Workgroup: Independent Stream
Internet-Draft:
draft-chuang-replay-resistant-arc-11
Published: 20 February 2024
Intended Status: Experimental
Expires: 23 August 2024
Authors: W. Chuang
Google, Inc.
B. Gondwana
Fastmail Pty Ltd

Replay Resistant Authenticated Receiver Chain

Abstract

DKIM (RFC6376) is an IETF standard for the cryptographic protocol to authenticate email at the domain level and protect the integrity of messages during transit. Section 8.6 defines a vulnerability called DKIM Replay as a spam message sent through a SMTP MTA DKIM signer, that then is sent to many more recipients, leveraging the reputation of the signer. We propose a replay resistant cryptographic based protocol that discloses all SMTP recipients and signs them, allowing a receiver or any third party to verify that the message went to the intended recipient. If not then then potentially the message is replayed. Moreover it leverages ARC (RFC8617) and sender defined forwarding path to build a "chain of custody" that accurately defines the SMTP forwarding path of the message. This also allows the protocol to detect DKIM and ARC replay attacks and other attacks.

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.

Table of Contents

1. Introduction

This protocol provides a technique to authenticate email by domain that is replay resistant. It leverages the features of ARC to name ADMD in the email forwarding path and to publish the intermediate results. It then discloses all SMTP recipients as signed RFC822 headers by the sender which allows a receiver to verify if the mail was directed to the appropriate recipient. The results MAY be used by spam filtering to apply some local policy, and/or applied to DMARC policy evaluation as one of its input email authenticators.

Existing email authentication techniques have known limitations. DKIM suffers from being vulnerable to replay attacks as described in draft-ietf-dkim-replay-problem. Spammers utilize an account on a sender that supports signing with DKIM to capture a spammy message with a valid DKIM signature. The spam is then broadcast to many

victim recipients. Because ARC is based on DKIM signing, ARC is similarly vulnerable to replay.

The broader goals of this internet-draft are outlined here:

* Any party can independently verify that a message traveled along a path as intended by the originator (original sender) to the receiver (last receiver). This prevents DKIM and ARC replay attacks, and SPF shared tenancy attacks.

* Receivers can determine if the message was modified along the path and by whom.

* Be able to tolerate parties not participating in the new protocol. Make sure to have reasonable partial failure modes for non-participating parties including incentives to encourage future participation.

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].

1.1.1. Definitions

SMTP transport and particular email senders and receivers are defined in [RFC5321]. Email payload and headers are defined in [RFC5322]. This document uses [RFC5598] email flow definitions, which describes the interactions between the parties in sending email. In particular these parties assist the email senders and receivers in email transport. draft-ietf-dkim-replay-problem adds context to those mailflows for the DKIM replay problem.

1.1.1.1. Acronyms

ADMD: ADministrative Management Domain is defined as [RFC5598] and represents the independent operational scope authorship, handling, and receiving.

ARC: Authenticated Received Chain [RFC8617] - is a protocol that is meant to resolve some of the issues for DMARC [RFC7489] to fix the problems that DMARC policy rejects caused by mail forwarding. ARC uses a digital signing mechanism derived from DKIM to protect the integrity of the Authentication-Results of a forwarder and a

versioning mechanism to describe the forwarders. ARC suffers from similar replay issues as DKIM.

Authentication-Results: A header containing a list of email authentication validation methods, results and comments as specified in [RFC8601].

DKIM: DomainKeys Identified Mail [RFC6376] standard for the cryptographic protocol to authenticate email at the domain level and protect the integrity of messages during transit.

DKIM replay As defined in [RFC6376] section 8.6- a vulnerability called DKIM Replay as a spam message sent through a SMTP MTA DKIM signer, that then is sent to many more recipients, leveraging the reputation of the signer.

DMARC: Domain-based Message Authentication, Reporting, and Conformance [RFC7489]- defines a sender defined message handling policy for spoofed messages to be applied when a message is delivered at some receiving SMTP server.

MDA Message Delivery Agent defined in [RFC5598].

MSA Message Submission Agent defined in [RFC5598].

MTA Message Transfer Agent defined in [RFC5598].

SPF: Sender Policy Framework [RFC7208] standard for authenticating sending servers typically based on IP address.

1.2. Defining and Propagating Sender Identity in Mail Flow

This section outlines how ARC and DKIM are used by the email authentication methods defined in this document, though several details are left for later sections. This protocol leverages ARC and DKIM for declaring protocol settings and protecting the integrity of the headers and message body, and ARC for propagating authentication results. At message origination, this uses DKIM-Signature tag/values for declaring settings and optionally ARC-Seal tag/values instead. For message forwarding, this uses ARC-Seal tag/values for declaring settings. After the email receiver evaluates the email authentication results, these results are published and propagated to the subsequent receivers via ARC-Authentication-Results. This protocol updates ARC-Authentication-Results with new method status, properties and comments as defined in Section 1.3.

This protocol identifies and names the ADMDs by the signer domain as defined in the DKIM-Signature "d=", and the ARC-Seal and ARC-Message-Signature "d=" SDID as described in the internet-draft draft-chuang-identifying-email-forwarding. The traversed MAIL FLOW

forwarding path is defined as a vector of these domains, and is further defined in Section 1.4.

This specification mandates that the ADMDs participating in this protocol explicitly identify themselves with a DKIM-Signature or ARC-Seal tags "dara" or "darn". At the originating sender, participants MAY declare participation with a tag in the DKIM-Signature if the recipient declaration and signing as described later is covered by To and Cc, otherwise they MUST declare with a tag in the ARC-Seal. Later this document will describe when to use X-Signed-Recipient (formerly Forwarded-to) header which is protected by the ARC-Seal. Additionally if the message is forwarded, participation MUST be declared in a tag in the ARC-Seal. Participants will declare an identified path of ADMD nodes from the originating sender ADMD to the receiving ADMD with the "dara" tag. If the message exits the identified path into some naive, protocol unaware ADMD the aware ADMD denotes this using the "darn" tag, allowing for mitigations for this scenario. The tags and their use are further specified in Section 1.3.1.2.

This protocol REQUIRES that the *From* header domain MUST align with DKIM-Signature "d=" domain or ARC-Message-Signature "d= domain at the starting ADMD in the mail flow path thereby ties the From identity to the cryptographic signer as described in draft-chuang-identifying-email-forwarding. This allows any receiver or third party to verifiably determine that the message was sent by the signer. This ADMD defines the MSA ADMD, i.e. the responsible originating sender. Some forwarders such as mailing-lists modify the message and to prevent DMARC misalignment, they resign the message with their own DKIM signature and rewrite the From header aligned to their domain. From header rewriting hinders discovering the original sender. As this protocol is insensitive to message message modification, forwarders using this protocol MAY choose not to From header rewrite or resign the message with DKIM when they detect the receiver supports this protocol. Receivers may consider applying methods to recover the originating sender's From header by using methods such as draft-chuang-mailing-list-modifications.

If the originating sender performs ARC signing, the ARC the ARC-Authentication-Results MUST be empty as results will otherwise be non-sensical. When the message is delivered to the inbox by the MDA, it MAY strip the ARC-Seal and ARC-Message-Signature but leave behind the ARC-Authentication-Result. Partially stripping the ARC set makes termination identifiable and more difficult to replay as signatures are missing. A message lacking ARC-Seals and ARC-Message-Signatures but containing ARC-Authentication-Result has been delivered to the inbox. Seeing such a message in delivery may be replayed and is denoted by an ARC verification *fail* status.

This protocol protects against malicious use of these ARC headers by REQUIRING message signing and verification between ADMDs. In addition there MAY be ARC signing and verification internal to the ADMD. Having this outbound message body signing invariant permits the receiver to verify the integrity of the message as sent by the prior ADMD. To verify the integrity of the ARC sets then, a receiver MUST verify the previous ARC set's ARC-Message-Signature and verify each ARC set's ARC-Seal signature from "i=N" (receiver's ARC set number) to "i=1" (originating sender or first forwarder) as well as the presence of all headers in the ARC set as defined in [RFC8617]. If the receiver sees a verification failure from the immediate sender's "i=N-1" ARC-Message-Signature, this MUST result in an ARC verification *fail* status. ARC-Message-Signature verification failures from "i=N-2" to "i=1" are tolerated, meaning their failure does not indicate a failing ARC result e.g. mailing-list modification. All ARC-Seal verification failures from "i=N-1" to "i=1" are treated as ARC verification *fail* status. The result of the verification is published in the Authentication-Result and the ARC-Authentication-Result with a tag "arc=". Even if the receiver notes that a prior receiver publishes a ARC verification fails, this specification asks the receiver to continue ARC generation and verification to provide forensics evidence via the ARC-Authentication-Results. For example the SPF authentication results of the potentially malicious sender MAY help identify that sender to some subsequent receiver. The propagated ARC verification failure will help prevent inadvertent use of the authentication results by subsequent receivers.

1.3. Declare All Recipients and Affirm (DARA)

1.3.1. Concepts

This email authentication protocol uses validating signed headers against the envelope headers. It features a looks up mechanism to support forwarders that are unaware of the protocol. Also it publishes enough information for a third party to independently validate the results given by SMTP sender and receiver.

1.3.1.1. Declaring All Recipients

The specified email authentication protocol is resistant against replay attacks by explicitly identifying all recipients in the headers, including when the recipient is "hidden" such as *Bcc:* or Mailing-lists. That way when a signed message arrives, the receiver can check if the RCPT TO recipient correctly is a subset of the recipient in the signed message header. If not, then the message MAY be part of a replay attack. When To: and Cc: recipients are declared by their headers, they MUST be specified in the "h=" header list and signed by DKIM-Signature or ARC-Message-Signature. For blind carbon

copy, while a Bcc: header might be added, it can be stripped by subsequent forwarders. Instead we create a new _X-Signed-Recipient: _ header that includes an ARC set versioning number to indicate which ADMD sent the message to a new (formerly Forwarded-to:) recipient. It MAY include one or more comma separated recipients. Whitespaces in the recipient list are ignored. The local part of the address may be obfuscated so long as it's consistently done so that 3rd party membership verification can be done.

```
X-Signed-Recipient: i=1; user@example.com, user2@example.com
```

As part of the DARA protocol, recipients not declared by To: or Cc: MUST be declared with the *X-Signed-Recipient:* header. This supports the email forwarder and mailing list scenario where we also use the *X-Signed-Recipient* header to indicate that a message is sent to a new recipient. *X-Signed-Recipient: _MUST be propagated by forwarders unmodified. For the privacy of "hidden" recipients and to prevent their identity from being visible to other recipients via the _X-Signed-Recipient: header*, the message MUST be split and signed exclusively for each *X-Signed-Recipient:* recipient. This means the header is visible only to that recipient. Messages sent to a new ADMD but with the same recipient identity disclosed by a prior *X-Signed-Recipient* MAY elect to optimize header space by skipping adding a redundant *X-Signed-Recipient* header.

To protect the integrity of the *X-Signed-Recipient:* header, they MUST be hashed and signed by ARC-Message-Signature as follows: Collect all *X-Signed-Recipient:* headers and hash them following the header processing algorithm in [RFC6376] section 5.4. Potentially there may also be additional X- headers from This hash is published in the ARC-Message-Signature header as "fh=" tag and base64 hash value. DARA aware verifiers can recompute the hash and check it against the hash contained in the "fh=" tag to verify the integrity of the *X-Signed-Recipient:* headers as well as the *To:* and *Cc:* headers. (Additional headers MAY be processed if the receiver is aware of draft-chuang-mailing-list-modifications) In addition hash the list of ARC-Message-Signature headers. For example:

```
ARC-Message-Signature: i=1; fh=abcd...
X-Signed-Recipient: user@example.com
```

If the originating sender uses a DKIM-Signature, the To and Cc headers MUST be present in the sender's DKIM-Signature, and signed.

1.3.1.2. Protocol Awareness

Senders and receivers MAY variously support the DARA protocol or not, so the protocol needs to be tolerant of ADMDs that don't support the protocol. For example a naive mailing list sender

sending to a protocol aware receiver SHOULD NOT have traffic rejected simply because it didn't follow the protocol. Yet simultaneously, the DARA protocol needs to discourage abuse by spammers seeking to use the naive ADMD path for replay. The protocol calls for the DARA aware senders to lookup the capability of the receiver in supporting DARA and disclose that capability in the message. All ADMD supporting the DARA protocol SHOULD publish a DNS TXT policy record. The DARA aware sender SHOULD look up the receiver's policy record as described next or look up an internal list of receivers that support DARA. The following paragraph describes the DARA DNS policy record and disclosure statement, and the following paragraph describes when the ADMD does not support DARA.

When the ADMD indicates it supports DARA via DNS, the ADMD publishes a DNS TXT policy record at the dara well known location based on the MX host domain. More specifically this specification calls for performing a MX lookup to obtain the derived hostnames. Take the highest priority hostname, and, prepended with a "_dara" label to find the dara well known domain that contains the DARA TXT policy record. The format of the DARA policy record are tag/values in form of the textual representation in [RFC6376] section 3.2. The policy record MUST start with a DARA version tag "v=" with a DARA version number that MUST be set to "DARA_1.0". The lookup also discovers the destination domain name, and that destination domain MUST match the ADMD's ARC-Seal "d=" signing domain [RFC8617] which enables tracing this domain *From* sender to receiver as described later. The signing domain name is specified by the tag "dara=" with value being that domain name. Once discovered, this domain is copied to "dara=<domain>" domain that is then placed in the sender's DKIM-Signature or ARC-Seal. The "dara=" tag/value indicates support by the receiver for the DARA as well as the identity of the intended receiver signing domain. The following is an example of a DARA DNS policy record for example.org that normalizes to example.com. The TXT record is published at *_dara.example.org* and contains:

```
v=DARA_1.0; dara=example.com
```

If no such DNS TXT policy record is found or not in the list of supported domains, then the receiver does not support the DARA protocol. This is indicated by the tag "darn=" with the receiving domain as the value. This is placed in the sender's DKIM-Signature or ARC-Seal. The "darn=" tag indicates to subsequent DARA aware receivers that there was an intermediate naive forwarder. Also, when there is spam, instead of penalizing the sender that is DARA aware, the receiver MAY elect to apply the reputation penalty to the receiving domain that is naive to DARA.

A DARA aware receiver MAY elect to check the sender's policy if it suspects that a malicious forwarder was acting as a Man-In-The-Middle and has stripped off some prior sender's DARA policy. If it detects a DARA declaration in the sender's DNS policy, but not declared in the message, the receiver MAY elect to treat the message as spam.

1.3.1.3. Receiver Verification

A DARA aware receiver looks in the message to determine how to do DARA validation. First it looks for the most recent ARC-Seal (if present) using the ARC set number to determine recency. If not present then it looks for a DKIM-Signature. When found, a DARA aware receiver verifies the integrity of the header, then looks for a DARA tag/values and these are interpreted as follows. If the tag is "dara=", then the receiver MUST validate the recipients, and if it fails verification, treat the message as DARA unauthenticated with the implication that the message might be replayed. The recipient verification process for a given forwarder is to collect all the recipients in the *To*, *Cc* and prior *X-Signed-Recipient* headers. In particular, for a forwarder i=n, the verifier collects all X-Signed-Recipient headers from i=1 to i=n-1. It verifies that they are signed appropriately and if not fails the verification. If the message only contains a DKIM-Signature, the verifier checks that the To and Cc headers are present in the DKIM-Signature "h=" header list, and signed. Otherwise it checks for the presence of the "fh=" tag in the ARC-Message-Signature. Next it checks the integrity of the X-Signed-Recipient headers by validating the "fh=" hash if present. The receiver collects all *To:*, *Cc:* and_ X-Signed-Recipient:_ headers and hash them following the header processing algorithm in [RFC6376] section 5.4, then checks the hash against the value associated with the "fh=" tag. (Additional headers MAY be processed if the receiver is aware of draft-chuang-mailing-list-modifications) In addition hash the list of ARC-Message-Signature headers. If this mismatches, this is treated as failing verification. Assuming headers integrity, the receiver then collects all the RCPT TO recipients as the envelope recipients. The receiver then verifies that the envelope recipients are a subset of the signed headers. If not a subset, this too is treated as failing verification.

As with other email authentication methods, the receiver's verifier is free to apply a locally defined policy against unauthenticated email. Next if the sender's tag is "dara=", the verifier SHOULD treat validation success as *pass*, and validation failure as *fail*. If the sender' tag is "darn=", the verifier SHOULD treat recipient verification failure as *neutral* and SHOULD treat success as *pass*. This discretionary validation mode is to support the scenario of DARA unaware ADMDs that may cause false positive validation

failures. The domain value associated with the "darn=" tag helps identify the naive ADMD in processing local policy.

After the receiver's verifier applies the "dara=" or "darn=" policy as described above, the result of this verification MUST be published in the ARC-Authentication-Results. The verifier describes the result with [RFC8601] method "dara", and a result value of *pass*, *fail* or *neutral*. Receivers MUST declare the RCPT TO identity of the user that accepted the delivered message as a property header.i=<recipient email address>. This is to enable 3rd party mail flow validation as will be described shortly. As above, the local part may be obfuscated so long as it's consistently done. For example the ARC-Authentication-Result could look like:

ARC-Authentication-Result: i=2; dara=pass header.i=user@example.com

1.3.1.4. 3rd Party Verification

A third party verifier MUST be able to verify that DARA results from the sender and receiver using only values in the message headers and DNS. First the verifier identifies the sender and receiver. The sender may be identified by ARC-Seal with an ARC set number preceding the receiver or DKIM-Signature if no prior ARC-Seal is discovered. The sender's "dara=" or "darn=" policy declaration in the ARC-Seal or DKIM-Signature. The receiver's results will be found in the ARC-Authentication-Results. For both the sender and receiver, the integrity of the headers are checked i.e. checking the ARC-Seal and then the "fh=" hash. If it passes, then verifier determines the sender's declaration of the receiver's DARA support, by looking for "dara=" tag in the DKIM-Signature or ARC-Message-Signature. The value of the "dara=" domain MUST match the receiver's ARC-Seal's "d=" domain, and the receiver's ARC seal MUST verify. The 3rd party verifier SHOULD also check to see if the ARC-Authentication-Result dara property "header.i=" is a subset of the declared and signed header so far. In addition, if the header.i domain address is the same as the ARC-Message-Signature d= domain, then it can be said that the forwarder is aligned. If these steps pass, the 3rd party verification *passes*. If verification at any individual fails, the 3rd party verification *fails*. The above procedure can later be used by the Chain verification algorithm in Section 1.4 to construct verification across multiple senders and receivers in the mail flow.

1.3.1.5. DMARC

These protocols can act as authenticators for DMARC [RFC7489]. As noted in the Section 1.4, the From header domain can be aligned with the DKIM-Signature d= domain and/or the ARC-Message-Signature "d=" domains. This helps identify the originator of the message, and can call this *originator aligned*. In addition, the specification says

that if the ARC-Authentication-Result dara property "header.i=" domain is the same as a ARC-Message-Signature d= domain, and if is properly a member of it's sender's declare recipient list, we can say the forwarder is properly identified. We can call this *forwarder aligned*. If the ARC-Message-Signature validates, then can call this *fully forwarder aligned*. If the message has a originator alignment, and each forwarder is aligned, then the message is aligned, and this specification calls for this result to be a DMARC authenticating result equivalent to SPF or DKIM.

1.3.2. Definitions

DNS TXT Policy tag/values

* "v=": Value of "DARA_1.0" if the ADMD supports the DARA verification protocol.

* "dara=": Value of the receiver's ARC-Seal "d=" domain

DKIM-Signature or ARC-Seal tags/values

* "dara=": Value of the receiver's ARC-Seal "d=" domain when the receiver is DARA capable as found from the DARA DNS policy record.

* "darn=": Value of RFC822 recipient's domain name when the receiver is naive of DARA.

ARC-Authentication-Results method

* "dara=": Value of *pass* if recipient validation passes, otherwise *fail*. In some circumstances this tag/value may be unset or be treated as *neutral*.

1.3.3. Examples

1.3.3.1. Originator ⇒ Mailing-List ⇒ Receiver

Originator outbound

```
DKIM-Signature: d=originator.example.com; dara=mailinglist.example.com
To: list@mailinglist.example.com
```

Mailing-List inbound (after ARC seal)

```
ARC-Seal: i=1; d=mailinglist.example.com;
ARC-Authentication-Results: i=1; dara=pass
     header.i=list@mailinglist.example.com (rcpt.to
     list@mailinglist.example.com matches signed header)
DKIM-Signature: d=originator.example.com; dara=mailinglist.example.com
To: list@mailinglist.example.com

   Mailing-List outbound (after ARC reseal)

X-Signed-Recipient: i=1; user@receiver.example.org
ARC-Seal: i=1; d=mailinglist.example.com...
ARC-Message-Signature: i=1; fh=...
ARC-Authentication-Results: i=1; dara=pass
     header.i=list@mailinglist.example.com (rcpt.to
     list@mailinglist.example.com matches signed header)
DKIM-Signature: d=originator.example.com; dara=mailinglist.example.com
To: list@mailinglist.example.com

   Receiver inbound (after ARC seal)

ARC-Seal: i=2; d=receiver.example.org...
ARC-Message-Signature: i=2; fh=...
ARC-Authentication-Results: i=2; dara=pass
        header.i=user@receiver.example.org (rcpt.to
        user@receiver.example.org matches signed header)
X-Signed-Recipient: i=1; user@receiver.example.org
ARC-Seal: i=1; d=mailinglist.example.com...
ARC-Message-Signature: i=1; fh=...
ARC-Authentication-Results: i=1; dara=pass
     header.i=list@mailinglist.example.com (rcpt.to
     list@mailinglist.example.com matches signed header)
DKIM-Signature: d=originator.example.com; dara=mailinglist.example.com
To: list@mailinglist.example.com
```

1.3.3.2. Originator ⇒ First Receiver; Replay ⇒ Victim Receiver

```
   Originator outbound (after ARC seal)

DKIM-Signature: d=originator.example.com; dara=receiver.example.com
To: user@receiver.example.com

   First receiver inbound (after ARC seal)

ARC-Seal: i=1; d=receiver.example.com
ARC-Authentication-Results: i=1; dara=pass
     header.i=user@receiver.example.com (rcpt.to
     user@receiver.example.com matches signed header)
DKIM-Signature: d=originator.example.com; dara=receiver.example.com
To: user@receiver.example.com
```

Above message captured by spammer, modified (add additional headers) and then resent. A spammer might send the message to john.doe@victim.example.net which would be unspecified in the headers.

Victim (last) receiver inbound (after ARC seal)

```
ARC-Seal: i=2; d=victim.example.net
ARC-Authentication-Results: i=2; dara=fail
     header.i=john.doe@victim.example.net (rcpt.to
     john.doe@victim.example.net mismatches signed header);
ARC-Seal: i=1; d=receiver.example.com
ARC-Authentication-Results: i=1; dara=pass
     header.i=user@receiver.example.com (rcpt.to
     user@receiver.example.com matches signed header)
DKIM-Signature: d=originator.example.com; dara=receiver.example.com
To: user@receiver.example.com
```

1.3.3.3. Originator ⇒ Naive-Forwarder ⇒ Receiver

This describes a message sent through Bcc to a forwarder that does not support DARA.

First outbound (after ARC seal)

```
ARC-Seal: i=1; d=originator.example.com; darn=naive.example.com;
ARC-Message-Signature: i=1; fh=...
X-Signed-Recipient: i=1; user@naive.example.com
Bcc: user@naive.example.com
```

The naive forwarder changes the recipient address from user@naive.example.com to user@aware.example.com, and the envelope recipient will change accordingly. aware.example.com supports DARA.

Final inbound (after ARC seal).

```
ARC-Seal: i=2; d=aware.example.com
ARC-Authentication-Results: i=2; dara=neutral
     header.i=user@aware.example.com (rcpt.to
     user@aware.example.com mismatches signed header);
ARC-Seal: i=1; d=originator.example.com; darn=naive.example.com;
ARC-Message-Signature: i=1; fh=...
X-Signed-Recipient: i=1; user@naive.example.com
Bcc: user@naive.example.com
```

At receiver, the declared and signed recipient user@naive.example.com will mismatch the envelope recipient user@aware.example.com, and fail DARA. However the protocol is set to optional verification with "darn=", and so does not report the failure. The domain specified naive.example.com by "darn=" may be

useful by spam filters at the receiver. For example the SPF HELO domain may match the "darn=" domain.

1.4. Chain of Custody

The local results of DARA can be combined into a path of verified ADMD domains from the originating sender to the receiver. As noted earlier, the ADMD are defined by the ARC-Message-Signature "d=" domains with FROM header alignment ADMD as the originating sender. The sender-defined receivers are described by the "dara=" tag at the sender containing the receiving domains and create sender-receiver pairs or metaphorical link in a chain. The originating sender defines the provenance of the message and the connected pairs create a "Chain of Custody" of the message. Chain building and verification can help detect if replay potentially occurred when there is a verification error. More specifically, a validation error can indicate there is a protocol unaware forwarder, or there is a malicious sender attempting to take the message and reinject it along a new path outside the intent of the originating sender. The verifier can check the prior sender's DARA declaration of "darn=" vs "dara=" to determine whether the unaware but benign scenario applies, or the aware sender but malicious scenario applies. If the malicious scenario, then it is up to the receiver's local policy to determine what receiver does with the result. The protocol for this verification is described in more detail in subsequent paragraphs.

The verified path that the message traverses can be used as the message flow identifier in a reputation system. Unlike purely domain based reputation systems, a path based one can help differentiate benign message flows from malicious ones to help identify replay or other abuse by identifying the spammer forwarding malicious content.

1.4.1. Chain Building Algorithm

The following defines an algorithm for path building using DARA identifiers. We define the nodes of a path as the ARC-Message-Signature "d=" identities and whose edges are sender-receiver pairs. Because building the edges of a path is a repeated process across edges that are like links, we call this Chain of Custody building or Chaining for short. It starts at the destination at ARC set "i=N", and walks through the ARC headers to the originating sender's ARC set "i=1" or the DKIM-Signature. The edge is defined as a pair of nodes (d_{n-1} , d_n) where the sender's ARC instance number "i=n-1" and receiver's "i=n". Further "d_{n-1}=" is the sender's ARC-Message-Signature "d=" domain, and "d_n=" is the receiver's ARC-Message-Signature "d=" domain. Next the sender's "dara=" domain d_n and the receiver's ARC-Seal "d=" domain d'_N MUST match. Similarly the ARC-Authentication-Result dara property header.i at n must be a subset of the signed and declared recipient list as defined at the sender

n-1. If so, edge building considers this a local *pass*. If the "dara=" result is missing, the verifier checks if there is instead a corresponding "darn=" tag at this or prior ARC set, then specifies an edge result of *neutral*, otherwise as *fail*. This recursively is extended for (d_{N-2} , d_{N-1}) i.e. for ARC set "i=n-2" and so forth for each n instance number to 1. At instance number 1, the verifier attempts to extend to a DKIM-Signature that is From header aligned and contains a "dara=" tag. If so, the DKIM-Signature is treated as a virtual "i=0", and the verifier checks if the DKIM-Signature "dara=" domain matches the ARC-Seal i=1 "d=" domain.

Local Chain verifier is done for each ARC set n following the above edge building from "i=N" to "i=1" and builds two vectors. One vector keeps the local chain results and the other ARC-Message-Signature "d=" domains. The verifier assumes that results from ARC header and message-body signature verification, DARA verifications have already run and the results already populate the ARC-Authentication-Results. For ARC set "i=N" to ARC set "i=2", the verifier MUST evaluate the local result, meaning the ARC result (i.e. from ARC seal verification and sometime ARC message-signature verification), edge building result, and DARA verification result. If it *passes*, the local Chain result is *pass*. Otherwise if any of them are *neutral* is *softfail*, and the rest pass, the result is *neutral*. Otherwise the result is *failure*. Further local policy MAY modify the ARC message-signature result (perhaps due to future work around draft-kucherawy-dkim-transform or draft-chuang-mailing-list-modifications) We recommend with the Chaining protocol to continue verification even if the sender's Chain result is failure or neutral, to provide forensics evidence for subsequent receivers. At the originating sender's ARC set "i=1" corresponding to d_1 or DKIM-Signature corresponding to d_0 the verifier first verifies alignment between header *From* domain and the ARC-Seal "dara=" domain. That domain defines d_1 or d_0 and the verifier looks up the DARA policy associated with the domain which MUST exist. If they are not aligned, then the local Chain verification is considered *neutral* as the message may have been forwarded from some unaware domain. In addition the ARC seal validation for origination MUST *pass* or local Chain verification is considered *fail*. Once these checks pass, then Chain building for "i=1" is considered to pass. The local Chain results is added onto the result vector at that index for all indexes, and similarly the ARC-Message-Signature "d=" domain onto the domain vector.

To compute the global Chain result, the verifier walks over the vector of results. The global Chain result is initialized to *pass*. Starting from "i=N" index to "i=1", if the local result is *fail* then the global result is *fail*, else if local result is *neutral* then the global is *neutral*. If the local result is *fail*, then the domain result is cleared from that index to i=1. This will inserts a

failure indication e.g. "arc-fail" at that index. If there are multiple failures, this chooses the most specific error as the cause e.g "dara-fail" over "arc-fail". This then truncates cleared domain entries from the domain list. If the local result is *fail*, this walk halts. If the local result is *neutral*, and there is a "darn=" then this inserts the domain in the domain list after the current index which helps identify it in the constructed path. A synthetic *neutral _result is also inserted in the result path. This also similarly extends the path when "i=1" and the message doesn't originate at that domain (missing alignment between the _From* header domain and ARC-Seal "d=" domain) to better identify the flow. The global Chain result is published ARC-Authentication-Results as a "chain=". If the result is *pass*, then the message is considered to be *authenticated* by DARA, otherwise *unauthenticated*.

1.4.2. Modified Body Algorithm

The protocol can detect when a message is modified along the forwarding path by looking at the current and previous message body hash and comparing them to find for changes. If the message content is considered spammy and phishy, then ADMDs that may have contributed to that problematic message body content MAY have their reputation per domain reputation of ADMDs negatively impacted. Other ADMDs that are proven to not have contributed message content SHOULD NOT be affected.

1.4.3. Definitions

ARC-Authentication-Results tags

* "chain=": Value of *pass* if local results and prior nodes are all passes, otherwise if "snr=" was present in the flow then *neutral*, else *fail*.

1.4.4. Examples

The following two examples illustrate working DARA/Chain-Building verification. This is followed by an example of DKIM replay attack. The second to last example is illustrative of how this protocol behaves with a SPF upgrade attack. The last example demonstrates a modified message body by a forwarder. (Other examples do not have a forwarder that modifies the message) .

1.4.4.1. Originator ⇒ Mailing-List ⇒ Receiver

This is an example of mail being sent from one Mail-Box-Provider to another through a Mailing-List where all ADMDs participate in DARA. In this illustrative example, we show the construction of the headers.

```
   Originator (after ARC seal)

ARC-Seal: i=1; d=originator.example.com;
    dara=mailinglist.example.com
ARC-Message-Signature: i=1; d=originator.example.com
ARC-Authentication-Results: i=1
From: user@originator.example.com
To: mailing.list@mailinglist.example.com

   Mailing-List outbound (after ARC seal)

ARC-Seal: i=2; d=mailinglist.example.com;
    dara=destination.example.com
ARC-Message-Signature: i=2; d=mailinglist.example.com
ARC-Authentication-Results: i=2; dara=pass; chain=pass
ARC-Seal: i=1; d=originator.example.com;
    dara=mailinglist.example.com
ARC-Message-Signature: i=1; d=originator.example.com
ARC-Authentication-Results: i=1
From: user@originator.example.com
To: mailing.list@mailinglist.example.com

   Receiver inbound (after ARC seal)

ARC-Seal: i=3; d=receiver.example.com
ARC-Message-Signature: i=3; d=receiver.example.com
ARC-Authentication-Results: i=3; dara=pass; chain=pass
ARC-Seal: i=2; d=mailinglist.example.com;
    dara=destination.example.com
ARC-Message-Signature: i=2; d=mailinglist.example.com
ARC-Authentication-Results: i=2; dara=pass; chain=pass
ARC-Seal: i=1; d=originator.example.com;
    dara=mailinglist.example.com
ARC-Message-Signature: i=1; d=originator.example.com
ARC-Authentication-Results: i=1
From: user@originator.example.com
To: mailing.list@mailinglist.example.com
```

The global Chain verification result is *pass* and the message is considered DARA/DMARC *authenticated*. The constructed path is [originator.example.com, mailinglist.example.com, receiver.example.com].

1.4.4.2. Originator ⇒ Mailing-List (From rewrite) ⇒ Receiver

This is an example of mail being sent from one Mail-Box-Provider to another through a Mailing-List where all ADMDs participate in DARA. In this illustrative example, we show the construction of the headers.

```
   Originator (after ARC seal)

DKIM-Signature: d=originator.example.com; dara=mailinglist.example.com
From: user@originator.example.com
To: list@mailinglist.example.com

   Mailing-List outbound (after ARC reseal)

X-Signed-Recipient: i=1; user@receiver.example.org
ARC-Seal: i=1; d=mailinglist.example.com...
ARC-Message-Signature: i=1; fh=...; d=mailinglist.example.com...
ARC-Authentication-Results: i=1; dara=pass
     header.i=list@mailinglist.example.com (rcpt.to
     list@mailinglist.example.com matches signed header);
     dkim=pass header.i=@originator.example.com
DKIM-Signature: d=mailinglist.example.com; dara=receiver.example.org
From: user@mailinglist.example.com
To: list@mailinglist.example.com

   Receiver inbound (after ARC seal)

ARC-Seal: i=2; d=receiver.example.org...
ARC-Message-Signature: i=2; fh=...; d=receiver.example.org...
ARC-Authentication-Results: i=2; dara=pass
        header.i=user@receiver.example.org (rcpt.to
        user@receiver.example.org matches signed header);
     dkim=pass header.i=@mailinglist.example.com
X-Signed-Recipient: i=1; user@receiver.example.org
ARC-Seal: i=1; d=mailinglist.example.com...
ARC-Message-Signature: i=1; fh=...; d=mailinglist.example.com...
ARC-Authentication-Results: i=1; dara=pass
     header.i=list@mailinglist.example.com (rcpt.to
     list@mailinglist.example.com matches signed header);
     dkim=pass header.i=@originator.example.com
DKIM-Signature: d=mailinglist.example.com; dara=receiver.example.org
From: user@mailinglist.example.com
To: list@mailinglist.example.com
```

The global Chain verification result is *pass* and the message is considered DARA *authenticated*. The constructed path is [mailinglist.example.com, receiver.example.com]. The receiver can also tell that the DKIM signature and From header was rewritten.

1.4.4.3. Originator ⇒ Naive-Forwarder ⇒Aware-Forwarder ⇒Aware-Receiver

This demonstrates a naive forwarder naive.example.com that does not support DARA/Chain. The headers represent what would be seen after inbound delivery to the receiver.

```
ARC-Seal: i=3; d=receiver.example.com
ARC-Message-Signature: i=3; d=receiver.example.com
ARC-Authentication-Results: i=3; dara=pass; chain=neutral
ARC-Seal: i=2; d=intermediate.example.com;
    dara=receiver.example.com
ARC-Message-Signature: i=2; d=intermediate.example.com
ARC-Authentication-Results: i=2; chain=neutral
ARC-Seal: i=1; d=originator.example.com; darn=naive.example.com
ARC-Message-Signature: i=1; d=originator.example.com
ARC-Authentication-Results: i=1
To: user@naive.example.com
```

The global Chain verification result is *neutral* and the message is considered DARA *unauthenticated*. The constructed path is [originator.example.com, naive.example.com, intermediary.example.com, receiver.example.com].

1.4.4.4. Originator ⇒ Receiver ; Spammer ⇒ Victim Receiver

Headers as seen by the receiver ADMD.

```
ARC-Seal: i=2; d=receiver.example.com
ARC-Authentication-Results: i=2; dara=pass; chain=pass
ARC-Seal: i=1; d=originator.example.com;
    dara=receiver.example.com
ARC-Authentication-Results: i=1
To: user@receiver.example.com
```

Final headers as seen by the victim ADMD after replay injection to victim.example.com domain.

```
ARC-Seal: i=3; d=victim.example.com
ARC-Authentication-Results: i=3; chain=fail
ARC-Seal: i=2; d=receiver.example.com
ARC-Authentication-Results: i=2; dara=pass; chain=pass
ARC-Seal: i=1; d=originator.example.com;
    dara=receiver.example.com
ARC-Authentication-Results: i=1
To: user@receiver.example.com
```

Note at ARC set #2, it does not set a "dara=" tag, causing a path discontinuity. Due to the path discontinuity, the global Chain verification result is *fail* and the message is considered DARA *unauthenticated*. The constructed path is [dara-fail].

1.5. Privacy Considerations

The DARA techniques depend upon declaring all recipients into the mail headers, and signing them. This could leak Bcc and mailing list recipients to each other who don't have an expectation of seeing

other hidden recipients. To prevent sharing of hidden recipients with each other, the message must be processed for each Bcc and mailing-list recipient where each recipient is uniquely declared and signed.

1.6. Security Considerations

Care must be taken in selecting the ARC-Seal "d=" sealing domain specified with "dara=" as described in Section 1.3.1.2. This protocol permits sharing a sealing domain across many different MX domain identities. However forwarders doing this should be aware that receivers' reputation systems may be tied to that shared sealing identity. Forwarders SHOULD match their sealing domain to their MX domain identity when possible.

1.7. IANA Considerations

This document has no IANA actions yet.

2. Normative References

[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>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/rfc/rfc5321>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/rfc/rfc5322>.

[RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, DOI 10.17487/RFC5598, July 2009, <https://www.rfc-editor.org/rfc/rfc5598>.

[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>.

Appendix A. Acknowledgments

Thanks goes to Emanuel Schorsch, Bruce Nan, Brandon Long, John R. Levine, and Murray S. Kucherawy for their knowledgeable feedback. Many thanks also to Marc Bradshaw for his contributions to concepts of authenticating senders.

Authors' Addresses

Weihaw Chuang
Google, Inc.

Email: weihaw@google.com

Bron Gondwana
Fastmail Pty Ltd

Email: brong@fastmailteam.com