Title: DKIM by Daniel Black

Abstract

Domain Keys Identified Mail (DKIM) is an email anti-spoofing protocol using digital signatures. It provides senders with a way to prove email from their domain has not been tampered or forged, and provides receivers a mechanism to validate email content without prior arrangement. Author Domain Signing Practices (ADSP) supports the lack of prior arrangement by indicating, based on the email *From* header field, whether a sender signs all email. Difficulties can arise because of email lists as they sometimes modify email content, and preserve *From* header fields, hence invalidating signatures and ADSP verification. However this paper suggests solutions for mailing list developers and their administrators and work-arounds for the receiver handling invalid signatures on email lists.

Introduction

Spoofed emails cost organisations time and money, whether the email socially engineers an employee to install malware, transfer money elsewhere, or just the time to read it. An organisation can also suffer costs repairing relationships and reputation with clients that got socially engineered with the organisation's email address.

Detecting spoofed emails can be hard. It requires specialised knowledge around SMTP[1] and Internet Message Format[2] to even have a chance of determining if an email is spoofed. DKIM is a protocol, compatible with current email, which allows recipients to automatic validate spoofed emails from participating domains.

The benefits to an organisation of deploying DKIM signing are:

- spoofed email can be filtered or marked with warnings without staff effort or training; and
- email is signed in a way that can be validated using clear forensic techniques if needed.

Under the constraints of retrofitting an email integrity mechanism into a 1982 standard still under extensive use, the IETF DKIM working group[3] set out to define a rugged signing and validation protocol that remained compatible with modern usage. Thus, the aim of DKIM is to allow signing domains to claim responsibility for the use of a given email address. It does this by putting a digital signature on an email as follows:

1. a RSA key pair is generated and the DKIM signing software is configured to use the private key;
2. the RSA public key is put in DNS in the form described by the proposed DKIM standard[4];
3. emails sent through the email server are DKIM signed by adding an additional email header containing the DKIM signature;
4. the sender's email server sends the email to the recipient's email server;

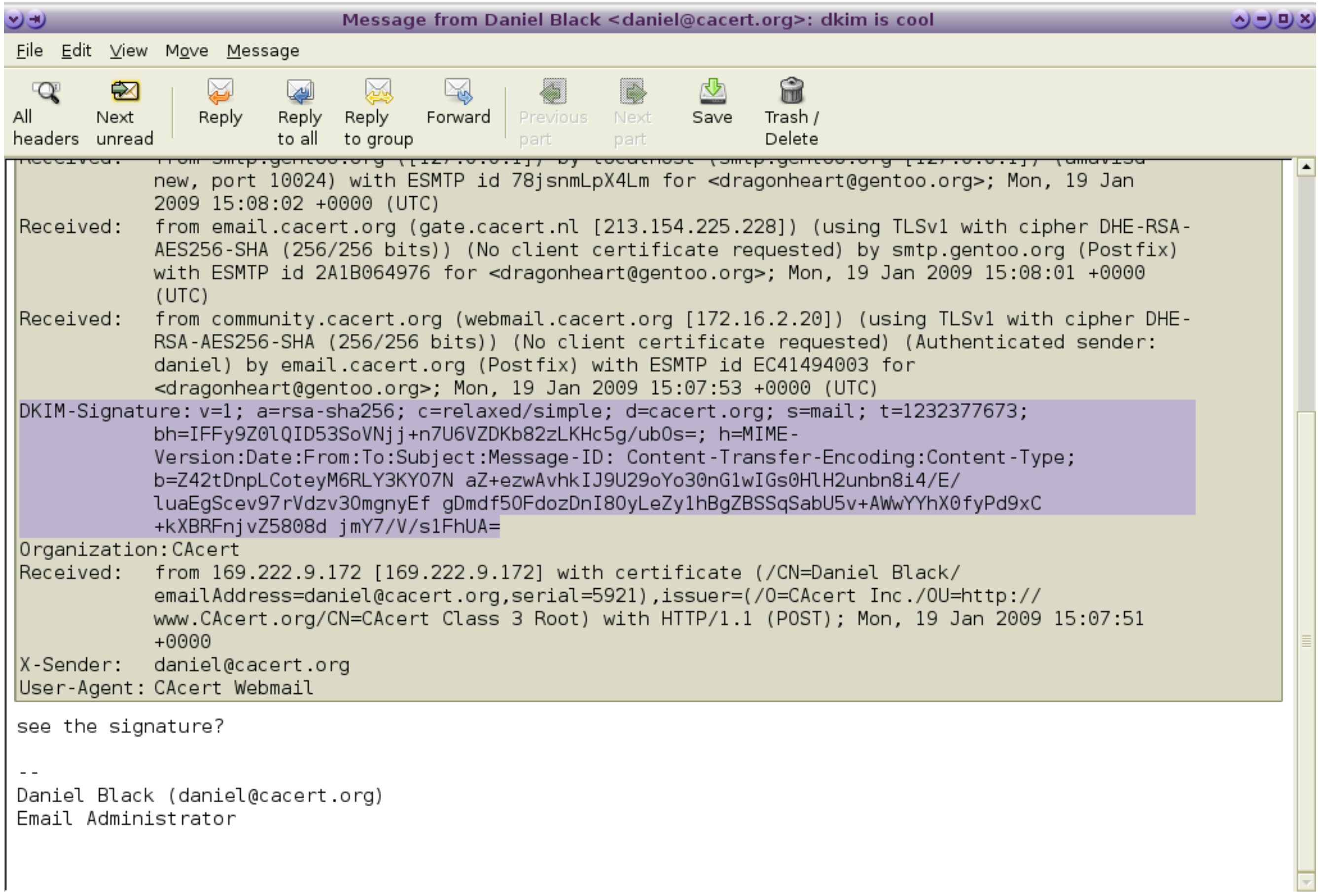
1 http://tools.ietf.org/html/rfc5321 Simple Mail Transfer Protocol
2 http://tools.ietf.org/html/rfc5322 Internet Message Format
3 http://tools.ietf.org/wg/dkim/ DKIM Status Pages
4 http://tools.ietf.org/html/rfc4871 DKIM Signatures

5. the recipient email server, having DKIM verification software installed, sees the DKIM signature and validates it by retrieving the public key from DNS;
6. If a DKIM signature is missing or invalid, the ADSP[5] DNS record (if present), may indicate that a DKIM signature should be there and what action the sender domain recommends; and
7. The email server makes an email filtering policy decision based on the verification status.

DKIM Signatures

The DKIM design was guided by RFC 4686[6]. The working group wanted to develop an automated verification mechanism for email centred around domain verification. S/MIME and PGP where not used as they where signing chain based, did not protect email headers and came with no key distribution mechanism.

DKIM signatures are RSA signatures that have been base64 encoded. The signature covers two hashes, one for the headers of the email and another for the body. This means sending the same body to many recipients while changing only the *To:* header, for example, does not require computing a new hash over the body (the larger part) for each message. The body hash is computed once. For a newsletter distribution, this can be a huge savings in compute costs.



```
              new, port 10024) with ESMTP id 78jsnmLpX4Lm for <dragonheart@gentoo.org>; Mon, 19 Jan
              2009 15:08:02 +0000 (UTC)
Received:     from email.cacert.org (gate.cacert.nl [213.154.225.228]) (using TLSv1 with cipher DHE-RSA-
              AES256-SHA (256/256 bits)) (No client certificate requested) by smtp.gentoo.org (Postfix)
              with ESMTP id 2A1B064976 for <dragonheart@gentoo.org>; Mon, 19 Jan 2009 15:08:01 +0000
              (UTC)
Received:     from community.cacert.org (webmail.cacert.org [172.16.2.20]) (using TLSv1 with cipher DHE-
              RSA-AES256-SHA (256/256 bits)) (No client certificate requested) (Authenticated sender:
              daniel) by email.cacert.org (Postfix) with ESMTP id EC41494003 for
              <dragonheart@gentoo.org>; Mon, 19 Jan 2009 15:07:53 +0000 (UTC)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=cacert.org; s=mail; t=1232377673;
              bh=IFFy9Z0lQID53SoVNjj+n7U6VZDKb82zLKHc5g/ub0s=; h=MIME-
              Version:Date:From:To:Subject:Message-ID: Content-Transfer-Encoding:Content-Type;
              b=Z42tDnpLCoteyM6RLY3KY07N aZ+ezwAvhkIJ9U29oYo30nG1wIGs0HlH2unbn8i4/E/
              luaEgScev97rVdzv30mgnyEf gDmdf50FdozDnI80yLeZy1hBgZBSSqSabU5v+AWwYYhX0fyPd9xC
              +kXBRFnjvZ5808d jmY7/V/s1FhUA=
Organization: CAcert
Received:     from 169.222.9.172 [169.222.9.172] with certificate (/CN=Daniel Black/
              emailAddress=daniel@cacert.org,serial=5921),issuer=(/O=CAcert Inc./OU=http://
              www.CAcert.org/CN=CAcert Class 3 Root) with HTTP/1.1 (POST); Mon, 19 Jan 2009 15:07:51
              +0000
X-Sender:     daniel@cacert.org
User-Agent: CAcert Webmail

see the signature?

--
Daniel Black (daniel@cacert.org)
Email Administrator
```

Illustration 1: DKIM Signature

Not all email headers are signed. Trace headers get added by email servers so are generally not covered

[5] http://tools.ietf.org/html/rfc5617 DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP)

[6] http://tools.ietf.org/html/rfc4686 Analysis of Threats Motivating DKIM

by the the header signature. As indicated in the DKIM-Signature in the above illustration, the DKIM signer selects which email headers it signs and in which order they are passed to the signing algorithm. This is important as it allows signatures to validate even if the signed headers are reordered in transit.

DKIM Signing

DKIM signing emails is very low risk. The process of signing adds an email header field as it goes out that will be passed but ignored by MTAs and mail user agents that are not DKIM-aware.

The summary of steps to deploy DKIM signing is as follows:

1. Select a product[7] that works with your border MTA and install it;
2. generate a DKIM key pair and publish the public key;
3. ensure emails are being signed correctly using a email verification reflectors[8];
4. ensure all emails 'From:' your domain are being signed. Consider remote users, marketing, web services with email functionality like notifications and password resets;
5. Optionally deploy an Author Domain Signing Practices DNS record (RFC 5617).

Deploying ADSP is not totally without risk as described later in Handling Email Lists. Deploying a ADSP record of *dkim=discard* can be performed on high value domains. An ADSP record of *dkim=all* should be ok with email lists however there may be some MTAs performing DKIM verification that over-zealously filter broken signatures.

The signer should sign a minimal set of headers[9], including well known headers, that are important and visible to the user[10].

Canonicalisation is an optional normalisation in the specification to account for mail transport agents (MTA) modification and may be specified as *relaxed* for the header and body individually if some tolerance for reformatting in transit is desirable. As headers are more likely to be altered and body signature schemes, like PGP and S/MIME, have discouraged alterations of email bodies, the recommend canonicalisation for DKIM header/body is *relaxed/simple*[11]. The header/body canonicalisation is specified in the *c=* tag in the signature below (Text 1: DKIM Signature).

DKIM Validation

DKIM validation is also a simple process. The difficult bit is the policy of what to do with the result (this is covered in the next section DKIM Filtering policy).

7 http://www.dkim.org/deploy/index.html DKIM-Soft-Services
8 http://testing.dkim.org/reflector.html Reflectors (DKIM verification email addresses)
9 http://tools.ietf.org/html/rfc5585#section-4.1 DKIM Service Overview - Basic Signing
10 http://tools.ietf.org/html/rfc4871#section-5.4 DKIM Signatures - Determine the Header Fields to Sign
11 http://sourceforge.net/mailarchive/forum.php?thread_name=alpine.LNX.1.10.0811032322010.5401%40starfish.lotspeich.org&forum_name=dkim-milter-discuss Header folding and verification (and following thread), Erik Lotspeich

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=cacert.org; s=mail;
          t=1254323235; bh=cnmRL6/g3jTIdqWLy+nYQuO3/7fgBm3kFUkMUdbA8eU=;
          h=Message-ID:Date:From:MIME-Version:To:Subject:References:
          In-Reply-To:Content-Type; b=y0AAOU0cpS2YyoNmoVi9vFeerVr7zIm4iG2Fpm
          KIGBWrpUCuqB7pHFLEM5MiRPNFH9PJqcnYACfIYsyLuk7+MS6kdtq7NbYzNB9yE+1ib
          BibOXhmggqHmuHHdQMbE+N4C+MuauPIV05BlrdneZ4SZjPayo7hOR7uHqAQKH4iLPw=
```

Text 1: DKIM Signature

Using the sample DKIM signature above, the selector is *mail* is indicated by *s*= and the domain is *cacert.org* as indicated by *d*=. The public key for this signature is stored in DNS as a TXT record type at the location *{selector}*._domainkey.*{domain}*. The DKIM key for *mail._domainkey.cacert.org* is '*v=DKIM1;g=*;k=rsa;p=MIGfMA0GCSq...IDAQAB*'. The *v*= indicates the version, the *g=** indicates this key is valid for all emails in the domain *cacert.org*, *k*= is the key type and *p*= is the public key.

The other important tags of a DKIM-Signature are (from RFC5871 section 3.5):

- *bh*= the hash of the body part of the message in base64
- *b*= the signature based on the key and computed on the header hash that includes the DKIM-Signature header field with the exception of the *b*= tag (hence including the *bh*= tag)

The process of validating this DKIM-Signature is described in RFC4871 section 3.7.

To assist with the policy the validation process may also fetch the Author Domain Signing Practice (ADSP) DNS entry associated with the author domain specified as _adsp._domainkey.*{authordomain}* where *authordomain* is the domain in the email *From* header field.

The DKIM validation product should provide an *Authentication-Results* email header field[12] to use with DKIM filtering, and optionally an inbuilt policy handling processing.

DKIM Filtering policy

The result from the DKIM validation is a set of results that you can use for your email filtering policy. These are listed in RFC5451 section 2.4.1[13]. Likewise the DKIM-ADSP also generates a set of results[14] as do other verification schemes[15].

To make sure you are assessing your policy and not the *Authentication-Results* header field put there by an email spoofer you should remove or reject any *Authentication-Results* for your domain at the border MTA[16]. Below is a postfix rule used to detect and reject forged results headers[17], i.e. those that claim to have been added by an internal or trusted MTA.

Here **.example.com* is what is used as the *authserv-id* (RFC5451 2.3) in the DKIM validation product.

12 http://tools.ietf.org/html/rfc545 Message Header Field for Indicating Message Authentication
13 http://tools.ietf.org/html/rfc5451#section-2.4.1 Message Header Field for Indicating Message Authentication Status - DKIM and DomainKeys Results
14 http://tools.ietf.org/html/rfc5617#section-5.4 DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP) -Authentication-Results Result Registry Update
15 http://www.iana.org/assignments/email-auth/email-auth.xhtml Email Authentication Parameters
16 http://tools.ietf.org/html/rfc5451#section-1.6 Trust Environment
17 This is not fully compatibility with the CFWS part of the formal Authenticated-Results specification (RFC 5451)

```
/^Authentication-Results: .*\.example\.com[^.a-z0-9-]/ REJECT spoofing
          Authentication-Results from my domain is just rude
```

Text 2: Postfix header_check rules to prevent spoofed Authentication-Results headers

At first you could assume a email filtering policy is easy and make a rule like:

```
if dkim=fail or dkim-adsp=fail or dkim-asdp=discard then reject (or drop)
```

Text 3: Strict DKIM Policy

If you do this, and ADSP of *all* or *discardable*, you will eventually come across the case that one user inside your domain subscribes to an email list that keeps the author's *From* address, invalidates the signature by subject or body modification. In this case the email is sent correctly by the your MTA, is received correctly by the email list software, has its signature invalidated by the list software (potentially removing the signature), and gets received with one of the fail conditions above. Your end user may not appreciate this.

Handling Email Lists

As a general email receiver you can handle an invalid signature in a number of ways including:

1. The email is rejected or discarded (recommended for ADSP *dkim=discardable*), or filed as spam;
2. Manage a whitelist of email lists (by envelope sender, IP, or DKIM signature domain of the list) that are exempt from the authors' DKIM ADSP policy;
3. If the email list does DKIM validation, use the *Authentication-Results* header field created by the email list software in combination with a whitelist (#1 above).
4. Use the ADSP failure as criteria to modify the message with a "forgery warning" that the end user will notice;
5. Use the ADSP failure to increase a spam score;
6. Deploy an email client add-on that shows the results of the trusted[18] Authentication-Results headers in a user friendly way.

The most desirable policy of an email list from the receiver's point of view is to ensure the list software does not break the author domain signatures. Method #2 of validating on DKIM signature of the list is a close second. Communicating the strategies below to email list managers and developers is an important step in raising their awareness. This will hopefully give you a better product and service.

Running and Developing Email List Software

If you run an email list and want to be DKIM friendly there are a few things you should be done. The first is to act like a good receiver. To do this you should DKIM validate emails and apply a DKIM policy to reject those with invalid signatures and missing signatures where an ADSP policy has *dkim=all* or *dkim=discardable*. As email lists are rarely chained together there is a possibility that

18 http://tools.ietf.org/html/rfc5451#section-7.2 Misleading Results

dropping emails based on DKIM validation will have no negative consequences for you. If you want to accept the possibility then you have the options described in Handling Email Lists to consider.

As an intermediary email list not the end validation stage, adding an *Authentication-Results* header field is recommended and provides a way for the receiving domain to deploy validation method #3.

In addition to performing verification, an email list should also act like an email sender. It needs however to act like a special sender as it is not the author domain. The end receiver of the email could attempt an ADSP verification of the email hence preserving the validity of the Author Domain Signature is important. There are two ways of doing this:

1. disable every option that modifies the email message as these can break the DKIM signature; or
2. Rewrite the *From* header field to contain the mailing list's address rather than that of the original author. Ensure the *Reply-to* or *Sender* header fields match the sender for mail client compatibility. This may introduce other incompatibilities so use with caution.

If configuring the mail list software in this way is not possible it is recommended you reject emails with a ADSP policy of *dkim=discardable*[19].

You should also add a DKIM signature if you validated the *DKIM-Signatures* sent to the list server.

Future

Reading so far you may be under the impression that all that is required to deploy DKIM is get some system administrators into action. For DKIM signing and verification this is entirely true.

For DKIM email policy handling there is still scope for a little work to make products easier to work with. Implementing easy to manage whitelists to handle email list signature breaking is needed for medium to large scale email gateways.

Email list software needs to have one or both of the options presented in Running and Developing Email List Software implemented. The email list manager Sympa[20] is well on the way to adding new features and Mailman development has stalled[21].

Mail clients could use enhancements and plugins to express to the user the Authenticated-Results generated within the administrative domain (RFC5451). A small example of what can be done is shown with the DKIM verification status plugin for RoundCube[22]. There are outstanding requests for Mozilla[23] and Kmail[24] products, and Evolution and Claws also do not have this feature either.

If you are going be developing DKIM standard based applications there are email lists with very knowledgeable people to support you[25].

19 http://mipassoc.org/pipermail/dkim-dev/attachments/20090930/e767bd81/attachment.html [dkim-dev] dkim validation software and mail lists, Serge Aumont on Sympa design after discussion.
20 http://www.sympa.org/manual/dkim DKIM features for Sympa
21 http://wiki.list.org/display/DEV/DKIM DKIM Development (Mailman)
22 http://rcmplugins.wladik.net/ DKIM verification status plugin (for RoundCube)
23 https://bugzilla.mozilla.org/show_bug.cgi?id=265226 Bug 265226 Implement DomainKeys (DKIM/RFC 4871)
24 https://bugs.kde.org/show_bug.cgi?id=204712 Bug 204712 Mark DKIM signed messages
25 http://mipassoc.org/mailman/listinfo/dkim-dev dkim-dev -- DKIM Developer's Discussion List

Conclusion

The development of the standards suite associated with DKIM so far has been extraordinary. The development of DKIM products and interoperability testing has highlighted the importance of open standards and cooperative development in achieving a common goal.

Product developers have produced some great products to perform signing and verification. Small enhancements will allow for the customers to deal with current email list signature invalidation easily.

With a little development effort, and a lot of small deployment efforts, this long-standing class of email spoofing can be significantly reduced.

Thanks

- IETF DKIM working group – for working out all these standards
- Product Developers – for giving everyone the chance to defeat email spoofing
- Murray S. Kucherawy – for opendkim and reviewing this paper