A Review of Yahoo!'s DomainKeys Technology

David Purdue

Sun Microsystems

<davidp@sun.com>

ABSTRACT

Email should be one of the most powerful communication mediums at our disposal. However email abuse reduces the effectiveness of this medium for all users. Various technical means to combat abuses such as spam and phishing have been proposed, and in this paper one of those, DomainKeys, is examined.

DomainKeys provides a means to digitally sign all email that emanates from a given domain. Public keys for the digital signature algorithm are distributed via the Domain Name Systems (DNS), rather than trying to set up a Public Key Infrastructure (PKI). Email recipients can retrieve the public key from DNS to validate the signature on the incoming email.

The paper looks in detail at how DomainKeys achieves domain authentication for email, and then assesses how well it can meet the stated aim of improving the email user experience before comparing DomainKeys to other related systems.

I conclude that DomainKeys succeeds in providing an authentication framework that is useful in and of itself, but that the usefulness of this framework in combating spam is limited.

1. Introduction

Email should be one of the most powerful communication mediums at our disposal. Three characteristics that make it powerful are that it is cheap, it is fast, and it is very widely available.

Unfortunately these same characteristics open email to abuse - and two of the main forms of abuse are "spam" - bulk dissemination of unsolicited emails seeking your dosh - and "phishing attacks" - fraudulent attempts to obtain confidential information from you such as account numbers, credit card numbers and passwords. These abuses are reducing the effectiveness of email as a communication medium for all users.

Of course, the best way to combat spam is to pass laws making it illegal - this is very effective, according to the Australian Communications Authority[1]. 

Back in the real world, several technical mechanisms have been proposed to reduce or eliminate spam and phishing, and this paper examines the DomainKeys[1] technology that has been proposed by Yahoo![2]

The paper will start by giving an explanation of how the DomainKeys mechanism works. I will go on to give an assessment as to how effective the technology will be, and then compare it to other related technologies before drawing conclusions.

1. DomainKeys is a trademark of Yahoo!
2. Yahoo! is a registered trademark of Yahoo! Inc.

1.1 My Life As An Emailer

The stated aim of DomainKeys is to improve the user experience of email, and so my assessment of DomainKeys is sure to be clouded by how it will affect my own use of email. So as a bit of background, here is the kind of emailer I am:

I maintain three different mailboxes: one for work, one for AUUG and one for "home" (which encompasses everything else). Each mailbox lives on a different (mail receiving) server in a different administrative domain.

I have a number of different email addresses which I use as the "`From:`" address for different purposes - e.g. `david.purdue@auug.org.au` if writing about AUUG business, `davidp@sun.com` if writing on work business.

There are many different ways I connect to the Internet - attached at work, attached at home, dial up when travelling, wireless hotspot, conference network, etc. How I am connected determines which outbound mail server I use to deliver my mail. All mail servers will deliver mail for all my "`From:`" addresses.

I subscribe to a number of mailing lists. In addition, my email address(es) are published for people to contact me. Hence I expect to receive

some emails from people I do not know, without these emails being considered "unsolicited".

About 50% of the email I receive is spam.

Finally, I am a customer of the Loadsawealth[3] Bank and use their Internet banking. They email me from time to time.

2. How DomainKeys Works - High Level

The DomainKeys technology aims to assure the recipient of a piece of email that it originated in the domain that it claims to be from. Figure 1 gives a high level overview of how it does this.

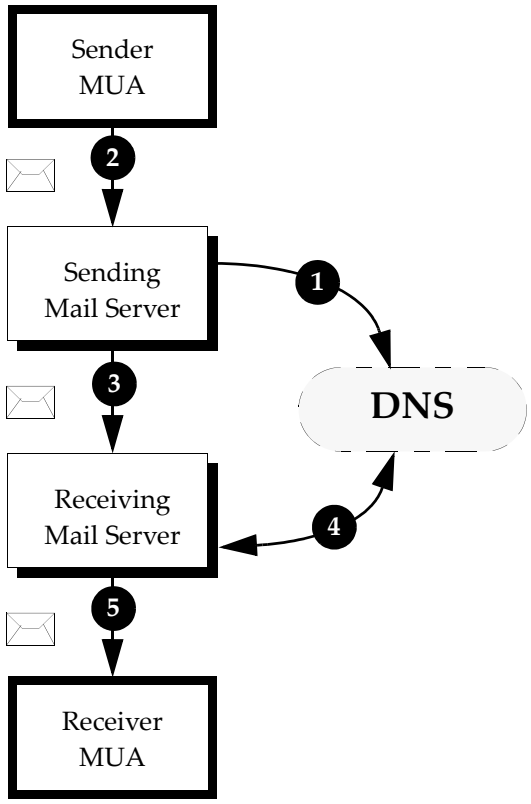

Figure 1: DomainKeys high level overview.

1. The Sending Mail Server for the domain generates a private/public key pair for a digital signature algorithm, and publishes the public key via the Domain Name System (DNS).
2. Someone within the domain writes an email, which their Mail User Agent (MUA) send to the Sending Mail Server (which acts as the "outbound MTA" for the domain in question).
3. The Sending Mail Server uses the private key generated earlier to create a digital signature of the message. This is embedded into the message (as an additional header). The email is then sent to its destination in the normal way.
4. On receiving the email, the Receiving Mail Server determines which domain the email claims to be from, and looks up that domain's public key. It uses that public key to verify the signature within the email headers.
5. If the signature is verified, then the message is made available for the recipient. If not, then local policy dictates what to do (forward with a warning, discard, etc.).

An overview of DomainKeys can be found at [10].

3. How DomainKeys Works - More Detail

The DomainKeys mechanism has been submitted as an Internet Draft[2]. This draft, while incomplete, gives more detail on how DomainKeys is to be implemented, and it is worth reviewing some of that detail to help in the evaluation of DomainKeys' effectiveness.

From the outset, DomainKeys is designed as an extensible protocol: at almost every level where there are multiple ways to achieve an outcome there is a way to plug alternate ways into the protocol. However, for the sake of interoperability, no alternative that is not mentioned in the standard should be used, and currently there is only one alternative given for at each juncture.

A couple of points to note: First, while there is nothing stopping DomainKeys being implemented within an MUA, the draft is written around implementing DomainKeys with boundary MTA's - i.e. the server that last handles outbound email within an organisation, and the first that handles email coming into an organisation. Second, while DomainKeys could be used to authenticate individual users, the focus is on authenticating all mail that comes from a given domain. We will return to these points later.

3.1 Digital Signature Algorithm

While there is scope to plug in any digital signature algorithm, the only one specified to date is RSA. The draft gives instructions for using OpenSSL to implement all the required RSA operations (generating keys, generating signatures, verifying signatures).

3.2 Embedding Information In DNS

DomainKeys uses DNS for the distribution of public keys. The keys (and related sender policy

3. Name changed to protect, well, me.

information) are embedded in TXT records associated with the domain in questions.

To avoid conflicts, all DomainKeys information is stored under the "`_domainkey.`" namespace. So, for example, all DomainKeys information for Loadsawealth Bank would be within the "`_domainkey.loadsawealth.com`" namespace.

DomainKeys allows for multiple concurrent public keys to be used by each domain. "Selectors" are used to indicate which public key an individual email is using. A selector appears in DNS as a subdomain in the `_domainkey.` namespace.

The interpretation of selectors is totally arbitrary and up to the sending domain's policy. Thus they could be used to provide separate public keys for each outbound mail relay, such as:

- `sydney._domainkey.loadsawealth.com`
- `melbourne._domainkey.loadsawealth.com`
- `brisbane._domainkey.loadsawealth.com`

or they could be used to regularly replace a domain's public key:

- `2004jun._domainkey.loadsawealth.com`
- `2004jul._domainkey.loadsawealth.com`
- `2004aug._domainkey.loadsawealth.com`

Note that there must be a selector even if the domain only uses a single public key.

The public key to use for signature verification is attached as a TXT record to the selector. Apart from the key itself, the record can contain information on "granularity" (i.e. which addresses - left of the '@' - within a domain the key can be used for, the default being all), key type (currently only RSA is defined), human readable notes, and a "testing" flag - to indicate that this key is used for testing only. If the public key data is blank, it indicates that the key has been revoked.

3.3 Which Domain To Sign For?

When a Sending Mail Server receives an email to sign, it has to determine which domain the email claims to be coming from.

To strictly comply with the draft, emails to be signed should have only one "`From:`" header. However, there is an implication that if multiple "`From:`" headers are present then the first one encountered should be used. The sending domain is the string to the right of the '@' in that email address.

In cases where a domain can not be extracted from the "`From:`" header, or if the extracted domain is not one that the Sending Mail Server controls (i.e. knows a private key for), then no DomainKey authentication will be done, and it becomes a local policy decision how to handle the message (pass it on, drop it, etc.).

3.4 Preparing The Message For Signing

To provide assurance that the message you receive not only came from who claimed to send it, but also is the message that was sent, the digital signature is generated over the whole message.

However, since just one bit of difference in the message can make a large difference to the digital signature, the sending and receiving mail servers must agree on which bytes of the message are to be included in calculating the signature. This agreement is the "canonicalisation".

DomainKeys allows for a range of canonicalisation algorithms, but currently only one is defined: "simple." This just takes the whole message (including any headers in place), any line terminator is replaced with CRLF and empty trailing lines are removed. Note that this operation only affects the string passed to the digital signature algorithm, the message delivered to the Receiving Mail Server is not changed.

3.5 Attaching The Signature

The signature is attached as a "`DomainKey-Signature:`" header that is placed immediately before the headers that were included in the signed text. The information included in this header is:

- The signature algorithm used (currently only RSA encryption of a SHA1 hash is specified).
- The domain the email is from.
- The selector - to identify which private key for the domain was used.
- Which canonicalisation algorithm was used (currently only "simple" is specified).
- The query method to retrieve the public key (currently only "dns" is specified).
- The actual signature data.

3.6 Sender Policy

The sending domain attaches a TXT record to the "`_domainkey.`" domain name to indicate to receivers how to handle mail that fails to verify - the "sender policy". Currently, a sender can only indicate that the whole domain is in testing mode, or that all messages from this domain are signed (and hence any message that can not be verified should be treated with suspicion).

3.7 Message Verification

On receipt of a DomainKeys email, the Receiving Mail Server must go through to answer the question, "Is this email all it claims to be?"

First the "`DomainKey-Signature:`" and "`From:`" headers must be extracted from the message. If there are multiple `DomainKey-Signature:` headers, then use the last. Use the first `From:` header. If the first `From:` header is before the last `DomainKey-Signature:` header, then treat the message as unverified.

If a domain can not be extracted from the `From:` header, or the domain does not match the one specified in the `DomainKey-Signature:` header, then treat the message as unverified.

Next the selector and domain values are used to extract the relevant public key from DNS. There are quite a few things that could go wrong here: the data may not exist in DNS, the data returned may not match the DomainKeys format, the key type (algorithm) specified in the data may not match the key type of the signature, or the key may have been revoked. In all these cases, treat the email as unverified.

Now the Receiving Server is ready to check the signature. The same canonicalisation procedure needs to be applied - only this time it is applied to everything below the `DomainKey-Signature:` header. This is then fed to the signature verification algorithm, which yields a yes or no answer.

If the signature fails to verify then the Receiving Server should go back to DNS to consult the Sender Policy. It could be that the Sender was just testing DomainKeys.

At this stage the Receiver's local policy should be applied. For example, the Receiver may elect to drop all email signed with a revoked key. Even if a signature is successfully verified, the Receiver may elect to drop all email from certain domains, e.g. `spammers-r-us.com`.

Finally, the email can be passed on (if not dropped) to the recipient's MUA. The Receiver needs to indicate the result of DomainKeys verification, and it does this by embedding a "`DomainKey-Status:`" header in the email. Possible values for this header are:

- **`good`** - a signature was verified.
- **`bad`** - the signature failed verification
- **`no key`** - the key for the signature was not found
- **`revoked`** - the key has been revoked.
- **`no signature`** - there was no DomainKeys signature in the headers.
- **`bad format`** - the signature contained unexpected data.
- **`non-participant`** - the Sender indicated it does not participate in DomainKeys.

4. Reflections on DomainKeys

4.1 Incomplete Standard

The DomainKeys draft has enough details to start developing implementations, but could not be considered a complete standard. Part of this is due to having to adopt ad-hoc approaches in areas where the author would rather have relied on other standards. These areas include:

- MTA to MUA communication - how does the MTA show the results of DomainKeys verification to the MUA? The draft opts for using an email header.
- Canonicalisation - email systems are notorious for changing the email that goes through them. At a minimum they add "`Received:`" headers to the front of a message, but they can also add headers at other points and change the values of headers. Thus, there will be much debate over the best canonicalisation scheme. The scheme proposed has simplicity - however I personally disagree with including all headers in the signature. I think that only those headers of relevance to most users (`Subject:`, `Date:`, `From:`, `To:`, `Cc:`) should be included. Other headers could be changed in transit and most users would not notice or care. Even this approach has problems, as some MTA's rewrite address headers, e.g. converting "`fred@eg.com (Fred Nurque)`" to "`Fred Nurque <fred@eg.com>`" - semantically equivalent but with totally different digital signatures. So canonicalisation of address and date formats would also be required.
- Granularity - DomainKeys provides a hook to specify that a given public key only applies to a subset of addresses within the domain. As an example, this would be good for giving road warriors their own private key (see discussion on road warriors below). However, as the draft stands you can only indicate that a key applies to all addresses in the domain - there is no standard way to indicate a subset.
- Public Keys in DNS - there is no standard way to store and retrieve public key data in DNS, so DomainKeys has fallen back on using TXT records.

- Sending Domain Policy - Currently there is no standard way for a domain to indicate sending policy to recipient domains, although the IETF MARID Working Group is developing one. The _domainkey TXT record used to indicate DomainKey sender policy should be considered an interim measure.

4.2 No PKI

DomainKeys does not rely on a Public Key Infrastructure. Each domain owner generates their own key pair and published the public key via DNS. There is no need for certificates.

This makes the system quicker and easier (and, looking at the prices certificate authorities charge, cheaper) to deploy.

For the nervous who just must have digital certificates, there is an extension within the draft for sending the public key with a message as an X.509 certificate.

4.3 DomainKeys & DNS

DomainKeys relies on the security of the DNS, in as much as when you ask DNS for a public key, you want to be sure that the answer you get back is the actual public key for that domain. Currently DNS does not make that assurance.

There is also a concern over the added load that DomainKeys with make on the DNS as a whole. Given that there are those who are already trying to stretch the DNS to its limits[5], this concern is not wholly unfounded.

4.4 Mailing Lists

One advantage of DomainKeys is that the signature remains valid even if the email passes through a number of relays before reaching its final destination. This makes it ideal for mailing lists or other systems that relay messages - as long as they do not alter the message of headers.

4.5 Mailing Houses

Another nice result of using selectors is that it is safe to contract send out email on your behalf.

Say, for example, Loadsawealth Bank needs to send out new Internet Banking Terms And Conditions to all their customers. They could contract a mailing house to do the emails. Loadsawealth would provide the addresses and a DomainKey private key, and put the public key in DNS with the "mailshot20040901" selector. The mailing house could sign the emails as coming from Loadsawealth Bank. AS soon as the mail shot is done and all emails delivered, the mailshot20040901 selector key would be revoked.

4.6 Private Key Management

The aim of DomainKeys is to provide assurance that a signed email really came from the domain it claims to come from. This assurance is only valid if the private keys are kept private. They must be protected in the same way that the private keys for SSL web servers are protected.

The "selector" mechanism that allows for multiple private keys within a domain helps here. It means that each individual private key can be used and stored in a minimal way. Should a private key be compromised it can be easily revoked.

4.7 Road Warriors

The Road Warrior could be sending email out with a variety of `From:` addresses and via a variety of outbound MTA's. The MTA may not have authority to sign the address being used, and due to port blockages, the road warrior may not be able to use the domain's "home" MTA as their outbound MTA. Thus DomainKeys leaves road warriors somewhat out in the cold.

Eventually it will probably come to pass that the road warrior's MUA gains the ability to attach a DomainKeys signature. However, this will lead to increased problems of private key management.

This should only be adopted once the problem of specifying granularity is sorted out - I would not want to see a private key that is valid for the whole domain installed on a laptop in the field.

4.8 The Open Relay Problem

Note that DomainKeys on its own does not solve the problem of open relays, and in some ways makes it worse. Administrators of Sending Mail Servers must take great care that they only sign and forward email that legitimately originates from inside their domain and is authorised to be sent. If they fail to do this, the outbound relay could be sending unauthorised emails with the assurance that they are, in fact, authorised.

4.9 Shouldn't We Just Use End-to-end Encryption?

Yes, ideally all email would be signed and encrypted from the author's MUA to the recipient's. We have the technology.

The problem with end-to-end encryption is the burden placed on users to make it work.

There are competing technologies to achieve it (PGP/GnuPG vs. S/MIME), as well as problems of key management, PKI, and user education to use these effectively.

DomainKeys does not prevent users from using end-to-end encryption. It complements that with a server-to-server mechanism that is transparent to users yet provides a measure of assurance regarding the origins of emails.

4.10 But… Will It Work?

DomainKeys aims to provide a way to authenticate email that is simple, cheap and sufficiently effective. It is certainly simple and cheap, but, assuming a wide deployment, will it be effective?

In terms of preventing spam, DomainKeys will add more data to my existing spam prevention system. So if spammers participate in DomainKeys, I can automatically reject mail from domains that allow spammers. If, as they do today, they try to hide their identity by claiming their email comes from a false or stolen address, then if the domain of the stolen address participates in DomainKeys I can reject the mail they send since it won't have a valid signature.

As a corollary, if spammers choose my address as their fraudulent `From:` and people complain to me about sending them spam, I can put my hand on my heart and say it was not me - check the signature.

However, there is nothing to force a domain to adopt DomainKeys. So, as I do not instantly reject email from people I do not know, I will still have to apply spam filtering to incoming email, and while more spam will get detected and rejected automatically, there is not sufficient disincentive to stop spammers altogether.

When phishing attacks are considered, however, DomainKeys can be seen to be much more effective. Most phishing attacks are forged to appear to come form a financial institution, and these are just the sort of organisation that would participate in DomainKeys and sign every single message they send. So an email from one of these institutions that either does not have a signature or has a bad signature can be safely (and automatically) ignored. This can be set up as a simple filter in my MUA.

5. Comparison To Related Systems

5.1 PGP Moose

The PGP Moose[7] was developed by Greg Rose to address a similar problem: forged `Approved:` headers on news articles in moderated USENET news groups.

Newsgroup moderators use PGP Moose to generate a digital signature for the article, which is embedded in the article as an additional header. The encryption mechanism used to generate the digital signature is, funnily enough, PGP.

A news server can check the validity of a signature in an article, and cancel the article automatically if it is not valid.

PGP Moose could very easily have been the inspiration for DomainKeys.

5.2 SPF

SPF, or the Sender Policy Framework[4], is a similar system to DomainKeys in that it uses information published in DNS to make assertions about the validity of email. Like DomainKeys, it is the subject of an Internet Draft[9].

To give a very brief summary, SPF publishes (in TXT or RR records) a list of machines that are allowed to send email on behalf of that domain. Then, in SMTP conversation, if mail claims to come from a certain domain (using the envelope address in the `MAIL FROM:` command), but the IP address of the sender is not listed as an authorised sender for that domain, then the mail is rejected.

SPF is getting a lot of press as the answer to spam[6], but I regard it as a weaker system than DomainKeys.

SPF lacks the flexibility of DomainKeys - it can not be used for mailing lists or other email forwarding systems. It also leaves road warriors totally out in the cold.

It has some of the same weaknesses as DomainKeys - currently there is no way to validate the information you get from DNS, you just have to trust it. In addition, there is an additional weakness, since someone trying really hard could forge the from address in the IP packets of the SMTP conversation.

SPF is championed by AOL, and Microsoft has come on board, merging their Caller ID proposal into SPF.

But fundamentally SPF does not use cryptography, DomainKeys does, and crypto is cool. 

6. Availability

Yahoo! is working with Sendmail, who are incorporating DomainKeys into both the free and commercial versions of Sendmail. Sendmail

has released an alpha version of this code as open source[8].

In addition, Yahoo! is developing a reference implementation that can be plugged in to other MTA's, which is also available as open source[3].

Yahoo! is keeping tight control on the intellectual property (IP) wrapped up in DomainKeys[10], but are granting free licenses to use any patent they hold that is needed to implement DomainKeys, and free use of the DomainKeys trademark to apply to any system that conforms to the specification in the Internet draft. They allow you to "make, use sell, offer for sale, import, or yodel implementations". However, there are some who disagree with any such controls over IP, which will stop them using DomainKeys implementations on ideological grounds.

7. Conclusions

DomainKeys, while immature, shows promise of easily adding extra authentication for the origin of emails.

While this will not eliminate spam and phishing attacks, it will make phishing much more difficult, and will provide extra information that will be useful to spam filters.

In this regard, it is of more use to email senders who wish to assure recipients of their *bona fides* than it is to general email recipients, who may well see large volume of spam anyway.

References

[1] **AAP**, "Anti-spam laws are working: ACA", *The Age*, 22 July 2004, `http://theage.com.au/articles/2004/07/22/1090464779848.html`

[2] **Mark Delany**, *Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys)*, May 2004, `http://www.ietf.org/internet-drafts/draft-delany-domainkeys-base-00.txt`

[3] **Mark Delany & Russell Nelson**, *DomainKeys Library for email servers and clients*, 29 July 2004, `http://domainkeys.sourceforge.net/`

[4] **IC Group Inc.**, *Sender Policy Framework*, 2004, `http://spf.pobox.com/`

[5] **Danny O'Brien**, "Tune in to the net's untapped power", *New Scientist*, 31 July 2004, p. 22

[6] **Riva Richmond**, "Heavyweights fight back against spam", *The Australian Financial Review*, no. 11352, 11 June 2004, p.67

[7] **Greg Rose**, *PGP Moose*, November 1998, `http://people.qualcomm.com/ggr/pgpmoose.html`

[8] **Sendmail Inc.**, *dk-milter: DomainKeys by Yahoo! plug-in for the sendmail MTA*, 8 August 2004, `http://sendmail.net/dk-milter/`

[9] **M. W. Wong & M. Lentczner**, *The SPF Record Format and Test Protocol*, 11 July 2004, `http://spf.pobox.com/draft-ietf-marid-protocol-00.txt`

[10] **Yahoo!** Inc., *DomainKeys: Proving and Protecting Email Sender Identity*, 2004, `http://antispam.yahoo.com/domainkeys`