R. Roman, J. Zhou, and J. Lopez, "An Anti-spam Scheme Using Pre-challenges", *Computer Communications*, vol. 29, pp. 2739-2749, 2006.
http://doi.org/10.1016/j.comcom.2005.10.037
NICS Lab. Publications: https://www.nics.uma.es/publications

An anti-spam scheme using pre-challenges *

Rodrigo Roman[2], Jianying Zhou[1], and Javier Lopez[2]
[1]*Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613;*
[2]*E.T.S. Ingenieria Informatica, University of Malaga, 29071, Malaga, Spain*
roman@lcc.uma.es, jyzhou@i2r.a-star.edu.sg, jlm@lcc.uma.es

Abstract. Unsolicited Commercial Email (UCE), or Spam, is nowadays an increasingly serious problem to email users. A number of anti-spam schemes have been proposed in the literature and some of them have been deployed in email systems, but the problem has not been well addressed. One of those schemes is challenge-response, in which a challenge, ranging from a simple mathematical problem to a hard-AI problem, is imposed on an email sender in order to forbid machine-based spam reaching receivers' mailboxes. However, such a scheme introduces new problems for the users, e.g., delay of service and denial of service. In this paper, we propose a *pre-challenge* scheme, which is based on the challenge-response mechanism and takes advantage of some features of email systems. It assumes each user has a challenge that is defined by the user himself/herself and associated with his/her email address, in such a way that an email sender can simultaneously retrieve a new receiver's email address and challenge before sending an email in the first contact. Some new mechanisms are employed in our scheme to reach a good balance between security against spam and convenience to normal email users. Our scheme can be also used for protecting other messaging systems, like Instant Messaging and Blog comments.

Keywords: Electronic Mail, Anti-Spam, Internet Security

1. Introduction

Email is one of the most valuable tools for Internet users nowadays. Unlike postal mail, email allows people living at any place of the Earth to communicate and interchange information almost instantaneously. It can contain and attach any digital information (from plaintext to complex objects), and the cost of transmission of a single message is minuscule once the infrastructure costs are paid.

However, the vulnerabilities and flaws in email protocols allow malicious users to send Unsolicited Commercial Email (UCE), or *Spam*. It can be defined as advertising messages (mostly for fraudulent products) neither expected nor desired by the intended receivers. Since it is very easy to flood users' mailboxes with little investment, spam is a big threat to email systems, resulting in the loss of time and money to email users.

A lot of research in the area of anti-spamming has been done in the past years. From statistical analysis to challenge-response, researchers tried to seek effective solutions to the spam problem. One of those solutions is *challenge-response*, which applies an old idea from Internet protocols to mail systems: when a sender sends an email to a receiver, he/she is given a challenge from that receiver which must be solved before the email reaches the receiver's mailbox. However, challenge-response schemes introduce some new problems for the users such as *delay of service* (when a sender waits for the arrival of the challenge from a receiver) and *denial of service* (when challenges are redirected to a victim's address if spammers use that victim's address as the source address).

In this paper, we propose a *pre-challenge* scheme, which is based on challenge-response mechanisms, preserving their benefits while avoiding their drawbacks (e.g., management of mailing list and error messages). It assumes that each user has a challenge associated with his/her email address, in such a way that an email sender can simultaneously retrieve a new

* Part of this work appeared in [31]. The first author's work was done during his attachment to Institute for Infocomm Research under its sponsorship.

receiver's email address and challenge before sending an email in the first contact. Each user will define his/her own challenge, which can range from a simple question about the user himself/herself to a hard-AI problem that only a human can solve.

Our scheme is easy to be integrated into existing email systems as it is a standalone solution, without changing the other party's software and configuration. In addition, our scheme does not create obvious inconvenience to normal email users, since they just have to solve a simple problem before mailing to a protected user for the first time. Our scheme also manages mailing list messages and processes mail error messages without any problem. Finally, our scheme offers protection against email harvesting.

The rest of the paper is organized as follows. In section 2 we review the reasons that make email systems an easy target for spamming, as well as the methods that are used by spammers to hide their identities. In section 3, we summarize the existing solutions against spam and analyze their limitations and/or problems. After that, we present our solution in section 4, and further discuss it in section 5. Finally, we conclude the paper in section 6.

2. Vulnerabilities in Email Systems

The original SMTP protocol was introduced in 1982 [1]. At that time security was not a major concern [2], mainly because the Internet was limited to a small number of hosts [3], and all users were trustful. Nowadays the scenario is very different: the Internet has around 170 million hosts [3], and continuous attacks have made security issues a priority. However, we are basically using the same email protocol as 20 years ago (with some minor modifications [4]).

An email is just a text message with some headers that transport certain information: (i) source, (ii) destination, (iii) path through the Internet, (iv) body of the message, and (v) extra headers. In a typical SMTP email delivery, a client MTA (i.e. a mail server) manages the outgoing emails from a certain network. This client MTA delivers emails using SMTP to their destination (server MTAs), each of which stores the incoming emails of its network.

The main problem in SMTP is the lack of authentication. When an email is received, it is not possible to know whether the source of the email is who claims to be. This is precisely the flaw that spammers make use of. (They cannot be traced, thus can take advantage of their anonymity in the network.) If a spammer only controls his own computer (client computer), he/she can spoof the source and destination headers (e.g., “From:”, “To:”, “Reply-To:”). These headers are part of the email content, so they are filled by the user who sent the email. The spammer can spoof these headers, changing them to a nonexistent email address or to another email address.

An bigger problem emerges when a spammer controls a client MTA. In this case the spammer can build an ISP that provides services to other spammers (*spam farm*). All the spam is sent through the ISP MTAs that are configured to provide anonymity to the outgoing emails. This anonymity is achieved by changing the source/destination headers and erasing all the headers that contain the path of the email at “Received:” headers.

Additionally, the client MTA of the spammer can try to hide its own identity by using a fake domain as a parameter inside the SMTP initial command <HELO> or <EHLO> sent to the server MTA. Although the real IP address of the connection will be recorded by the SMTP specification and can be used (with a reverse DNS process) to detect the client MTA of the spammer, the real source of spam is much harder to be discovered.

Before sending their junk, spammers must know the email addresses of their victims. There are many ways to harvest email addresses from the web (e.g., using web agents to search web sites or usenet posts). It is also possible to obtain addresses directly from a mail server with a side

attack over the SMTP protocol. The process is simple: automatic programs controlled by spammers contact a certain mail server, and after the initial messages they send repeatedly the control message "RCPT TO: <email>". If the answer of the mail server is "250 OK", then <email> belongs to that mail server.

3. Previous Work

Since the SMTP protocol is standard and widely deployed, it's impossible to change the protocol without changing the entire email infrastructure. That would require a slow migration, just like the actual deployment of IPv6. Therefore, most of the research in the area of spam control focuses on avoiding spam while maintaining the actual SMTP protocol and email infrastructure in order to ensure compatibility. This implies that anti-spamming solutions must be based on the operation with email headers or on specific implementation approaches.

One of the headers that can provide information regarding an eventual spam of the incoming email is the "Received:" headers. Since they indicate the path of the email through the Internet, they give information about the sender. Thus, we can obtain the client MTA from those headers, and check if that MTA is a source of spam (open relay or spam farm). There are some projects [5] that try to identify misconfigured email MTAs or major sources of spam. However, it does not work effectively against individual spammers, and innocent client MTAs might be blocked.

Another header that can be used against spamming is the address of the receiver, with policies or password-like extensions. In [6], a policy is encoded inside the email address. That policy can indicate a wide range of actions, from the expiration date to a complex program-like policy. When an email is received, the policy is checked. Subsequently, the email is discarded if the policy is not fulfilled. In [7 – 9], the address of the receiver is extended with a sequence of characters that act like a password. Every password is unique for a pair or group of users, and in most cases a proof of computational task [12] is needed in order to obtain the password. These solutions work well in some scenarios (e.g., using mail addresses in computer-based systems like web forums). However, as the email addresses created in such schemes are very hard to remember, they may cause problems when used by humans.

There are multiple works dealing with email content analysis based on artificial intelligence (AI) and statistical techniques [10,11]. They try to distinguish whether an email comes from a legitimate user or from a spammer by assigning a "spam score" (with a level of sensitivity that can be adjusted both automatically and by system administrators) to any incoming message. This approach can lead to false positives, and spammers may actively try to bypass the classifier algorithms.

Other implementation approaches against spam include *micropayments*, *challenge-response*, and *obfuscation* schemes. Micropayment schemes [12 – 15] are applied to email systems in order to prevent spammers sending millions of emails. This requires the user or client MTA to compute a moderately hard function in order to gain access to the server MTA. If a spammer wants to send a large number of emails to a certain server MTA, he/she must take substantial time to finish computation before sending each of the emails, making the business unprofitable. Such an approach is difficult to be applied to those client devices with very weak computing capability (e.g., mobile phones).

In challenge-response schemes [16 – 19], whenever an email from an unknown user is received, a challenge is sent back to that user. The solution to that challenge can be very simple (e.g., just a reply of the challenge), very complicated (e.g., a hard-AI problem like CAPTCHA [21]), or time consuming (e.g., using the micropayment schemes seen above). Only when the correct response is received, the emails of that user are allowed to enter into the receiver's mailbox. These schemes do not work when a human user is not involved in sending emails (e.g., in the

case of mailing lists). Moreover, these schemes may introduce some new problems such as delay of service (when a sender waits for the arrival of the challenge from a receiver) and denial of service (when challenges are redirected to a victim's address if spammers use that victim's address as the source address), etc.

In the obfuscation scheme, email addresses are displayed in an obfuscated format (e.g., John HIPHEN Smith AT yahoo DOT com), from which senders can reconstruct the real email addresses. It does not require any software from the user side or from the server side. However, the problem with this scheme is the constraints that the human users face when constructing the obfuscated addresses. As the combinations are limited, it allows AI-based harvest programs to easily retrieve real addresses. Moreover, once the spammer captures the email, there is no protection against spam (unless other solutions are utilized).

The technical solutions discussed above have their own limitations or weakness on countering again spam. Many of email users still receive large amount of junk mails everyday, and the spam problem has yet been well addressed.

The solution by law enforcement is also being discussed. For example, Korea requires that an ad mail must be marked with @ in the subject line. But how about junk mails sent from other countries? The question is that not all countries are enforcing the law. That means the Korean law cannot punish the junk mail senders in other countries. Another issue is repudiation. A junk mail can advertise a merchant's product, but the merchant may not be the sender. If the merchant is sued, he/she may claim that someone else sent it. (There is lack of authentication, not to say non-repudiation.) That means the solution by law enforcement alone may not work well. A good technical solution is still necessary to counter against spam.

4. A Pre-Challenge Scheme

We have two guidelines in designing a new scheme to counter against spam.

- The solution should be standalone, no need to change the other party's software and configuration. Put differently, any potential sender must not install any plug-in in order to send an email to a protected mailbox.
- The solution should not create obvious inconvenience to normal email users. In other words, there should be a good balance between convenience and security.

4.1 Overview

As stated, our pre-challenge scheme is based on challenge-response mechanisms in the sense that both of them impose a challenge that must be solved by a potential sender. However, in the pre-challenge scheme, the sender retrieves the receiver's email address together with his/her challenge simultaneously (see Fig. 1). Once the challenge is solved, the answer will be included inside the email.

When a mail from an unknown sender arrives, the receiver's system tests whether that mail contains an answer to the challenge. If the test turns out positive, the sender is *white-listed*. That means future mails from this sender will get into the receiver's mailbox without being checked again.

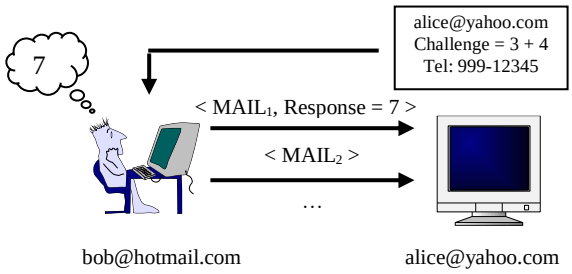

Figure 1. Basics of the Pre-Challenge Scheme.

The goal of our scheme is to check whether there is really a human behind a sender's computer. The reason is that spammers use automatic programs to send their propaganda, and they feed these systems with email addresses obtained by searching web sites and mail servers. However, it is a bit hard for these programs to retrieve a challenge that matches an email address and even harder to answer each of these challenges. Therefore, whenever a spam arrives to destination, it will be automatically discarded if no correct answer to the challenge is attached.

In comparison with a challenge-response scheme, our pre-challenge scheme preserves its benefits while avoiding its drawbacks, as we explain in the following:

- Suppose a sender wants to send an email to a receiver for the first time. In the case of a challenge-response scheme, the sender's MTA would need to start an off-line three-way handshake with the receiver's MTA in order to get the receiver's challenge. On the contrary, in our pre-challenge scheme, because the receiver's challenge could be available in advance, the sender can directly solve the challenge and send the email to the receiver[1]. Therefore, there is no delay even for receiving mails from unknown senders, and there are less overheads on the MTAs.
- With a challenge-response scheme, if spammers forge a sender's address in their mails, the challenges will be sent to that address. Thus, if the address belongs to a real user, this user will be the target of a DDoS attack [20]. Within the pre-challenge scenario, this attack will not take place because a receiver need not reply an unknown sender's request for a challenge.
- A challenge-response scheme can work with mailing list only if some rules are manually introduced. Moreover, it cannot handle mail error messages properly. As we will show in section 5.1 and section 5.4, the pre-challenge scheme manages mailing list systems and processes mail error messages without any problem.

Another benefit of the pre-challenge scheme is the continuous protection that the scheme provides against email harvesting. When a correct email address is retrieved by a spammer, he/she needs to retrieve the solution of the pre-challenge at the same time to make the address usable (and sellable in, for example, CD collections). But the user can change the pre-challenge at any time (see section 4.2), making the combination <email, solution> useless.
The pre-challenge scheme can be easily integrated with the actual email infrastructure, because it does not require any change to the existing email protocols, like POP3, SMTP and IMAP. It can be implemented as a "plug-in" to any email server, which must provide and maintain a set of lists and policy rules (see sections 4.3, 4.4 and 4.5), and must be able to interact with the email account owners for updating the pre-challenge solution and providing some manual procedures (as adding new addresses to the *white-list*). Since in most cases the email servers for receiving and sending messages are different, both servers must belong to the same sub-domain to facilitate the secure sharing and management of the required lists and policy rules.

[1] The frequency of challenge update is a security parameter decided by the receiver, based on his/her own experience, to control the risk of replay attacks from spammers.

4.2 Challenge Retrieval and Update

A challenge associated with an email address is defined by its human owner. Each user has one challenge at a time to be used by all incoming emails, and the challenge can be updated at any time at his/her own discretion. The challenge can range from a simple question (e.g., "what is the name of my dog?" in a pet-related blog) or mathematical operation to a hard-AI problem that only a human can solve (e.g., CAPTCHA [21]).

Normally a user's challenge is published next to this user's email address. Since any potential sender must retrieve the email address of the receiver before contacting him/her, challenge and email address can be accessed at the same time. However, in certain cases, a challenge may not be accessed directly. Instead, a URI may be provided to retrieve the challenge.

Since the challenge is not restrained to obfuscate a valid email address, which has a fixed structure (name, domain), the user has more freedom to produce it. When stored inside a website, the challenge can take advantage of its form and content – personal information, the theme and visual appeal of the website etc. Static environments (e.g., business cards or newspapers) can store directly the answer to the challenge, because spammers do not target them in order to harvest addresses.

The strength of the pre-challenge scheme is related to the strength of the challenge and to the huge number of email users. An easy challenge can be solved in almost no time by an automatic program, and advances in the area of artificial intelligence simplify the task of partially solving certain hard-AI problems [22]. However, since every user chooses his/her own challenge and the way to publish it in his/her own website, automatic spammer programs must be able to detect the challenge, identify its type (e.g., "is it an obfuscation problem or a human-recognition problem?"), understand it, and solve it.

If the challenge is published in a static environment, it may become outdated. If a user's challenge has been updated but a sender knows neither the latest challenge nor the location of the challenge, the sender can still email to that user by including the answer of an old challenge. Then the sender will receive automatically the latest challenge in the reply from that user. We have more discussions on this problem in section 4.5 and section 5.2.

Finally, another option for retrieving a challenge is using a majordomo style service [25]. In this service, a potential sender requests to an email server what is or where is located the challenge of a receiver. To prevent spammers to use this service as a collector of valid email addresses, the service must return a false challenge for every non-existent user.

4.3 Data Structures

The pre-challenge scheme requires certain data structures to accomplish its tasks. The two most important structures are the actual challenge (or a URI where the challenge can be found), and the solution to the challenge. By using these structures it is possible to advertise the actual challenge and to check whether an incoming mail has solved the challenge. Additionally, the solutions to old challenges must be stored, as discussed later.

Other structures needed by the scheme are the *white-list* and the *reply-list* (both used by some challenge-response schemes), and the *warning-list*, that is a structure specifically created for our new scheme. Each of those structures contains a list of email addresses and, optionally, a timestamp that indicates the time an email can be in the list.

White-List. The *white-list* contains email addresses in such a way that emails coming from those addresses are accepted without being checked. With this list, an email sender who has

already solved the challenge in the past need not solve it again in the future, even if the owner of the protected account changes the actual challenge. Some email senders may even be *white-listed* by a receiver at the set-up phase if they are already known. Those senders are marked in order to send a confirmation when receiving their first message (see section 4.5). This list could be manually modified by a human user.

Reply-List. The *reply-list* contains email addresses of those users to which the local user has sent email to, and has not replied yet. The use of this list is justified because the local user is the one who initiated the communication with those users; hence, there is no need to check any challenge when replies are received. This list will be managed automatically by the local user's system.

Warning-List. The *warning-list* contains email addresses of users that have sent an email containing the answer of an old challenge. The existence of this list is justified because an email message with an old response will cause a reply from the receiver indicating the new challenge. With this list, the local user does not need to send that reply more than once. This list will be reset every time when the challenge is updated, and will be managed automatically by the local user's system.

Every mail address protected by the pre-challenge scheme must have one unique instance of these data structures. However, there is an exception to this rule: if a certain user has a group of different but related email addresses (i.e., mails composed by the user have different addresses in the "From:" header and the "Reply-to" header), those mail addresses should share the same data structures (at least the *reply-list*).

4.4 Security Levels

The pre-challenge scheme can be configured to work at two security levels, *high security* and *low security* (see Fig. 2). The main difference between these two levels is how the *reply-list* is queried.

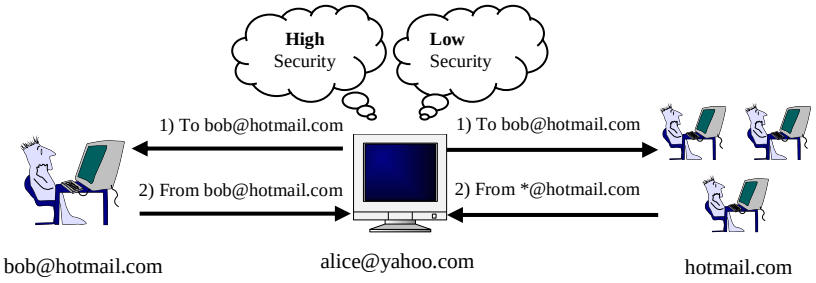

Figure 2. High Security Level and Low Security Level.

The scheme starts working at the high security level of protection. High security means that all queries in the *reply-list* are done by looking for a <user, domain> match, and the matched entry will be erased from the *reply-list*. For instance, when an email is received from *bob@hotmail.com*, the fields "From:" and "Reply-To:" are checked, and the *reply-list* will be queried for a <bob, hotmail.com> match.

On the other hand, low security means that all queries in the *reply-list* are done by looking for a <*, domain> match. Therefore, when an email is received from *bob@hotmail.com*, the fields "From:" and "Reply-To:" are checked too, and the *reply-list* will be queried for a <*, hotmail.com> match.

The reason why the pre-challenge scheme needs these two levels of security is that some email accounts have different addresses for receiving and for sending email. This usually happens with mailing lists, and this issue will be discussed in section 5.1.

4.5 Architecture

Now we explain the design of our pre-challenge scheme. To simplify the explanation, we assume that user *A* is using the pre-challenge scheme while user *B* is not.

(1) When *A* sends an email to *B*, *B*'s email address is added to the *reply-list* if it is not already in the *white-list*.

```
A → B: message
        if Not In (B, white-list) then
           Add(B, reply-list)
        Endif
```

(2) When *B* sends an email to *A*, *A* checks if *B*'s address is listed in the *white-list*. If this is the case, the mail reaches *A*'s mailbox. Additionaly, if that email is the first message *A* received from *B*, *B* receives a confirmation email.

```
B → A: message
        if In (B, white-list) then
           A → MailBox: message
           if Marked (B, white-list) then
              A → B: confirmation
              UnMark (B,white-list)
           endif
           exit
```

(3) If *B* is listed in the *reply-list*, the mail reaches *A*'s mailbox and *B* is added to the *white-list*. We should point out that the query to the *reply-list* is different according to the level of security being applied, as seen in section 4.4. In case of using a high security level, *B* is erased from the *reply-list* because *A* received the reply expected from *B*.

```
        else
        if In (B, reply-list) then
           A → MailBox: message
           Add (B, white-list)
           if Security (HIGH) then
              Remove (B, reply-list)
           endif
           exit
```

(4) If *B* is not listed in any list, the system checks whether the challenge of the email has been solved. If it is solved, the mail reaches *A*'s mailbox and *B* is added to the *white-list*. Additionally, *B* receives a confirmation email.

```
else
if Solved(message, challenge) then
   A → MailBox: message
   Add (B, white-list)
   A → B: confirmation
   exit
```

(5) If it is not solved but the message has a solution to an old challenge, the system checks if *B* is listed in the *warning-list*[2]. If that is the case, the mail is discarded. Otherwise, *B*'s address is added to the *warning-list* and *B* gets a reply containing information about the new challenge.

```
else
if Solved(message, old challenges) then
   if In (B, warning-list) then
      A → Trash: message
   Else
      Add (B, warning-list)
      A → B: actual challenge
   Endif
   Exit
```

(6) If it is not solved and has no solution, the email is discarded without any reply to *B* indicating this fact. The problem of accidental discard of a legitimate email will be addressed in section 5.3.

```
else
   A → Trash: message
endif
```

It should be noted, however, that discarding the message does not mean that the user cannot read it. The scheme can be configured for labeling the message with a "spam score" and placing it in a special fold of the mailbox if the owner of that mailbox desires so.

4.6 Normal Scenarios

When a normal user wants to send his/her email to a receiver in the first contact, the following scenarios should be considered.

Scenario 1. *A* wants to send an email to *B* who is working with the pre-challenge scheme. In this scenario, *A* retrieves *B*'s email address along with the challenge. Then *A* solves the challenge and includes it in his/her first mail.

- If the challenge is the one actually in use by *B*, the pre-challenge scheme will accept the incoming mail and will add *A*'s address into the *white-list*. Afterwards, whenever *A* sends additional emails to *B*, they will not be checked by the pre-challenge scheme.
- If the challenge that *A* has retrieved (and solved) is an old one, the pre-challenge scheme will not accept the incoming mail. However, the scheme will discover that the information received corresponds to a solution for an old challenge. Thus, a reply will be sent containing (or pointing to) the actual challenge. Also, *A*'s address will be added into the *warning-list* in order to avoid sending the same reply to *A*.
- If *A* does not know whether *B* works with the pre-challenge scheme (either because *A* is not used to computers, or because *B*'s challenge is not available when obtaining *B*'s email

[2] Note, the *warning-list* will be reset whenever the challenge is updated.

address), then the pre-challenge scheme silently discards *A*'s email because it contains no solution to any challenge. This may lead to a problem of accessibility. (*A* does not know whether the mail reached its destination or whether *B* ignored the email.) This issue will be further discussed in section 5.3.

A special case of this scenario is that *A* and *B* know each other (their addresses are stored in their respective address books) before *B* activates the pre-challenge scheme. In this case, *B* can add the addresses stored in his/her address book directly into the *white-list*, hence *A* will not need to solve any challenge (as seen in section 4.3).

Scenario 2. *A* who is working with the pre-challenge scheme wants to send an email to *B* that will reply the former. Since the pre-challenge scheme adds any outgoing email address to the *reply-list*, whenever the first reply arrives, it will be automatically accepted and *B* (that sends back the reply) is *white-listed*. Later on, for any incoming email from *B* will be automatically accepted.

Scenario 3. *A* who is working with the pre-challenge scheme wants to interact with a computer-based receiver (e.g., a mailing list). Because of the complexity of this scenario, it will be discussed separately in section 5.1.

4.7 Spam Scenarios

When a spammer wants to send his/her advertisements to a final user that operates the pre-challenge scheme, he/she faces the following scenarios.

Scenario 1. The spammer only retrieves the email address of a target, but not his/her challenge. When the spam is sent to the target, it will be silently discarded because no solution to a challenge (old or actual) is included.

Scenario 2. The spammer only retrieves the email address of a target, and impersonates as a sender that happens to be in the receiver's *white-list*. Here a problem arises: due to the lack of authentication in the email infrastructure, it is not possible to distinguish between a message from a trusted sender and a message from a spammer, hence the message will be accepted.

All schemes that use a *white-list* share this problem, but this is not a serious issue because spammers must find the white-listed senders for all the addresses he/she want to spam. And for millions of addresses to spam, this is unprofitable.

Scenario 3. The spammer can retrieve both the email address and its challenge, and try to solve the challenge in order to reach the target's mailbox. The spammer can even use hacker's tactics, intercepting recent emails from legitimate senders to obtain valid answers of challenges. When a user detects such a spam, he/she can simply update his/her challenge and remove the spammed email address from the *white-list* to stop the spam.

It could seem that a spammer, using a little investment (solving one challenge), can send many pieces of spam to a given email address (a replay attack). It could also seem that a group of spammers interchange their solved challenges of the corresponding users in order to lessen each spammer's effort on accessing the victims' mailboxes. However, what spammers want is to send millions of messages. In addition, as the challenges are different for every user and a challenge can be a hard-AI problem that only a human can solve, the task of repeatedly solving or sniffing a new challenge per user, or hiring cheap labor in order to send spam, becomes unprofitable.

5. Further Discussion

Here we further discuss how our scheme works for users in a mailing list, and whether our scheme can make a challenge easily available to users and make users to be sure on the delivery status of an email. We also discuss how to manage mail error messages, and the application of our scheme in other systems like Instant Messaging (IM) systems.

5.1 Mailing Lists

Mailing lists [23 – 25] share a common behavior. Firstly, a user contacts a mailing list in order to join the list. Then, the mailing list sends a challenge to the user in order to prove that the user is a real person. When the challenge is solved the user is added into the mailing list, and receives every message that is sent to the list. Finally, when the user wants to leave the mailing list, he/she solves another challenge.

If the user is enabled with the pre-challenge scheme, he/she faces a problem: behind the process of a mailing list there is a computer. This computer manages the subscription and distribution processes using non-standard automatic methods.

A possible solution would be to add to the system a mail analyzer; so the pre-challenge system can analyze a reply and adapt its behavior in managing those mails. However, this approach is not desirable because a specific module must be added for the analysis of the behavior of every mailing list, thus adding a substantial overload to the system.

Fortunately, there is a better solution to this problem. The core of the solution relies on two premises. First, all mailing list implementations have a common characteristic: all their mails come from the same domain (their "From:" line in the header shares the same domain). Second, a user normally only subscribes to a few mailing lists in a year.

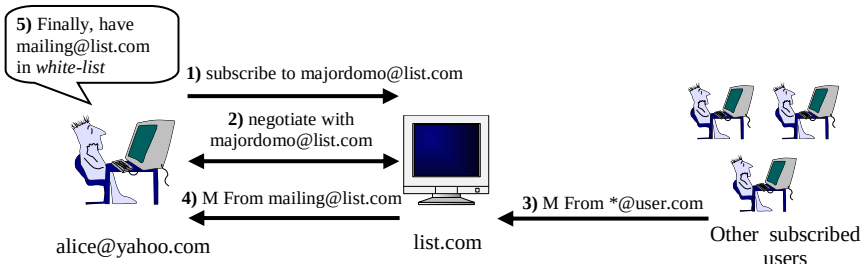

Figure 3. Process of Subscription to a Mailing List.

Therefore, a user can switch to the low security level (see section 4.4) whenever he/she wants to subscribe to a mailing list. At the low security level, all the incoming mails from the mailing list domain (including all the challenges and all the messages from the mailing list) that have a match in the *reply-list* are accepted into the user's mailbox and their senders are *white-listed*. When the user finally receives the first mail of the mailing list, he/she switches to the high security level (see Fig 3).

The risk of inserting a spammer inside the user's *white-list* while the user is at the low security level is very low, because the spammer's email address must have the same domain as the people in the user's *reply-list*.

Also, the user can set up the system not for adding the incoming mails to the *white-list* when running at the low security level, but for adding to a temporary *white-list* instead. He/She will decide later whether to add (manually) them into the final *white-list*.

5.2 Availability

A challenge defined by a user might be placed either with the email address or separately, and it can be published either in a dynamic environment (like a web site) or in a static environment (e.g., newspaper or business card).

It is clear that some availability problems exist when the challenge is not published along with the email address. If a sender cannot obtain the challenge of a new receiver and solve it, his/her email may not be able to reach the receiver's mailbox. This may happen either because the sender has no access (e.g., via the Internet) to the challenge, or the place (e.g., a web site) that contains the challenge is under a denial of service attack.

Finally, there is an availability problem that is common for both pre-challenge and challenge-response schemes: A challenge easy for a normal user might be impossible to solve for a disabled user. For example, a blind user will find impossible to solve a challenge based on images without help.

As a conclusion, if the challenge is published along with the email address we have (almost) no problems of availability. But if the challenge is published in another place, it can be outdated or may be inaccessible. It might be good to provide both the challenge and a URL that point to the challenge in static environments for better availability. In case the URL does not work, the challenge (even if outdated) can still be used by an email sender to get in touch with a new receiver. (The receiver will reply with the latest challenge on receiving the answer of an old challenge.)

5.3 Accessibility

One of the main issues in the pre-challenge scheme is that an incoming email from a new sender without answering a challenge is automatically discarded, and the sender is not notified. This approach avoids the increment of Internet traffic due to the responses to spammers' mails, but also introduces a problem: a normal sender is not sure whether a receiver really got the email.

A possible solution is to define a standard prefix in each email address that is enabled with the pre-challenge scheme. In such a way, the sender knows clearly that a challenge should be answered in his/her first email to such a receiver and a notification is expected should the email reach the receiver's mailbox.

There is an alternative solution if the pre-challenge scheme is implemented at the MTA level. In this solution, the sender is warned of the invalid answer of challenge using the error reporting mechanism of the SMTP delivery negotiation protocol. This protocol works as follows:

1. The client MTA of the sender side contacts the server MTA of the receiver side. After exchange of several control messages that indicates who is the sender and who is the receiver, the client MTA requests permission to start sending the content of the email (using the *"DATA"* command). Afterwards, the server MTA allows the operation (replying with a "*354 Enter mail, end with a single "."*. " command).
2. The client MTA sends the content of the email to the server MTA, ending with a single ".". Next, the server MTA checks if the email must be accepted or rejected[3]. If it is accepted, the server answers with the "*250 2.5.0. OK*" command. If it is rejected, the server answers with the "*554 Transaction failed* " command.

[3] As stated in the SMTP specification: "If the verb is initially accepted and the 354 reply issued, the DATA command should fail only if [...] the server determines that the message should be rejected for policy or other reasons" [4].

3. If the negotiation fails, the client MTA creates an email that includes the cause of the error and the undelivered email. That email is sent to the original sender, if the client MTA does not manage his/her emails.

When the server MTA checks if the email is valid at step 2, it can search for the answer of the recipient's challenge in the email. Logically, at that point it has all the information (sender, receiver, email content) necessary for this task. If the check fails, it returns "*554 Transaction failed: Bad answer of challenge*" (indicating where the actual challenge is).

By using this solution, the final user will receive an error message if he/she sends an email with an invalid answer of a challenge, without increasing the Internet bandwidth in most cases. We have more discussions on managing error messages in section 5.4.

5.4 Managing Mail Error Messages

During the SMTP delivery negotiation between two MTAs, if an email cannot be delivered to its recipient, the client MTA has to send the original sender an email containing an error message. Errors can range from an invalid recipient to over-quota mailboxes, or (as seen in the previous section) pre-challenge errors.

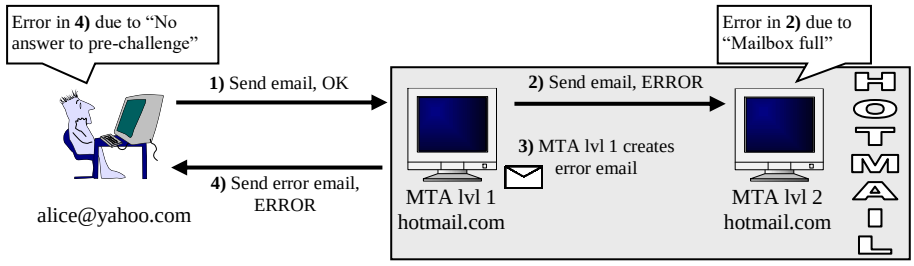

Figure 4. Problems Dealing with Automatic Messages.

A problem arises when the error message is not created by the MTA of the client that implements the pre-challenge scheme. An example is shown in Fig. 4. In the example, the error happens at MTA lvl 2, thus MTA lvl 1 creates and sends an error message back to the original sender. But MTA is a computer and will not include any answer of a challenge inside the error message. Therefore, it will not reach client's protected mailbox - a problem of availability.

This problem can be solved based on two premises. First, error messages can be identified with the "message/delivery-status" header, and have attached the email that caused the problem. Second, all emails have a unique ID issued by the original client MTA, stored in the "Message-ID" header.

When an error message arrives, the pre-challenge scheme accepts the email if both address of the recipient and ID of the original message are inside the *reply-list*. Thus, it is necessary to add the ID of outgoing emails to the *reply-list*. If the pre-challenge scheme is implemented at the client's machine, every outgoing email must store an identification number in an extra header, since the "Message-ID" field is added in the client MTA.

A spammer can try to take advantage of this approach, forging both the ID and the recipient of the original message in an error message, in order to bypass the scheme. He/She can send a fake error message directly to the user, or create a "real" error message using another MTA and forging the source address fields. Nevertheless, this attack can be done only once, due to how the *reply-list* is managed. Also the spammer must wiretap the communication channel in order to capture the ID, which is unprofitable for massive spamming.

5.5 Other Applications

The purpose of the pre-challenge scheme is to stop machine-based spam in the email architecture. However, the scheme can also be applied to other messaging services, like Instant Messaging (IM) and Blog comments.

5.5.1 IM Spam

IM systems basically provide instant communication services between two peers, and location services between a group of users or "Buddy List". These users must first register in the IM server in order to contact other users, thus avoiding spontaneous machine-based communications. Moreover, users have mechanisms for reviewing any peer, and can ban any suspicious user. Due to these features, spam is not a common problem in IM systems.

However, some IM services are not free from the spam problem, like ICQ's World-Wide Pager [26]. These services allow anonymous users to send an instant message, using an html form, to any IM user. As no authentication is performed, some programs can use these services for sending spam directly to IM users in real-time.

Since that type of IM services are embedded in web sites, the pre-challenge scheme can be used, allowing users to propose a challenge in order to use these IM services. In such a way, machine-based IM spam will not be profitable, as explained in this paper.

5.5.2 Blog Spam

Weblogs (or simply *blogs*) [27] are a type of web application where a user or group of users post bits of information in a common webpage that, in most cases, can be accessed by everyone. The information provided inside a single blog can cover a wide range of topics, from personal information to news digests or technical discussions. One inherent feature of blogs allows visitors to post a written comment into any information included inside the blog. Using this feature, it is possible to extend, correct or criticise the contents of the blog.

Unfortunately, blogs are also the target of spammers. Blog Spam abuses the comment system by submitting automatically comments with a link to, in most cases, an advertisement website. The purpose of blog spam is to cheat search engines into ranking the advertisement websites higher than others (as search engine classification algorithms give priority to a website when it has a high number of relevant hyperlinks pointing to it), thus giving them more chances to appear sooner when users receive the results of their search. As a side effect, blog spam makes relevant comments more difficult to find and read.

One solution to stop the economics of blog spam has been proposed by the same search engines that are the target of this threat [28]. The solution is very simple: when a hyperlink has the *"rel=nofollow"* tag, a search engine will not use that hyperlink in their classification algorithms, and the referenced website will not increase its ranking. However, this approach will not stop spammers to continue posting blog spam. Moreover, the search engines will ignore any link included in the users' comments.

Our pre-challenge scheme can be used as a part of the comments system in order to protect blogs against blog spam. In this approach, users must solve a pre-challenge (defined by the owner of the blog) before posting any comment. As explained before, machine-based blog spam will not be able to include their fraudulent hyperlinks.

6. Conclusion

Spam is a serious problem to many email users, and a lot of research on anti-spamming has been done in the past years. Since the current mail system is based on an unauthenticated architecture, it is hard to be 100% spammer-proof without introducing significant overheads on both the system and email users.

In this paper, we presented a pre-challenge scheme for spam controlling, based on challenge-response systems but avoiding their drawbacks. The purpose of our solution is to reach a good balance on security against spam and convenience to normal users.

Our scheme assumes that every user has a challenge associated with his/her email address. Therefore, if a sender wants to send an email to a receiver with no previous contact, he/she must first solve the challenge, and send both the message and the answer of the challenge. Since the challenge is defined by every user, and (in most cases) the challenge is a hard-AI problem, the overheads of harvesting <email address, answer of challenge> pairs will be so high that spammers' business will be unprofitable.

Our scheme is a standalone solution, since there is no need to install software or change the configuration in the sender's side. A sender can simply add the answer of a challenge in the subject line before mailing to a protected user for the first time. Our scheme allows email senders to have no delay in reaching the receiver's mailbox, and prevents the denial of service attack if the origin of the email is forged. Our scheme also manages mailing list messages and error messages properly. Finally, the pre-challenge scheme can be used for protecting other messaging systems such as *Instant Messaging* and *Blog comments*.

This scheme can also be used jointly with other major anti-spam solutions, like micropayments and artificial intelligence techniques. The reason behind this compatibility is because the type of protection that the pre-challenge scheme provides is centered in the protection of email against harvesting, thus leaving the door open to other solutions such as content analysis. This layered approach can provide a higher level of protection against spam. Moreover, the scheme can be integrated with authentication solutions like DomainKeys [29] or Identity-Based Encryption [30] that provides source domain authentication, hence thwarting attacks like using forged senders to bypass the *white-list* checking.

As a final note, a prototype of an anti-spam system based on the pre-challenge scheme is being implemented. The experiment result will help us to better access its security and usability.

References

[1] J. Postel. *Simple Mail Transfer Protocol*. RFC 821, Internet Engineering Task Force, August 1982.
[2] WBGLinks. *The Complete History of Hacking*. http://www.wbglinks.net/pages/history/.
[3] R. Zackon. *Hobbes' Internet Timeline*. http://www.zakon.org/robert/internet/timeline/.
[4] J. Klensin. *Simple Mail Transfer Protocol*. RFC 2821, Internet Engineering Task Force, April 2001.
[5] SBL. http://spamhaus.org/.
[6] J. Ioannidis. *Fighting Spam by Encapsulating Policy in Email Addresses*. In Proceedings of NDSS'03 (Network and Distributed System Security), February 2003.
[7] E. Gabber, M. Jakobsson, Y. Matias, and A. Mayer. *Curbing Junk E-Mail via Secure Classification*. In Proceedings of FC'98 (Financial Cryptography), pages 198--213, February 1998.
[8] R. J. Hall. *How to Avoid Unwanted Email*. Communications of the ACM, 41(3):88-95, March 1998.
[9] L. F. Cranor and B. A. LaMacchia. *Spam!*. Communications of the ACM, 41(8):74--83, August 1998.

[10] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. *A Bayesian Approach to Filtering Junk Email*. In Proceedings of AAAI'98 Workshop on Learning for Text Categorization, July 1998.
[11] P. Cunningham, N. Nowlan, S. J. Delany, and M. Haahr. *A Case-Based Approach to Spam Filtering that Can Track Concept Drift*. In Proceedings of ICCBR'03 Workshop on Long-Lived CBR Systems, June 2003.
[12] C. Dwork and M. Naor. *Pricing via Processing or Combatting Junk Mail*. In Proceedings of Crypto'92, pages 139--147, August 1992.
[13] C. Dwork, A. Goldberg, and M. Naor. *On Memory-Bound Functions for Fighting Spam*. In Proceedings of Crypto'03, pages 426-444, August 2003.
[14] M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber. *Bankable Postage for Network Services*. Proceedings of the 8th Asian Computing Science Conference, Mumbai, India, December 2003.
[15] Penny Black Project, Microsoft Research. http://research.microsoft.com/research/sv/PennyBlack/.
[16] M. Jakobsson, J. Linn, and J. Algesheimer. *How to Protect against a Militant Spammer*. Cryptology ePrint archive, Report 2003/071, 2003.
[17] M. Iwanaga, T. Tabata, and K. Sakurai. *Evaluation of Anti-Spam Method Combining Bayesian Filtering and Strong Challenge and Response*. In Proceedings of CNIS'03 (Communication, Network, and Information Security), December 2003.
[18] SpamArrest. http://spamarrest.com/faq/.
[19] SpamCap. http://www.toyz.org/cgi-bin/wiki.cgi?SpamCap.
[20] J. Mirkovic, J. Martin, and P. Reiher. *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms*. University of California, Computer Science Department, Technical Report #020018.
[21] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. *CAPTCHA: Using Hard AI Problems for Security*. In Proceedings of Eurocrypt'03, pages 294--311, May 2003.
[22] G. Mori and J. Malik. *Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA*. IEEE Conference on Computer Vision and Pattern Recognition, Madison, WI, June 2003.
[23] Ezmlm Mailing List. http://www.ezmlm.org/.
[24] Mailman Mailing List. http://www.list.org/.
[25] Majordomo Mailing List. http://www.greatcircle.com/majordomo/.
[26] ICQ Pager. http://www.icq.com/panels/messagepanel/.
[27] R. Blood. *The Weblog Handbook: Practical Advice on Creating and Maintaining Your Blog*. Perseus Publications. 2002.
[28] Google Blog. *Preventing Blog Spam (January 18,2005)*. http://www.google.com/googleblog/2005/01/preventing-comment-spam.html
[29] Yahoo DomainKeys. http://antispam.yahoo.com/domainkeys/.
[30] D. Boneh and M. Franklin. *Identity Based Encryption from the Weil Pairing*. Crypto'01, pages 213-229, August 2001.
[31] Rodrigo Roman, Jianying Zhou, and Javier Lopez. *Protection against Spam Using Pre-Challenges*. Proceedings of 20th IFIP International Information Security Conference, Chiba, Japan, May 2005.