less desirable than embedded stamps, but they are better than no stamps at all. The mechanism involves sending a postage-due notice (PDN) to the message originator. The message originator follows the instructions in the PDN and runs a browser-based stamp generator. After the stamp is delivered back to the message recipient, the message is released and delivered.

The postage-due notice is a way an e-mail sender can generate postage after they have sent their message. When they receive a PDN, the message originators sees a piece of e-mail containing an explanation, a URL, and a fragment of their original message. The fragment is there for context so that the recipient can confirm that the PDN is for something they sent and not some spoofed piece of mail from a spammer.

The explanation section gives reasons for the PDN and an explanation of what to do and what to expect. Believe it or not, we have found that people don't always read the explanation and react inappropriately.

The URL is simply the most convenient way of invoking a stamp generator and passing parameters to it. The user clicks on the URL, invokeing the browser-based stamp generator with given parameters. When it is done, the PDN delivers the stamp via cgi gateway to the end recipient.



Camram is not a challenge-response system. The PDN mechanism has elements of a challenge-response system in that a notification is given to the user and they can do something to release the trapped message. But the system was designed for embedded stamps and the postage-due notice is merely a failsafe mechanism.

Suppose a really determined spammer starts generating postage with their spam. Users can fight back by changing how much postage they charge for access.

With some additions to the postage-due notice definition, PDNs for the new rate can be satisfied automatically by individuals and still not provide any significant advantage to spammers.

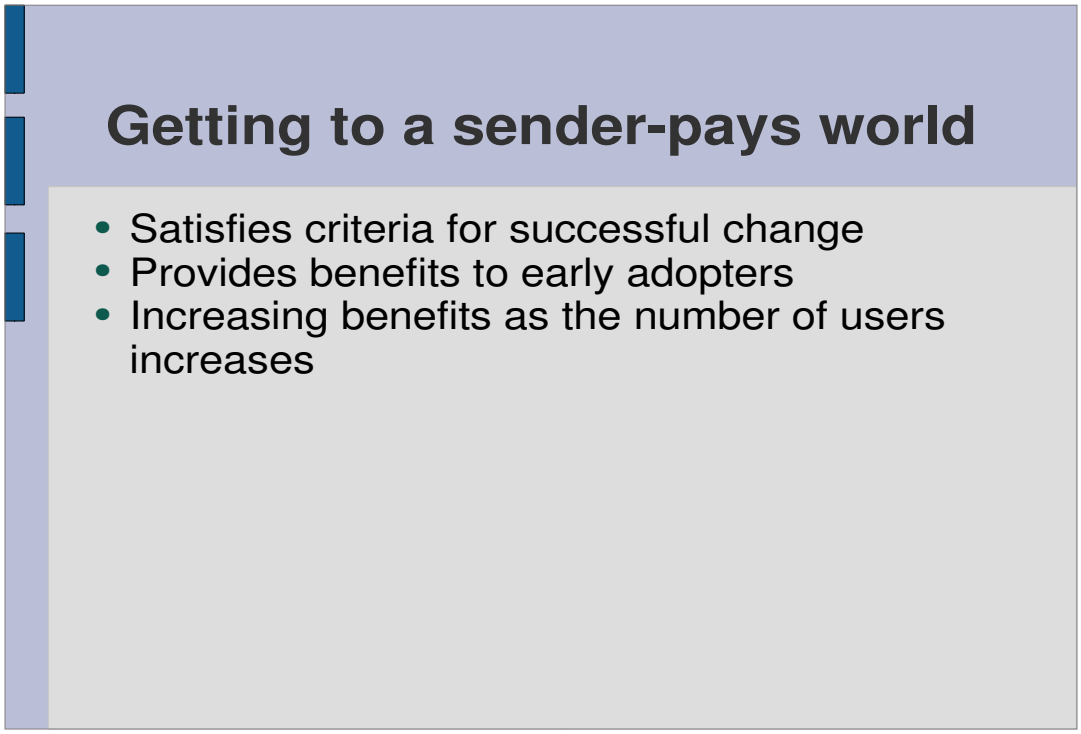
Unlike challenge-response systems using Turing tests based on images, or following complicated instructions, the Camram system does not provide any barriers that would impede use by blind-, mobility-, or cognitively impaired users. As a result, most sales and marketing departments should have no problems using the Camram solution.

Unlike many challenge-response systems, Camram has no dependence on centralized infrastructure. It can operate on individual user systems up through large-scale enterprise systems. The only dependency is for an external Web server capable of delivering the PDN stamp generator on demand. This can be satisfied by ordinary personal Web service since the process is only delivering static content.

Q: If two users of a challenge-response system e-mail each other, will they see the challenges?

A: Probably not.

In the Camram system, if two users generate postage-due notices to each other, each will see the postage due notice because each notice contains a stamp. There's never any chance of a deadlock.



To succeed, any antispam solution requiring a change in infrastructure needs to follow the rules of change management and be familiar, accessible, and reversible. Additionally, any antispam system needs to provide value to the first user and not wait for a critical mass of users to show return on investment. I believe Camram satisfies these criteria.

In the early stages of Camram adoption, the primary components supplying spam-filtering benefits are the white list and spam-discriminator filters. The white list improves accuracy because it retains knowledge of who the person or organization communicates with. The Bayesian filter accuracy is further improved through the postage-due system.

As the number of users increases, more and more stamps are used which further improves the accuracy of the overall system. The point is reached where only spam is passed to the Bayesian filter. At that point, one could migrate to a purely postage-based system and eliminate the Bayesian filter entirely.

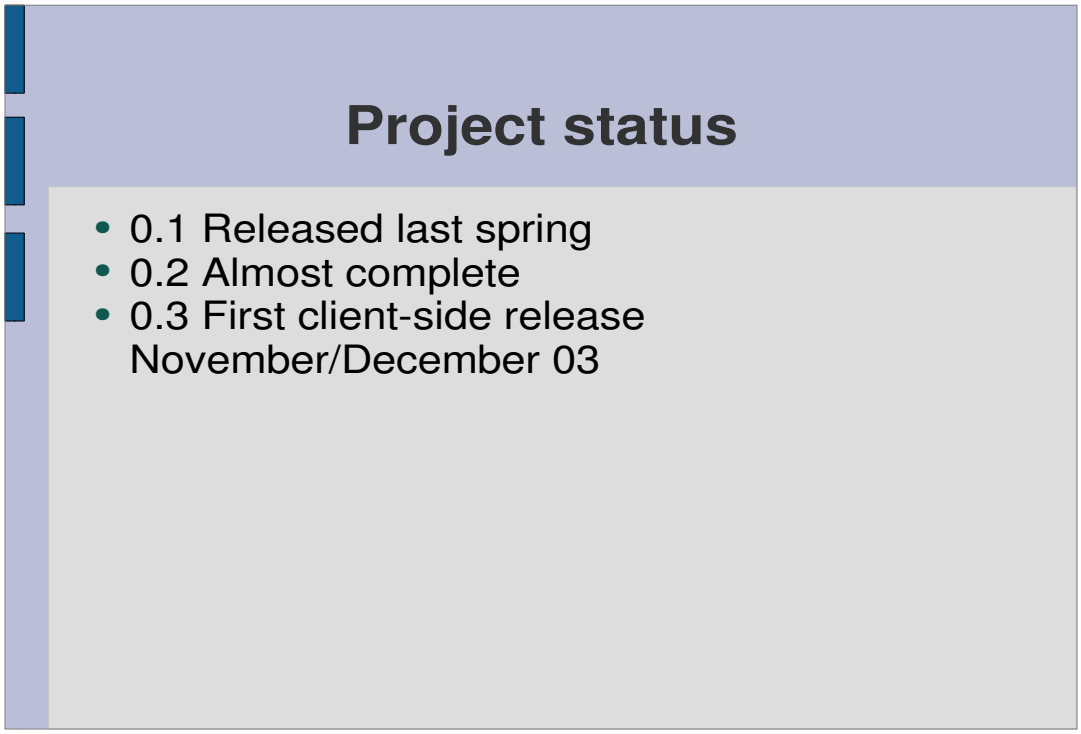
Invariably someone asks the question "but what about mailing lists, won't they have to generate stamps for everyone on the list?" In a word, no. In the current version of camram, mailing lists are not white-listed, but for the most part, messages on my mailing lists are not trapped by the discriminator. Part of the challenge is that white-listing mailing list addresses and searching all the addresses in a message combine to make a significant hole that spammers can exploit. In a bit we will discuss how white-listing by public key will make mailing list white-listing extremely foolproof.

Another frequent question is: how much horsepower will a spammer need to generate stamps? I ran the calculations assuming stamp generation required only three seconds per stamp. If a spammer is filling a T1 pipe with spam, s/he will need to run approximately 140 machines 24x7 to satisfy the demand. These won't be cheap machines either: they will need the very best in ventilation and cooling because generating stamps really heats up CPUs.

The primary way of dealing with spammer horsepower is to add a couple of bits to the postage value of a stamp. This will quadruple the load and the number of machines necessary to generate stamps in bulk. This question does point out the necessity of sizing the stamp correctly at the very beginning and developing a mechanism for semiautomatic increases in postage size. (The problem with automatic postage increases is that they can be used as a denial-of-service attack against postage systems.)

Hardware acceleration may indeed present a problem. The solution is to develop other proof-of-work stamps so that we can change the stamp modality if an attack should come from the hardware corner. On the other hand, one could look at this is an opportunity for hardware vendors. Make a stamp generating chip that one can fit into every PC or USB bus, raise postage rates accordingly and the hardware advantage goes away or is at least made very expensive.

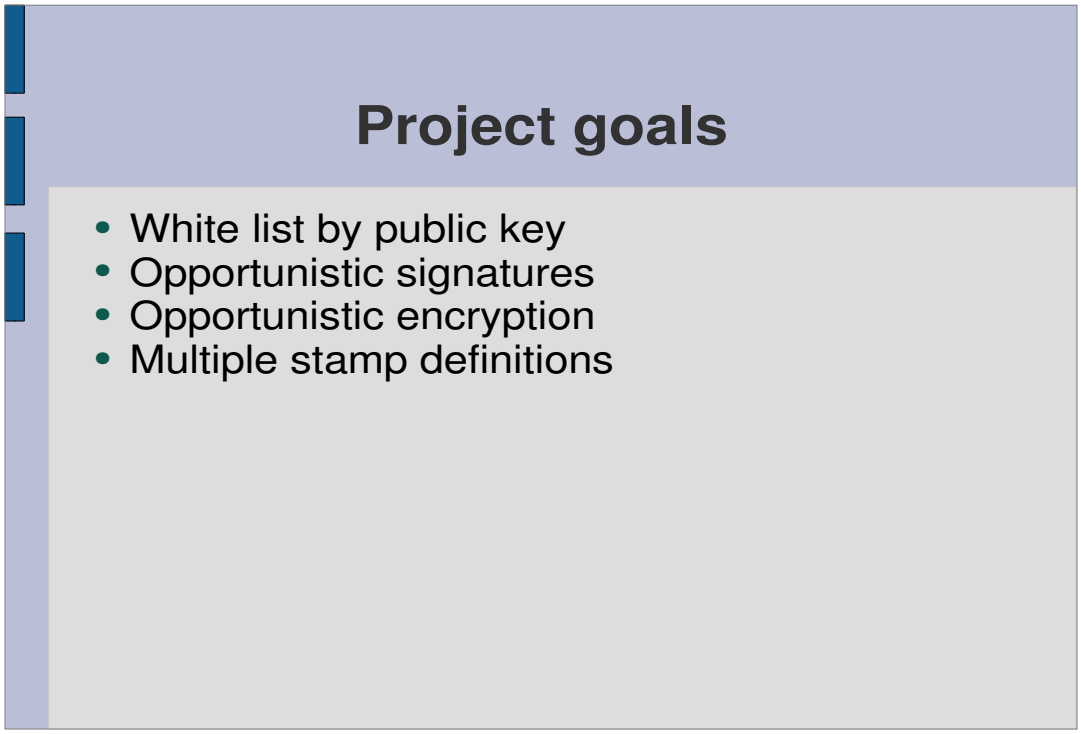
Legitimate commercial mass mail is almost an oxymoron. Granted, one has to deal with invoices and statements of orders and these can be stamped without causing undue hardship or added expense. Items such as invoices can even be covered by the "friends fly free" white-listing mechanism. However, this opens a door to never-ending catalogs, offers, etc. (I'm still getting offers from Gateway even though it has been at least eight years since I purchased anything from them.) At the end of the day, if a business has not established a legitimate relationship with the customer, then they should generate stamps. If they abuse the nature of the relationship, then they should be blacklisted and no amount of stamps will help.



This past spring I released the 0.1 version of camram. It was a typical rough and ready system. I installed it on three servers: harvee.org, eggo.org, and andrewandsons.com. Results in all three places are favorable. Very little legitimate mail is lost in the spamtrap and that which is caught can be easily found, white-listed, and released.

0.2 is almost complete. It will contain a working postage-due notification system, improved Bayesian filter, Web interfaces for user configuration editing, better support for individual versus global control of a mailbox, and background stamp generation. At the point of release, I will be seeking a number of guinea pigs to help shake out the system.

The 0.3 will feature primarily easier install/upgrade process and changes necessary to make Camram run on the client side.



Relatively early in the life of project, someone raised the idea of propagating keys with every mail message. It was then noted that it would be possible to sign messages instead of generating hashcash stamps. If you've got the keys, you can encrypt messages as well. A good idea but one that must come after the basic system works.

Once you have the ability to white-list by public key using opportunistic signatures on messages, you then have the ability to white-list mass broadcasts such as mailing lists without creating opportunities for spammers.

Look for it in release 0.3 or 0.4.