Network Working Group
Internet-Draft
Obsoletes: 4686 (if approved)
Intended status: Standards Track
Expires: 20 April 2026

B. Gondwana
Fastmail Pty Ltd
17 October 2025

A method for describing changes made to an email
draft-gondwana-dkim2-modification-alegbra-04

Abstract

This memo describes a method for describing the changes made to an email during common email modifications, for example those caused by mailing lists and forwarders.

While this is general enough to be used for any changes, it is anticipated that this method will normally be used for removing added data rather than large complex changes.

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 20 April 2026.

Copyright Notice

Table of Contents

1. Background and motivations

Currently, when an email is sent with a DKIM signature, the message can go through multiple forwarders and still be authenticated, however if a single change is made to a header which is covered by the signature, or to the body, then the signature no longer validates - and it's impossible for the receiver to know what was changed, or even if the entire message was replaced.

By producing a way to describe changes, the recipient can examine the sections which were changed and determine whether the change was malicious. Along with signatures which validate against previous versions, this can be used to allow signing systems to take accountability for only the version of the message which passed through their system.

2. The MailVersion Header Field

This document describes an ordered set of header fields, each of which describes the message at a specific version, along with instructions on how to convert that version back to the previous version.

2.1. MailVersion Header

The format of the file is a tag-list. TODO: we need to decide where we pull tag-list from!

Tag	Type	Value
v	position	Revision number (range: 1 to 100)
bh	base64	Body Hash value for this revision
bin.n.m	base64	Hash for the binary representation of the numbered mime part
b	body-recipe	Recipe to replicate the previous version of the message body
h.header	head-recipe	Recipe to replicate the previous version of the named header field

Table 1

2.2. body-recipe

The Body Recipe is a comma separated list of instructions. Each instruction starts with a prefix. Commas can be followed by optional whitespace.

Prefix	Value	Action
c:	start-end	Copy the lines (inclusive) numbered from 1.
b:	base64	Decode the base64 to get the value of a line to insert.
t:	text	Copy the exact text to get the value of a line to insert.
z	none	If present, says that changes have been made to the body which can not be described to get back to the earlier version, meaning the signing system takes accountability for the full content.

Table 2

2.3. header-recipe

The Header Recipe is a comma separated list of instructions. Each instruction starts with a prefix. Commas can be followed by optional whitespace.

While key names are case insensitive, implementations SHOULD create the header with the same case as the key.

Prefix	Value	Action	
d:	integer	'*'	Delete the indexed (numbered from 1) copy of this header field, or all copies
b:	base64	Decode the base64 to get the value of a header field to insert	
t:	text	Copy the exact text to get the value of a header field to insert	
z	none	If present, says that changes have been made to the named header field which can not be described to get back to the earlier version, meaning the signing system takes accountability for the full content of this message.	

Table 3

2.4. Examples

Example for a message which has had Subject and From replaced, and Reply-To added.

```
From: brong@fastmailteam.com.dmarc.fail
To: dkim2@lists.ietf.org
Reply-To: dkim2@lists.ietf.org
MailVersion: v=2;
 bh=[...];
 h.Subject=d:*,t:A replacement for DKIM;
 h.From=d:*,b:YnJvbmdAZmFzdG1haWx0ZWFtLmNvbQo=;
 h.Reply-To=d:*
```

Example:

```
MailVersion: v=2; bh=[...]; b=c:1-500,c:520-520
```

Example - a URL was substituted in the content of the body (complex, but still easily doable!)

```
MailVersion: v=3;
 bh=[...];
 b=c:1-500,
   b:PGEgaHJlZj0iaHR0cHM6Ly93d3cuZXhhbXBsZS5jb20iPkV4YW1wbGU8L2E+Cg==,
   c:501-702
```

The decision whether to use ’b’ or ’t’ is up to the system creating the diff, however ’t’ has a limited set of characters that are safe to use in tag-values

Likewise, it is expected that ’c’ will normally be used to copy lines directly from the new message, however in cases where a message needs to transit 7 bits systems cleanly, the email modifier may need to re-encode the octets of the original message, and this allows for doing so, albeit at some expense in header bloat!

3. Iterative application

To get back to the original message and confirm that it was unchanged, it is necessary to apply this algorith iteratively.

For example if you receive a message for which there is a modification to the headers at v=3 and a modification to both headers and body at v=2, to recreate the original message you would first apply the header changes from v=3, then apply the header and body changes for v=2. If this doesn’t create a message which validates with the initial v=1 hash, then some hop has corrupted the message.

4. Security

TBA

5. IANA Considerations

TBA

6. Informative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/rfc/rfc2046>.

[RFC3284] Korn, D., MacDonald, J., Mogul, J., and K. Vo, "The VCDIFF Generic Differencing and Compression Data Format", RFC 3284, DOI 10.17487/RFC3284, June 2002, <https://www.rfc-editor.org/rfc/rfc3284>.

Appendix A. Changes from Earlier Versions

A.1. draft-gondwana-dkim2-modification-algebra-03

* Rename header to 'MailVersion'

* Remove all requirements that it integrates with DKIM2.

* Add body hash and per-mime-part hashes (NOTE: this is a bunch of extra calculation, so definitely to discuss)

A.2. draft-gondwana-dkim2-modification-algebra-02

* change the header format to have unique keys, making it fit the ABNF for these types of headers.

* allow easier editing of multi-value headers by always popping the first header and always prepending newly added headers.

* change body to use d.0, d.1, etc with the program in the value, so that the program ordering is reliable regardless of the parser used to read the header.

A.3. draft-gondwana-dkim2-modification-algebra-02

* change to using line numbers rather than octet offsets

* remove d= base64 decoding capability

* for multiple lines; require a separate b= or t= for each line

A.4. draft-gondwana-dkim2-modification-algebra-01

* rename 'DKIM2-Diff' headers to 'DKIM2-Delta'

* add 'z=y' option to DKIM2-Delta-Body for "complete replacement"

* add d= base64 decoding option to DKIM2-Delta-Body

A.5. draft-gondwana-dkim2-modification-algebra-00

* original version

[[This section to be removed by RFC Editor]]

Author’s Address

Bron Gondwana
Fastmail Pty Ltd
Level 2, 114 William Street
3000
Australia
Phone: +61 457 416 436
Email: brong@fastmailteam.com