Messaging Anti-Abuse Working Group

DKIM Implementation

MAAWG Training Series

Segment 3 of 4 on DomainKeys Identified Mail

From the onsite training course at the MAAWG 18th General Meeting
San Francisco, February 2010

Messaging Anti-Abuse Working Group

DKIM Implementation – Video Segments

Segment 1 20 mins.	Segment 2 20 mins.	Segment 3 18 mins.	Segment 4 35 mins.
Theory	Theory	Practical	Practical
• General DKIM Architecture • What DKIM Is and Isn't	• DKIM Protocol Details • Separate Mail Streams & Signing Practices	• Planning • Keys and Policies	• Signing Software • Verifying Software • Testing, Other Topics • Q&A

Messaging Anti-Abuse Working Group

Segment 3 Covers

Practical:

- Planning
- Keys and Policies

Murray S. Kucherawy
Principal Engineer
Cloudmark
msk@cloudmark.com

Messaging Anti-Abuse Working Group

DKIM Implementation – "How"

Murray S. Kucherawy
Principal Engineer, Cloudmark <msk@cloudmark.com>

February 15, 2010

Planning Your Deployment

- Get in the right mindset
  - Consider the mail from your domain as a flow or stream
  - Then consider how receivers will evaluate or classify your mail
  - Do you really want it all to be one unified stream?

Planning Your Deployment

- Get in the right mindset
  - Given your mail from `user@host.domain`, receivers will probably focus on the `host.domain`
    - `user@host.domain` is way too much data to track; spammers randomize the `user`
    - Determining `domain` is actually difficult; the "top" domain might have one, two or even three labels (`.com` vs `.co.uk` vs. `.toronto.on.ca`)

Planning Your Deployment

- Get in the right mindset
  - Now think about the idea of *reputation*
    - A measure of value or desirability associated with your mail stream based on past messages
  - Do you want all your mail grouped under one reputation, or is it beneficial to allow them to earn separate reputations?
    - For example, should a mail campaign from your sales/marketing group be able to impact the reputation of your transactional mail?

Planning Your Deployment

- Get in the right mindset
  - In general, best practice is to make a separate subdomain for each major mail stream coming from your domain
    - So if `marketing.example.com` sends a batch of mail that makes the world mad and start filtering, `orders.example.com` won't suffer

Planning Your Deployment

- Once you have your subdomains chosen, it's time to think about planning out your keys
- Keys are specific to domains, so the more subdomains you have, the more keys you need
- For security reasons, you might want to change your keys once in a while
  - Just like you change passwords once in a while (right?)

Planning Your Deployment

- Selecting Key Rotation Policy
  - How long do your keys live?
  - Similar in nature to your password change policy
- Selecting Key Divisions (*selectors*)
  - Department?
  - Mail campaign?
  - User?
  - Month or Year?
- Things To Consider
  - Every new selector generated requires changing signer configuration and DNS
  - May require some overlap
  - 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

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

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

- 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

- By the way, what are private and public keys?
  - A pair of associated “keys” (involving some very large prime numbers) forming a “pair”
    - Use one to encrypt, the other to decrypt
    - Give one out (public) and keep one (private)
    - Something encrypted by the private key can be decrypted by anyone that can get the public key, thus he/she can be sure it was encrypted by the private key holder
    - Something encrypted by the public key can only be decrypted by the private key

Creating and Publishing Keys

- And while we're at it, what is signing and verifying?
  - To sign, compute a *hash* of some data
    - Produces a large, unique sequence of bits (hash) representing that data
  - Encrypt the hash with a private key
    - Much cheaper than encrypting the whole message, and privacy is not a requirement
  - At the receiver, re-do the hash, then decrypt the signature with the public key
    - If the output (original hash) matches the second hash, we say the signature verified

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?
  - 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`"
  - 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: the OpenDKIM open source package
  - `opendkim-genkey` generates a key pair, outputs a DNS TXT record containing the public key (for nameserver) and a PEM file containing the private key (for signing filter)
    - 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?
    - DNS people 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
  - `opendkim-testkey` will read your private key and get your public key from DNS and then see if they are associated
  - Any 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
  - Proposed standard (RFC5617)
  - 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 have a valid signature)
  - 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!

Creating and Publishing Signing Policy

- Be careful with “all” and “discardable”
  - Remember, they mean “Expect our mail to arrive with a valid author domain signature”
  - How can you be sure all your mail will get through without being modified?
  - Some mail may be discarded or redirected because of changes outside of your control