DKIM Implementation – “How”

Murray S. Kucherawy
Principal Engineer, Cloudmark

June 8, 2009

Planning Your Deployment

- Selecting Key Rotation Policy
  - How long do your keys live?
  - Similar in nature to your password change policy
- Selecting Key Divisions
  - Department?
  - Mail campaign?
  - User?
  - Month?
- Things To Consider
  - Every new selector generated requires changing signer configuration and DNS
  - DNS changes may be complicated at your site

Planning Your Deployment

- Local Mail Routing Policy
  - May now have to funnel your outgoing traffic through a smaller set of MTAs (i.e. the ones that sign) than you're currently using
  - Copying keys is dangerous, so you'll want to minimize it
- Considerations about Roaming Users
  - Do they sign with their own machines, or route through yours?
    - Anything that can sign as your domain can impact your reputation. Do you trust your roaming users to maintain safe machines?
  - If they do their own signing, do you give them your main private key(s), or let them make their own?
    - See above about key copying
    - Could be another DNS headache

Planning Your Deployment

- Key Delegation
  - If you use a mass mail outsource company you might want to enable them to sign mail on your behalf
    - Create a new key pair and give them the private key for signing and publish the matching public key
    - Or you can accept and publish a public key they give you
  - Definitely do not have them use your existing keys!

Creating and Publishing Keys

- Creating a key pair requires two fairly simple OpenSSL commands
  - OpenSSL comes standard on most UNIX systems these days, but you can also get the latest from `http://www.openssl.org`
- You may have to upgrade to be fully DKIM compliant
  - Prior to v0.9.8 of OpenSSL the SHA256 hash function was not included, but DKIM requires it for signing

Creating and Publishing Keys

- First, generate the private key:
  - `openssl -genrsa -out` *`file bits`*
    - Generates a new RSA private key using the specified number of *bits* as the key size and writes it out to the specified *file*
  - Larger numbers of bits increase security by geometrically increasing the difficulty of cracking the key
    - Also result in slower processing as well as possible DNS transport issues
    - Common practice with DKIM is 1024-bit keys

Creating and Publishing Keys

- Next, using the private key, generate the public key:
  - `openssl rsa -in file1 -pubout -out file2 -outform PEM`
    - Generates a public key based on the private key found at *file1* and requests it in PEM format written to *file2*

Creating and Publishing Keys

- What a PEM format public key looks like:

```
-----BEGIN PUBLIC KEY----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDh2vbhJTijCs2qbyJcwRCa8WqD
TxI+PisFJofaPtoDJy0Qn41uNayCajfKADVcLqc87sXQS6GxfchPfzx7Vh9crYdx
RbN/o/URCuZsKmym1i1IPTwRLcXSnuKS0XDs1eRW2WQHGYlXksUDqSHWOS3ZO1W5
t/FLcZHpIll/80xs4QIDAQAB
-----END PUBLIC KEY-----
```

- This is a base64 encoding of the key with delimiters
- Now we need to stick this someplace where other verifying agents can retrieve it in order to verify our signed messages
- DKIM uses the DNS TXT records for this, so we need to turn the above into one of those

Creating and Publishing Keys

- DKIM requires a few more bits of information in the published key record:
  - What selector name do you want to use?
  - What kind of key is it?
  - Should verifiers be told that you're only testing?
  - Which of your users can use it?
  - What hash methods do you support for signing?
  - Some other stuff we'll skip for now

Creating and Publishing Keys

- Now build your TXT record
  - What kind of key is it? "`k=rsa`"
  - Should verifiers be told that you're only testing? "`t=y`"
  - Which of your users can use it? "`g=*`" or "`g=username`"
  - What hash methods do you support for signing? "`h=sha256`"
  - Separate them with semi-colons
    - And spaces if you wish

Creating and Publishing Keys

- Then append the public key
  - Take the PEM form
  - Remove the begin and end tags
  - Copy that base64 text as-is into the TXT record, preceded by "p="
- Do DNS record wrapping if desired
  - Break the record into palatable substrings
  - Wrap the set of substrings in parentheses

Creating and Publishing Keys

- So start with this:

```
-----BEGIN PUBLIC KEY----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDh2vbhJTijCs2qbyJcwRCa8WqD
TxI+PisFJofaPtoDJy0Qn41uNayCajfKADVcLqc87sXQS6GxfchPfzx7Vh9crYdx
RbN/o/URCuZsKmym1i1IPTwRLcXSnuKS0XDs1eRW2WQHGYlXksUDqSHWOS3ZO1W5
t/FLcZHpIll/80xs4QIDAQAB
-----END PUBLIC KEY-----
```

- …and end with this:

```
selector._domainkey   IN      TXT     ( “k=rsa; t=y; g=*; ”
    “p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDh2vbhJTijCs2qbyJcwRCa8WqD”
    “TxI+PisFJofaPtoDJy0Qn41uNayCajfKADVcLqc87sXQS6GxfchPfzx7Vh9crYdx”
    “RbN/o/URCuZsKmym1i1IPTwRLcXSnuKS0XDs1eRW2WQHGYlXksUDqSHWOS3ZO1W5”
    “t/FLcZHpIll/80xs4QIDAQAB” )
```

- Post that in your DNS, reload, and go!

Creating and Publishing Keys

- Tools to make this easy
  - Taken from Sendmail's open source package called *dkim-milter*
  - `dkim-genkey` generates a key pair, outputs a DNS TXT record containing the public key and a PEM file containing the private key
    - Doesn't do the line breaking for you so it's all on one line
    - Works fine, just not as pretty as it could be
  - Command line flags let you change selector name, number of bits, granularity, etc.

Creating and Publishing Keys

- Other DNS considerations
  - Good idea to set the TTL low during testing and rollout
    - In case you need to change something
    - Increases number of queries because it decreases caching
  - Make `_domainkey` a subdomain?
    - Can then delegate it to the mail admins without giving up control of the whole zone
    - Depends on your IT infrastructure

Creating and Publishing Keys

- Testing your installation
  - Need to make sure your private key (with which you will sign) and public key (with which others will verify) agree
  - `dkim-testkey` will read your private key and get your public key from DNS and then see if they are associated
  - Any other output means verifiers will have difficulty
    - Maybe DNS hasn't distributed its updates yet?

Creating and Publishing Keys

- Testing your installation
  - Can also do this manually
  - Retrieve your public key from DNS, write it to a file
  - Edit it to remove TXT record tags, so just the key remains
  - Extract your public key from the private key as before with the `openssl` command
  - Use `diff` to see if they match

Creating and Publishing Signing Policy

- Author Domain Signing Practices
  - Not part of base DKIM and not yet standard
  - Protocol for declaring that a particular sending domain signs all of its own mail
- Select a signing policy for verifiers to consider
  - No policy (mail may or may not be signed)
  - Sign all (expect mail from this domain to be signed)
  - Discard (toss mail that doesn't have a valid signature)

Creating and Publishing Signing Policy

- Post this in your DNS at a specific location
- For example:

```
_adsp._domainkey  IN   TXT  “dkim=all”
```

- Essentially a software version of the well-known signing agreement between eBay/PayPal and Yahoo!

Configuring to Sign Mail

- Generally you have the following steps:
  - Install your signing agent (may require an MTA upgrade)
  - Tell it which mail to sign
    - Which SMTP clients, which users/domains
    - Only signs for `localhost` by default
  - Tell it where the keys are and which ones to use
    - For which domains and users
  - Select signing options
  - Go!

Configuring to Sign Mail

- Key selection
  - One key pair (*selector*) could be used for everything, or you can use different selectors for different purposes
  - Need to consider the implications of grouping versus dividing
    - Grouping is simple to manage
    - Dividing makes it easier to distinguish among domains
    - Important in reputation, for example
  - Third-party signatures are controversial

Configuring to Sign Mail

- Consider signing options
  - Do body length limits?
    - Helps with mail sent to lists
    - Has security implications
  - List absent header fields?
    - Prevents them from being added without breaking verification
  - Never sign certain header fields?
    - Allows them to be changed, removed or reordered enroute without affecting verification

Configuring to Sign Mail

- Consider signing options
  - Set signature expirations?
    - Signature will no longer validate after a specific time has passed
  - Which canonicalizations to use?
    - “relaxed” tolerates minor rewrites such as spacing changes, while “simple” is maximum strictness
  - Include forensic data?
    - Allows a verifier to see if header fields changed in transit, preventing verification

Configuring to Sign Mail

- Steps specific to *dkim-milter*
  - Install the filter
  - Select a rendezvous socket
    - Filter will listen for connections from MTAs at the designated socket
    - Security considerations
  - Put private keys someplace safe
    - Filter needs read access to them, but nobody else does
  - Make a list of which keys are used for which users/domains

Configuring to Sign Mail

- Steps specific to *dkim-milter*
  - Write a configuration file
    - Signing options
    - Domain and key selection rules
    - Auto-restart
    - What socket to use
    - What SMTP clients should have mail signed
  - Start the filter
  - Configure the MTA to connect to the filter and restart it

Configuring to Sign Mail

- Sample `dkim-filter.conf` contents for signing

```
AlwaysSignHeaders       Subject
AutoRestart             True
Background              True
Canonicalization        simple/simple
Diagnostics             Yes
KeyFile                 /var/db/dkim/sign200905.key.pem
InternalHosts           /etc/mail/dkim/internal
LogWhy                  true
Selector                sign200905
SignatureAlgorithm      rsa-sha256
Socket                  inet:8891@localhost
Syslog                  Yes
```

Configuring to Verify Mail

- Generally you have the following steps:
  - Install your verifying agent (may be an MTA upgrade)
    - Might be the same as the signing agent
  - Tell it which mail to verify
    - Which SMTP clients, which users/domains
    - Might just be “everyone”
  - Select verifying policy options
  - Throw the switch!

Configuring to Verify Mail

- Verification policy options
  - DKIM specifies that an unsigned message and one with a bad signature should be treated the same
  - Any other verification choices are specific to the implementation you use, not to DKIM itself
  - Some common ones are discussed here

Configuring to Verify Mail

- Verification policy options
  - Require certain headers be signed even if absent
    - A favourite is Subject:, since MUAs generally display it
    - Modification or addition both invalidate signatures
  - Require a minimum of additional text when messages are signed with "`l=`"
    - Prevents replay attacks with undesirable appended text

Configuring to Verify Mail

- Verification policy options
  - Do something with "z=" (forensics) header fields?
    - Can't do anything other than figure out why a verification failed if it was caused by a header change
  - Authentication-Results: header fields
    - What *authserv-id* to use internally?

Configuring to Verify Mail

- Verification policy options
  - Apply ADSP?
    - Signers might want you to discard mail that's not signed or lacks a valid signature
    - You could end up rejecting/quarantining mail that was accidentally damaged
    - Not a standard yet, still under revision
  - How much clock drift on signatures is allowed?
    - To tolerate misconfigured clocks out there

Configuring to Verify Mail

- Sample `dkim-filter.conf` contents

```
ClockDrift              300
DiagnosticDirectory     /var/db/dkim/DIAGNOSTICS
DNSTimeout              10
InternalHosts           /etc/mail/dkim/internal
LogWhy                  true
Socket                  inet:8891@localhost
ADSPDiscard             Yes
Syslog                  Yes
Statistics              /var/db/dkim/dkim-stats.db
```

Testing Your Setup

- Once configured for signing, send a test message to an autoresponder
  - Check `http://www.dkim.org` for a current list
- Autoresponder will try to validate your message and send the results back to you
- The reply will also be signed, so your verifier can take a crack at it
- Of course, if you run two disjoint sites, you can do this yourself

Beyond Basic DKIM

- RFC5451 defines a header field called Authentication-Results that can be used to tell MUAs and other filters what the results of DKIM were
  - Can also be used for SPF, Sender-ID, etc.
- There are some security considerations around using this
  - In particular, dealing with spoofs from outside
  - Read the spec, even if you plan to do this some other way!

Beyond Basic DKIM

- Domain reputation
  - OK, so `example.com`'s signature verified. So what?
    - Spammers can sign their mail just like you can
  - An MUA or filter that considers a verified signature to be ultimate approval is being foolish
    - What if one were to register `marri0tt.com` and send signed phishes? Would the average user be fooled?

Beyond Basic DKIM

- Domain reputation
  - Reputation seeks to associate value with a name
  - Both commercial and open source efforts are in development

Beyond Basic DKIM

- Reporting
  - Many sites wish to be advised of unusual activity
    - DKIM failures might mean phishing or unexpected problems in transit
  - Draft proposal to extend DKIM and ADSP to announce requests for such advisories
    - Can request reports of incidents such as unsigned messages or failed validations
    - Can request SMTP rejections with specific text, or can request ARF reports

Beyond Basic DKIM

- Doing it on your own
  - The *dkim-milter* package includes a C library called *libdkim* that can be used to build your own DKIM-aware applications
  - Includes full HTML documentation

Who's Doing It Now

- Service providers
  - AOL (verifying)
  - Yahoo! (verifying)
  - Gmail (signing and verifying)
- Popular web sites
  - LinkedIn, Facebook
  - eBay, FTD
- Vendors
  - …just about everyone
  - Several open source implementations

What's Up At MAAWG

- Might want to check out some of the other panels at this conference
  - Authentication update
    - Discusses recent and upcoming standards activity
  - Reputation document review
    - MAAWG is working on a Best Current Practices document

References

- General Information: `http://www.dkim.org`
- DKIM is defined in RFC4871 (standards track)
  `http://www.ietf.org/rfc/rfc4871.txt`
- ADSP is currently an IETF DKIM WG draft
  `http://www.ietf.org/ID.html`, draft-ietf-dkim-ssp
- Authentication-Results is defined in RFC5451 (standards track)
  `http://www.ietf.org/rfc/rfc5451.txt`
- dkim-milter
  `http://sourceforge.net/projects/dkim-milter`
- DKIM reporting is currently an IETF individual submission draft
  `http://www.ietf.org/ID.html`
  draft-kucherawy-dkim-reporting
- ARF is currently an IETF individual submission draft
  `http://www.ietf.org/ID.html`
  draft-shafranovich-feedback-report

Questions & Answers

- Now's the time!