Internet Engineering Task Force (IETF)
Request for Comments: 5863
Category: Informational
ISSN: 2070-1721

T. Hansen
AT&T Laboratories
E. Siegel
Consultant
P. Hallam-Baker
Default Deny Security, Inc.
D. Crocker
Brandenburg InternetWorking

May 2010

DomainKeys Identified Mail (DKIM) Development, Deployment, and Operations

Abstract

DomainKeys Identified Mail (DKIM) 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 defines a domain-level digital signature authentication framework for email, using public key cryptography and using the domain name service as its key server technology. This permits verification of a responsible organization, as well as the integrity of the message content. DKIM will also provide a mechanism that permits potential email signers to publish information about their email signing practices; this will permit email receivers to make additional assessments about messages. DKIM's authentication of email identity can assist in the global control of "spam" and "phishing". This document provides implementation, deployment, operational, and migration considerations for DKIM.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5863.

Table of Contents

1. Introduction

DomainKeys Identified Mail (DKIM) allows an organization to claim responsibility for transmitting a message, in a way that can be validated by a recipient. This document provides practical tips for those who are developing DKIM software, mailing list managers, filtering strategies based on the output from DKIM verification, and DNS servers; those who are deploying DKIM software, keys, mailing list software, and migrating from DomainKeys [RFC4870]; and those who are responsible for the ongoing operations of an email infrastructure that has deployed DKIM.

The reader is encouraged to read the DKIM Service Overview document [RFC5585] before this document. More detailed guidance about DKIM and Author Domain Signing Practices (ADSP) can also be found in the protocol specifications [RFC4871], [RFC5617], and [RFC5672].

The document is organized around the key concepts related to DKIM. Within each section, additional considerations specific to development, deployment, or ongoing operations are highlighted where appropriate. The possibility of the use of DKIM results as input to a local reputation database is also discussed.

2. Using DKIM as Part of Trust Assessment

2.1. A Systems View of Email Trust Assessment

DKIM participates in a trust-oriented enhancement to the Internet's email service, to facilitate message handling decisions, such as for delivery and for content display. Trust-oriented message handling has substantial differences from the more established approaches that consider messages in terms of risk and abuse. With trust, there is a collaborative exchange between a willing participant along the sending path and a willing participant at a recipient site. In contrast, the risk model entails independent, unilateral action by the recipient site, in the face of a potentially unknown, hostile, and deceptive sender. This translates into a very basic technical difference: in the face of unilateral action by the recipient and even antagonistic efforts by the sender, risk-oriented mechanisms are based on heuristics, that is, on guessing. Guessing produces statistical results with some false negatives and some false positives. For trust-based exchanges, the goal is the deterministic exchange of information. For DKIM, that information is the one identifier that represents a stream of mail for which an independent assessment is sought (by the signer).

A trust-based service is built upon a validated Responsible Identifier that labels a stream of mail and is controlled by an identity (role, person, or organization). The identity is acknowledging some degree of responsibility for the message stream. Given a basis for believing that an identifier is being used in an authorized manner, the recipient site can make and use an assessment of the associated identity. An identity can use different identifiers, on the assumption that the different streams might produce different assessments. For example, even the best-run marketing campaigns will tend to produce some complaints that can affect the reputation of the associated identifier, whereas a stream of transactional messages is likely to have a more pristine reputation.

Determining that the identifier's use is valid is quite different from determining that the content of a message is valid. The former means only that the identifier for the responsible role, person, or organization has been legitimately associated with a message. The latter means that the content of the message can be believed and, typically, that the claimed author of the content is correct. DKIM validates only the presence of the identifier used to sign the message. Even when this identifier is validated, DKIM carries no implication that any of the message content, including the RFC5322.From field [RFC5322], is valid. Surprisingly, this limit to the semantics of a DKIM signature applies even when the validated signing identifier is the same domain name as is used in the RFC5322.From field! DKIM's only claim about message content is that the content cited in the DKIM-Signature: field's h= tag has been delivered without modification. That is, it asserts message content integrity -- between signing and verifying -- not message content validity.

As shown in Figure 1, this enhancement is a communication between a responsible role, person, or organization that signs the message and a recipient organization that assesses its trust in the signer. The recipient then makes handling decisions based on a collection of assessments, of which the DKIM mechanism is only a part. In this model, as shown in Figure 1, validation is an intermediary step, having the sole task of passing a validated Responsible Identifier to the Identity Assessor. The communication is of a single Responsible Identifier that the Responsible Identity wishes to have used by the Identity Assessor. The Identifier is the sole, formal input and output value of DKIM signing. The Identity Assessor uses this single, provided Identifier for consulting whatever assessment databases are deemed appropriate by the assessing entity. In turn, output from the Identity Assessor is fed into a Handling Filter

engine that considers a range of factors, along with this single output value. The range of factors can include ancillary information from the DKIM validation.

Identity Assessment covers a range of possible functions. It can be as simple as determining whether the identifier is a member of some list, such as authorized operators or participants in a group that might be of interest for recipient assessment. Equally, it can indicate a degree of trust (reputation) that is to be afforded the actor using that identifier. The extent to which the assessment affects the handling of the message is, of course, determined later, by the Handling Filter.

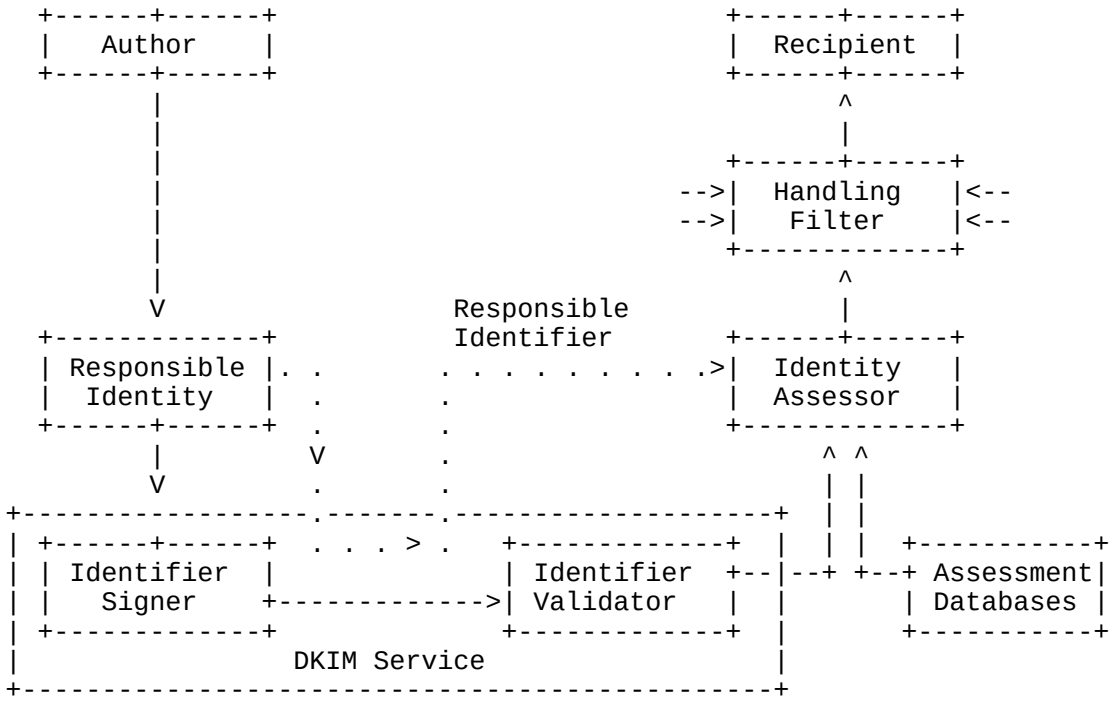

Figure 1: Actors in a Trust Sequence Using DKIM

2.2. Choosing a DKIM Tag for the Assessment Identifier

The signer of a message needs to be able to provide precise data and know what that data will mean upon delivery to the Assessor. If there is ambiguity in the choice that will be made on the recipient side, then the sender cannot know what basis for assessment will be used. DKIM has three values that specify identification information and it is easy to confuse their use, although only one defines the

formal input and output of DKIM, with the other two being used for internal protocol functioning and adjunct purposes, such as auditing and debugging.

The salient values include the s=, d= and i= parameters in the DKIM-Signature: header field. In order to achieve the end-to-end determinism needed for this collaborative exchange from the signer to the assessor, the core model needs to specify what the signer is required to provide to the assessor. The update to RFC 4871 [RFC5672] specifies:

> DKIM's primary task is to communicate from the Signer to a recipient-side Identity Assessor a single Signing Domain Identifier (SDID) that refers to a responsible identity. DKIM MAY optionally provide a single responsible Agent or User Identifier (AUID)... A receive-side DKIM verifier MUST communicate the Signing Domain Identifier (d=) to a consuming Identity Assessor module and MAY communicate the User Agent Identifier (i=) if present.... To the extent that a receiver attempts to intuit any structured semantics for either of the identifiers, this is a heuristic function that is outside the scope of DKIM's specification and semantics.

The single, mandatory value that DKIM supplies as its output is:

d= This specifies the "domain of the signing entity". It is a domain name and is combined with the selector to form a DNS query. A receive-side DKIM verifier needs to communicate the Signing Domain Identifier (d=) to a consuming Identity Assessor module and can also communicate the User Agent Identifier (i=) if present.

The adjunct values are:

s= This tag specifies the selector. It is used to discriminate among different keys that can be used for the same d= domain name. As discussed in Section 4.3 of [RFC5585], "If verifiers were to employ the selector as part of an assessment mechanism, then there would be no remaining mechanism for making a transition from an old, or compromised, key to a new one". Consequently, the selector is not appropriate for use as part or all of the identifier used to make assessments.

i= This tag is optional and provides the "[t]he Agent or User Identifier (AUID) on behalf of which the SDID is taking responsibility" [RFC5672]. The identity can be in the syntax

of an entire email address or only a domain name. The domain name can be the same as for d= or it can be a sub-name of the d= name.

NOTE: Although the i= identity has the syntax of an email address, it is not required to have those semantics. That is, "the identity of the user" need not be the same as the user's mailbox. For example, the signer might wish to use i= to encode user-related audit information, such as how they were accessing the service at the time of message posting. Therefore, it is not possible to conclude anything from the i= string's (dis)similarity to email addresses elsewhere in the header.

So, i= can have any of these properties:

* Be a valid domain when it is the same as d=

* Appear to be a subdomain of d= but might not even exist

* Look like a mailbox address but might have different semantics and therefore not function as a valid email address

* Be unique for each message, such as indicating access details of the user for the specific posting

This underscores why the tag needs to be treated as being opaque, since it can represent any semantics, known only to the signer.

Hence, i= serves well as a token that is usable like a Web cookie, for return to the signing Administrative Management Domain (ADMD) -- such as for auditing and debugging. Of course in some scenarios the i= string might provide a useful adjunct value for additional (heuristic) processing by the Handling Filter.

2.3. Choosing the Signing Domain Name

A DKIM signing entity can serve different roles, such as being the author of content, the operator of the mail service, or the operator of a reputation service that also provides signing services on behalf of its customers. In these different roles, the basis for distinguishing among portions of email traffic can vary. For an entity creating DKIM signatures, it is likely that different portions of its mail will warrant different levels of trust. For example:

* Mail is sent for different purposes, such as marketing versus transactional, and recipients demonstrate different patterns of acceptance between these.

* For an operator of an email service, there often are distinct sub-populations of users warranting different levels of trust or privilege, such as paid versus free users, or users engaged in direct correspondence versus users sending bulk mail.

* Mail originating outside an operator's system, such as when it is redistributed by a mailing-list service run by the operator, will warrant a different reputation from mail submitted by users authenticated with the operator.

It is therefore likely to be useful for a signer to use different d= subdomain names, for different message traffic streams, so that receivers can make differential assessments. However, too much differentiation -- that is, too fine a granularity of signing domains -- makes it difficult for the receiver to discern a sufficiently stable pattern of traffic for developing an accurate and reliable assessment. So the differentiation needs to achieve a balance. Generally, in a trust system, legitimate signers have an incentive to pick a small stable set of identities, so that recipients and others can attribute reputations to them. The set of these identities a receiver trusts is likely to be quite a bit smaller than the set it views as risky.

The challenge in using additional layers of subdomains is whether the extra granularity will be useful for the Assessor. In fact, excessive levels invite ambiguity: if the Assessor does not take advantage of the added granularity in the entire domain name that is provided, they might unilaterally decide to use only some rightmost part of the identifier. The signer cannot know what portion will be used. That ambiguity would move the use of DKIM back to the realm of heuristics, rather than the deterministic processing that is its goal.

Hence, the challenge is to determine a useful scheme for labeling different traffic streams. The most obvious choices are among different types of content and/or different types of authors. Although stability is essential, it is likely that the choices will change, over time, so the scheme needs to be flexible.

For those originating message content, the most likely choice of subdomain naming scheme will by based upon type of content, which can use content-oriented labels or service-oriented labels. For example:

transaction.example.com
newsletter.example.com
bugreport.example.com
support.example.com
sales.example.com
marketing.example.com

where the choices are best dictated by whether they provide the Identity Assessor with the ability to discriminate usefully among streams of mail that demonstrate significantly different degrees of recipient acceptance or safety. Again, the danger in providing too fine a granularity is that related message streams that are labeled separately will not benefit from an aggregate reputation.

For those operating messaging services on behalf of a variety of customers, an obvious scheme to use has a different subdomain label for each customer. For example:

widgetco.example.net
moviestudio.example.net
bigbank.example.net

However, it can also be appropriate to label by the class of service or class of customer, such as:

premier.example.net
free.example.net
certified.example.net

Prior to using domain names for distinguishing among sources of data, IP Addresses have been the basis for distinction. Service operators typically have done this by dedicating specific outbound IP Addresses to specific mail streams -- typically to specific customers. For example, a university might want to distinguish mail from the administration, versus mail from the student dorms. In order to make the adoption of a DKIM-based service easier, it can be reasonable to translate the same partitioning of traffic, using domain names in place of the different IP Addresses.

2.4. Recipient-Based Assessments

DKIM gives the recipient site’s Identity Assessor a verifiable identifier to use for analysis. Although the mechanism does not make claims that the signer is a Good Actor or a Bad Actor, it does make

it possible to know that use of the identifier is valid. This is in marked contrast with schemes that do not have authentication. Without verification, it is not possible to know whether the identifier -- whether taken from the RFC5322.From field, the RFC5321.MailFrom command, or the like -- is being used by an authorized agent. DKIM solves this problem. Hence, with DKIM, the Assessor can know that two messages with the same DKIM d= identifier are, in fact, signed by the same person or organization. This permits a far more stable and accurate assessment of mail traffic using that identifier.

DKIM is distinctive, in that it provides an identifier that is not necessarily related to any other identifier in the message. Hence, the signer might be the author's ADMD, one of the operators along the transit path, or a reputation service being used by one of those handling services. In fact, a message can have multiple signatures, possibly by any number of these actors.

As discussed above, the choice of identifiers needs to be based on differences that the signer thinks will be useful for the recipient Assessor. Over time, industry practices establish norms for these choices.

> Absent such norms, it is best for signers to distinguish among streams that have significant differences, while consuming the smallest number of identifiers possible. This will limit the burden on recipient Assessors.

A common view about a DKIM signature is that it carries a degree of assurance about some or all of the message contents, and in particular, that the RFC5322.From field is likely to be valid. In fact, DKIM makes assurances only about the integrity of the data and not about its validity. Still, presumptions of the RFC5322.From field validity remain a concern. Hence, a signer using a domain name that is unrelated to the domain name in the RFC5322.From field can reasonably expect that the disparity will warrant some curiosity, at least until signing by independent operators has produced some established practice among recipient Assessors.

With the identifier(s) supplied by DKIM, the Assessor can consult an independent assessment service about the entity associated with the identifier(s). Another possibility is that the Assessor can develop its own reputation rating for the identifier(s). That is, over time, the Assessor can observe the stream of messages associated with the identifier(s) developing a reaction to associated content. For example, if there is a high percentage of user complaints regarding

signed mail with a d= value of "widgetco.example.net", the Assessor might include that fact in the vector of data it provides to the Handling Filter. This is also discussed briefly in Section 5.4.

2.5. Filtering

The assessment of the signing identifier is given to a Handling Filter that is defined by local policies, according to a potentially wide range of different factors and weightings. This section discusses some of the kinds of choices and weightings that are plausible and the differential actions that might be performed. Because authenticated domain names represent a collaborative sequence between signer and Assessor, actions can sometimes reasonably include contacting the signer.

The discussion focuses on variations in Organizational Trust versus Message Stream Risk, that is, the degree of positive assessment of a DKIM-signing organization, and the potential danger present in the message stream signed by that organization. While it might seem that higher trust automatically means lower risk, the experience with real-world operations provides examples of every combination of the two factors, as shown in Figure 2. For each axis, only three levels of granularity are listed, in order to keep discussion manageable. In real-world filtering engines, finer-grained distinctions are typically needed, and there typically are more axes. For example, there are different types of risk, so that an engine might distinguish between spam risk versus virus risk and take different actions based on which type of problematic content is present. For spam, the potential damage from a false negative is small, whereas the damage from a false positive is high. For a virus, the potential danger from a false negative is extremely high, while the likelihood of a false positive when using modern detection tools is extremely low. However, for the discussion here, "risk" is taken as a single construct.

The DKIM d= identifier is independent of any other identifier in a message and can be a subdomain of the name owned by the signer. This permits the use of fine-grained and stable distinctions between different types of message streams, such as between transactional messages and marketing messages from the same organization. Hence, the use of DKIM might permit a richer filtering model than has typically been possible for mail-receiving engines.

Note that the realities of today’s public Internet Mail environment necessitate having a baseline handling model that is quite suspicious. Hence, "strong" filtering rules really are the starting point, as indicated for the UNKNOWN cell.

The table indicates differential handling for each combination, such as how aggressive or broad-based the filtering could be. Aggressiveness affects the types of incorrect assessments that are likely. So, the table distinguishes various characteristics, including: 1) whether an organization is unknown, known to be good actors, or known to be bad actors; and 2) the assessment of messages. It includes advice about the degree of filtering that might be done, and other message disposition. Perhaps unexpectedly, it also lists a case in which the receiving site might wish to deliver problematic mail, rather than redirecting or deleting it. The site might also wish to contact the signing organization and seek resolution of the problem.

STREAM RISK * ORGANIZATIONAL TRUST: 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

Figure 2: Trust versus Risk Handling Tradeoffs Example

[LEGEND]

AXES

Stream Risk: This is a measure of the recent history of a message stream and the severity of problems it has presented.

Organizational Trust: This combines longer-term history about possible stream problems from that organization, and its responsiveness to problem handling.

CELLS (indicating reasonable responses)

Labels for the cells are meant as a general assessment of an organization producing that type of mail stream under that circumstance.

Benign: There is some history of sending good messages, with very few harmful messages having been received. This stream warrants filtering that does not search for problems very aggressively, in order to reduce the likelihood of false positives.

Diligent: The stream has had a limited degree of problems and the organization is consistently successful at controlling their abuse issues and in a timely manner.

Pristine: There is a history of a clean message stream with no problems, from an organization with an excellent reputation. So, the filter primarily needs to ensure that messages are delivered; catching stray problem messages is a lesser concern. In other words, the paramount concern, here, is false positives.

Unknown: There is no history with the organization. Apply an aggressive level of "naive" filtering, given the nature of the public email environment.

Typical: The stream suffers significant abuse issues and the organization has demonstrated a record of having difficulties resolving them in a timely manner, in spite of legitimate efforts. Unfortunately, this is the typical case for service providers with an easy and open subscription policy.

Protected: An organization with a good history and/or providing an important message stream for the receiving site is subject to a local policy that messages are not allowed to be blocked, but the stream is producing a problematic stream. The receiver delivers messages, but works quickly with the organization to resolve the matter.

Malicious: A persistently problematic message stream is coming from an organization that appears to contribute to the problem. The stream will be blocked, but the organization’s role is sufficiently troubling to warrant following up with others in the anti-abuse or legal communities, to constrain or end their impact.

Negligent: A persistently problematic message stream is coming from an organization that does not appear to be contributing to the problem, but also does not appear to be working to eliminate it. At the least, the stream needs to be blocked.

Compromised: An organization with a good history has a stream that changes and becomes too problematic to be delivered. The receiver blocks the stream and works quickly with the organization to resolve the matter.

3. DKIM Key Generation, Storage, and Management

By itself, verification of a digital signature only allows the verifier to conclude with a very high degree of certainty that the signature was created by a party with access to the corresponding private signing key. It follows that a verifier requires means to (1) obtain the public key for the purpose of verification and (2) infer useful attributes of the key holder.

In a traditional Public Key Infrastructure (PKI), the functions of key distribution and key accreditation are separated. In DKIM [RFC4871], these functions are both performed through the DNS.

In either case, the ability to infer semantics from a digital signature depends on the assumption that the corresponding private key is only accessible to a party with a particular set of attributes. In a traditional PKI, a Trusted Third Party (TTP) vouches that the key holder has been validated with respect to a specified set of attributes. The range of attributes that can be attested in such a scheme is thus limited only to the type of attributes that a TTP can establish effective processes for validating. In DKIM, TTPs are not employed and the functions of key distribution and accreditation are combined.

Consequently, there are only two types of inference that a signer can make from a key published in a DKIM key record:

1. That a party with the ability to control DNS records within a DNS zone intends to claim responsibility for messages signed using the corresponding private signature key.

2. That use of a specific key is restricted to the particular subset of messages identified by the selector.

The ability to draw any useful conclusion from verification of a digital signature relies on the assumption that the corresponding private key is only accessible to a party with a particular set of

attributes. In the case of DKIM, this means that the party that created the corresponding DKIM key record in the specific zone intended to claim responsibility for the signed message.

Ideally, we would like to draw a stronger conclusion, that if we obtain a DKIM key record from the DNS zone example.com, that the legitimate holder of the DNS zone example.com claims responsibility for the signed message. In order for this conclusion to be drawn, it is necessary for the verifier to assume that the operational security of the DNS zone and corresponding private key are adequate.

3.1. Private Key Management: Deployment and Ongoing Operations

Access to signing keys needs to be carefully managed to prevent use by unauthorized parties and to minimize the consequences if a compromise were to occur.

While a DKIM signing key is used to sign messages on behalf of many mail users, the signing key itself needs to be under direct control of as few key holders as possible. If a key holder were to leave the organization, all signing keys held by that key holder need to be withdrawn from service and, if appropriate, replaced.

If key management hardware support is available, it needs to be used. If keys are stored in software, appropriate file control protections need to be employed, and any location in which the private key is stored in plaintext form needs to be excluded from regular backup processes and is best not accessible through any form of network including private local area networks. Auditing software needs to be used periodically to verify that the permissions on the private key files remain secure.

Wherever possible, a signature key needs to exist in exactly one location and be erased when no longer used. Ideally, a signature key pair needs to be generated as close to the signing point as possible, and only the public key component transferred to another party. If this is not possible, the private key needs to be transported in an encrypted format that protects the confidentiality of the signing key. A shared directory on a local file system does not provide adequate security for distribution of signing keys in plaintext form.

Key escrow schemes are not necessary and are best not used. In the unlikely event of a signing key becoming lost, a new signature key pair can be generated as easily as recovery from a key escrow scheme.

To enable accountability and auditing:

- Responsibility for the security of a signing key needs to ultimately vest in a single named individual.

- Where multiple parties are authorized to sign messages, each signer needs to use a different key to enable accountability and auditing.

Best practices for management of cryptographic keying material require keying material to be refreshed at regular intervals, particularly where key management is achieved through software. While this practice is highly desirable, it is of considerably less importance than the requirement to maintain the secrecy of the corresponding private key. An operational practice in which the private key is stored in tamper-proof hardware and changed once a year is considerably more desirable than one in which the signature key is changed on an hourly basis but maintained in software.

3.2. Storing Public Keys: DNS Server Software Considerations

In order to use DKIM, a DNS domain holder requires (1) the ability to create the necessary DKIM DNS records and (2) sufficient operational security controls to prevent insertion of spurious DNS records by an attacker.

DNS record management is often operated by an administrative staff that is different from those who operate an organization’s email service. In order to ensure that DKIM DNS records are accurate, this imposes a requirement for careful coordination between the two operations groups. If the best practices for private key management described above are observed, such deployment is not a one-time event; DNS DKIM selectors will be changed over time as signing keys are terminated and replaced.

At a minimum, a DNS server that handles queries for DKIM key records needs to allow the server administrators to add free-form TXT records. It would be better if the DKIM records could be entered using a structured form, supporting the DKIM-specific fields.

Ideally, DNS Security (DNSSEC) [RFC4034] needs to be employed in a configuration that provides protection against record insertion attacks and zone enumeration. In the case that NextSECure version 3 (NSEC3) [RFC5155] records are employed to prevent insertion attack, the OPT-OUT flag needs to be clear. (See [RFC5155] section 6 for details.)

3.2.1. Assignment of Selectors

Selectors are assigned according to the administrative needs of the signing domain, such as for rolling over to a new key or for the delegation of the right to authenticate a portion of the namespace to a TTP. Examples include:

jun2005.eng._domainkey.example.com

widget.promotion._domainkey.example.com

It is intended that assessments of DKIM identities be based on the domain name, and not include the selector. While past practice of a signer can permit a verifier to infer additional properties of particular messages from the structure DKIM key selector, unannounced administrative changes such as a change of signing software can cause such heuristics to fail at any time.

3.3. Per-User Signing Key Management Issues

While a signer can establish business rules, such as the issue of individual signature keys for each end-user, DKIM makes no provision for communicating these to other parties. Out-of-band distribution of such business rules is outside the scope of DKIM. Consequently, there is no means by which external parties can make use of such keys to attribute messages with any greater granularity than a DNS domain.

If per-user signing keys are assigned for internal purposes (e.g., authenticating messages sent to an MTA (Mail Transfer Agent) for distribution), the following issues need to be considered before using such signatures as an alternative to traditional edge signing at the outbound MTA:

External verifiers will be unable to make use of the additional signature granularity without access to additional information passed out of band with respect to [RFC4871].

If the number of user keys is large, the efficiency of local caching of key records by verifiers will be lower.

A large number of end users is be less likely to do an adequate job of managing private key data securely on their personal computers than is an administrator running an edge MTA.

3.4. Third-Party Signer Key Management and Selector Administration

A DKIM key record only asserts that the holder of the corresponding domain name makes a claim of responsibility for messages signed under the corresponding key. In some applications, such as bulk mail delivery, it is desirable to delegate use of the key. That is, to allow a third party to sign on behalf of the domain holder. The trust relationship is still established between the domain holder and the verifier, but the private signature key is held by a third party.

Signature keys used by a third-party signer need to be kept entirely separate from those used by the domain holder and other third-party signers. To limit potential exposure of the private key, the signature key pair needs to be generated by the third-party signer and the public component of the key transmitted to the domain holder, rather than have the domain holder generate the key pair and transmit the private component to the third-party signer.

Domain holders needs to adopt a least-privilege approach and grant third-party signers the minimum access necessary to perform the desired function. Limiting the access granted to third-party signers serves to protect the interests of both parties. The domain holder minimizes its security risk and the TTP signer avoids unnecessary liability.

In the most restrictive case, domain holders maintain full control over the creation of key records. They can employ appropriate key record restrictions to enforce limits on the messages for which the third-party signer is able to sign. If such restrictions are impractical, the domain holder needs to delegate a DNS subzone for publishing key records to the third-party signer. It is best that the domain holder NOT allow a third-party signer unrestricted access to its DNS service for the purpose of publishing key records.

3.5. Key Pair / Selector Life Cycle Management

Deployments need to establish, document, and observe processes for managing the entire life cycle of an asymmetric key pair.

3.5.1. Example Key Deployment Process

When it is determined that a new key pair is required:

1. A Key Pair is generated by the signing device.

2. A proposed key selector record is generated and transmitted to the DNS administration infrastructure.

3. The DNS administration infrastructure verifies the authenticity of the key selector registration request. If accepted:

    1. A key selector is assigned.

    2. The corresponding key record is published in the DNS.

    3. Wait for DNS updates to propagate (if necessary).

    4. Report assigned key selector to signing device.

4. The signer verifies correct registration of the key record.

5. The signer begins generating signatures using the new key pair.

6. The signer terminates any private keys that are no longer required due to issue of replacement.

3.5.2. Example Key Termination Process

When it is determined that a private signature key is no longer required:

1. The signer stops using the private key for signature operations.

2. The signer deletes all records of the private key, including in-memory copies at the signing device.

3. The signer notifies the DNS administration infrastructure that the signing key is withdrawn from service and that the corresponding key records can be withdrawn from service at a specified future date.

4. The DNS administration infrastructure verifies the authenticity of the key selector termination request. If accepted,

    1. The key selector is scheduled for deletion at a future time determined by site policy.

    2. Wait for deletion time to arrive.

    3. The signer either publishes a revocation key selector with an empty public-key data (p=) field, or deletes the key selector record entirely.

5. As far as the verifier is concerned, there is no functional difference between verifying against a key selector with an empty p= field, and verifying against a missing key selector: both

result in a failed signature and the signature needs to be treated as if it had not been there. However, there is a minor semantic difference: with the empty p= field, the signer is explicitly stating that the key has been revoked. The empty p= record provides a gravestone for an old selector, making it less likely that the selector might be accidentally reused with a different public key.

4. Signing

Creating messages that have one or more DKIM signatures requires support in only two outbound email service components:

- A DNS Administrative interface that can create and maintain the relevant DNS names -- including names with underscores -- and resource records (RR).

- A trusted module, called the signing module, which is within the organization's outbound email handling service and which creates and adds the DKIM-Signature: header field(s) to the message.

If the module creates more than one signature, there needs to be the appropriate means of telling it which one(s) to use. If a large number of names are used for signing, it will help to have the administrative tool support a batch-processing mode.

4.1. DNS Records

A receiver attempting to verify a DKIM signature obtains the public key that is associated with the signature for that message. The DKIM-Signature: header in the message contains the d= tag with the basic domain name doing the signing and serving as output to the Identity Assessor and the s= tag with the selector that is added to the name, for finding the specific public key. Hence, the relevant <selector>._domainkey.<domain-name> DNS record needs to contain a DKIM-related RR that provides the public key information.

The administrator of the zone containing the relevant domain name adds this information. Initial DKIM DNS information is contained within TXT RRs. DNS administrative software varies considerably in its abilities to support DKIM names, such as with underscores, and to add new types of DNS information.

4.2. Signing Module

The module doing signing can be placed anywhere within an organization's trusted Administrative Management Domain (ADMD); obvious choices include department-level posting agents, as well as

outbound boundary MTAs to the open Internet. However, any other module, including the author's MUA (Mail User Agent), is potentially acceptable, as long as the signature survives any remaining handling within the ADMD. Hence, the choice among the modules depends upon software development, administrative overhead, security exposures, and transit-handling tradeoffs. One perspective that helps to resolve this choice is the difference between the increased flexibility, from placement at (or close to) the MUA, versus the streamlined administration and operation that is more easily obtained by implementing the mechanism "deeper" into the organization's email infrastructure, such as at its boundary MTA.

Note the discussion in Section 2.2 concerning the use of the i= tag.

The signing module uses the appropriate private key to create one or more signatures. (See Section 6.5 for a discussion of multiple signatures.) The means by which the signing module obtains the private key(s) is not specified by DKIM. Given that DKIM is intended for use during email transit, rather than for long-term storage, it is expected that keys will be changed regularly. For administrative convenience, it is best not to hard-code key information into software.

4.3. Signing Policies and Practices

Every organization (ADMD) will have its own policies and practices for deciding when to sign messages (message stream) and with what domain name, selector, and key. Examples of particular message streams include all mail sent from the ADMD versus mail from particular types of user accounts versus mail having particular types of content. Given this variability, and the likelihood that signing practices will change over time, it will be useful to have these decisions represented through run-time configuration information, rather than being hard-coded into the signing software.

As noted in Section 2.3, the choice of signing name granularity requires balancing administrative convenience and utility for recipients. Too much granularity is higher administrative overhead and might well attempt to impose more differential analysis on the recipient than they wish to support. In such cases, they are likely to use only a super-name -- right-hand substring -- of the signing name. When this occurs, the signer will not know what portion is being used; this then moves DKIM back to the non-deterministic world of heuristics, rather than the mechanistic world of signer/recipient collaboration that DKIM seeks.

5. Verifying

A message recipient can verify a DKIM signature to determine if a claim of responsibility has been made for the message by a trusted domain.

Access control requires two components: authentication and authorization. By design, verification of a DKIM signature only provides the authentication component of an access control decision and needs to be combined with additional sources of information such as reputation data to arrive at an access control decision.

5.1. Intended Scope of Use

DKIM requires that a message with a signature that is found to be invalid is to be treated as if the message had not been signed at all.

If a DKIM signature fails to verify, it is entirely possible that the message is valid and that either there is a configuration error in the signer's system (e.g., a missing key record) or that the message was inadvertently modified in transit. It is thus undesirable for mail infrastructure to treat messages with invalid signatures less favorably than those with no signatures whatsoever. Contrariwise, creation of an invalid signature requires a trivial amount of effort on the part of an attacker. If messages with invalid signatures were to be treated preferentially to messages with no signatures whatsoever, attackers will simply add invalid signature blocks to gain the preferential treatment. It follows that messages with invalid signatures need to be treated no better and no worse than those with no signature at all.

5.2. Signature Scope

As with any other digital signature scheme, verifiers need to consider only the part of the message that is inside the scope of the message as being authenticated by the signature.

For example, if the l= option is employed to specify a content length for the scope of the signature, only the part of the message that is within the scope of the content signature would be considered authentic.

5.3. Design Scope of Use

Public key cryptography provides an exceptionally high degree of assurance, bordering on absolute certainty, that the party that created a valid digital signature had access to the private key corresponding to the public key indicated in the signature.

In order to make useful conclusions from the verification of a valid digital signature, the verifier is obliged to make assumptions that fall far short of absolute certainty. Consequently, mere validation of a DKIM signature does not represent proof positive that a valid claim of responsibility was made for it by the indicated party, that the message is authentic, or that the message is not abusive. In particular:

- The legitimate private key holder might have lost control of its private key.
- The legitimate domain holder might have lost control of the DNS server for the zone from which the key record was retrieved.
- The key record might not have been delivered from the legitimate DNS server for the zone from which the key record was retrieved.
- Ownership of the DNS zone might have changed.

In practice, these limitations have little or no impact on the field of use for which DKIM is designed, but they can have a bearing if use is made of the DKIM message signature format or key retrieval mechanism in other specifications.

In particular, the DKIM key retrieval mechanism is designed for ease of use and deployment rather than to provide a high assurance Public Key Infrastructure suitable for purposes that require robust non-repudiation such as establishing legally binding contracts. Developers seeking to extend DKIM beyond its design application need to consider replacing or supplementing the DNS key retrieval mechanism with one that is designed to meet the intended purposes.

5.4. Inbound Mail Filtering

DKIM is frequently employed in a mail filtering strategy to avoid performing content analysis on email originating from trusted sources. Messages that carry a valid DKIM signature from a trusted source can be whitelisted, avoiding the need to perform computation and hence energy-intensive content analysis to determine the disposition of the message.

Mail sources can be determined to be trusted by means of previously observed behavior and/or reference to external reputation or accreditation services. The precise means by which this is accomplished is outside the scope of DKIM.

5.4.1. Non-Verifying Adaptive Spam Filtering Systems

Adaptive (or learning) spam filtering mechanisms that are not capable of verifying DKIM signatures need to, at minimum, be configured to ignore DKIM header data entirely.

5.5. Messages Sent through Mailing Lists and Other Intermediaries

Intermediaries, such as mailing lists, pose a particular challenge for DKIM implementations, as the message processing steps performed by the intermediary can cause the message content to change in ways that prevent the signature passing verification.

Such intermediaries are strongly encouraged to deploy DKIM signing so that a verifiable claim of responsibility remains available to parties attempting to verify the modified message.

5.6. Generation, Transmission, and Use of Results Headers

In many deployments, it is desirable to separate signature verification from the application relying on the verification. A system can choose to relay information indicating the results of its message authentication efforts using various means; adding a "results header" to the message is one such mechanism [RFC5451]. For example, consider the cases where:

- The application relying on DKIM signature verification is not capable of performing the verification.
- The message can be modified after the signature verification is performed.
- The signature key cannot be available by the time that the message is read.

In such cases, it is important that the communication link between the signature verifier and the relying application be sufficiently secure to prevent insertion of a message that carries a bogus results header.

An intermediary that generates results headers need to ensure that relying applications are able to distinguish valid results headers issued by the intermediary from those introduced by an attacker. For

example, this can be accomplished by signing the results header. At a minimum, results headers on incoming messages need to be removed if they purport to have been issued by the intermediary but cannot be verified as authentic.

Further discussion on trusting the results as relayed from a verifier to something downstream can be found in [RFC5451].

6. Taxonomy of Signatures

As described in Section 2.1, a DKIM signature tells the signature verifier that the owner of a particular domain name accepts some responsibility for the message. It does not, in and of itself, provide any information about the trustworthiness or behavior of that identity. What it does provide is a verified identity to which such behavioral information can be associated, so that those who collect and use such information can be assured that it truly pertains to the identity in question.

This section lays out a taxonomy of some of the different identities, or combinations of identities, that might usefully be represented by a DKIM signature.

6.1. Single Domain Signature

Perhaps the simplest case is when an organization signs its own outbound email using its own domain in the SDID [RFC5672] of the signature. For example, Company A would sign the outbound mail from its employees with d=companyA.example.

In the most straightforward configuration, the addresses in the RFC5322.From field would also be in the companyA.example domain, but that direct correlation is not required.

A special case of the single domain signature is an author signature as defined by the Author Domain Signing Practices specification [RFC5617]. Author signatures are signatures from an author’s organization that have an SDID value that matches that of an RFC5322.From address of the signed message.

Although an author signature might, in some cases, be proof against spoofing the domain name of the RFC5322.From address, it is important to note that the DKIM and ADSP validation apply only to the exact address string and not to look-alike addresses or to the human-friendly "display-name" or names and addresses used within the body of the message. That is, it only protects against the misuse of a precise address string within the RFC5322.From field and nothing else. For example, a message from bob@domain.example with a valid

signature where d=d0main.example would fail an ADSP check because the signature domain, however similar, is distinct; however, a message from bob@d0main.example with a valid signature where d=d0main.example would pass an ADSP check, even though to a human it might be obvious that d0main.example is likely a malicious attempt to spoof the domain domain.example. This example highlights that ADSP, like DKIM, is only able to validate a signing identifier: it still requires some external process to attach a meaningful reputation to that identifier.

6.2. Parent Domain Signature

Another approach that might be taken by an organization with multiple active subdomains is to apply the same (single) signature domain to mail from all subdomains. In this case, the signature chosen would usually be the signature of a parent domain common to all subdomains. For example, mail from marketing.domain.example, sales.domain.example, and engineering.domain.example might all use a signature where d=domain.example.

This approach has the virtue of simplicity, but it is important to consider the implications of such a choice. As discussed in Section 2.3, if the type of mail sent from the different subdomains is significantly different or if there is reason to believe that the reputation of the subdomains would differ, then it can be a good idea to acknowledge this and provide distinct signatures for each of the subdomains (d=marketing.domain.example, sales.domain.example, etc.). However, if the mail and reputations are likely to be similar, then the simpler approach of using a single common parent domain in the signature can work well.

Another approach to distinguishing the streams using a single DKIM key would be to leverage the AUID [RFC5672] (i= tag) in the DKIM signature to differentiate the mail streams. For example, marketing email would be signed with i=@marketing.domain.example and d=domain.example.

It’s important to remember, however, that under core DKIM semantics, the AUID is opaque to receivers. That means that it will only be an effective differentiator if there is an out-of-band agreement about the i= semantics.

6.3. Third-Party Signature

A signature whose domain does not match the domain of the RFC5322.From address is sometimes referred to as a third-party signature. In certain cases, even the parent domain signature

described above would be considered a third-party signature because it would not be an exact match for the domain in the RFC5322.From address.

Although there is often heated debate about the value of third party signatures, it is important to note that the DKIM specification attaches no particular significance to the identity in a DKIM signature ([RFC4871], [RFC5672]). The identity specified within the signature is the identity that is taking responsibility for the message, and it is only the interpretation of a given receiver that gives one identity more or less significance than another. In particular, most independent reputation services assign trust based on the specific identifier string, not its "role": in general they make no distinction between, for example, an author signature and a third-party signature.

For some, a signature unrelated to the author domain (the domain in the RFC5322.From address) is less valuable because there is an assumption that the presence of an author signature guarantees that the use of the address in the RFC5322.From header is authorized.

For others, that relevance is tied strictly to the recorded behavioral data assigned to the identity in question, i.e., its trust assessment or reputation. The reasoning here is that an identity with a good reputation is unlikely to maintain that good reputation if it is in the habit of vouching for messages that are unwanted or abusive; in fact, doing so will rapidly degrade its reputation so that future messages will no longer benefit from it. It is therefore low risk to facilitate the delivery of messages that contain a valid signature of a domain with a strong positive reputation, independent of whether or not that domain is associated with the address in the RFC5322.From header field of the message.

Third-party signatures encompass a wide range of identities. Some of the more common are:

Service Provider: In cases where email is outsourced to an Email Service Provider (ESP), Internet Service Provider (ISP), or other type of service provider, that service provider can choose to DKIM-sign outbound mail with either its own identifier -- relying on its own, aggregate reputation -- or with a subdomain of the provider that is unique to the message author but still part of the provider’s aggregate reputation. Such service providers can also encompass delegated business functions such as benefit management, although these will more often be treated as trusted third-party senders (see below).

Parent Domain: As discussed above, organizations choosing to apply a parent-domain signature to mail originating from subdomains can have their signatures treated as third party by some verifiers, depending on whether or not the "t=s" tag is used to constrain the parent signature to apply only to its own specific domain. The default is to consider a parent-domain signature valid for its subdomains.

Reputation Provider: Another possible category of third-party signature would be the identity of a third-party reputation provider. Such a signature would indicate to receivers that the message was being vouched for by that third party.

6.4. Using Trusted Third-Party Senders

For most of the cases described so far, there has been an assumption that the signing agent was responsible for creating and maintaining its own DKIM signing infrastructure, including its own keys, and signing with its own identity.

A different model arises when an organization uses a trusted third-party sender for certain key business functions, but still wants that email to benefit from the organization's own identity and reputation. In other words, the mail would come out of the trusted third party's mail servers, but the signature applied would be that of the controlling organization.

This can be done by having the third party generate a key pair that is designated uniquely for use by that trusted third party and publishing the public key in the controlling organization's DNS domain, thus enabling the third party to sign mail using the signature of the controlling organization. For example, if Company A outsources its employee benefits to a third party, it can use a special key pair that enables the benefits company to sign mail as "companyA.example". Because the key pair is unique to that trusted third party, it is easy for Company A to revoke the authorization if necessary by simply removing the public key from the companyA.example DNS.

A more cautious approach would be to create a dedicated subdomain (e.g., benefits.companyA.example) to segment the outsourced mail stream, and to publish the public key there; the signature would then use d=benefits.companyA.example.

6.4.1. DNS Delegation

Another possibility for configuring trusted third-party access, as discussed in Section 3.4, is to have Company A use DNS delegation and have the designated subdomain managed directly by the trusted third party. In this case, Company A would create a subdomain benefits.companya.example, and delegate the DNS management of that subdomain to the benefits company so it could maintain its own key records. When revocation becomes necessary, Company A could simply remove the DNS delegation record.

6.5. Multiple Signatures

A simple configuration for DKIM-signed mail is to have a single signature on a given message. This works well for domains that manage and send all of their own email from single sources, or for cases where multiple email streams exist but each has its own unique key pair. It also represents the case in which only one of the participants in an email sequence is able to sign, no matter whether it represents the author or one of the operators.

The examples thus far have considered the implications of using different identities in DKIM signatures, but have used only one such identity for any given message. In some cases, it can make sense to have more than one identity claiming responsibility for the same message.

There are a number of situations where applying more than one DKIM signature to the same message might make sense. A few examples are:

Companies with multiple subdomain identities: A company that has multiple subdomains sending distinct categories of mail might choose to sign with distinct subdomain identities to enable each subdomain to manage its own identity. However, it might also want to provide a common identity that cuts across all of the distinct subdomains. For example, Company A can sign mail for its sales department with a signature where d=sales.companya.example and a second signature where d=companya.example

Service Providers: A service provider can, as described above, choose to sign outbound messages with either its own identity or an identity unique to each of its clients (possibly delegated). However, it can also do both: sign each outbound message with its own identity as well as with the identity of each individual client. For example, ESP A might sign mail for its client Company B with its service provider signature d=espa.example, and a second client-specific signature where d= either companyb.example or companyb.espa.example. The existence of the service provider

signature could, for example, help cover a new client while it establishes its own reputation, or help a very small volume client who might never reach a volume threshold sufficient to establish an individual reputation.

Forwarders: Forwarded mail poses a number of challenges to email authentication. DKIM is relatively robust in the presence of forwarders as long as the signature is designed to avoid message parts that are likely to be modified; however, some forwarders do make modifications that can invalidate a DKIM signature.

Some forwarders such as mailing lists or "forward article to a friend" services might choose to add their own signatures to outbound messages to vouch for them having legitimately originated from the designated service. In this case, the signature would be added even in the presence of a preexisting signature, and both signatures would be relevant to the verifier.

Any forwarder that modifies messages in ways that will break preexisting DKIM signatures needs to sign its forwarded messages.

Reputation Providers: Although third-party reputation providers today use a variety of protocols to communicate their information to receivers, it is possible that they, or other organizations willing to put their "seal of approval" on an email stream, might choose to use a DKIM signature to do it. In nearly all cases, this "reputation" signature would be in addition to the author or originator signature.

One important caveat to the use of multiple signatures is that there is currently no clear consensus among receivers on how they plan to handle them. The opinions range from ignoring all but one signature (and the specification of which of them is verified differs from receiver to receiver), to verifying all signatures present and applying a weighted blend of the trust assessments for those identifiers, to verifying all signatures present and simply using the identifier that represents the most positive trust assessment. It is likely that the industry will evolve to accept multiple signatures using either the second or third of these, but it can take some time before one approach becomes pervasive.

7. Example Usage Scenarios

Signatures are created by different types of email actors, based on different criteria, such as where the actor operates in the sequence from author to recipient, whether they want different messages to be evaluated under the same reputation or a different one, and so on.

This section provides some examples of usage scenarios for DKIM deployments; the selection is not intended to be exhaustive but to illustrate a set of key deployment considerations.

7.1. Author's Organization - Simple

The simplest DKIM configuration is to have some mail from a given organization (Company A) be signed with the same d= value (e.g., d=companya.example). If there is a desire to associate additional information, the AUID [RFC5672] value can become uniqueID@companya.example, or @uniqueID.companya.example.

In this scenario, Company A need only generate a single signing key and publish it under their top-level domain (companya.example); the signing module would then tailor the AUID value as needed at signing time.

7.2. Author's Organization - Differentiated Types of Mail

A slight variation of the one signature case is where Company A signs some of its mail, but it wants to differentiate among categories of its outbound mail by using different identifiers. For example, it might choose to distinguish marketing, billing or transactional, and individual corporate email into marketing.companya.example, billing.companya.example, and companya.example, respectively, where each category is assigned a unique subdomain and unique signing keys.

7.3. Author Domain Signing Practices

7.3.1. Introduction

Some domains might decide to sign all of their outgoing mail. If all of the legitimate mail for a domain is signed, recipients can be more aggressive in their filtering of mail that uses the domain but does not have a valid signature from the domain; in such a configuration, the absence of a signature would be more significant than for the general case. It might be desirable for such domains to be able to advertise their intent to other receivers: this is the topic of Author Domain Signing Practices (ADSP).

Note that ADSP is not for everyone. Sending domains that do not control all legitimate outbound mail purporting to be from their domain (i.e., with an RFC5322.From address in their domain) are likely to experience delivery problems with some percentage of that mail. Administrators evaluating ADSP for their domains needs to carefully weigh the risk of phishing attacks against the likelihood of undelivered mail.

This section covers some examples of ADSP usage. For the complete specification, see [RFC5617].

7.3.2. A Few Definitions

In the ADSP specification, an address in the RFC5322.From header field of a message is defined as an "Author Address", and an "Author Domain" is defined as anything to the right of the '@' in an author address.

An "Author Signature" is thus any valid signature where the value of the SDID matches an author domain in the message.

It is important to note that unlike the DKIM specification, which makes no correlation between the signature domain and any message headers, the ADSP specification applies only to the author domain. In essence, under ADSP, any non-author signatures are ignored (treated as if they are not present).

Signers wishing to publish an Author Domain Signing Practices (ADSP) [RFC5617] record describing their signing practices will thus want to include an author signature on their outbound mail to avoid ADSP verification failures.

7.3.3. Some ADSP Examples

An organization (Company A) can specify its signing practices by publishing an ADSP record with "dkim=all" or "dkim=discardable". In order to avoid misdelivery of its mail at receivers that are validating ADSP, Company A needs to first have done an exhaustive analysis to determine all sources of outbound mail from its domain (companyA.example) and ensure that they all have valid author signatures from that domain.

For example, email with an RFC5322.From address of bob@companyA.example needs to have an author signature where the SDID value is "companyA.example" or it will fail an ADSP validation.

Note that once an organization publishes an ADSP record using dkim=all or dkim=discardable, any email with an RFC5322.From address that uses the domain where the ADSP record is published that does not have a valid author signature is at risk of being misdelivered or discarded. For example, if a message with an RFC5322.From address of newsletter@companyA.example has a signature with d=marketing.companyA.example, that message will fail the ADSP check because the signature would not be considered a valid author signature.

Because the semantics of an ADSP author signature are more constrained than the semantics of a "pure" DKIM signature, it is important to make sure the nuances are well understood before deploying an ADSP record. The ADSP specification [RFC5617] provides some fairly extensive lookup examples (in Appendix A) and usage examples (in Appendix B).

In particular, in order to prevent mail from being negatively impacted or even discarded at the receiver, it is essential to perform a thorough survey of outbound mail from a domain before publishing an ADSP policy of anything stronger than "unknown". This includes mail that might be sent from external sources that might not be authorized to use the domain signature, as well as mail that risks modification in transit that might invalidate an otherwise valid author signature (e.g., mailing lists, courtesy forwarders, and other paths that could add or modify headers or modify the message body).

7.4. Delegated Signing

An organization might choose to outsource certain key services to an independent company. For example, Company A might outsource its benefits management, or Organization B might outsource its marketing email.

If Company A wants to ensure that all of the mail sent on its behalf through the benefits providers email servers shares the Company A reputation, as discussed in Section 6.4, it can either publish keys designated for the use of the benefits provider under companyA.example (preferably under a designated subdomain of companyA.example), or it can delegate a subdomain (e.g., benefits.companyA.example) to the provider and enable the provider to generate the keys and manage the DNS for the designated subdomain.

In both of these cases, mail would be physically going out of the benefit provider’s mail servers with a signature of, e.g., d=benefits.companya.example. Note that the RFC5322.From address is not constrained: it could be affiliated with either the benefits company (e.g., benefits-admin@benefitprovider.example, or benefits-provider@benefits.companya.example) or the companyA domain.

Note that in both of the above scenarios, as discussed in Section 3.4, security concerns dictate that the keys be generated by the organization that plans to do the signing so that there is no need to transfer the private key. In other words, the benefits provider would generate keys for both of the above scenarios.

7.5. Independent Third-Party Service Providers

Another way to manage the service provider configuration would be to have the service provider sign the outgoing mail on behalf of its client, Company A, with its own (provider) identifier. For example, an Email Service Provider (ESP A) might want to share its own mailing reputation with its clients, and might sign all outgoing mail from its clients with its own d= domain (e.g., d=espa.example).

When the ESP wants to distinguish among its clients, it has two options:

- Share the SDID domain and use the AUID value to distinguish among the clients, e.g., a signature on behalf of client A would have d=espa.example and i=@clienta.espa.example (or i=clienta@espa.example).

- Extend the SDID domain, so there is a unique value (and subdomain) for each client, e.g., a signature on behalf of client A would have d=clienta.espa.example.

Note that this scenario and the delegation scenario are not mutually exclusive. In some cases, it can be desirable to sign the same message with both the ESP and the ESP client identities.

7.6. Mail Streams Based on Behavioral Assessment

An ISP (ISP A) might want to assign signatures to outbound mail from its users according to each user's past sending behavior (reputation). In other words, the ISP would segment its outbound traffic according to its own assessment of message quality, to aid recipients in differentiating among these different streams. Since the semantics of behavioral assessments are not valid AUID values, ISP A (ispa.example) can configure subdomains corresponding to the assessment categories (e.g., good.ispa.example, neutral.ispa.example, bad.ispa.example), and use these subdomains in the d= value of the signature.

The signing module can also set the AUID value to have a unique user ID (distinct from the local-part of the user's email address), for example, user3456@neutral.domain.example. Using a user ID that is distinct from a given email alias is useful in environments where a single user might register multiple email aliases.

Note that in this case, the AUID values are only partially stable. They are stable in the sense that a given i= value will always represent the same identity, but they are unstable in the sense that

a given user can migrate among the assessment subdomains depending on their sending behavior (i.e., the same user might have multiple AUID values over the lifetime of a single account).

In this scenario, ISP A can generate as many keys as there are assessment subdomains (SDID values), so that each assessment subdomain has its own key. The signing module would then choose its signing key based on the assessment of the user whose mail was being signed, and if desired, include the user ID in the AUID of the signature. As discussed earlier, the per-user granularity of the AUID can be ignored by verifiers; so organizations choosing to use it ought not rely on its use for receiver side filtering results. However, some organizations might also find the information useful for their own purposes in processing bounces or abuse reports.

7.7. Agent or Mediator Signatures

Another scenario is that of an agent, usually a re-mailer of some kind, that signs on behalf of the service or organization that it represents. Some examples of agents might be a mailing list manager, or the "forward article to a friend" service that many online publications offer. In most of these cases, the signature is asserting that the message originated with, or was relayed by, the service asserting responsibility. In general, if the service is configured in such a way that its forwarding would break existing DKIM signatures, it needs to always add its own signature.

8. Usage Considerations

8.1. Non-Standard Submission and Delivery Scenarios

The robustness of DKIM’s verification mechanism is based on the fact that only authorized signing modules have access to the designated private key. This has the side effect that email submission and delivery scenarios that originate or relay messages from outside the domain of the authorized signing module will not have access to that protected private key, and thus will be unable to attach the expected domain signature to those messages. Such scenarios include mailing lists, courtesy forwarders, MTAs at hotels, hotspot networks used by traveling users, and other paths that could add or modify headers, or modify the message body.

For example, assume Joe works for Company A and has an email address joe@companya.example. Joe also has an ISP-1 account joe@isp1.example.com, and he uses ISP-1’s multiple address feature to attach his work email address, joe@companya.example, to email from his ISP-1 account. When Joe sends email from his ISP-1 account and uses joe@companya.example as his designated RFC5322.From address,

that email cannot have a signature with d=companya.example because the ISP-1 servers have no access to Company A's private key. In ISP-1's case, it will have an ISP-1 signature, but for some other mail clients offering the same multiple address feature there might be no signature at all on the message.

Another example might be the use of a forward article to a friend service. Most instances of these services today allow someone to send an article with their email address in the RFC5322.From to their designated recipient. If Joe used either of his two addresses (joe@companya.example or joe@isp1.example.com), the forwarder would be equally unable to sign with a corresponding domain. As in the mail client case, the forwarder can either sign as its own domain or put no signature on the message.

A third example is the use of privately configured forwarding. Assume that Joe has another account at ISP-2, joe@isp-2.example.com, but he'd prefer to read his ISP-2 mail from his ISP-1 account. He sets up his ISP-2 account to forward all incoming mail to joe@isp1.example.com. Assume alice@companyb.example sends joe@isp-2.example.com an email. Depending on how companyb.example configured its signature, and depending on whether or not ISP-2 modifies messages that it forwards, it is possible that when Alice's message is received in Joe's ISP-1 account, the original signature will fail verification.

8.2. Protection of Internal Mail

One identity is particularly amenable to easy and accurate assessment: the organization's own identity. Members of an organization tend to trust messages that purport to be from within that organization. However, Internet Mail does not provide a straightforward means of determining whether such mail is, in fact, from within the organization. DKIM can be used to remedy this exposure. If the organization signs all of its mail, then its boundary MTAs can look for mail purporting to be from the organization that does not contain a verifiable signature.

Such mail can, in most cases, be presumed to be spurious. However, domain managers are advised to consider the ways that mail processing can modify messages in ways that will invalidate an existing DKIM signature: mailing lists, courtesy forwarders, and other paths that could add or modify headers or modify the message body (e.g., MTAs at hotels, hotspot networks used by traveling users, and other scenarios described in the previous section). Such breakage is particularly relevant in the presence of Author Domain Signing Practices.

8.3. Signature Granularity

Although DKIM's use of domain names is optimized for a scope of organization-level signing, it is possible to administer subdomains or otherwise adjust signatures in a way that supports per-user identification. This user-level granularity can be specified in two ways: either by sharing the signing identity and specifying an extension to the i= value that has a per-user granularity or by creating and signing with unique per-user keys.

A subdomain or local part in the i= tag needs to be treated as an opaque identifier and thus need not correspond directly to a DNS subdomain or be a specific user address.

The primary way to sign with per-user keys requires each user to have a distinct DNS (sub)domain, where each distinct d= value has a key published. (It is possible, although not advised, to publish the same key in more than one distinct domain.)

It is technically possible to publish per-user keys within a single domain or subdomain by utilizing different selector values. This is not advised and is unlikely to be treated uniquely by Assessors: the primary purpose of selectors is to facilitate key management, and the DKIM specification recommends against using them in determining or assessing identities.

In most cases, it would be impractical to sign email on a per-user granularity. Such an approach would be

likely to be ignored: In most cases today, if receivers are verifying DKIM signatures, they are in general taking the simplest possible approach. In many cases, maintaining reputation information at a per-user granularity is not interesting to them, in large part because the per-user volume is too small to be useful or interesting. So even if senders take on the complexity necessary to support per-user signatures, receivers are unlikely to retain anything more than the base domain reputation.

difficult to manage: Any scheme that involves maintenance of a significant number of public keys might require infrastructure enhancements or extensive administrative expertise. For domains of any size, maintaining a valid per-user keypair, knowing when keys need to be revoked or added due to user attrition or onboarding, and the overhead of having the signing engine constantly swapping keys can create significant and often unnecessary management complexity. It is also important to note

that there is no way within the scope of the DKIM specification for a receiver to infer that a sender intends a per-user granularity.

As mentioned before, what might make sense, however, is to use the infrastructure that enables finer granularity in signatures to identify segments smaller than a domain but much larger than a per-user segmentation. For example, a university might want to segment student, staff, and faculty mail into three distinct streams with differing reputations. This can be done by creating separate subdomains for the desired segments, and either specifying the subdomains in the i= tag of the DKIM Signature or by adding subdomains to the d= tag and assigning and signing with different keys for each subdomain.

For those who choose to represent user-level granularity in signatures, the performance and management considerations above suggest that it would be more effective to do so by specifying a local part or subdomain extension in the i= tag rather than by extending the d= domain and publishing individual keys.

8.4. Email Infrastructure Agents

It is expected that the most common venue for a DKIM implementation will be within the infrastructure of an organization's email service, such as a department or a boundary MTA. What follows are some general recommendations for the Email Infrastructure.

Outbound: An MSA (Mail Submission Agent) or an outbound MTA used for mail submission needs to ensure that the message sent is in compliance with the advertised email sending policy. It needs to also be able to generate an operator alert if it determines that the email messages do not comply with the published DKIM sending policy.

An MSA needs to be aware that some MUAs might add their own signatures. If the MSA needs to perform operations on a message to make it comply with its email sending policy, if at all possible, it needs to do so in a way that would not break those signatures.

MUAs equipped with the ability to sign ought not to be encouraged. In terms of security, MUAs are generally not under the direct control of those in responsible roles within an organization and are thus more vulnerable to attack and compromise, which would expose private signing keys to intruders and thus jeopardize the integrity and reputation of the organization.

Inbound: When an organization deploys DKIM, it needs to make sure that its email infrastructure components that do not have primary roles in DKIM handling do not modify message in ways that prevent subsequent verification.

An inbound MTA or an MDA can incorporate an indication of the verification results into the message, such as using an Authentication-Results header field [RFC5451].

Intermediaries: An email intermediary is both an inbound and outbound MTA. Each of the requirements outlined in the sections relating to MTAs apply. If the intermediary modifies a message in a way that breaks the signature, the intermediary.

+ needs to deploy abuse filtering measures on the inbound mail, and

+ probably also needs to remove all signatures that will be broken.

In addition, the intermediary can:

+ verify the message signature prior to modification.

+ incorporate an indication of the verification results into the message, such as using an Authentication-Results header field [RFC5451].

+ sign the modified message including the verification results (e.g., the Authentication-Results header field).

8.5. Mail User Agent

The DKIM specification is expected to be used primarily between Boundary MTAs, or other infrastructure components of the originating and receiving ADMDs. However, there is nothing in DKIM that is specific to those venues. In particular, MUAs can also support DKIM signing and verifying directly.

Outbound: An MUA can support signing even if mail is to be relayed through an outbound MSA. In this case, the signature applied by the MUA will be in addition to any signature added by the MSA. However, the warnings in the previous section need to be taken into consideration.

Some user software goes beyond simple user functionality and also performs MSA and MTA functions. When this is employed for sending directly to a receiving ADMD, the user software needs to be considered an outbound MTA.

Inbound: An MUA can rely on a report of a DKIM signature verification that took place at some point in the inbound MTA/MDA path (e.g., an Authentication-Results header field), or an MUA can perform DKIM signature verification directly. A verifying MUA needs to allow for the case where mail has been modified in the inbound MTA path; if a signature fails, the message is to be treated the same as a message that does not have a signature.

An MUA that looks for an Authentication-Results header field needs to be configurable to choose which Authentication-Results header fields are considered trustable. The MUA developer is encouraged to re-read the Security Considerations of [RFC5451].

DKIM requires that all verifiers treat messages with signatures that do not verify as if they are unsigned.

If verification in the client is to be acceptable to users, it is essential that successful verification of a signature not result in a less than satisfactory user experience compared to leaving the message unsigned. The mere presence of a verified DKIM signature cannot be used by itself by an MUA to indicate that a message is to be treated better than a message without a verified DKIM signature. However, the fact that a DKIM signature was verified can be used as input into a reputation system (i.e., a whitelist of domains and users) for presentation of such indicators.

It is common for components of an ADMD's email infrastructure to do violence to a message, such that a DKIM signature might be rendered invalid. Hence, users of MUAs that support DKIM signing and/or verifying need a basis for knowing that their associated email infrastructure will not break a signature.

9. Security Considerations

The security considerations of the DKIM protocol are described in the DKIM base specification [RFC4871].

10. Acknowledgements

The effort of the DKIM Working Group is gratefully acknowledged.

11. References

11.1. Normative References

[RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC 4871, May 2007.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[RFC5451] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", RFC 5451, April 2009.

[RFC5585] Hansen, T., Crocker, D., and P. Hallam-Baker, "DomainKeys Identified Mail (DKIM) Service Overview", RFC 5585, July 2009.

[RFC5617] Allman, E., Fenton, J., Delany, M., and J. Levine, "DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP)", RFC 5617, August 2009.

[RFC5672] Crocker, D., "RFC 4871 DomainKeys Identified Mail (DKIM) Signatures -- Update", RFC 5672, August 2009.

11.2. Informative References

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4870] Delany, M., "Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys)", RFC 4870, May 2007.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

Appendix A. Migration Strategies

There are three migration occasions worth noting in particular for DKIM:

1. Migrating from DomainKeys to DKIM.

2. Migrating from a current hash algorithm to a new standardized hash algorithm.

3. Migrating from a current signing algorithm to a new standardized signing algorithm.

The case of deploying a new key selector record is described elsewhere (Section 3.5).

As with any migration, the steps required will be determined by who is doing the migration and their assessment of:

- the users of what they are generating, or

- the providers of what they are consuming.

Signers and verifiers have different considerations.

A.1. Migrating from DomainKeys

DKIM replaces the earlier DomainKeys (DK) specification. Selector files are mostly compatible between the two specifications.

A.1.1. Signers

A signer that currently signs with DK will go through various stages as it migrates to using DKIM, not all of which are required for all signers. The real questions that a signer needs to ask are:

1. how many receivers or what types of receivers are *only* looking at the DK signatures and not the DKIM signatures, and

2. how much does the signer care about those receivers?

If no one is looking at the DK signature any more, then it’s no longer necessary to sign with DK. Or if all "large players" are looking at DKIM in addition to or instead of DK, a signer can choose to stop signing with DK.

With respect to signing policies, a reasonable, initial approach is to use DKIM signatures in the same way that DomainKeys signatures are already being used. In particular, the same selectors and DNS key records can be used for both, after verifying that they are compatible as discussed below.

Each secondary step in all of the following scenarios is to be prefaced with the gating factor "test, then when comfortable with the previous step's results, continue".

One migration strategy is to:

- ensure that the current selector DNS key record is compatible with both DK and DKIM
- sign messages with both DK and DKIM signatures
- when it's decided that DK signatures are no longer necessary, stop signing with DK

Another migration strategy is to:

- add a new selector DNS key record only for DKIM signatures
- sign messages with both DK (using the old DNS key record) and DKIM signatures (using the new DNS key record)
- when it's decided that DK signatures are no longer necessary, stop signing with DK
- eventually remove the old DK selector DNS record

A combined migration strategy is to:

- ensure that the current selector DNS key record is compatible with both DK and DKIM
- start signing messages with both DK and DKIM signatures
- add a new selector DNS key record for DKIM signatures
- switch the DKIM signatures to use the new selector
- when it's decided that DK signatures are no longer necessary, stop signing with DK
- eventually remove the old DK selector DNS record

Another migration strategy is to:

- add a new selector DNS key record for DKIM signatures
- do a flash cut and replace the DK signatures with DKIM signatures
- eventually remove the old DK selector DNS record

Another migration strategy is to:

- ensure that the current selector DNS key record is compatible with both DK and DKIM
- do a flash cut and replace the DK signatures with DKIM signatures

Note that when you have separate key records for DK and DKIM, you can use the same public key for both.

A.1.1.1. DNS Selector Key Records

The first step in some of the above scenarios is ensuring that the selector DNS key records are compatible for both DK and DKIM. The format of the DNS key record was intentionally meant to be backwardly compatible between the two systems, but not necessarily upwardly compatible. DKIM has enhanced the DK DNS key record format by adding several optional parameters, which DK needs to ignore. However, there is one critical difference between DK and DKIM DNS key records. The definitions of the "g" fields:

g= granularity of the key: In both DK and DKIM, this is an optional field that is used to constrain which sending address(es) can legitimately use this selector. Unfortunately, the treatment of an empty field ("g=;") is different. DKIM allows wildcards where DK does not. For DK, an empty field is the same as a missing value, and is treated as allowing any sending address. For DKIM, an empty field only matches an empty local part. In DKIM, both a missing value and "g=*;" mean to allow any sending address.

Also, in DomainKeys, the "g" field is required to match the address in "From:"/"Sender:", while in DKIM, it is required to match i=. This might or might not affect transition.

If your DK DNS key record has an empty "g" field in it ("g=;"), your best course of action is to modify the record to remove the empty field. In that way, the DK semantics will remain the same, and the DKIM semantics will match.

If your DNS key record does not have an empty "g" field in it ("g=;"), it's probable that the record can be left alone. But the best course of action would still be to make sure that it has a "v" field. When the decision is made to stop supporting DomainKeys and to only support DKIM, it is important to verify that the "g" field is compatible with DKIM, and typically having "v=DKIM1;" in it. It is strongly encouraged that if use of an empty "g" field in the DKIM selector, include the "v" field.

A.1.1.2. Removing DomainKeys Signatures

The principal use of DomainKeys is at boundary MTAs. Because no operational transition is ever instantaneous, it is advisable to continue performing DomainKeys signing until it is determined that DomainKeys receive-side support is no longer used, or is sufficiently reduced. That is, a signer needs to add a DKIM signature to a message that also has a DomainKeys signature and keep it there until they decide it is deemed no longer useful. The signer can do its transitions in a straightforward manner, or more gradually. Note that because digital signatures are not free, there is a cost to performing both signing algorithms, so signing with both algorithms ought not be needlessly prolonged.

The tricky part is deciding when DK signatures are no longer necessary. The real questions are: how many DomainKeys verifiers are there that do *not* also do DKIM verification, which of those are important, and how can you track their usage? Most of the early adopters of DK verification have added DKIM verification, but not all yet. If a verifier finds a message with both DK and DKIM, it can choose to verify both signatures, or just one or the other.

Many DNS services offer tracking statistics so it can be determined how often a DNS record has been accessed. By using separate DNS selector key records for your signatures, you can chart the use of your records over time, and watch the trends. An additional distinguishing factor to track would take into account the verifiers that verify both the DK and DKIM signatures, and discount those from counts of DK selector usage. When the number for DK selector access reaches a low-enough level, that's the time to consider discontinuing signing with DK.

Note, this level of rigor is not required. It is perfectly reasonable for a DK signer to decide to follow the "flash cut" scenario described above.

A.1.2. Verifiers

As a verifier, several issues need to be considered:

A.1.2.1. Ought DK signature verification be performed?

At the time of writing, there is still a significant number of sites that are only producing DK signatures. Over time, it is expected that this number will go to zero, but it might take several years. So it would be prudent for the foreseeable future for a verifier to look for and verify both DKIM and DK signatures.

A.1.2.2. Ought both DK and DKIM signatures be evaluated on a single message?

For a period of time, there will be sites that sign with both DK and DKIM. A verifier receiving a message that has both types of signatures can verify both signatures, or just one. One disadvantage of verifying both signatures is that signers will have a more difficult time deciding how many verifiers are still using their DK selectors. One transition strategy is to verify the DKIM signature, then only verify the DK signature if the DKIM verification fails.

A.1.2.3. DNS Selector Key Records

The format of the DNS key record was intentionally meant to be backwardly compatible between DK and DKIM, but not necessarily upwardly compatible. DKIM has enhanced the DK DNS key record format by adding several optional parameters, which DK needs to ignore. However, there is one key difference between DK and DKIM DNS key records. The definitions of the g fields:

g= granularity of the key: In both DK and DKIM, this is an optional field that is used to constrain which sending address(es) can legitimately use this selector. Unfortunately, the treatment of an empty field ("g=;") is different. For DK, an empty field is the same as a missing value, and is treated as allowing any sending address. For DKIM, an empty field only matches an empty local part.

v= version of the selector It is advised that a DKIM selector have "v=DKIM1;" at its beginning, but it is not required.

If a DKIM verifier finds a selector record that has an empty "g" field ("g=;") and it does not have a "v" field ("v=DKIM1;") at its beginning, it is faced with deciding if this record was:

1. from a DK signer that transitioned to supporting DKIM but forgot to remove the "g" field (so that it could be used by both DK and DKIM verifiers); or

2. from a DKIM signer that truly meant to use the empty "g" field but forgot to put in the "v" field. It is advised that you treat such records using the first interpretation, and treat such records as if the signer did not have a "g" field in the record.

A.2. Migrating Hash Algorithms

[RFC4871] defines the use of two hash algorithms: SHA-1 and SHA-256. The security of all hash algorithms is constantly under attack, and SHA-1 has already shown weaknesses as of this writing. Migrating from SHA-1 to SHA-256 is not an issue, because all verifiers are already required to support SHA-256. But when it becomes necessary to replace SHA-256 with a more secure algorithm, there will be a migratory period. In the following, "NEWHASH" is used to represent a new hash algorithm. Section 4.1 of [RFC4871] briefly discusses this scenario.

A.2.1. Signers

As with migrating from DK to DKIM, migrating hash algorithms is dependent on the signer’s best guess as to the utility of continuing to sign with the older algorithms and the expected support for the newer algorithm by verifiers. The utility of continuing to sign with the older algorithms is also based on how broken the existing hash algorithms are considered and how important that is to the signers.

One strategy is to wait until it’s determined that there is a large enough base of verifiers available that support NEWHASH, and then flash cut to the new algorithm.

Another strategy is to sign with both the old and new hash algorithms for a period of time. This is particularly useful for testing the new code to support the new hash algorithm, as verifiers will continue to accept the signature for the older hash algorithm and ought to ignore any signature that fails because the code is slightly wrong. Once the signer has determined that the new code is correct AND it’s determined that there is a large enough base of verifiers available that support NEWHASH, the signer can flash cut to the new algorithm.

One advantage migrating hash algorithms has is that the selector can be completely compatible for all hash algorithms. The key selector has an optional "h=" field that can be used to list the hash algorithms being used; it also is used to limit the algorithms that a

verifier will accept. If the signer is not currently using the key selector "h=" field, no change is required. If the signer is currently using the key selector "h=" field, NEWHASH will need to be added to the list, as in "h=sha256:NEWHASH;". (When the signer is no longer using SHA-256, it can be removed from the "h=" list.)

A.2.2. Verifiers

When a new hash algorithm becomes standardized, it is best for a verifier to start supporting it as quickly as possible.

A.3. Migrating Signing Algorithms

[RFC4871] defines the use of the RSA signing algorithm. Similar to hashes, signing algorithms are constantly under attack, and when it becomes necessary to replace RSA with a newer signing algorithm, there will be a migratory period. In the following, "NEWALG" is used to represent a new signing algorithm.

A.3.1. Signers

As with the other migration issues discussed above, migrating signing algorithms is dependent on the signer’s best guess as to the utility of continuing to sign with the older algorithms and the expected support for the newer algorithm by verifiers. The utility of continuing to sign with the older algorithms is also based on how broken the existing signing algorithms are considered and how important that is to the signers.

As before, the two basic strategies are to 1) wait until there is sufficient base of verifiers available that support NEWALG and then do a flash cut to NEWALG, and 2) use a phased approach by signing with both the old and new algorithms before removing support for the old algorithm.

It is unlikely that a new algorithm would be able to use the same public key as "rsa", so using the same selector DNS record for both algorithms’ keys is ruled out. Therefore, in order to use the new algorithm, a new DNS selector record would need to be deployed in parallel with the existing DNS selector record for the existing algorithm. The new DNS selector record would specify a different "k=" value to reflect the use of NEWALG.

A.3.2. Verifiers

When a new hash algorithm becomes standardized, it is best for a verifier to start supporting it as quickly as possible.

Appendix B. General Coding Criteria for Cryptographic Applications

NOTE: This section could possibly be changed into a reference to something else, such as another RFC.

Correct implementation of a cryptographic algorithm is a necessary but not a sufficient condition for the coding of cryptographic applications. Coding of cryptographic libraries requires close attention to security considerations that are unique to cryptographic applications.

In addition to the usual security coding considerations, such as avoiding buffer or integer overflow and underflow, implementers need to pay close attention to management of cryptographic private keys and session keys, ensuring that these are correctly initialized and disposed of.

Operating system mechanisms that permit the confidentiality of private keys to be protected against other processes ought to be used when available. In particular, great care needs to be taken when releasing memory pages to the operating system to ensure that private key information is not disclosed to other processes.

Certain implementations of public key algorithms such as RSA can be vulnerable to a timing analysis attack.

Support for cryptographic hardware providing key management capabilities is strongly encouraged. In addition to offering performance benefits, many cryptographic hardware devices provide robust and verifiable management of private keys.

Fortunately, appropriately designed and coded cryptographic libraries are available for most operating system platforms under license terms compatible with commercial, open source and free software license terms. Use of standard cryptographic libraries is strongly encouraged. These have been extensively tested, reduce development time and support a wide range of cryptographic hardware.

Authors' Addresses

Tony Hansen
AT&T Laboratories
200 Laurel Ave. South
Middletown, NJ 07748
USA

EMail: tony+dkimov@maillennium.att.com

Ellen Siegel
Consultant

EMail: dkim@esiegel.net

Phillip Hallam-Baker
Default Deny Security, Inc.

EMail: phillip@hallambaker.com

Dave Crocker
Brandenburg InternetWorking
675 Spruce Dr.
Sunnyvale, CA 94086
USA

EMail: dcrocker@bbiw.net