Cryptographic Protocols to Prevent Spam

Version of September 21, 2005

Amir Herzberg

Computer Science Department, Bar Ilan University, Ramat Gan, Israel
`herzbea@cs.biu.ac.il, http://AmirHerzberg.com`

Email's main (and initial) use is professional and personal communication. Email is very efficient, convenient and low-cost, especially when automatically sent by programs to many recipients ('bulk email'). Therefore, email is also used to distribute other messages: from unsolicited offerings and ads, to malicious content such as viruses and scams. Users are rarely interested in these messages; the result it that the vast majority of email messages are undesirable and discarded. These unwanted messages are often referred to as *spam*. Spam is very annoying to users, and wastes considerable time and resources of email users and service providers.

There are few conventions for *content labels*, used to identify advertising or other potentially undesired messages. For example, some anti-spam legislation requires advertisers to use special prefixes such as 'ADV:' in the subject line [42, 43]. Messages that present correct content labels are not problematic, since they can be efficiently discarded by (most) users, who are not interested in them; therefore, we will exclude them from our definition of spam. Unfortunately, most messages containing potentially-undesired content do *not* contain appropriate content label, i.e. are spam.

We therefore prefer to use the term *spam* to refer to undesirable messages without appropriate content label. Spam senders (*spammers*) not only avoid putting correct 'warning labels' on their messages, but in fact often use different evasive techniques to make it hard to distinguish between their spam messages and desirable professional/personal communication. For example, spammers may add spaces or other special characters between letters, or inject intentional, minor spelling errors (`Viiagrra`).

Spamming wastes considerable machine and human resources - most notably, the recipient's time. Indeed, spamming is reducing the usefulness of email as a communication mechanism these days. Many users reduce or avoid the use of email, most users limit the distribution of their email address, and many desirable messages are lost by aggressive (human and automated) filtering. As a result, there are many proposals and mechanisms trying to control and prevent spam.

Spam can be a problem in any open, efficient, low-cost messaging system, and indeed spam is used e.g. in instant messaging; most of our discussion is relevant to spam on any messaging technology. However, the problem is most acute, at least so far, for email, and therefore we will discuss also some email specific issues.

We begin this manuscript by discussing the basic architecture and relevant properties of the email system, in Section 1, and the 'vicious cycle' between spam, malware and zombies, in Section 2. In Section 3, we present a categorization of spam controls. We discuss each of the three main categories in the following sections: content filtering schemes in Section 4; identity-based spam controls in Section 5; and spam controls based on charging spammers (costs) in Section 6.

1 Internet E-Mail and Spam

The Internet email architecture defines four types of agents (servers[1] and clients) handling email messages:

– Mail User Agents (MUAs), also referred to as email clients, are used by end-users to send and receive email.
– Mail Submission Agents (MSAs), also referred to as outgoing mail servers, are the servers to which the sending MUA communicates to submit (send) email messages.
– Mail Delivery Agents (MDAs), also referred to as incoming mail servers, are the servers which keep the mailbox of incoming messages for each email users, until the user downloads them to the MUA or erases them.
– Mail Transfer Agents (MTAs) are 'intermediate' mail servers, which facilitate the forwarding of email messages from the MSA to the MDA. MTAs may relay (forward) messages to other MTAs, to reach the destination MDA.

A typical setup is illustrated in Figure 1. In this setup, we show two Mail User Agents (MUA) of senders Alice and Carl, using the same Mail Submit Agent (MSA) to send messages to the same recipient, Bob. The MSA forwards the messages to Bob's Mail Delivery Agent (MDA), via MTA A, a Mail Transfer Agents (MTA) in Alice's domain; email messages are forwarded between the mail agents using the *Simple Mail Transfer Protocol (SMTP)* [36]. There may be zero or multiple MTAs through which the message flows in Alice's domain; the last MTA along the path of the message in the sending domain is sometimes called the *outgoing border MTA*. The outgoing border MTA, e.g. MTA A, transfers the message to the *incoming border MTA* in the recipient's domain, e.g. MTA B, which transfers the message to the recipient's (Bob) Mail Deliver Agent (MDA), directly or via a series of intermediate MTAs.

Bob's MUA contacts his MDA to download messages, often using the *Post Office Protocol (POP)* [30], the *Internet Message Access Protocol [28]*, or a web-form for web-mail. This process should be authenticated, to prevent others from reading or modifying Bob's mailbox. Web-mail is usually protected, like other sensitive websites and web services, e.g. by sending user-name and password over a secure (encrypted) connection. POP and IMAP support several authentication mechanisms, from simple transmission of user-names and passwords in the clear with the POP `USER` and `PASS` commands, to more advanced authentication protocols such as using S/key and Kerberos, described in [29].

[1] The three email server agents (MSA, MDA and MTA) are often co-located, and may be performed by a single server software module.

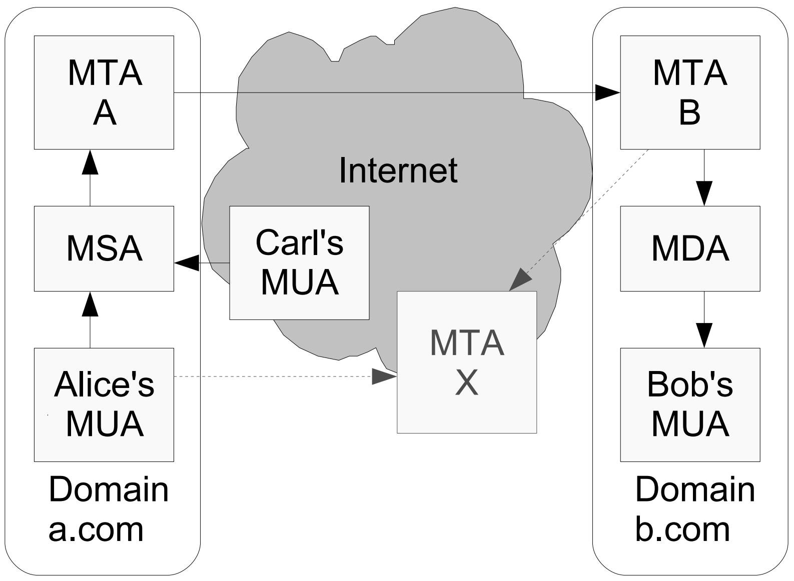

Fig. 1. Typical E-Mail Architecture and Flows: Normal email flow from Alice's MUA to Bob's is via Alice's MSA, one or more MTAs of Alice's domain, one or more MTAs of Bob's domain, Bob's MDA and finally Bob's MUA. Domains usually prevent direct SMTP to outside (over default port 25), e.g. Alice's MUA submission to MTA X. Carl's MUA submission to the MSA from outside its domain, usually requires authenticated communication.

For maximal connectivity, Internet email does *not* require any prearrangement between the outgoing email border MTA, e.g. MTA A, and the incoming email border MTA, e.g. MTA B. In particular, the connection between the two is not necessarily authenticated cryptographically; MTA B identifies MTA A simply by the Internet Protocol (IP) source address *src* specified in the packets it receives from MTA A. If MTA A misbehaves, e.g., sends (unacceptably high amount of) spam, then MTA B may detect this and refuse connections from it (i.e. from source address *src*). The spam-receiving MTA B may also report the spam-originating MTA A to different spam-control services, most typically blacklists of suspected spamming IP addresses and domains; see details in subsection 5.5.

This mechanism allows a receiving, incoming border MTA (e.g. MTA B) to block incoming email based on the IP address of the sending, outgoing border MTA (e.g. MTA A), if it has received spam from the IP address or if it included in a blacklist which MTA B uses. Notice that if the adversary is able to use multiple IP addresses when sending email, using SMTP, then all of these addresses may need to be blocked. Luckily, with the current version of the Internet Protocol (IP version 4, aka IPv4), there is a shortage of IP addresses, and a substantial cost associated with each IP address. Therefore, spammers are not likely to acquire a huge number of IP addresses, and blocking by IP addresses is effective.

A spammer may try to use an IP address belonging to a victim. Since email is transferred using SMTP, which is interactive and runs over a TCP connection, this requires the adversary to *intercept* packets sent to the victim's IP address, as well as to send packets with the victim's IP address specified as source. In this case, we say that the spammer has *Man In The Middle (MITM) capabilities* for the victim. Luckily, obtaining MITM capabilities requires the spammers to control routers, which are usually well protected by the operator of the victim's domain; this is much harder than 'just' using a fake IP sender address[2].

To deal with the cases where an attacker buys a domain of IP addresses, or (more likely) obtains MITM capabilities over a set of IP addresses belonging to a domain, recipients and blacklists take advantage of the fact that IP addresses are distributed in consequtive blocks. Therefore, a server or blacklist receiving spam from a significant fraction of the addresses in a block of addresses, may put the *entire* block of addresses on blacklist of suspected sources of spam.

This creates a motivation for domains to control the amount of spam outgoing from them, in order to avoid being 'black listed'. In particular, domains try to prevent unauthorized computers within the domain from sending mail directly to other domains, and possibly getting the domain blacklisted. This especially holds for computers that may receive different IP addresses, e.g. for dial-up lines.

[2] The IP protocol does not prevent the use of spoofed (false) sender IP address. Many Internet Service Providers (ISPs) block spoofed IP packets from their customers - but not all. Therefore, attackers may find an ISP allowing sending of packets with spoofed sender IP address, but much less likely to be able to intercept packets sent to a victim's IP address.

To limit the ability to transfer email (using SMTP) out of the domain to only designated border MTA server, domains use the fact that the SMTP connection between two border MTAs, e.g. A and B, is by default over TCP port 25. Therefore, many domains block attempts by unauthorized computers in the domain, to connect over port 25 to outside the domain; see the dotted arrow from Alice's MUA directly to MSA X in Figure 1. This policy is usually referred to as *port 25 blocking*.

In addition, domains need to limit the amount of spam sent by the authorized outgoing border MTA, e.g. MTA A. The main mechanisms available to the domain are to use content filtering to identify and discard outgoing spam (see Section 4, to identify and block spamming users, and to enforce reasonable quotas on total email sent by each user. The two last measures require identification of the email sender; this is usually trivial for a user within the domain, and somewhat more complex for a user outside the domain.

To allow a customer of the domain to connect to its mail servers from outside the domain, e.g. Carl's MUA connection to the MSA in domain A, many domains require Carl's MUA to use an authenticated connection. This can be done in a variety of ways, e.g. from using SMTP AUTH extension [33], which simply sends passwords in the clear, to using cryptographic authentication mechanisms such as running the SMTP connection over the Secure Socket Layer (SSL) protocol. Such authenticated connections are often done over port 587 rather than 25, to avoid being blocked by the remote ISP (hosting Carl's MUA), as a result of the port 25 blocking policy.

When an MTA receives an email message, it transfers this message to an email agent on the path toward the destination; this forwarding function is often referred to as *relaying*. In the early days of the Internet, most mail servers allowed e-mail relaying from any source to any destination; such a mail server is called an *open relay*.

More precisely, an open relay is an MTA that relays messages from outside its domain, toward a destination also outside its domain, and without ability to authenticate the source. For example, if MTA X in Figure 1 relays messages from Alice's domain to Bob's domain, while not belonging to either domains and not authenticating the MTA A, than MTA X is an open relay. Open relays are often abused by spammers, to send spam and avoid identification of their IP address. For example, if MTA B is an open relay, and relayed spam messages from MTA A to say MTA X in domain X, then MTA X may conclude that MTA B is a source of this spam. Therefore, mail servers are currently expected not to operate as open relay. See [32].

Furthermore, most mail servers, e.g. MTA B in Figure 1, will refuse to accept messages from known open relays, e.g. MTA X, or from any MTA belonging to a domain that is known to issue a lot of spam. This is usually facilitated by comparing the IP address of the sending MTA, against blacklists containing IP addresses of suspect (spamming) open relays.

Once spam is detected, this should be communicated to the spam-receiving border MTA (e.g. MTA B), for it to identify the spamming border MTA (e.g.

MTA A). This requires tracing back the sequence of mail agents thru which the message passed. To allow such trace-back, each mail agent along the path to the recipient, identifies the previous mail agent along the path, from which it received the email, in a RECEIVED: line which it adds to the email header; the RECEIVED: line specifies the time the email was received, and the domain name and IP address of the sending agent.

This process provides a reasonable level of identification of the path of (conforming) mail agents, and allows blocking of mail from blacklisted agents. By backtracking using the sequence of RECEIVED: headers, we can identify the source of a spam message or at least a non-conforming server along the path, e.g. an open relay.

However, this process does *not* ensure, as currently deployed, validation of the *sender* email account presented by the Mail User Agent (MUA) to the user. MUA present the sender's e-mail address from the FROM: or SENDER: headers of the message. In fact, many MSAs allow end-users to use arbitrary sender email address in the FROM: or SENDER: email message headers. This is an *intentional* design decision of Internet email standards [36, 37].

In particular, this allows a user to use the same email address when she sends[3] mail, while using multiple MSAs to submit mail (e.g. from work, home, cafe). Furthermore, when a MTA receives email from another domain, e.g. MTA B from MTA A, it currently does *not* validate that the email address specified in the FROM: field belongs to the sending domain. Indeed, such a restriction breaks many legitimate, common email scenarios, including e.g. many mailing list mechanisms. Some proposed spam-controls try to authenticate the sender's email address, or at least the domain address. Some even restrict senders to use email address in the domain, in spite of the problems this may create e.g. for (existing) mailing lists. However, *currently, email sender address cannot be trusted* - it can be easily spoofed.

Such spoofed email is used to send spam, e.g. to appear as mail from a known sender, and pass automated filters as well as 'manual filters' (cause user to read it rather than discard it). Spoofed email is also used to mislead users regarding the source of the message, e.g. in phishing attacks [4, 20]. The spoofed (fake) email addresses could be simply made-up, picked from public sources (e.g. web pages, forums), or picked from address books of penetrated computers (a common technique for viruses). In an extreme case, the spoofed sender address could belong to a specific victim. The attacker uses as sender address the address of a victim, in a phishing attack against customers of the victim, or when the attacker is trying to 'frame' the victim, i.e. make it appear that the victim has sent spam or other problematic email. See more details in the next section.

[3] Users may need to send using a specific email address, e.g. since when sending to a 'subscription only' mailing list, or for privacy considerations (e.g. not exposing whether sending from office or from a cafe). If users only care to receive replies to a specific email address, and do not care to use different sending email address, then they could specify the fixed address in the REPLY-TO: header.

2 The Vicious Cycle of Spam, Malware and Zombies

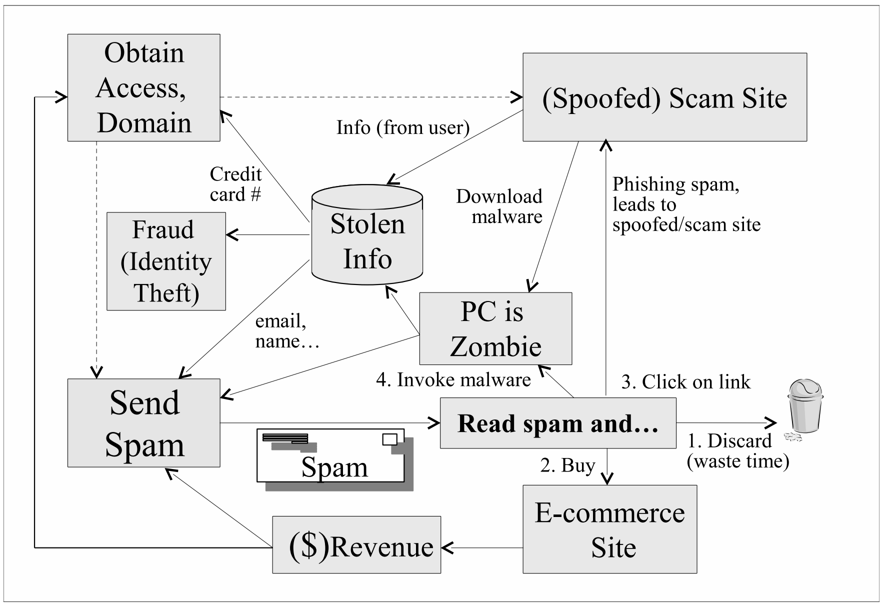

Fig. 2. The Vicious Cycle of Spam: the result of receiving spam may be just discarding it (1), but spammers hope it will instead result in purchase (2), providing profit and funds for more spamming activities. Spam from rogue sources often contains either links to spoofed/scam sites (3) or malware (e.g. virus) (4). Both malware and scam sites try to collect consumer information and use it unfairly, e.g. abuse his credit card number or online bank account, again providing profits and funds for more spamming. Malware is also used to send spam from 'zombied' computers (i.e. computers controlled by the adversary via the malware).

Spam is closely related to other threats to computer security, and in particular to different forms of malicious software ('malware') such as viruses and Trojan horses, and also to spoofed web sites and other online scams. There is a 'vicious cycle' in which spam plays a central role. The essence of the cycle are the following observations:

- Most of spam is produced by zombies, i.e. end-user computers controlled by hackers. Estimates are in the range of 80% of the spam produced by zombies.
- The main tool to gain control of zombies is by distribution of malware, such as viruses.
- The main mechanism to distribute malware is by spam, and indeed, malware accounts to a large percentage of the spam traffic.

– Spam and zombies also provide direct profit to attackers; spam - by revenue from ads and sales, and zombies - by collecting information, e.g., passwords and documents.

The vicious cycle is shown in more details in Figure 2. The cycle begin when a spammer obtains Internet access, and possibly also a domain, and begins to send spam (left side). The critical point in the process is the response of the user (or his agent) to the incoming spam. When a user reads a spam message, there are four main possibilities:

1. The user may simply discard the spam; here, the damage is mainly the waste of time by the user, as well as the bandwidth, storage and processing costs.
2. When spam contains advertising content, the user may actually respond favorably to the advertisement, e.g., by buying the advertised product or service.
3. Often, spam contains links to a malicious web site used as part of a scam; we use the term phishing for the method of luring users to a scam web site by spam email. Scam web sites usually trick users into providing them with sensitive, personal information, in different ways; one of the most common is by misrepresenting (*spoofing*) the scam web site as a trustworthy web site, e.g., a large bank, and asking users to login to their account, thereby collecting user accounts and passwords.
   Some of the sensitive information collected in this manner may be of direct financial value, e.g. credit card numbers or password for online banking. Attackers can abuse such information to steal funds, some of which may be used to finance additional spamming operations, e.g., purchase new domain names. Attacker can use other personal information, such as names, email accounts and areas of interest, to form new spam messages to this user and users related to her, increasing the effectiveness of the scam and making automated filtering harder.
   Finally, scam web sites may try to cause execution and/or installation of malicious programs such as viruses on the client's machine, by including them as active content on the web page (scripts, applets, etc.), or by convincing the user to download them. This is similar to execution or installation of malicious content attached to the spam, as we discuss next.
4. Spam messages often contain malicious programs (*malware*) such as viruses and Trojan-horse programs. Such malware, running on the user's computer, often use this computer's resources, including email accounts and email address books, to send additional spam messages. Finally, malicious websites (often spoofed sites, i.e., clones of 'good' sites), as well as some malware (*spyware*), collect personal information, used for fraud (especially identity theft) and for composing harder-to-filter spam messages.

3 Categorization of Spam-Prevention Techniques

Spam is an annoying, familiar problem to most email users, and it often looks (deceptively) easy to solve. As a result, there have been a very large number of proposals on how to deal with Spam. Unfortunately, many of the proposals ignore some of the basic realities of email, often by evaluating the value of the proposal in a scenario where the proposal is widely or globally adopted, and sometimes requiring major changes to the email system. The reality is that while Spam annoys many people, solutions should allow for gradual adoption and minimize disruption or changes to current email usage. Unrealistic proposals are often referred to as FUSSP, standing for *Final Ultimate Solution to the Spam Problem*.

Even ignoring some obvious FUSSP proposals, there are many deployed and proposed technical mechanisms for preventing spam, or at least controlling its proliferation; we refer to such means as *spam controls*. Spam controls differ in many aspects and assumptions, and there are often heated debates on merits and drawbacks of different approaches, often arguing whether a given proposal is a FUSSP (i.e. unrealistic) or good.

In Figure 3, we suggest a categorization appropriate for some of the most of the important spam controls. There are so many different ideas and approaches, that this categorization is quite complex; to avoid excess, we did not include (important) non-technical mechanisms, e.g. laws and regulations [42, 43]. In the rest of this section, we explain the categorization.

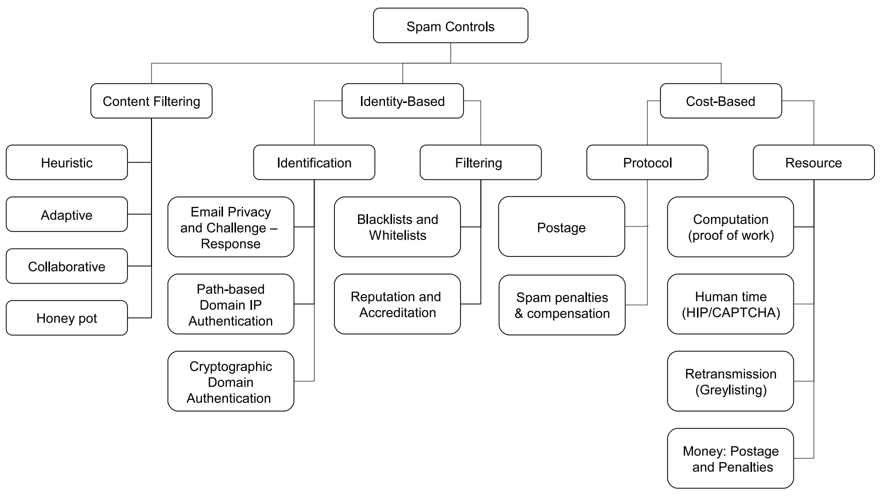
Spam Controls
Content Filtering
Identity-Based
Cost-Based
Heuristic
Adaptive
Collaborative
Honey pot
Identification
Filtering
Protocol
Resource
Email Privacy and Challenge – Response
Path-based Domain IP Authentication
Cryptographic Domain Authentication
Blacklists and Whitelists
Reputation and Accreditation
Postage
Spam penalties & compensation
Computation (proof of work)
Human time (HIP/CAPTCHA)
Retransmission (Greylisting)
Money: Postage and Penalties

Fig. 3. Categorization of Spam Controls. Identity-based and cost-based scheme require adoption by both senders and recipients (or their agents), while content filtering requires only adoption by recipients.

The top level categories are *content filtering, identity-based* and *cost-based* spam controls. Content filtering is clearly the most widely deployed. Content filtering tries to distinguish between spam and non-spam (sometimes called *ham*) messages. Therefore, content filtering is subject to errors, usually referred to as *false negatives* and *false positives.*

A *false negative* is a spam message which was not detected by the content filter, and delivered to the user. Much worse are *false positives*: innocent, non-spam (ham) messages, which the content filter mistakenly categorizes as spam (and blocks). Users are very sensitive to false positives, which may result in lost messages. Commercial and experimental content filters claim very low false negative rates, and even lower false negative rates, e.g., 0.5% and 0.03% [15].

Content filters are easy to deploy, requiring adoption only by the recipient and providing immediate relief from (most) spam. The combination of ease of deployment and low false positive and false negative rates, makes content filtering widely used. However, as we explain in Section 4, content filtering does have some limitations, motivating the development and deployment of complementing spam controls.

Spam controls *not* based on content filtering can be broken into two top-level categories: *cost based spam controls* and *identity based spam controls.* Cost based spam controls attempt to ensure that sending spam would involve non-negligible costs, monetary or otherwise. The cost is either for every message sent (where the term postage is appropriate), only for spam messages, or for every message, but with mechanisms for refunding the payment for non-spam messages.

Identity-based spam controls involve two aspects:

Identification of the sender or of another party responsible for the message. Identification can help not to control spam, but also to prevent phishing and other fraudulent (spoofed) email. Some identification mechanisms use cryptographic authentication mechanisms such as digital signatures, others rely on the difficulty of intercepting responses sent to specific (victim) recipient.

Identity-based Filtering are mechanisms for determining whether to receive messages from a particular identity. These include blacklists, whitelists, reputation systems and accreditation services.

We discuss the different Identity-based spam controls in Section 5 below.

Finally, cost-based spam controls try to reduce the motivation of spammers, by forcing them to pay a fee for sending spam (or every email). Some cost-based spam controls also compensate the spam victims (recipients). We discuss cost-based spam controls in Section 6 below.

4 Content Filtering Spam Controls

Content filtering spam controls try to identify email messages that belong to one of several categories of potentially-undesirable email, e.g. messages contain advertising, offensive or malicious content. This function could have been trivial, if senders always marked such messages using a standard convention, e.g. using the word ADV: in the subject, as required by several anti-spam laws [42, 43]. However, most spam messages ignore such laws and conventions. Content filtering tries to detect spam by inspecting the content of the messages, often including header fields. Content filters use a variety of mechanisms to detect spam, often combining the 'scores' of several mechanisms to reach the conclusion. These mechanisms include:

Heuristic filters search for known patterns[4] of some common spam messages, e.g., certain words (`XXX, FREE`...) or files containing a known viruses. The patterns are collected using heuristic and manual processes.
Known patterns filters are relatively weak, since spammers are usually aware of the patterns filtered against in a default installations, and users rarely add patterns. Therefore, it is easy for the attackers to avoid the known patterns while still conveying the same spam message to the human reader, e.g. using spacer characters (`v*i*a*g*r*a`) or minor spelling errors (`viiagraa`).

Adaptive (learning, Bayesian) filters collect statistics from collections of email messages and from user's actions on received messages, to automatically categorize (identify) spam vs. non-spam messages. Currently, statistical (aka adaptive) filters are quite effective at identifying spam, with excellent reported rates of only 0.03% false positives (non-spam mistaken for spam) and 0.5% false negatives (undetected spam) [15].
However, as statistical filters become more common, spammers may gradually adjust by avoiding or mangling any string which is likely to be filtered. Of course, spammers will have to avoid extreme mangling, since that may by itself be easy to detect.
Furthermore, spammers can use similar learning algorithms to test their messages and modify as needed to avoid detection, especially by identifying and adding words or text which is common in non-spam messages (for the target group of recipients). Spammers may also disrupt adaptive filters by intentionally adding words and phrases from legitimate mail to the spam, thereby increasing the false positive rate to unacceptable levels. Further research may be needed, to provide independent confirmation of the excellent claimed rates for adaptive filters, and evaluate whether spammers are able to adapt and avoid detection and/or increase false positive rates.

Collaborative filters operate by collaboration of multiple users, some or all of which identify spam by manually evaluating the content of a message. This relies on the ability of humans to identify spam; indeed, a trivial version of such 'human filtering' is a secretary filtering incoming messages. Collaborative filters simply share the filtering work among multiple persons, usually all in the capacity of email recipients. When user Alice identifies and marks a spam message in her inbox, her Mail User Agent (MUA) will automatically alert the content filtering mechanisms of other users (running on each MUA or as a service in a mail server). Collaborative filtering mechanisms

[4] Sometime, these patterns are referred to as 'signatures', not to be confused with cryptographic digital signatures.

are most useful as a complement to adaptive content filters, by providing timely updates to block new variants of spam. Collaborative filters need to take into account that users may often err, and identify non-spam a spam or vice verse.

Honeypots use special email accounts which are made available to spammers, typically by placing them in public web sites, along with human-readable notice asking *not* to send any mail to the honeypot address. The honeypot accounts should never receive any legitimate mail. Therefore, any message sent to the honey-pot accounts is spam. The spam collected by the honeypot is used to adjust the filtering mechanisms, as well as to identify spamming users and domains, for identity-based filtering and possibly to take other actions against spammers, e.g. litigation.

State-of-the-art content filtering mechanisms are very effective against current spam. However, content filtering results in an ongoing battle between spammers and filters, with spammers designing messages to bypass filters, testing them against current filters. Some of the techniques already exploited by spammers to avoid content filters include sending very short messages, containing mostly a pointer to a website, 'hiding' the message e.g. inside graphics to make it hard to extract the text, and presenting the spam as a bounced email message[5] returned to the user. Spammers may also use openly available messages, e.g. from public mailing lists, and from personal mailboxes exposed by malware, to create messages which are more similar to legitimate mail for the target user.

There are other limitations of mail filters. In particular, mail filters may not be able to correctly distinguish between some important classes of content. This includes identification of age-inappropriate videos and distinguishing between malware and benign programs. Therefore, it is advisable to combine content filtering with complementing spam controls, based on identification and/or payments.

5 Identity-Based Spam Controls

In this section we discuss spam controls based on the *identity* of a party responsible for the email, usually meaning either the originating email sender, the outgoing border MTA of the sending domain, or a resending agent such as an email forwarding service. The identity is established using *identification and authentication mechanisms*; then, the receiving party uses *identity-based filtering* mechanisms, such as reputation and accreditation services, to determine whether to allow email from a specific sender identity.

As discussed in Section 1, the main goal of Internet email is connectivity. This has problematic consequences for email identification mechanisms, including:

- The sender's email address, specified in the FROM: email message header field [37], may be unrelated and unknown to the sending domain, which should be specified in the HELO/EHLO SMTP command [36].

[5] Bounced email messages are often referred to as *Delivery Status Notifications* (DSN).

- In several important, common cases, email messages are processed and by *resending agents*, on behalf of the originator or the recipient. This happens, in particular, by recipient forwarding services, e.g. from fixed email e.g. ALICEACM.ORG or current email e.g. ALICEWONDERLAND.COM. Mailing lists are another important example. Such services may add or change header fields identifying the source of the message, including FROM, SENDER, MAIL FROM and RESENT-FROM.
- Mail Transfer Agents (MTAs) are allowed to add and modify headers and even the message body, in certain ways. The main original motivation for changing the message was interoperability across platforms, but there are other motivations, e.g. to add indication of anti-virus filter.
- Incoming Mail Servers (MTAs) are willing to receive mail from unknown domains.

In the first part of this section, consisting of four subsections, we discuss different mechanisms for identification and/or authentication of the entity responsible for an email message. These mechanisms can help avoid spamming senders and domains, as well as to protect against other attacks using fake ('spoofed') e-mail sender identities, such as the prevalent 'phishing' attacks [4, 20]. In the rest of this section, we discuss identity-based filtering mechanisms, that determine how to handle email from a particular sender or other responsible party (where the main options are to deliver it or to reject it).

5.1 Identification based on Email Path Security

The most natural approach to identify the sender of an email message, is by using the email address fields in the message. It is especially tempting to use the FROM field, which identifies the original sender, and which is normally the only sender identification presented to the receiving user. However, as noted above, using the email sender identification is problematic, mainly since spammers often use incorrect, unauthorized and misleading sender email addresses.

In spite of this problem, several spam control mechanisms identify the email senders, using email addresses. Some of these mechanisms rely on cryptographic authentication; we discuss these in subsection 5.4 below. In this subsection, we review other identification methods based on email address, whose security is based on email path security - namely, the difficulty for an attacker (spammer) of *intercepting* an email message sent to an email address not belonging to the attacker.

Identification by sender email address The most basic mechanism is identification using the sender email address, as specified in the respective header field (mainly, FROM and/or SENDER). As explained in Section 1, attackers can use spoofed (fake) email sender address. Yet, this mechanism is valuable only when the attackers are unlikely to know or to guess an acceptable sender email address. For example, such systems may allow only email from senders whose addresses are included in the user's email address book, in the hope that a spammer is unlikely to use these specific addresses. Of course, this assumes email path security, in particular, that the attacker is unable to view messages sent to or from the user, otherwise the attacker can put addresses from these messages in his spam sent to the user.

The level of security of such schemes is not very high; in particular, they often fail since some correspondents machine become zombies, i.e. controlled by an attacker, and then the attacker may use email addresses from address books and email messages exposed on the zombie machine. To reduce the chances of detection of the zombie machine, the attackers will often send these spoofed messages from another email address found in the messages, rather than using the zombie's email address.

Another problem of these schemes is their handling of email from new correspondents; these cannot be identified. We next discuss a mechanisms that extends sender email identification,

Challenge-response validation In challenge/response schemes, the recipient of 'suspected' email returns the message to the sender, with a brief message asking her to resend the message, with precise instructions. The recipient may include some indicator in the response, to detect it is a valid response, and therefore deliver the message; this may be as simple as a special e-mail address or a (pseudo)random identifier attached by the recipient. Some systems will use challenge-response only for initial interaction, and later rely on identification using the sender email address and/or a unique recipient's email address.

The goal of challenge-response systems is to validate that the email was sent with a valid email account for receiving 'bounces' (email error messages), normally specified in the MAIL FROM: SMTP header line. Many spammers use invalid bounce and source addresses, so they may not be able to respond.

Of course, this technique may not help if the spammer uses a valid email account. Indeed, unfortunately, as explained in Section 2, spam is often sent from machines controlled illegitimately by the spammer (zombies). Challenge-response will block a spammer sending email from a zombie computer, if the zombie does not respond to the incoming challenge correctly. However, if the response to the challenge is trivial, e.g. simply using 'reply', then it is easy for the spammer to program the zombie to automatically issue the response; therefore once such a scheme becomes widely deployed, it may become ineffective. Therefore, some of the challenge-response proposals involve a 'Human Interaction Proof (HIP) challenge', which is designed to require interactive effort by a real person to provide correct response; we discuss HIP challenges in more detail in Section 6.1.

Challenge-response systems are of the most controversial spam controls. Many opponents claim that they cause unacceptable annoyance to the sender of the email, who may fail to re-send the message. Another frequent criticism, especially against HIP challenges, is that a spammer may either send spam disguised as a challenge, or use HIP challenge to force users to solve HIP for the spammer, and use this for spamming others. Indeed, some challenge-response opponents even suggest that the sender's anti-spam mechanisms may, or even should, automatically discard any such challenges; if this attitude is in fact implemented widely, this may be a fatal blow to challenge-response systems.

To avoid annoying senders, some of the challenge-response proposals include a sender MUA enhancement that responds automatically, for convenience; however this may be abused (e.g. in zombie), and may yet be unacceptable to some senders not wishing to install such a tool or to retransmit. Finally, challenge-response systems introduce additional latency and overhead (retransmissions, synchronization of address books, change of address, etc.).

Unlisted recipient email address In the simple version of 'unlisted recipient email address', the recipient gives her email only to known, trusted peers, and never publishes it, hoping that it will not be known - or used - by spammers. This is a very common, low-tech solution. It is often combined with generation of 'disposable' alternate email addresses, used in less-private forums (and disposed of, when abused by spammers).

Unfortunately, this simple version is not very secure. First, attackers can find a surprisingly-large fraction of the user names by simply scanning the target mail server, using large lists of common user names; this is often called a *dictionary attack*. Second, the e-mail can be exposed when sending the same message to multiple recipients, especially if copying a mailing list.

In a more advanced version, the recipient uses a unique unlisted address per each (known) sender. This technique combines the 'return address validation' and the basic 'unlisted email address' techniques described above; the goal of this extension is to perform 'return address validation' only on the first email received from any particular mail sender (to reduce the annoyance to the sender). To facilitate this, the recipient sends, in his reply to the sender, a special email address, which also reaches the recipient, but contains also a unique identifier allowing the recipient to identify this particular sender. The special email address may be generated automatically, possibly using a cryptographic mechanism as described in [14, 16].

5.2 Path-based Authentication of IP-Address

Internet email is sent using the Simple Mail Transfer Protocol (SMTP), as described in Section 1. SMTP is interactive; the sending main agent must respond properly to messages from the receiving mail agent in order to transfer the messages. Furthermore, SMTP is connection-based, and communicates by sending messages over TCP - and TCP already requires the sender to respond to packets from the recipient. TCP sends packets using the IP (Internet Protocol), which uses an IP address field in each packet to route it to its destination. Therefore, we can identify sending mail agents using their IP address, in one of two ways:

– Immediately at the receiving mail agent, who is using the sending mail agent's IP address, or

– Later on, using the 'tracing information' included in the email header (in the RECEIVED FROM: fields, which include the IP address of each email agent involved in forwarding the message).

Identification of the IP address is not secure against Man In The Middle (MITM) adversaries, which are able to receive and respond to packets sent with an IP destination of a victim, different the adversary's IP address. However, many spammers are not able to intercept packets sent to a different IP address; IP address identification works well in this case.

Notice, furthermore, that the IP address of the sending mail agent is not necessarily related to the email source addresses specified in the email message (in the FROM: and /or SENDER: fields, or to the address used to send bounces MAIL FROM: fields).

5.3 Path-based Authentication of Domain Name

Internet email allows a recipient, e.g. MTA B, to receives email from a server in another domain, e.g. MTA A, even when the specified sender email address, and/or the address specified to report errors (bounce address), belong to a third domain, e.g. WONDERLAND.NET. For example, the incoming border MTA B in Figure 1, may accept messages from MTA A, e.g. ALICE@WONDERLAND.NET. Such email may be legitimate, e.g. if Alice has an email account at domain WONDERLAND.NET. Normally, both Alice and the WONDERLAND.NET domain administrator are happy with this scenario, since it makes it easier for Alice to use ALICE@WONDERLAND.NET as her primary email account.

However, in many cases, a domain, e.g. WONDERLAND.NET, may not want other domains to issue mail with a sender or bounce address in the domain, e.g. ALICE@WONDERLAND.NET. In particular, by preventing the use of the address ALICE@WONDERLAND.NET as bounce address, the domain avoids receiving bounces to spam which used this address (illegitimately). Similarly, a bank may not want other domains to be able to issue 'phishing email', specifying without authorization a sender email in the bank's domain, e.g. SUPPORT@BANK.COM.

In this section, we discuss mechanisms that allow an administrator of a domain, e.g. WONDERLAND.NET, to specify an *email authorization policy*, specifying which sending, outgoing border MTA is authorized to send email which specifies sender and/or bounce address in the domain WONDERLAND.NET. This can help prevent email spoofing, by establishing that the mail server sending the message is authorized to send mail for senders from the specified source domain. In this section we focus on *path-based email authorization mechanisms*, where the outgoing MTA is specified by its IP address, relying on the difficulty for a (non-MITM) attacker to send email with a fake IP address. Path-based authorization mechanisms are very simple to implement, and require minimal computing and communication overhead.

There have been several proposals of path-based authorization mechanisms, with significant adoption by a significant fraction of email domains, and extensive efforts to consolidate proposals and reach a standard; some of the early proposals are [8], followed by [5], proposed by Microsoft. The most important proposals currently promoted are the Sender Policy Framework (SPF) [50], and Sender-ID [24], which is driven by Microsoft. See a comparison by the Messaging Anti-Abuse Working Group [25].

A main component of path-based authorization mechanisms, which is essentially identical in the two main specifications [50, 24], is the *email policy record*. The policy record of a domain contains list of IP addresses (or IP address ranges), allowed to send e-mail which uses an address of the domain (as responsible for the message). In both specifications, the policy record is distributed as a record in the Domain Name System (DNS), chosen as a convenient and efficient distribution mechanism. Notice that currently, DNS is not secure against Man In The Middle (MITM) attackers, who can cause incorrect DNS replies; this may allow such an adversary to cause false positive and false negative failures for policy-based authorization mechanisms, based on retrieving policy from DNS. In principle, policy records could be signed to prevent this threat; in reality, this is usually considered an overkill for the limited goals of policy-based authorization mechanisms.

At the very end of the policy records, the domain provides information regarding the completion of the list of IP addresses authorized to send mail for the domain. The main options are:

-all The list is complete, and *only* servers whose IP address appear on the list are allowed to send email on behalf of the domain.

all The list is probably complete, but not with absolute certainty; there may be some servers allows to send email on behalf of the domain, whose IP address was forgotten and not included in the list.

+all The list is incomplete, and there are probably one or more servers, allowed to send email on behalf of the domain, which are not on this list.

The main differences between the SPF proposal and the Sender ID proposal are in the identification of the domain responsible to a given email, and in the handling of out-of-domain mail services such as forwarding and mailing lists.

The identification of the domain responsible for a message is complex, due to the use of several relevant email message header fields, identifying the sender and other parties relevant to the transmission, including:

FROM**:** identifying the originator.

SENDER: identifying an agent who sent the mail on behalf of the originator (if not the same as the originator).

MAIL FROM**:** identifying the 'bounce to' address (to which bounce reports are sent).

HELO/EHLO **identity:** this is content of the SMTP HELO or EHLO commands, which the standard specifies to be the domain name of the sending SMTP agent (the 'SMTP client'). Note that this may differ from the domain of the sender, e.g. when an ISP A.COM is transfering email of customer ALICE@WONDERLAND.NET.

Both SPF and Sender ID allow identification and authorization using the MAIL FROM: address, identifying the 'bounce-to' address. This works as follows. Suppose incoming border MTA, say MTA B, receives email from MTA A, in another domain, with MAIL FROM: ALICE@WONDERLAND.NET. Then MTA B looks up the email policy record of WONDERLAND.NET, typically by an appropriate query to the Domain Name System (DNS). If such a policy record exists, then MTA B checks if the IP address of MTA A appears in the list of authorized mail senders for WONDERLAND.NET. If it does not, and the policy record ends with -ALL, then the email is rejected. The idea is that such mail, with unauthorized MAIL FROM: field, is probably unrelated to ALICE@WONDERLAND.NET - viruses and spammers often abuse sender email addresses. If MTA B transfers the email, and this email is eventually bounced (which is likely), then ALICE@WONDERLAND.NET will receive this bounce of a message she never sent. Namely, publishing the policy record allows WONDERLAND.NET to avoid receiving such 'fake bounces'.

This method works well when legitimate email passes directly from the sender domain to the recipient domain. However, this is not always the case; there are many cases where email has to pass through an intermediate domain, including:

1. Mail sent by ALICE@WONDERLAND.NET, while visiting domain A.COM.
2. Mail forwarded, e.g. from Bob's university account, BOBS@BIU.AC.IL, to his current account, BOBSPONGE.COM.
3. Mail distributed by a mailing list.

In most of these cases, except some mailing lists, the intermediary mail agent do *not* change the MAIL FROM address. This may cause such mail to be rejected by the receiving MTA B. The Sender-ID specification stipulates that intermediate domains *should* change the MAIL FROM field, to an address under its control. Should that address eventually receive a 'bounce' of the original message, that 'bounce' should be forwarded to the original MAIL FROM address. This altered address should refuse to receive any other messages (except bounces of the original message). We are not aware of any intermediary mail agents that implement these specifications at this time.

Sender ID also allows identification and authorization using the *Purported Responsible Address (PRA)*. The PRA is not an email header field; instead, it is the result of a simple algorithm applied to an email header, whose goal is to identify the entity responsible for this email: the original sender, or an email agent such as forwarding service. The PRA algorithm is specified in [26]. The idea is that the PRA is, as the name suggests, the party responsible for the email - in particular, for it not to be spam. For this to be meaningful, this information should be exposed clearly to the end user.

However, currently most email clients only display the FROM: or SENDER: fields, which are not validated by Sender ID (or SPF). Sender ID specifies that email clients *should* display the PRA, and considering the Microsoft is the main proponent of Sender ID, this may be implemented in future Microsoft email clients. It remains to be seen whether this will be helpful to the non-expert email recipients, or be a confusing, complicated indication that users will ignore.

An even worse possibility is that some mail clients may present messages that passed Sender-ID's PRA validation as 'authenticated', with only the value of the FROM header, even when the (validated) PRA is different; this may make it actually easier for attackers to spoof email sender identity, e.g. in phishing attacks. This threat is sometimes referred to as email laundry attack.

Like the MAIL FROM identification, the PRA identification also works well when legitimate email passes directly from the sender domain to the recipient domain, but difficulties arise when email has to pass through an intermediate domain. The Sender-ID specification stipulates that such intermediaries add a RESENT-FROM header or a SENDER header, allowing the PRA algorithm to result in the identity of the intermediary. We are not aware of any intermediary mail agents that implement these specifications at this time.

The SPF specification allows, in addition to the MAIL FROM identity, also the use of the domain identity specified in the SMTP HELO or EHLO command, which is the first command sent from SMTP the sending (client) SMTP agent to the receiving (server) SMTP agent. According to SMTP standards [36, 34], this entity should be the name of the sending domain, although not all SMTP implementation actually do this. However, when properly implemented, this is always an identifier of the domain transferring the message, be it an intermediary domain or the origin domain. However, this identifier is *never* presented to the receiving user.

The fact that a policy records exists for any of the SPF/Sender-ID identifiers (MAIL FROM, PRA and HELO/EHLO), and the inclusion of the IP address of the sending MTA in this record, does not necessarily imply that this message is 'good' (e.g. non-spam or not spoofed). After all, a spammer (or other attacker) can set up the policy records for his own domain. Furthermore, setting up new domains may be easier than changing IP addresses, therefore blacklisting spamming domains is of limited value.

However, well behaved, non-spamming domains usually use long-lived domain names. Therefore, domain names, possibly validated using the policy record against the IP address of the sender, can be used as an identifier for validating (positive) ratings of a domain. In subsection 5.6 below we discuss such 'positive' rating mechanisms, by reputation and accreditation services.

5.4 Cryptographic Email Authentication

Some of the most important spam controls, are based on cryptographic authentication of senders and/or of sending domains. Cryptographic message authentication mechanisms include two functions: $t = auth_{k_a}(m)$ receives key k_a and message m and outputs *tag* t, and $validate_{k_v}(m, t)$ receives key k_v (identical or related to k_a), message m and tag t and outputs *true* if and only if $t = auth_{k_a}(m)$. There are two families of cryptographic message authentication mechanisms we can use:

Message Authentication Code (MAC) schemes, e.g. HMAC [23], use shared secret keys, i.e. $k_a = k_v$ above. Most MAC schemes are quite efficient, and

therefore their overhead, even if applied to entire email traffic, would be reasonable for servers and negligible for most end-users. However, MAC schemes require the sender and recipients to share a secret key, kept hidden from the adversaries. When this is not a viable option, e.g. when the sender does not have a prior, long-term relationship with the recipient, or when there are many recipients (e.g. mailing-list), then the other cryptographic authentication mechanism, digital signatures, may be more appropriate.
Indeed, most message authentication proposals for spam control use digital signatures, with few exceptions such as the Signed Envelope Sender (SES) protocol [40]. SES uses MAC schemes (referring to it as 'digest signature') to validate the message, when returned by the receiving (incomign) MTA to the *sending (and validating) outgoing MTA*. The recipient trust the validation response, since it receives it as an answer to a query to a known IP address, i.e. based on the difficulty of intercepting packets sent to a server's IP address, and not based on cryptography.

Digital signature schemes, e.g. RSA [44], use a private key k_a to sign (authenticate) a message before sending it, and a public key, k_v, known to all recipients, to validate the identity of the sender. Digital signature algorithms are often computationally intensive, i.e. can create substantial overhead. However, the same signature can be validated by multiple recipients, which could be significant for email messages with many recipients; furthermore, there is no need in pre-established shared secret key between sender and recipient. Most standard signature mechanisms can be validated off-line, i.e. not only upon message receipt (when there may be interaction with the signer), but also afterward, and possibly by a third party (not the recipient).

Using cryptographic authentication mechanisms to identify the party responsible for spam may appear trivial: require senders to digitally sign the email. However, a deeper reflection reveals several challenges, including:

– Legitimate, minor changes to email message on its transit may render the signature invalid.
– Recipients need to know and trust the signer's public key. A public key certificate (PKC) signed by a trusted Certificate Authority (CA) is a possible solution, but agreeing on trusted certificate authorities, and issuing certificates to senders or their MTAs, are difficult and time consuming tasks.
– What identifier can we use for the signer? The public key used to validate the signature is a secure identifier, but this is not meaningful for the recipient. Using the sender's email address is meaningful for the recipient, but has its own problems: there are several candidate fields in the email headers (mainly FROM:, SENDER: and MAIL FROM:, and some mail agents may change these fields, e.g. a mailing list agent often changes FROM and/or SENDER. Furthermore, the signer should be the party responsible for the message; is this always the original sender, or can/should other entities be the responsible parties, e.g. a mailing list server?

The basic idea for using cryptographic message authentication to control spam is to authenticate the party responsible for each email message m (typically, the sender). This is done by attaching to m an authentication tag $authk_a(m)$. The intuitive approach is to directly attach the tag to the message body, with some separator; this is done e.g. by (early versions of) PGP [51]. This requires recipient's Mail User Agent (MUA) to support this tagging mechanism, namely validate it and remove it from the message displayed to the user.

S/MIME [35] takes a more standard approach, and sends the tag as a distinct MIME part [31]; this is easy to process by S/MIME-enabled recipients and to ignore by MIME-enabled recipients who do not support S/MIME. However, this may create difficulties if the validation is done by a mail server (MTA/MDA) or some clients, where MIME may be unavailable or cause significant overhead. S/MIME is also insensitive to most email headers, which are often very relevant to spam.

Therefore, several recently proposed email message authentication mechanisms, send the tag in header fields, which are typically not shown to end-users by MUAs. The most important proposal being developed by the Intenet Engineering Task Force (IETF) is called *Domain Keys Identified Mail (DKIM)* [3]; it is a merger of DomainKeys [13] and Identified Internet Mail (IIM) [22]. DKIM adds a new, special header field, DKIM-SIGNATURE. Other proposals, e.g. SES [40], use the existing email header fields, e.g. in SES the MAIL FROM: field of the email 'envelop', appended by each mail agent through which the message flows (and used to identify the return-path).

A challenge for cryptographic email message authentication, is the *mail mangling* often performed, for different reasons, by mail servers. Namely, mail servers (usually MTAs) may perform certain modifications on messages, e.g. insertion of 'white space' such as $< cr >< lf >$ to break long lines. Furthermore, sometimes mail contents is changed beyond simple formatting, e.g. by adding prefix and/or suffix text ('this message was scanned by...'). Finally, mail agents often add header fields (e.g. RECEIVED:), and may also remove header fields (e.g. for efficiency or to hide sensitive route information), modify them or reorder them; as a result, solutions that sign all or some of the email message header fields, are more sensitive to mangling than solutions that sign only the message body. Cryptographic authentication is sensitive to *any* change in the message, and even if $m' \in mangle(m)$ is only a 'minor change' to m such as adding a $< cr >< lf >$, still $validate_{k_v}(m', auth_{k_a}(m))$ will return $false$.

The solution to this consists of two mechanisms: *canonicalization* and *limited scope*. By limiting the scope of the signed message parts, we avoid parts which we consider non-essential and modifiable. Namely, many email signing standards apply the authentication only to specific parts of the message, e.g. defined as $scope(m)$. For example, S/MIME [35] and PGP [51] scope includes only the message body, while DKIM and SES allow the signer (sending MUA/MSA/MTA) to extend the scope to include arbitrary header fields. DKIM also that allows specifying the number of bytes in the body of the email that are included in the computation of the tag, using the L= attribugte. When the L=l is included, any byte after the first l is not authenticated, allowing intermediate agents to append trailers such as added by some mailing lists and anti-viruses, but potentially also an ad added by a spamming intermediary.

Canonicalization deals with possible modification (mangling) of the signed message parts by different mail agents along the path, potentially invalidating the signature. For most messages and signers, some common modifications as white- space replacement and wrapping of long header lines do not materially change the message, and can be permitted. Such senders may use the NOWSP canonicalization function [3]. In other cases, any modification may not be acceptable, in which case the default, null canonicalization algorithm is appropriate; DKIM refers to this as the SIMPLE canonicalization algorithm. Other canonicalization functions can be defined, see [35, 31, 3].

The canonicalization process is as follows. The authenticating mail agent applies a *canonicalization function cannon* to $scope(m)$ and authenticates (signs) the result. The canonicalization function has the property of returning the same value for every mangled version of $scope(m)$, namely for every message m' such that $scope(m') \in mangle(scope(m))$ holds $cannon(scope(m')) = cannon(scope(m))$. Yet, *cannon* returns a different value for every 'really different' message m'' i.e. when $scope(m'') \notin mangle(scope(m))$, i.e. $cannon(scope(m'')) \neq cannon(scope(m))$.

Cryptographic message authentication can be applied at several stages in the email transmission process:

Authenticated email submission and transfer ('hop by hop'): many MSA require mail submissions to be authenticated, often cryptographically, at least when receiving submission from outside the administrative domain. A MTA may also authenticate messages received from the sending MSA/MTA, to foil strong adversaries such as a MITM (e.g. using network sniffing device). This authentication is 'hop by hop', involving only two mail agents involved in a single SMTP connection (over TCP); therefore, we can use various authentication protocols, such as TCP MD5 extension, IP-Sec AH or ESP, SSL/TLS, and others. Since authentication is performed here before processing by the mail server, mangling is not involved and canonicalization is not necessary.

Authenticated email content and sender ('end to end'): The other extreme approach to authentication, performs message authentication at the sender MUA (Alice's), and validation at the receiving MUA (Bob's). Sender authentication could be an effective mechanism for email recipient to identify spam. In this scenario, the recipient knows the email senders, and trusts the sender not to send spam; furthermore, both uses cryptographic-enhanced email services. Specifically, the sender MUA authenticates, using MAC or signature, the email sent, and the recipient MUA validates this. This solution provides excellent protection against email spoofing, and therefore also against spamming.

Several standards and systems offer source authentication mechanisms for email messages using digital signatures and public keys, e.g. S/MIME [35] and PGP [51]. These solutions also provide message encryption, which is irrelevant. We believe that currently, a majority of email clients support

at least one of these solutions. Integrating such authentication mechanisms with anti-spam controls should be straightforward, when there is a long-term relationship between sender and recipient (and therefore the public keys are known, and the recipient trusts the sender not to send spam).
However, only few email users, including corporations, use these or other cryptographic mechanisms to authenticate their outgoing email messages. There are different explanations to this important, problematic situation:

1. Both S/MIME and PGP embed the signatures as parts of the mail message, following the Multimedia Internet Mail Extensions (MIME) format [31]. As a result, email messages authenticated S/MIME and PGP will be unreadable and confusing to recipients using older mail clients, which are not MIME enabled. However, note that a substantial and growing majority of MUAs are MIME-enabled.
2. S/MIME depents on Public Key Infrastructure, specifically on senders using identity public key certificates obtained from Certificate Authorities (CAs). This situation exists also for web sites, but there it is solved by having, in each browser, the list of public keys of all trusted CAs; and each protected web site obtains certificate from one of these CAs. This may be more complex to deploy for email, since there may be a need to certify keys for each employee.
3. As mentioned above, digitally signing a message implies that its authenticity and origin may be proven later on to any party which has the appropriate public key. Some corporations and individuals may be reluctant to leave such undeniable proof of sending messages, which may be used in litigations. Notice that this is not a problem when using MAC, as well as special signature schemes which do not provide long-lasting proofs.

Authenticated email domain: An alternative is to perform email message authentication, but only authenticating the *domain* from which the message was sent, and not the particular sender. This can assure the receiving border MTA, or other mail agents (possibly the recipient's MUA or the MDA), that the message was received from that specific domain. Since the message authenticity is protected cryptographically, the validation ensures that the message was not modified en route. This protects against possibly malicious modification (e.g. phishing attack) or addition of an 'advertisement trailer', but requires re-authentication also after minor text processing such as by mailing lists or virus scanners, unless excluded by scope restrictions or canonicalization function (see above).
While there is not yet a standard mechanism for email domain authentication, the DKIM [3] proposal seem to have good chances, with strong support and multiple implementations. DKIM requires that the signing domain be the same, or a parent domain, of the domain to which the sender email address belongs. This is the right-hand side of the address in the FROM: or SENDER: field; for example, WONDERLAND.NET can be the signing domain for email with FROM: ALICE@MAIL.WONDERLAND.NET. We note that this prevents domains from signing messages for customers whose sending email

address is in a different domain; unfortunately, this 'breaks' common email scenarios permitted by existing email standards [36, 34]; see Section 1.

Currently, DKIM specifies that recipients (verifiers) should retrieve the validation key of the sending domain by an approriate query to the Domain Name System (DNS), somewhat similar to the queries used by SPF and sender-ID [50, 24], see subsection 5.3. The motivation for this is simplicity - the DNS system already provides some level of control for each domain over its records - and efficiency of DNS query mechanisms. However, the DNS system as currently deployed is not very secure, and therefore DNS replies may be forged by some attacks . If this is a concern, more secure mechanisms may be used, such as retrieving a public key certificate from a Certificate Authority (CA) trusted by the recipient. Such mechanisms were part of the Identified Internet Mail (IIM) proposal which was a predecessor of DKIM, see [22].

Sender or domain authentication establishes the identity of the entity (user/domain) responsible for an incoming message. Using this identification, we can apply different policies and mechanisms to determine if the message should be delivered or rejected, as we explain in the following subsections.

5.5 Blocking Spammers using Blacklists

In the previous subsections, we presented techniques for identifying the source of the email or a party responsible for an email message. In this subsection and the reminder of this section, we assume some identification mechanism was performed. We describe different identity-based filtering mechanisms, for determining whether to receive or reject email, based on this identification of the sender or of the party responsible for the message. Specifically, in this subsection we focus on the most basic identity-based filtering mechanisms, based on blacklists (also called block lists) of 'bad' domains and/or senders.

Sender's blacklists: Senders may maintain 'blacklists' containing email addresses of known spammers. However, such blacklists are of limited value, since many spammers frequently change the declared source email address in their email (in the FROM: header). Spammers can change the source email address either by using a false source address, or by opening new email accounts in different free services. Indeed, to limit this abuse of free email services, most providers now require new users to perform a Human Interaction Proof (HIP) upon opening a new account, in the hope that this is not easy to automate; see more details in subsection 6.1.

Sender's blacklists are mostly effective to filter out email from few credible organizations, that nevertheless send unsolicited mail. We next discuss the more useful technique of domain blacklists.

Domain Blacklists: in this technique, the email recipient or her mail server compares the address of the sending mail domain, against blacklists[6] of spamming-related email domains.

Usually, the receiving Mail Transfer Agent (MTA) compares the IP address of the sending MTA, against the blocks of blacklisted IP addresses of suspected-spamming domains. This is more reliable than blacklisting a domain name, since spammers can open new domains with very little cost, often using stolen credit card numbers. Using IP version 4, whose address space is scarce, it is difficult and expensive for spammers to repeatedly change IP addresses. Notice that this situation may change with the adoption of IP version 6, where address space may not become freely available.

Some blacklists operate on the basis of *domain names* rather than IP addresses. However, the cost for registering a new domain name is small (few dollars), therefore domain names are hardly suitable for blacklisting. An exception is when a domain become valuable, by gaining reputation and credentials, as discussed in the next subsection. In this case, blacklisting can be effective, since the reputation and credential mechanisms have made the identifier (domain name) valuable. Another exception where the use of domain name blacklists is appropriate, is the blacklisting of domain names of established mail service providers (e.g. hotmail, gmail) and Internet Service Providers (e.g. aol.com); of course, most ISPs will rarely blacklist one of these large domains.

Many blacklists use the Domain Name System (DNS) to distribute records; namely, lists identifying 'spamming domains' are entered as textual records in the DNS. The use of DNS as a distribution mechanism is merely for performance and implementation simplicity considerations; other distribution mechanisms may also be used.

Clearly, the users of a blacklist rely on its contents to be correct, avoiding false positives (unjustifiably blacklisted domains) as well as false negatives (spamming domains which escape the RBL). However, maintaining a correct blacklist is not easy, and requires considerable efforts and resources. In particular, attackers can send fake complaints and use other techniques to 'poison' a blacklist and enter a victim, non-spamming server into it, with the goals of harming the victim server and/or of reducing confidence in the blacklist service; and blacklist operators may act negligently or even intentionally permit false positives or negatives, for commercial, ideological or other irrelevant reasons. Indeed, blacklist services may differ substantially in their quality (false positives and negatives rates). Furthermore, different blacklists may use very different criteria, e.g.:

- List only open relays
- List IP addresses which were used to send spam
- List IP address blocks which belong to known spammers or which were used as a destination for replies to spam
- List blocks of dial-up IP addresses (which are not expected to run a mail server)

[6] These types of blacklists are often referred to as block-lists and as Realtime Blackhole Lists (RBL).

– Combination of the above

There are few websites comparing blacklists, however, so far there is no standard, reliable mechanism to compare and grade blacklists.

As mentioned before, most of spam is sent from zombie (broken-into) machines. As a result, domains cannot fully avoid issuing spam; this makes the determination of a spamming domain more difficult and open to interpretations. Of course, domains take different measures to limit the amount of spamming by a zombie, including defining quotas on outgoing email per user, applying content filtering to the outgoing email, and preventing end-user machines from directly sending mail to other domains (port 25 blocking). These steps allow most Internet Service Providers to avoid being added to most blacklists.

However, blacklisting is not a precise discipline; there are many different lists, with different motivation and compensation mechanisms for their operators, different listing criteria, different mechanisms for removing a domain in case of error or when spamming ceases, and different levels of integrity.

Furthermore, some blacklists identify very large IP blocks as suspect spam, even to the extent of blocking entire countries or large areas. This may harm the provision of Internet services in these areas, causing collateral damage to non-spamming users. We present more refined filtering means in the next subsection.

5.6 Filtering based on White-lists, Reputation and Accreditation Services

In the first part of this section, we discussed several mechanisms for identifying a sender, a sending domain/MTA or a resending agent taking responsibility for an email message. The identifications resulted in one of the following identifiers: an email address, an IP address, a domain name, or a public key. In this subsection, as in the previous one, we discuss how to filter email based on the identification of its source, or of a party responsible for its transmission.

In the previous subsection we discussed blacklisting of suspected spammers, which depends on difficulty for spammers of obtaining new identifiers. Blacklists are problematic, mainly since spammers can obtain new identifiers with very limited expense; IP addresses may be the only identifier with significant price, and this is only due to the current shortage of IP addresses, which may be removed if and when IP version 6 is adopted.

Therefore, in this subsection, we explore the remaining alternative: whitelists, or more generally, reputation and accreditation services. In this subsection we discuss solutions based on identifying credible, trustworthy senders, and allowing email from them. We begin with simple white-lists of 'good' senders, and then discuss more advanced reputation and accreditation mechanisms.

Sender's white-lists: In this technique, the recipient maintains a list of legitimate email sender addresses. Email from these senders is accepted (without additional filtering), while email from other senders is discarded or filtered. Email recipient often use this technique to minimize the risk of false positives for email, when using content filtering. However, this technique fails whenever the sender changes her email account. Also, since viruses are used by spammers to collect legitimate email addresses, often from address books, this technique could fail if the virus found the correct addresses (e.g. in address book of a joint friend).

The simplest whitelists are these maintained and used by a single entity, e.g. Bob's MUA or his mail server. Namely, if Bob communicates (frequently) with Alice using ALICE@WONDERLAND.COM, or adds ALICE@WONDERLAND.COM to his email address book, then the filter allows email from ALICE@WONDERLAND.COM. The filter will apply content filtering, blacklisting and other techniques to email from other sources. Indeed, most chat systems require explicit recipient authorization before transferring messages to the recipient; this is a 'whitelist only' filtering mechanism.

Whitelisting by sender name only is subject to spoofing (fake name) attacks. For better security, Bob could have used Alice's public key as an identifier, if Alice signed her messages to Bob. However, in both cases, this requires Alice to be known to Bob in advance; in fact, Bob should *trust* that Alice is not a spammer.

Reputation services allow recipients to establish trust, and accept communication, from unknown, but reputable, senders. Senders, or any entity taking responsibility for an email message, provide a *proof of reputation* from one or more reputation service providers, more commonly referred to as *rating services*; of course, recipients should trust at least one of the reputation services to establish trust.

Proofs of reputation are usually provided by a digital signature by the reputation service over a *Reputation Record RR*, namely $Sign_{RS.s}(RR)$ where $Sign$ is the signature algorithm and $RS.s$ is the (secret) signature key of the rating service RS. Reputation systems may also use other methods to allow recipients to validate the reputation record, as alternatives to digitally signing it, such as a direct, potentially protected, query from recipients, or distributing the reputation records via the Domain Name System (DNS).

The reputation record RR contains identifier(s) for the sender, such as its public key, email address and/or IP address, validity period for the record, and reputation data such as rank (e.g. *highly trusted non-spammer*) and historical performance (e.g. *no spam since ¡date¿*). When the public key is included in a digitally-signed reputation record, it essentially becomes a certificate, and standard certificate format such as X.509 may be used. A public key in the reputation record allows recipients to validate that the message was not modified in transit, and potentially to present proof of spam to the reputation service and/or to other parties such as blacklists; these techniques are extended by accreditation services, described below.

Reputation services, like the simpler blacklist services, are based on trust by recipients. Recipients must allow for some spamming even from a sender with good reputation data; this may be a spammer who built reputation in order to be trusted, or a victim whose machine was taken over by a spammer.

Some reputation services may charge a fee from the listed sending entities. Unfortunately, this may create a potential financial incentive for senders (mail servers) to be permissive and allow (some of) their customers to actually send spam or other illegitimate mail, and it could be impractical or at least difficult, for mail recipients to detect this or 'punish' such mail source servers.

A related risk to reputation services is that of false positives, i.e. unjustified loss of reputation due to intentionally-misleading, 'bad-mouthing' input from attackers. Therefore, reputation systems should be complemented with an efficient, automated process to handle complaints, based on a clear definition of violations (spam). We describe such a mechanism in section 6.3.

Accreditation Services for High-Risk Senders Reputation services work well, provided that the sender has the required reputation... However, reputation (as a good sender, non-spammer) is typically developed over a long period of (successful, non-spamming) usage. *Accreditation service* allow sending of messages even by senders without good reputation, typically new senders, by using monetary penalties to make it unprofitable to send spam.

Accreditation services operate rather similarly to reputation services described above, and are usually based on an *accreditation certificate* from one or more service providers. Accreditation certificates extend the reputation records presented earlier. Accreditation certificates are usually provided by a digital signature by the accreditation service over an *Accreditation Record AR*, namely $Sign_{AS.s}(AR)$ where *Sign* is the signature algorithm and $AS.s$ is the (secret) signature key of the accreditation Services. The accreditation record may be identical to the reputation record RR presented earlier, but may also contain indication of the amount to be forfeited if the message is spam, and method of reporting or 'proving' spam.

Furthermore, to prevent a rogue sender from using the same accreditation certificate to send a large amount of spam to multiple recipients, accreditation certificates are usually issued *per recipient*, and specify the recipient identity (typically, by specifying the recipient's email address).

A high-level overview of the operation of an accreditation service is presented in Figure 5.6. In the first step, the sender requests the accreditation certificate for the specific recipient from the accreditation service. After receiving it, the sender sends the email message to the recipient, using standard email submission and transfer process, together with the accreditation certificate. Typically, the certificate is encoded as header field e.g. using DKIM, as as described in subsection 5.4. The certificate may contain a hash of the specific email message, but more commonly it will contain the public key of the sender and a maximal number of messages permitted to send to this recipient, and the sender will append also her own signature over the message and over a spam-indication label (e.g. 'ADV' or 'non-ADV').

When the recipient's MUA receives the message together with the accreditation certificate (and optionally signed by the sender), it will first check the spam label against the filtering rules of the recipient, and check the the validation ser-

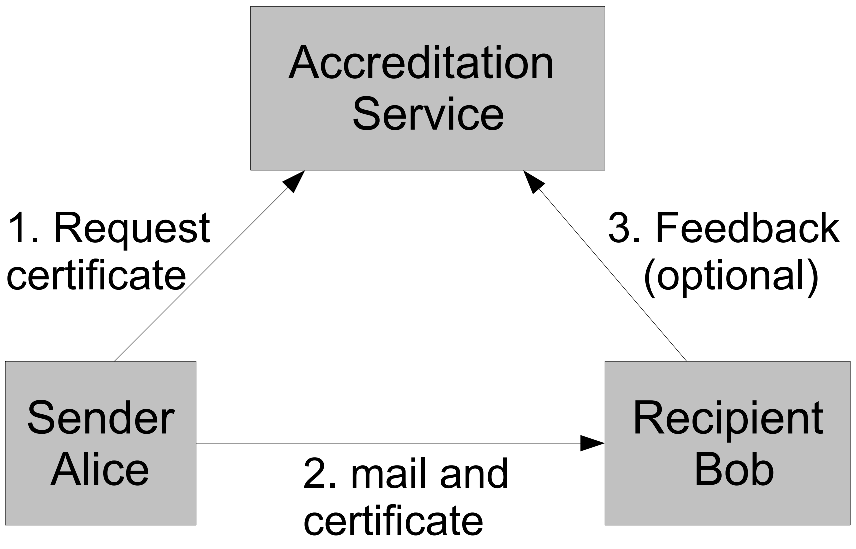

Fig. 4. High-level overview of the operation of an accreditation service. The accreditation service may issue certificate per specific recipient, or per time period.

vice listed is one of these approved by the end user. The MUA may next validate that the sender is not included in a locally-kept blacklist of spamming senders; the MUA will add senders to this list upon receiving spam from them, thereby allowing (non-spamming) senders to use the same accreditation certificate to send multiple messages to the same recipient. Next, the MUA next validates the digital signatures, and if these are valid, displays the message to the user. Based on the user's response, the MUA may deduct that the message was in fact spam or non-spam, upon which it may inform the accreditation service as well as other reputation services and blacklists.

As with reputation services, there may be financial incentives for accreditation services to be negligent or intentionally to sign incorrect accreditation records. One may simply assume that the accreditation services are trustworthy; however, this assumption may not be justified, and if we are forced to make it, it may result in a very small number of very large authorities, which may not be a desirable outcome.

Here, again, one possible solution may be to use reputation systems, wherein each participant establishes some reputation based on positive and negative reports from other participants, and where the reputation is also taken into account in determining the credibility of reports. However, the design and analysis of a provably correct and efficient reputation system remains an open problem. In the next section, we present an alternate solution to the problem, based on the use of cryptographic protocols for automated complaint resolution, including guaranteed compensation for spam.

6 Cost-Based Spam Controls

Email is an inexpensive communication mechanism, which is one of the reasons spam is such a lucrative business. Therefore, one way to control spam is to exact some price, monetary or otherwise, for sending messages. Often, e.g. when the price is monetary, it can also provide compensation to the recipient and/or service provider for the costs of handling the spam.

However, introducing cost for sending email is difficult, possibly even more when the cost is by a monetary payment. In particular, users may object to pay for email, which is currently (mostly) free. This may be solved by protocols ensuring that users pay only for spam, or are refunded for paying for non-spam messages. We discuss such protocols in subsection 6.2 below.

Users resistance to paying for email may be reduced, is the 'payment' is not monetary, but involves some user effort and/or computational resources. Furthermore, monetary payments require interoperability between senders and recipients, and acceptable overhead - even for the relatively low-charges which may be acceptable for email ('micropayments'). Considerable efforts were made to develop such globally-interoperable and low-overhead micropayment schemes, see e.g. [18], but so far none of these was truly successful. All of this motivates the use of alternate, *non-monetary cost mechanisms*, which we discuss in the next subsection.

6.1 Non-Monetary Cost Mechanisms

Non-monetary cost mechanisms allow acceptable overhead without requiring global adoption. The non-monetary cost mechanisms are often also more acceptable to users, since they do not involve a financial, monetary charge. Indeed, as we will see below, users may be completely oblivious to some of the mechanisms. We next present the main categories of non-monetary cost mechanisms.

Cost of Human Effort and Human Interaction Proofs (HIP) One of the reasons that spam is so inexpensive, is the fact that it is sent 'in bulk', automatically by programs, with minimal manual work. Indeed, some define spam as 'unsolicited bulk email (UBE)' or as 'commercial bulk email (CBE)'. Therefore, one method to control spam is by ensuring that some amount of manual effort was spent for sending email, in such way that prevents sharing the same effort among multiple recipients. These spam controls force the sender to perform some task that is supposedly easy for humans, yet hard to perform by a computer program. This kind of tasks is referred to as *Human Interaction Proofs (HIP).* HIP are applicable in many situations, to ensure that some process requires manual, human effort and cannot be fully automated. Applications include ensuring effort for sending a specific email, opening a free webmail account, joining a forum or mailing-list, providing ratings and voting.

Human Interaction Proofs were first proposed in [2], using the name *CAPTCHA*, standing for , *Completely Automated Public Turing test to tell Computers and*

Humans Apart. The most familiar HIP is probably an image containing a string, but where the string is displayed in mangled form, and it seems that human pattern recognition abilities are required to identify the string. See example in Figure 6.1.

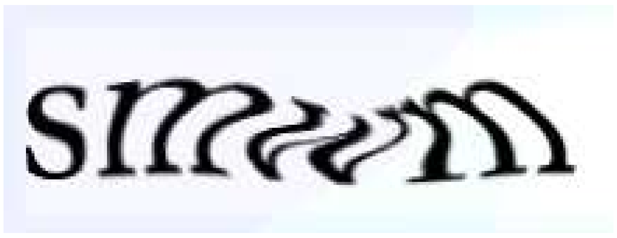

Fig. 5. Example of a Human Interaction Proof (HIP), aka CAPTCHA, from [49]. This HIP/CAPTCHA obsures the string SMWM from computer interpretation, mainly by twisting the letters.

The goal of using HIP against spam is that a spammer will have to spend considerable resources to perform these tasks since they require human attention, making spamming not (or at least considerably less) profitable. There are different proposals for CAPTCHA, usually requiring different sorts of visual recognition abilities. A possible drawback of these methods is the burden on senders, some of whom may refuse or fail to complete the task; an extreme case is the inability of blind and some other handicapped users from responding correctly to HIP. Furthermore, there is always the threat that improved programs and algorithms may enable automated solution of the HIP, without requiring human interaction at all. For example, a recent paper showed that at least for the task of identifying a single (mangled) character, a program achieved *better* results than humans, i.e. this is *not* a good HIP [7].

The protection offered by HIP is also limited by the relative ease, to an attacker, of having the HIP provided by a real human, for free or for low cost. In particular, some attackers, e.g. porn sites, may present the HIP challenge to their visitors, and use the response for the real site; see Figure 6.1; this is often referred to as a Man In The Middle (MITM) attack. Furthermore, attackers may simply hire low-paid individuals to solve the HIP.

Cost of Computation, or Proof of Computational Work This family of techniques, proposed in [12, 11, 5, 6], is a computational alternative to the 'Cost of Human Effort' techniques we described above. Again, we require a

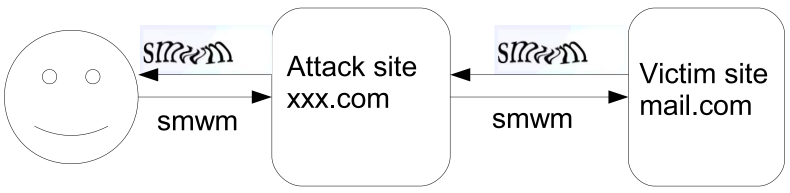

Fig. 6. A 'Man In The Middle' attack on a victim site using HIP, e.g. mail.sys, by asking visitors to another site, say xxx.com, to solve the same HIP.

proof of substantial investment of effort or some other resource, before granting services, e.g. before processing an incoming message or before opening a 'free' webmail account. Assuming that the adversary has limited resources, this limits the amount of services consumed, e.g. the number of (spam) messages sent by this adversary, or number of webmail accounts opened by the adversary (where we normally also limit the number of messages sent by each account). This prevents abusive overuse of the service. In the 'proof of computation work' techniques, the investment is in computational resources rather than in human effort.

To prevent reuse, the 'proof of work' computation should be related to a unique *challenge*, possibly selected by the service provider. The challenge may simply be an email message, including destination and date, allowing destination to prevent token reuse; or it may be a value selected by the service provider (webmail server, recipient MUA).

The core of 'proof of work' techniques is a pair of functions, $work(m, c)$ and $validate(x, m, c)$, with the following properties:

Correctness: for every m, c holds: $validate(work(m, c), m, c) = True$.

Upper-bound on work: There is a known algorithm $WORK$ to compute $work(m, c)$ with resources bounded by $R(c)$, where $R(c)$ is a monotonously increasing function. Intuitively, this implies that *work* is not too hard.

No major shortcuts: Given c together with l different values $m_1, \ldots, m_l$, there is no (known) algorithms that computes values $x_1, \ldots, x_l$ such that for $i = 1, \ldots, m$ holds: $validate(x_i, m_i, c) = True$, with resources less than $R'(l, c)$, where $R'(l, c)$ is a monotonously increasing function (typically linear in l). Intuitively, this implies that *work* can not be computed much faster than by using the known algorithm $WORK$.

Efficient validation: There is an efficient algorithm for computing *validate*.

For example, the simple Hashcash function [6] receives two inputs: a challenge (e.g. message) m and a cost parameter c. The sender must find a value a value x such that the $c - th$ least-significant bits of $h(m||x)$ are all zero. The function h is an agreed upon cryptographic hash function, such as SHA-1.

A spammer will have to spend considerable resources for performing these computations, making spamming not (or at least considerably less) profitable.

One concern with this approach is that spammers, who send their spam from zombies (penetrated computers, see Figure 2), may not care much about the computational cost, while innocent users end up wasting their computer resources (on every email sent by their computer, legitimate or spam).

Another concern is that this technique may prevent sending email from computationally limited or energy-sensitive client devices, such as some mobile devices. Some of the recent proposals for 'proof of work' functions try to address this concern. For example, in [11], the most efficient solution requires the use of large amounts of storage, which are not feasible in primary memory (RAM). As a result, the computational cost depends on the speed of access to (large but slower) secondary storage (e.g. disk), where the difference between low-end and high-end devices is less drastic.

A major problem with both cost of computation and cost of human effort, is that they are hard to bootstrap. Namely, as long as most senders do not attach payments or proofs of work, receivers will not insist on them; and as long as receivers do not insists on attached payments or proofs of work, senders have little incentive to produce them. We next describe a propsals for a 'cost mechanism' that does not have this drawback.

Cost of Retransmission ('Grey-listing') One of the reasons allowing email to be (almost) free, is the high efficiency of transferring email messages using the SMTP protocol (see Section 1). Therefore, one possible approach for introducing cost to email and in particular to spam, is to reduce this efficiency - at least for suspect spammers. Harris [17] proposed an innovative technique for introducing such cost, which has a very attractive feature: it relies only on standard SMTP implementation by the sending MTA, and therefore may be deployed by a receiving MTA without the awareness or support of the sending MTA. This technique, called *domain/MTA greylisting*, uses the fact that the SMTP standard [34] requires sending mail agents to resend, within few hours, if the transfer failed due to potentially temporary reasons (e.g. server load).

Specifically, grey-listing works as follows. When a boundary MTA receives an email from an untrusted domain or MTA, it replies with a 'transient error' indication. The receiving boundary MTA also writes to temporary storage the the details of the rejected email, and delivers it if it is received again within several hours. Standard compliant sending MTAs will re-send the message after few hours, and therefore the server eventually delivers their mail (albeit with some delay); spammers, who usually send many messages simultaneously, may fail to resend or will have to spend substantial resources. Grey-listing is effective against many of the current spam programs, but causes only limited damage to spammers which do implement retransmission. However, another advantage of greylisting is that by delaying processing of mail from unknown MTAs, it increases the likelihood that a spam message will be blocked, by hitting an updated black-listing containing the sending MTA or domain, or an updated content filter.

6.2 Postage and Penalty Protocols

In this subsection, we discuss cost-based spam controls, where spammers are forced to pay in real money to send spam, thereby reducing or eliminating the profitability and therefore the motivation to send spam.

There are two natural methods to apply payments as spam controls: to charge senders for each message, like regular postage, or to charge only for spam, as a penalty. Charging only for spam (i.e. using *penalty protocols*) is of course preferable, since we do not want to unnecessarily penalize non-spam email senders. However, penalty protocols require an agreed-upon decision mechanism to determine that a particular message is spam. A simpler alternative, which still may avoid charging non-spammers, is to charge for each message, but to cancel (refund) or at least reduce the charge, once the message is identified as non-spam; we refer to this approach as *refundable postage protocols*. We now describe each of these approaches.

Refundable and Non-refundable Postage Protocols Sending email is extremely inexpensive, in fact, the cost is negligible to many users - and spammers. This is one of the properties that make email so attractive to spammers, and allows spammers to profit from very low response rates and very limited income per message. This is also very different from the usage of traditional communication channels such as physical mail, where there is significant cost in the form of postage (as well as other costs such as printing). Therefore, one natural form of spam control is to impose postage fees on email. This approach was proposed in [39, 27], and later by Microsoft [1] as part of their *Penny Black* project.

We present a simplified design of a postage-based spam control in Figure 6.2. The sender, say Alice, or a mail agent (MUA/MSA/MTA) on her behalf, requests ('buys') a stamp for the email message, from Alice's Payment Service Provider (PSP). The stamp is a payment message approved (signed) by the PSP, assuring the recipient (Bob) that Alice has paid (a specified amount) for sending this specific message, or for sending messages to this recipient.

The mail agent then sends the email message, together with the stamp, to to the recipient. The recipient, or his mail agent (MUA/MDA/MTA), checks if the payment is sufficient and validates that it is willing to accept messages with postage from this PSP (there may be an initial exchange by which the sender knows which PSP is agreeable to the recipient); the recipient may also validate that the sender is not listed in his blacklist of spammers. Recipients may also contact the PSP to 'deposit' the stamp; most proposals advocate that recipients actually receive some or all of the payment, in which case depositing the stamp is necessary to ensure payment.

Next, Bob's MUA displays the message to Bob. Based on Bob's reaction or feedback, the spam filtering mail agent may know if the message was in fact spam or not. It it was spam, Alice may be added to Bob's blacklist, so that additional email from her will be discarded (even if a stamp is attached to it - preventing double-spending of the same stamp by spammers, while allowing

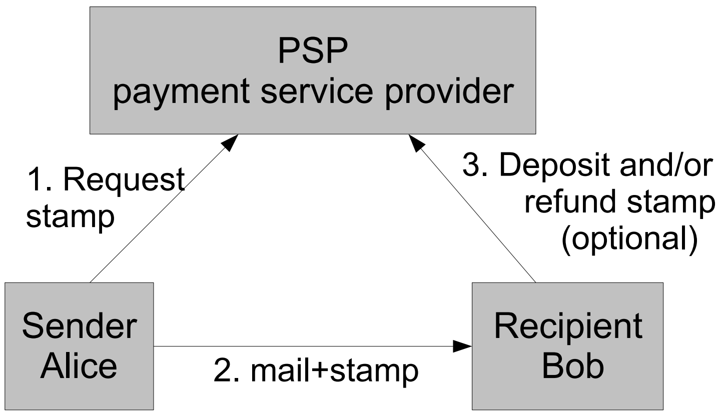

Fig. 7. Simplified design of a postage-based spam control. Sender requests ('buys') a stamp at the Payment Service Provider (PSP), and sends it to the recipient. Recipient validates and deposits the stamp, and then delivers the email to the user. If the user indicates that the email is non-spam, the recipient may refund the sender at the PSP.

double-use of the same stamp by non-spammers). If the message is clearly non-spam, then Bob's MUA may contact the PSP and instruct it to refund Alice; schemes that allow recipients to refund senders of non-spam are called *refundable*. Other schemes are *non-refundable*, i.e. postage cannot be refunded; however even in this case, Alice may be refunded by receiving an email message reply from Bob.

Penalty Protocols Postage protocols require payment for any message, spam or non-spam. Recipients may be able to refund non-spammers, however this is voluntary, and senders may be concerned about not being refunded even for non-spam. An alternative to this is the use of *penalty protocols*, where senders only pay for actually sending spam.

The main challenge in moving from postage protocols to penalty protocols is the need for handling and resolution of abuse (spam) complaints. In addition, if the payment commitment is done as a service of the domain sending the email, then the agent signing on behalf of the domain should ensure that the (signed) FROM: headers and log files will allow the domain to resolve any possible spam complaint regarding this email message. However, current email signing proposals, as discussed in subsection 5.4, still leave the complaint resolution process unspecified, and in practice - a slow and expensive manual process.

In order to automate complaint resolution, we must establish a clear process for validating a complaint, i.e. a clear test to decide if a given message is spam. Here, we return to our definition of spam in the beginning of this manuscript. We use the term *spam* to refer to undesirable message that does *not* contain an appropriate ('warning') *content label.* To automate the complaint resolution process, the content label must be authenticated (signed) together with the message.

We describe SeARP, a specific penalty-based spam control protocol, in the next subsection.

6.3 The Secure Automated Resolution Protocol

In this subsection, we present the *Secure Automated Resolution Protocol (SeARP)* of [19, 21]. SeARP is a cryptographic protocol for controlling spam, by providing compensation from senders of spam to (unwilling) recipients of spam. The parties agree in advance on the amount of compensation, as well as on the identity of one or more *resolve agent (RA)*, to which the parties give the final authority on whether a given message had the right content label or not (i.e. was spam). The SeARP protocol also supports automated-evidence reputation management, for identity-based spam control, but we will not cover these aspects here.

SeARP is built on cryptographic message authentication, and the domain authentication mechanisms of DKIM [3], with few extensions, including:

– SeARP adds an explicit, authenticated *content-label* field, similar to the Email Policy Indication proposed at [5][sections 8-9]. As a simple example, a content-label may have one of the two values: *ADV* for advertizing/commercial messages, and *non-ADV* for personal or professional messages.
– SeARP allows the mail-originating domain to authenticate the messages using either digital signatures, or the (efficient, shared-key) Message Authentication Code (MAC). The use of MAC is appropriate, when there is a long-term trust relationship between it and the receiving domain.
– SeARP allows the email sender address to belong to a *different* domain from the signing or authenticating domain. When the sender address is in a different domain, it may be spoofed or used with authorization; recipients may decide if they are willing to accept such messages, or prefer to discard them in the fear of forgery. SeARP, in such cases, only vouches for the relation between the message and the content label attached to it.
– SeARP adds additional authenticated fields to the message, described below, to aid in automated complaint resolution and compensation. For example, one of these fields identifies the resolve agent (RA) agreed to by both sender and recipient.

The *resolve agent (RA)* is a crucial entity in SeARP. It is distinct from the mail agents of the sender and of the recipient. However, the sender and recipient should both agree, in advance, to accepting the RA's judgements as to wheter a given content label l is appropriate for a given message m (or it was spam). Typically, when the RA determines the m sent by A to B is spam, i.e. it was sent with an inappropriate label l, then B receives compensation from A. The reverse may also happen, i.e. if RA receives a spam-complaint from B on a message m with label l from A, and determines this was *not* spam, then A may be entitled to compensation from B, e.g. for time wasted on the complaint process.

The SeARP protocol is therefore run between three parties: the sending mail agent, the receiving mail agent, and the resolve agent (RA). For simplicity, we assume there are only two SeARP-equipped mail agents involved, the MSA to which Alice submits the message, and the MDA which delivers the message to Bob. Also for simplicity, we assume that the sending agent knows that the receiving agent supports SeARP. Finally, we note that since SeARP determines penalties, there should be appropriate payment and/or credit mechanisms set up to facilitate payments; in particular, in a simple scenario we may consider that the RA is also acting as a Payment Service Provider to both sender and recipient, and maintaining accounts (reserves) for both of them. See [19, 21] for complete details, without these simplifications (for example, for supporting a PSP which is distinct from the RA).

Each of the SeARP agents interacts with the other SeARP agents by exchanging messages, and interacts with a human: sender, recipient and a (human) *resolver*, whom we call Ross. Ross is performing the final resolution service for the RA, and is compensated from fees paid to the RA by the parties; specifically, the sender (if the resolution is *spam*) or the recipient (if the resolution is *non-spam*, i.e. false complaint).

These humans (Alice, Bob, Ross) are responsible for mapping for the message to label mapping: Alice should provide the correct label for the message, and both Bob and Ross should judge if the label is correct. Note that while we refer to these entities as 'humans', this is only a simplification, as in reality there may be software and hardware module interfacing between the agents and any actual person. For example, it is quite likely that both RA and recipients will use content-filtering techniques to perform the initial evaluation.

We now present the main ideas of SeARP, focusing on the simplified scenario in Figure 8, where SeARP is used to control spam for messages sent from user Alice to user Bob, via two mail agents implementing SeARP, one for Alice and the other for Bob, and using a single, agreed-upon resolve agent, operated by Ross. For more details and advanced features, including the migration scenarios and support for multiple agents, see [19, 21][7].

We see in the figure the three agents (Alice's MSA, Bob's MDA, and Ross' RA), and the three humans connected to each agent: Alice (sender), Bob (recipient), and Ross (resolver). The operation of the protocol proceeds according to the following steps:

1. Initially, Bob specifies a set of 'acceptable labels' L. Bob's MDA should deliver to Bob only messages with labels $l \in L$.

[7] In [19] we used the name SICS instead of SeARP.

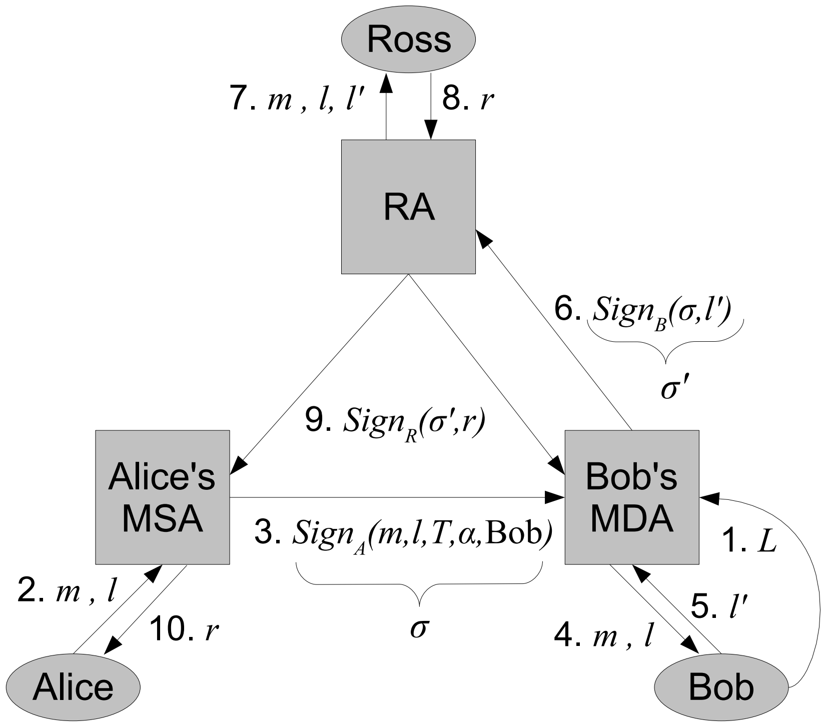

Fig. 8. Overview of the Secure Automated Resolution Protocol (SeARP). Initially (1), Bob defines acceptable labels to his MDA. Next (2), Alice sends a message m with label l. In (3), Alice's MSA sends the message towards Bob, together with a signature σ, which should be sufficient to ensure Bob and the RA the appropriate compensation, if the RA will role that the m was spam. Next (4), Bob's MDA validates the message and the signature, and displays m to Bob. Next (5), Bob indicates if the message was spam. If Bob claims that the message was spam, then, in (6), his MDA requests resolution from the RA. In (7, 8) the RA requests and receives the resolution from Ross. Then, at (9), the RA informs the parties of the resolution. Finally in (10), Alice's MSA informs Alice of the resolution and penalty.

2. Alice submits message m with label l to her MSA.
3. Alice's MSA sends m, l, σ to Bob's MDA, where $\sigma = Sign_A(m, l, T, \alpha, \mathrm{B}ob)$ is Alice's MSA signature over the message m, the label l, the validity period for the message T, additional information α allowing Alice's MSA to recover the details of this submission (e.g., index to log file entry), and finally Bob's identity (BOB).
4. Bob's MDA confirms that the message is still valid (comparing current time to T in the message), and that $l \in L$. If so, it delivers m, l to Bob.
5. Bob indicates to his MDA the 'real' label of the message, l'. If $l' = l$ then the message was properly labeled, otherwise the message is spam.
6. If Bob claims that the message was spam, i.e., $l' \neq l$, then Bob's MDA sends a 'complaint' $\sigma' = Sign_B(\sigma, l')$ to the Resolve Agent (RA), signed by the signing key of Bob's MDA.
7. The RA forwards m, l, l' to the human resolver Ross.
8. Ross evaluates m, l, l' and sends the resolution r to the RA. The resolution indicates the penalty for spam (if the correct label is indeed $l' \neq l$), or a penalty for Bob's falsely accusing Alice of sending spam, and thereby wasting the resolver's resources. In either case, the resolver receives compensation.
9. The RA sends the signed resolution $Sign_R(\sigma', r)$ to Alice's MSA and to Bob's MDA. If the RA is also the Payment Service Provider for Alice and Bob, it will update accordingly the reserves of the deposits of both parties.
10. If the resolution indicates a penalty for Alice's MSA since the message is determined to be indeed spam (l is not correct label for m), then Alice's MSA informs Alice of the resolution and penalty. This is where we use α, to aid Alice's MSA in securely identifying the sender, Alice. Alice's MSA may also take additional measures such as imposing more conservative rate controls on Alice, e.g. limit it to sending up to a certain number of messages/bytes per day.

The complete SeARP protocol supports several refinements, described in [21], including:

Separable, Interoperable Payment Services: The simplified version presented above used a single trusted authority (the RA), to both resolve the spam/abuse complaints, and to transfer money based on the resolutions. It may be desirable to separate the payment (money transfer) function from the resolving function, and handle it by a separate entity, often called a Payment Service Processor (PSP). In fact, Bob's MDA may even use a separate PSP from Alice's MSA.

Guaranteed Payment Certificate: In the common case where the recipient does not trust the sender directly, we need to extend step 2 above as follows: Alice's MSA may need to attach a valid certificate signed by the RA or a PSP agreeable to Bob's MDA and to the RA. This certificate guarantees payment in case the message is spam to Bob's MDA and to the RA. In the simplified version we described above, we assumed that Bob's MDA is always aware of the availability of funds for this payment.

Warn Sender of Suspect Spam: The complete version of SeARP allows Bob's MDA/MUA to return the message to Alice, via her mail agent, automatically when it suspects that the message may be spam, and before actually delivering the message to Bob. This allows Alice's mail agent to potentially ask Alice before proceeding, possibly using a HIP to ensure the response is really from a human user (Alice) and not from malware turning Alice's PC into a zombie for sending spam. Only if Alice insists, The MDA/MUA may use different mechanisms in identifying suspect messages, such as content filtering. In addition, Alice's MSA may use content filtering and other techniques, to filter messages initially (before sending toward Bob). Of course, Alice may elect to automatically commit to paying for spam in order to avoid having to answer such queries, at her own risk.

Bob's MDA delivers the message to Bob only if Alice 'insists' that the message is not spam; if Alice 'admits' that the original label was incorrect, the penalty it is supposed to pay for spamming may be reduced or waived. This may help in many practical scenarios such as a zombie controlling Alice's MUA, and further helps to avoid quick depletion of the sender's credit at the PSP by outstanding messages. This warning mechanism requires relatively simple additions to the basic protocol as described.

Migration Path: The complete protocol includes several mechanisms to support early deployments, including support for deployment only by a single mail agent along the path. For example, when deployed only by the MSA, SeARP may ease and partially-automate the process of handling abuse complaints.

Early deployment may often not involve the MUAs of the sender and/or of the recipient, which have a critical role in the operation of the basic SeARP process as described above and illustrated in Figure 8. Specifically, a SeARP-enabled sending MUA will allow senders to attach proper content labels when sending messages requiring labels (e.g., a customer sending a virus-sample to anti-virus lab), and prompt users to re-affirm that the label sent with a message is correct, if challenged by the recipient.

Similarly, a SeARP-enabled receiving MUA will allow users (Bob) to easily identify spam (i.e., messages with incorrect content labels), and invoke the SeARP automated complaint resolution process. When the receiving MUA is not SeARP-enabled, a SeARP implementation in the sending domain or even in the receiving domain may try to intercept any abuse complaint sent (often manually) by the recipient (Bob).

For more details and additional migration-path mechanisms, see [19, 21].

Establishing the Agreement and Terms: The complete protocol includes support for negotiation of the terms and conditions, e.g. the penalties for sending different sorts of spam.

Support for MAC: we described the protocol as authenticating messages always using digital signatures. However, in some scenarios, Bob's MDA may be willing to tolerate a small risk of Alice's MSA repudiating having sent a particular spam message, for the improved efficiency of using symmetric-key MAC instead of asymmetric-key digital signatures. Different mechanisms can be deployed to agree on the initial shared key, from using a key agreement protocol based on public-keys and certificates, to simply sending it in the clear and relying on the relative difficulty of packet interception attacks (similar to the authentication in SES [40]).

Involvement of Multiple MTAs: our description focused on SeARP deployment at the sender's MSA and the recipient's MDA. However, the protocol easily extends to deployment (also or only) in one or more MTA in the sender and recipient's domains.

Deniable Sending: In the protocol presented, Alice's MDA signs every outgoing message of Alice automatically, to allow automated resolution of abuse complaints. However, this has the side effect of leaving a long-lasting proof of the submission of that message by Alice. This may be a concern, e.g. Alice may be concerned of potential litigation based on her statements.

Encrypted Messages: The resolution process above requires the RA to receive the message sent and determine whether l is correct label for it. However, this will fail if the message was encrypted so that only Bob can decipher it (when reading the message). The complete protocol allows the use use of encrypted messages; the idea is that recipients can send to the RA the original message, so that the RA can reconstruct the encrypted message and validate the signature. For details, see [21].

Support Reputation Systems: The SeARP process described above relied completely on providing compensation to spam recipients, rather than trying to use reputation mechanisms, such as black-lists, to aid in deciding on the messages from a given sender/domain. Furthermore, we did not explain how SeARP resolutions may be used as input to build reputation of agents. While these mechanisms are not essential for SeARP, they may be valuable; for details see [21, 19].

7 Conclusions and Discussion

The biggest obstacle to the acceptance of advanced spam controls, may be the need for cooperation between (honest) senders and recipients, and/or their mail servers. Indeed, there are many aspects in the design of most of the proposed spam controls, targeted to support the gradual migration and adaptation, and provide value even to early adopters.

Postage and penalty mechanisms face additional obstacles. In particular, their deployment seems to require modifications to the agreements between email service providers and their customers, especially to ensure a deposit to cover fines for possible future spamming. We believe, however, that this process is inevitable, especially in light of the current insecurity of most computers connected to the Internet these days (i.e. the abundance of zombies). Indeed, from discussions with some Internet connectivity providers, it turns out that many of them already apply financial penalties to spammers, typically by closing spamming accounts (with a financial implication). We also expect different financial arrangements and service levels, e.g. some users may prefer a flat-fee service providing them with protection against viruses and other malware, and assuming responsibility for any spam sent by their computer.

We also believe that most users will agree to set a reasonable limit for messages sent daily, which translates into limited spamming value and limited damage to the user as well as to spam recipients and email servers carrying the spam. In fact, we hope and believe that many users will appreciate the value of becoming aware of a penetration to their computer, by their ISP or mail service provider informing them of spam originating from their computer. Growing user awareness may also result in development and adoption of better computing security mechanisms. In this way, penalty-based spam controls, such as SeARP, may help reduce the number of insecure computers connected to the Internet, thereby 'breaking' the 'vicious cycle of spam' illustrated in Figure 2.

Internet security may further benefit from the message authentication provided by cryptographic spam controls (based on authentication, reputation, accreditation and penalties). Such improved security and accountability may make it more difficult for email viruses to propagate and will help to prevent other email spoofing, phishing and eavesdropping attacks. Furthermore, some of these mechanisms can also provide the key-distribution required to provide confidentiality (encryption) of email messages.

References

1. 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. Available online from http://research.microsoft.com/research/sv/PennyBlack/.
2. L. von Ahn, M. Blum, N.J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In E. Biham, editor, Eurocrypt '03, pages 294–311. Springer-Verlag, 2003.
3. E. Allman, J. Callas, M. Delaney, M. Libbey, J. Fenton and M. Thomas, DomainKeys Identified Mail (DKIM), draft-allman-dkim-base-00 (work in progress), July 2005.
4. The Anti-Phishing Working Group, http://antiphishing.net/.
5. The Coordinated Spam Reduction Initiative, Microsoft corporation, February 2004.
6. A. Back, Hashcash - A Denial of Service Counter-Measure, manuscript, available at http://www.hashcash.org/papers/, August 2002.
7. K. Chellapilla, K. Larson, P. Simard and M. Czerwinski, Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs), Second Conference on Email and Anti-Spam (CEAS 2005), July 2005.
8. H. Danisch, A DNS RR for simple SMTP sender authentication, Internet-draft, online at http://www.danisch.de/work/security/antispam.html, first version December 2002, current version October 2003.
9. P. DesAutels (Ed.), Yang-hua Chu, Brian LaMacchia and Peter Lipp, PICS Signed Labels (DSig) 1.0 Specification, W3C Recommendation, http://www.w3.org/TR/REC-DSig-label, May 1998.
10. V. Duchovni, J. Gilmore, A. Herzberg, B. Laurie, P. Metzger, discussions at the cryptography@metzdowd.com list on the topic 'why "penny black" etc. are not very useful (could crypto stop spam??)', January 2004.

11. C. Dwork, A. Goldberg and M. Naor, On Memory-Bound Functions for Fighting Spam, Advances in Cryptology - CRYPTO 2003, LNCS 2729, Springer, 2003, pp. 426-444.
12. C. Dwork and M. Naor, Pricing via Processing or Combating Junk Mail, Crypto '92, pp.139 - 147, 1992.
13. DomainKeys: Proving and Protecting Email Sender Identity, http://antispam.yahoo.com/domainkeys.
14. E. Gabber, M. Jakobsson, Y. Matias and A. Mayer, Curbing Junk E-Mail via Secure Classification, proceedings of Financial Cryptography, pp. 198-213, Feb. 1998.
15. P. Graham. A plan for spam, Aug 2002. Available at http://www.paulgraham.com/spam.html.
16. R.J. Hall, Channels: Avoiding Unwanted Electronic Mail, to appear in Communications of the ACM. Also available at URL: ftp://ftp.research.att.com/dist/hall/papers/agents/channels-long.ps.
17. E. Harris, 'The Next Step in the Spam Control War: Greylisting', http://projects.puremagic.com/greylisting/whitepaper.html, August 2003.
18. A. Herzberg, Micropayments, pp. 245-280, chapter 12 in Payment Technologies for E-Commerce, Editor Prof. Weidong Kou, Springer-Verlag, ISBN 3-540-44007-0, 2003.
19. A. Herzberg, Controlling Spam by Secure Internet Content Selection, in proceedings of the fourth Security in Communication Networks (SCN) conference, Carlo Blundo and Stelvio Cimato (Eds.), pp. 337-350, Amaplfi, Italy, Sept. 2004.
20. A. Herzberg and A. Gbara, Protecting (even) Naïve Web Users, or: Preventing Spoofing and Establishing Credentials of Web Sites, DIMACS technical report 2004-23, April 2004. Online at http://eprint.iacr.org/2004/155/.
21. A. Herzberg and J. Levi, The Secure Automated Resolution Protocol, work in progress.
22. Identified Intenet Mail, version 3, a CISCO Whitepaper, online at http://www.identifiedmail.com/IIM
23. H. Krawczyk, M. Bellare, and R. Canetti, HMAC: Keyed-Hashing for Message Authentication. Internet RFC 2104, February 1997.
24. J. Lyon and M. W. Wong, Sender ID: Authenticating E-Mail, Internet-Draft, available at: http://www.ietf.org/internet-drafts/draft-lyon-Sender ID-core-01.txt, May 2005.
25. The Messaging Anti-Abuse Working Group, Important Considerations for Implementers of SPF and/or Sender ID, white paper, available at http://www.maawg.org/about/whitepapers/spf_sendID/, July 8, 2005.
26. J. Lyon, Purported Responsible Address in E-Mail Messages, work in progress (Internet-Draft), May 2005.
27. K. McCurley, Deterrence Measures for Spam, presented at the RSA Conference, January, 1998. Available online at http://www.almaden.ibm.com/cs/k53/pmail/.
28. RFC 1730, Internet Message Access Protocol - Version 4, M. Crispin, December 1994.
29. RFC 1731, IMAP4 Authentication Mechanisms, J. Myers, December 1994.
30. RFC1939 Post Office Protocol - Version 3, J. Myers and M. Rose (Editors), May 1996.
31. RFC2045, Multipurpose Internet Mail Extensions (MIME), Part One: Format of Internet Message Bodies, N. Freed and N. Borenstein (Editors), November 1996.
32. RFC2505 (BCP0030), Anti-Spam Recommendations for SMTP MTAs, G. Lindberg, February 1999. RFC2554

33. SMTP Service Extension for Authentication, J. Myers, March 1999.
34. Request for comments 2821, Simple Mail Transfer Protocol (SMTP), J. Klensin, Editor, http://www.ietf.org/rfc/rfc2821.txt, April 2001.
35. Request for comments 2633, Ramsdell, B., Editor, S/MIME Version 3 Message Specification, June 1999.
36. Request for comments 821, Simple Mail Transfer Protocol (SMTP), J. B. Postel, Editor, http://www.ietf.org/rfc/rfc821.txt, August 1982.
37. D. Crocker, Request For Comments 822, Standard for the Format of ARPA Internet Text Messages, http://www.ietf.org/rfc/rfc822.txt, August 1982.
38. V. Schiavone, D. Brussin, J. Koenig, S. Cobb, R. Everett-Church, Trusted Email Open Standard - A Comprehensive Policy and Technology Proposal for Email Reform, An ePrivacy Group White Paper, available at http://www.eprivacygroup.net/teos/teos_release.pdf, May, 2003.
39. J. D. Tygar , B. S. Yee and N. Heintze, Cryptographic Postage Indicia, Lecture Notes In Computer Science Vol. 1179, Proceedings of the Second Asian Computing Science Conference on Concurrency and Parallelism, Programming, Networking, and Security, pp. 378–391, December, 1996.
40. S. Goodman, J. Couzens, R. Moser and S. Gathman, The Signed Envelope Sender (SES) Protocol, manuscript, November 2004.
41. A. Schwartz and S. Garfinkel, Stopping Spam, O'Reilly & Associates 1998.
42. D. E. Sorkin, Spam Laws: State Legislation Relating to Unsolicited Electronic Mail, National Conference of State Legislatures, Assembly on State Issues, April 20, 2002.
43. E. Moustakas, C. Ranganathan, P. Duquenoy, Combating Spam Through Legislation: a Comparative Analysis of US and European Approaches, Second Conference on Email and Anti-Spam (CEAS), California, July 2005.
44. R. L. Rivest, A. Shamir, and L. M. Adleman, A Method for Obtaining Digital Signatures and Public-key Cryptosystems, Communications of the ACM, vol. 21, no. 2, pp. 120–126, February 1978.
45. Technical responses to spam, Taughnnock Networks whitepaper, November 2003.
46. An overview of E-Postage, Taughnnock Networks whitepaper, June 2003, Updated February 2004.
47. P. Hoffman, Internet Mail Consortium Report, Unsolicited Bulk Email: Definitions and Problems, IMCR-004, October 5, 1997. http://www.imc.org/imcr-004.html
48. Paul Hoffman, Dave Crocker, Internet Mail Consortium Report, Unsolicited Bulk Email - Mechanisms for Control, IMCR-008, May 4, 1998. http://www.imc.org/imcr-008.html.
49. Wikipedia, Captcha, see http://en.wikipedia.org/wiki/Captcha, 2005.
50. M. W. Wong and W. Schlitt, Sender Policy Framework (SPF) for Authorizing Use of Domains in E-MAIL, Internet draft, available at: http://www.ietf.org/internet-drafts/draft-schlitt-spf-classic-02.txt, June 6, 2005.
51. P. R. Zimmerman. The Official PGP User's Guide. MIT Press, Boston, 1995.