Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

B. Gondwana
Fastmail Pty Ltd

3 November 2025

A method for describing changes to emails
draft-gondwana-dkim2-mailversion-00

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.

This method also captures hashes of important features of the message, allowing validation that the changes were described correctly, and allowing a signature which covers the Mail-Version header to, by extention, ensure that the important content of the message is unchanged.

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 7 May 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. By undoing the changes, it is possible to recreate a message which matches the original signature, and hence provide accountability for the content which was present in the copy of the message to which the signature was applied.

2. The Mail-Version 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.

To well formed, a message must have a monotonically increasing set of Mail-Version header fields, with the first having mv=1 and each having an increasing integer number, with no gaps.

2.1. Mail-Version Header

The format of the value is a tag-list. Tag-lists contain key=value; key2=value2; - they can be wrapped, and all whitespace is normalised to a single space.

TODO: we need to specify tag-list.

Tag	Type	Value
mv	position	Revision number (range: 1 to 100)
h	headerlist	List of headers signed by the hh field.
a	hash-alg	Hash Algorithm used (at least sha256 must be supported); used for hh, bh and ph
hh	base64	Header Hash for the named headers, using the ’relaxed’ header algorith from [DKIM]
bh	base64	Body Hash, using the ’relaxed’ body algorithm from [DKIM]
ph.n.m	base64	Hash for the binary representation of the numbered mime part, after removal of any content-transfer-encoding (this is the data returned by FETCH BINARY[n.m] as described in [IMAP] section 7.5.2)
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

TODO: do we want to add ’bh.n.m’ to hash the un-decoded body of individual MIME parts, and hh.n.m to hash the headers of the MIME part as well?

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.

The presence of a recipe replaces the previous value, so the tag-value b= creates an empty body, while the tag-value b=b: creates a message with a single blank line for the body.

Prefix	Value	Action
c:	start-end	Copy the lines (inclusive) numbered from 1. E.g. "c:1-3" copies the first three lines.
b:	base64	Decode the base64 to get the value of a line to insert. The base64 value does NOT include the trailing CRLF, which must be added at the end of the line, hence "b:" means insert an empty line, and "b:SGVsbG8=" inserts the line "Hello".
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.

The presence of a recipe removes every instance of the named header field before applying the recipe, so the tag-value h.foo= removes all instances of Foo from the message.

Prefix	Value	Action
c:	start-end	Copy the values of the header-fields with the indexes, numbered from 1 and starting at the bottom;
b:	base64	Decode the base64 to get the value of a header field to insert. The base64 value does NOT include the trailing CRLF, which must be added at the end of the line, hence "b:" means insert a header-field with an empty value.
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

NOTE: the headers are inserted from the bottom, i.e. prepended to the message in the order that they are named.

e.g. given a message with headers:

Foo: three
Foo: two
Foo: one

And the recipe: h.Foo=c:1-1,b:Zm91cg==,c:2-3

The output will be: ~~~ Foo: three Foo: two Foo: four Foo: one ~~~

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
Mail-Version: mv=2;
 h.Subject=b:QSByZXBsYWNlbWVudCBmb3IgREtJTQ==;
 h.From=b:YnJvbmdAZmFzdG1haWx0ZWFtLmNvbQo=;
 h.Reply-To=
```

Example:

```
Mail-Version: mv=3; b=c:1-500,c:520-520
```

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

```
Mail-Version: mv=4;
 b=c:1-500,
   b:PGEgaHJlZj0iaHR0cHM6Ly93d3cuZXhhbXBsZS5jb20iPkV4YW1wbGU8L2E+Cg==,
   c:501-702
```

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

4. Security

Since a Mail-Version header can be used to recreate any email content, implementations need to be aware that this could be used to bypass security checks, and passing the generated message to a parser could expose it to content that would otherwise be blocked by earlier security checks, e.g. the base64 output could generate 8 bit content or NULL bytes that would otherwise be blocked by a simple filter.

5. IANA Considerations

IANA is requested to add to the Permanent Message Header Field Names registry the following record.

* Header Field Name: Mail-Version
* Template:
* Protocol: mail
* Status: standard
* Trace: no
* Reference: this document

6. Normative References

[DKIM] 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>.

[IMAP] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/rfc/rfc9051>.

Appendix A. Changes from Earlier Versions

A.1. draft-gondwana-dkim2-mailversion-00

* Rename this draft to "mailversion".
* Rename the header field to ’Mail-Version’
* Change ’v=’ to ’mv=’ so it matches DKIM2-Signature and doesn’t confuse protocol versions
* Removed ’t:’ optional fields entirely.
* Added ha= to select the hash algorithm and documentation about how the hashes are calculated
* Renamed bin.partspec to ph.partspec for "part hash" and described it with reference to IMAP.

* Added security considerations

* Added IANA considerations

A.2. 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.3. 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.4. 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.5. 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.6. 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