

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: October 7, 2015

M. Kucherawy
April 5, 2015

A List-safe Canonicalization for DomainKeys Identified Mail (DKIM)
draft-kucherawy-dkim-list-canon-01

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 structured message such as conversion between transfer encodings or addition of new message parts. 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 generating a canonicalization that can allow easy detection of modified content while still being valid for the content it originally signed. It also presents a use profile of DKIM that takes advantage of this capability.

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."

This Internet-Draft will expire on October 7, 2015.

Internet-Draft

DKIM List Canonicalization

April 2015

Copyright Notice

Copyright (c) 2015 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 'list' Canonicalization Description	3
3.1.	Preparing Content	4
4.	'The 'lh=' Signature Tag	5
5.	Use Profile	6
6.	Security Considerations	6
6.1.	Imported from DKIM	6
6.2.	Added Content May Not Be Safe	7
7.	IANA Considerations	7
7.1.	DKIM-Signature Canonicalization Body Registry	7
7.2.	DKIM-Signature Tag Specifications Registry	7
8.	References	7
8.1.	Normative References	7
8.2.	Informative References	7
Appendix A.	Example	8
Appendix B.	To-Do	10
Appendix C.	Acknowledgements	10

Internet-Draft

DKIM List Canonicalization

April 2015

1. Background

DomainKeys Identified Mail [[RFC6376](#)] (DKIM) 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](#)].

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 a new body canonicalization for DKIM that includes a partial signature for each message part in a message structured using Multipurpose Internet Mail Extensions (MIME; see [[RFC2045](#)]). This allows a clear delineation between the author-generated content (which would be signed by the author) and content added downstream (which would be signed by the other actor). A DKIM verifier can then determine whether the author-generated content is intact, and then identify and verify the content that was added later.

The utility of this mechanism is predicated on the notion that agents that modify signed messages will do so in ways compatible with MIME.

[2.](#) Definitions

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

[3.](#) The 'list' Canonicalization Description

This section defines the 'list' body canonicalization algorithm.

Kucherawy

Expires October 7, 2015

[Page 3]

Internet-Draft

DKIM List Canonicalization

April 2015

Put simply, the list canonicalization constructs a hash tree of the MIME structure of the message after each part has been decoded (for those with a Content-Transfer-Encoding field). The hash used is implied by the signature algorithm to be used (see the DKIM "a=" tag). Each of the hashes can be made a part of the signature to allow for more precise part validation, and identification of added content.

[3.1.](#) Preparing Content

A message is prepared for canonicalization by applying the following steps in order:

1. Create an empty tree. Each node of the tree includes the following components:
 - A. The MIME type and subtype of the part, expressed as would be found in a Content-Type header field, with no whitespace or comments;
 - B. The unencoded content represented by the MIME part at this node;
 - C. A series of octets that will contain a hash of the content;
 - D. A series of zero or more pointers to other (child) nodes.
2. If the message is not encoded using MIME, insert a node at the root of the tree using a type/subtype of "text/plain" and the

full body content. The hash is not initialized.

3. If the message is encoded using MIME, then the tree is populated in a way that mirrors the MIME structure of the message. In particular, the outermost MIME object will appear at the root node, and the only nodes that have children are those with a MIME type of "multipart". The hashes are not initialized.
4. For each leaf node, compute a hash of the content of that node. Store the hash in the node.
5. For each non-leaf node, if all of its child nodes now have computed hashes, concatenate the hashes (with order preserved), and compute and store a hash of the concatenation.
6. Repeat the previous step until all hashes in the tree have been populated.

When this canonicalization is in use, the "bh=" tag will contain the

hash stored at the root of the tree. The processes for signing and verification are otherwise unchanged.

4. 'The 'lh=' Signature Tag

A signer can include an "lh=" tag, defined here, to make more than just the root hash information available to verifying agents. This permits identification of the specific part of the MIME structure that was modified, added or removed by an intermediary.

The "lh=" tag is constructed by performing an in-order traversal of the canonicalization tree described in [Section 3.1](#). At each node, each of the following is output, separated by a colon character (ASCII 0x3A):

1. A base64 expression of the hash at that node;
2. The MIME type of that node;
3. An integer expression of the number of children at that node.

Between each node's output, a comma character (ASCII 0x2C) is output.

Reconstruction of the MIME tree can be accomplished by the following steps:

1. Create a tree "T" containing a single empty node.
2. Create an empty node queue, "Q".
3. Create an information queue "I", containing the sequence of node information fields found in the "lh=" tag.
4. Select the root node of the tree. Call this node "N".
5. Extract the first batch of node information ("B") from the "lh=" tag.
6. Store the hash and MIME type from "B" into "N".
7. Enqueue the specified number of empty nodes into "Q", and attach them all as children of "N".
8. If "I" and "Q" are both empty, terminate. If one is empty and the other is not, an error has occurred.
9. Extract the next batch of node information from "I", as "B".

10. Dequeue the next node from "Q", as "N".
11. Return to step 6.

By comparing the hashes in and structure of this tree to those in the canonicalized tree, a receiver can identify parts of the tree (or entire subtrees) that have been modified. Parts not covered by the signature can also be identified.

[5.](#) Use Profile

The intended use of this mechanism is to affix two DKIM signatures to a message. The first signature is added by the Author, and canonicalizes the original message in its entirety. The second signature is added by a modifying intermediary, such as a mailing

list manager (MLM).

When verifying, the Author signature on an unmodified message would pass verification. For a modified message, in the typical case, the verification step would observe that the Author signature failed but the intermediary's signature verified. When the "lh=" tag is present, it is possible to reconstruct the MIME structure of the signed message and compare it to that of the received message, including hashes of the content seen by each party. By comparing hash values at each node of the MIME structures, it is possible to determine in which MIME parts changes were made and/or new parts added or removed by the intermediary. The verifying agent can then determine whether those changes are acceptable before allowing the message to continue toward delivery.

It is also possible to determine which agents in the handling chain took responsibility for which parts of the content. For example, while a Mediator's signature might indicate that the mediator is responsible for the entire (rewritten) message, it might also be possible to determine that the Author takes responsibility for all but one part of the message as well. The excluded part would be the part added by the Mediator, and can be handled separately from the Author's content.

[6.](#) Security Considerations

[6.1.](#) 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.

[6.2.](#) Added Content May Not Be Safe

When the use profile described in [Section 3](#) is applied, it is important to note that the added content was not signed by the Author domain, but only by the domain of the intermediary. Operators that might grant preferential handling based on valid DKIM signatures from favorable domains; assuming that appended content in the presence of such signatures does not mean the appended content is necessarily

safe.

[7.](#) IANA Considerations

[7.1.](#) DKIM-Signature Canonicalization Body Registry

IANA is requested to add the following entry to the DKIM-Signature Canonicalization Body Registry:

Type: list
Reference: [this document]
Status: active

[7.2.](#) DKIM-Signature Tag Specifications Registry

IANA is requested to add the following entry to the DKIM-Signature Tag Specifications Registry:

Type: lh
Reference: [this document]
Status: active

[8.](#) References

[8.1.](#) Normative References

- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", [RFC 2045](#), November 1996.
- [RFC6376] Crocker, D., Hansen, T., and M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures", STD 76, [RFC 6376](#), September 2011.

[8.2.](#) Informative References

- [RFC5598] Crocker, D., "Internet Mail Architecture", [RFC 5598](#), July 2009.
- [RFC6377] Kucherawy, M., "DomainKeys Identified Mail (DKIM) and

[Appendix A](#). Example

To illustrate the use of this addition to DKIM, consider a message whose header and content are as follows:

```
From: sender@example.com
To: recipient@example.net
Date: Mon, 23 Mar 2015 11:21:33 -0700
Subject: test message
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="foobar"

--foobar
Content-Type: text/plain

Text part #1

--foobar
Content-Type: text/plain

Text part #2

--foobar--
```

Figure 1: Example Message

The MIME structure in this message can be represented as a tree. A node with media type "multipart" has a set of one or more children nodes, each of which starts with the corresponding boundary. A node of any other type contains actual content, and has no descendents, but has siblings under the same parent node. Thus, as a tree, the example message might be represented thus:

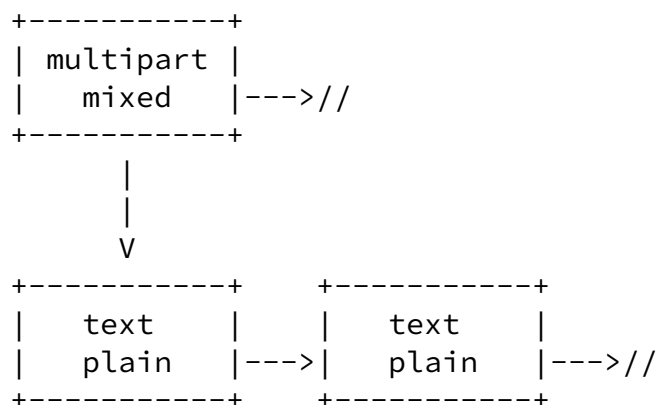


Figure 2: MIME structure

signature-generating entities.

Kucherawy

Expires October 7, 2015

[Page 9]

Internet-Draft

DKIM List Canonicalization

April 2015

[Appendix B.](#) To-Do

Explain how this works when the input message is not already a MIME message. Probably just canonicalize it as a multipart/mixed with a single text/plain in it.

Handle more complex MIME structures from the author, such as something that's already multipart/mixed with some non-trivial structure to it.

[Appendix C.](#) Acknowledgements

The original idea was proposed by Ned Freed.

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

Author's Address

Murray S. Kucherawy

EMail: superuser@gmail.com

Kucherauw

Expires October 7, 2015

[Page 10]