Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

W. Chuang
Google

7 July 2025

Domain Name Specification for DKIM2
draft-chuang-dkim2-dns-02

Abstract

The updated DomainKeys Identified Mail (DKIM2) permits an organization that owns the signing domain to claim some responsibility for a message by associating the domain with the message through a digital signature. This is done by publishing to Domain Name Service (DNS) of the domain a public key that is then associated to the domain and where messages can be signed by the corresponding private key. Assertion of responsibility is validated through a cryptographic signature and by querying the Signer's domain directly to retrieve the appropriate public key. This document describes DKIM2 DNS record format and how to find the record.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 January 2026.

Table of Contents

1. Introduction 2
2. Terminology and Definitions 3
2.1. Signers 3
2.2. Verifiers 3
2.3. Identity 4
2.4. Identifier 4
2.5. Signing Domain Identifier (SDID) 4
2.6. Identity Assessor 4
2.7. Whitespace 4
2.8. Imported ABNF Tokens 5
2.9. Common ABNF Tokens 5
3. Protocol Elements 5
3.1. Selectors 5
3.2. Tag=Value Lists 7
3.3. Signing and Verification Algorithms 8
3.4. Key Management and Representation 8
3.4.1. Textual Representation 9
3.4.2. DNS Binding 11
4. IANA Considerations 12
5. Security Considerations 12
6. Normative References 12
Appendix A. Acknowledgments 13
Author's Address 13

1. Introduction

The updated DomainKeys Identified Mail (DKIM2) permits an organization that owns the signing domain to claim some responsibility for a message [RFC5322] by associating the domain with the message through a digital signature. This is done by publishing to Domain Name Service (DNS) of the domain a public key that is then associated to the domain [RFC1034] and where messages can be signed by the corresponding private key. Assertion of responsibility is validated through a cryptographic signature and by querying the Signer' s domain directly to retrieve the appropriate public key.

This document describes DKIM2 DNS record format and how to find the record. The updated DomainKeys Identified Mail (DKIM2) adopts a subset of the DKIM [RFC6376] DNS specification, but to prevent ambiguity from using the using normative references to RFC6376, this document copies over the RFC6376 and updates as necessary. However,

this document still does normatively reference RFC6376 for the IANA registrations. This document and the other DKIM2 documents do not deprecate RFC6376 as it is expected that both DKIM and DKIM2 will co-exist for at least some period of time.

INFORMATIVE NOTE: another reason for separating the DKIM2 DNS specification from DKIM, is that in the future, it is likely that DKIM2 will diverge from the parent specification.

There are other updated DomainKeys Identified Mail (DKIM2) documents as well: the first document describes the motivation; the second document describes the headers.

2. Terminology and Definitions

This section defines terms used in the rest of the document. DKIM2 is designed to operate within the Internet Mail service, as defined in [RFC5598]. Basic email terminology is taken from that specification.

Syntax descriptions use Augmented BNF (ABNF) [RFC5234]. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words take their normative meanings only when they are presented in ALL UPPERCASE.

2.1. Signers

Elements in the mail system that sign messages on behalf of a domain are referred to as Signers. These may be MUAs (Mail User Agents), MSAs (Mail Submission Agents), MTAs (Mail Transfer Agents), or other agents such as mailing list exploders. In general, any Signer will be involved in the injection of a message into the message system in some way. The key issue is that a message must be signed before it leaves the administrative domain of the Signer.

2.2. Verifiers

Elements in the mail system that verify signatures are referred to as Verifiers. These may be MTAs, Mail Delivery Agents (MDAs), or MUAs. In most cases, it is expected that Verifiers will be close to an end user (reader) of the message or some consuming agent such as a mailing list exploder.

2.3. Identity

This specification DKIM2 calls for the Identity to correspond to an organization while DKIM permitted person, or role as well. In the context of email architecture specification [RFC5598], this corresponds to an ADministrative Management Domain (ADMD). Some examples include the the author's organization, an ISP along the handling path, an independent trust assessment service, and a mailing list operator.

2.4. Identifier

A label that refers to an identity.

2.5. Signing Domain Identifier (SDID)

A single domain name that is the mandatory payload output of DKIM and that refers to the identity claiming some responsibility

2.6. Identity Assessor

An element in the mail system that consumes DKIM2's payload, which is the responsible Signing Domain Identifier (SDID). The Identity Assessor is dedicated to the assessment of the delivered identifier. Other DKIM2 (and non-DKIM2) values can also be used by the Identity Assessor (if they are available) to provide a more general message evaluation filtering engine. However, this additional activity is outside the scope of this specification.

2.7. Whitespace

There are three forms of whitespace:

* WSP represents simple whitespace, i.e., a space or a tab character (formal definition in [RFC5234])

* LWSP is linear whitespace, defined as WSP plus CRLF (formal definition in [RFC5234]).

* FWS is folding whitespace. It allows multiple lines separated by CRLF followed by at least one whitespace, to be joined.

The formal ABNF for these are (WSP and LWSP are given for information only):

```
WSP = SP / HTAB
LWSP = *(WSP / CRLF WSP)
FWS = [*WSP CRLF] 1*WSP
```

The definition of FWS is identical to that in [RFC5322] except for the exclusion of obs-FWS.

2.8. Imported ABNF Tokens

The following tokens are imported from other RFCs as noted. Those RFCs should be connsidered definitive.

The following tokens are imported from [RFC2045]:

* "qp-section" (a single line of quoted-printable-encoded text)

Tokens not defined herein are imported from [RFC5234]. These are intuitive primitives such as SP, HTAB, WSP, ALPHA, DIGIT, CRLF, etc.

2.9. Common ABNF Tokens

The following ABNF tokens are used elsewhere in this document:

```
hyphenated-word = ALPHA [ *(ALPHA / DIGIT / "-") (ALPHA / DIGIT) ]
ALPHADIGITPS = (ALPHA / DIGIT / "+" / "/")
base64string = ALPHADIGITPS *([FWS] ALPHADIGITPS)
[ [FWS] "=" [ [FWS] "=" ] ]
```

3. Protocol Elements

Protocol Elements are conceptual parts of the protocol that are not specific to either Signers or Verifiers. The protocol descriptions for Signers and Verifiers will be described in a separate document. NOTE: This section must be read in the context of those elements.

3.1. Selectors

To support multiple concurrent public keys per signing domain, the key namespace is subdivided using "selectors". For example, selectors might indicate the names of office locations (e.g., "sanfrancisco", "coolumbeach", and "reykjavik"), the signing date (e.g., "january2005", "february2005", etc.), or even an individual user.

Selectors are needed to support some important use cases. For example:

* Domains that want to delegate signing capability for a specific address for a given duration to a partner, such as an advertising provider or other outsourced function.

* Domains that want to allow frequent travelers to send messages locally without the need to connect with a particular MSA.

* "Affinity" domains (e.g., college alumni associations) that provide forwarding of incoming mail, but that do not operate a mail submission agent for outgoing mail.

Periods are allowed in selectors and are component separators. When keys are retrieved from the DNS, periods in selectors define DNS label boundaries in a manner similar to the conventional use in domain names. Selector components might be used to combine dates with locations, for example, "march2005.reykjavik". In a DNS implementation, this can be used to allow delegation of a portion of he selector namespace.

ABNF:
selector = sub-domain *("." sub-domain)

The number of public keys and corresponding selectors for each domain is determined by the domain owner. Many domain owners will be satisfied with just one selector, whereas administratively distributed organizations can choose to manage disparate selectors and key pairs in different regions or on different email servers.

Beyond administrative convenience, selectors make it possible to seamlessly replace public keys on a routine basis. If a domain wishes to change from using a public key associated with selector "january2005" to a public key associated with selector "february2005", it merely makes sure that both public keys are advertised in the public-key repository concurrently for the transition period during which email may be in transit prior to verification. At the start of the transition period, the outbound email servers are configured to sign with the "february2005" private key. At the end of the transition period, the "january2005" public key is removed from the public-key repository.

INFORMATIVE NOTE: A key may also be revoked as described below. The distinction between revoking and removing a key selector record is subtle. When phasing out keys as described above, a signing domain would probably simply remove the key record after the transition period. However, a signing domain could elect to revoke the key (but maintain the key record) for a further period. There is no defined semantic difference between a revoked key and a removed key.

While some domains may wish to make selector values well-known, others will want to take care not to allocate selector names in a way that allows harvesting of data by outside parties.

INFORMATIVE OPERATIONS NOTE: Reusing a selector with a new key (for example, changing the key associated with a user's name) makes it impossible to tell the difference between a message that didn't verify because the key is no longer valid and a message that is actually forged. For this reason, Signers are ill-advised to reuse selectors for new keys. A better strategy is to assign new keys to new selectors.

3.2. Tag=Value Lists

DKIM2 uses a simple "tag=value" syntax in several contexts, including in messages and domain signature records. Values are a series of strings containing either plain text, "base64" text (as defined in [RFC2045], Section 6.8), or "qp-section" (ibid, Section 6.7). The name of the tag will determine the encoding of each value. Unencoded semicolon (";") characters MUST NOT occur in the tag value, since that separates tag-specs.

INFORMATIVE IMPLEMENTATION NOTE: Although the "plain text" defined below (as "tag-value") only includes 7-bit characters, an implementation that wished to anticipate future standards would be advised not to preclude the use of UTF-8-encoded [RFC3629] text in tag=value lists.

Formally, the ABNF syntax rules are as follows:

```
tag-list = tag-spec *( ";" tag-spec ) [ ";" ]
tag-spec = [FWS] tag-name [FWS] "=" [FWS] tag-value [FWS]
tag-name = ALPHA *ALNUMPUNC
tag-value = [ tval *( 1*(WSP / FWS) tval ) ]
; Prohibits WSP and FWS at beginning and end
tval = 1*VALCHAR
VALCHAR = %x21-3A / %x3C-7E
; EXCLAMATION to TILDE except SEMICOLON
ALNUMPUNC = ALPHA / DIGIT / "_"
```

Note that WSP is allowed anywhere around tags. In particular, any WSP after the "=" and any WSP before the terminating ";" is not part of the value; however, WSP inside the value is signicant.

MUST be interpreted in a case-sensitive manner. Values MUST be processed as case sensitive unless the specific tag description of semantics specifies case insensitivity.

Tags with duplicate names MUST NOT occur within a single tag-list; if a tag name does occur more than once, the entire tag-list is invalid.

Whitespace within a value MUST be retained unless explicitly excluded by the specific tag description.

Tag=value pairs that represent the default value MAY be included to aid legibility.

Unrecognized tags MUST be ignored.

Tags that have an empty value are not the same as omitted tags. An omitted tag is treated as having the default value; a tag with an empty value explicitly designates the empty string as the value.

3.3. Signing and Verification Algorithms

DKIM2 supports multiple digital signature algorithms. Two algorithms are referenced by this specification at this time: rsa-sha256 and ed25519-sha256. rsa-sha256 is specified in [RFC6376] while ed25519-sha256 is specified in [RFC8463]. Signers and Verifiers MUST implement rsa-sha256 and SHOULD implement ed25519-sha256. Futher they MUST NOT validate the legacy DKIM [RFC6376] rsa-sha1 algorithm i.e. ignore the public key and signatures associated with it.

INFORMATIVE NOTE: rsa-sha256 enjoys virtually universal deployment while ed25519-sha256 has very little deployment at the creation of this draft. It does provide better capability in reduced key size, and faster signing and verification speed. Support for at least two algorithms provides algorithmic diversity that provides defense against some unforeseen future algorithm breakage. Thus implementation of ed25519-sha256 will be changed to MUST upon the completion of the DKIM2 specification. rsa-sha1 algorithm is deprecated for DKIM2 as sha1 is demonstrateably broken.

3.4. Key Management and Representation

Signature applications require some level of assurance that the verification public key is associated with the claimed Signer. Many applications achieve this by using public-key certificates issued by a trusted third party. However, DKIM2 can achieve a sufficient level of security, with significantly enhanced scalability, by simply having the Verifier query the purported Signer’ s DNS entry (or some security-equivalent) in order to retrieve the public key. DKIM2 keys can potentially be stored in multiple types of key servers and in multiple formats. The storage and format of keys are irrelevant to the remainder of the DKIM2 algorithm. Parameters to the key lookup algorithm are the type of the lookup (the "q=" tag), the domain of the Signer (the "d=" tag of the DKIM2 signature header field), and the selector (the "s=" tag).

```
public_key = dkim2_find_key(q_val, d_val, s_val)
```

This document defines a single binding, using DNS TXT RRs to distribute the keys. Other bindings may be defined in the future.

3.4.1. Textual Representation

It is expected that many key servers will choose to present the keys in an otherwise unstructured text format (for example, an XML form would not be considered to be unstructured text for this purpose). The following definition MUST be used for any DKIM2 key represented in an otherwise unstructured textual form.

The overall syntax is a tag-list as described in Section 3.2. The current valid tags are described below. Other tags MAY be present and MUST be ignored by any implementation that does not understand them.

v= Version of the DKIM key record (plain-text; RECOMMENDED, default is "DKIM1"). If specified, this tag MUST be set to "DKIM1" (without the quotes). This tag MUST be the first tag in the record. Records beginning with a "v=" tag with any other value MUST be discarded. Note that Verifiers must do a string comparison on this value; for example, "DKIM1" is not the same as "DKIM1.0".

INFORMATIVE NOTE: DKIM2 reuses a subset of the DKIM textual representation including this version number for compatibility with existing key deployment. Consequently DKIM2 does not increment the version number.

ABNF:

```
key-v-tag = %x76 [FWS] "=" [FWS] %x44.4B.49.4D.31
```

h= Acceptable hash algorithms (plain-text; OPTIONAL, defaults to allowing all algorithms). A colon-separated list of hash algorithms that might be used. Unrecognized algorithms MUST be ignored. Refer to Section 3.3 for a discussion of the hash algorithms implemented by Signers and Verifiers. The set of algorithms listed in this tag in each record is an operational choice made by the Signer.

ABNF:

```
key-h-tag = %x68 [FWS] "=" [FWS] key-h-tag-alg
*( [FWS] ":" [FWS] key-h-tag-alg )
key-h-tag-alg = "sha1" / "sha256" / x-key-h-tag-alg
x-key-h-tag-alg = hyphenated-word ; for future extension
```

k= Key type (plain-text; OPTIONAL, default is "rsa"). Signers and Verifiers MUST support the "rsa" key type. The "rsa" key type indicates that an ASN.1 DER-encoded [ITU-X660-1997] RSAPublicKey (see [RFC8017], Sections 3.1 and A.1.1) is being used in the "p=" tag. (Note: the "p=" tag further encodes the value using the base64 algorithm.) Unrecognized key types MUST be ignored.

ABNF:
key-k-tag = %x76 [FWS] "=" [FWS] key-k-tag-type
key-k-tag-type = "rsa" / x-key-k-tag-type
x-key-k-tag-type = hyphenated-word ; for future extension

n= Notes that might be of interest to a human (qp-section; OPTIONAL, default is empty). No interpretation is made by any program. This tag should be used sparingly in any key server mechanism that has space limitations (notably DNS). This is intended for use by administrators, not end users.

ABNF:
key-n-tag = %x6e [FWS] "=" [FWS] qp-section

p= Public-key data (base64; REQUIRED). An empty value means that this public key has been revoked. The syntax and semantics of this tag value before being encoded in base64 are defined by the "k=" tag.

INFORMATIVE RATIONALE: If a private key has been compromised or otherwise disabled (e.g., an outsourcing contract has been terminated), a Signer might want to explicitly state that it knows about the selector, but all messages using that selector

ABNF:
key-p-tag = %x70 [FWS] "=" [[FWS] base64string]

INFORMATIVE NOTE: A base64string is permitted to include whitespace (FWS) at arbitrary places; however, any CRLFs must be followed by at least one WSP character. Implementers and administrators are cautioned to ensure that selector TXT RRs conform to this specification.

s= Service Type (plain-text; OPTIONAL; default is "*"). A colon-separated list of service types to which this record applies. Verifiers for a given service type MUST ignore this record if the appropriate type is not listed. Unrecognized service types MUST be ignored. Currently defined service types are as follows:

* matches all service types

email electronic mail (not necessarily limited to SMTP)

This tag is intended to constrain the use of keys for other purposes, should use of DKIM2 be defined by other services in the future.

ABNF:

```
key-s-tag = %x73 [FWS] "=" [FWS] key-s-tag-type
*( [FWS] ":" [FWS] key-s-tag-type )
key-s-tag-type = "email" / "*" / x-key-s-tag-type
x-key-s-tag-type = hyphenated-word ; for future extension
```

t= Flags, represented as a colon-separated list of names (plain-text; OPTIONAL, default is no flags set). Unrecognized flags MUST be ignored. The defined flags are as follows:

y This domain is testing DKIM2. Verifiers MUST NOT treat messages from Signers in testing mode differently from unsigned email, even should the signature fail to verify. Verifiers MAY wish to track testing mode results to assist the Signer.

s Any DKIM2 signature header fields using the "i=" tag MUST have the same domain value on the right-hand side of the "@" in the "i=" tag and the value of the "d=" tag. That is, the "i=" domain MUST NOT be a subdomain of "d=". Use of this flag is

RECOMMENDED unless subdomaining is required.

ABNF:

```
key-t-tag = %x74 [FWS] "=" [FWS] key-t-tag-flag
*( [FWS] ":" [FWS] key-t-tag-flag )
key-t-tag-flag = "y" / "s" / x-key-t-tag-flag
x-key-t-tag-flag = hyphenated-word ; for future extension
```

3.4.2. DNS Binding

A binding using DNS TXT RRs as a key service is hereby defined. All implementations MUST support this binding.

3.4.2.1. Namespace

All DKIM2 keys are stored in a subdomain named "_domainkey". Given a DKIM2 signature header field with a "d=" tag of "example.com" and an "s=" tag of "foo.bar", the DNS query will be for "foo.bar._domainkey.example.com".

3.4.2.2. Resource Record Types for Key Storage

The DNS Resource Record type used is specified by an option to the query-type ("q=") tag. The only option defined in this base specification is "txt", indicating the use of a TXT RR. A later extension of this standard may define another RR type.

Strings in a TXT RR MUST be concatenated together before use with no intervening whitespace. TXT RRs MUST be unique for a particular selector name; that is, if there are multiple records in an RRset, the results are undefined.

TXT RRs are encoded as described in Section 3.4.1.

4. IANA Considerations

This document reuses the IANA registrations in [RFC6376] in Section 7.

5. Security Considerations

This document recommends thorough review and adherence to the Security Consideration in [RFC6376] in Section 8.

6. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/rfc/rfc1034>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/rfc/rfc2045>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/rfc/rfc5322>.

[RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, DOI 10.17487/RFC5598, July 2009, <https://www.rfc-editor.org/rfc/rfc5598>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/rfc/rfc6376>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/rfc/rfc8017>.

[RFC8463] Levine, J., "A New Cryptographic Signature Method for DomainKeys Identified Mail (DKIM)", RFC 8463, DOI 10.17487/RFC8463, September 2018, <https://www.rfc-editor.org/rfc/rfc8463>.

Appendix A. Acknowledgments

The parent document to this is DKIM [RFC6376] from which the vast majority of content is copied from with only slight editing to update for DKIM2. The DKIM RFC was editted by Dave Crocker, Tony Hansen, and Murray Kucherawy. Credit for the content and specification for DKIM, from which DKIM2 is derived from, goes to them.

Author’s Address

Wei Chuang
Google
Email: weihaw@google.com