Workgroup: Independent Stream
Internet-Draft:
draft-chuang-mailing-list-modifications-04
Published: 20 February 2024
Intended Status: Experimental
Expires: 23 August 2024
Authors: W. Chuang
Google, Inc.

Tolerating Mailing-List Modifications

Abstract

Mailing-lists distribute email to multiple recipients by forwarding and potentially modifying messages to document the distribution to the recipients. Unfortunately forwarding breaks SPF (RFC7208) authentication and message modification breaks DKIM (RFC6376) authentication. This document is based on ARC (RFC8617) to provide a framework to describe forwarding with extensions to tolerate common mailing-list message modifications. This specification characterizes the mailing-list transforms such that a receiver can reverse them to enable digital signatures verification and attribution of the message content. These message modifications are: 1) adding a description string to the Subject header, 2) rewriting the From header, 3) removing the original DKIM-Signature and 4) appending a footer to the message body. This also specifies those modifications for the purpose of making them reversible.

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 23 August 2024.

Copyright Notice

Table of Contents

1. Introduction

Mailing-lists have long complicated email authentication. They break SPF [RFC7208] authentication due to forwarding and break DKIM [RFC6376] authentication due to message modification that used to identify the mailing-list. This specification provides methods to restore authentication even in the presence of forwarding and message modification. Being able to restore authentication is particularly important as senders may specify a sender-defined receiver email handling policy that may prevent delivery of the message as defined in DMARC [RFC7489]. Moreover malicious content is currently attributed to all parties in the mail flow. This specification permits a receiver to attribute the email content to its author be it the sender or the mailing-list.

The approach in this document is to be highly opinionated about only supporting common case mutations to lessen the implementation burden upon mailing-lists and receivers. This also seeks to eliminate any burden for messages that don't go through mailing-lists. At

origination, it is sufficient to sign a message with a DKIM signature as typically done already. It does ask those mailing-lists that wish to use this specification to characterize the modifications it performs at each forwarding step. The goal is to permit receivers to reverse the modification so that the message hash can be recovered and the prior signature verified at each forwarding step, be it from DKIM or ARC. This allows the receiver to determine which forwarder or the originating sender contributed which content, in the received message. This specification uses the ARC [RFC8617] framework to describe each forwarder, and then record the mutation performed at each forwarding step. Consequently this attribution enables more precise reputation systems and UI features. The supported modifications are: 1) adding a description string to the Subject header, 2) rewriting the "From" header, 3) rewriting the original DKIM-Signature and 4) appending a footer to the message body. This document specifies the characterization of the modifications and how to apply the modifications.

The validation results in this specification are orthogonal to the results in

draft-chuang-replay-resistant-arc. In addition to better supporting DMARC in the presence of mailing-list modifications, this specification enables attribution of malicious content back to the author. However this specification is vulnerable to replay much like DKIM and ARC. draft-chuang-replay-resistant-arc validation is tolerant of header and message body modifications but unable to provide attribution. It also detects and prevents replay. A receiver may want to combine the results of these two validations to create a strong authentication result that provides greater certainty that a message was sent by some originating sender through a mailing-list.

This specification assumes that originators and forwarders are named by the methods specified in the internet-draft draft-chuang-identifying-email-forwarding. That specification calls for a naming method of participants in the mail flow using a mail flow tag. That also names the ADMDs by the signer domain as defined in the DKIM-Signature "d=" SDID domains, and the ARC-Seal and ARC-Message-Signature "d=" SDID domains.

1.1. Terminology and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Algorithm

The specification in this document is divided into procedures computed at the mailing-list forwarders and at the receiver and is based on the procedures in ARC [RFC8617]. The steps at the forwarder are further divided at the forwarder into the inbound validation procedures and outbound forwarding procedures. The receiver only performs the inbound validation procedures. The inbound validation procedures are variable length corresponding to the depth of the ARC sets. At each ARC set corresponding to some earlier forwarder, the receiver computes any reversing procedure described later for any described mailing-list mutations to obtain the message hash of the message as seen at the inbound of the mailing-list. This hash comes from an ARC-Message-Signature if the sender is a forwarder and a DKIM-Signature (or possibly ARC-message-signature) if the sender is the originator. That signature MUST verify correctly, otherwise this is considered a signature verification failure by this specification. Interpretation of that failure is described later.

1.2.1. Forwarder Characterization

Conformant forwarders characterize message mutations to enable receivers to interpret mutations and potentially verification failures. A description is added as a tag-value to the ARC-Message-Signature mail flow description tag "m" as defined in the internet-draft draft-chuang-identifying-email-forwarding. The following mail flow values are considered mutators:

mailing_list: Describes a forwarder that distributes messages to multiple recipients and any modification conforms to the specification in this document.

ofs: Outbound-Filtering-Service is a mutating forwarder that works on behalf of the originating sender. The sender is presumed to have delegated responsibility to sign on behalf of the originating sender.

ifs: Inbound-Filtering-Service mutating forwarder that works on behalf of the final receiver. The receiver is presumed to have an established trust model with the gateway outside of this document and, and validation failure result is interpreted by the receiver.

Receivers MAY use the ARC-Message-Signature d= SDID domain to identify trusted forwarders. Receivers SHOULD NOT use the mail flow tag-value description to detect the existence of such trust relationships, as those descriptions exist for human operators to interpret gateway's actions.

1.2.2. Header Rewriting Characterization

A mailing-list forwarder that rewrites a Subject, From or DKIM-Signature headers can store the prior values by substituting with a modified header name i.e. prefixing "X-Prior-" to the original name. Moreover the header name retains the original capitalization, its place with respect to other headers, and is not ambiguous if there are multiple headers of the same name. First the bottoms-up line count of all headers is computed starting from the 0th position. Then all headers to be rewritten are collected, ordered by position to be prepended to the headers list and given a bottoms-up line count position. This is an offset from "X-Prior-" header to it's rewritten counterpart. It is assumed that the prior header has the following format:

```
<header name>:<value including any whitespaces>
```

The conserved header name is prefixed with a "X-Prior-". Then an ARC instance number tag "i" i.e. "=<#>," and line count tag "l" i.e. "l=<bottoms-up line count offset>" are prepended to the header value and separated by a semicolon and one whitespace. The position of this "i" and "l" is important because they may otherwise be ambiguous with the original header's tag-values e.g DKIM-Signature. Thus the conserved header is:

```
X-Prior-<header name>: i=<#>; l=<bottoms-up line count offset>;
     <value including any whitespaces>
```

For example:

```
X-Prior-Dkim-Signature: i=1; l=4; d=example.com; b=...
X-Prior-Subject: i=1; l=3; Meeting on the 5th
```

This stores the prior header in a format that SHOULD be ignored by subsequent forwarders, and if not then will cause a verification failure. The forwarder then rewrites or deletes the header as before by prepending the rewritten header to the message or skipping prepending for deletion. To protect the integrity of these headers, these headers are hashed separately from ARC-Message-Signature "h" header list. The forwarder generates the header hash by hashing the "X-Prior-" headers (and any "Content-Footer" header described below) found in bottoms up order as found in the headers. This specification calls for the header hash to be explicitly added to the ARC-Message-Signature with a tag "fh" as defined in the internet-draft draft-chuang-replay-resistant-arc. Conformant forwarders even without mailing-lists mutations MUST report header hash explicitly to better differentiate and tolerate body modifications when verifying headers.

To reverse the rewriting or deletion of headers, all "X-Prior-" headers are collected at the given ARC instance number. The prior header name is extracted from the "X-Prior-" header, and the original value extracted and restored. The rewritten header is found by taking the "X-Prior-" header line count position and adding the offset in the "l" tag's value. Rewritten headers are deleted in top down order.

1.2.3. Message Footer Characterization

Mailing-lists may choose to add a message footer to the body of the message to notify the recipient of the mailing-list distribution action and conformant mailing-list will describe this mutation to the receiver. One method is to append the footer text to all content-type text in the top level of MIME [RFC2045] part "multipart/alternative". Another method is to append a new successor MIME part for a "multipart/mixed" part, and that MIME part contains the footer as a content-type text. If the message is not mime formatted, the entire body is treated as text and the mailing-list may append a text footer at the bottom of the message.

Conformant mailing-list forwarders characterize these changes with a new "Content-Footer" header. It contains a list of semicolon and whitespace separated tag-values using the formatting specifications in DKIM. The first of these is an ARC set instance number describing which forwarder added the header. When the mailing-list appends a text part, the receiver characterizes it with the beginning and ending byte offset of the MIME part or message-body if MIME unformatted. If the mailing-list adds a text MIME part as footer, it describes the mime part with its purpose tag-value. The Content-Footer header is added to the MIME part for MIME formatted messages, and to the message headers for unformatted messages. To protect the integrity of the header, in the latter case, the Content-Footer is hashed along with "X-Prior-" headers in the same bottoms up order. In the former case, Content-Footer headers added to the message-body are protected by the message body hash. The following are the Content-Footer header tag-values:

i Instance tag. Must be the first tag-value. The value contains the instance number

b Footer beginning octet offset from the start of the text MIME part or the message-body if no MIME unformatted. MUST always appear with a footer ending offset.

e Footer ending octet offset from the start of the text MIME part or the message-body if no mime unformatted. MUST always appear with a footer beginning offset.

m

Mime-part purpose description. If the added MIME part contains the text footer, then the value is "footer". If the added MIME part is "multipart/mixed" and meant to support adding the text footer, then the value is "mixed". If the text footer is appended, then this tag-value is not added.

For example:

```
Content-Footer: i=1; b=100; e=200
Content-Footer: i=2; m=footer
```

To reverse the footer addition, the receiver looks at the headers to see if the message is MIME formatted. If not, the receiver looks for a "Content-Footer" in the headers. If so, the receiver looks for "Content-Footer" in the MIME tree content headers. For each "Content-Footer" found header at the current ARC set instance, it reverses the footer mutation. For offset values designated by beginning and ending offset tags, it removes the text from that MIME part or the message-body designated. For a MIME part text footer, it deletes it along with any supporting "multipart/mixed" MIME part. Then the receiver deletes the Content-Footer header from the updated message if not done so already.

1.2.4. Validation

The receiver verifies prior ARC sets per the procedure described in ARC [RFC8617]. In addition, the receiver validates the ARC sets starting from the largest instance number found to the smallest. First the receiver verifies the given instance ARC-Message-Signature or DKIM-Signature as appropriate. Then the receiver computes the rewritten header hash taking the header hash computed by ARC-Message-Signature at the given instance number or DKIM-Signature. Then the hash is taken for "X-Prior-" headers and "Content-Footer" header if found in the headers at the current ARC set instance number and all prior ones. This is verified against the header hash value associated with the tag "fh", reporting signature failure if it mismatches.

Next the receiver determines whether it needs to reverse any header or footer mutations at that ARC set instance by looking for the ARC-Message-Signature mutation tag "m=". For values of "mailinglist", it attempts to reverse mutations keeping the resulting message so that further validations are possible. The receiver attempts to provide header reversing procedures given in "Header Rewriting" section and body reversing procedures given in "Message Footer" section. For values "gateway" the receiver MAY apply local policy to interpret subsequent validation failures. For all other mutation tag "m"

values, it assumes no mutations are present or outside the scope of mailing-list modifications.

In addition, the originating sender's DKIM-signature or ARC-Message-Signature MUST successfully verify. If so and all prior signatures verify, then the result is a "pass". Verification failures are subject to interpretation in "Footer Characterization" section, and potentially indicate a "fail". The result of this procedure is written in Authentication-Result [RFC8601] and ARC-Authentication-Result with a method named "reverse" as the REVERSE result.

Informationally, if the receiver implements draft-chuang-replay-resistant-arc, this specification suggests modifying draft-chuang-replay-resistant-arc PATH results to take into account the REVERSE result. At each ARC set instance where PATH recursively combines the local DARA results, if REVERSE reports "fail" then the PATH result reports "fail". Because *REVERSE _combined with DARA* _represents a higher bar of verification than DARA alone, the receiver applies local policy when interpreting the PATH result.

1.3. Mailing-List Modifications

When a message body or Subject header is modified by a forwarder, the sender's DKIM signature will no longer validate. To mitigate this forwarders MAY elect to replace the DKIM signature with their own with a new message hash that takes into account the modifications. Because the signature is not DMARC aligned, senders also MAY rewrite the From header to take ownership of the message. The following creates a specification making the message body or Subject header modification, replacing the DKIM signature and applying characterization headers.

1.3.1. Subject Header Modification

The mailing-list may want to communicate to the recipient that a message went through a mailing-list by modifying the Subject header to append the name of the mailing-list. Typically the name is put between brackets e.g. "[name]" and prepended to the Subject header content. This specification supports arbitrary text changes by saving the earlier inbound version of the Subject header's content in the "X-Prior-Subject" header. The change in Subject text will break the DKIM-Signature so mailing-lists MAY rewrite DKIM-Signature to update the message header hash and resign the signature. Similarly they MAY update the From header to DMARC align the DKIM-Signature "d=" domain with the From header domain. Typically the From address is the set to the mailing-list address to further identify the mailing-list. This specification supports saving the earlier inbound version of the DKIM-Signature and From headers in

the "X-Prior-DKIM-Signature" and "X-Prior-From" headers respectively.

For example the original message looks like:

```
Dkim-Signature: d=example.com; b=...
From: john.doe@example.com
Subject: Meeting on the 5th
```

A mailing-list rewrite Subject, From and DKIM-Signature headers and saves the original content in the "X-Prior-" headers

```
Dkim-Signature: d=mailinglist.example.com; b=...
From: mailinglist@mailinglist.example.com
Subject: [mailing-list] Meeting on the 5th
X-Prior-Dkim-Signature: i=1; l=3; d=example.com; b=...
X-Prior-From: i=1; l=3; john.doe@example.com
X-Prior-Subject: i=1; l=3; Meeting on the 5th
```

1.3.2. Body Modification

The mailing-list may want to communicate to the recipient that a message went through a mailing-list by adding a text footer describing the mailing-list. Typically this is done by appending that text description at the bottom of message body text. This specification supports three different footer organizations, depending on the MIME structure, content type and content transport encoding of the MIME parts of the message. In particular the organization depends if its MIME parts can be concatenated, meaning whether a text footer can be appended and removed without having to re-encode the original content to preserve the original content. For example appending new content to "base64", and "binary" will likely introduce artifacts between the original message and the interpretation footer. MIME parts that can be concatenated are defined as having one of the following type and subtype of Content-type:

- *text/plain
- *text/html

and one of the following mechanisms for Content-transfer-encoding:

- *7bit
- *8bit
- *quoted-printable

Similarly, non-text MIME parts, and complex multipart MIME parts don't lend themselves to appending text. One exception to this is "multipart/alternative" as will be described shortly. To handle these scenarios, this specification calls for adding a new text MIME part footer that can handle any encoding or any mime structure but requires altering the MIME tree.

The following procedure describes how the mailing-lists MAY choose from one of those formats to add a footer. Here a mailing-list should evaluate in the same order as these three sections, following the steps in each section. If any footer addition is successful, then the footer algorithm stops.

1.3.2.1. Unstructured or Text Message Body

The simplest structure applies when the message lacks [RFC2045] MIME structure or the top level MIME part can be concatenated. Here the forwarder appends a text footer with the same content type and content transfer encoding as the message-body. When a message lacks MIME struct, the default message content-type is "text/plain" (section 5.2) and default content transfer encoding is "7bit" (section 6.1). Next a "Content-Footer" header is prepended describing the start and ending message body octet offsets. Reversing this transform is a straightforward deletion of the footer at the offsets given.

For example given an original message of:

```
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable
From: john.doe@example.com

This is the message body.
```

A text footer can be appended as follows:

```
Content-Footer: i=1; b=25; e=67
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable
From: john.doe@example.com

This is the message body.

============
This is a mailing-list footer.
```

1.3.2.2. Multipart/Alternative

When the top level MIME part is Content-type "multipart/alternative", the forwarder checks if any of the immediate children

MIME parts can be concatenated. If so, it attempts to append a text footer with the same content type and content transfer encoding as the children MIME part. Next it prepends a Content-Footer header in the child MIME part header description with the start and end octet offsets from the beginning of the part. If a footer can be added to a child MIME part, then this is considered success and the algorithm can halt. The reversing algorithm is to look for the top level MIME part with Content-type "multipart/alternative". If so then look in the immediate child MIME parts and delete the text at the offsets given by the Content-Footer.

For example given an original message of:

```
Content-Type: multipart/alternative; boundary="abcd"
From: john.doe@example.com

--abcd
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable

This is the message body.

--abcd--
```

A text footer can be appended as follows:

```
Content-Type: multipart/alternative; boundary="abcd"
From: john.doe@example.com

--abcd
Content-Footer: i=1; b=25; e=67
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable

This is the message body.

============
This is a mailing-list footer.

--abcd--
```

1.3.2.3. MIME Part Footer

The following procedure adds a MIME "multipart/mixed" at the top level and has as its children the original top level MIME part as the first child and the footer MIME text part as the second child. This procedure will always succeed no matter the MIME structure or MIME content transfer encoding, however this modifies the MIME tree. The top level MIME "multipart/mixed" is denoted by adding a "Content-Footer" header purpose description with "mixed" value and

the footer MIME part denoted by adding a "Content-Footer" header purpose of "footer" value. To reverse the mutation, the original top-level mime part is moved back to the actual top-level and delete the "multipart/mixed" and the footer MIME parts.

For example given an original message of:

```
Content-Type: multipart/mixed; boundary="abcd"
From: john.doe@example.com

--abcd
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: base64

VGhpcyBpcyB0aGUgbWVzc2FnZSBib2R5Lg==

--abcd--
```

A "multipart/mixed" and text footer MIME parts can be appended as follows:

```
Content-Footer: m=mixed
Content-Type: multipart/mixed; boundary="efgh"
From: john.doe@example.com

--efgh
Content-Type: multipart/mixed; boundary="abcd"

--abcd
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: base64

VGhpcyBpcyB0aGUgbWVzc2FnZSBib2R5Lg==

--abcd--
--efgh
Content-Footer: m=footer
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable
============
This is a mailing-list footer.

--efgh--
```

1.4. Examples

These examples are informational.

1.4.1. Originator ⇒ First Mailing-List ⇒ Second Mailing-List ⇒Receiver

This message is sent through two mailing-lists to some receiver. The originating sender creates and signs a message as follows:

```
DKIM-Signature: d=example.com
From: john.doe@example.com
Subject: A really big announcement

It's Jane Doe's birthday tomorrow!
```

The first mailing-list adds a Subject header prefix and message-body footer, and denotes this using the procedures and headers specified in this document. It denotes that it performed mailing-list mutations in the ARC-Message-Signature.

```
ARC-Message-Signature: i=1; m=mailinglist...
Content-Footer: i=1; b=34; e=78
DKIM-Signature: d=mailinglist.example.com...
From: school@mailinglist.example.com
Subject: [school list] A really big announcement
X-Prior-DKIM-Signature: i=1; l=3; d=example.com...
X-Prior-From: i=1; l=5=3; john.doe@example.com
X-Prior-Subject: i=1; l=3; A really big announcement

It's Jane Doe's birthday tomorrow!

============
This is the school mailing-list.
```

The second mailing-list adds a Subject header prefix and message-body footer, and denotes this using the procedures and headers specified in this document. It denotes that it performed mailing-list mutations in the ARC-Message-Signature.

```
ARC-Message-Signature: i=2; m=mailinglist...
Content-Footer: i=2; b=79; e=124
DKIM-Signature: d=mailinglist.example.com...
From: district@mailinglist.example.com
Subject: [district list] [school list] A really big announcement
ARC-Message-Signature: i=1; m=mailinglist...
Content-Footer: i=1; b=34; e=78
X-Prior-DKIM-Signature: i=2; l=5; d=mailinglist.example.com
X-Prior-From: i=2; l=5; school@mailinglist.example.com
X-Prior-Subject: i=2; l=5; [school list] A really big announcement
X-Prior-DKIM-Signature: i=1; l=3; d=example.com
X-Prior-From: i=1; l=3; john.doe@example.com
X-Prior-Subject: i=1; l=3; A really big announcement

It's Jane Doe's birthday tomorrow!

============
This is the school mailing-list.

============
This is the district mailing-list.
```

The receiver sees the above message on inbound delivery, and attempts to verify the ARC message signature at the i=2. Upon success, the receiver notices that there were mailing-list mutations at ARC set i=2. It applies the REVERSE validation algorithm to reverse the mutations from the second mailing-list. After applying the reverse procedure, the reversed message looks like:

```
ARC-Message-Signature: i=2; m=mailinglist...
ARC-Message-Signature: i=1; m=mailinglist...
Content-Footer: i=1; b=34; e=79
DKIM-Signature: d=mailinglist.example.com
From: school@mailinglist.example.com
Subject: [school list] A really big announcement
X-Prior-DKIM-Signature: i=1,l=3, d=example.com
X-Prior-From: i=1; l=3; john.doe@example.com
X-Prior-Subject: i=1; l=3; A really big announcement

It's Jane Doe's birthday tomorrow!

============
This is the school mailing-list.
```

Again the receiver attempts to verify the ARC message signature now at the i=1. Upon success, the receiver notices that there were mailing-list mutations at ARC set i=1. It applies the REVERSE validation algorithm to reverse the mutations from the first mailing-list, obtaining the original message. The reversed message looks like:

```
ARC-Message-Signature: i=2; m=mailinglist...
ARC-Message-Signature: i=1; m=mailinglist...
From: john.doe@example.com
DKIM-Signature: d=example.com...
Subject: A really big announcement

It's Jane Doe's birthday tomorrow!
```

   The resulting reversed headers and message body DKIM-Signature verifies, and the REVERSE passing result is published in the ARC-Authentication-Result:

```
ARC-Authentication-Result: i=3; mx.example.com; reverse=pass...
ARC-Message-Signature: i=2; m=mailinglist...
Content-Footer: i=2; b=79; e=124
DKIM-Signature: d=mailinglist.example.com...
From: district@mailinglist.example.com
Subject: [district list] [school list] A really big announcement
ARC-Message-Signature: i=1; m=mailinglist...
Content-Footer: i=1; b=34; e=78
X-Prior-DKIM-Signature: i=2; l=5; d=mailinglist.example.com...
X-Prior-From: i=2; l=5; school@mailinglist.example.com
X-Prior-Subject: i=2; l=5; [school list] A really big announcement
X-Prior-DKIM-Signature: i=1; l=3; d=example.com...
X-Prior-From: i=1; l=3; john.doe@example.com
X-Prior-Subject: i=1; l=3; A really big announcement

It's Jane Doe's birthday tomorrow!

============
This is the school mailing-list.

============

   This is the district mailing-list.
```

1.5. Security Considerations

Care must be used if the reversed transformed message is used for authentication. The reversed transformed message is vulnerable to replay attacks and "hides" the potentially spammy contribution from the mailing-lists.

1.6. IANA Considerations

There are no requests at this time.

2. 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/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>.

[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>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/rfc/rfc7208>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/rfc/rfc7489>.

[RFC8601] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", RFC 8601, DOI 10.17487/RFC8601, May 2019, <https://www.rfc-editor.org/rfc/rfc8601>.

[RFC8617] Andersen, K., Long, B., Ed., Blank, S., Ed., and M. Kucherawy, Ed., "The Authenticated Received Chain (ARC) Protocol", RFC 8617, DOI 10.17487/RFC8617, July 2019, <https://www.rfc-editor.org/rfc/rfc8617>.

Author's Address

Weihaw Chuang
Google, Inc.

Email: weihaw@google.com