P2P-Based Collaborative Spam Detection and Filtering

Ernesto Damiani[1] Sabrina De Capitani di Vimercati[1] Stefano Paraboschi[2] Pierangela Samarati[1]
(1) Università degli Studi di Milano – Dip. di Tecnologie dell'Informazione – 20163 Crema, Italy
(2) Università degli Studi di Bergamo – Dip. di Ingegneria Gestionale e dell'Informazione – 24044 Dalmine, Italy
{damiani,decapita,samarati}@dti.unimi.it, parabosc@unibg.it

Abstract

Spam is one of the major problems of today email systems. While many solutions have been proposed to automatically detect and filter spam, spammers are getting more and more technically sophisticated and aware of internal workings of anti-spam systems, finding ways to disguise their emails to get around the different controls that can be enforced.

In this paper, we propose a decentralized privacy-preserving approach to spam filtering. Our solution exploits robust digests to identify messages that are a slight variation of one another and a structured peer-to-peer architecture between mail servers to collaboratively share knowledge about spam.

1. Introduction

Spam has been known as a major problem since long, but its impact on the global network infrastructure has now reached epidemic proportions (`www.spamcop.net/spamstats.shtml`). In the earliest days of spam, users could simply delete the offending messages. Later, when spam became more common, several client-side spam filtering tools became available, but they were often unreliable: users had to scan alleged spam to ensure that no important messages were deleted by mistake, with an increasing loss of time. Due to customers' complaints, governments started to contemplate anti-spam legislation [4], while several companies began offering spam filtering products to mail server operators and ISPs. When these countermeasures first reached the market, it seemed that simple economics could support a rapid eradication of spam: filtering out 95% of spam would suffice to increase the spammers' cost to reach the same audience by a factor of 20. It looked therefore reasonable to assume that high-accuracy filters could put a definitive end to spam, as few spammers had profit margins big enough to meet the cost increase. Unfortunately, things went quite differently: while most commercial anti-spam filters claim a much higher success rate than 95% in identifying spam, a huge amount of it still winds up in users' in-boxes, even when client-side and server-side filters are used in conjunction.

It may be argued that this lack of success in the war against spam is partly due to the elusive nature of the notion, which is difficult to identify by means of a software program. Of course, many messages leave no doubt: drugs, pornography, fraud and viruses now vastly outweigh the occasional unsolicited product or service sales pitch. However, there are many borderline cases where what is spam for a user could be useful information for the next person (e.g., an opening for a job position), and it may seem unwise to curb the potential of email as a mass communication channel in the spur of indignation against spam. Recently, some approaches based on the development of a P2P network for the collaborative sharing of knowledge about spam between users have been proposed [10, 15]. While these approaches represent a step toward the design of a P2P collaborative spam filtering solution, they do not pay adequate attention to the aspects of message confidentiality and robustness against attacks.

In this paper, we build on the idea that a P2P-based reputation mechanism can help in determining *what a community considers to be spam* and getting rid of it. Our proposal is aimed at achieving both *flexibility* and *effectiveness*. We first provide a critical analysis of the desiderata of spam control systems (Section 2) and of drawbacks and limitations of current solutions (Section 3). We then describe a structured peer-to-peer architecture between mail servers together with related protocols for the propagation and sharing of spam identifiers (Section 4).

2. Desiderata for spam control

While the main functions of an anti-spam filter may seem obvious, its non-functional properties have been less frequently discussed.[1] The main non-functional, global requirements that a spam control system should offer can be summarized as follows.

1 An interesting discussion on this topic can be found in the archives of the IETF Asrg mailing list, `https://www1.ietf.org/mailman/listinfo/asrg`.

Compatibility with current email infrastructure. Openness and flexibility of the email distribution system are the two main success factors of email as an information interchange channel. Therefore, any effective anti-spam solution should not impose too many constraints on existing email protocols and architectures.

Absence of false positives. Blocking unsolicited spam messages should not be an obstacle to the propagation of legitimate emails. *False positives* are innocent messages that get mistakenly identified as spam. For most users, missing a legitimate e-mail message is much worse than receiving spam, so a filter that yields false positives is considered unacceptable.

Collaborative identification of spam. Collaborative identification of spam exploits the fact that every spam message is usually sent by an automatic system to many recipients. In general, function "spam/ham" is not a computable function, and an accurate determination can only be based on the evaluation of the collective opinion of the user population.

Robustness against attacks by spammers. Spammers are becoming more and more technically sophisticated. An effective anti-spam system should be designed in the spirit of a security solution, assuming the worst-case scenario and the capability of spammers to pro-actively analyze filtering strategies and take measures to overcome them.

Protection of email confidentiality. It is of paramount importance that the content of email messages is never openly revealed by the anti-spam systems (e.g., when comparing messages or keywords). Sound privacy guarantees are important to ensure acceptance by end users.

3. Current anti-spam solutions

We briefly describe how current anti-spam solutions deal with the above requirements, underlining some open issues that will be tackled by our approach. Our discussion takes into account two main aspects: the filters' *software architecture* and their internal *filtering engines*.
Architecturally speaking, currently available spam filtering systems rely on the following three main approaches.

Email client plug-in. Such solutions consist in software plug-ins installed on every computer that needs spam filtering. This approach is mainly used for organizations with few computers whose users are in charge of managing the filters; it proves awkward to adopt by larger organizations. Web-based or mobile access to email is not protected by plug-ins, because filtering only operates on the computer where the plug-in is running.

Centralized filtering server. In this architecture, a single anti-spam filter runs on a centralized organization-wide mail server. This approach, like the following one, eliminates the need to deploy software to email clients or to train users. Centralized filters typically allow for customization of filtering rules to fit the organization's requirements. Centralized filters have the disadvantage that they do not typically use the specific preferences and opinions of the user.

Gateway Filtering. In this approach, all inbound email is routed through a filtering gateway before being delivered to the mail server. Gateway services work well with web-based and mobile access to email, and may increase robustness since they queue emails if the client network or server is off-line. On the other hand, the gateway itself is a single point of failure and may be difficult to manage in presence of multiple mail servers within an organization.

A correct approach to spam filtering should not mandate any of the above choices. P2P architectures can provide high flexibility, because they smoothly adapt themselves to the underlying network and emerging application architectures. Also, component-based design should be used to deploy anti-spam filters on single-user mailers residing on personal computers as well as on organization-wide mailers running on server clusters.

As far as the internal algorithms are concerned, current anti-spam filtering engines can be classified in the following categories.

List-based filtering. This solution was among the first to be proposed against spam. Unlike all the following, it is a coarse-grained technique operating at the server level. Today, both blacklisting and white-listing [8] are considered ineffective, although server-based solutions adopt them as an auxiliary technique often to be integrated with challenge-response [13]. Some lists of known spammers are available, like the Mail Abuse Prevention System – Realtime Blackhole List (MAPS RBL). However, blacklisting sources has become less effective since spammers learned to change their source address to get around the recipient's defenses. Overall, list-based filtering is a coarse grained solution that exhibits a limited degree of dynamicity: on the one side, responses to novel spam sources may be relatively slow, on the other it is often costly and difficult for an organization to be removed from these lists, even if the insertion was not correct or due to a transitory condition. An evolution of whitelisting is represented by special codes issued by organizations and companies like TRUST-e (www.truste.org) and Habeas (www.habeas.com) that identify legitimate communications. However, whitelists limit the open nature of email communication

Rule-based filtering. Rule-based filters assign a spam "score" to each email (`spamassassin.org`) based on whether the email contains features typical of spam messages, such as fake SMTP components, keywords, HTML formatting like fancy fonts and background colors. A major problem with rule-based scores is that since their semantics is not well-defined, it is difficult to aggregate them (when a message includes multiple spam-related features, how should its total score be computed?) and to establish a threshold that can actually limit the number of false positives. Also, experience has shown that spammers quickly learn feature-based rules and freely investigate ways to overcome them.

Source authentication/challenge-response. This technique requires the senders of a suspect email to reply to a recipient's challenge by simply clicking on "reply" and then "send". It looks promising inasmuch it has a one-time development cost compared to the permanent costs of forever looking for and implementing new filter-based technologies. Also, while filter-based systems need to include a *quarantine* inbox for suspect e-mails to avoid false positives, source authentication requires no quarantine box. However, there is one major obstacle: source authentication requires collective adoption of a modification to the current open e-mail architecture, which is unlikely to happen in the short term.

Monitoring by a human operator. While at first sight manual inspection may look like the best solution, the rate of human error (especially when the inspector is a different person from the intended recipient) should not be underestimated. Furthermore, monitoring by humans is very expensive and violates the final recipient's privacy.

Bayesian word distribution filters. Bayesian tools like SpamProbe (`spamprobe.sourceforge.net`) assign an actual frequency-based probability to (tokenized) words (or possibly on word pairs or triples) as spam indicators based on previous experience. Bayesian filters are initialized on a corpus of known spam and continuously revise probabilities [6] according to the incoming message flow. The overall probability of a message being spam is then computed by aggregating the single word probabilities. The main advantage of the Bayesian approach is that probabilities' aggregation is well understood; therefore Bayesian filters' performance is easier to measure and forecast. However, a main drawback of Bayesian filters is that they havethe hardest time blocking messages that do not lexically look like spam, e.g., messages composed of a single line of text inviting the recipient to check out a URL. Also, the Bayesian techniques may exhibit a latency both in the initial training and in responding to messages built on previously unknown vocabularies. Finally, Bayesian filters can be bypassed introducing noise within the most recognizable terms and adding a relatively high number of random words to reduce detection power.

Collaborative spam filtering. In collaborative approaches, server-side automatic monitoring systems compare incoming messages to known spam as classified by an automatic mechanism or by final recipients. These solutions have achieved considerable success as they overcome the single point of failure typical of centralized architecture.

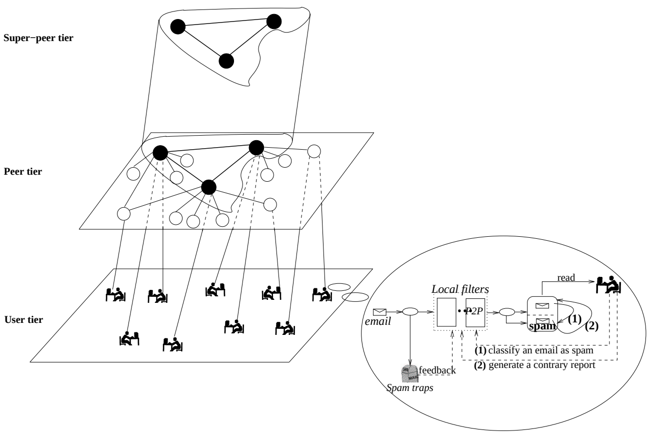

Figure 1. Network topology

All the solutions presented above have strengths and weaknesses. The solutions currently adopted that offer the best performance are those that integrate many different approaches. An example of these is SpamAssassin, which started as a rule-based filtering solution, but now integrates many different components implementing the other strategies. The advantage of an integrated solution is that it is difficult for a spammer to fool all the filters at the time; rather, techniques that operate against a filter are likely to increase the chance of detection with another filter.

4. P2P architecture and protocol

We propose a P2P-based collaborative solution that can be integrated with the filtering techniques above and that permits the detection of a larger family of spam messages. Figure 1 illustrates the architecture of the mail filtering/tagging mechanism as well as the P2P a network topology by which information about spam is shared and propagated. The network topology is based on a three-tiered architecture, with users at the lower level and a P2P network connecting mail servers above them. The P2P network has two distinct roles for the peers [14], so that some mail servers can play the role of super-peers.

Each mailer together with its set of users form a *cluster*. Intra-cluster data communication takes place via direct links between the users and their mailer, while inter-cluster communication takes place via the P2P network. The reason why in our approach users themselves do not participate as nodes of the P2P network is twofold: performance and privacy. In particular, spam reports by users are communicated by the mail server without indication of the identity of the users who originated them.[2] Each mail server knows the identity of its users (although it does not propagate it in association with reports), so we can safely assume that each user is identified by her mailer via a unique identifier.

At first sight, mailer identification may seem a bigger problem, since in P2P systems identities are usually a major concern, especially if heavyweight certificate-based techniques need to be avoided. However, our solution can rely on two aspects: (1) machines playing a specific role as mail servers are likely to have a network-wide name registered in the Internet *Domain Name System* (DNS), responsible for translating names to IP addresses, and (2) our protocol permits the construction of reputations for network participants, that strongly limit the impact that a malicious user may produce by acting as a mailserver into the P2P network.

4.1. Spam reports recording and sharing

A P2P network can be used for the exchange among mail servers of several pieces of information that contribute to the identification of spam. Since the spammer has the goal to make identification difficult, most of the components of the spam message are not adequate identifiers since they can be easily manipulated by the program managing the mass mailing. Considering the structure of email messages, we identify the following candidates as robust spam features: message digests, originating mail servers (represented by the content of the first SMTP's `Received` record generated by a local mail server), and URLs within the message. Each component requires a careful analysis and an ad hoc treatment (e.g., URLs can only contribute in a negative way, otherwise spammers could have an advantage in the introduction of URLs with a good reputation). URLs and mail servers can be represented by a combination of precise information whose sharing can be enforced with reputation protocols (e.g., [1, 7]) exploiting the advantages of DHT networks like Kademlia [9], or Chord [12]. In this paper we focus then on message digests, which require a P2P network with specific features, since comparison of digests requires similarity searches (two messages are considered the same if their digests differ in a limited number of bits).

We assume that each message m can be identified by a digest D_m that is robust against typical disguising attempts. In the following we consider two messages to be *the same message* if they map to a similar digest. In our experiments we used 256-bit digests and two messages were considered similar if their digests differed by at most 74 bits, in arbitrary positions within the digest [2].

For simplicity in the exposition, we use the term message to denote either m or D_m when the fact that we refer to a message or its digest is clear from the context. At each tier, information is maintained about spam detected or received. Intuitively, the idea is that the super-peers in the network maintain a distributed collection of spam digests that peers have identified; peers can query this collection to obtain information about unknown emails.

We assume each mail server s is associated with a pair of keys, $\langle \mathrm{PK}_s, \mathrm{SK}_s \rangle$ that it can use to sign outgoing communications. In the following we use $[\textit{statement}]_s$ to indicate a *statement* from a server s together with the corresponding signature and public key for verification. In other words, $[\textit{statement}]_s$ stands as a shorthand for the triple $\langle \textit{statement}, \mathrm{S}(\textit{statement}, \mathrm{SK}_s), \mathrm{PK}_s \rangle$.

We also assume that each mail server maintains the catalogs and control information described in Figure 2. In addition, the mail server s maintains a reputation $\textit{rep}_{s'}$ for each other mail server s' in the network. Such a reputation assesses the trust that s has in spam notifications coming from s' and allows it weighing them accordingly. Such reputations are similar to the credibility reputations introduced in [1] and to the reputations produced by Eigentrust [7]. How reputations are defined and maintained may differ for different mail servers and it can directly exploit the solutions presented in [1, 7].

4.2. Protocol

In our approach, a mail server identifies spam by means of users. If a mail server decides that a message is to be classified as spam, it sends a spam report to its super-peers. On the other hand, super-peers can be queried by mail servers wishing to know whether similar messages have been reported as spam. Upon reception of a query from a mail server, the super-peers perform a polling at the super-peer level. The polling consists in broadcasting the digest to the other nodes at the super-peer level of the P2P network and collecting possible significant records they might have registered. Consequently, the super-peers directly inquired by the mail server will return to it their local spam records as well as those they received from other super-peers.

To better illustrate the protocol and how spam reports are propagated through the P2P network, let us now examine what happens at the different levels of the network (a pseudo-code description is also provided in the appendix).

4.2.1. User tier

At the user tier, users receive emails. Upon reception of a message m, a user can report the fact that m is spam to its own mailer. We envision two principal tools to generate spam reports: users may directly classify m as spam

2 If the mail server manages a small community of users worried about their privacy, the mail server may choose to participate to the P2P network only as a client, analogous to the free-riders of current file sharing networks.

Catalog	**Description** (for each message D_m)
Received(D_m, no_copies_m)	The number of copies of D_m directed to its users that the mail server s has received
SpamInternalReports$(D_m, \{u_1, \ldots, u_n\})$	Users $\{u_1, \ldots, u_n\}$ who have reported D_m as spam
Contrary$(D_m, \{u_1, \ldots, u_n\})$	Users $\{u_1, \ldots, u_n\}$ who have submitted a contrary report about D_m
Spam(D_m)	List of messages classified as spam
SpamReports$(D_m, reports_m)$	Set $reports_m$ of signed spam reports each of the form $[D_m, conf_{s',m}]_{s'}$

Threshold	Description
Thr_copies	number of occurrences of similar messages that triggers the suspicion of a bulk mailing
Thr_int_reports	number of user reports about similar messages needed by the mail server to classify it as spam
Thr_ext_reports	threshold that measures whether external reports are sufficient to consider similar messages as spam
Thr_contrary	number of contrary reports considered as a sufficient indication that similar messages should have not been tagged as spam

Figure 2: Information maintained at each mailserver s

using the "spam" or "junk" button that many mail clients already offer (e.g., www.mozilla.org/mailnews/spam.html), or m may be addressed to a *spam trap*.[3]

If the email received by the user has been tagged as spam by the mail server, and the user agrees with that, the user does not need to do anything else. On the contrary, if the user does not agree with the current assessment of the message (i.e., she thinks the message is genuine or otherwise a useful one) she can send a *contrary report* to her mailer.

4.2.2. Peer tier

At the peer tier, each mail server s receives emails directed to its users as well as spam notifications or contrary reports from its users. Let us examine the different cases.

Email processing. For every message m received which has not been previously recognized as spam, mail server s updates the *Received* catalog, adding an entry for D_m or incrementing the corresponding no_copies_m. If after this update no_copies_m reaches threshold *Thr_copies*, s will send a *Spam_inquiry* message to the super-peers inquiring whether the message has been reported as spam by other mailers. In response to such a query, s will receive a set of *Spam_inquiry_resp* messages including a set of signed spam reports of the form $[D'_m, conf_{s',m}]_{s'}$ on messages with identical or similar digest, which it can then evaluate to determine whether m is to be considered spam. Intuitively, s can perform an aggregation of the reports, weighting them differently depending on the reputations of the mail servers involved. In other words, the aggregation can be expressed as the sum of the $conf_{s',m}$ received each weighted with the corresponding $rep_{s'}$. If the aggregation of the reports produces a value greater than the specified threshold *Thr_ext_reports*, then s considers the message as spam, and adds it to the *Spam* catalog, so to be able to tag as spam any other copy of the message directed to its users. Finally, s updates reputations associated with the mail servers that have sent an answer to the *Spam_inquiry* message.

User spam report evaluation. Each time mail server s receives from a user a spam notification about a message m, it updates the catalog *SpamInternalReports* by adding the user to the list of those who reported the message as spam (or adding an entry for it if the user is the first to report). If this update causes the cardinality of the list to pass threshold *Thr_int_reports*[4], s classifies the message as spam and: *1)* adds it to the *Spam* catalog, so that any other occurrence of the message directed to its users will be received by the user in the folder for automatically classified spam; *2)* sends a *Spam* message including a spam report of the form $[D_m, conf_{s,m}]_s$ to its super-peers reporting that it considers the message to be spam (a confidence level $conf_{s,m}$ of 1 can be assumed as a default).

User contrary report evaluation. Each time mail server s receives a contrary report stating that a user does not agree with the tagging of a message m as spam, s adds the contrary report to the *Contrary* catalog. If the ratio between the number of users that have generated a spam report for m and the number of users that have submitted a contrary report reaches the threshold *Thr_contrary*, s can decide to remove m from the *Spam* catalog. Furthermore, if the classification of the message as spam was deduced by information received by the P2P network, s can decrease the reputation of the mail servers that reported that the message was spam. By contrast, if the identification of m as spam was done internally (since more users that *Thr_int_reports* had classified the message as spam), s can submit a revoke message to its super-peers revoking the reports previously sent.

4.2.3. Super-peer tier

At the super-peer tier, there are mail servers that, besides the functionality just described, serve also as collectors and pollers of spam reports. The additional workload of the super-peer is related to managing spam reports and spam inquiries coming from the mail servers that refer to it or from other super-peers.

3 Spam traps are fictions email addresses used to detect bulk mailing to addresses detected by crawlers.

4 In principle, different thresholds could be used for triggering the local spam catalog and the notification to the super-peers.

Spam report processing. Upon reception of a new spam report *Spam*$([D_m, \mathit{conf}_{s,m}]_s)$ from mail server s about message D_m, the super-peer will add a corresponding entry in its *SpamReports* catalog. If D_m does not belong to the *Spam* catalog, then the super-peer may add it to its catalog.

Spam inquiry processing. Upon reception of a query *Spam_inquiry*$([D_m]_s)$ from mail server s, the super-peer will broadcast the query to other super-peers in the P2P network. Each of the super-peers receiving the query will respond on the network returning the reports about digests similar to D_m that appear in its *SpamReports* catalog. The super-peer directly inquired will return all the reports received as well as those it has locally stored to the inquiring mail server that will process them as illustrated above. As for all the communications of our protocol, the query response will be signed by the super-peer.

4.3. Notes on the approach

It is worth to draw the attention on some key aspects of our solution and explain the rationale behind them.

A first aspect worth noticing is that super-peers provide only a communication channel between mail servers and do not perform an intermediate aggregation of reports. This way mail servers can receive the original reports from their peers and rate them according to the reputations they associate with each of such peers. The rationale for adopting this solution (in contrast to having super-peer's aggregating reports) is twofold. First, it does not mandate complete trust in the super-peers: since each report is signed by the mail server that has sent it, super-peers cannot fake reports.[5] Second, in our solution reports are weighted with the reputation of who expresses them. The use of reputations allows giving to reports coming from peers which proved reliable in the past a higher weighting than those expressed by new mail servers we are unsure of. Assigning a reputation to the different mail servers requires distinguishing the reports coming from them.

Another aspect worth noticing is that every communication by the nodes of the P2P network is always signed. The reason for this is to guarantee the authenticity of the report content as well as of its originator.

One may object that our solution appears resource demanding; however, the requested resources are local storage, computational power, and network bandwidth which have a limited cost compared with user time. User time is the most precious resource, and its waste is the highest cost the economy is today paying for spam.

5 The fact that super peers could selectively discard reports is taken care of by the fact that each mail server uses more than one super-peer.

5. Security considerations

In our approach, two key security aspects have to be taken into consideration. The first aspect is that the digest mechanism must satisfy stringent requirements and needs a careful design. In [2] we described an approach to produce digests of mail messages that is robust against typical disguising attacks (e.g., random addition, thesaurus substitution, perceptive substitution, and aimed addition). The second is related to the robustness of an open P2P network to malicious users. Indeed, a spammer can attack the system at the infrastructure level or at the vote generation level. At the infrastructure level, the spammer can have the major impact if it assumes the role of super-peer, and then selectively expunge votes against its messages. The defense against this attack is twofold. First, we assume a node connects to a super-peer only if it is a reliable node registered in the DNS system as the mail server for a domain and if it has already acquired a good reputation within the P2P network. The second protection is given by the fact that the node will connect with several super-peers, as customary in current networks [5]. This will allow the node to compare the results of the queries coming from the different nodes and it will permit to identify anomalies.

At the vote generation level, a spammer can attack the system by posing as a mail server and propagating contrary reports for its own messages. However, even if the spammer succeeds in impersonating a mail server, its reports will typically be not considered as they have no associated reputation. The spammer can try to acquire a good reputation by doing a good service to the community and voting spam for some time as it is detected, and then exploit this reputation to protect the distribution of its spam. This attack is mitigated by the fact that to acquire a good reputation the spammer has to spend significant resources, producing a concrete benefit for the community, and the good reputation that is acquired is rapidly dissipated as soon as it starts again spreading faked contrary reports [11].

Instead of attacking the system, the spammer can try to fool it by sending to each mail server a number of spam messages below *Thr_int_reports* (i.e., number of users that have to report a message as spam) and *Thr_copies* (i.e., number of copies of the same email received by the mail server). This attack is applicable only if the spammer can acquire information about these thresholds, but this is in itself a difficult task, as each mail server will individually choose its parameters; also, since forwarding mechanisms are relatively common, the final destination of a message is not detectable by the email address alone, and the number of distinct mail servers is several orders of magnitude less than the number of email addresses, increasing in a corresponding way the costs for the spammer.

Another kind of attack is represented by malicious users attempting to sabotage the delivery of an email message that is not spam; e.g., a spammer may desire to stop the notification to the mailing list of mail server administrators that a new release of the P2P spam blocking client can be retrieved. Contrary reports allow detecting such situations and act accordingly.

As a final remark, we note that generic attacks, like denial-of-service attacks on the super-peers of the network, are extremely difficult to realize, as it is a basic property of P2P networks that they are able to resist to these attacks. The experience on P2P networks for file sharing demonstrates the capacity to manage significant loads.

6. Related work

Our solution falls in the class of collaborative spam filtering. Two of the most known systems in this area are Vipul's Razor (`razor.sourceforge.net`), now reincarnated in the commercial system SpamNet (`http://www.cloudmark.com`), and Distributed Checksum Clearinghouse [3]. SpamNet uses a centralized catalog for storing and sharing digests of known spam. When a user identifies an email as spam, the email is sent to the SpamNet server. When a message is accessed by a mail client, a unique digest of the message is sent to the SpamNet server to determine whether the message is a known spam. If the digest matches one digest in the spam catalog, further checks are executed that exploit the full knowledge that the server has of the spam message. Users are characterized by a reputation, managed internally by the SpamNet server, that gives a greater weight to users that have demonstrated to be good classifiers. However, if a user accidentally blocks an email that contains important information, the email itself (which can be confidential) is automatically transferred to SpamNet. Overall, the system exhibits a single point of failure and the requirement that spam is centrally collected has a critical impact on user confidentiality.

Distributed Checksum Clearinghouse is based on a number of open servers that maintain databases of message checksums (a checksum is similar to the concept of digest in the previous systems). Mail users and ISPs can submit checksums of all messages received. The database records how many copies of each message are submitted. If requested, the DCC server can return a count of how many instances of a message have been recorded. If the returned count is higher than a threshold set by the client and according to local whitelists the message is unsolicited, the DCC client can then log, discard, or reject the message. The DCC initiative is very interesting and it already offers a useful service to the Internet community. It also demonstrates the advantages of a distributed architecture for spam detection. However, DCC does not explicitly consider the security aspects that are introduced by the use of an open P2P architecture and that have been an important subject of our investigation. Currently, a malicious DCC client can report a message to a DCC server as having been received many times thus generating false positives (i.e., an email is marked as "bulk" by a DCC server that is not). On message identification, DCC uses techniques that try to strip the messages of random noise and then it produces hashes on specific portions of the message; this technique is more sensitive to attacks than the digest functions that we propose in the paper.

The protocol presented in this paper builds on a previous proposal using reputation to assess peers and resource reliability in resource sharing. However, the spamming protocol presented in this paper differs from [1] in several respects related to signing and encrypting. First, while [1] assumes that each communication is encrypted for confidentiality (also due to the pseudo-anonymous nature of the considered scenario) here confidentiality of reputation is not considered an issue and therefore it suffices to provide a signature for guaranteeing the integrity of the reports. Second, the polling protocol in [1] included some direct communication with randomly chosen nodes from which a vote (corresponding to our report) had been received to counteract possible masquerading attacks. Reputation management and lightweight authentication based on DNS allows avoiding the burden of such additional checking in the present context. Third, in [1], a polling could have reported both positive and negative votes. Here, we use a single kind of report: the consideration of both positive and negative votes (intuitively reports stating that a message is genuine) would have considerably increased the workload of the system and affected the user as well. While we can imagine spam notifications come at no cost to the user, the situation would be different if users were required to return a positive report for every genuine message they receive. While the choice of not dealing with positive votes seems mandatory, complete absence of positive votes opens the door to abuses. Contrary reports solve this problem.

7. Conclusions and future work

In this paper we presented a solution exploiting the P2P potential to make a contribution to reduce the level of spam. An important strength of our proposal is that it bases on an open distributed architecture and does not rely on any authority or centralized control. We believe that our solution offers the opportunity to demonstrate how research on P2P networks, that has until now been perceived by a great part of the research community as mainly a mechanism to share copyrighted material, can be immediately adapted to contribute to the solution of an important and visible problem.

8. Acknowledgments

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

References

[1] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in P2P systems. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):840–854, July/August 2003.

[2] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Using digests to identify spam messages. Technical report, University of Milan, 2004.

[3] Distributed checksum clearinghouse. http://www.rhyolite.com/anti-spam/dcc/.

[4] EC Directive on Privacy and Electronic Communications (2002/58/EC).

[5] Gnutella. http://rfc-gnutella.sourceforge.net/.

[6] P. Graham. Better bayesian filtering. In *Proc. of the 2003 Spam Conference*, Cambridge, January 2003.

[7] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of the Twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.

[8] Mail abuse prevention system, llc. http://mail-abuse.org/.

[9] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of IPTPS02*, Cambridge, USA, March 2002.

[10] J. Metzger, M. Schillo, and K. Fischer. A multiagent-based peer-to-peer network in java for distributed spam filtering. In *Proc. of the 3rd CEEMAS*, Czech Republic, June 2003.

[11] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, March 2001.

[12] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the SIGCOMM 2001*, San Diego, California, USA, August 2001.

[13] Tagged message delivery agent (TMDA). http://tmda.net/.

[14] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. of the 19th Int'l Conference on Data Engineering*, Bangalore, India, March 2003.

[15] F. Zhou, L. Zhuang, B. Zhao, L. Huang, A. Joseph, and J. Kubiatowicz. Approximate object-location and spam filtering on peer-to-peer systems. In *Proc. of the ACM/IFIP/USENIX International Middleware Conference*, June 2003.

A. Pseudo-code description of the protocol

```
USER TIER [for each user u]

1. Email processing
  1.1. Receive email m
  1.2. If u recognizes m as spam or u is a spam trap then
       a spam report is sent to the mail server
  1.3. If m is tagged as spam and u recognizes it as relevant then
          a contrary report is sent to the mail server

PEER TIER [for each peer p]

1. Email processing
  1.1. Receive email m and compute $D_m$
  1.2. If $D_m \simeq D' \wedge D' \in$ Spam then m is tagged as spam
       else if $D_m \simeq D' \wedge D' \in$ Received then
              update Received($D'$,no_copies$+1$)
           else insert Received($D_m$,no_copies: $1$)
           If no_copies $\geq$ Thr_copies then
              Send message Spam_inquiry($\mathrm{PK}_p$,$[D_m]_{\mathrm{SK}_p}$) to super-peers
              Expect back response messages from super-peers
              Spam_inquiry_resp($\{(\mathrm{PK}_{p_1}$,$[D'_m$,conf$_{p_1,m}]_{\mathrm{SK}_{p_1}})\ldots\}$)
              If Aggr((rep$_{p_1}$,conf$_{p_1,m}$),...) $\geq$ Thr_ext_reports then
                 Add $D_m$ in Spam
                 Update reputations associated with $p_1 \ldots$

2. User spam report evaluation
  2.1. Receive from user u a spam report including digest $D_m$
  2.2. Update SpamReports($D_m$,$\{u_1 \ldots u_n \cup u\}$)
  2.3. If $|\{u, u_1, \ldots, u_n\}| \geq$ Thr_int_reports then
          Add $D_m$ in Spam
          Send a spam message to super-peers
          Spam($\mathrm{PK}_p$,$[D_m$,conf$_{p,m}]_{\mathrm{SK}_p}$)

3. User contrary report evaluation
  3.1. Receive from user u a contrary report including digest $D_m$
  3.2. Update Contrary($D_m$,$\{u_1, \ldots, u_n \cup u\}$)
  3.3. If $|\{u, u_1, \ldots, u_n\}| \geq$ Thr_contrary then
          Delete $D_m$ from the Spam catalog
          If $D_m$ has been internally tagged as spam then
             Send a revoke message to the super-peers
          else Decrease reputation associated with mail servers that
               reported $D_m$ as spam

SUPER-PEER TIER [for each super-peer s]

1. Spam message processing
  1.1. Receive from peer p a spam message
       Spam($\mathrm{PK}_p$,$[D_m$,conf$_{p,m}]_{\mathrm{SK}_p}$)
  1.2. Add Spam($\mathrm{PK}_p$,$[D_m$,conf$_{p,m}]_{\mathrm{SK}_p}$) in the SpamReports catalog
  1.3. If $D_m \notin$ Spam then Add $D_m$ in the Spam catalog

2. Spam inquiry processing
  2.1. Receive from peer p a Spam_inquiry message
       Spam_inquiry($\mathrm{PK}_p$,$[D_m]_{\mathrm{SK}_p}$)
  2.2. Broadcast the Spam_inquiry message toward other super-peers
  2.3. Expect back response messages from super-peers
  2.4. Send a Spam_inquiry_resp to p including all the reports on
       similar digests
       Spam_inquiry_resp($\{(\mathrm{PK}_{p_1}$,$[D'_m$,conf$_{p_1,m}]_{\mathrm{SK}_{p_1}})\ldots\}$)
```