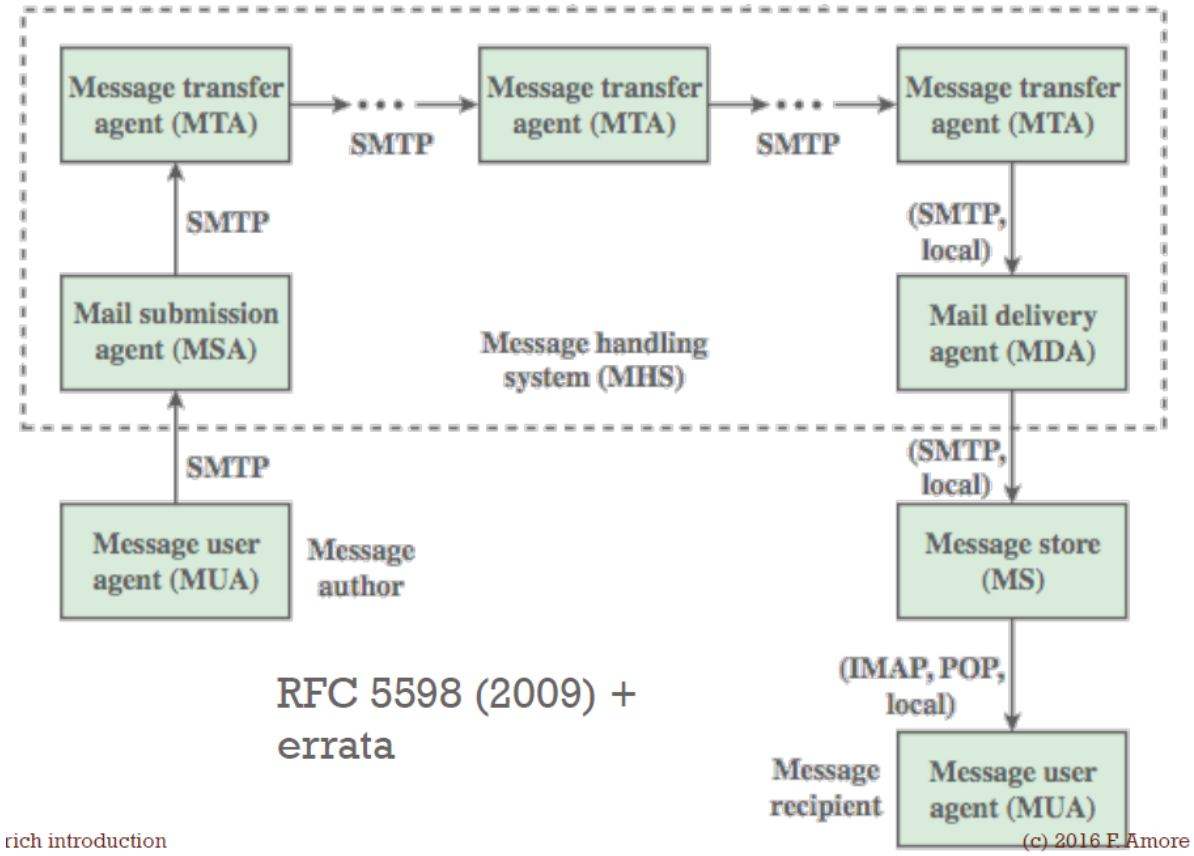
Internet E-Mail Architecture



E-mail is a technology for exchanging digital messages from an author to one or more recipients across the Internet. The e-mail system follows the **store-and-forward** model, i.e. servers accept, forward, store and deliver messages. Neither the users nor their computers are required to be online simultaneously.

An email message is composed by three parts:

- the **message envelope**
- the **message header**, containing control information like the sender's email address, one or more recipient's email address, subject, timestamps etc.
- the **message body**

Originally they supported only 7-bit ASCII communications medium. Thanks to MIME, the supported types have been augmented (later). The protocol used by the email system is called **SMTP** (Simple Mail Transfer Protocol). The SMTP protocols exploits the message envelope in order to communicates delivery parameters. The"@" char in an email address separates two fundamental part: the part before the "@" indicates the local part of the address, instead, the part next indicates a *domain name* or a *fully qualified domain name*. Client applications, in order to access the associated mailbox, exploits either POP (Post Office Protocol) or IMAP (Internet Message Access Protocol). It could be used also a proprietary protocol like Microsoft Exchange.

In the picture above we can identify a lot of entities:

- MUA (Message User Agent): it is the computer program used to access and manage a user's email (sender and recipient sides).
- MSA (Mail Submission Agent): computer program/software agent that receives email messages from a MUA and cooperates with a mail transfer agent (MTA) for delivering the mail. It is an intermediary entity.

- MTA (Message Transfer Agent): software that transfers email messages from one computer to another using a client-server application architecture. They implements both sides, client and server, in order to allow forwarding in both directions.
- MDA (Message Delivery Agent): computer software component that is responsible for the delivery of the email message to a local recipient's mailbox.
- MRA (Message Retrieval Agent): computer application that receives or fetches email from a remote mail server (that previously accepted and stored the messages of an user). It works with a MDA to deliver mail to a local or remote email mailbox.

Operation Overview:

- Alice composes a message using her MUA. She specifies the recipient's email address and presses the "send" button.
- The Alice's MUA formats the message in the email format and submits the latter to the local MSA, i.e. the one that is related to the domain of the Alice's email address.
- The MSA resolves the domain name related to the recipient's email address.
- The DNS responses with the MX entry related to that domain. It will correspond to a MTA server running from Bob's side.
- So, the forward operation completes. Maybe, the found Bob's MTA will forward by itself the message to other MTAs before reaching the correct MDA.
- Bob, through its MUA, retrieves the delivered mail exploiting POP or IMAP protocol.

Notice that:

- Alice may use a webmail service, instead of a local MUA.
- Alice's computer may run its own MTA.
- Bob could use a webmail service to retrieve his mails.
- There are several mail exchange servers (MXs) for a single domain.
- Email messages are NOT secured.

An Internet email message consists of two major sections:

- Header: structured collection of metadata containing information about the email message.
- Body: basic content of the email, **unstructured** text, like a letter.

Notice that the header is separated from the body by a blank line.

The header is obviously unique. It contains a lot of fields in a dictionary form *name-value*. Informally, each field is on a different line, the first character starts its name that ends before the ":" char. After the separator we can find the field value. If the first character is a space or a tab, the following content is related to the previous header field.

Mandatory fields:

- From: The email address of the author.
- Date: The local time and date when the message was written.

Suggested fields:

- Message-ID: automatically generated. Used to prevent multiple delivery and to reference other messages in a "communication" through In-Reply-To.
- In-Reply-To: Message-ID of the message that is a reply to.

Common fields:

- To: the email address(es) of the message's recipient(s).
- Subject: A brief summary of the message's topic.
- Bcc: Blind Carbon Copy. Email addresses included in the delivery list but not displayed in the message data.

- Cc: Carbon Copy. As before, but displayed.
- Content-Type: Information about how the message is to be displayed, usually a MIME type (later).
- References: like In-Reply-To (using Message-ID). It contains the list of the Message-IDs used in the communication.
- Reply-To: Address that should be used to reply to the message.
- Sender: Address of the *ACTUAL* sender acting on behalf of the author listed in the "From" field.

Some fields are added in order to trace the message's path like Received and Return-Path. Other headers will be used in order to perform security checks like Authentication-Results or Received-SPF.

MIME (Multipurpose Internet Mail Extensions)

It is a Internet standard that extends the supported formats. It includes non-text attachments, message bodies with multiple parts, header info in non ascii characters. It allows, generally speaking, of any type of file or document to be included in an email message (images, audio, video, spreadsheets etc.). The main idea is to convert a non ascii content in 7-bit ascii content, then after instructing the recipient's MUA, it is decoded in the MIME format.

MIME Headers are:

- MIME-Version
- Content-Type: fundamental, it defines the content type of a nontextual content. A charset is an optional parameter, if absent, ASCII is implied. E.g. Content-Type: text/plain; charset="ISO-8859-1".
- Content-Transfer-Encoding
- Content-Id
- Content-Description: a textual explanation of nontextual contents.

A MIME Type consists of a primary type and a sub type specified after the "/" char. E.g. .txt mapped into text/plain, .htm mapped into text/html, .jpg mapped to image/jpg. If no standard MIME type is matched, the general application/octet-stream is used.

Content-Transfer-Encoding supports 5 encoding types: 7bit, default and backward compatible, 8bit, includes also non ascii characters, binary, non ascii with unlimited-length lines, Base64, 6-bit blocks of data are encoded into 8-bit ASCII characters, Quoted-printable, non ascii char are encoded as an equal sign plus an ASCII code, e.g. "=C3=A0".

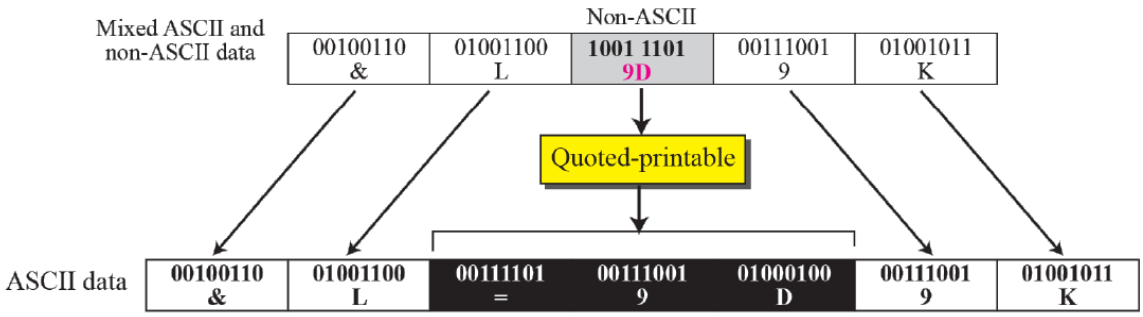
Base64

As I said before, a non text content is first converted into an ASCII representation and then, after transmission, it is decoded back to its original format.
MIME uses Base64 encoding:

- Take 3 bytes of data and put them into a 24-bit buffer.
- Work them extracting 6-bit at time (4 words).
- Use each word as an index into ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
- After this operation, it builds a 4 char ASCII string.
- Use zero, one or two "=" symbol for padding at the end.

Quoted Printable

ASCII char are left unchanged, except for "=" itself. NON ASCII char are represented using the "=" symbol followed by the ASCII encoding of the hexadecimal representation of the byte. Words are 8-bit long.



MIME standard allows also the inclusion of multiple part of non textual contents, each one associated with its own content type. MIME requires "multipart" as the first content type, in order to prepare the recipient's MUA to correctly parse each sub content. Each sub content is well separated by a boundary that is specified as parameter of the first content type multipart.
A multipart content type supports different sub types:

- Mixed: for sending files with different "content-type" headers.
- Digest: to send multiple text messages.
- Alternative: each part is an "alternative" version of the same (or similar) content.
- Encrypted: used in case of confidentiality. It includes also decryption information for the recipient.

Also nested multipart contents are allowed by the standard. We must care about of using different boundary delimiter.

```
From: Some One <someone@example.com>
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="XXXXboundary text"

This is a multipart message in MIME format.

--XXXXboundary text
Content-Type: text/plain

this is the body text

--XXXXboundary text
Content-Type: text/plain;
Content-Disposition: attachment;
        filename="test.txt"

this is the attachment text

--XXXXboundary text--
```

In order to include HTML content, we must use Content-Type: text/html. It allows to insert inline links, images, buttons and so forth. The problem is that, HTML is stressed by attackers in order to perform phishing attacks and the spread of malicious software.

SPAM

The term **SPAM** is used in order to characterized some emails as unwanted, useless and malicious. They can contain frauds, malware, chain letters. These examples are joint to a low-quality automatic language translation (accuracy increasing over time thanks to AI Natural Language processing).

Objective goals of spammers:

- Selling products and services

- Selling low-qualities and/or fakes goods/medicines
- Distributing malwares.
- Phishing, i.e. username/password stealing.
- Validate email addresses, i.e. links that refer to some service that records the access from an email.

HTML Images could be used for email address validation: in a remote http server is prepared a small downloadable image. Then this image is included in the email body having the same color of the background (to be not visible to the user). Then build an alias table that creates associations between emails and images, e.g. send to a@a.org the image 1.gif, if 1.gif is downloaded, then a@a.org is active.
Pay attention also to tiny URL embedded in the email body, check them not clicking on them!

Email Validation Systems

Three main systems:

- Sender Policy Framework (SPF): prevents e-mail spam by detecting email spoofing through verification of sender IP addresses.
- DomainKeys Identified Mail (DKIM): allows to check that incoming mail from a domain is authorized by that domain's administrators and that the email (including attachments) has not been modified during transport.
- Vouch by Reference (VBR): implements sender certification by third-party entities. Certification is based on DKIM.

Sender Policy Framework

SPFv1 protects the sender address (in envelope) by allowing the owner of a domain to specify a mail sending policy, i.e. which mail servers are authorized to send mail from the domain, using special DNS records of type SPF. Receivers can verify the SPF records and may reject messages from unauthorized sources BEFORE receiving the body of the message. If the server accepts the sender, and so the recipients and the body of the message, it should insert a Return-Path field in the message header in order to store the sender address. Notice that other email addresses are not prevented by forgery with SPF, e.g. From address.

E.g.

```
example.net TXT "v=spf1 mx a:pluto.example.net include:aspmx.googlemail.com -
all"
```

This is an example of SPF record for the domain example.net. The above DNS entry tells that:

- The version of the protocol is SPF version 1 (v=spf1).
- The incoming mail servers (the MXes DNS entries of the domain example.net)a re allowed/authorized to also send mail for example.net (mx).
- The resource pluto.example.net is authorized too (a:pluto.example.net).
- Everything that is considered legitimate by gmail.com is considered legitimate for example.net too (include:aspmx.googlemail.com).
- Others are considered illegitimate (-all).

Some mail servers, like Google ones, could simply verify the client's IP, instead of checking the sender's domain name. If an IP is not recognized, the message is rejected.

Drawbacks: maintenance is very difficult since records must be usually updated, failing SPF check doesn't mean directly reject the message (is one factor that could bring to a rejection policy), SPF breaks when a message is forwarded, SPF doesn't prevent From address spoofing in the header.

DKIM

DomainKeys Identified Mail (DKIM) is a specification for cryptographically signing email messages, permitting a signing domain to claim responsibility for a message. Message recipients can verify the signature by querying the signer's domain directly to retrieve the appropriate public key and thereby can confirm that the message was attested to by a party that owns the correct private key for the signing domain.

While an email is in transit it can be modified, invalidating so the signature. Headers and body are subjected to a canonicalization algorithm. It could be applied, independently for headers and body, a relaxed or simple canonicalization algorithm.

Notice that multiple signs can be applied while a mail is in transit, in order to verify more than one mail server identity for a domain. Since a domain could be associated with more than one public key, in the DKIM headers must be specified a selector: a selector is a plaintext string which, along with the domain name, allows recipient's mail servers to retrieve uniquely the public key to use in order to verify correctly the DKIM signature.

DKIM Example:

```
Received: by mail-wg0-f44.google.com with SMTP id dr12so5400749wgb.35
        for <damore@dis.uniroma1.it>; Mon, 18 Mar 2013 14:17:04 -0700 (PDT)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=gmail.com; s=20120113;
        h=x-received:mime-version:in-reply-to:references:from:date:message-id
         :subject:to:content-type;
        bh=I7Gc1zNUyy13QDKdzeRoGrgVCJaaKCpVqUjIPSV24P8=;
        b=u1yT9znzpgvzRm4/hiXZKtrq77auuYbqT7HjzpKAL4siHsKKlCZNgElIiPXLHk6Y6l
         7daYBXnicBUiZLkU5jaoo/uK+IocGZNbCEJ0nC0A42mNxX4GkL84JiMNjXvdd4wMTvMF
         IUUgjQLk710OZYas9rCSMCkK48e8SeVbTFnAF42BhqF4rIXbHN/9PhlUy7AXuqnE1SSy
         BtRfS28eSlO7xjRR7Lkg+VHgsAIhMRn/SNVle1T09lXwWIJSXayjlPzQREb1DYQM8B6n
         xSuqSIwztkshtTd2BjC2JrORKXa+tUeTBZjA3vzDKiG7dMqEMJxMN9i2GN8VK2IiAR69
         Kkig==
```

DKIM must specify some parameters in order to allow recipients to check the signature validity:

- a: is the hash/signing algorithm
- d: signing domain
- i: signing entity
- s: selector to use in order to retrieve correct public key for the domain d
- c: canonicalization algorithm used
- t: timestamp of the signature
- h: list of headers included in the signature (dkim-signature is implied)
- b: the signature itself
- bh: the hash of the canonicalized body part of the message

DMARC

Domain-based Message Authentication, Reporting, and Conformance (DMARC) is a standard that helps protect email senders and recipients from spam, spoofing and phishing. DMARC allows an organization to publish a policy that defined its email authentication practices and provides instructions to receiving mail servers on how to enforce them.

DMARC defines methods for:

- Publish email authentication practice
- State what actions should be taken on mail that fails authentication checks
- Enable reporting of these actions taken on mail claiming to be from a specific domain.

It is not an authentication protocol, but it is built on SPF and DKIM in order to provide it.

The domain policies are published and listed as DNS records by the domain's administrator. It is a TXT record named *_dmarc.domainname*.

```
_dmarc.mydomain.com. IN TXT "v=DMARC1\; p=none\; rua=mailto:dmarc-
aggregate@mydomain.com\; ruf=mailto:dmarc-afrf@mydomain.com\; pct=100"
```

It must be read from left to right:

- DMARC policy for *mydomain.com*.
- version DMARC1, v=DMARC1.
- No preferred policy, p=none.
- The mailbox to which aggregate report should be sent is dmarc-aggregate@ mydomain.com, rua=mailto:dmarc-aggregate@ mydomain.com.
- Forensic reports' mailbox is dmarc-afrf@ mydomain.com, ruf=mailto:dmarc-afrf@ mydomain.com
- The percentage to which the policy must be applied is 100%, pct=100.

The supported policy are three: none, treat the email as it would be without any DMARC validation, quarantine, accept the mail but place it in the spam folder, reject, message is rejected.

When an inbound mail server receives an email, it uses DNS to look up DMARC policy for the domain contained in the message's "From" header. Then, it perform three action:

- Check if the DKIM signature is valid.
- Check if the message came from an IP address allowed by the domain's SPF records.
- Check for proper "domain alignment in the message headers.

The latter concept is related to SPF and DKIM usage: DMARC domain alignment matches a message's "From" domain with information relevant to these other standards:

- For SPF, the message's From domain and its Return-Path domain must match.
- For DKIM, the message's From domain and its DKIM d parameter domain must match.

After performing these checks, the server is ready to apply the policy defined by the domain's administrator. At the end, the receiving mail server will report the outcome.

Notice that Aggregate Reports are XML documents that show statistical data about the message and report authentication results, Forensic Reports, instead, are individual copies of messages which failed authentication enclosed in a special email message formatted as AFRF specifies.

The check outcomes are reported in the mail headers:

```
Authentication-Results: mx.google.com;
       dkim=pass header.i=@enisa.europa.eu header.s=enisadkim
header.b=B8Ea0tk+;
       spf=pass (google.com: domain of prokopios.drogkaris@enisa.europa.eu
designates 139.91.222.30 as permitted sender)
```

Vouch By Reference

VBR is a protocol for implementing sender certification by third-party entities. Certification authorities provide vouch for the reputation of senders by verifying the domain name that is associated with transmitted email. After signing the message through DKIM, a VBR-info header is added to the message header: it includes domain name that is being certified, the content type of the message and a list of one or more vouching services that vouch for the sender for that kind of content:

```
VBR-Info: md=domain.name.example; mc=type;
mv=vouching.example:vouching2.example
```

A receiver can authenticate the message's domain name using DKIM or SPF. He obtains also the name of the vouching services that it trusts, either from among the set supplied by the sender or from a locally configured set of preferred vouching services. DNS allows also to verify the correctness of a claimed vouching service. It can ask for:

domain.name.example._vouch.vouching.example

The returned data, if any, is a list of all types that the service vouches. They must match the one added in the message content.
The result of the authentication can be inserted as a Authentication-Result header (as before).

Basic Email Analysis

Topics:

- Email spoofing
- Email tracking

Email spoofing

Activity of altering the email's sender address to the purpose of making the message looking like originated from other sender. It is extremely easy using SMTP without authentication mechanism. In order to check for spoofing, we must analyze the complete message (full header + body): standard email clients normally hide most of the header, we need to retrieve the integral and original message. We must check fields like From, Return-Path, Reply-To, Received, etc. We must check for domain names and IP Addresses.

PGP

Pretty Good Privacy is a standard created in 1991. It empowers people to take their privacy into their own hands. There is no third-party authority that guarantees for a domain's keys, it is all based on the trustness level of the community.

Security requirements:

- Confidentiality.
- Authentication.
- Message Integrity.
- Non-Repudiation.

PGP Authentication:

- Sender creates the message m.
- Make the SHA-1 160-bit hash of the message.
- Concatenates the RSA signature of the hash to the message m (using its own private key).
- Receiver decrypts and recovers hash code from the signature (using public key of the sender).
- Receiver computes the hash of the message and check if the two hashes match.

PGP Confidentiality:

- Sender generates a 128-bit session key.
- Encrypts the message with the session key.
- Attaches RSA encrypted session key to the encrypted message (using public key of the receiver).
- Receiver decrypts and recovers session key (using its own private key).
- Receiver uses session key to decrypt the message.

PGP Authentication & Confidentiality:

- Sender first creates signatures and attaches it to the message.
- Then he generates a session key, encrypts the message and the signature with the session key, finally attaches the RSA encrypted key to the encrypted message plus signature (using public key of the receiver).
- Receiver retrieves the session key, decrypts the message and check for the signature's validity.

Notice that PGP, by default, compresses message after signing BUT before encrypting using ZIP compression algorithm.

The encrypted payload, for compatible reason, is encoded through Base64.

Since PGP requires a random key for each message, it defines also a pseudo random generator based on keystrokes. The time when it receives keystroke is stored along with the 8-bit value of the keystroke itself.
Each PGP user maintains a pair of keyrings:

- Public key ring contains all the public keys of other PGP users known to the current one, indexed by key ID (least significant 64 bit of the key itself).
- Private key ring contains the public/private key pairs for this user indexed by key ID and encrypted keyed from an hashed passphrase. Security of private keys depends on the passphrase security.

No third certificate authorities are included in PGP. Each user is its own CA. The concept that PGP wants to reach is a web of trust, i.e. my trustiness level depends on how many PGP users trust my identity (public key). E.g. if I trust A, and A trust B, so I trust B. If I partly trust A and A trust B, I deem legitimate B.

ARC (Authenticated Received Chain)

Previously, we described a lot of standards that provide some kind of information in order to recognize email address spoofing and spam. There are some flaws with those approaches:

- Using DMARC with a strict policy (p=reject), may classify legitimate messages as malicious one, resulting in a block of these messages. This could happen for example through indirect mail flows such as mailing lists, forwarding or filtering services.
- Moreover, forwarding causes SPF to fail even if origin was legit.
- Forwarders often alter messages, breaking DKIM signatures, e.g. virus scan result, removed attachments and so on so forth.

Design choices for ARC were:

- Originator of message makes no changes.
- Keep the Authentication-Results content intact from the first ARC intermediary forward.
- Allow for multiple hops in the indirect mail flow.
- ARC headers can be verified at each hop.
- Scalable approach as Internet is.
- Define ARC independently of DMARC.
- Message's recipient seeing an authentication failure with DMARC may choose to check ARC headers.
- If ARC headers are intact, they can see and validate Authentication-Results content reported by the ARC participants.
- Depending on the trustiness level of an intermediary result, message's recipient may choose to use ARC information to override the DMARC authentication checks (locally).

Keeping intact ARC chains allows all the intermediary hops to look at DKIM, DMARC and SPF checks results as seen by first hop. Moreover, signatures from participating intermediaries can be reliably linked to their domain name. It allows also to alter the message, since the previous check is kept intact, a new one can be generated starting from an altered message. ARC provides a kind of system tracking to judge intermediaries' behavior.

Notice that ARC does not say anything about "trustworthiness" of the message sender or intermediaries. it does not say nothing about the content of the message and an intermediary might still inject bad content or remove some ARC headers.

In order to implement ARC, three new header fields are added:

- ARC-Authentication-Results (AAR): Archived copy of Authentication-Results.
- ARC-Seal (AS): Includes some tags and a DKIM-style signature of any preceding ARC headers.
- ARC-Message-Signature (AMS): A DKIM-style signature of the entire message except ARC-Seal headers.

For the AAR, an "i=" tag is prepended, in order to keep track of ARC headers sequence.
AMS uses "i=" tag too and "v=" tag is missing instead.
AS is populated with key=value pairs. "b=" contains the signature of all ARC headers, "a=/d=/s=" tags are the same of DKIM standard, "cv=" indicates whether ARC chain validated as received by the reporting intermediary, "i=" sequence number as before.

First the Authentication-Results content is copied into a new ARC-Authentication-Results header, prefixed to the message. Then an ARC-Message-Signature is computed for the message, including newest AAR header, and prefixed to the message (not including ANY AS headers). Finally ARC-Seal is computed and prefixed too. Notice that the order of headers is not relevant for the validity checks.

The "i=" tags is used to allows multiple ARC Header sets to be grouped correctly, eliminates reliance on the order of headers being inserted, compare with order of insertion of various authentications.

Each participant determines "cv=" value of its ARC-Seal as follows:

- All ARC-Seal headers must validate.
- The "cv=" value for those AS headers must be Pass.
- The most recent ARC-Message-Signature must validate.

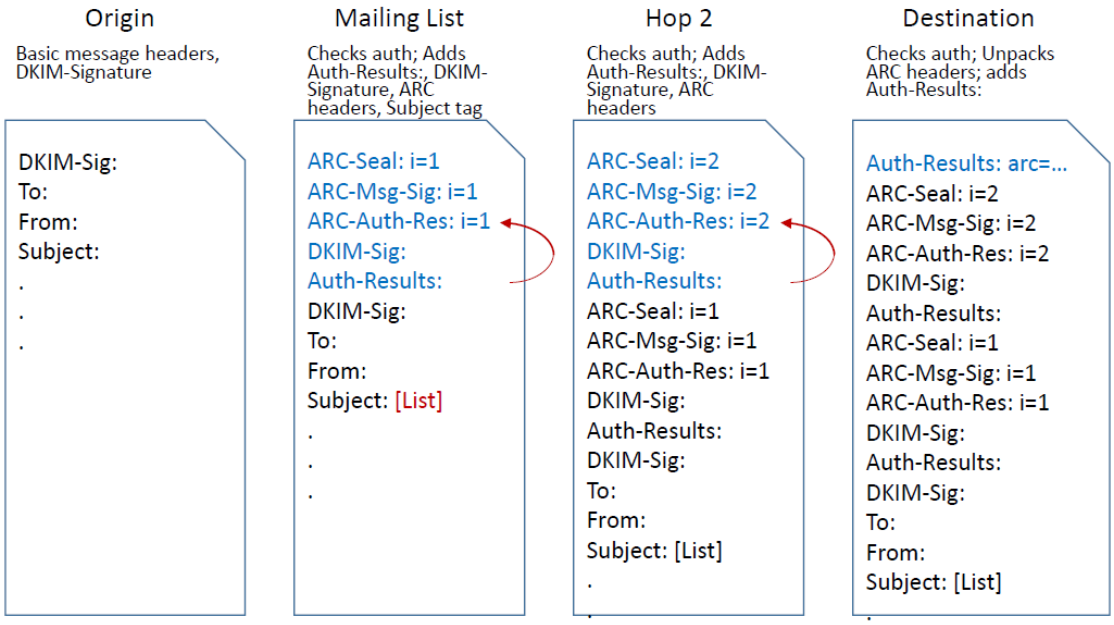
A set of ARC Headers must be added if:

- A message is gonna be subject to changes or alterations that will break previous DKIM signatures, like stripping attachments or appending disclaimers and footers.
- When the message crosses a trust boundary, for example exiting a trusted domain.

Different organizations, indeed, have different configurations, but generally speaking, they check for ARC on inbound messages and insert ARC when messages are outbound.

A set of ARC Headers must not be added if:

- A message will be delivered to a mailbox within the local organization.
- When the anonymity is a requirement for the message. Since the ARC Chain builds a trusted and verifiable chain of entities, this could be a problem if some information is required to be hidden.



- A message is produced and must be sent to a mailing list. It is attached its DKIM signature.
- Then an intermediary mail server will check for domain's authentication. The result is put in an Authentication-Results header. A new DKIM-signature is added too. Since the subject is changed, it creates a set of ARC headers with sequence number i=1.
- The same steps are performed by the second hop. New authentication-results is prefixed as for a new DKIM signature. Then a new set of ARC headers is prefixed.
- The destination server will "unpack" the ARC chain and will perform all the authentication checks required, i.e. validating all ARC-Seal chain and the last ARC-Message-Signature (of the entire message). The verdict is pass or fail and it is contained in the "arc=" tag.