RC23995 (W0606-148) June 30, 2006
Computer Science

IBM Research Report

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

Barry Leiba
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Jim Fenton
Cisco Systems, Inc.
San Jose, CA

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

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

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

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

Abstract

Email protocols were not designed to provide protection against falsification of a message's address of origin, referred to 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. Using the associated DKIM sender signing policy 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 policies 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 (Klensin, 2001), in the mail "headers" (Resnick, 2001), 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 (Rosencrance, 2004; Leahy, 2004).

Clearly, something must be done to curtail spoofing; the ability to send messages while purporting to be another sender is by itself 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; see Wong and Schlitt, 2005), and Sender ID (Lyon and Wong, 2005), related techniques that differ in some details. CSV (Certified Sender Validation; see Crocker, Otis, and Levine, 2005) 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) (Leiba and Borenstein, 2004). 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 and I have an agreement that we always digitally sign our email to each other, then we can always be sure when one of us has sent the other legitimate mail, or when someone is trying to pretend to be one of us, sending mail to the other. There are signature techniques already standardized for applying signatures to email (Ramsdell, 1999; Callas et al, 1998), although the meaning of these signatures is subtly different from that of a DKIM signature.

There are a few problems, though, with using this scheme in general:

1. It assumes 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.
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 owner* 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, a "sender signing policy", allowing senders to communicate information about their practices to potential recipients.

The DKIM base specification (Allman, Callas, Delany, Libbey, Fenton, Thomas, 2005) tells signers how to create the signatures and include them in their messages, and tells verifiers how to interpret and verify the signatures, and what to do if they receive mail that is unsigned or that has a signature that fails verification. The DKIM signing policy specification (Allman, Delany, Fenton, 2005) tells senders how to specify their signing policies, and tells verifiers how to retrieve that information and use it. Taken together, the two specifications provide one method of defense against spoofing.

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.

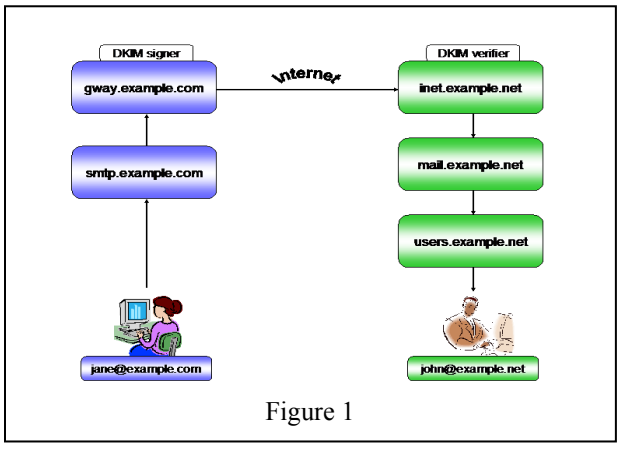

Figure 1

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*.

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 policies allows a sending domain to convey information to verifiers about how it chooses to sign the mail originating within it. This can give a domain the ability to defend its name against improper use, and to protect its reputation (see the discussion of signing policies, 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" (or blacklist, but the case for whitelisting is the more compelling one) 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. Note that signatures, by themselves, do not give verifiers any useful information about unsigned mail.

The addition of signing policies *does* provide such useful information, for sending domains that publish policies. 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 policies, 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 policies 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 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 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 (see Resnick, 2001), 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.

3.3 Selecting Headers

DKIM allows the signer to choose to sign some or all of the message headers. Since many of the headers do not contain information significant to the sender or recipient of the message, signers might choose not to sign them all. Headers are the parts of the message that are most vulnerable to change in transit, so leaving insignificant headers 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 all end-user mail programs (*mail user agents*, or MUAs) 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 be signed, and so DKIM requires the inclusion of *From* in the list of signed headers. It also requires the inclusion of any applicable "role" headers, such as *Sender* or *Resent-From*, where a header is applicable if it defines a role that the signing domain played in the transmission of the message.

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 (FIPS, 2002). 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 with the canonical message body. Before the 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; see Jonsson and Kaliski, 2003), 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; c=simple;
    d=example.com; s=appliances;  i=@store.example.com;
    t=1117574938; x=1118006938; h=from:to:subject:date;
    b=dzdVyOfAKCdLXdJOc9G2q8LoXSlEniSbav+yu
      U4zGeeruD00lszZVoG4ZHRNiYzR
```

Figure 2

4 DKIM Signing Policies

Sender signing policies 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 the issues in question.

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

1. Some messages from this entity are *not* signed. This explicitly conveys the same information as does the lack of a signing policy.
2. All messages from this entity *are* signed. Signatures created by third parties (mailing lists, etc.) are acceptable.
3. All messages from this entity *are* signed, and signatures created by third parties should *not* be accepted.
4. This domain uses individual signing policies. Please do another signing-policy query for the individual address.

Signing policies can be defined separately for subdomains, with the parent domain's policy 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 policy 3, while the latter might use policy 2, or even policy 1. 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 work in practice, what heed will be paid to the policies, 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 Policy

If there is *no* valid signature, or if the signed identity does not match the address in the message's *From* header, the verifier must check the signing policy of the

domain in the *From* address. The verifier retrieves the policy through a DNS query of TXT records for the pseudo-domain "_policy._domainkey.*domain*", where "*domain*" is obtained from the *From* address (there are more rules here; see the signing policy specification (Allman, Delany, and Fenton, 2005) for the details).

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 policy says – 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 policies 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 approximately 6 independent implementations of DKIM which have been tested for interoperability. Some commercial products are available with DKIM signing and/or verification capability, 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.

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.

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.

References

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

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

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

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

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

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.

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.

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.

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

Lyon, J. and M. Wong, "Sender ID: Authenticating E-Mail", Internet Draft, http://www.ietf.org/internet-drafts/draft-lyon-senderid-core-01.txt (work in progress), May, 2005.

Lyon, J. "Purported Responsible Address in E-Mail Messages", Internet Draft, http://www.ietf.org/internet-drafts/draft-lyon-senderid-pra-01.txt (work in progress), May, 2005.

Wong, M. and W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-MAIL, version 1", Internet Draft, http://www.ietf.org/internet-drafts/draft-schlitt-spf-classic-02.txt (work in progress), June, 2005.

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), October, 2005.

Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM)", Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-dkim-base-00.txt (work in progress), February, 2006.

Allman, E., Delany, M., and J. Fenton, "DKIM Sender Signing Policy", Internet Draft, http://www.ietf.org/internet-drafts/draft-allman-dkim-ssp-01.txt (work in progress), October, 2005.

Fenton, J. "Analysis of Threats Motivating DomainKeys Identified Mail (DKIM)", Internet Draft, http://www.ietf.org/internet-drafts/draft- ietf-dkim-threats-01.txt (work in progress), March, 2006.