Internet Draft
Document: draft-bonatti-generic-antispam-00
Proposed Category: Experimental

Chris Bonatti
IECA, Inc.

12 May 2004

A Generalized Mechanism for Control of Unwanted Application Communications

STATUS OF THIS MEMO

ABSTRACT

This draft describes a new anti-spam technique that could be applied to e-mail or (in principle) any push-mode application. It includes a discussion of problem background, a description of the proposed technique, and an analysis of the effectiveness of the approach.

1. INTRODUCTION

This draft describes a generic mechanism that can be incorporated into Internet applications to allow application user agents (UAs) to automatically separate legitimate from fraudulent communications for the purpose of facilitating selective filtering mechanisms. This mechanism might, for example, be incorporated in electronic mail (e-mail) UAs or domain gateways to aid in the rejection of spam. If used on a widespread basis, this technique has the potential to dramatically reduce the volume of spam reaching users. Thus deprived of recipients, spammers will shift to other, more profitable means of advertising. This mechanism could

likewise be applied to other push-mode applications (e.g., instant messaging, VoIP) to prevent undesirable communications.

1.1 Problem Description

Unwanted bulk e-mail, or spam, is regarded as the Internet plague of the early 21st century. To date, the e-mail industry has dealt rather poorly with the spam threat. Many halfway measures have been instituted that have been largely ineffective at stemming the tide, but which have caused a lot of pain and angst among users. If you've ever tried traveling and plugging your laptop's e-mail into local service providers, you know something of this pain. Internet mailing lists are now frequently moderated, and have controlled submission because of spam. This adds dramatically to the effort required to maintain a list, and detracts from its functionality. Filters applied either at receiving UAs or at Message Transfer Agents (MTAs) provide some spam relief, but are often unreliable because of the frequent occurrence of invalid or inaccurate header information. Newer filters based on the content of the message offer some promise, but these have resulted in a sort of "arms race" between filter vendors and spammers, with each trying to gain the upper hand.

Open relay was never the problem. Mailing lists were never the problem. Yet we took steps to hobble both. Filters are never going to be wholly effective because they are trying to analyze ever-changing fraudulent headers and body data. Establishing control over the set of originators from which recipient domains will elect to receive mail is the real problem. Only by addressing this problem directly will we manage to curb spam.

Furthermore, the definition of what is spam lies solely with the user. As has been occasionally noted, one man's spam is another man's ham. However, the average user today does not take advantage of even the limited control they might have over the problem, via receive-side filters etc. Most just want the problem to "go away". Another way to express this is that they want service providers to block spam without the added complexity that user control implies. However, most server-side filtering leads to significant rates of false positive and false negative spam detection. So any realistic solution must operate by default without much user input, and yield a reduced rate of false spam detection.

Certainly there are many existing techniques that would facilitate giving the recipient better control over the originators from which messages will be received. The Secure Multipurpose Internet Mail Extensions (S/MIME) standards, as well as Open Pretty Good Privacy (OpenPGP), are capable of establishing strong authentication of the actual originator. Other technologies such as Transport Layer Security (TLS) and Internet Protocol Security (IPsec) are capable of providing strong authentication between application layer entities. These could be used to indirectly provide assurance of the originator identity and return path. However, all of these techniques require deployment of strong cryptography and some form of Public Key Infrastructure (PKI). Years of PKI deployment history suggest that deploying any of these technologies in a ubiquitous enough manner to support anti-spam measures is virtually impossible. A simpler, more self-contained solution is required to achieve the widespread degree of implementation necessary.

Several characteristics emerge as requirements for a prospective simple and self-contained spam-blocking mechanism. Such a solution must enable recipients or recipient domains to reliably reject unsolicited message if they so choose without breaking the existing e-mail infrastructure. The solution may assume that most users have relatively small sets of partners with whom they exchange e-mail on a regular basis. It may assume that most users do not have a frequent requirement to receive unsolicited e-mail from unknown parties. Most importantly, the solution may assume that spammers will not be able to access message sent to spoofed originator addressed. This represents the Achilles heel of most spam schemes.

1.2 Architectural Context

The key architectural advantage that the Internet has exhibited over the years is based on the principle of clustering complexity at the edge of the system, while keeping the core infrastructure as simple as possible. This "simple core" principle offers advantages in scalability and interoperability. The principle has proven its value in the deployment of TCP/IP suite, and the widespread deployment of SMTP. However, most of the anti-spam measures to date have attacked the problem by modifying and complicating the e-mail core system. This leads to challenges in policing the uniform deployment of features, and leads to more complex sets of failure modes. Any effective anti-spam technology must embrace the "simple core" principle by pushing the complexity as far outside the core infrastructure as possible.

In the case where anti-spam filtering takes place in the recipient UA, the complexity is as close to the system edge as possible. In this case, the mechanism must be implemented locally to the UA and benefits only a single user. This configuration is shown in the figure below.

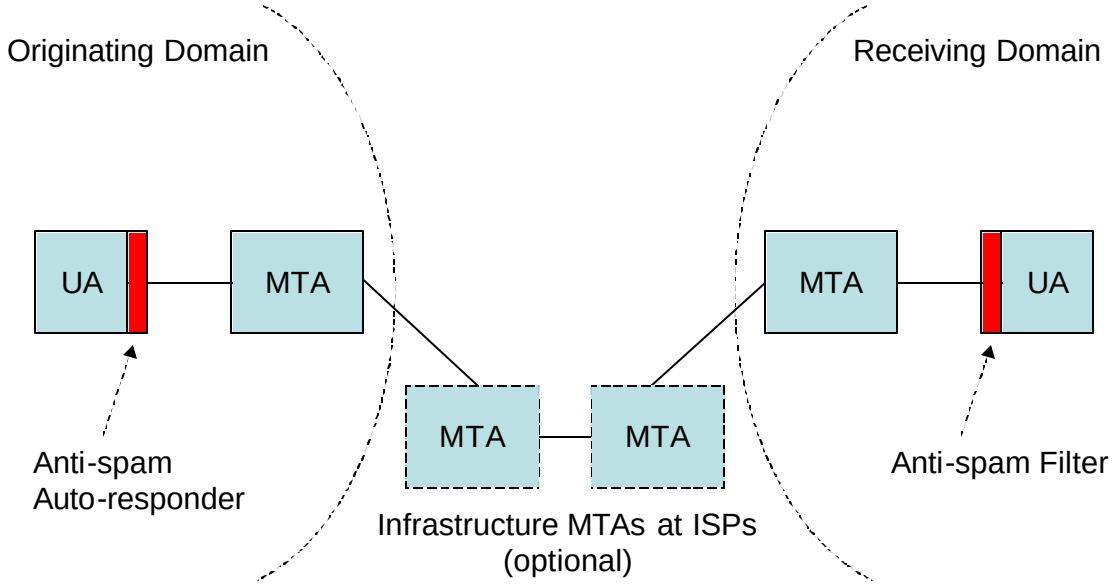

Figure 1 – Individualized Spam Protection

In the case where anti-spam filtering serves an entire recipient domain, the complexity affects the gateway or MTA components of the recipient domain. The mechanism has the

capability to provide benefit to the entire receiving domain. However, the originating UAs will need to implement any aspects of the mechanism individually in order to maintain individual level authentication. This configuration is shown in the figure below.

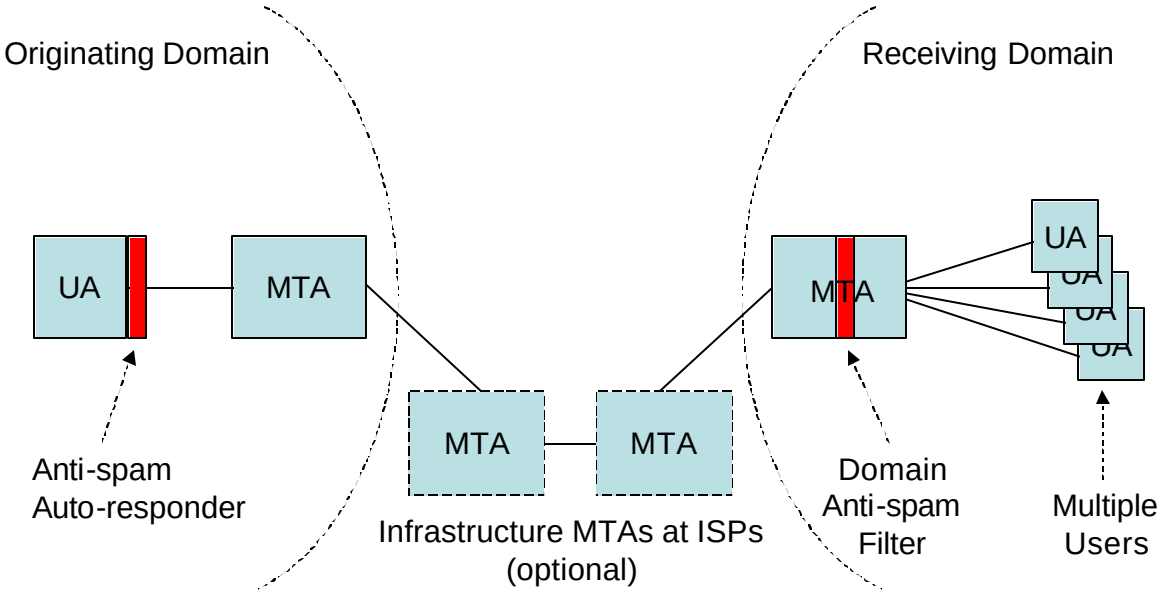

Figure 2 – Domain Spam Protection

1.3 Threat Environment

The extent of the threat against any potential anti-spam technology is increasingly high. Offshore mass e-mailing firms are reputed to be retaining freelance hackers and crackers to enhance the capability of their messages to penetrate filters. These potential attackers are able to bring a high level of analytical sophistication to bear in attacks upon any anti-spam technologies. For example, recent efforts to limit spam through deployment of "Baynesian" smart content filters have been defeated by spammers using a combination of statistical modeling and inert keyword padding. This level of sophistication is fueled by a strong profit motive. Regardless of how many users are offended by spam, a finite number of recipients will respond. Given a sufficiently large recipient list this is sufficient to justify moderate expenditure on the part of the spammers to preserve their "advertising" revenue stream.

Attacks that might be mounted by spammers are multifold. Not only is the spammers main product a form of attack, but domains and organizations perceived to be acting against the interests of hackers and crackers have been specifically targeted for Internet Protocol (IP) Denial of Service (DoS) and other network layer attacks. In this paper, however, we will constrain our concern to variants of attack via the main threat vector; namely unwanted application communications. The main attack variants within this set include:

- Impersonation of an invalid source address – This is the most common class of communications, where the indication of originator is set to some invalid value merely to mask the true originator's identity.
- Impersonation of a valid, but unknown source address – This is also fairly common attack, whereby spammers will randomly employ valid by incorrect values for the originator based on previously harvested addresses. This will enable the originator to pass a validity check in the DNS.
- Impersonation of a valid, and known source address – Same as above, except that the address used is known to the target. This may enable the attack to pass a list-based filter mechanism.
- Impersonation of the recipient's own address – This is a blind spot to many filter mechanisms, but are usually readily detectable by the user.
- Targeted non-delivery notifications – In this technique the spammer sends a message to an invalid address in a valid domain, and impersonates the true target of the attack as the originator. This results in a non-delivery notification being sent from a valid server to the target, often containing the spammer's original message.
- Spam beacons – Many unwanted communications contain executable code or hyperlinks that can alert the attacker of the successful communication, or attempt to gain access to other information.
- Malicious code dissemination – Malicious code dissemination is often commingled with other unsolicited communications, compounding the detection problem.
- Malformed protocols – Keywords of header fields or HTML tags are sometimes deliberately malformed in order to avoid detection yet elicit a predictable behavior by the receiving system.
- Keyword obfuscation – Keywords in the content of the communication are misspelled, thereby evading filter mechanisms.
- Inert keyword padding – Inert (e.g., frequently invisible) text includes lists of keywords specifically formulated to make the communication fit the profile of a legitimate communication, thereby defeating statistical analysis filters.

While the spammers' revenue stream provides the source of their analytical sophistication, it is also a key weakness that can be turned against them. Spammers are able to milk a relatively healthy revenue stream from their clients because the cost of their operations are underwritten by the vast infrastructure of the Internet. Internet Service Providers (ISPs) bear a particularly heavy portion of that burden. However, like traditional advertisers, spammers must demonstrate to their clients a certain level of return for their fees. If spam filters can sufficiently reduce the size of the audience for a spammer, the reduction in the spammer's level of return will cause the revenue stream to dry up and make the enterprise unprofitable. This

means that even although spam filtering at the edge is not effective in blocking spam traffic in the infrastructure , it should result in a reduction in the level of spam traffic via feedback effects.

Despite their sophistication, spammers suffer from relative scarcity of resources. Their profit margins are entirely based on low cost overhead, so they generally lack the "big iron" necessary to attack cryptographic systems.

However, the "Achilles heel" of spam is the desire of the perpetrators to maintain their anonymity. This forces them to spoof the originator address, making deliberate attempts at reverse communication fail. This common denominator to the attacks can be exploited to formulate a solution.

2. THE SOLUTION

The solution to this situation from, an architectural standpoint, is to embed an access control decision function in the application code to automatically manage whether or not delivery of each communication will be permitted. This aspect of the solution is not unique, but resembles the e-mail filtering capability already embedded in many UAs. However, as spam's Achilles heel is the spoofing of the originator address and other e-mail headers, we can dramatically improve the effectiveness of this access control function by incorporating a rudimentary handshake process. This handshake process must have the following properties:

- It must bring result in a rate of erroneous denials as close as possible to zero.
- It can assume that the spammer does not have access to legitimate users' mailboxes.
- It must be sufficiently strong to resist moderate attack from cryptographically savvy programmers.
- It must not require a large infrastructure to support its operation.
- It must pass the "grandmother test", in that it requires sufficiently little attention that anyone can operate it.

2.1 Handshake Procedure

The proposed solution offers a simple handshake that satisfies all of these conditions. It will allow recipients (or receiving domains) to require the presence of a hashed token in their messages. The solution would work like this:

1. Unsolicited e-mail from unknown@foo.com arrives in mail server in domain xyx.abc.com.

2. xyz.abc.com blocks delivery of the message, and sends back a specially formatted message (as described in section 2.2 below) containing an eXtensible Markup Language (XML) form soliciting the hashed token, and including a randomly generated secret key for this sender and the message-ID of the original unsolicited e-mail message.

3. xyz.abc.com retains a copy of key sent to unknown@foo.com in its Originator Key Database (OKD) indexed under unknown@foo.com. This record is retained for a finite period unless validated. The retention period is defined by the Sender Access Policy (SAP).

4. If, and only if, unknown@foo.com proves to be the sender's an accurate address, they will receive the XML form containing the key. If the XML form is not received and processed within the retention period of xyz.abc.com, then the original unsolicited message was properly denied access, and the prospective user unknown@foo.com must begin the process anew.

5. The UA software for unknown@foo.com decodes the XML form and stores the key from domain xyx.abc.com in its Recipient Key Database (RKD) indexed under domain xyz.abc.com.

6. unknown@foo.com looks up the original unsolicited e-mail according to the message-ID included in the received XML form. If the message has been deleted or cannot be located, then the equivalent of a non-receipt notification should be presented to the user.

7. unknown@foo.com employs the newly received key in the RKD to generate a hashed token (as described in section 2.3 below) and resends the original unsolicited message amended to include the token in a new RFC-822 heading extension.

8. On receipt of this resent message, xyz.abc.com will detect the token extension, look up the key for unknown@foo.com in OKD, and either grant or deny delivery depending upon whether the token value is correct.

9. unknown@foo.com may employ the existing key in its RKD in future messages to generate the hashed token extension.

Variations in this procedure are possible to provide additional functionality depending on the requirements of the user. If a prospective recipient requires exclusion of messages generated by automated processes, then step (2) can include part of the key in a distorted image to make parsing difficult. This feature consists of existing technology employed by web servers today. If xyz.abc.com receives some critical number of unsolicited message from unknown@foo.com without the token extension, it could add unknown@foo.com to a local blacklist and cease responding to future requests. This prevents the OKD from growing without bound in a denial of service (DoS) attack. Another variation would be to allow a facility for unknown@foo.com to send a different XML form to xyz.abc.com at a future time to change their key in the OKD.

Alternately, the xyz.abc.com could periodically issue new keys to unknown@foo.com at regular intervals defined by the SAP.

A key factor in the procedure is the handling of incoming messages containing the token extension, but not employing the proper key. In this event, step (8) dictates that the delivery of the message would be denied. However, consideration must also be given to reissuing a new key to unknown@foo.com. The conditions under which a new key should be issued may be subject to the SAP.

2.2 Secret Key Transmission

The response message to an unsolicited e-mail message (as outlined in clause 2.1 step 2 above) will consist of a Message Disposition Notification (MDN) prepared in accordance with [MDN]. The MDN will include a new extension field named *Identity-Key* that will convey the originator address or the unsolicited message, and a new base64 encoded random secret key. The secret key will be stored in the OKD indexed by the originator address. A notional example of such an MDN is shown below.

```
Reporting-UA: somebody@xyz.abc.com
Arrival-Date: Fri, 27 Feb 2004 04:00:59 -0500 (EST)

Original-Recipient: rfc822;somebody@xyz.abc.com
Final-Recipient: rfc822; somebody@xyz.abc.com
Disposition: automatic-action/MDN-sent-automatically; denied
Original-Message-ID: <200402272301.23456@foo.com>
Identity-Key: <unknown@foo.com>; WIwMXUxL2llY2F3ZWIvcHViGaWxlAxNzo1NS
```

Some variation in the MDN fields used is expected to accommodate local implementation needs. Note that the MDN extension field *Identity-Key* shown above would require formal registration by the Internet Assigned Numbers Authority (IANA).

The MDN response shall be generated either automatically only if indicated in the SAP. In accordance with [MDN] clause 2.1 if there are multiple *Return-Path* headers, the *Return-Path* header is absent, or the *Return-Path* header differs from the address in the *Disposition-Notification-To* header.

The size of the key to be issued by the MDN is somewhat arbitrary, since it is not used for any cryptographic operation per se. The key only provides a secret value for use in later proving the identity of an originator. The key size should be established by the user as part of the SAP.

The UA may choose to reissue new keys to existing originators represented in the OKD on a periodic basis. Whether this occurs and how often should be defined by the SAP.

MDNs containing the *Identity-Key* extension should not be routinely presented to users of UAs that support the extension. This MDN is intended to facilitate key transfer and signal that this spam control technique is in use, and offers few if any benefits to the user. For UAs that do not support the extension, formatting the key transfer as an MDN has the benefit that refusal of

message by the spam filter can be properly indicated. Visibility of these MDNs in properly cooperating systems may cause user confusion in conflict with the "grandmother test", because the message in question is to be automatically retransmitted.

2.3 Token

Future messages from unknown@foo.com will be granted access to pass through the receive filter at xyz.abc.com provided that they contain an instance of the *Identity-Token* heading extension that matches their address and key. The *Identity-Token* extension will consist of the recipient address, a timestamp to provide a measure of liveness, and a hash generated over these two values and the originator's secret key. Note that a random number might also need to be included in this value to provide sufficient entropy depending on the size of the key used. The hash will employ the Secure Hash Algorithm (SHA-1) defined in [SHA-1]. The originator will locate the proper key by searching for the recipient address in the RKD. A notional example of an *Identity-Token* extension is shown below.

```
Identity-Token: <somebody@xyz.abc.com>; Fri, 27 Feb 2004 04:00:59
     -0500 (EST); MjAwMS4wOS4yNSAxCIEU6XFxERUwwNS1GaWxlcy5
```

Note that the MIME header shown above will require formal registration by IANA.

Canonicalization of the of the hashed information shall consist of encoding exactly the characters presented in the recipient address portion and the dates fields delimited by exactly one space character (i.e., ASCII 32 decimal, 20 hex). No line terminators (i.e., carriage return or line feed) or other whitespace shall be included in the hash. The bytes to be hashed based on the above example would consist of the following. The "*" characters indicate 128 bytes of the secret key.

	BYTE OFFSET															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
(+0)	<	s	o	m	e	b	o	d	y	@	x	y	z	.	a	b
(+16)	c	.	c	o	m	>	;		F	r	i	,		2	7	
(+32)	F	e	b		2	0	0	4		0	4	:	0	0	:	5
(+48)	9		-	0	5	0	0		(	E	S	T	)	;		*
(+64)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+80)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+96)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+112)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+128)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+144)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+160)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(+176)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	
(+192)																

Figure 3 – Canonicalization of the Hash Contents

The *Identity-Token* extension will be multi-valued. In the event that a message is being sent to multiple recipients that require this spam-control mechanism, an instance of the *Identity-Token* extension should be included for each such recipient.

When received, at least one instance of an Identity-Token extension should validate correctly for the purported message originator as a condition for the message to be displayed to the user. The recipient UA should search the tokens present in the message for its own address. If a token with the proper address it not found, the message should be treated as if no token were present. If more than one token contains the proper address, the recipient UA shall process exactly one such token. Selection of which token to process shall be a local matter. To validate the token, the recipient UA shall regenerate the has as per the canonicalization described above using the address and time stamp as provided in the extension and the key associated with the originator that has been retrieved from the OKD. Once generated, the hash should be compared to the hash presented in the Identity-Token extension. If the two hashes match exactly, the token shall be considered validated and the message shall be displayed to the user. If the hashes fail to match, the token fails validation. In event the token fails to validate, the message shall not be presented to the user. It shall be identified in the SAP whether or not a new key shall be sent to an originator in the case of a failed token validation.

The *Identity-Token* extension should not be routinely presented to the recipient user. The token is solely to facilitate automated access control, and offers few if any benefits to the user. Visibility of these tokens may cause user confusion in conflict with the "grandmother test".

2.4 Databases

The OKD is the foundation in the UA for recognizing previously validated recipients. The OKD consists of a simple database indexing keys previously issued in MDN according to the originator addresses to which they were issued. When a message is delivered to a UA, the purported originator address is looked up OKD. If the address is not found, the message is blocked and an MDN is sent as per clause 2.2. This will result in a new key being added to the OKD with the response field marked with the date by which a response from that originator is required (according to the SAP). If the originator address is contained in the OKD, then the associated key is used to validate any received token. Any time a particular key entry is used, the response field is cleared. The OKD may be purged periodically to remove any records for which the response date has passed. An illustration of the OKD structure is shown below.

Originator Address	**Originator Secret Key**	**Response**
dingbat@foo.com	b21wYXEtUFBDMjAwMi1VcGdyYWRlLUNvdXBvbi5wZG	<none>
unknown@foo.com	cy56aXANCjIwMDEuMDkuMjUgMTc6NTUgQiBFOlxcRE	29-Feb
.	.	.
.	.	.
.	.	.
bill@another.net	Y2EuY29tL3RlbXA0NTYgMDExMjEzLUNvbXBhcS1QUE	<none>
ted@nameless.edu	cGFxLVBQQzIwMDItVXBncmFkZS1SZWNlaXB0LnBkZi	3-Mar

The RKD is the foundation in the UA for identifying the proper key(s) to use for token generation on a given message. The RKD is similar to the OKD, but is indexed by the prospective recipient's address from which a key was received in a prior MDN. When a user is sending a message, they will look up each prospective recipient in the RKD. For each recipient found in the RKD, a separate *Identity-Token* extension will be generated and added to the message. If a recipient is not in the RKD, it may indicate that they have not yet provided a key, or that they do not support this mechanism for spam control. An illustration of the RKD structure is shown below.

Recipient Address	Recipient Secret Key
mary@scots.edu	b21wYXEtUFBDMjAwMi1VcGdyYWRlLUNvdXBvbi5wZG
xyz.abc.com	cy56aXANCjIwMDEuMDkuMjUgMTc6NTUgQiBFOlxcRE
.	.
.	.
.	.
t1431@mamma.net	Y2EuY29tL3RlbXA0NTYgMDExMjEzLUNvbXBhcS1QUE
dave@umich.edu	cGFxLVBQQzIwMDItVXBncmFkZS1SZWNlaXB0LnBkZi

Both the OKD and RKD might reasonably be implemented as part of a local address book or directory service. While the content of the databases is sensitive, the degree of protection that must be afforded to the database is relatively limited. It is only necessary to prevent disclosure or the key values to prospective spammers. In many circumstances, localizing the data to the user's home domain or account is sufficient protection. Since the key values in the RKD are assigned on a per-user basis, the user-association of the information must be preserved. The same is true for the OKD, except that the OKD may be used to support spam filtering at the domain level.

2.5 Sender Access Policy

The SAP defines a number of operational characteristics that affect whether the sender's message will be granted permission to be delivered. The SAP is entirely under the control of the receiving UA, or in the case of the filtering for an entire domain the receiving MTA. This puts the receiving in control of what sort of messages are acceptable. Characteristics that would be defined by the SAP include the following.

- **Response Delay** – The period of time that a new originator key will be retained in the OKD before a response is required
- **Originator Rekey** – An indication whether an originator may submit an XML form to change their own key in the OKD
- **Key Size** – Defines the size in bytes or the key to be issued for new originators.

- **Rekey Period** – Defines the period of time after which new keys will be issued to prospective originators
- **Automation Exclusion** – Defines whether or not to exclude messages generated by automated processes
- **Blacklist Exclusion Count** – Indicates how many unsolicited messages without the token extension will be tolerated from a given originator, after which point that originator will be added to a local blacklist and the UA will cease to respond to future requests from that address
- **Blacklist Purge Period** – Indicates how long entries should remain in the local blacklist
- **Whitelist Users** – Allows the user to manually configure the system to admit messages appearing to be from certain users without employing the challenge and response mechanism. This will allow for interoperability with users whose UAs do not support the mechanism.
- **Reissue on Bad Key** – Indicates whether a new key should be sent in response to an incoming messages containing the token extension, but not employing the proper key
- **Automatic Response** – Indicates whether or not the MDN containing the originator's key shall be generated automatically, or whether user confirmation shall be sought

For each of these operational characteristics, the recipient user shall be given control. However, in the interest of passing the "grandmother test" it is necessary to establish reasonable default settings for each of these. Customization of these parameters might be hidden behind an "advanced options" button in the SAP controls. The default values should provide reasonable performance in spam rejection without causing operational problems. The following default settings are proposed.

SAP Parameter	Default Value
Response Delay	7 days
Originator Rekey	No
Key Size	128 byte (1024 bit)
Rekey Period	12 months
Automation Exclusion	No
Blacklist Exclusion Count	Yes
Blacklist Purge Period	1 month
Whitelist Users	(empty)

SAP Parameter	Default Value
Reissue on Bad Key	Yes
Automatic Response	Yes

3. ANALYSIS OF APPROACH

In order for it to be considered worthwhile to conduct experiments with the candidate protocol extensions, a certain amount of analysis is required to provide confidence that they will perform as expected and stand up to attack in the proposed operational environment. This section identifies the operational characteristics that are both advantageous and disadvantageous, and possible weaknesses that could be exploited by spammers or their hacker allies.

3.1 Operational Advantages

This proposed solution should reject spam from non-existent addresses because the MDNs containing the key will not reach the spammer. It should reject mail from valid but usurped addressed because the usurped user won't respond to the XML MDN. The solution has the capacity to reject mail from automated systems if coupled with other existing technologies for ensuring human users. It also has the potential to dramatically reduce the level of false positive spam detections because known communication partners will employ the correct key in preparing their messages.

The proposed mechanism incorporates the concept of a flexible SAP under recipient user (or organization) control. This is important as it preserves the principle of complexity to the edge. The default policy recommended should address the needs of a broad user community.

The recipient portion of the anti-spam system can be implemented entirely on the server side. This allows the implementation to provide anti-spam protection to an entire organization or site. It also may facilitate roll-out of the mechanism in heterogeneous domains employing a variety of different e-mail UAs. The originator portion of the system could also be implemented entirely on the server side to facilitate roll-out, but this configuration is not recommended (see clause 3.2).

The solution can operate relatively autonomously according to the default SAP to provide anti-spam protection even to relatively unsophisticated users. This is important not only because it helps to satisfy the “grandmother test” condition, but because it will allow it to block spam for a wide range of users who cannot (or will not) use less turnkey technology. Widespread blocking of spam is the key to reducing the level of spam by undercutting the spammers’ economic model.

The cryptography employed in the proposed solution is relatively simple, so that implementation is not likely to be a barrier to the average implementer. Similarly, the RKD could be easily integrated into most existing address book implementations, something already quite common in e-mail UAs.

The proposed solution requires zero infrastructure. This maintains the principle of a simple core, and thereby allows incremental deployment, good scalability, and ultimately improved interoperability.

The proposed mechanism can help to achieve a much lower rate of false rejections in spam filtering. This can have very positive impacts on user acceptance; especially in business environments where reliable e-mail might be considered crucial. It also contributes to satisfying the "grandmother test".

3.2 Operational Disadvantages

E-mail UAs that employ filtering based on this proposed mechanism will not interoperate well with e-mail UAs that do not support the proposed extensions. The ability to configure a whitelist in the SAP will mitigate this to some extent, but maintaining a large whitelist has disadvantages. First, each address in the whitelist represents an address that might be exploited by a spammer. Second, management of a large whitelist may be overly onerous for the user.

The sizes of both the OKD and RKD scale linearly in proportion to the number of parties with which the user communicates. This may create a scalability issue for users who communicate with large numbers of other users. However, since most users have relatively small sets of partners with whom they exchange e-mail this may not be a serious problem. Also, perhaps "power users" have power platforms from which to run.

Repeated attempts to penetrate the filter mechanism can result in rapid expansion of the OKD. Users who receive large volumes of spam might experience OKD scalability issues. This can be managed to some extent by shortening the response delay in the SAP. However, this comes at the expense of requiring a faster response from legitimate users.

Spammers attempts to impersonate a known communication partner might result in that partner being automatically blacklisted. If this occurs then future communications from that partner would be blocked constituting false positive spam detections.

Implementation of the originator portion of the anti-spam system can introduce weaknesses to the system. If the RKD and token generation are performed by a proxy agent, such as a local mail server etc., then all a spammer in that local domain must do is impersonate a different local user in order to employ their set of key. Since the feasibility of identity spoofing with SMTP has been amply demonstrated, this seems a likely attack to anticipate.

3.3 Possible Weaknesses and Vulnerabilities

The spammer has the option of trying to attack this mechanism my sending a seemingly legitimate message with an originator or reply-to address that corresponds to a mailbox that is accessible to them. In this event, the spammer would automatically receive a key that would allow them to get messages through to the target. However, the key would only function for messages seeming to come from that address, so subsequent attempts to use that address to spam

the target could be dealt with by adding the address to the blacklist. Also, it since mailbox access is required to obtain the key in the first place, it is perhaps possible to identify the spammer via their service provider.

The spammer might intercept or otherwise observe the MDN returned to a legitimate user, thereby learning their key and enabling subsequent spamming of the target. In this event, the spammer could impersonate that user and successfully spam the target user. However, since the key is pair-wise between those two users, the spammer would need to repeat this process for every target. Assuming that the spammer could gain access to working address/key combinations for every target user, the odds of the address being identical for any of the targets are poor. So the spammer would need to vary the spoofed originator on a per-target basis, and maintain a very large RKD. Of course, none of this would prevent the target users from blacklisting the address in question making the whole exercise for naught.

The spammer might impersonate a legitimate user and generate tokens for spamming message to conduct a brute force attack on the key. This is impractical because the repeated attempts would stand out, allowing the target filter to add the purported address to the blacklist. Furthermore, since the spammer could not be assured of a response when the correct key was used, the odds of the correct key going undetected are high.

Excessive use of the whitelist feature in the SAP can introduce weaknesses in the spam protection capabilities of the system. Each address in the whitelist is vulnerable to impersonation by spammers. Of course, since the spammer has no way of knowing what addresses the target has in their whitelist, exploiting this weakness is somewhat problematic.

Spammers might bombard the target user with large numbers of messages that do not contain the proposed token in an attempted DoS attack. While this may result in blacklist, the main protection from this attack is the lack of profit motive on the part of the spammers. In other words, this attack falls outside the scope of what we term spam.

4. CONCLUSION

This mechanism would give recipient users or domains a powerful tool to reject mail from non-existent addresses, valid but usurped addressed, and messages from automated systems. The approach supports commonly desired policy constraints. The recipient half of the system can be implemented entirely on the server side. The cryptography used does not have to be extreme. This seems to me simple, but offering a lot of advantages.

A program of simulation is recommended, followed by a limited implementation as a plug-in for one or more e-mail UA. If testing shows this mechanism to be effective in blocking unwanted e-mail communication and achieving a low rate of false rejections, then a derivative of this technique should be considered for Standards Track. Use of a similar technique for other applications other than e-mail (e.g., instant messaging, chat) should also be explored.

5. REFERENCES

5.1 Normative References

[MDN] RFC 2298: *An Extensible Message Format for Message Disposition Notifications*, R. Fajman, March 1998.

[SHA-1] FIPS PUB 180-1: Secure Hash Standard, National Institute of Standards and Technology, 17 April 1995.

5.2 Informative References

[REPORT] RFC 1892: *The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages*, G. Vaudreuil, January 1996.

[MIME3] RFC 2047: *MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*, K. Moore, November 1996.

[MSGFMT] RFC 2822: *Internet Message Format*, P. Resnick, April 2001.

6. AUTHOR'S ADDRESS

Christopher Bonatti
IECA, Inc.
15309 Turkey Foot Road
Darnestown, MD 20878-3640
BonattiC@ieca.com