

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 6, 2021

M. Kucherawy
July 5, 2020

Recognized Transformations of Messages Bearing DomainKeys Identified
Mail (DKIM) Signatures
draft-kucherawy-dkim-transform-02

Abstract

DomainKeys Identified Mail (DKIM) introduced a mechanism whereby a mail operator can affix a signature to a message that validates at the level of the signer's domain name. It specified two possible ways of converting the message body to a canonical form, one intolerant of changes and the other tolerant of simple changes to whitespace within the message body.

The provided canonicalization schemes do not tolerate changes in a message such as conversion between transfer encodings or addition of new message content. It is useful to have these capabilities to allow for transport through gateways, and also for transport through handlers (such as mailing list services) that might add content that would invalidate a signature generated using the existing canonicalization schemes.

This document presents a mechanism for declaring that a message underwent any of a handful of well-defined transformations prior to being re-signed by a mediator, so that a verifier might rewind such modification(s) and thereby confirm that the original signature still verifies against the original content.

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 <http://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."

Internet-Draft

DKIM Transformations

July 2020

This Internet-Draft will expire on January 6, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Background	3
2.	Definitions	3
3.	The 'tf' DKIM Signature Tag	4
4.	DKIM Operational Flow	4
4.1.	Detail	5
5.	The 'subject' Transformation	6
6.	The 'footer' Transformation	6
7.	The 'mimeify' Transformation	7
8.	The 'add-part' Transformation	8
9.	The 'mime-wrap' Transformation	9
10.	Discussion	10
11.	Security Considerations	11
11.1.	Imported from DKIM	11
11.2.	False Transformation Claims	12
12.	IANA Considerations	12
12.1.	DKIM Transformations Registry	12
13.	References	13
13.1.	Normative References	13
13.2.	Informative References	14
Appendix A.	Example	14
Appendix B.	Acknowledgements	15
Appendix C.	To-Do List	15
Appendix D.	Change Log	16

1. Background

DomainKeys Identified Mail (DKIM) [[RFC6376](#)] defines a mechanism whereby a verified domain name can be attached to a message, or portion of a message, using a cryptographic signature. It presents two possible schemes for converting the header block to a canonical form, and similarly two schemes for canonicalizing the body. In each case, one scheme permits no changes whatsoever, and the other permits limited changes restricted to areas such as whitespace munging, case changing, and header field wrapping.

Some agents deliberately, but innocently, modify content in transit. A prime example of this is mailing lists, which might add a prefix to the Subject field of a message, add list-specific information to the header (in the form of new header fields), or append administrivia to the body of messages before they are re-mailed to the list subscribers. Use of mailing lists with respect to DKIM, and a discussion of related challenges, can be found in [[RFC6377](#)]. The urgency to solve this family of problems increased dramatically with the large-scale introduction of Domain-based Message Authentication, Reporting, and Conformance (DMARC) [[RFC7489](#)].

There is a desire to have DKIM signatures survive transit through lists. One way to do this is to make use of DKIM's "l=" tag which limits the portion of the body that is signed. This exposes an attack vector, however, since one can simply append any content to a partly-signed message and the signature will continue to verify. (See [Section 8.2 of \[RFC6376\]](#).)

This document defines an incremental mechanism to declare that a signature is being applied to message content after some number of a small set of well-defined, reversible content transformations. The message verifier can then reverse the effect of the claimed transformation(s) and, theoretically, recover the original content and confirm its integrity relative to an original signature.

The utility of this mechanism is predicated on the notion that agents that modify signed messages will do using only the known (registered) transformations, and that common transformations will be registered as they are developed.

[2.](#) Definitions

Numerous terms used here, especially "Author" and "Mediator", are defined in [[RFC5598](#)].

For the purposes of this experiment, a transformation is "reversible" if at the time the message is received, the verifier has enough

information to recover the pre-transformation content. For example, a transformation that removes a MIME part with an undesired media type or filename extension cannot be undone by the receiver because it cannot restore content it doesn't have; such a transformation is not reversible and thus not a candidate for consideration here. However, a transformation that adds a specific header field to a message is reversible because the verifier can simply remove the header field.

[3.](#) The 'tf' DKIM Signature Tag

This section defines the 'tf' DKIM signature tag.

The presence of this tag is an indication to a verifier that the agent adding this signature transformed the original message between receipt (and verification of any previously-applied signature) and retransmission, and that such transmission was one of a set that are common, well-defined, and reversible.

The value of this tag is one of the transformations registered in the DKIM Message Transformations registry. See [Section 12](#).

Using ABNF, as defined in [[RFC5234](#)]:

```
sig-tf-tag      = %x74.66 [FWS] "=" [FWS] sig-tf-tag-trans
sig-tf-tag-trans = Token *("," Token)
                  ; expected to be a list of one or more
                  ; transformation names found in the DKIM
                  ; Message Transformations registry
```

"Token" is imported from [[RFC2045](#)], and "FWS" is imported from [[RFC6376](#)].

A verifier finding a signature with the "tf" tag present but bearing a value it does not recognize ignores its presence (other than including it in hash computation).

[4.](#) DKIM Operational Flow

In all cases, DKIM operations involving this tag begin with a message author generating content and submitting it to the appropriate Message Submission Agent (MSA). The MSA is presumed to have some kind of DKIM signature generation capability, and thus the message will have an author domain signature attached to it.

When a message arrives at a Mediator or other intermediary that wishes to distribute an altered form of the author's content, such as a Mailing List Manager (MLM) configured to do so, it generates an

additional DKIM signature with the new form of the content as input. This second includes the "tf" tag, announcing which known transformation(s) was applied to the message prior to creation of the Mediator's signature. Importantly, the original signature is not removed from the message nor is it altered in any way.

Since DKIM-compliant verifiers ignore signature tags of which they are not aware, this is a purely incremental change as it will not interfere with the deployed DKIM infrastructure.

A DKIM verifier aware of this tag will first confirm that the Mediator's signature is valid. On doing so, it can then apply the reverse of the claimed transformation. This will restore the message to the form and content originally submitted by the Author, and the Author's signature will then be valid over the restored content.

This might be used to confirm that a message which passed through a Mediator can still be considered to have a valid Author signature, satisfying policy checks such as those described in [[RFC7489](#)].

[4.1.](#) Detail

1. Author A generates message M, addressed to recipient R, which is a Mediator (a mailing list manager, for example).
2. Author submits M to its MSA.
3. MSA generates and attaches DKIM signature S(M), and sends the message toward its destination, which is the servers accepting messages for R.
4. The message arrives at R. R might verify signature S(M) and apply any local policy if the verification fails.
5. R selects an ordered list of one or more of the registered, reversible transformations, T1, T2, etc., to be applied to M. The complete list is referred to as T. (The reverse operations will be called T' as an ordered list, whose order is the reverse of T, and the individual reverse transformations are called T1', T2', etc.) R thus generates T(M) as the new content. The new content necessarily includes S(M).
6. R also generates a signature of T(M), which is S(T(M)). This new signature includes the "tf" tag defined above, identifying the ordered sequence of transformations T that was used in the previous step. It then sends the message toward its final destination(s). For a mailing list manager, this would be all of the current list subscribers.

7. The Mediator version of the message arrives at ultimate recipient Z. Assuming no unexpected damage to the message in transit, Z will be able to validate S(T(M)).
8. If the verifier is not aware of this tag and its meaning, or if the verifier is not aware of how to reverse the identified transformation, normal DKIM verification continues from here, and this modified algorithm terminates. It would be expected that S(T(M)) would be valid, but S(M) would not.
9. The compliant verifier applies T' to the validated message content. By definition, T'(T(M)) is M. (If this is not true, then at least one of the original transformations was not reversible.) Since the Mediator preserved Author signature S(M), the verifier can now attempt to validate the Author signature

against the recovered original content.

5. The 'subject' Transformation

Mailing list services commonly apply a "tag" to the Subject field of a message identifying the message as having been distributed as part of a list. By far the most common tag method is to prefix the Subject field with the name of the list in square brackets (ASCII 0x5b and 0x5d), possibly followed by a space and a sequence number. Accordingly, this transformation describes exactly such a mutation. Specifically, the mutation is the addition of a string to the beginning of the Subject field comprised of alphanumeric characters, a limited set of punctuation, or digits, surrounded by square brackets, possibly including and followed by whitespace. In ABNF terms, the string is described by:

```
s-punct = 0x45 / 0x5f / 0x2f / 0x20 / 0x2e
s-tag    = 0x5b 1*( ALPHA / DIGIT / s-punct ) 0x5d 1*FWS
```

Thus, the reverse operation is simply the removal of any such substring at the front of the Subject field.

If there is no Subject field prefix matching the above ABNF, then the transformation reversal cannot be computed and an error is returned.

6. The 'footer' Transformation

Mailing lists sometimes add a "footer" to a message, typically consisting of a small number of lines of text identifying the name of the list and some other administrivia, and usually including a URL where subscriptions can be managed or list archives can be found. Such trivial text edits are reversible, so these too are a candidate for this mechanism.

A "footer" for the purposes of this capability is all text below a trivial boundary marker. A boundary comprises a line of text made up solely of two or more hyphen or underscore (0x2d or 0x5f) characters. Therefore, reversing this transformation is accomplished by searching backwards, a line at a time, from the end of the message, until such a line is found. When found, the message is truncated such that the line and all lines after it are removed.

If no such line is found, then the transformation reversal cannot be computed and an error is returned.

7. The 'mimeify' Transformation

The "mimeify" transformation converts a message that is not formatted according to Multipurpose Internet Mail Extensions (MIME) [[RFC2045](#)], and converts it to that form. This allows a Mediator to place the original content in one MIME part, and its own additional content in a second MIME part. The reverse transformation is to remove the second MIME part altogether, and then strip away all MIME structure, leaving only the original author content.

More specifically, the transformation follows these steps:

1. A "MIME-Version" header field is added, as described in [[RFC2045](#)].
2. A "Content-Type" header field is added, also as described in [[RFC2045](#)]. The media type is "multipart/mixed". A unique compliant boundary is also generated.
3. Two MIME parts are created. The first MIME part is of type "text/plain", and contains the body of the original message. The second MIME part contains whatever content the Mediator is configured to add, and uses a media type appropriate to that content.
4. The body of the message is replaced with the following, in a manner compliant with [[RFC2045](#)], namely:
 - A. the boundary;
 - B. an optional "Content-Type" header field indicating the original content used the default "text/plain" media type, and the optional "charset" parameter;
 - C. a line break;

- D. the body of the original message;

- E. a line break;
- F. the boundary;
- G. any MIME header fields needed to introduce the content the Mediator wishes to add;
- H. a line break;
- I. the Mediator's content;
- J. the terminating boundary.

The reverse of this transformation is as follows:

1. Extract the full content of the first MIME part.
2. Discard the entire message body, and replace it with the extracted content above.
3. Remove the "Content-Type" and "MIME-Version" header fields.

If any step cannot be completed because the stated header field or content cannot be located, an error is returned.

8. The 'add-part' Transformation

The "add-part" transformation augments a multipart message that is already formatted according to MIME by appending an additional part that includes the content the Mediator wishes to add.

This transformation cannot be used unless the media type of the message as a whole (the one named in the Content-Type field in the header of the message itself) is "multipart/mixed". Simply put, a new part within the existing set of parts is added at the end, containing the Mediator's content.

More specifically, the transformation follows these steps:

1. Determine the MIME boundary used to separate parts, found in the top-level Content-Type header field.
2. At the point of the terminating boundary in the original message, insert a non-terminating instance of the same boundary.

3. After the new boundary, write any MIME fields needed to introduce the content the Mediator wishes to add.
4. Insert a line break, followed by the Mediator's content, and an additional line break.

The reverse of this transformation is as follows:

1. Locate the last instance of the boundary found in the Content-Type header field of the message itself.
2. Delete the content from that point in the message until the terminating instance of the boundary.

If any setp cannot be completed because the stated MIME part cannot be located, an error is returned.

9. The 'mime-wrap' Transformation

The "mime-wrap" transformation augments a message that is already formatted according to MIME by enclosing the existing MIME structure in a new layer. This new layer contains two parts: the original MIME structure in its first part, and the Mediator content in its second part.

More specifically, the transformation follows these steps:

1. Remove the Content-Type header field from the message.
2. Generate a new Content-Type header field, compliant with [\[RFC2045\]](#), with media type "multipart/mixed", and a boundary.
3. The body of the message is replaced with the following, in a manner compliant with [\[RFC2045\]](#), namely:
 - A. the new boundary;
 - B. the previously deleted Content-Type header field;
 - C. a line break;
 - D. the entire original content of the message;
 - E. a line break;
 - F. the new boundary;

- G. any MIME header fields needed to introduce the content the Mediator wishes to add;
- H. a line break;
- I. the Mediator's content;
- J. the terminating instance of the new boundary.

This leaves the new message as a MIME message with two parts at the outermost layer; the original message appears as the first part, and the Mediator's content is the second part.

The reverse of this transformation is as follows:

1. Extract the Content-Type header field from the first MIME part in the message. This appears immediately after the first MIME boundary in the message.
2. Replace the Content-Type header field of the message with the one extracted above.
3. Extract the content of the first MIME part in the message. This appears between the first two instances of the outermost MIME boundary.
4. Replace the entire message body with the extracted MIME part.

If any step cannot be completed because the stated MIME part cannot be located, an error is returned.

10. Discussion

[Section 3.5 of \[RFC6376\]](#) defined an optional DKIM signature tag ("z=") that can be used to reconstruct the header field set that was signed by the author. When a signature fails to verify, this information could conceivably be used to replay the correct (original) header fields through canonicalization and possibly yield a passing result.

Doing this augmented replay blindly would allow a signature to pass when it failed because some alteration correctly rendered the original content invalid or even dangerous. This is manifestly not an error. Identifying which mutations of the original content ought to be permissible necessarily relies on heuristics and possibly local knowledge. However, a mutation universally considered to be tolerable should become part of the canonicalization process rather than being identified and handled in this manner. Moreover, if two

implementations apply different heuristics, the result of verification is no longer deterministic. As a result, [\[RFC6376\]](#) asserts that use of the "z=" content, if present, can only be used for diagnostic purposes.

In contrast, the proposal here enumerates a handful of specific mutations known to be safe, and in common use, that are also reversible, which means the Author's original content can be unambiguously recovered and subjected to the usual signature verification process even though the message has been legitimately modified by a Mediator.

It does not take much imagination to conceive of a legitimate message using the capability described here that fails some part of the process. For example, the "footer" transformation does not account for a footer block that itself contains a boundary marker, and so reversing that transformation as described would produce a wrong result. This is harmless, however, as the verifier then is no worse off than it was before in that it still doesn't have the original content, and thus operates as if none of this proposal was applied (i.e., the original signature still fails). The proposal is only incremental to what DKIM can provide when it actually does work to recover original content.

It is expected that the definitions of the known transformations will evolve over time as we gain community experience with what works.

As with DKIM itself, there are local policy decisions that can come into play. Some DKIM verifiers insist that, for example, the Subject field be included in the signed content, and will disregard a valid DKIM signature where that is not the case. This requirement exceeds what DKIM specifies, but verifiers have such discretion if they feel

it enhances user protection. So it is with this proposal: The fact that an Author signature can validate after certain transformations are reversed does not obligate the verifier to change its handling. In particular, an operator may decide that reversal of certain transformations is too fragile to render better handling, and it is free to apply that discretion.

11. Security Considerations

11.1. Imported from DKIM

[Section 8 of \[RFC6376\]](#) discusses numerous security considerations relevant to DKIM. Of particular interest here is [Section 8.2](#), which discusses concerns regarding signatures that still verify in the presence of added message content.

Kucherawy

Expires January 6, 2021

[Page 11]

Internet-Draft

DKIM Transformations

July 2020

11.2. False Transformation Claims

Conceivably, some of these transformations or those registered in the future could be computationally expensive or require non-trivial ephemeral resource allocation (e.g., storage), especially for large or complex messages. An attacker could send signatures in claiming some or all of the known transformations on a message which a participating verifier would then attempt to execute, presumably in an attempt to recover original content, as a denial of service attack.

There is little reason to believe that any given transformation might be applied more than once, or that certain combinations have any practical application (e.g., "footer" is unlikely to be useful when combined with any of the MIME transformations). This experimental document does not explicitly proscribe these, but implementers may choose to detect such strange requests and disregard them.

12. IANA Considerations

12.1. DKIM Transformations Registry

IANA is requested to create a new registry in the DomainKeys Identified Mail (DKIM) Parameters group called the "DKIM Transformations Registry". This registry will enumerate known

reversible content transformations that might be made by Mediators to messages bearing DKIM signatures.

Entries in this registry include all of the following:

Name: A simple name for a reversible message transformation;

Description: A terse description for the transformation;

Specification: A reference to a stable specification in which this transformation and its reverse are clearly described;

Status: Must be one of:

- * "active", meaning the transformation is in current use;
- * "deprecated", meaning the transformation is not in current use.

An entry may be added or updated in this registry only when it meets the requirements of the "Specification Required" rules found in [\[RFC5226\]](#). The Designated Expert will confirm that the referenced specification is clear and complete, and that the transformation and its reverse are not ambiguous.

The initial entries in this registry are as follows, all with status "active":

add-part: Defined in [Section 8](#) of this document. The simple description is "append an extra text MIME part to a MIME-formatted message".

footer: Defined in [Section 6](#) of this document. The simple description is "append a plain text footer to an unformatted message".

mime-wrap: Defined in [Section 9](#) of this document. The simple description is "wrap a MIME-formatted message in a new multipart layer".

mimeify: Defined in [Section 7](#) of this document. The simple description is "convert a non-MIME message to a MIME message".

subject: Defined in [Section 5](#) of this document. The simple description is "prepend a tag to the Subject field".

[13.](#) References

[13.1.](#) Normative References

- [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/info/rfc2045>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [RFC 5226](#), DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [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/info/rfc5234>>.
- [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/info/rfc6376>>.

Kucherawy

Expires January 6, 2021

[Page 13]

Internet-Draft

DKIM Transformations

July 2020

[13.2.](#) Informative References

- [RFC5598] Crocker, D., "Internet Mail Architecture", [RFC 5598](#), DOI 10.17487/RFC5598, July 2009, <<https://www.rfc-editor.org/info/rfc5598>>.
- [RFC6377] Kucherawy, M., "DomainKeys Identified Mail (DKIM) and Mailing Lists", [BCP 167](#), [RFC 6377](#), DOI 10.17487/RFC6377, September 2011, <<https://www.rfc-editor.org/info/rfc6377>>.
- [RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based

[Appendix A](#). Example

This section presents a simple demonstration of the proposed capability using the "subject" transformation described in [Section 5](#). Since only a header field is modified in that case, the example does not include the message body. Also, only some fields are shown, and base64 fields reflecting hashes contain mock values and may be truncated as their values are not germane to this demonstration.

First, the header of the original message:

```
From: Alice Participant <apart@example.com>
To: IETF DKIM WG <dkim@ietf.example.org>
Subject: I have an idea!
Date: Sun, 5 Jul 2020 13:26:23 -0700 (PDT)
DKIM-Signature: v=1; d=example.com; s=tolkein; a=rsa-sha256;
  c=relaxed/simple; t=1593980799;
  h=Date:From:To:Subject;
  bh=5jBgWS5CnwV6HNp7irm1aMWW/V00YHhvFIQldGZn7v0=;
  b=Zaed9V18tBX789K2fpIG0H...
```

Alice sends this message with its fabulous DKIM idea to the list. The list software receives it, prefixes a tag to its Subject field, and then relays it to the list subscribers. The list software thus emits the following mutated message:

```
From: Alice Participant <apart@example.com>
To: IETF DKIM WG <dkim@ietf.example.org>
Subject: [ietf-dkim] I have an idea!
Date: Sun, 5 Jul 2020 13:26:23 -0700 (PDT)
```



```
DKIM-Signature: v=1; d=example.com; s=tolkein; a=rsa-sha256;  
c=relaxed/simple; t=1593980799;  
h=Date:From:To:Subject;  
bh=5jBgWS5CnwV6HNp7irm1aMWW/V00YHhvFIQldGZn7v0=;  
b=Zaed9V18tBX789K2fpIG0H...  
DKIM-Signature: v=1; d=ietf.example.org; s=rabbit; a=rsa-sha256;  
c=relaxed/simple; t=1593980802; tf=subject;  
h=Date:From:To:Subject;  
bh=cwpuQruv+3/b493YEQBqBLS8UgNGP+rQ6fuhJ2csvdQ=;  
b=l0+hCymcA93hq6Pex20sCFfdDjomrBUe7JVRSfmJN...
```

Bob is subscribed to this list, so it arrives at his Mail Transfer Agent (MTA) which has a DKIM verifier participating in this experiment. Were it to attempt to validate the "example.com" signature, it would fail because the Subject field fed to the signing algorithm is not the same as that fed to the verifying algorithm.

However, the "ietf.example.org" tag does verify. It furthermore contains the "tf" tag indicating a Subject field mutation occurred. The verifier thus re-attempts the "example.com" signature verification after having applied the reverse of the Subject field mutation. In so doing, the hash algorithm at the verifier now receives the same content that was originally passed to the signing algorithm at "example.com", which means the Author signature validation succeeds. This fact can then be used to augment whatever local policy decision might otherwise have been made in the absence of a valid author domain signature.

[Appendix B.](#) Acknowledgements

The original team developing this concept included Michael Adkins and Wez Furlong.

The author wishes to acknowledge (names) for their comments during the development of this document.

[Appendix C.](#) To-Do List

- o Experimentally implement at least the "subject" and "footer" transformations.

[Appendix D.](#) Change Log

02: Add a simple example. Mention verifier handling discretion.
Some language tidying.

01: Resurrection; add "subject" and "footer", and generally more
prose.

00: Initial revision.

Author's Address

Murray S. Kucherawy

EMail: superuser@gmail.com

