To be presented at the
MIT Spam Conference
March, 25, 2010

# Reducing Spam via Trustworthy Self Regulation by Email Senders

Naftaly H. Minsky
Department of Computer Science
Rutgers University
New Brunswick, NJ, 08903 USA
Email: minsky@cs.rutgers.edu

February 9, 2010

**Abstract**

This paper introduces an email sending technique, called *trustworthy self regulation* (TSR), which enables the receiver of an email message to recognize the sending protocol that generated it. The availability of this sending technique is expected to help induce email users to send messages via spam-immune protocols preferred by their destination users—thus producing less spam. The TSR email sending technique, in turn, employs a middleware called *law-governed interaction* (LGI).

The TSR-based communication involves no text-based filtering, no dependency on blacklistings, and no coercion by ISPs. And it can be deployed incrementally, because it can operate along with all these conventional anti-spam measures, as a complement to them, and because it involves no changes to the SMTP protocol.

If widely deployed, we claim that the TSR-based email would result in a significant reduction of traffic of spam over the Internet, and of unwanted emails that individual users have to contend with. And these results would be achieved without incurring the undesirable side effects of conventional anti-spam measures, like the blocking of valid mail by filtering, and by coercive measures imposed by ISPs. However, wide deployment of TSR over the Internet would require the deployment of the trusted infrastructure of LGI, which is yet to be done.

## Contents

## 1 Introduction

One can distinguish between three elements on which anti-spam measures can be based: (a) the content of messages; (b) the reputation of email senders, and of the servers (MTAs) through which messages are being forwarded; and (c) the behavior of the email senders themselves, or more precisely, the protocols being used for sending email.

The first two of these elements are the most commonly used ones in conventional anti-spam technology. They are used primary for the filtering of messages at the receiver's side. Such filtering is, on one hand, an impressive success, as it shields individual users from the vast majority of spam sent to them. On the other hand, filtering has some serious and well known drawbacks, among the most important of which are the following: (1) the *false positive* phenomena, which causes the blocking of valid (non-spam) messages, thus undermining the credibility of email as a reliable communication medium; (2) filtering does not seem to decrease the traffic of spam over the Internet—along with the resulting distribution of malware—and may even contribute to its unrelenting growth (a recent 31/3/09 NYT article reports that spam constitutes 94% of the overall email traffic over the interent, a percentage that continues to grow); and (3) there is a continuing "arm race" between filtering and spamming, with no end in sight—and this arm race is quite expensive for the defenders.

In addition, blacklisting, which is an important factor of filtering, has a dark side (pun intended) due to the way blacklists are created. Here is how Lawrence Lessig [7] describes the formation of blacklists, in the context of email.

> *A large number of network vigilantes—by which I mean people acting for the good in the world without legal regulation—have established lists of good and bad e-mail servers. [The] blacklists are compiled by examining the apparent rules the email server uses in deciding whether to send e-mail. Those servers that do not obey the vigilante's rules end up on a blacklist.*

This can be quite harmful, because once a given sender or ESP is included in a blacklist $B$, messages sent through it would be blocked by any MSA or MTA that happen to subscribe to $B$. So, from the viewpoint of email users, inclusion in a blacklist functions as a decree regarding the message they cannot receive, of cannot send effectively. And it is an obscure decree at that, because email users usually do not know which blacklists are used for filtering their messages, and they certainly do not know how these blacklists operate. This would not be a serious problem if there were an agreement about what kind of behavior should place an ESP in a blacklist. But there is no such agreement, as each "vigilante" operates according to its own rules. As pointed out by Lessig, one unfortunate consequence of this kind of obscure decree, which is applied indiscriminately, is that *there is often no appeal of the decision to be included on a black list* (a similar point has been made in [5]).

**Focusing on Spam-Immune Sending Protocols:** Given these limitations of the first two elements of anti-spam measures, we turn in this paper to the third element mentioned above, which plays only a secondary role in conventional anti-spam efforts. That is, we will focus on anti-spam techniques that are based on the email sending protocols, rather than on the text of message, or on black and white listings. The basis for such techniques is that a sending protocol can be *immune* from generating spam, in the sense that messages generated by such a protocol are likely (even if not certain) to be considered valid by their receivers. Indeed, several immune protocols have been identified in the literature. They include, among others: (a) protocols that requires *payment*, of some kind, for every message sent [4]; (b) *rate limiting* protocols [12]; and (c) protocols that respect the wish of users to *opt out* from being sent messages to [6]. (The reasons for considering these protocols immune are fairly self evident, and are discussed in the above cited papers (without the term "immune"); these reasons are not repeated here.)

The relevance of immune protocols to our subject matter is, of course, that if many email users—particularly those that send balk email—would adopt such protocols for sending email, the traffic of spam would be reduced. And email senders may be induced to use immune protocols if they know that most of those they send messages to would not admit a message to their inboxes unless it has been generated according to some chosen immune protocols.

But for a receiver of an email to distinguish between messages on the basis of their sending protocol, he must be able to recognize the sending protocol that generated messages he receives. Unfortunately, as we shall see below, conventional email mechanisms do not support such recognition, despite serious, although not entirely successful, attempts to do so for some specific immune protocols. This, we believe, is the reason for the relatively minor impact that immune sending protocol are having on the volume of spam.

**On the Conventional Use of Immune Protocols:** Let us examine this situation for the three immune protocols mentioned above. First, consider the payment protocol that requires micropayement for the sending of every email message. Various cryptographic techniques have been devised to enable the receiver to recognized that a message he is getting has been paid for. But these techniques are not quite satisfactory, as explained by Herzberg [4], as follows:

> *Monetary payments require interoperability between senders and recipients, [with] acceptable overhead - even for the relatively low-charges which may be acceptable for email (micro payments) ... Considerable efforts were made to develop such globally-interoperable and low-overhead micropayment schemes ... but so far none of these was truly successful.*

And many others agree with this assessment, see [3] for example.

Second, suppose that an email user Bob is willing to admit into its inbox only messages created by a rate limiting protocol. The problem is that when receiving a message from Alice, Bob cannot tell the rate in which Alice sends messages, because he just got one of them. (We will often use the names Alice and Bob for the sender of an email and for its receiver, respectively.) Indeed, no means have been proposed so far, which would enable a receiver of a message to recognize that it has been send under such a protocol. Rate limiting protocols are nevertheless employed. Not by the senders or receivers of messages, but by some ESPs which throttle the messages sent by their blacklisted senders [12]. Unfortunately, one cannot trust all, or even most, of the heterogeneous ESPs operating over the Internet to carry out this, or any other, anti-spam measure, or to do it correctly. For this, and other reasons, rate limiting protocols are considered ineffective [3].

Third, consider *opt out* protocols, which constitutes a natural protection against unsolicited communication. The importance of this type of sending protocols is evident from the fact that it is required by the laws of several countries (in particular by the CAN-SPAM act of the US Congress) [6]. But as argued in [11], such laws proved ineffective, mostly because of the difficulty to enforce them. No technical means have been proposed for a receiver to recognize a message sent under this protocol, and thus be confident that he can block this source of messages by opting out of it. It is true that many institutional email senders, and electronic mailing lists, are build to abide by this protocol. But one cannot trust every sender over the Internet, least of all professional spammers, to do so.

**Our Approach to Spam Reduction:** This paper introduces an email sending technique, called *trustworthy self regulation* (TSR), which enables the receiver of an email message to recognize the sending protocol that generated it. The availability of this sending technique is expected to help induce email users to send messages via spam-immune protocols preferred by their destination users—thus reducing the volume of spam. The TSR email sending technique, in turn employs a middleware called *law-governed interaction* (LGI) [9].

The use of TSR-based emailing involves no text-based filtering, no coercion by ISPs, and no dependency on blacklistings. But it can operate as a complement to, and in conjunction with, all these conventional anti-spam measures. Also, TSR involve no changes to the SMTP protocol, and can be deployed incrementally.

A broad adoption of TSR-based emailing is expected to result in a substantial reduction of spam over the Internet; but such adoption would require the deployment of the infrastructure of LGI over the Internet. Although the deployment and maintenance of this infrastructure would not be inexpensive, its cost could be amortized by the many other potential applications of LGI which would utilize the same infrastructure used for TSR-based email.

The rest of this paper is organized as follows: Section 2 is an outline of the proposed trustworthy self regulation approach to spam reduction. Section 3 is an overview of the law-governed interaction (LGI) middleware on which this paper is based. Section 4 is a schematic description of the implementation of TSR-Email. And Section 5 concludes this paper.

## 2 An Outline of the Trustworthy Self Regulation (TSR) Approach to Spam Reduction

We start this outline of the TSR approach by introducing the concept of *law-based trust* (*L-trust*, for short), which is the key element of this approach; then, in Section 2.2, we discuss the manner in which L-trust is supported. Section 2.3 introduces several examples of the realization of immune sending protocols under TSR. And in Section 2.4 we outline the way in which TSR can be applied to the spam problem.

**A Note About Terminology:** Since the TSR-based emailing proposed here employs the LGI middleware, we will henceforth replace the term "protocol," as used in the introduction, with the term "law," which is the technical[1] term for a protocol under LGI.

## 2.1 The Concept of Law-based Trust

The concept discussed here is one of the basic elements of LGI, introduced in [8] as "regularity based trust". Here we introduce a slightly less general version of this concept, specialized to email communication, which we call "law-based trust" (or L-trust).

Before defining this concept we introduce two conditions on which it depends, and which are supported by LGI.

1. There is a language for specifying message-sending laws (namely, what we have previously called "sending protocols"), which we call here simply a *law language*. Laws written in this language are called *TSR laws*, or simply laws, and most of them are not immune of spam.

   Since, as we shall see, immune laws tend to be stateful—i.e., sensitive to the history of email sending by a given sender—it is important to point out that the law language in question is expressive enough to specify *stateful laws*.

2. There is an SMTP-compliant mechanism for sending email messages, subject to any given TSR-law $\mathcal{L}$; such a message is called an $\mathcal{L}$*-email* (or a *TSR-email*, in a context where the specific law under which this email has been sent is of no concern.)

We now define the concept of L-trust as follows.

**Definition 1 (L-Trust)** *We say that a recipient of an email message—or its MDA (mail delivery agents)—has a law-based trust in this message, if it can determine with reasonably justified confidence, whether or not this message is a TSR-email; and if so, it can identify the law under which this message has been sent.*

It is worth pointing out that L-trust does not require the receiver of an email to have any trust in the sender. What L-trust does require is a trustworthy middleware, as we shall see below.

[1]One reason for choosing the term "law" is because it is a constraint that is enforced under LGI, and can therefore be trusted to be observed by all actors subject to it—it is thus analogous, in a sense, to laws in physics.

## 2.2 On Scalable Support of L-Trust

The question that confront us here is how can a receiver of a message recognize the law that governs its sending process, without having any information about the original sender of this message; and how can such recognition be done scalably, in particular when there is a wide range of laws that may govern the sending of a given message (i.e., all the laws that can be written by a given law-language).

Let us first consider a simpler question, where scalability is not required, and where one needs to recognize only whether a given arriving message has been sent under a specific law $\mathcal{L}$, or not. The answer to this question is straightforward and well known. All the sender has to do to ensure the receiver that the message being sent to him is an $\mathcal{L}$-email, is to send it via an intermediary $T^{\mathcal{L}}$, which is widely trusted by email users to operate in compliance with law $\mathcal{L}$. Such an intermediary, which we call a *controller*, serves here as a *trusted computing base* (TCB)—trusted, in this case, for forwarding emails subject to law $\mathcal{L}$. The concept of TCB is, of course, common in secure computing in general, and critical to secure communication over the Internet. As a prominent example, the root server of the DNS is a TCB, which needs to be almost universally trusted for Internet communication to be possible.

But the required scalability, and the multiplicity of possible laws complicates this problem. Consider scalability first. Unlike DNS servers, the controller $T^{\mathcal{L}}$ must be directly involved in the transfer of each $\mathcal{L}$-email. So, it would be prohibitively unscalable to use a single controller $T^{\mathcal{L}}$ to mediate all $\mathcal{L}$-emails over the Internet. Nor can we use such scalability-enhancing techniques as replication and caching, because the laws that needs to be supported might be highly stateful—as we shall see in the following section. This suggests that we will need a large number of such controllers, each serving one, or a limited number of email senders. And all these controllers need to be trusted by anybody who gets messages mediated by them. In other word, we need to have a *decentralized trusted computing base*, or a DTCB.

Second, if there can be a wide range of possible email-laws, and if there are many such laws in use, and they may change not infrequently, then we are facing the following difficulty. If we are to build a specific controller to operate under every law in use, adapting this controller to any change in its law, it would be impractical to expect such controllers to be widely trusted.

It follows that these controllers need to be generic, i.e., *trusted to operate under arbitrary TSR-laws.* That is, each controller should be able to operate under any well formed law written in the given law language.

Such a distributed set of generic controllers is provided under LGI via what is called a *controller service* or *CoS*. This service authenticates its controllers via a digital certificates, vouching for their correctness. It should be pointed out however, that the CoS provided by LGI is experimental, and is not widely available for email users over the Internet. The formation of a widely available CoS is mostly an industrial matter; we will return to this issue later in this paper.

So, if a given user, say Alice, wants to send emails subject to a given law $\mathcal{L}$, she needs to acquire the use of one of the available generic controllers, providing it with the law $\mathcal{L}$ to operate under—we call this act "the adoption of a controller under law $\mathcal{L}$." While this controller serves to mediate the emails sent by Alice, subject to law $\mathcal{L}$, it is denoted by $T^{\mathcal{L}}_{Alice}$. When an email user Bob receives a message from Alice sent via $T^{\mathcal{L}}_{Alice}$, it can recognize it as an $\mathcal{L}$-email, broadly in the following manner. First Bob would examine the certificate provide to him by $T^{\mathcal{L}}_{Alice}$, vouching for its authenticity as a valid generic controller. And second, it will examine the hash of law $\mathcal{L}$, sent to it by $T^{\mathcal{L}}_{Alice}$, which enable Bob to identify the law under which this message has been sent.

## 2.3 A Sample of TSR-based Immune Email-Sending Laws

We introduce here a sample of potentially useful immune laws in the context of the TSR approach to email. All these laws are specified informally, and their formal specification under LGI is straightforward.

The first two of these laws deal with the well know *payed postage* and *rate limiting* protocols. As we shall see, the realization of these protocols via the TSR approach is more effective than their conventional realization, and it tends to provide for a wide range of useful variations. The third law is a realization of the *opt out* (along with *opt in*) protocol, which has been required by some legislators [6], but so far has no satisfactory computation support. The fourth law to be discussed deals with communication between the members of a given community, and it demonstrates that immunity from spam may be defined differently for different email users. Finally, we introduce some hybrid laws, that combine several elements of immunity, and may be the most likely types of laws to be used in practice.

### 2.3.1 Payed Postage ($\mathcal{PP}$) Law

This law is a realization of a type of payment protocols mentioned in the introduction, whose aim is to ensure that the sender pays for every message she sends, thus making it less likely for spam to be generated under this law. Under the conventional approach to paid postage protocol, every message is to include a virtual stamp, which is a certificate signed by a trusted Payment Service Provider (PSP), asserting, in effect, that the sender has payed for the right to send a message to a specific target. But, as has already been pointed out, this kind of sending protocol requires interoperability between senders and recipients, with acceptable overhead. But despite considerable efforts, no such globally-interoperable and low-overhead micropayment scheme has been discovered [4].

The gist of the payed postage ($\mathcal{PP}$) law we propose is as follows (formulation of similar laws under LGI have been published widely, see [9] for example).

1. A user Alice, who intends to send $\mathcal{PP}$-emails, starts by adopting a controller $T^{\mathcal{PP}}_{Alice}$, and instructs it to purchases stamps from a specified stamp vendor—which needs to be trusted, just like the PSP of the traditional approach. Unlike the payed stamps under the conventional approach, these stamps do not specify an addressee, and can thus be used by Alice to send messages to any target, as we shall see below. As a consequence, Alice can buy her stamps in bulk, and these stamps (or, rather, just a count of how many they are) would be maintained in the state of the controller $T^{\mathcal{PP}}_{Alice}$ used by Alice to send her $\mathcal{PP}$-messages.

2. A controller $T^{\mathcal{PP}}_{Alice}$ permits the sending of email only if it has at least one stamp in its state, and every email sent by this controller would consume one of the stamps possessed by it.

3. No stamps are sent to the target of the message. And none is required, because due to L-trust, Bob, the receiver of such a message, can detect that it has been sent subject to law $\mathcal{PP}$; and assuming that Bob is familiar with the provisions of this law, he will know that the message he is getting has been paid for.

It is worth noting that due to the ability of the sender to buy stamps in bulk, we are not dealing with micropayments here.

**useful Variants of this Law:** One can easily implement a host of variants of this law, such as the following.

*1: Protecting against botnet attack:* The $\mathcal{PP}$ law outlined above is vulnerable to the following kind of attack[2]: If the computer serving Alice is made into a zombie, the attacker can use up all the stamps in the state of Alice's controller $T^{\mathcal{PP}}_{Alice}$ for his own purpose, without paying for them himself. To make law $\mathcal{PP}$ less vulnerable to such an attack one can add to it the following provision.

The modified $\mathcal{PP}$ law would provide for a default upper bound—say 200—of messages that can be sent in a single day via any controller operating under law $\mathcal{PP}$. But the sender may authorize the sending of another bunch of 200 messages (subject to the availability of stamps) via a one-time password—and such authorization can be made any number of time during the day. Since the attacker is not likely to have the right password, he would be able to send no more than 200 messages a day via a single zomby.

*2: A small allowance of free messages per day:* Most users who generally do not send many messages would benefit from adding the following provision to law $\mathcal{PP}$: Every sender can send a small number (say 100) messages a days, without paying for them by stamps. With this provision, an occasional email sender would rarely have to purchase stamps.

*3: Requiring authentication for purchasing large number of stamps:* One may argue that for purchasing truly large number of stamps—say over a million—paying for them should not be sufficient. Rather, the buyer should be required to authenticate herself, in some manner, to the stamp vendor (not necessarily to the targets of her messages). Moreover, such authentication, which is very easy to require under an LGI-law, may enable the stamp vendor to refuse selling stamps to zombies, and it may be useful for making payments to the vendor. (Clearly, the occasional email sender under variant 2 above, would not be required to authenticate himself).

### 2.3.2 A Rate Limiting ($\mathcal{RL}$) Law

We introduce here a realization of the rate limiting protocol, via a TSR law $\mathcal{RL}$. This is a straightforward law that does not allow a controller $T^{\mathcal{RL}}_{Alice}$ to forward messages sent by Alice at a higher rate than specified by this law. (Rate limiting laws have been implemented frequently in various LGI applications.)

As has already been pointed out, the conventional way for implementing this protocol is to have it enforced by various ESPs and ISPs that throttle the flow of messages coming from senders suspected of producing spam [12]. And

[2]This vulnerability has been pointed out to the author by Yaron Minsky

we have explained, this technique suffers from several serious drawbacks. First, it is anything but reliable, because, in general, ESPs and ISPs cannot be trusted to do such throttling; and second, the reliance on blacklists for deciding whom to throttle is very problematic.

The rate limiting via law $\mathcal{RL}$ does not have these limitations. The decision to send messages under this law is purely voluntary, and the receiver of an $\mathcal{RL}$-message can tell that it has been produced via a rate-limited process.

### 2.3.3 A Law ($\mathcal{IO}$) that Supports Opting In and Out

Providing users with the ability to *opt out* of any source of email is an effective protection against unsolicited messages; and as we shall see below, an *opt in* capability, if carefully defined, can be useful as well. Therefore, the legislators of several countries required various versions of these capabilities in their laws—such as the CAN-SPAM act of the US Congress. But although many electronic mailing lists are build to abide by such provisions, their legislations proved ineffective, mostly due to difficulty to enforce them [11]. And we know of no proposal to enforce these provisions via technical means.

But TSR provides a novel and effective approach to both *opt in* and *opt out* capabilities, exemplified by the following law (called $\mathcal{IO}$). We introduce this law below by describing schematically the behavior of a controller $T^{\mathcal{IO}}_{Alice}$, adopted by a email-sender Alice to operate under this law.

1. A controller $T^{\mathcal{IO}}_{Alice}$ that manages the email sending on behalf of sender Alice maintains two mutually exclusive lists of addresses in its state: (a) the opt-in list of those that opted to accept messages from Alice, and (b) the opt-out list of those that rejected such messages.

2. Email users can enter their addresses to either of the above lists, or remove them, by appropriate messages to the controller $T^{\mathcal{IO}}_{Alice}$ (not to Alice's email address).

3. The $T^{\mathcal{IO}}_{Alice}$ controller would forward arbitrary messages, initiated by Alice, to any address on the opt-in list. But it would not forward messages to any address on the opt-out list.

4. The $T^{\mathcal{IO}}_{Alice}$ controller can forward structurally restricted *introductory messages* to any address not in the opt-out list. The introductory messages, intended to introduce the sender to an email user, may have a header of a specified structure, and a very brief text (say a hundred characters) which is unlikely to be sufficient for advertising purposes.

We note that a sender Alice operating under this law, who want to get rid of the opt-out list accumulated in her $T^{\mathcal{IO}}_{Alice}$ controller can do so simply by adopting another controller. But by doing so she will also lose her opt-in list. Also note that the immunity of this law can be enhanced by combining it with paid postage elements, as we shall see in Section 2.3.5.

### 2.3.4 A Law ($\mathcal{GC}$) of Group Communication

Consider a group $G$ of people who generally view messages received from other members of $G$ as acceptable, i.e., they do not view such messages as spam. Such a group may, for example, consist of the members of a given team, department, or organization. It may even be a set of practitioners of a certain profession, such as cardiologist, who value each other's opinions, and trust each other not to send spam.

Members of such a group may be distributed all over the Internet, operating under different ISPs, and using different ESPs. But we assume here that they can all authenticate their group membership, and their name, via a specified CA (or a set of CAs). Under this assumption, we consider the following sending law which we call $GC$ (for "group communication").

> **The $GC$ sending-law:** Only members of $G$ can send messages under this law, and each message will identify its sender via its certified name.

A member $r$ of group $G$ can accept TSR-based $\mathcal{GC}$-messages from any sender $s$, because he can be confident that $s$ is a member of $G$, since such messages can only be sent by members of this group. And it is worth pointing out, that to gain this confidence there is no need for the sender $s$ to authenticate himself to $r$ as a member of $G$, it is sufficient for $s$ to authenticate himself to his controller $T^{\mathcal{GC}}_{s}$. This is important because if the receiver $r$ of such a message would require every sender to authenticate itself to him as a member of $G$, it would need to know the public key of every member of this group, which is not scalable with the size of the group. (Note that it is possible to write a law which is parametrized by the identity of the group, so that people belonging to many different groups can use essentially the same law for communication with their groups.)

### 2.3.5 Hybrid Laws

Each of the laws discussed above represents a more or less single strategy about spam reduction. One can also have hybrid laws that combine several such strategies.

For example, note that law $\mathcal{IO}$ above is vulnerable to spammers who decide to send large numbers of introductory messages, despite the limitation imposed by this law on the structure and length of such messages. One can defend against such an attack by requiring a payment for such message, via stamps, as under law $\mathcal{PP}$ above. And if introductory messages are intended to be used sparingly, for people to introduce themselves to old lost friends, or to potential friends, one can make the payment for each stamp fairly substantial, say \$1.

Second, it makes much sense to add the opting in and out provisions of the $\mathcal{IO}$ law to any of the other laws we considered, perhaps after an appropriate adaptation. As an example, consider doing so with the $\mathcal{GC}$ law of group communication. Given the fact that this law deals with communication between somehow related people, one may start with a default of a universal *opt in* state. So that initially everybody in the given group would be able to send messages to everybody else, while allowing group members to opt out, and perhaps back in, at their discretion.

## 2.4 Spam Reduction via Incremental Deployment of TSR-Based email

Assuming that the controllers designed for mediating TSR-email are provided broadly over the Internet, we claim that it is conceivable (if not likely) that the following interdependent trends would progressively develop, resulting in a significant reduction of spam traffic over the Internet, and of unwanted emails that individual users have to contend with. And such reduction of spam is expected to be achieved without incurring the undesirable side effects of conventional anti-spam measures, like the blocking of valid mail. These trends, which need to develop concurrently, feeding on each other, are the following:

- Several TSR-laws will become popular for their immunity from spam, at least from the viewpoint of some classes of users. We will refer to such popular immune laws as *i-laws.* And a standard will develop for publishing these laws, identifying each of them via unique id, and enabling users to download any i-law they want, to be used for sending or receiving of TSR-email. This standard would also provide for documentation of each law, and for public discussion of its merits and limitations.

- Substantial number of users will choose an i-law (or several of them) for preferential treatment, such as admitting messages sent under their preferred laws directly to their inbox, skipping filtering. Alternatively, messages sent under a preferred i-law may be subjected to some kind of lightweight filtering.

  And standards will develop for publishing the preferred i-laws of users, along with their email addresses.

- Email users would increasingly employ TSR for sending their messages, subject to one of the laws declared as preferred by each destination.

There are two reasons to believe that these trends would materialize, resulting in widespread deployment of TSR-based email, once the ability to send TSR-messages is made available broadly over the Internet. First, using TSR-email is a win-win proposition for both the receivers of email and for senders. Receivers would benefit by giving preferential treatment to email sent subject to email-laws that provide them with immunity from spam, without incurring adverse effects, such as false-positives that mar filtering. And the senders would benefit by sending TSR-email, subject to the laws preferred by those they send their email to, because by doing so they increase the chance for their email to be accepted.

Second, as we will see in Section 4, neither the senders of TSR-messages, nor their receivers, need to commit themselves exclusively to TSR-based email. The senders can continue sending conventional, non-TSR, email, via the same MSAs they use for sending TSR email; and the receivers can accept conventional emails via the same MDA used for TSR email. So, engagement in TSR-based communication can be done incrementally.

But how does one jump-start these trends? Perhaps the best way to do so is for TSR-based email to be adopted by a significant community of users, for communication within the community. Such a community may be a consortium of universities, or of businesses; the US federal government; the US military, and the likes. And from such an initial core group of users, TSR-email may spread out in an incremental manner to broader set of email users.

### 2.4.1 Making TSR into a Standard of Email Communication

So far we have assumed that the sending and receiving of TSR-email is entirely voluntary, and done as a complement of standard email. But if and when TSR-email proves itself as useful, and becomes common, it is not inconceivable for it to be made into the standard, which may make it even more effective. This may be done, broadly, as follows.

First, the set of i-laws would be organized into what is called, under LGI, a *conformance hierarchy* [2] of laws. The root $\mathcal{L}_0$ of this hierarchy would become the global law of email-sending. But the hierarchy of i-laws would include any number of laws defined as subordinate to $\mathcal{L}_0$, and thus—due to the structure of LGI law-hierarchies—would automatically conform to the root law $\mathcal{L}_0$. Second, the SMTP protocol would have to be changed, requiring all messages to be sent via TSR, subject directly to the root law $\mathcal{L}_0$, or to one of its subordinate laws, that conform to $\mathcal{L}_0$.

Making TSR into a standard would not entirely eliminated the need for filtering, but it might end the ongoing *arm race* between filtering and spamming.

## 3 The Law-Governed Interaction (LGI) Mechanism—an Overview

This section is not absolutely necessary for the understanding of the rest of this paper; it is introduced here mostly for the sake of completeness.

Broadly speaking, LGI is a regulatory mechanism that enables an open and heterogeneous group of distributed *actors* to engage in a mode of interaction *governed* by an explicitly specified and strictly enforced policy, called the *law* of this group. By "actor" we mean an arbitrary process, whose structure and behavior is left unspecified; and an actor engaged in an LGI-regulated interaction, under a law $\mathcal{L}$, is called an $\mathcal{L}$-*agent*. LGI thus turns a set of disparate actors, which may not know or trust each other, into a *community* of agents that can rely on each other to comply with the given law $\mathcal{L}$. This is done via a distributed collection of generic components called *private controllers*, one per $\mathcal{L}$-agent, that are trusted to mediate all interaction between these agents, subject to the specified law $\mathcal{L}$ (as illustrated in Figure 1).

All told, LGI goes well beyond conventional access control, in its ability to cope with the increasing size, openness, and heterogeneity of distributed systems. It is, in particular, inherently decentralized, and thus scalable even for a wide range of stateful policies. And it is very general. A prototype of LGI has been recently released, and has a growing community of users.

This section provides only a very brief overview of LGI. For more information, the reader is referred to the LGI tutorial and manual [9], and to a host of published papers (see [10, 1, 2] in particular).

**Agents and their Private Controllers:** An $\mathcal{L}$-agent $x$ is a pair $x = \langle A_x, T_x^{\mathcal{L}} \rangle$, where $A_x$ is an *actor*, and $T_x^{\mathcal{L}}$ is its *private controller*, which mediates the interactions of $A_x$ with other LGI-agents, subject to law $\mathcal{L}$. Each controller $T_x^{\mathcal{L}}$ maintains the *control state* (or, "cState") of agent $x$, which is some function of the history of interaction of $x$ with other community members. The nature of this function, and its effect on the ability of $x$ to communicate, is largely defined by the law $\mathcal{L}$. The concept of law is defined in the following section. The role of the controllers is illustrated in Figure 1, which shows the passage of a message from an actor $A_x$ to $A_y$, as it is mediated by a pair of controllers, first by $T_x^{\mathcal{L}}$, and then by $T_y^{\mathcal{L}}$—both operating, in this case, under the same law, although interoperability between different laws is supported by LGI as well. One of the significant aspects of such mediation is that under LGI every message exchange between LGI-agents involves dual control: on the sides of both the sender of a message, and of its receiver. Note however that this is not the case when an LGI-agents interact with an actor not operating under LGI, as in the case of TSR—more about which in Section 3.5.

It should be pointed out that private controllers are actually hosted by what we call *controller pools*—each of which is a process of computation that can operate several (typically several hundreds) private controllers, thus serving several different agents, possibly subject to different laws. (Henceforth we will often refer to controller pools as "controllers," expecting the resulting ambiguity to be resolved by the context.) The set of controller-pools available to a given application (or a set of application) is called a *controller service* or CoS.

## 3.1 The Concept of Law Under LGI

An *LGI law* (or, simply, a *law*) is defined in terms of three elements: (a) a set $E$ of *regulated events*; (b) a set $O$ of *control operations*; and (c) the *control-state* ($CS_x$) associated with each agent $x$. More specifically, $E$ is the set of events—such as the sending and arrival of a message—that may occur at any agent, and whose disposition is subject to the law. $O$ is the set of operations that can be mandated by a law, to be carried out at a given agent, upon the occurrence of regulated events at it. In a sense, these operations constitute the *repertoire* of the law—i.e., it is the set of operations that the law is able to mandate. This set includes operations like forwarding a message, and updating the state of a given agent. Finally, the *control-state*, or simply the state, of an LGI agent is the state maintained by the controller of this agent agent, which is distinct from the internal state of the actor of

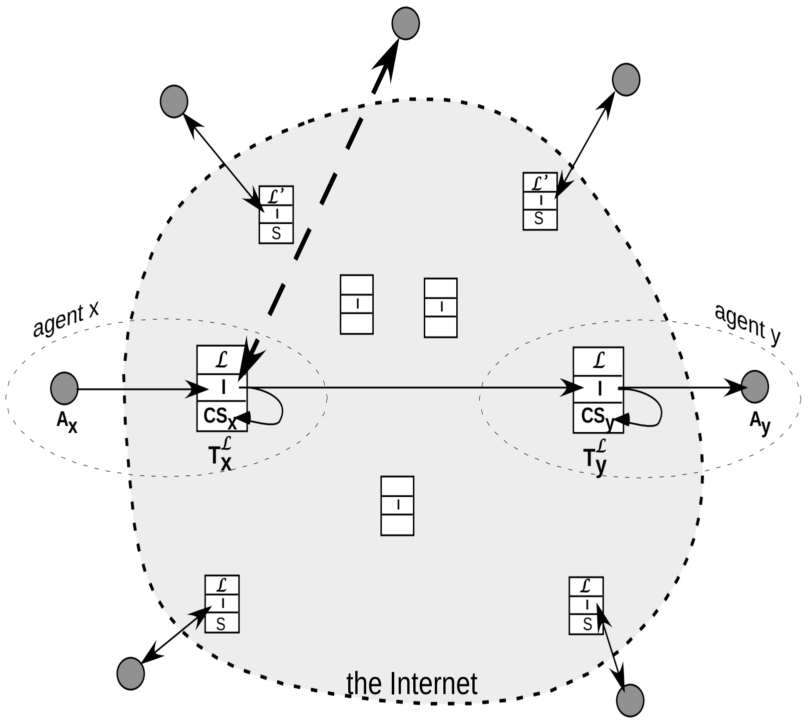

Figure 1: Interaction via LGI: Actors are depicted by circles, interacting across the Internet (lightly shaded cloud) via their private controllers (boxes) operating under law L. Agents are depicted by dashed ovals that enclose (actor, controller) pairs. Thin arrows represent messages, and thick arrows represent modification of state. Finally, a thick dashed arrow represents communication between an LGI-controller and an actor not operating under LGI. **This is the type of communication used in this paper for email.**

that agent. This state, which is initially empty, can change dynamically in response to the various events that occur at it, subject to the law under which this agent operates.

Now, The role of a law under LGI is to decide what should be done in response to the occurrence of a regulated event at an agent operating under this law. This decision, which is called the *ruling of the law*, consists of a sequence of zero or more control operations from the set $O$. More formally, a law is defined as follows:

**Definition 2 (law)** *Given a set $E$ of all regulated events, a set $O$ of all control operations, and a set $S$ of all possible control-states, a law $\mathcal{L}$ is a function: $\mathcal{L} : E \times S \rightarrow O^*$*

In other words, *a law maps every possible $(event, state)$ pair into a sequence of zero or more control operations, which constitute the* ruling *of the law.*

Note that this definition does not specify a language for writing laws. This for several reasons: First, because despite the pragmatic importance of choosing an appropriate law-language, this choice has no impact on the semantics of the model itself, as long as the chosen language is sufficiently powerful to specify all possible functions of the form of Definition 1. Second, by not specifying a law-language we provide the freedom to employ different law-languages for different applications domains, possibly under the same mechanism. Indeed, the implemented Moses mechanism employs two different law-languages, one based on the logic-programming language Prolog, and the other based on Java.

**The Local Nature of LGI Laws, and their Global Sway:** Our concept of law differs structurally from the conventional concept of AC policy. One important characteristic of LGI laws is that they are inherently local. Without going into technical details, locality means that an LGI law can be complied with, by each member of the community subject to it, without having any direct information of the coincidental state of other members. This locality is a critical aspect of LGI for two major reasons: First, because locality is necessary for decentralization of law enforcement, and thus for scalability even for stateful policies. And second, because locality facilitates interoperability between different laws, and enables the construction of law-hierarchies, as has been shown in [2].

Remarkably, although locality constitutes a strict constraint on the structure of LGI laws, it does not reduce their expressive power, as has been proved in [9]. In particular, despite its *structural locality*, an LGI law can have *global effect* over the entire $\mathcal{L}$-community—mostly because all members of that community are subject to the same law—and can, thus, be used to establish *mandatory*, community wide, constraints.

### 3.2 On the Basis for Trust Between Members of a Community

For an $\mathcal{L}$-agent $x$ to trust its interlocutor $y$ to observe law $\mathcal{L}$, it is sufficient for $x$ to have the assurance that the following three conditions are satisfied: (a) the exchange between $x$ and $y$ is mediated by correctly implemented private controllers $T_x$ and $T_y$, respectively; (b) both controllers operate under law $\mathcal{L}$; and (c) the $\mathcal{L}$-messages exchanged between $x$ and $y$ are transmitted securely over the Internet.

The first of these conditions is the hardest to satisfy. The other two condition are straightforward. To ensure condition (b), that is that the interacting controllers $T_x$ and $T_y$ operate under the same law, LGI adopts the following protocol: When forwarding a message, a controller, say $T_x$, appends to it a *one way hash* `H` of its law. The controller of the interlocutor, $T_y$ in this case, would accept this as a valid $\mathcal{L}$-message only if `H` is identical to the hash of its own law. Of course, such an exchange of hashes of the law can be trusted only if condition (a) is satisfied. Finally, to ensure the validity of condition (c), above, the messages sent across the Internet—between actors and their controllers, and between pairs of controllers—should be digitally signed and encrypted. These conventional, but rather expensive, measures have not been employed in the current implementation of LGI, but they are easy to deploy.

### 3.3 Other Features of LGI, and its Performance

We will list here some of the notable features of LGI, which we were not able to discuss in this short overview, and will provide references to them for the interested reader. These features are: (1) the concept of *enforced obligation*, that provides LGI with important proactive capabilities; (2) the treatment of *exceptions*, which provides LGI with fault tolerance capabilities; (3) the treatment of *certificate*, which is obviously necessary for the regulation of distributed computing; and (4) *interoperability* between different laws. All this are discussed in the LGI Manual [9]. In addition, the important concept of conformance hierarchy between laws, and another look at interoperability, are provided in [2].

Finally, we point out that the performance of LGI is discussed in [9]. In a nutshell, the overhead due to the LGI mediation is between 30 and 100 microseconds, for the types of laws we used in most of our studies.

### 3.4 A Controller Service (CoS)

We have created what we call a *controller service* (CoS) that maintain and operate a distributed and trustworthy collection of generic LGI-controllers, which can be adopted for operation under any valid law. This set of controllers constitute *distributed trusted computer base*, or DTCB, of LGI, which replaces the traditional concept of TCB.

Such a controller service can be operated by a given organization for its own internal use; or even by a federation of organizations, for the use of its members. Note, however, that in order to use LGI all over the Internet—which is necessary for many applications, but in particular for TSR-based email—one needs a commercial company, such as Cisco, google, Microsoft or IBM, to operate such a CoS for profit. This has not been done yet, although there are no serious technical impediments to it. It should also be pointed out that there is a work underway to further enhance the security of the controller, in particular, via TPM technology. But even our current CoS structure should be sufficient for many applications, such as for email.

It should be pointed out that the deployment and maintenance of a trustworthy CoS that is widely available over the Internet, is a formidable proposition, which is not likely to be carried out only for anti-spam purposes. But LGI has a very wide range of applications, such as securing B2B commerce, and supporting the governance of SOA-based enterprise systems, and of the of grid-like federations of institutions (see [9] for a review of some of these applications). It is for the sake of this type of critical applications that a trustworthy controller service may end up being deployed over the Internet. And if and when such a service is deployed, it would be usable for TSR-base emailing as well.

### 3.5 On the Use of the Full Power of LGI by TSR-based Email

The TSR mechanism, as described in this paper, does not use the dual control—on the sides of both the sender of a message, and of its receiver—provided by LGI for each message exchange. This very important aspect of LGI is not being used here because it seems that for email exchange there is no serious need to maintain a state on the receiver side, and thus no need for controller on the side of the receiver.

However, duel control can be used to regulate the message exchange between the sender of an email, and other agents with which it interacts via regular (non-email) messages, such as the postage vendors in the case of the $\mathcal{PP}$-law described in Section 2.3.1. Also, dual LGI control would be useful for regulating the sending of email messages by a given LGI-community of agents. But such use of dual control in the context of email is beyond the scope of this paper.

## 4 On the Implementation of TSR-Email

This section is a schematic description of the implementation of TSR-based email, employing a slightly modified version of the LGI mechanism. This implementation requires no change in the SMTP protocol. But it involves replacing the MSA (Mail Submission Agent), MDA (Mail Delivery Agent) and (MUA) (Mail User Agent)—serving those who wants to send and receive TSR-emails—with modified versions, denoted by MSA', MDA', and MUA', respectively. The updates in question would incorporates appropriate treatment of TSR-email, along with the conventional treatment of regular email by these agents.

Also, we assume here the existence of a *controller service* (CoS) that maintain a set LGI-controllers—upgraded with the ability to communicate via SMTP—which is trusted broadly by the prospective users of this type of email.

We distinguish between three phases of TSR-emailing: (1) the initial phase, which is what a sender needs to do before sending $\mathcal{L}$-messages under any given law $\mathcal{L}$; (2) the email exchange phase, which deal with the passage of a TSR-email from its sender to its destination, along with the possible replies to such a message; and (3) the background phase, which consists of possible activities that may be carried out, in the background of the message exchange, depending on the law at hand.

The following discussion of these phases—which is based on an experimental implementation of TSR—is rather schematic, but the missing details are easy do supply, and it can be done in a variety of ways. Our discussion below is illustrated by Figure 2. (Note that the numbered arrows representing messages, and other actions, belonging the email exchange phase are solid, while those belonging to the initial and background phases are dashed.)

**The Initial Phase:** Before sending any TSR-emails subject to law $\mathcal{L}$, an email user Alice must send a regular (TCP/IP) message (arrow #1) to one of the generic controllers maintained by the CoS, to *adopt* it as her mediator for sending $\mathcal{L}$-emails. From the LGI viewpoint, this adoption creates a new $\mathcal{L}$-agent whose controller is $T^{\mathcal{L}}_{Alice}$ and whose actor is Alice herself (or, more precisely, her MUA'). (Note that Alice can adopt any number of controllers, to operate under different laws, which can be used by her to send TSR-emails under these laws.)

What actually happens during the creation of $T^{\mathcal{L}}_{Alice}$ depends on the law $\mathcal{L}$, under which it has been adopted. Here are some examples. (i) Under our group communication law $\mathcal{GC}$ (cf. Section 2.3.4), the controller would attempt to authenticate Alice as a member of the group of radiologists which it is designed to serve; and it would record the official name of Alice in this group in the control-state of $T^{\mathcal{GC}}_{Alice}$, in order to add it to messages sent by Alice. But if Alice would fail such authentication the controller would self destruct, preventing Alice from sending messages under this law. (ii) Under the payed postage ($\mathcal{PP}$) Law, introduced in Section 2.3.1, the controller may be required by its law to purchase an initial set of stamps on behalf of Alice.

And there are other types of initializing action that the controller $T^{\mathcal{L}}_{Alice}$ may perform at this stage. A highly recommended action to be required by almost any sending law would have the controller acquire from the sender—Alice in this case—a password, to be use by her MSA' to authenticate itself to the controller as the one working on behalf of Alice. This would make it much harder for attackers to impersonate Alice by sending message through different MSAs. As another example, consider a type of sending law $\mathcal{L}$ that prohibits any given sender to operate via more than one controller at a time. To ensure such uniqueness, controller $T^{\mathcal{L}}_{Alice}$ would report its adoption by to a designated registry, which should be programmed to kill this controller, if Alice already has a controller operating under law $\mathcal{L}$.

**The Exchange of TSR-emails:** Assuming that Alice already adopted a controller $T^{\mathcal{L}}_{Alice}$, as discussed above, we describe now the passage of an $\mathcal{L}$-email, for an arbitrary law $\mathcal{L}$, sent by Alice to Bob (denoted by thick solid arrows in Figure 2); along with a possible reply by Bob (denoted by thinner arrows). This message exchange is described below as a time sequence of steps.

a. Alice sends her email to her MSA' (arrow #2), specifying in its header the address of the controller $T^{\mathcal{L}}_{Alice}$ that is to be used for handling this message. This message may also contain a password intended to be used by the MSA' to convince the controller that it works on behalf of Alice. (Such a password identification would be carried out by the

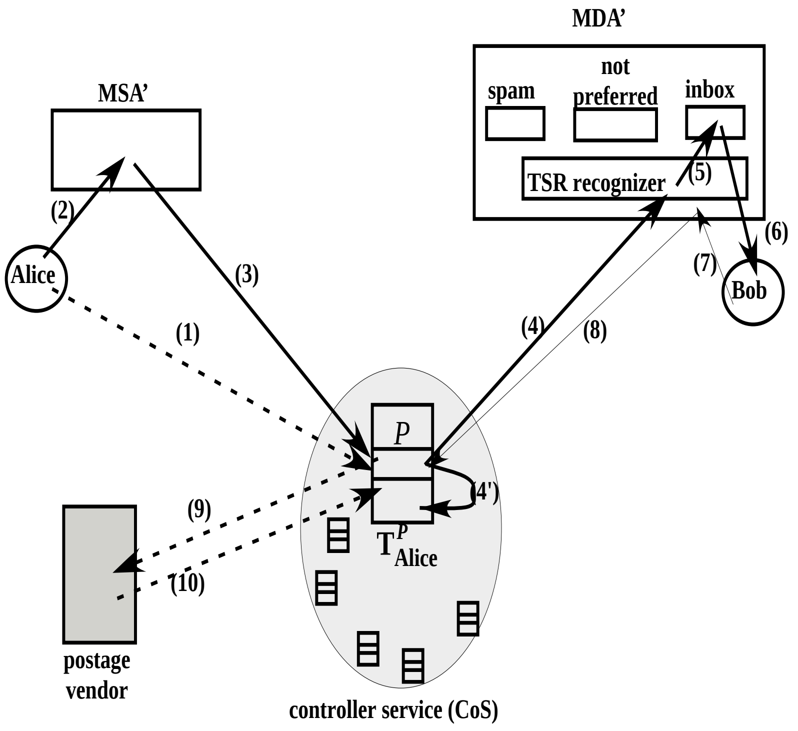

Figure 2: TSR-Based Email

controller if it is so mandated by law $\mathcal{L}$ at hand).

b. The MSA' of Alice operates mostly as a conventional MSA, but it sends this email to controller $T^{\mathcal{L}}_{Alice}$ (arrow #3), along with the email-address of Bob.

c. When the controller $T^{\mathcal{L}}_{Alice}$ gets a message from the MSA' of Alice, it would respond in the manner mandated by the law $\mathcal{L}$ under which it operates. Here are some examples of the kind of things $T^{\mathcal{L}}_{Alice}$ may do under various laws upon receiving the sent message. (i) Under the paid postage law $\mathcal{PP}$, the controller would forward the sent email only if the number of stamps it has in its state is greater than zero, consuming one of its stamps. (The sending of a message to Bob is depicted by arrow #4; the consumption of a stamp, done by updating the state of the controller, is depicted by arrow #4') (ii) Under the rate limiting law $\mathcal{RT}$, the sending of the message to Bob may be delayed as required by the law.

d. The MDA' of Bob will determine if the message it got is a TSR-message, and if so, it will identify the law under which it has been sent. This identification, carried out by what is depicted as *TSR recognizer* in Figure 2, is the essence of the TSR capability provided by LGI. It is done, essentially, by verifying the authenticity of $T^{\mathcal{L}}_{Alice}$, by examining the certificate signed by the CA associated with the CoS, and by examining the hash of the law, both of which are attached to any message sent by an LGI controller.

   If the incoming message has been found to be a $\mathcal{L}$-email, and if $\mathcal{L}$ is one of the preferred laws of Bob, then the message will be placed in the inbox of Bob (arrow#5), although Bob may decide to subject it to some kind of lightweight filtering as well. If the incoming message has been found to be a TSR-message but subject to a law not preferred by Bob, or if it is a conventional, non-TSR message, then it may submitted to the conventional filtering process, and may end up being placed either in the spam folder, or in the folder we call here "not preferred"—although other treatments of such messages are conceivable.

   It is important to emphasize that a MDA' should be able to handle conventional, non-TSR messages, although it may be built not to place them in the inbox of its client, as described above.

e. Bob's reply to an $\mathcal{L}$-email he got from Alice is carried out via a regular—non-TSR—message; but it is not sent directly to the sender Alice, but to her controller $T^{\mathcal{L}}_{Alice}$ (see arrows #7 and #8). We distinguish between two kind of such replies: *regular reply* and *control reply*.

   A regular reply is an email intended to the original sender of message $m$—Alice in this case—and possibly to those CCied on this message. It is the job of controller $T^{\mathcal{L}}_{Alice}$ to forward the reply to these users, subject to law $\mathcal{L}$.

   A control reply is intended to affect the state of the controller $T^{\mathcal{L}}_{Alice}$ itself. For example under the opt in/out law $\mathcal{IO}$, introduced in Section 2.3.3, the opting in and out would be carried out via such a control reply.

**Background Activities of the Controller:** Finally, a controller $T^{\mathcal{L}}_{Alice}$ may engage in various activities mandated by its law, during the process of message sending. For example, the paid postage law $\mathcal{PP}$ may be written to instruct the controller to replenish its supply of stamps when only few of them remained in the controller's state. Such autonomous activity by the controller is represented by arrows #9 and #10 in Figure 2.

## 5 Conclusion

The TSR-based email mechanism proposed in this paper, enables the receiver of an email message to recognize the sending protocol that generated it. This mechanism aims to induce email users to voluntarily send their messages subject to the spam-immune sending protocols preferred by their destinations, thus producing less spam.

TSR communication involves no text-based filtering, no dependency on blacklistings, and no coercion by ISPs. Yet, if widely deployed, we claim that the TSR-based email would result in a significant reduction of the traffic of spam over the Internet, and of unwanted emails that individual users have to contend with. And these results would be achieved without incurring the undesirable side effects of conventional anti-spam measures, like the blocking of valid mail by filtering, and by coercive measures imposed by ISPs.

TSR-based emailing can be deployed incrementally, in conjunction with, and as a complement to, the conventional anti-spam measures, involving no changes to the SMTP protocol. But, as has been pointed out briefly in Section 2.4.1, TSR can eventually be made into a standard that would imposed some global sending law $\mathcal{L}_0$ over all emails, while allowing users the freedom to choose—for the sending or receipt of email—any law defined as subordinate to $\mathcal{L}_0$ in the LGI hierarchy of laws, and thus conforming to it. Making TSR into a standard would not entirely eliminated the need for filtering, but it might end the ongoing *arm race* between filtering and spamming.

Finally, it should be pointed out, again, that the deployment and maintenance of a trustworthy controller service (CoS) that is widely available over the Internet, is a formidable proposition, which is not likely to be carried out only for anti-spam purposes. But LGI has a very wide range of applications, such as securing B2B commerce, and supporting the governance of SOA-based enterprise systems, and of the of grid-like federations of institutions. It is for the sake of this type of critical applications that a trustworthy controller service may end up being deployed over the Internet. And if and when such a service is deployed, it would be usable for TSR-base emailing as well.

## References

[1] X. Ao, N. Minsky, and V. Ungureanu. Formal treatment of certificate revocation under communal access control. In *Proc. of the 2001 IEEE Symposium on Security and Privacy, May 2001, Oakland California*, May 2001.

[2] X. Ao and N. H. Minsky. Flexible regulation of distributed coalitions. In *LNCS 2808: Proc. European Symp. on Research in Computer Security (ESORICS)*, Oct. 2003.

[3] J. Goodman, G. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *CACM*, 50:25–33, 2007.

[4] Amir Herzberg. *Cryptographic Protocols to Prevent Spam.* Prentice-Hall, 2005.

[5] Bogdan Hoanka. How good are our weapons in the spam wars? *IEEE Technology and Society Magazine*, Spring 2006.

[6] N. Leavitt. Vendors fight spam's sudden rise. *Computer*, February 2007.

[7] L Lessig. *Code, Version 2.0.* Basic Books, 2006.

[8] N. H. Minsky. Regularity-based trust in cyberspace. In *In LNCS 2692: the Proceedings of the First International Conference on Trust Management, Crete, Greece*, May 2003. (also available at `http://www.cs.rutgers.edu/~minsky/pubs.html`).

[9] Naftaly H. Minsky. *Law Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction, and a Reference Manual)*, February 2006. (available at `http://www.moses.rutgers.edu/ documentation`).

[10] Naftaly H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, July 2000.

[11] Sebo Tladi. The regulation of unsolicited commercial communications (spam) : is the opt-out mechanism effective? *Sabinet Online Journal*, 1(4):178–192, 2008.

[12] Z. Zhong, K. Huang, and K. Li. Throttling outgoing spam for webmail services. In *Proc. of Second Conference on Email and Anti-Spam (CEAS-05)*, July 2005.