Messaging Anti-Abuse Working Group

DKIM Implementation

MAAWG Training Series
Complete Slide Set for All Segments-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

MAAWG

Segment 1 Covers

Theory:

- General DKIM Architecture
- What DKIM Is and Isn't

Dave Crocker
MAAWG Senior Advisor
Principal, Brandenburg InternetWorking
dcrocker@bbiw.net

Messaging Anti-Abuse Working Group

DKIM Implementation – "*What*"

Dave Crocker

Brandenburg InternetWorking

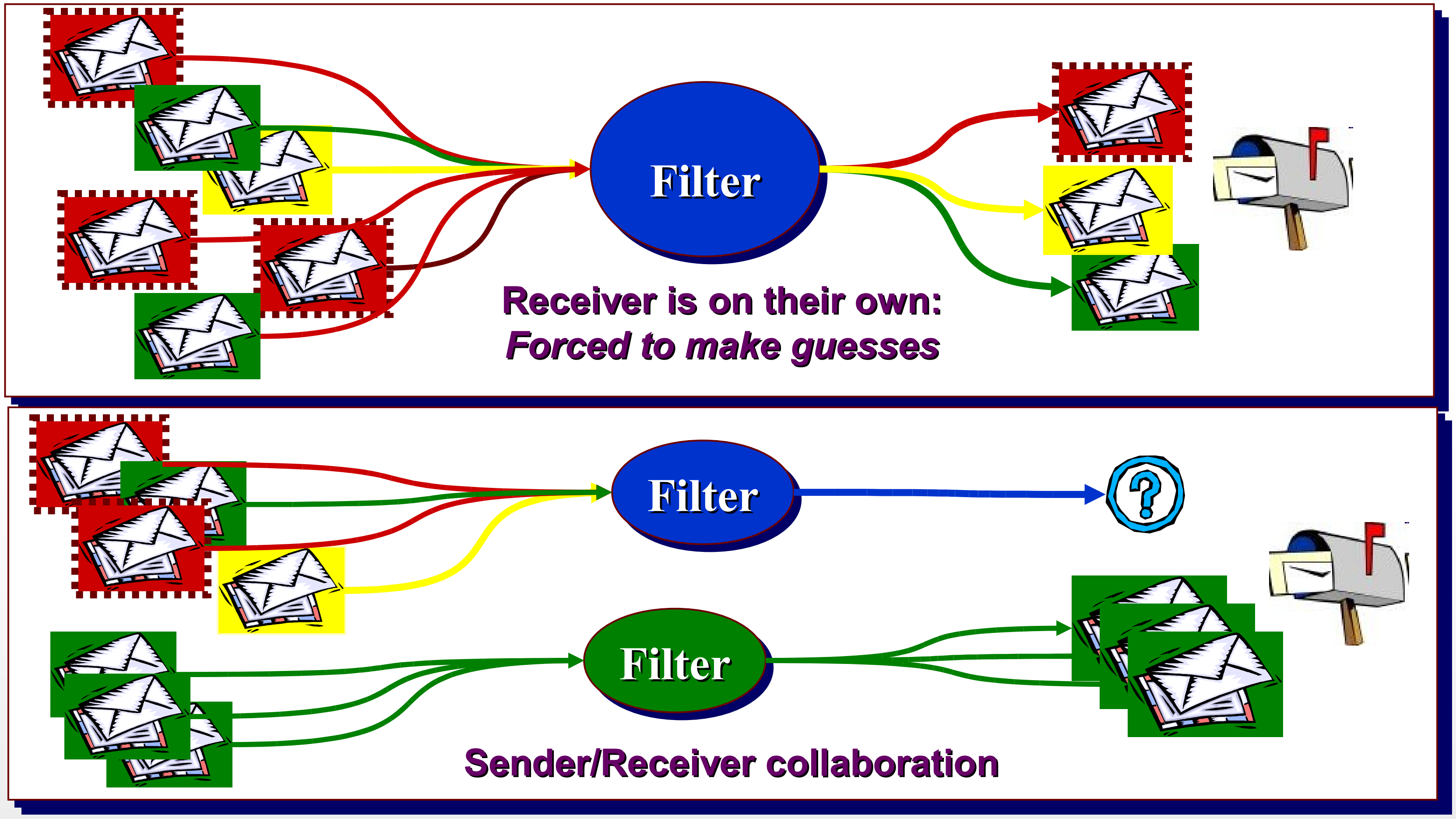
and

Senior Technical Advisor, MAAWG

Agenda –
What: DKIM in Trusted Email

- Trust vs. Mistrust
- What is DKIM and What is it for?
- DKIM Service Architecture
- Signature Basics
- ADSP
- Reporting Basics

Mistrust vs. Trust



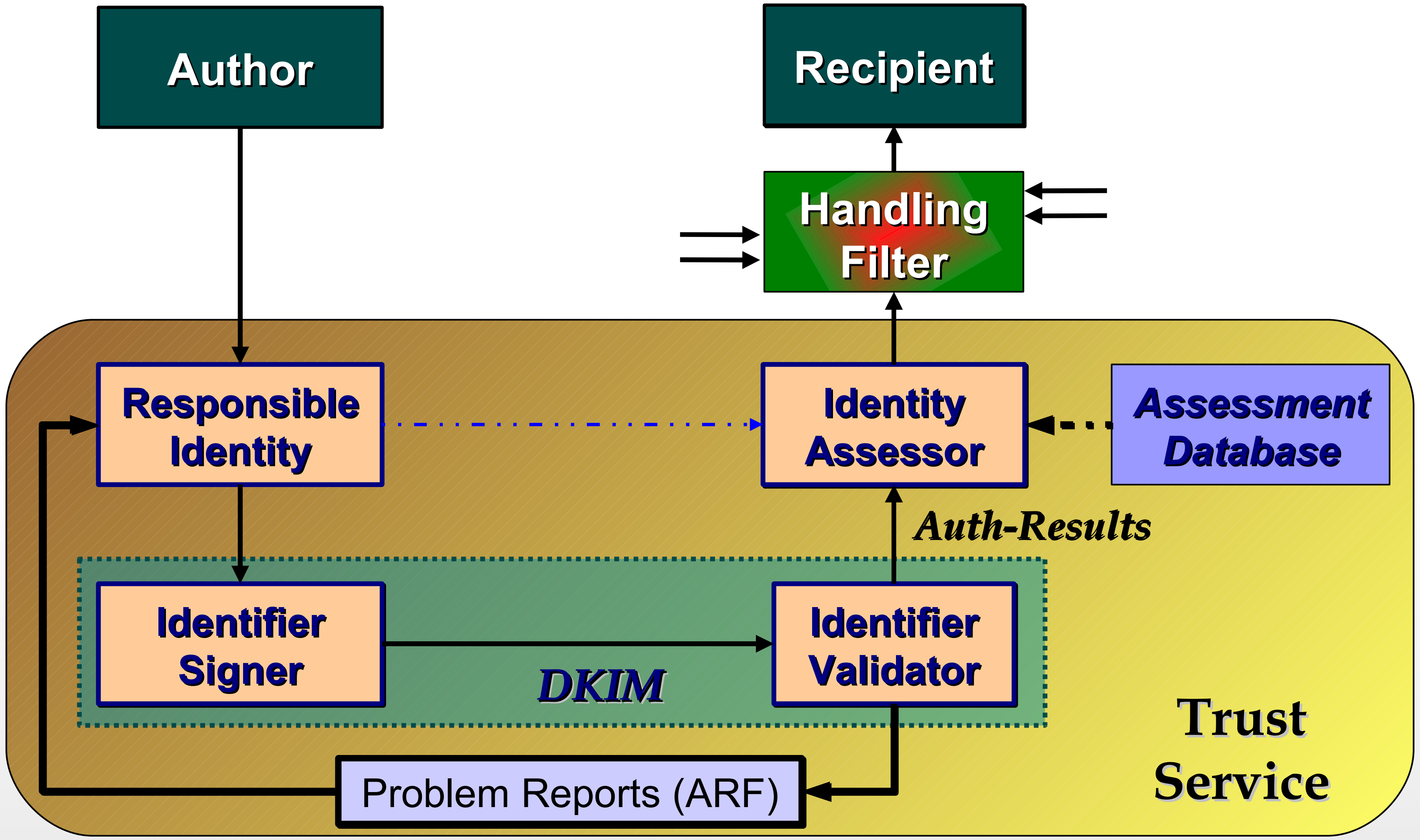
Differential Handling, with Trust as a Component

MAAWG

Organizational Trust

Stream Risk		**Low**	**Medium**	**High**
	Low	**BENIGN**: *Moderate filter*	**DILIGENT**: *Mild filter*	**PRISTINE**: *Accept*
	Medium	**UNKNOWN**: *Strong filter*	**TYPICAL**: *Targeted filter*	**PROTECTED**: *Accept & Contact*
	High	**MALICIOUS**: *Block & Counter*	**NEGLIGENT**: *Block*	**COMPROMISED**: *Block & Contact*

Trust Service Architecture

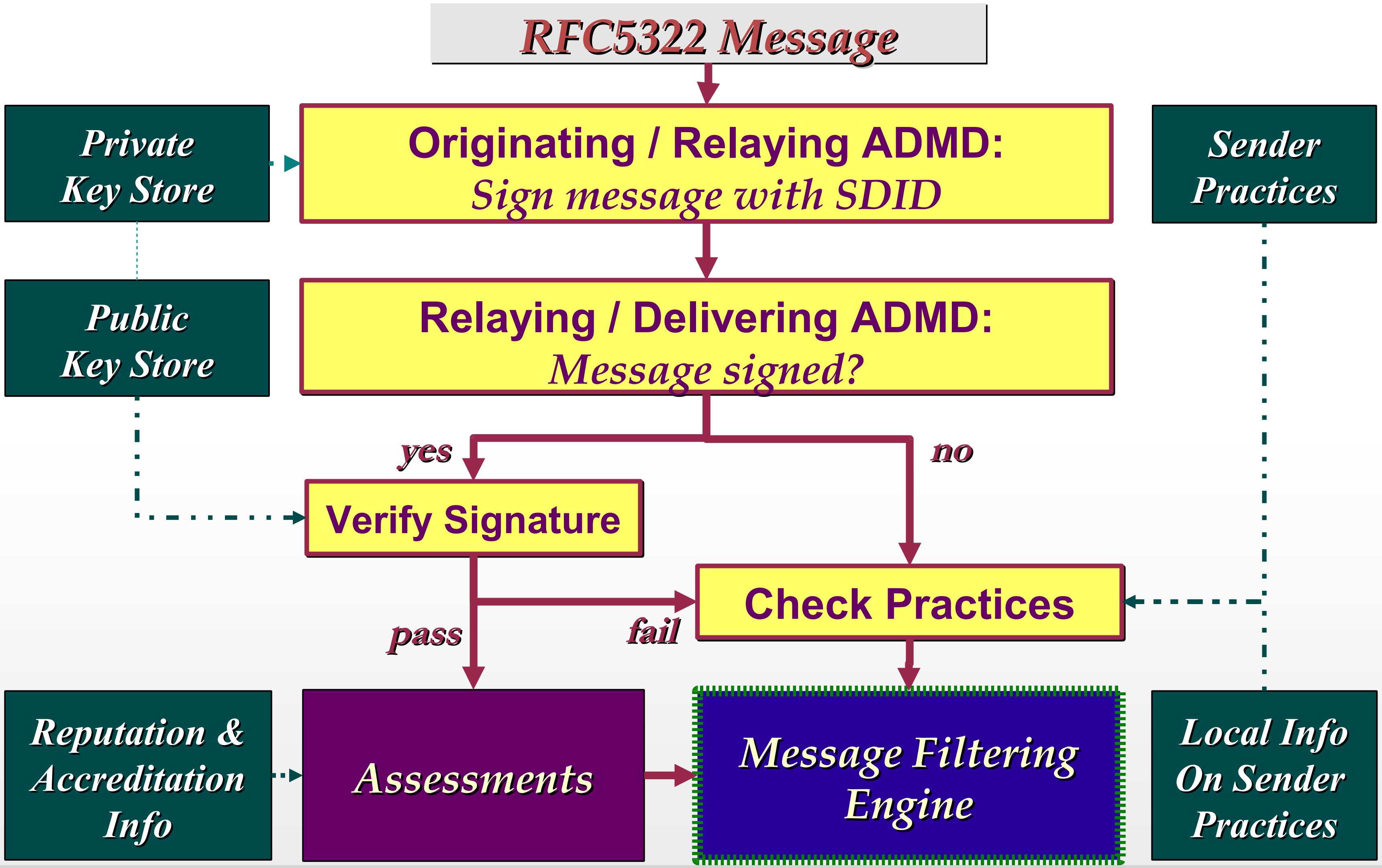


What is DKIM for?

- **Means a message is not spam**
- **Guarantees delivery**
- **Puts a domain name on a message**
- **Validates a message**
- **Authenticates the author or origin of a message**
- **Authenticates the sender of a message**

- **What DKIM *really* does**
  - Allows an organization to claim responsibility for transmitting a message, in a way that can be validated by a recipient.
  - The organization can be the author's, the originating sending site, an intermediary, or one of their agents.
  - A message can contain multiple signatures, from the same or different organizations involved with the message.

DKIM Service Architecture



RFC5322 Message
Private Key Store
Originating / Relaying ADMD: Sign message with SDID
Sender Practices
Public Key Store
Relaying / Delivering ADMD: Message signed?
yes
no
Verify Signature
Check Practices
pass
fail
Reputation & Accreditation Info
Assessments
Message Filtering Engine
Local Info On Sender Practices

Messaging Anti-Abuse Working Group

DKIM Implementation

MAAWG Training Series

Segment 2 of 4 on DomainKeys Identified Mail

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

MAAWG

Messaging Anti-Abuse Working Group

DKIM Implementation – Video Segments

Segment 1
20 mins.

Theory

- General DKIM Architecture
- What DKIM Is and Isn’t

Segment 2
20 mins.

Theory

- DKIM Protocol Details
- Separate Mail Streams & Signing Practices

Segment 3
18 mins.

Practical

- Planning
- Keys and Policies

Segment 4
35 mins.

Practical

- Signing Software
- Verifying Software
- Testing, Other Topics
- Q&A

Segment 2 Covers

Theory:

- DKIM Protocol Details
- Separate Mail Streams and Signing Practices

Dave Crocker
MAAWG Senior Advisor
Principal, Brandenburg InternetWorking
dcrocker@bbiw.net

Public Key – DNS Record

- **Query name combined from**
  - Selector *(for key rotation, s=)*
  - "._domainkey."
  - Signing Domain Identifier (SDID, d=)
- **Stored in TXT**
- **Major parameters**
  - v: Version of the DKIM key record
  - p: Public key data
  - n: Human readable notes

Signing and Verifying

Signing

- **Choose**
  - Private/public key
  - Signing Domain ID (SDID)
  - Selector
  - Header fields to sign
- **Compute hash**
- **Encrypt hash**
- **Create DKIM-Signature: field**

Verifying

- **Compute hash**
  - Note fields listed in DKIM-Signature field h= tag
- **Fetch public key**
  - From s=, d= field tags
- **Decrypt hash**
- **Compare**

DKIM-Signature: header field

- **Primary tags**
  - **a: The algorithm used to generate the signature**
  - **b: The signature itself**
  - **bh: The hash of the canonicalised body**
  - **c: Message canonicalization**
  - **d: The signing domain**
  - **h: List of header fields that are signed**
  - **s: The selector**
- **Additional tags**
  - **t: Signature timestamp**
  - **v: Version**
  - **i: Additional information about the identity of the user or agent for which this message was signed**

Identifying Mail Streams

- **An organization has multiple “types” of mail**
  - Corporate
  - Transactions (purchase order, order confirmation...)
  - Proposals
  - Marketing mass mailings
  - Customer Support
- **Label them with different DKIM d= subdomains**
- **Allow different reputations to develop**

ADSP:
Author Domain Signing Practices

- **Exploring mistrust**
  - What if no signature based on From: field domain?
- **Domain owner can publish practices for signing with From: field domain**
- **DNS TXT record under**
  - _adsp._domainkey.<from domain>
- **Practices:**
  - unknown, all, discardable

Status

- **Signing**
  - Proposed Standard
  - Updated
  - Minor -bis effort just starting
- **Deployment & Operations doc**
  - Going through final approval
- **ADSP**
  - Published.
- **Pending**
  - Assessment standards that use DKIM???

References

- **DKIM home page – http://dkim.org**
  - DKIM 3-slide Teaser
  - DKIM Service Overview – RFC 5585
  - FAQ
  - Wikipedia entry on DKIM
  - Development, Deployment and Operations
  - Three myths about DKIM
  - Examples and analysis, countering the myth that DKIM is expensive
  - Discussion Lists

- **DKIM Signatures – RFC 4871 + RFC 5672**
- **ADSP – RFC 5617**
- **Auth-Results – RFC 5451**
- **ARF – http://mipassoc.org/arf/ http://www.ietf.org/dyn/wg/charter/marf-charter.html**

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

MAAWG

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

MAAWG

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 <u>really</u> 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/FLcZHpIl1/80xs4QIDAQAB
-----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

DKIM Implementation

MAAWG Training Series

Segment 4 of 4 on DomainKeys Identified Mail

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

MAAWG

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

Segment 4 Covers

Practical:

- Signing Software
- Verifying Software
- Testing and Other
- Q&A

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

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” implies 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 *opendkim*
  - 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 *opendkim*
  - 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 `opendkim.conf` contents for signing all of a single domain with one key

```
AlwaysSignHeaders        Subject
AutoRestart              True
Background               True
Canonicalization         relaxed/simple
Diagnostics              Yes
Domain                   example.com
KeyFile                  /var/db/dkim/sign201002.key.pem
InternalHosts            /etc/mail/dkim/internal
LogWhy                   true
Mode                     sign
Selector                 sign201002
SignatureAlgorithm       rsa-sha256
Socket                   inet:8891@localhost
Syslog                   Yes
```

Configuring to Sign Mail

- Sample file contents for signing multiple domains (v2.0.0 or later)
- `/etc/mail/opendkim.conf`:

```
KeyTable                  /etc/mail/dkim/keytable
InternalHosts             /etc/mail/dkim/internal
SigningTable              /etc/mail/dkim/signingtable
Socket                    inet:8891@localhost
```

- `/etc/mail/dkim/keytable`

```
opskey  ops.example.com:ops:/etc/mail/dkim/keys/ops
mktgkey mktg.example.com:mktg:/etc/mail/dkim/keys/mktg
execkey exec.example.com:exec:/etc/mail/dkim/keys/exec
preskey exec.example.com:pres:/etc/mail/dkim/keys/president
defkey  example.com:default:/etc/mail/dkim/keys/default
```

- `/etc/mail/dkim/signingtable`

```
ops.example.com            opskey
mktg.example.com           mktgkey
president@exec.example.com preskey
exec.example.com           execkey
.example.com               defkey
```

Configuring to Sign Mail

- A note about OpenDKIM
  - All of the lookup tables referenced in `opendkim.conf` can be:
    - Comma-separated lists
    - Flat files
    - Files matching by regular expressions
    - Sleepycat databases (hash/btree)
    - LDAP directory lookups
    - SQL queries

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 <u>if</u> 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
  - How much clock drift on signatures is allowed?
    - To tolerate misconfigured clocks out there

Configuring to Verify Mail

- Verification policy options
  - Do you want to trust third-party signatures?
    - Again, this is still controversial
    - By default, OpenDKIM only uses author signatures when making final decisions, but you can tell it there are other domains you trust

Configuring to Verify Mail

- Sample `opendkim.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
```

More Complex Policy Options

- Experience has shown that there is a very wide variety of site-specific needs with respect to mail flow and policy enforcement
- Adding features to configuration files to keep up with changing environments is an uphill battle
- As a result, OpenDKIM now (as of v2.0.0) has hooks that allow one to write scripts to enforce policy

More Complex Policy Options

- *Lua* is the scripting language chosen
  - http://www.lua.org
  - Reference books available
- Three scripting entry points
  - *Setup*: Observe properties of message, decide whether to sign (and which key(s) to use), or verify, or both
  - *Screen*: For signed messages, examine the signatures and decide which ones to use and which to discard
  - *Final*: For signed messages, examine the results of processing each signature and decide what to do with the message
- `opendkim` exports message information and a bunch of access and utility functions to the Lua interpreter, then runs your script

More Complex Policy Options

- Sample trivial setup script:

```
-- See if {auth_author} was set
author = odkim.get_mtasymbol(ctx, "{auth_author}")

-- If it's not from an internal source or
-- authenticated, just verify it
if odkim.internal_ip(tx) == 0 and author == nil then
     odkim.verify(ctx)
     return nil
end

-- Since we got this far, we're signing; make a
-- signing request using the key "defkey" from the
-- KeyTable
odkim.sign(ctx, "defkey")

-- That's it!
return nil
```

More Complex Policy Options

- Sample trivial screen script:

```
--  retrieve the count of signatures on the message
nsigs = odkim.get_sigcount(ctx)
if nsigs == nil then
        return nil
end

--  get the From: domain
fdomain = odkim.get_fromdomain(ctx)
if fdomain == nil then
        return nil
end

--  for each signature, ignore it if it's not from the sender's domain
for n = 1, nsigs do
     sig = odkim.get_signhandle(ctx, n)
     sdomain = odkim.sig_getdomain(sig)
     if fdomain ~= sdomain then
         odkim.sig_ignore(sig)
     end
end
```

More Complex Policy Options

- Sample final script

```
 --  retrieve the count of signatures on the message
nsigs = odkim.get_sigcount(ctx)
if nsigs == nil then
        return nil
end

--  If the message had too much stuff added to it (more than 120 bytes)
--  then bounce it
for n = 1, nsigs do
     sig = odkim.get_sighandle(ctx, n)
     bodylen = odkim.sig_bodylength(sig)
     canonlen = odkim.sig_canonlength(sig)
     if bodylen > canonlen + 120 then
         odkim.set_reply(ctx, "554", "",
                         "Too much data after DKIM-protected body")
         odkim.set_result(ctx, SMFIS_REJECT)
     end
end

--  That's it!
return nil
```

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 it and 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
  - In the DKIM world, we would use the domain name found in "`d=`", i.e. the domain that took responsibility for sending the message
  - Likely more value in finding good guys and letting them in than in finding bad guys and keeping them out
    - Bad reputations are very easily shed

Beyond Basic DKIM

- Domain reputation
  - Both commercial and open source efforts are in development
    - OpenDKIM has hooks for one of the open ones, which is still experimental

Beyond Basic DKIM

- Reporting
  - Many sites wish to be advised of unusual activity
    - DKIM failures might indicate phishing or unexpected problems in transit
  - Draft proposal to extend DKIM, ADSP and ARF to publish 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 *opendkim* package includes a C library called *libopendkim* that can be used to build your own DKIM-aware applications
  - Includes full HTML documentation

Who’s Doing It Now

MAAWG

- 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
  - DKIM and Signing Practices
    - Discusses current DKIM and ADSP-related topics

References

- General Information: `http://www.dkim.org`
- DKIM is defined in RFC4871 (standards track)
  `http://www.ietf.org/rfc/rfc4871.txt`
- Author Domain Signing Practices
  `http://www.ietf.org/rfc/rfc5617.txt`
- Authentication-Results is defined in RFC5451 (standards track)
  `http://www.ietf.org/rfc/rfc5451.txt`
- OpenDKIM
  `http://www.opendkim.org`
- DKIM reporting is currently an IETF individual submission draft
  `http://www.ietf.org/ID.html`
  draft-kucherawy-dkim-reporting
- ARF is now being advanced by the MARF working group
  `http://www.ietf.org/dyn/wg/charter/marf-charter.html`

Questions & Answers

- Now's the time!