

Network Working Group
Internet-Draft
Expires: November 14, 2005

J. Fenton
M. Thomas
Cisco Systems, Inc.
May 13, 2005

Identified Internet Mail
draft-fenton-identified-mail-02

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on November 14, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document describes extensions to the format of electronic mail messages and a public-key infrastructure to permit verification of the source of messages by either mail transport agents (MTAs) or mail user agents (MUAs). This may be used to implement a policy which, for example, favors the delivery of identified messages over messages lacking signatures or having incorrect signatures. Mechanisms by which signing of messages and verification of signatures can be performed by trusted MTAs, in order to minimize impact on end users,

Internet-Draft

Identified Internet Mail

May 2005

are also presented.

Table of Contents

1.	Introduction	3
2.	Requirements Language	5
3.	Identified Internet Mail Concepts	6
3.1	Application of Signatures	8
3.2	Verification of Signatures	8
4.	Message Format	10
4.1	Header and Verification Syntax	10
4.2	Relationship to MIME	11
5.	The Signature Header	12
5.1	IIM Signature Calculation	12
5.1.1	Canonicalization	13
5.2	IIM-Sig Tag Values	13
5.3	The Verification Header	16
5.4	Use of From header	17
6.	Key Management	18
6.1	Key Registration	18
6.1.1	Key Authorization via KRS	19
6.1.2	Key Authorization via DNS	22
6.1.3	Null Key Checks	23
6.1.4	Key Authorization Results	23
6.2	Policy Options	25
7.	Usage examples	27
7.1	Simple message transfer	27
7.2	Outsourced business functions	27
7.3	PDAs and Similar Devices	27
7.4	Mailing Lists	27
7.5	Affinity Addresses	28
7.6	Third-party Message Transmission	29
8.	DNS considerations	30
9.	Security Considerations	31
9.1	Potential Attacks	31
9.1.1	Key Registration Server Denial-of-Service Attack	31
9.1.2	Key Registration Server Stall Tactic	31
9.1.3	Misappropriated private key	32
9.1.4	Message Replay Attack	32
9.2	Other Considerations	33
10.	IANA Considerations	34
11.	Acknowledgements	35
12.	References	36

12.1	Normative References	36
12.2	Informative References	36
	Authors' Addresses	36
	Intellectual Property and Copyright Statements	38

[1.](#) Introduction

Internet users have recently been subjected to a torrent of unsolicited email messages. These generally take two forms:

1. Messages originated by "spammers" to send advertising or solicitation, or as part of a confidence scheme or other fraud
2. Messages sent automatically by worms and other malware attempting to infect additional systems

In both cases, a large proportion of such messages attempt to disguise their true source, in order to frustrate attempts to shut down the spammer, to disguise the identity of the infected system sending the message, or to make it appear that the message came from an entity the recipient trusts (e.g., a bank). Since SMTP permits senders to use any return address they wish, the addition of a signature to messages limits the opportunity of spammers or malware to forge return addresses, and thus provides some degree of accountability for the source of email messages.

As currently used, message signatures such as those generated by (for example) PGP are generally used to indicate authorship of the message by a particular individual. The intent here is somewhat different: we want to determine if the sender of the message has authorization (from the administration of the domain) for the use of an email address and associate the signature with the message itself. In order to make the signature transparent to recipients who do not intend to verify it, the signature is encapsulated in the message differently than in a signed PGP message. The public key used to sign the message is included in the header, and its binding with the From or Sender header in the message can be verified with the sending domain.

The email identification problem is characterized by extreme scalability requirements. There are currently on the order of 30

million domains and a much larger number of individual addresses. It is important to preserve the positive aspects of the current email infrastructure, such as the ability for anyone to communicate with anyone else without introduction. Due to the desire to retain this "any to any" characteristic of email and the (perhaps) lower standard of trust required to accept an email message as compared with, for example, processing a financial transaction, we present an alternative key management model here.

What is presented here is not by itself a solution to the spam problem. The intent is to give tools to the recipient to allow the classification and prioritization of desired mail. Since much

Fenton & Thomas

Expires November 14, 2005

[Page 3]

Internet-Draft

Identified Internet Mail

May 2005

undesirable mail is currently characterized by forged return addresses, the identification of such messages is a major focus of this effort. Some degree of accountability for the source of email messages will also result, although spammers will still have the ability to operate their own domains and key infrastructure, create large numbers of bogus identities, and sign messages. For this reason, identification of the sender is really a foundation on which accreditation and reputation services can be based. It is hoped that through a combination of such mechanisms, spam, fraudulent, and malware-generated messages may eventually be marginalized to the point that they are not the serious problem they are today.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [3].

[3.](#) Identified Internet Mail Concepts

Identified Internet Mail (IIM) provides a means by which cryptographic signatures can be applied to email messages to demonstrate that the sender of the message was authorized to use a given email address. Message recipients can verify the signature and consult the sender's domain to determine whether the key that was used to sign the message was authorized by that domain for that address. This confirms that the message was sent by an party authorized to use the sender's email address.

Just as an email administrator, by creating an email account, gives a user the ability to receive mail addressed to a given address, a means for authorizing the transmission of email messages is needed. The administrator can delegate the use of an address in several ways.

The administrator could operate a mail transfer agent (MTA) or mail submission agent (MSA) for the domain that authenticates the sender when accepting a message. This MTA or MSA would typically have a private key that is authorized to send mail for anyone in the domain. Alternatively, the administrator could register a public key for the sender with which, assuming the sender has an MTA or MUA with the appropriate software and the corresponding private key, the sender could sign his own outgoing messages. In the latter case, the private key would typically be authorized for one or more specific addresses that the sender is authorized to use.

One central principle used here is that an email address doesn't represent a user's identity. Rather, an email address is something that the owner or administrator of a domain delegates to a user. For example, John Doe may have the email address of jdoe@example.com, but only as long as he remains employed by example.com's owner (or as long as he uses example.com as an ISP). When this relationship changes, John's identity doesn't change, but his authorization to use jdoe@example.com does. For this reason the administrator of example.com, and not John, must have control over the authorization to use the jdoe@example.com address.

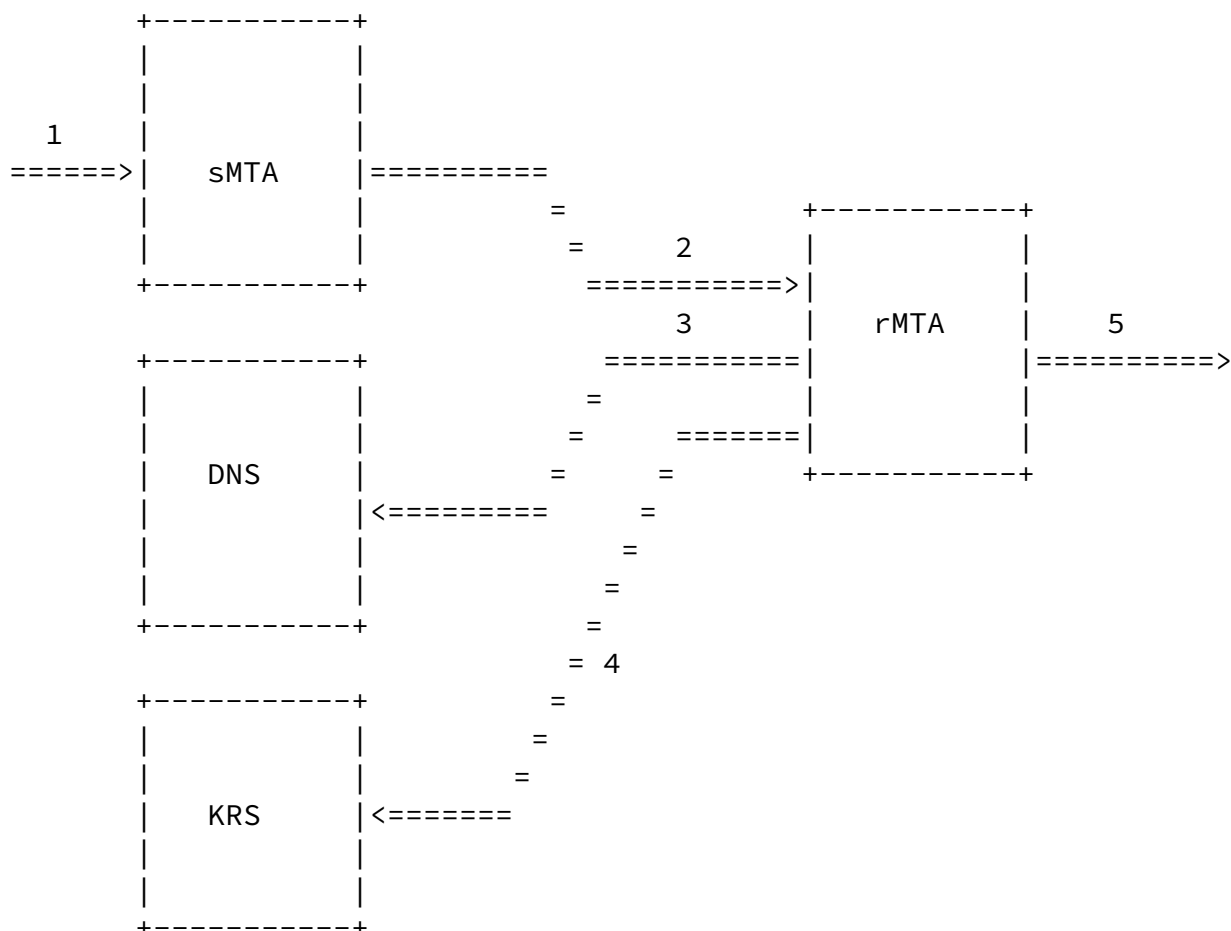
IIM permits keys to be authorized for specific email addresses ("user-level keys") and/or for all addresses in a domain ("domain-level keys"). User-level keying is needed to support some important use cases. For example:

- o Domains which want to authorize a partner, such as an advertising provider or other outsourced function, to use a specific address for a given duration.
- o Domains which want to allow frequent travelers to send messages locally without the need to connect with a particular MSA.

- o "Affinity" domains (e.g., college alumni associations) which provide forwarding of incoming mail but which do not operate a mail submission agent for outgoing mail.

It is expected that many domains will only authorize keys, probably held by outgoing MTAs, which are used for all addresses in the domain. Other domains might allocate a small number of user-level keys, but perform most signing at the domain level. A few domains,

A typical message flow for an IIM message is as follows:



1. The sending MTA receives mail from a source that it determines is allowed to send mail using its domain's name.

2. The sending MTA signs the mail and forwards it to the receiving

domain.

3. The receiving domain's MTA determines that there is a signature from the responsible domain which asserts that the public key can be verified via a KRS lookup. The MTA performs a DNS lookup to get the address of the sending domain's KRS. This is a key security property: that the sending domain has control of the contents of its DNS.
4. The receiving MTA formulates a query back to the sending domain's KRS with the fingerprint of the key that signed the message and the 2822 address of which it would like to determine the authorization.
5. The receiving MTA then decides the disposition of the mail based upon the result of the authorization query, typically replacing the IIM-VERIFY header with its opinion and forwarding it along to the receiving MUA.

[3.1](#) Application of Signatures

Identified Internet Mail (IIM) messages are characterized by the presence of a header, IIM-Sig, in the message. This header contains all of the information required to verify the internal consistency of the message itself, including the signature, the public key corresponding to the private key used to sign the message, options such as the choice of canonicalization and cryptographic algorithms, and information on how to verify the authorization of the supplied public key.

Messages may be signed either directly by the author's MUA, by an MSA, or by a subsequent MTA. If the signature is applied by other than the author's MUA, other mechanisms (such as SMTP AUTH, or access control over the network from which messages will be signed) SHOULD be used to ensure that only authorized messages are signed. This is needed to provide some assurance that an attacker will not be able to obtain message signing simply by choosing the right MTA through which to send.

[3.2](#) Verification of Signatures

An MTA at any point in the message transmission path or a recipient MUA may verify the authorization of the message. This is done by an internal check of the consistency of the message to ensure that the message is properly signed using the included signature and public key. A verifying MTA MUST also check the authorization of the public

key to be used with the given email address by consulting either a Key Registration Server (KRS) as described in [Section 6.1.1](#) or DNS as described in [Section 6.1.2](#) at the originating domain, as specified by the signature. This authorization check MAY be performed in parallel with the message consistency check. The message MUST be considered authorized only if the key is authorized and the message passes its consistency check.

If no signature is present, or if a message signature fails its consistency check, a "null key" check SHOULD be performed with the originating domain to determine its preference for how the message should be handled. If the null key check indicates that all legitimate messages are signed by the domain, the verifying MTA SHOULD discard unsigned messages. It SHOULD NOT generate a "bounce" message when doing so.

MUAs MAY frequently wish to avail themselves of the results of verification by MTAs within their trust domain, because such verifications are more likely to be performed in a timely manner, because the MUA doesn't implement message signature verification, or because the MUA is operating in an offline mode. Verifying MTAs MAY attach an IIM-VERIFY header to the message with the results of message verification to be acted upon by MUAs. MUAs using the information in this header SHOULD act only upon IIM-VERIFY headers applied by a known and trusted entity. In order to avoid the possibility of an MUA acting on a spoofed header sent with the message, verifying MTAs MUST remove any IIM-VERIFY headers already present in messages they process.

Internet-Draft

Identified Internet Mail

May 2005

[4.](#) Message Format

An identified internet mail message is in the format of a conventional mail message as defined in [RFC 2822](#) [5]. Two new headers, referred to as the signature and the verification header, are defined.

[4.1](#) Header and Verification Syntax

Identified Internet Mail uses a common encoding for the signature header (IIM-Sig), the verification header (IIM-Verify), as well as for key verification responses from a KRS or from DNS. This consists of a set of tag-value pairs consisting of a tag delimited by a ":" followed by a single value within double quotes separated from the following tag by a ";". Tags may be any number of alphanumeric characters, although at present only single-character tags are defined. A receiver decoding an unknown tag MUST silently discard the tag and value (after incorporating the tag-value pair into the hash calculation for the signature, if applicable).

Values are a series of double quoted strings containing either base64 text, plain text, or quoted printable text, as defined in [RFC 2045](#) [1], section 6.7. The context of the tag will determine the encoding of each value. As of this writing, the only tag which uses the quoted printable format is the "c" (copy) tag. The definition of plain text in this context is ASCII codes 32-126 decimal, with the exception of ASCII 34 (double quote).

Each value SHOULD be split into multiple lines as a series of quoted strings to keep the line length less than 78 octets. Decoders MUST interpret multiple quoted strings in a value as if they were all part of a single quoted line. Encoders MUST NOT violate the maximum line length of any particular header line. Decoders MUST ignore strings which span line breaks as the meaning is ambiguous given the way that mail header continuation lines are formed and often reformatted by intermediaries.

With the exception of the double quote character (which is converted to =22) and the equals sign (which is converted to =3D), received

headers compliant with [RFC 2822](#) [5] will contain no characters which require quoted printable conversion because they will have been encoded as described in [RFC 2047](#) [2]. The conversion of the quote and equals sign is required in order to unambiguously encapsulate the header in a quoted string. For example, the subject line:

Subject: Einstein's "E=mc**2"

would be encoded as:

Fenton & Thomas

Expires November 14, 2005

[Page 10]

Internet-Draft

Identified Internet Mail

May 2005

Subject: Einstein's =22E=3Dmc**2=22

Headers with characters which require conversion (perhaps from legacy MTAs which are not [RFC 2822](#) compliant) SHOULD be converted as described in [RFC 2047](#) [2]. Otherwise, the copied header may be converted to quoted printable form as described in section 6.7 of [RFC 2045](#) [1], with the additional conversion of double quote and equals sign characters to =22 and =3D respectively.

New tags MUST choose whether they are of type plain text, quoted printable, or base64. In general, the plain text type may be used if it is not permissible to have 8 bit and/or syntactically problematic values. Binary forms MUST be encoded as base64, and free form text (e.g., user supplied) MUST be typed as quoted printable.

The syntax of tags is as follows:

```
tagvaluepair = 1*ALPHA ":" DQUOTE 1*valuechar DQUOTE
               *(CRLF 1*WSP DQUOTE 1*valuechar DQUOTE)
valuechar     = %x20-21 / %x23-7E
               ;printing characters except double quote
```

[4.2](#) Relationship to MIME

With the exception of the canonicalization procedure described in [Section 5.1.1](#), the Identified Internet Mail signing process treats the body of messages as simply a string of characters. IIM messages may be either in plain-text or in MIME format; no special treatment is afforded to MIME content. Message attachments in MIME format are included in the content which is signed, provided the chosen canonicalization (in particular the body length count, if specified)

includes the portion of the message containing the attachment.

Fenton & Thomas

Expires November 14, 2005

[Page 11]

Internet-Draft

Identified Internet Mail

May 2005

[5.](#) The Signature Header

The Signature header MUST be included in a message in order for it to be considered an Identified Internet Mail message. It has the syntax:

```
signature = "IIM-Sig:" SP *([CRLF 1*WSP] tagvaluepair ";")
            tagvaluepair CRLF
```

The signature-text contains a number of fields which represent the signature itself, a public key used to create the signature, and related information. An example of a signature is as follows:

```
IIM-SIG: v:"1"; h:"iim.example.com"; d:"example.com"; z:"home";
m:"krs"; t:"1094844765.338603"; x:"432000"; a:"rsa-sha1";
b:"nofws:1192"; e:"Iw==";
n:"zCnd+ByA23/7WmiIwaIZ7Ez3DplzVMdRKP138IXLOvBVeaRZ4yWEPclZ/2Mda"
"s5Bs9RPWH0BGd3fx6j+txd0Xarv4Y8kpMqTexCOMFlDmatpXDXfFj3VI9o4G7"
"674gFTasaoPcvEfZCwcBgZD7T6sLZa3RTBUGzZq0shAMRpVek=";
s:"Tg67/+k8oltzxIBxN4mevOgbP/+xqxuT0ugZJ1VoaEm3bJ7JHA0y+X5FEMRF"
"/SLZ+GBYIA7wtEmjgbHNuVRnbJQWu732PRbI6UKNGocCEX0TvVdZxFTQzbh3x"
"zaEj6BIWx6GYIo8oWoeM3kzZTiip2pPhuvaXu9Ho+3eR81MZ4=";
c:"From: John Doe <jdoe@example.com>";
c:"Date: Fri, 10 Sep 2004 12:25:51 -0700 (PDT)";
c:"Subject: RE: Lunch"
```

[5.1](#) IIM Signature Calculation

All tags and their values in the IIM-Sig header are included in the cryptographic hash with the sole exception of the s: (signature) tag and its value. Tags that are not understood by the receiver are included. The tags and their values are simply concatenated to each other when forming the cryptographic hash in the order they are present in the IIM-Sig line. That is: "v1hiim.example.com[...]" Syntactic markers are NOT included and the value used in the hash is before encoding and after decoding. The final hash algorithm is as follows:

TRUNC (SHA1 (SIGTAGVALS, SHA1 (BODY)), 12)

where SIGTAGVALS is the encoding described above for the header tags/values and BODY is the SHA-1 hash of the body of the email itself, including the leading CRLF. Note that SHA-1 value of the body uses the full 16 bytes of the hash (i.e., not truncated).

When calculating the hash on base64 and quoted printable values, the hash is taken after the encoding. Likewise, the receiver MUST

incorporate the values into the hash before actually decoding the value.

[5.1.1](#) Canonicalization

In order to minimize the likelihood of signature mismatch due to innocuous message modification, a canonicalization algorithm MAY be specified by the signer of the message. Two canonicalization algorithms are currently defined, "plain" and "nofws". In addition, a body length count MAY be specified to limit the signature calculation to a subset of the body text.

Plain canonicalization is effectively the null canonicalization. Bytes of the body are included in the hash without modification. The "nofws" canonicalization removes all whitespace characters in the body and then strips the eighth bit of the data prior to calculating the hash. In this context, whitespace characters are defined as:

```
nofws-whitespace = 1*( SP / HTAB / CR / LF / %x0B / %x0C )  
                    ; last 2 are vertical tab and form feed
```

The body length count allows the signer of a message to permit data to be appended to the end of the body of a signed message. This capability is provided because it is very common for mailing lists to add trailers to messages (e.g., instructions how to get off the list). Until those messages are also signed, the body length count is a useful tool for the receiver since it MAY as a matter of policy accept messages having valid signatures with extraneous data, possibly highlighting the unsigned area. Alternatively, it may strip the extraneous data or reject the message outright. A signer MAY not want to permit appendages to messages and can set the length to "-1" to indicate that all bytes of the body are to be used to calculate the signature. The body length count is made following the canonicalization algorithm; whitespace characters are not counted when using the "nofws" algorithm.

Signers that wish to ensure that no modification of any sort may occur MUST specify the "plain" algorithm and a length of -1.

Despite the measures described above, some messages, particularly those containing 8-bit data, may be subject to modification in transit invalidating the signature. Messages containing 8-bit data SHOULD be converted to MIME format prior to signing, using a suitable content transfer-encoding such as quoted-printable or base-64.

[5.2](#) IIM-Sig Tag Values

The tags used in the signature are as follows. The type of each tag

is shown in parentheses after its name.

a: Algorithm (plain-text). One-way hash and public key algorithm. Currently this MUST be rsa-sha1. This tag MUST be present.

b: Body canonicalization (plain-text). This tag informs the receiver of the type of canonicalization used to calculate the signature and the number of bytes in the body of the email included in the cryptographic hash, starting from 0 at the CRLF preceding the body. Its value is of the form "Canon-algorithm:Length". Recognized values of Canon-algorithm are "plain" and "nofws". A

length of -1 specifies that all bytes of the body are to be included in the signature calculation. This tag is OPTIONAL and defaults to "plain:-1" if it is absent.

- c: Copied header (quoted-printable). A copied header is a header which the sender would like to cryptographically sign. No other headers are included into the cryptographic hash. The From header MUST be included; the Sender header MUST also be included if the signature is on behalf of a Sender address which is different from the From address. The copied headers SHOULD also include the Subject and Date headers and MAY include any other headers present at the time of signing at the discretion of the signer.

In calculating the hash, the value MUST be encoded as the copied header followed by a colon (":"), followed by a single space, followed by the rest of the value of the copied header.

- d: Domain of signer (plain-text). This tag denotes the signing domain. It is used to inform the receiver of the appropriate level of address that is considered the authoritative domain in this context. For example, if a message is received from jdoe@eng.example.com, the d: tag might indicate that the domain is example.com or eng.example.com. If this tag does not correspond to the hostname of the From or Sender address or to a parent domain, the signature MUST be ignored. This tag MUST be present.
- e: Public exponent (base-64). The RSA public exponent of the supplied signing key. This tag MUST be present.
- h: Signing host (plain-text). The hostname of the applier of the signature. This is purely informational and is not in verifying the signature. This tag is OPTIONAL.
- m: Method (plain-text). This tag determines which method the signer desires the receiver to use to check the authorization of the supplied public key. This MUST be either "krs" or "dns". This tag is OPTIONAL and defaults to "krs" if it is absent.

- n: Modulus (base-64). The RSA public modulus of the supplied signing key. Note that the key length is implicit with the number of decoded bits in the modulus. Signers MUST support key lengths of 1024 bits and SHOULD support 768 and 1536 bits. Receivers MUST

support key lengths of 768, 1024 and 1536 bits. This tag MUST be present.

- s: Signature (base-64). The RSA signature over the computed one-way hash. The format of the signed message follows PKCS #1v2 with PKCS 1.5 padding. That is, 02||PS||00||M, where PS is the padding, M is the signed hash. If the length of M is greater than 12, the first 12 octets MUST be used and the subsequent octets silently ignored. Refer to PKCS #1v2.1 [RFC 3447](#) [6] for the format of PS. This tag MUST be present.
- t: Timestamp (plain-text). The time that this signature was created. The format is the standard Unix seconds-since-1970 followed by a fractional number which MUST be guaranteed to be unique for this signing key. The intent of the fraction is to guarantee the uniqueness of any given signature at any particular instance. The value is expressed as an unsigned integer in decimal ASCII. This tag MUST be present.
- v: Version (plain-text). This MUST currently be set to "1". This tag MUST be present, and MUST be the first tag in the IIM-SIG header.
- x: Signature TTL (plain-text). Signature expiration in seconds-since-1970 format. Signatures MUST NOT be considered valid if the current time at the receiver is past the expiration date. The value is expressed as an unsigned integer in decimal ASCII. This tag MUST be present.
- z: Signature semantics (plain-text). This tag has two possible values: "home" and "routing". When a signing entity (MTA, MSA, or MUA) wants to assert the origin of a message, it tags the signature with the "home" value. An MTA applying a "home" signature SHOULD perform some sort of access control to filter out mail from outsiders trying to forge mail or SHOULD authenticate the submitter of the message. If the entity merely wants to attest that the mail passed through it on its way to the destination, it uses the "routing" value. Routing signatures are similar to signed Received headers and are of primarily forensic value. This tag SHOULD be present. A missing signature semantics tag should be interpreted as "home".

[5.3](#) The Verification Header

The verification header is an optional header which MAY be used to convey the verification of a message from an MTA to an MUA or a subsequent MTA within the same trust domain. It has the syntax:

```
signature = "IIM-Verify:" SP *([CRLF 1*WSP] tagvaluepair ";")
           tagvaluepair CRLF
```

The verification header indicates whether an MTA was able to successfully verify the message according to whatever policies it decides to use. A recipient MUA or MTA MAY decide to rely on the presence of a verification header in applying policy to the message (e.g., moving an unverified message to a lower-priority folder), or it may do such verification locally.

The verification header is not cryptographically protected, in order to avoid the need to manage keys for verifying MTAs. The verification header SHOULD be deleted from the header when the message is sent via SMTP outside the trust domain of the sender, and it MUST be discarded if it received from an SMTP peer that is not trusted by the recipient (normally one that is within the recipient's administrative control). There MUST be at most one verification header in a message; MTAs which perform message verification MUST ensure that they either agree with the contents of any existing verification header, or replace it with a new one.

An example of a verification header is as follows:

```
IIM-Verify: s:"y"; v:"y"; r:"68"; h:"iim.example.com"
```

The tags and values used by the verification header are as follows:

- s: Signature (plain-text). The value is "y" if there is a signature line from the sending domain (i.e., the domain suffix of the Sender or From header). Otherwise the value is "n". This tag MUST be present.
- v: Verify (plain-text). The value is "y" if the home domain's signature is both present and the public key operation verifies correctly. This tag MUST be present.
- r: Rating (plain-text). The value here is between -127 and 127 with negative values expressing an adverse rating, zero being neutral and positive values indicating a favorable rating. The rating value is completely at the discretion of the entity supplying the IIM-Verify header and MAY take into account many different factors

including the rating supplied by the home domain's KRS, local and

Internet-Draft

Identified Internet Mail

May 2005

third party ratings, and any other factors the verifying entity considers relevant. This tag SHOULD be present.

- h: Host (plain-text). This is the fully qualified domain name of the verifying host. It should be noted that since the IIM-Verify header is not cryptographically protected, users or subsequent MTAs which make use of the IIM-Verify header must independently ensure that it is not subject to tampering.
- c: Comment (plain-text). This is a free form string intended to convey a human readable comment about the operation. This is typically used to send diagnostic information for failed operations. This tag MAY be present.

[5.4](#) Use of From header

Identified Internet Mail associates the key in the message with either the Sender or From address of the message. This is done to allow mailing lists which re-originate messages and apply a Sender header (but retain the original From address) to sign the re-originated messages. However, it is the From address that is most commonly seen by the recipient, and it is important that if the Sender address is used to verify the message, that it be made visible to the user.

In order to retain the current semantics and visibility of the From header, verifying mail agents SHOULD take steps to ensure that the Sender address is prominently visible to the user if it is used to verify the message and is different from the From address. If MUA implementations that highlight the signed address are not available, this MAY be done by rewriting the From address in a manner which remains compliant with [RFC 2822](#) [5]. If performed, the rewriting SHOULD retrieve the original From header from one of the c: tags of the signature, and include the Sender address (also retrieved from one of the c: tags of the signature) in the display-name of the address. For example:

From: "John Q. User via <asrg-admin@ietf.org>" <user@example.com>

This sort of address inconsistency is expected for mailing lists, but might be otherwise used to mislead the recipient, for example if a message supposedly from `administration@your-bank.com` had a Sender address of `fraud@badguy.com`.

[6.](#) Key Management

In order for these signatures to be meaningful, a method for verifying the validity of the key used to sign the message needs to be available. The PGP [\[7\]](#) approach to this issue of trust is through the use of trusted introducers, where individual keys are signed by others that may be trusted by the user of the public key. However, such a model does not scale well to very large communities of users where several generations of trust would be required.

Another approach to this problem would be through the use of X.509 certificates. While certificates are attractive for many types of transactions, many consider it undesirable to require that any domain wishing to send signed mail must obtain a certificate through a recognized certificate authority. The ability to quickly and easily revoke the authorization for keys, especially in the case of user-level keys, is also a problem that is most easily solved by consulting the originating domain.

In IIM the method of verifying the validity of the key is an online query sent to the signing domain, or to a server designated by that domain. This process is referred to as a key registration or key authorization check. A hash or fingerprint of the key being verified is used to lookup a record in a database operated by the signing domain. Since the use of a fingerprint makes the size of the data required to verify key authorization independent of the length of the key (or certificate), PGP-style signed keys or X.509 certificates could be sent in messages as well, verified using the same mechanisms, and could perhaps be used for other purposes in addition to message signatures. The format for sending such keys and certificates is beyond the scope of this document.

[6.1](#) Key Registration

In order to receive email messages, domains typically use one or more MX (mail exchanger) resource records to indicate to where mail for that domain should be directed. Similarly, DNS resource records can be used directly or indirectly to verify the authorization of keys used to sign email messages, or to locate one or more hosts which may be considered authoritative to verify the association of keys with email addresses in the domain.

A new textual DNS resource record, referred to as a KR record, is defined for publishing and accessing information relating to key registration for a domain. The value 1010 (decimal) is being used for the KR record type in experimental use; this will need to change if and when IANA assigns a record type. An alternative method of publishing and accessing key registration information using TXT

records is described in [Section 8](#).

To accommodate different deployment needs, two methods of determining the authorization of public keys are defined. The choice of authorization method to be used for a particular message is specified by the signer in the method (m:) tag of the signature.

The first method uses HTTP to query a host, referred to as a Key Registration Server (KRS), which is located via a DNS record lookup. This method provides fine-grained control over key authorization; keys may be authorized for use with a list of specific email addresses, and multiple keys may be authorized for a given address. It also takes advantage of a great deal of existing infrastructure used to distribute web services, and can be hosted on an existing web server if desired.

The second method of verifying the authorization of a public key is to store the authorization directly in DNS KR records. This method is attractive for domains which have a relatively small number of domain-level keys, because there is no need to operate (or outsource the operation of) a KRS. This method is suitable primarily for keys which are authorized for an entire domain (typically used by outgoing MTAs).

Because of limitations imposed by DNS wildcards and the potential privacy issues with storing user email addresses in DNS records, the

KRS method SHOULD be used for user-level keys. For domains with only domain-level keys, authorization via either KRS or DNS MAY be used at the option of the sending domain. DNS may be easier for domains that have no externally-accessible Web server on which to run the KRS service; KRS may be easier for domains where the administration of mail services and name service is performed by different groups. Signing agents need only specify the form of key registration used by their domain. Verifying agents MUST support both the KRS and DNS methods.

In all cases, the domain referenced for authorization in the signature MUST be either the same as or a parent domain of the address being verified. For example, in order to verify the address tom@eng.example.com, the domain referenced by the signature could be eng.example.com or example.com, but not example2.com. Signatures violating this rule MUST be ignored.

[6.1.1](#) Key Authorization via KRS

The use of a Key Registration Server (KRS) provides maximum flexibility for domains which support user-level key authorization, and provides administrative separation between management of DNS

zones and email authorization.

A key registration server confirms (or denies) the binding between the specified email address used by the message and the key used to sign the message. It does so by receiving a query containing the key fingerprint (digest of the public key) and the email address being verified, which will be either the From or Sender address. It returns a value based on the policy of the sending domain as to whether the key is authorized to be used in sending a message from the specified address.

When the KRS method has been specified by the sender, the first step for the recipient is to consult its local cache of key authorizations, if any. If a result from a previous key authorization check has been cached and is still valid according to the time-to-live in the cached request, the verifier SHOULD use this result to establish whether the key is authorized for the address.

If the local cache check fails, the next step is to locate the

address(es) of the domain's KRS(es). This is done by doing a DNS lookup for a KR record for the domain specified in the d: tag of the signature. In order to satisfy this request, the zone file for the domain would contain records such as the following:

```
example.com. IN KR 10 10 378 "v:\"IIM1\"\\;s:\"200\"\\;  
                                k:\"http://www.example.com/KRS/\"\\;"  
example.com. IN KR 10 10 378 "v:\"IIM1\"\\;s:\"200\"\\;  
                                k:\"http://www2.example.com/KRS/\"\\;"
```

Once a URI for the KRS query has been obtained, verification of public keys from key registration servers is accomplished via a properly-formatted HTTP GET request. A sample request might be formatted as follows:

```
http://www.example.com/KRS/?domain=example.com  
    &name=john@example.com  
    &keyfp=WDQGpekHKCmKyKWk
```

The fields in the query are as follows:

name: The address (From or Sender) being verified.

keyfp: The public key fingerprint that was supplied in the IIM-Sig line. The fingerprint is created as follows: create the binary representation of the RSA exponent (e) and modulus (n) and concatenate them as e|n. Run this value through SHA1 over the full length and convert the first 12 bytes of the output of the SHA1 operation to base 64. That is, base64 (TRUNC (SHA1 ((e|n))),

12)

domain: The domain corresponding to the query to be performed. This is used primarily to allow a single KRS to support multiple domains, with each domain database being independently maintained. This value corresponds to the d: value in the signature being checked; it MUST be the same as or a parent domain of the address associated with the signature.

The following are some excerpts from a hypothetical KRS database:

#Auth	TTL	Address	Service	Key Fingerprint
-------	-----	---------	---------	-----------------

```
pass 86400 tom@eng.example.com SMTP 073FDD7DD6D6EF6D1413FD7B3C577EFC
# Tom's usual address
pass 86400 tom@example.com SMTP 073FDD7DD6D6EF6D1413FD7B3C577EFC
# Rewriting of Tom's address
pass 86400 dick@example.com SMTP 91881749E520D8F53B0B91BBDB8963D0D
# Dick's PC
pass 86400 dick@example.com SMTP 549D8949351DDA4E7C961E0F58727795
# Dick's PDA
fail 864000 harry@example.com SMTP 8C8252070CA9ED401DD2EE2A7B31A8CF
# Harry's stolen PC
pass 86400 harry@example.com SMTP 17E64AC44DD5F8891560919D3FC6EA52
# Harry's new PC
pass 86400 harry@example.com SMTP 073FDD7DD6D6EF6D1413FD7B3C577EFC
# Tom is Harry's administrative
# assistant, so Harry allows Tom
# to originate mail for him.
pass 604800 *@example.com SMTP 27985A61447CC8B514A82BFA4597174A
# Outgoing MTA key. MTA keys are
# less likely to require rapid
# revocation, hence the long TTL.
pass 86400 nobody@example.com SMTP *
# Any key will work for this addr
# NOT RECOMMENDED!
```

The above example illustrates much of the motivation behind creation of a network element, the KRS, for key verification. Support for multiple keys per address and multiple addresses per key would require, in general, wildcarding of both the key fingerprint and email address fields, something that is not possible in DNS. Direct key authorization via DNS would also require that users' email addresses be contained in DNS records, which might raise privacy concerns as DNS information is not considered private. Furthermore, the potentially large number and short time-to-live of user-level key authorization records may present loading issues for DNS.

The ability to configure multiple key registration servers for a

given domain is intended to provide a degree of fault-tolerance and distribution of the key-verification load. The availability requirement for key registration servers is somewhat higher than for mail exchangers (and probably more comparable to that of domain name servers) because real-time access to the key registration servers is

often required at the time an email message is received or relayed. Accordingly, each domain defining key registration servers SHOULD define at least two, and they SHOULD be located on different networks.

The key registration servers for a domain need to be kept in as close synchronization as possible. In particular, any key revocations that take place MUST be reflected immediately in all key registration servers for the domain.

This key management approach requires that only legitimate key-to-address bindings be registered on the key registration servers. Key registration servers MUST use a mechanism that ensures that only authorized users are able to deposit key fingerprints on the server and revoke them. This may involve a mechanism such as an authenticated HTTP exchange that requires the user's password in order to register a public key fingerprint for that user on the server.

In order to prevent harvesting of email addresses, KRSes MUST NOT respond with any email address other than that presented in the query or a more general address (for example, when the key fingerprint corresponds to a domain MTA).

[6.1.2](#) Key Authorization via DNS

In order to accommodate domains with a relatively small number of infrequently changing keys, a domain MAY choose to advertise the authorization of its keys via DNS. In the DNS model, caching of key authorization is provided by DNS itself, rather by a cache locally maintained by the verifier.

To check key authorization via DNS, the verifier forms a query for a KR record of the form <keyfp>.<domain>, where <keyfp> is the fingerprint of the public key, calculated as described above, expressed in base 64 format. A sample record is as follows:

```
WDQGpekHKCmKyKWk._krs.example.com. KR
"v:\"IIM1\"; s:\"200\"; r:\"100\"; t:\"3600\"; m:\"*@example.com\""
```

Unlike the KRS query method, key authorization via DNS is based only on the key fingerprint itself; the email address is not included in the query. This is done to simplify the query and because DNS does

not provide the wildcard functionality to support multiple specific email addresses being authorized for a particular key, as is possible with the KRS method.

[6.1.3](#) Null Key Checks

In the absence of an IIM signature on a message, it is desirable to provide a means by which the originating domain can express its policy on the signing of messages, and by implication its preference on how unsigned messages SHOULD be handled. This policy is determined through a process known as a Null Key Check.

When an unsigned message is received by a verifying MTA or MUA, a null key check SHOULD be performed. This check is performed by forming a DNS query for a KR record in the name of the "domain" in the message. Since there is no IIM-Sig containing a d: tag indicating the responsible domain, the null key check must be performed on the entire right-hand side of the email address of the From header, or in the case of a From header containing multiple addresses, the right-hand side of the email address in the Sender header. A sample DNS record is as follows:

```
example.com. KR "v:\"IIM1\"; s:\"200\"; a:\"fail\";  
                t:\"864000\"; m:\"*@example.com\""
```

The results are formatted as shown in [Section 6.1.4](#). Generally, only two of the three possible status values make sense: either the sending domain asserts that the message is to be presumed to be unauthorized, or that the message is of unknown authorization.

With only the above null key record in DNS, it might be possible for an attacker to avoid the null key check by using an address in an unknown subdomain of a legitimate domain (e.g., user@foo.example.com in the above example). For this reason, domains publishing a null key policy SHOULD publish both a record for their domain and a wildcard record covering subdomains. For example:

```
example.com. KR "v:\"IIM1\"; s:\"200\"; a:\"fail\";  
                t:\"864000\"; m:\"*@example.com\""  
*.example.com. KR "v:\"IIM1\"; s:\"200\"; a:\"fail\";  
                  t:\"864000\"; m:\"*@example.com\""
```

[6.1.4](#) Key Authorization Results

The KRS and DNS query methods share a common format for the query result with the exception that tagged values from DNS are quoted with single quotes rather than double quotes in order to make them easier

Internet-Draft

Identified Internet Mail

May 2005

to incorporate in typical zone files. Tags and their meanings are as follows:

- v: Version of the response. Currently this MUST be set to "IIM1". This tag MUST be present, and MUST be the first tag in the response. Responses not beginning with v:"IIM1" MUST be discarded.
- s: Status. Follows the general convention of SMTP/HTTP status values (i.e., 200, 300, 400, 500 semantics) with the following values defined:
 - 200: the lookup succeeded.
 - 201: the lookup succeeded, but the keyfp/name combination was not found
 - 500: any permanent failure.
- t: Time to live. Responses SHOULD be cached by the verifier so as to reduce the query/response load back to the KRS. Time to live is expressed in seconds from when the query was sent. This value is used only for KRS responses and is ignored if present in DNS responses. This value is OPTIONAL, and if absent, the response is not cached by the verifier.
- a: Authorization. This tag contains the authorization status of the given name and key fingerprint association. A value of "pass" indicates that the KRS/DNS approves of this key fingerprint/name combination. A value of "fail" indicates the KRS/DNS doesn't approve, because the key is unknown, unapproved for this name, revoked, or for any other reason. A value of "Unknown" indicates that the KRS/DNS doesn't have any specific information one way or the other. For signed messages "unknown" doesn't make much sense, but in the case of an unsigned message where the domain cannot ensure that all of its outgoing mail is signed, "unknown" status is probably appropriate. The verification in this case SHOULD treat the mail as if were unsigned.
- r: Rating. Like rating in the IIM-Verify, an integer between -127 and 127 which is at the sole discretion of the entity producing the rating. Normally, revoked keys from the home KRS would be given a (very) negative rating. This tag is REQUIRED unless the a: tag is present.

m: Matches. Some key fingerprints may in fact sign for more than the single address that is present in the query. In order to cut down trips to the KRS, the Matches field describes with normal Unix wildcard syntax what address patterns match this key fingerprint. This tag is REQUIRED. For example, m:"*@example.com" would inform

the cache logic of the requester that future queries from example.com with this key fingerprint be given the same rating.

k: KRS. Specifies the URL for a KRS to be used to complete the request. This tag allows the address of a KRS to be specified in a DNS record, which is referenced as described in [Section 6.1.1](#). If present, all other tags with the exception of v: and s: will be ignored. This tag is OPTIONAL and MUST NOT appear in a response from a KRS.

c: Comment. This is a free form string intended to convey a human readable comment about the operation. This is typically used to send diagnostic information for failed operations, etc. This tag is OPTIONAL.

Note that while the syntax of the matching pattern uses normal unix wildcard syntax, the semantics of the wildcarding are actually constrained to be a "longest prefix match" algorithm where the prefix components are allowed to be either the left hand side of the email address, or the successive subdomain components. In all cases, the scope of a Matches value MUST NOT exceed the domain of the KRS or DNS lookup used to retrieve the authorization. That is, an entity from example.com cannot say that it matches *@*.com since it is not authorized to sign for all .com domains.

[6.2](#) Policy Options

Identified Internet Mail by itself introduces no new policy with respect to handling email. However, the benefit of using IIM lies with the widespread deployment of policy which encourages the signing of email and eventually marginalizes unsigned messages.

One place where policy MAY be implemented is at the receiving user. The user MAY verify the signatures of messages as they are received, and place unsigned messages in a "bulk mail" or similar folder to be

read (if at all) on a lower-priority basis. This would typically be done through an enhancement to the mail user agent, probably at the time messages are downloaded via a protocol such as POP. Since the user must verify the authorization of all keys not in the user's key cache, this could lengthen message downloading times, and may present a problem for transitory users such as those on dial-up lines.

A recipient or intermediate MTA MAY verify the message signatures and add a verification header to incoming messages. This considerably simplifies things for the user, who can now use an existing mail user agent. Most MUAs have the ability to filter messages based on message headers or content; these filters would be used to implement whatever policy the user wishes with respect to unsigned mail.

A verifying MTA MAY implement a policy with respect to unsigned mail, regardless of whether or not it applies the verification header to signed messages. Separate policies MAY be defined for unsigned messages, messages with incorrect signatures, and in cases where the signature cannot be verified, for example if all the key registration servers are unreachable. Treatment of unsigned messages SHOULD be based on the results of the null key check described in [Section 6.1.3](#).

If the verifying MTA is able to verify the public key of the sender and check the signature on the message as the message is received, the MTA MAY reject the message with an error such as:

5yx Unsigned messages not accepted
5uv Message signature incorrect

If it is not possible to verify the authorization of the public key in the message, perhaps because the key registration server is not available, a temporary failure message could be generated, such as:

4yx Unable to verify signature - key registration server unavailable

[7.](#) Usage examples

[7.1](#) Simple message transfer

The above sections largely describe the process of signing and verifying a message which goes directly from one user to another. One special case is where the recipient has requested forwarding of the email message from the original address to another, through the use of a Unix `.forward` file or equivalent. In this case the message is typically forwarded without modification, except for the addition of a Received header to the message and a change in the Envelope-to address. In this case, the eventual recipient should be able to verify the original signature since the signed content has not changed, and attribute the message correctly.

[7.2](#) Outsourced business functions

Outsourced business functions represent a use case that motivates the need for user-level keying. Examples of outsourced business functions are legitimate email marketing providers and corporate benefits providers. In either case, the outsourced function would

like to be able to send messages using the email domain of the client company. At the same time, the client may be reluctant to register a key for the provider that grants the ability to send messages for any address in the domain.

With user-level keying, the outsourcing company can generate a keypair and the client company can register the public key for a specific address such as promotions@example.com. This would enable the provider to send messages using that specific address and have them verify properly. The client company retains control over the email address because it retains the authority to revoke the key registration at any time.

[7.3](#) PDAs and Similar Devices

PDAs are one example of the use of multiple keys per user. Suppose that John Doe wanted to be able to send messages using his corporate email address, jdoe@example.com, and the device did not have the ability to make a VPN connection to the corporate network. If the device was equipped with a private key registered for jdoe@example.com by the administrator of that domain, and appropriate software to sign messages, John could send IIM messages through the outgoing network of the PDA service provider.

[7.4](#) Mailing Lists

There is a wide range of behavior in forwarders and mailing lists

(collectively called "forwarders" below), ranging from those which make no modification to the message itself (other than to add a Received header and change the envelope information) to those which may add headers, change the Subject header, add content to the body (typically at the end), or reformat the body in some manner.

Forwarders which do not modify the body or signed headers of a message with a valid signature MAY re-sign the message as described below. Forwarders which make any modification to a message that could result in its signature becoming invalid SHOULD re-sign using an appropriate identification (e.g., mailing-list-name@example.net), but normally they SHOULD do so only for messages which were received with valid signatures or other indications that the messages being signed are not spoofed.

Forwarders which wish to re-sign a message MUST apply a Sender header to the message to identify the address being used to sign the message and MUST remove any pre-existing Sender header as required by [RFC 2822](#) [5]. The forwarder applies a new IIM-Sig header with the signature, public key, and related information of the forwarder. Previously existing IIM-Sig headers SHOULD NOT be removed.

[7.5](#) Affinity Addresses

"Affinity addresses" are email addresses users employ to have an email address that is independent of any changes in email service provider they may choose to make. They are typically associated with college alumni associations, professional organizations, and recreational organizations with which they expect to have a long-term relationship. These domains usually provide forwarding of incoming email, but (currently) usually depend on the user to send outgoing messages through their own service provider's MTA. They usually have an associated Web application which authenticates the user and allows the forwarding address to be changed.

With Identified Internet Mail, affinity domains could use the Web application to allow users to register their own public keys to be used to sign messages on behalf of their affinity address. This is another application that takes advantage of user-level keying, and domains used for affinity addresses would typically have a very large number of user-level keys. Alternatively, the affinity domain could decide to start handling outgoing mail, and could operate a mail submission agent that authenticates users before accepting and signing messages for them. This is of course dependent on user's service provider not blocking the relevant TCP ports used for mail submission.

[7.6](#) Third-party Message Transmission

Third-party message transmission refers to the authorized sending of mail by an Internet application on behalf of a user. For example, a website providing news may allow the reader to forward a copy of the message to a friend; this is typically done using the reader's email address. This is sometimes referred to as the "Evite problem", named

after the website of the same name that allows a user to send invitations to friends.

One way this can be handled is to continue to put the reader's email address in the From field of the message, but put an address owned by the site into the Sender field, and sign the message on behalf of the Sender. A verifying MTA SHOULD accept this and rewrite the From field to indicate the address that was verified, i.e., From: John Doe via news@news-site.com <jdoe@example.com>.

[8.](#) DNS considerations

Any discussion of key management rooted in DNS would be incomplete without addressing the choice of DNS records for that task. The architecturally preferred method is to use a new resource record type, but in practice some resolvers, DNS servers, and firewalls cannot accommodate a new resource record type. The common workaround for that problem is the use of an existing record such as TXT, perhaps with a distinguishing application-dependent prefix in order to avoid collisions with other uses of TXT records.

As discussed in [Section 6.1.3](#), wildcard DNS records are required to close a possible attack against the null key check. The use of a prefix would interfere with the with the required wildcard functionality. For that reason, null key records cannot use a distinguishing prefix. This does cause extra collisions with other uses of the domain's root TXT record space, which could increase the likelihood that a TXT record lookup for the domain will exceed the maximum UDP response size.

A compromise using both a new resource record (RR), known as a KR record, and TXT records is proposed.

Wherever a KR record including a key fingerprint is published, an identical TXT record **SHOULD** be published. The `_krs` prefix is used in all records with the exception of null key records. This prefix is used in both KR and TXT records to provide greater consistency between the corresponding resource records.

The use of the `_krs` prefix for all other key authorization data other than the null key record is intended to minimize collisions with other TXT records and to allow the `_krs` prefix to be delegated to the mail administrator, in situations where mail and DNS administration are done by different organizations.

Verifying agents **SHOULD** use the KR record if it is available in their environment. If the KR lookup fails, a lookup for the corresponding TXT record **SHOULD** be performed. The intent is to encourage the use of KR records. However, no migration strategy to eliminate the use of TXT records has been defined.

For brevity, examples show the use of the KR record only, but publication and lookup of the corresponding TXT records **SHOULD** be performed as well.

Internet-Draft

Identified Internet Mail

May 2005

[9.](#) Security Considerations

Fundamentally, the addition of signatures to email messages is all about security, although not in the usual way of ensuring the secrecy of data.

[9.1](#) Potential Attacks

It has been observed that any mechanism that is introduced which attempts to stem the flow of spam is subject to intensive attack. Identified Internet Mail needs to be carefully scrutinized to identify potential attack vectors and the vulnerability to each. Some of the attacks that have been considered are described in the following sections.

[9.1.1](#) Key Registration Server Denial-of-Service Attack

Since the key registration servers are distributed (potentially separate for each domain), the number of servers that would need to be attacked to defeat this mechanism on an Internet-wide basis is very large. Nevertheless, key registration servers for individual domains could be attacked, impeding the verification of messages from that domain. This is not significantly different from the ability of an attacker to deny service to the mail exchangers for a given domain, although it affects outgoing, not incoming, mail.

A variation on this attack is that if a very large amount of mail were to be sent using spoofed addresses from a given domain, the key registration servers for that domain could be overwhelmed with requests. However, given the low overhead of HTTP requests to the KRSes compared with handling of the email message itself, such an attack would be difficult to mount.

[9.1.2](#) Key Registration Server Stall Tactic

An attacker trying to "jam" the signature mechanism might set up a key registration server for a domain they control that responds very slowly or perhaps not at all. They then send a large number of messages from that domain, in an attempt to bring the signature verification mechanism to a crawl and get domains to turn it off. This could be mitigated by the use of appropriate timeouts on key lookups and possibly by adapting these timeouts to message load. Note that it is considerably easier to mitigate this attack when the

signature check is done by the terminating MTA than the MUA because of the MTA's ability to return a temporary failure when the key can't be retrieved.

[9.1.3](#) Misappropriated private key

If the private key for a user is resident on their computer and is not protected by an appropriately secure passphrase, it is possible for malware to send mail as that user. The malware would, however, not be able to generate signed spoofs of other senders' addresses, which would aid in identification of the infected user and would limit the possibilities for certain types of attacks involving socially-engineered messages.

A larger problem occurs if malware on many users' computers obtains the private keys for those users and transmits them via a covert channel to a site where they may be shared. The compromised users would likely not know of the misappropriation until they receive "bounce" messages from messages they are supposed to have sent. Many users may not understand the significance of these bounce messages and would not take action.

One countermeasure is to use a passphrase, although users tend to choose weak passphrases. Nevertheless, the decoded private key might be briefly available to compromise by malware when it is entered, or might be discovered via keystroke logging. The added complexity of entering a passphrase each time one sends a message would also tend to discourage the use of a secure passphrase.

A somewhat more effective countermeasure is to send messages through an outgoing MTA that can authenticate the sender and will sign the message using its key which is normally authorized for all addresses in the domain. Such an MTA can also apply controls on the volume of outgoing mail each user is permitted to originate in order to further limit the ability of malware to generate bulk email.

[9.1.4](#) Message Replay Attack

In this attack, a spammer sends a message to be spammed to an accomplice, which results in the message being signed by the

originating MTA (presuming that the sender doesn't have a valid individual key for the domain). The accomplice resends the message, including the original signature, to a large number of recipients, possibly by sending the message to many compromised machines that act as MTAs. The messages, not having been modified by the accomplice, have valid signatures.

Several techniques for dealing with this type of attack have been considered. One is to include in the request to the KRS not only the fingerprint of the signing key but also a fingerprint of the signature which would allow the KRS to detect, and flag as unauthorized, the use of the same signature a very large number of

times. This would require the KRS to maintain a cache of signature fingerprints, and would make caching of key registration data by verifying MTAs and MUAs impossible. Both of these factors are very undesirable from a performance standpoint. In addition, some checking of message date/time fields would need to be introduced in order to allow aging of the signature cache, where currently there is no assumption that the sender have a valid real-time clock.

A similar approach is for verifying MTAs and MUAs to cache the signatures themselves and detect duplications. However, only large recipient MTAs are likely to process enough of the spam messages in order to detect the duplications. Furthermore, there are legitimate use cases involving mail forwarding where the same message might take different paths to the same MTA, so this can only be applied in cases where unexpectedly large numbers of duplicate signatures are received.

Other partial solutions to this problem involve the use of reputation services to convey the fact that the specific email address is being used for spam, and that messages from that sender are likely to be spam. This requires a real-time detection mechanism (such as detection by the KRS as described above) in order to react quickly enough. However, such measures might be prone to abuse, if for example an attacker resent a large number of messages received from a victim in order to make them appear to be a spammer.

[9.2](#) Other Considerations

At present, a message can be forwarded giving the sender no

indication as to the recipient's actual location (IP address, domain, or eventual email address) on the Internet. A sender monitoring queries to its KRS might be able to infer some of this when the recipient's MTA, or even the actual recipient, checks the identification of incoming messages. In cases where this may be sensitive, trusted proxies SHOULD be employed by the recipient and/or their domain.

[10.](#) IANA Considerations

Use of the `_krs` prefix in TXT records will require registration by IANA. IANA will also need to allocate a permanent DNS resource record number for the newly-defined KR record type.

[11.](#) Acknowledgements

Dave Rossetti provided much of the original motivation to address this problem. In addition, thanks to Fred Baker, Mark Baugher, Patrik Faltstrom, Don Johnsen, Dave Oran, Shamim Pirzada, Sanjay Pol, Christian Renaud, and Dan Wing for their suggestions and much helpful discussion around this issue.

[12.](#) References

[12.1](#) Normative References

- [1] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", [RFC 2045](#), November 1996.

- [2] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", [RFC 2047](#), November 1996.
- [3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [4] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", [RFC 2234](#), November 1997.
- [5] Resnick, P., "Internet Message Format", [RFC 2822](#), April 2001.
- [6] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", [RFC 3447](#), February 2003.

[12.2](#) Informative References

- [7] Atkins, D., Stallings, W., and P. Zimmermann, "PGP Message Exchange Formats", [RFC 1991](#), August 1996.

Authors' Addresses

Jim Fenton
Cisco Systems, Inc.
MS SJ-24/2
170 W. Tasman Drive
San Jose, CA 95134-1706
US

Phone: +1 408 526 5914
Email: fenton@cisco.com

Cisco Systems, Inc.
MS SJ-9/2
170 W. Tasman Drive
San Jose, CA 95134-1706
US

Phone: +1 408 525 5386
Email: mat@cisco.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

Fenton & Thomas

Expires November 14, 2005

[Page 39]