DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification

Barry Leiba
IBM Research
Hawthorne, NY
leiba@watson.ibm.com

Jim Fenton
Cisco
San Jose, CA
fenton@cisco.com

ABSTRACT

Email protocols were designed to be flexible and forgiving, designed in a day when Internet usage was a cooperative thing. A side effect of that is that they were not designed to provide protection against falsification of a message's address of origin, referred to today as "spoofing". DomainKeys Identified Mail (DKIM) defines a mechanism for using digital signatures on email at the domain level, allowing the receiving domain to confirm that mail came from the domain it claims to. In conjunction with the forthcoming DKIM sender signing practices specification, the receiving domain may also have more information for deciding how to treat mail without a valid signature. The use of DKIM signatures and signing practices gives sending domains one tool to help recipients identify legitimate messages from their domain, and a reliable identifier that can be used to combat spam and phishing.

1 INTRODUCTION

Early antispam filtering involved "blacklisting" the senders of spam – refusing to accept or deliver mail from email addresses known to send spam. Unfortunately, the Internet standards for email do not prevent the sender from lying about his identity, at the protocol level [8], in the mail "headers" [14], or both. This "spoofing", as it's called, not only allows spammers to get around email-address blacklists, but also to lend credibility to their messages by spoofing a reputable domain. Initially a way simply to convince recipients to open the messages, rather than to delete them, spoofing reputable domains has evolved into a con-game called "phishing", resulting in estimated losses in 2004 of between one and two billion dollars [15],[9].

Clearly, something must be done to curtail spoofing; the ability to send messages while purporting to be another sender is in most cases undesirable. While curtailment will not *stop* phishing, and while spoofing cannot be stopped entirely without significant (and arguably undesirable) effects on Internet email as it is known today, making spoofing more difficult and providing domains with ways to protect their names and reputations are important steps against spam and phishing.

There have been two broad mechanisms proposed for *domain validation* – verifying that mail did or did not come from the domain it claims to have come from. One uses IP address; the other uses digital signatures. In the former category are SPF (Sender Policy Framework [16]), and Sender ID [11], related techniques that differ in some details. CSV (Certified Sender Validation [4]) also falls into this category.

In the second category are techniques that have the sender, or the sending domain, place a digital signature on the message. The signature can be verified later, by the recipient or by the receiving domain, and the verified signature can be used as evidence that the mail originated from where it says it does.

The two categories each have advantages, and are not in competition. It is important to note, in this discussion, that the use of many techniques, together, is the most effective way to combat spam and related maladies (phishing, viruses and worms, and other malware distributed through email) [10]. Discussion of the advantages and disadvantages of the two categories is outside the scope of this paper, which will focus on the design and deployment of one particular specification: DomainKeys Identified Mail.

The remainder of this paper will give an overview of DKIM, will discuss details of the mechanisms used and some of the choices made, and will show some practical deployment experience.

2 AN OVERVIEW OF DKIM

The concept behind DKIM is simple: If you receive a message from me bearing a valid digital signature, then you can be sure that it actually came from me. There are signature techniques already standardized for applying signatures to email, such as S/MIME [13] and OpenPGP [3], although the meaning of these signatures is subtly different from that of a DKIM signature.

There are a few problems, though, with using these pre-existing techniques:

1. They assume that the recipient's mail system knows how to deal with the signed messages. If it does not, the recipient sees a message cluttered with unintelligible things.

2. The message signature formats do not sign the message headers, and we'd like to protect the headers under the signature. There are ways to accomplish that with S/MIME and OpenPGP, but they result in an even worse experience for non-compliant recipients.

CEAS 2007 – Fourth Conference on Email and Anti-Spam, August 2-3, 2007, Mountain View, California USA

3. There is no mechanism, in the general case, for communicating the knowledge that I sign all my mail. What can work fine for pairs of known communication partners does not work in an environment where you want to receive mail from an Internet full of previously unknown senders.

Furthermore, there is a difference in the assertions made by DKIM from those made by S/MIME and OpenPGP. DKIM is designed to provide the *domain owne*r with control over the use of addresses in the domain, and the validity of keys used to sign messages in the domain is under the domain owner's control. On the other hand, the keying models used by S/MIME and OpenPGP do not necessarily involve the participation of the domain owner. This distinction becomes important when one considers that an ex-employee of a corporate domain, or an ex-customer of an ISP, might have a valid OpenPGP key or S/MIME certificate even though they no longer are authorized to use their former addresses in the domain.

DKIM defines a mechanism that "corrects" these problems by…

1. …putting the signature information into the message in a way that is transparent to most end users, and to systems that do not understand the signature mechanism.

2. …allowing the signer to include selected headers.

3. …defining, in an accompanying specification, "sender signing practices", allowing senders to communicate information about their practices to potential recipients.

The DKIM base specification [1] tells signers how to create the signatures and include them in their messages, and tells verifiers how to interpret and verify the signatures. The DKIM signing practices specification [2] tells senders how to specify their signing practices, and tells verifiers how to retrieve that information and use it. Taken together, the two specifications provide one method of defense against spoofing.

The DKIM base specification has recently been published (May 2007) by the IETF as a Proposed Standard, RFC 4871. The details of the base specification are, therefore, stable.

2.1 The Scope of DKIM

In the introductory discussion above, we talked about signing mail between "you" and "me". While DKIM can be used with that scope, it is not how DKIM is intended to be deployed. As suggested by the name, "*Domain*Keys Identified Mail", it is intended to be used at the *domain* level. A typical DKIM deployment would have a message signed by a mail transfer agent (MTA) of the *sending domain* before the message is sent out of that administrative domain. When the message reaches the domain of its intended recipient, an MTA in that *receiving domain* would verify the signature. Of course, any intermediate domain could also verify the signature, and could add its own signature as well, adding it to or replacing the original. Each of these cases will be discussed below in more detail.

The basic use case is shown in Figure 1, where jane@example.com sends a message to john@example.net. In this case, the DKIM signing is done at gway.example.com, and the verification is done at inet.example.net.

Because of this scope, most of the discussion in this paper will refer to the sending domain and the receiving domain, and will call them the *signer* and the *verifier*, respectively. We will occasionally make the distinction, as needed, between the *sending domain* and the individual *sender*, and between the *receiving domain* and the individual *recipient*.

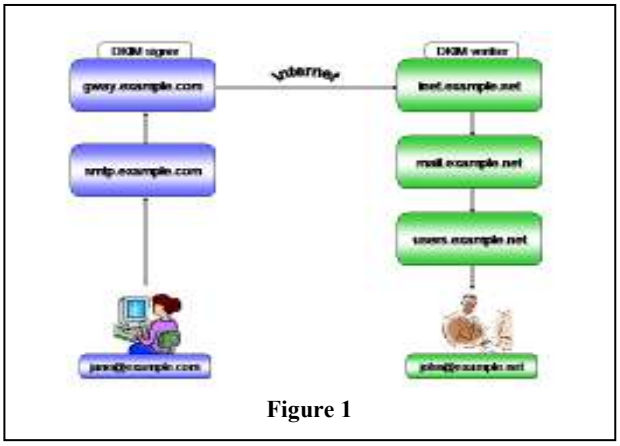

Figure 1

2.2 What DKIM Does for the Signer

DKIM signatures allow a signer to take responsibility for having placed a message into the network. The addition of signing practices allows a sending domain to convey information to verifiers about how it chooses to sign the mail it originates. This can give a domain the ability to defend its name against improper use, and to protect its reputation (see the discussion of signing practices, below; this advantage is limited, in the short term, until most recipients verify DKIM signatures). It may also allow signed mail to be handled preferentially by receiving domains that "trust" the sending domain in some sense.

2.3 What DKIM Does for the Verifier

DKIM signatures allow a verifier to determine that an email message did, indeed, come from the domain it says it did. This information can allow a verifier to "whitelist" a sending domain, for example by permitting verified messages from that domain to bypass more stringent inspection – inspection that may take more time and resources, and might be subject to *false positives* that could prevent the delivery of legitimate mail. Blacklists of signed domains can work in a similar fashion, although the case for whitelisting is more compelling. Note that signatures, by themselves, do not give verifiers any useful information about unsigned mail.

The addition of signing practices *does* provide such useful information, for sending domains that publish practices. By saying, for example, "We sign all mail originating from our domain," they allow the verifier to make a decision about how to handle unsigned mail – in this case, a verifier may choose to treat

an unsigned message with extra suspicion, or to discard it outright, at its discretion.

2.4 What DKIM Does NOT Do

This cannot be over-emphasized: DKIM is not, directly, an antispam technique. Rather, DKIM is expected to *enable* antispam and anti-phishing mechanisms, by making it harder to spoof legitimate domain names that participate in DKIM signing.

DKIM does not provide encryption, nor any other privacy features. Further, while its design allows for signing authority to be delegated from the domain owner to individual users, it is not meant for the use cases for which S/MIME and OpenPGP were designed. DKIM signers are making no assertions about having been the author of the *content* of the messages. For those sorts of features, S/MIME or OpenPGP are what senders should use (and DKIM can still work on top of that).

DKIM does not guarantee that a signed message will arrive undamaged. While DKIM pre-processes messages to minimize the chance of corruption (see the discussion of *canonicalization*, below), MTAs and mail gateways do change headers and bodies of email messages in ways that may make signatures unverifiable.

3 DKIM SIGNATURES

The DKIM specifications (*q.v.*) are the normative sources for the details of signatures, signing practices, and verification, and we will not repeat them here. This section, and the ones that follow, are meant to give enough information to understand how the system works – how signatures are created, represented, and verified, and what the signing practices do.

Once a signer has decided to sign a message, it must take the following steps, each of which we will discuss in more detail below:

1. Begin building the DKIM signature header.
2. Canonicalize and hash the message.
3. Select headers to be included in the signature.
4. Generate a cryptographic hash of the canonical message.
5. Generate a digital signature of the hash.
6. Add the DKIM signature header to the message.

3.1 The Signature Header

The DKIM signer must begin building the DKIM-Signature header now, since choices made through the process will be included in the header, and the header itself (minus the signature data, of course) will be covered by the signature. The first choices to go into the header are the domain and identity to be signed, and the *selector* to be used to identify the signing key.

In the simplest use case, the domain of the signing entity (the "d=" field in the signing header) is, of course, the domain doing the signing, and the identity of the signing agent (the optional "i=" field) is the same. In the example in Figure 1, above, *gway.example.com* would use *d=example.com*, and would omit the identity field. But suppose example.com were a hosting service, hosting different customers at *bank.example.com* and *store.example.com*. The hosting service might offer DKIM signing as part of the service, but would want to identify these separately. In that case, a message signed for the latter might use *d=example.com; i=@store.example.com*.

In order that different keys may be used in different circumstances for the same signing domain (allowing expiration of old keys, separate departmental signing, or the like), DKIM defines a *selector*, a name associated with a key, which is used by the verifier to retrieve the proper key during signature verification. The selector goes into the "s=" field.

The "q=" field must contain the name of the mechanism to be used to retrieve the verification key. This field exists to allow extension of DKIM to various key-management and key-distribution services. The current DKIM specification defines only one value, *q=dns*, which tells the verifier to retrieve the key using Domain Name Service (DNS), as described in the specification.

3.2 Canonicalization

The next choices to go into the signature header are the canonicalization algorithms for the headers and for the body. The names of these algorithms go into the "c=" field (as, for example, "c=simple/simple").

Canonicalization is necessary because of the long history of Internet email, the changes that have been made through that history, and the uncertainty of what a message may encounter en route to its destination. All email was once 7-bit US-ASCII, and, while much of the Internet now supports 8-bit ASCII, having that support at every node the message will traverse is uncertain. There are other, similar issues involving character encodings used, treatment of trailing white-space in message lines, "folding" and "unfolding" of header lines [14], and more.

The intent of canonicalization is to make a minimal transformation of the message (for the purpose of signing; the message itself is not changed, so the canonicalization must be performed again by the verifier) that will give it its best chance of producing the same canonical value at the receiving end. DKIM defines two header canonicalization algorithms ("simple" and "relaxed") and two for the body (with the same names). Experimentation so far has shown that if messages are sent with proper standard character encodings, "simple" is generally sufficient for the body.

Following the canonicalization process, the signer calculates a hash of the canonicalized message body using a hash algorithm as described in Section 3.4. The resulting "body hash" value becomes part of the DKIM-Signature header field, and provides additional information for diagnosing invalid signatures.

3.3 Selecting Header Fields

DKIM allows the signer to choose to sign some or all of the message header fields. Since many of the header fields do not

contain information significant to the sender or recipient of the message, signers might choose not to sign them all. Header fields are the parts of the message that are most vulnerable to change in transit, so leaving insignificant header fields unsigned may increase the chance that the signature can be successfully verified (at the expense of allowing some tampering, so the signer must make a trade-off here).

Because end-user mail programs (*mail user agents*, or MUAs) usually display the value of the message's *From* header to the end user, and since that is the primary target of spoofing, we consider it important that that header field be signed, and so DKIM requires the inclusion of *From* in the list of signed header fields.

Apart from those, DKIM strongly recommends the signing of *Subject*, *Date*, and all the MIME headers (such as *Content-Type*; note that MIME headers on message parts are not part of the message headers, but are automatically signed as part of the message body). It specifically advises against signing headers known to be removed or modified in transit (such as *Return-Path*), and suggests signing all others (the other side of the trade-off mentioned above).

The signer puts the list of the header names that will be signed into the "h=" field of the signature header. Header fields are signed in the order that they appear in this field, so the signature will be robust against header reordering in transit. The signature header itself (*DKIM-Signature*), absent the signature (the value of the "b=" field) is always signed, and is not explicitly listed in the list of signed headers.

3.4 The Hash

DKIM allows for multiple hash and signature algorithms, to provide for a transition to newer algorithms as it becomes advisable to switch to them. Unlike the case with HTTP clients and servers, for example, where an encryption suite can be negotiated at the time of the transaction, Internet email requires us to make a static choice and hope that the recipient understands the suite we have chosen. It is therefore not the intent to support multiple algorithms at the same time except to provide for such transitions.

There is currently one hash algorithm allowed by the DKIM specification: SHA-1 [5]. While hash collision issues have been discovered with SHA-1, we believe that those issues are not relevant to DKIM at this time (see the DKIM specification for a discussion of this). Still, transition to SHA-256 [*ibid.*] is likely soon, and might likely happen before DKIM becomes a Proposed Standard.

The hash algorithm name is the second part of the value of the "a=" field in the signature header (see below), and the hash is performed on the catenation of the canonical set of signed headers, which includes the body-hash value in the DKIM-Signature.. Before the combined hash is done, the signer may add optional "t=" and "x=" fields to the signature header, to specify the time the signature is being created and the time the signature will expire. The signer may also add the optional "l=" field to specify the body length that has been signed, which will allow the verifier to easily determine if additional text has been appended to the message in transit (as is done by some mailing-list handlers and forwarding services).

3.5 The Signature

As with hash algorithms, DKIM allows for transition of encryption algorithms by naming the algorithm in the "a=" field. The only currently supported encryption algorithm is "rsa" (PKCS#1 [7]), so signers must currently use *a=rsa-sha1* in their DKIM signature headers.

The signer signs the hash, using the specified encryption algorithm, puts the resulting signature into the "b=" field of the signature header, and adds the signature header to the beginning of the message header fields. An example of a completed signature header is shown in Figure 2.

```
DKIM-Signature: a=rsa-sha1; q=dns/txt; c=simple/simple;
    d=example.com; s=appliances;  i=@store.example.com;
    t=1117574938; x=1118006938; h=from:to:subject:date;
    bh=alIzndU2Nzg5jsEypzQ1njc4OTAxejr0NTY3ODkwdTI=;
    b=dzdVyOfAKCdLXdJOc9G2q8LoXSlEniSbav+yu
      U4zGeeruD00lszZVoG4ZHRNiYzR
```

Figure 2

4 DKIM SIGNING PRACTICES

Sender signing practices are a less-mature aspect of DKIM, and more experimentation and experience is needed to iron out the final details. We will describe here the current specification and discussions, and the issues in question.

As currently defined, senders may say one of the following things in their signing practices:

1. All messages from this entity *are* signed. Signatures created by third parties (mailing lists, etc.) are acceptable.

2. All messages from this entity *are* signed, and signatures created by third parties should *not* be accepted.

Signing practices can be defined separately for subdomains, with the parent domain's practices taking effect for unspecified subdomains. A bank that worries about phishing attacks against its customers could, for instance, create two subdomains, and use one (call it *official.bank-example.com*) for sending official mail, and the other (say, *people.bank-example.com*) for email that its employees use for less-sensitive situations, such as subscribing to (and posting to) mailing lists. The former would use signing practice 2, while the latter might use practice 1, or even omit the specification of signing practices altogether. Customers would be told to expect that all official mail from the bank would come from *official.bank-example.com*, and that any mail from addresses there that did not have a verified signature should not be believed.

There are still many considerations of how this will actually work, what heed will be paid to the signing-practices information, what unintended assumptions will be made by verifiers, and how this may be attacked by spammers and phishers. This aspect of DKIM will be evolving over the coming months, as there is more community discussion and more experimentation.

5 DKIM VERIFICATION

When a DKIM-compliant MTA receives an email message, that it decides it must verify (in the example in Figure 1, *inet.example.net* has received a message from *gway.example.com*), the message may be signed, or unsigned.

The message is considered to be signed if there is a *valid* DKIM-Signature header. The verifier must carefully check the signature header for validity.

5.1 Verifying a DKIM Signature

Using the contents of the *i*=, *d*=, and *s*= fields in the signature header, the verifier determines the desired key identity, and then uses the *q*= field and retrieves the key from the specified key store. For *q*=*dns*, the key is retrieved by getting DNS TXT records for "*selector*._domainkey.*domain*" (for the example in Figure 2, the records retrieved would be for *appliances._domainkey.example.com* (note that it does *not* use *store.example.com*, so in the case described there it is up to the *example.com* domain to keep track of which selectors are associated with which hosted subdomains). The verifier must then validate the retrieved key record, and extract the public key from it. Any failures in this process result in the signature's being declared invalid.

The verifier now uses the *c*=, *h*=, and *l*= (if present) fields to re-create the canonical message as originally signed. Using the *a*= field to determine the hash and encryption algorithms, it then computes the hash on the canonical message, decrypts the signature, and compares the two resulting hash values. If they are the same, then the signature is verified. If they are not, the signature is declared invalid.

5.2 Checking the Signing Practices

If there is *no* valid signature, or if the signing identity does not match the address in the message's *From* header, the verifier must check the signing practices of the domain in the *From* address. The verifier retrieves the policy through a DNS query. The *domain* for the query is obtained from the *From* address (see the signing practices specification [2] for more details, but, again, remember that this is still in flux).

5.3 The Verifier's Decision

Ultimately, what the verifier does with all this information – whether a signature was present or not, whether it verified or not, what the sender's signing practices say – is entirely up to the verifier. Verifiers may certainly treat messages with failed signatures as being more "suspicious" than those lacking signatures, but there are reasons for message signatures to fail (due to changes in transit) that do not reflect on the legitimacy of the message. Similarly, if the absence of a signature is considered worse than a failed signature, spammers will simply learn to put fake signatures on messages.

So the decision of what to do is a complex one, and involves more knowledge than DKIM alone provides. Verifiers may learn from patterns that they see themselves. Reputation and accreditation services may arise to provide recommendations beyond what the senders' own signing practices suggest. As noted before, the information can be used to help decide whether to subject the message to more scrutiny, with more or less aggressive spam filters, or to allow the message to bypass such processing.

Finally, the verifier may choose, apart from the options above, to convey some or all of the information to the final recipient of the message. Eventually, with a standardized mechanism to convey this information, MUAs can use this to alert the user to the trustworthiness (or lack thereof) of the message. For example, an MUA might display a verified *From* address in a different way than one that is not verified, so when a user gets mail from her bank, she can glance at the *From* field and make sure it's green (or has a check mark next to it, or some such indication). While DKIM is designed to operate in the infrastructure, MUA support will be key to maximizing its value.

6 DKIM DEPLOYMENT EXPERIENCE

At this writing there are a dozen or so independent implementations of DKIM that have been tested for interoperability. Some commercial products are available with DKIM signing and/or verification capability and some major domains are working on implementing the IETF-approved specification, although DKIM signatures are not yet being widely used.

DKIM (and its predecessor, DomainKeys) has received sufficient usage to demonstrate that it meets its goal of providing a signature that survives (maintains its validity) through the Internet mail system. This includes the use of "transparent forwarders" to allow recipients to use email addresses (such as college alumni association addresses) that are independent of their Internet service providers.

We have observed more than 20,000 domains producing messages with DKIM signatures, but we note that some of these are "disposable" domains which have been observed to send mail only for short periods of time. As with SPF, its use by domains of questionable reputation was expected (remember that neither DKIM nor SPF is meant to "identify" spam), and highlights the need for domain reputation and accreditation services. Some such services exist and have begun work on incorporating DKIM in their processes.

One area requiring further study is the use of DKIM signatures by mailing lists. Some mailing lists modify messages, by adding information relating to the mailing list, for example, in a manner that invalidates the message signature (such as prepending the mailing-list name to the subject). Such mailing lists can and should sign the messages following modification, but there are no known mailing lists doing so at this time.

7 CONCLUSIONS

The ability to spoof the origin addresses of messages is a design characteristic of Internet mail that has legitimate as well as illegitimate uses. Systems that authenticate email messages must

therefore be flexible enough to accommodate legitimate uses of spoofing, such as by mailing lists.

DKIM is designed with these characteristics in mind. As with any message authentication system, it is not a "magic bullet" to solve spam and phishing, but provides useful information about the origin of messages to form a basis for the application of whitelists, reputation, and accreditation of senders' email addresses.

8 ACKNOWLEDGEMENTS

The authors thank the other members of the DKIM design team, who worked diligently to produce a solid specification. Of special note are Mike Thomas of Cisco Systems, and Mark Delany and Miles Libbey of Yahoo!, all co-authors of the original specifications that became DKIM; and Eric Allman of Sendmail and Jon Callas of PGP Corporation, who spent a great deal of time and effort as editors of the final specification.

9 REFERENCES

[1] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM)", Internet Engineering Task Force, RFC 4871, May, 2007.

[2] Allman, E., Delany, M., and J. Fenton, "DKIM Sender Signing Practices", Internet Draft, http://www.ietf.org/internet-drafts/draft-allman-dkim-ssp-02.txt (work in progress), August, 2006.

[3] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", Internet Engineering Task Force, RFC 2440, November, 1998.

[4] Crocker, D., Otis, D., and J. Levine, "Client SMTP Authorization (CSA)", Internet Draft, http://www.ietf.org/internet-drafts/draft-crocker-csv-csa-00.txt (work in progress, expired), October, 2005.

[5] Federal Information Processing Standards (FIPS) "Publication 180-2, Secure Hash Standard (SHS)", U.S. DoC/NIST, 1 August, 2002.

[6] Fenton, J. "Analysis of Threats Motivating DomainKeys Identified Mail (DKIM)", Internet Engineering Task Force, RFC 4686, September, 2006.

[7] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", Internet Engineering Task Force, RFC 3447, February, 2003.

[8] Klensin, J., editor "Simple Mail Transfer Protocol", Internet Engineering Task Force, RFC 2821, April, 2001.

[9] Leahy, P. Statement introducing the "Anti-Phishing Act of 2004" on the US Senate floor, http://leahy.senate.gov/press/200407/070904c.html, Congressional Record, July, 2004.

[10] Leiba, B. and N. Borenstein, "A Multifaceted Approach to Spam Reduction", Conference on Email and AntiSpam 2004, July, 2004.

[11] Lyon, J. and M. Wong, "Sender ID: Authenticating E-Mail", Internet Engineering Task Force, RFC 4406, April, 2006.

[12] Lyon, J. "Purported Responsible Address in E-Mail Messages", Internet Engineering Task Force, RFC 4407, April, 2006.

[13] Ramsdell, B., editor "S/MIME Version 3 Message Specification", Internet Engineering Task Force, RFC 2633, April, 1999.

[14] Resnick, P., editor "Internet Message Format", Internet Engineering Task Force, RFC 2822, April, 2001.

[15] Rosencrance, L. "Trusted Electronic Communications Forum aims to fight online fraud", http://www.computerworld.com/securitytopics/security/story/0,10801,93871,00.html, Computerworld, June, 2004.

[16] Wong, M. and W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-MAIL, version 1", Internet Engineering Task Force, RFC 4408, April, 2006.