E-MAIL SECURITY

Leonardo Querzoni
querzoni@diag.uniroma1.it

OVERVIEW

The Internet e-mail system

- Architecture and basic functioning
- SMTP, POP, IMAP
- extensions (MIME)
- Email threats
- Infrastructure security: SPF, DKIM, ARC, DMARC
- End-to-end security: PGP, S/MIME

THE E-MAIL SYSTEM

E-mail is a method of exchanging digital messages from an author to one or more recipients, operating across the Internet/intranet

Among the oldest services available on the internet!

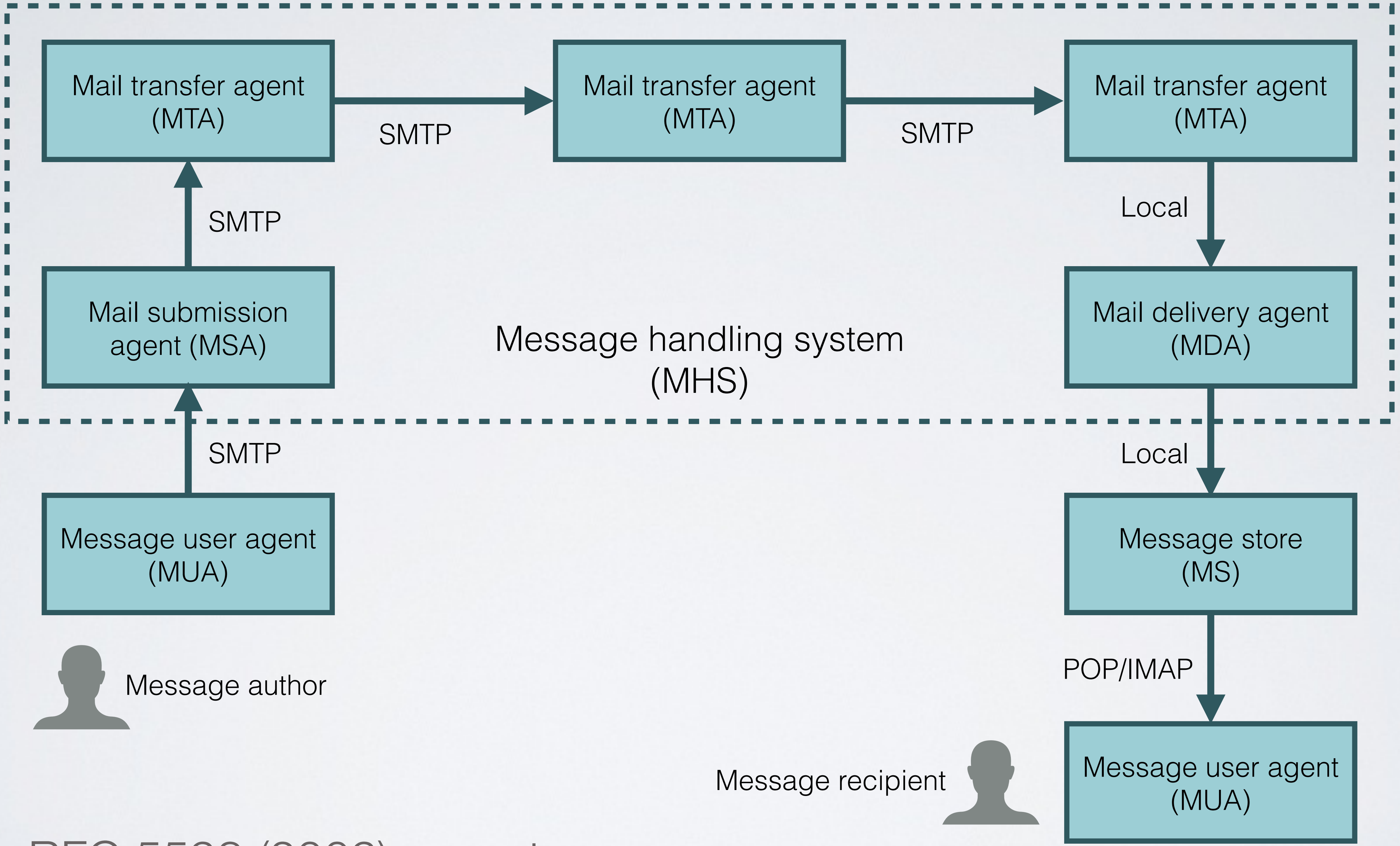
- is around 50 years old! It began in 1971 when Ray Tomlinson, a computer engineer, sent the first electronic message across the ARPANET, the precursor to the Internet, addressing the recipient with the "user@domain" scheme

Modern e-mail systems are based on a **store-and-forward model**: e-mail servers accept, forward, store and deliver messages

- neither the users nor their computers are required to be online simultaneously

Check RFC 5598

INTERNET E-MAIL ARCHITECTURE



Mail transfer agent (MTA)
SMTP
Mail transfer agent (MTA)
SMTP
Mail transfer agent (MTA)
SMTP
Local
Mail submission agent (MSA)
Message handling system (MHS)
Mail delivery agent (MDA)
SMTP
Local
Message user agent (MUA)
Message store (MS)
Message author
POP/IMAP
Message recipient
Message user agent (MUA)

RFC 5598 (2009) + errata

MUA, MSA, MTA

Message User Agent (MUA)

- used to access and manage a user's e-mail

Mail Submission Agent (MSA)

- receives e-mail messages from a MUA and cooperates with a mail transfer agent (MTA) for delivery of the mail
- it makes sure the message meets the standard format requirements

Mail Transfer Agent (MTA)

- transfers e-mail messages from one computer to another using a client–server application architecture
- MTAs implements both the client and server portions of the Simple Mail Transfer Protocol

MDA, MRA

Mail Delivery Agent (MDA)

- responsible for the delivery of e-mail messages to a local recipient's mailbox
- local message delivery is achieved through a process of handling messages from the MTA, and storing mail into the recipient's environment (typically a mailbox)

E-MAIL EXCHANGE

E-mail transmission across IP networks is carried by the **Simple Mail Transfer Protocol** (**SMTP**, RFC 821, 1982)

- last update: RFC 5321 (2008). Includes the extended SMTP (ESMTP) additions

SMTP communicates delivery parameters using a message envelope separate from the message (header and body) itself

An Internet e-mail address is a string of the form *user@domain*

- the part before the @ sign is the local part of the address
- the part after the @ sign is a fully qualified domain name

MESSAGE FORMAT

The **Internet Message Format (IMF)** is defined by RFC 5322

- support to MIME (RFC 2045 through RFC 2049), collectively called Multipurpose Internet Mail Extensions

Internet e-mail messages consist of two major sections:

- **Header** — Structured into fields such as From, To, CC, Subject, Date, and other information about the email.
- **Body** — The basic content, as unstructured text; sometimes containing a signature block at the end. This is exactly the same as the body of a regular letter.

The header is separated from the body by a blank line

HEADER

Each message has exactly one header, which is structured into fields. Each field has a name and a value. RFC 5322 specifies the precise syntax

- Informally, each line of text in the header that begins with a printable character begins a separate field and its name starts in the first character of the line and ends before the separator character ":"
- The separator is then followed by the field value. The value is continued onto subsequent lines if those lines have a space or tab as their first character. **Field names and values are restricted to 7-bit ASCII characters**. Non-ASCII values may be represented using MIME encoded words

Email header fields can be multi-line, and each line must be at most 76 characters long.

HEADER

Each message is identified through two kinds of IDs

Message-ID:

- Pertains to content and is globally unique.
- Its format is similar to that of a mailbox, with two parts separated by @
  - The right side specifies the domain or host that assigns the identifier
  - The left side contains a string that is globally opaque and serves to uniquely identify the message within the domain referenced on the right side.
- Has a variety of uses including threading, aiding identification of duplicates, and DSN (Delivery Status Notification) tracking.
- The MSA assigns the Message-ID:
- Example: Message-ID: <20241001102233.12345@example.com>

HEADER

Each message is identified through two kinds of IDs

ENVID

- Stands for "envelope identifier"
- Used for message-tracking purposes ([RFC3885], [RFC3464]) concerning a single posting/delivery transfer.
  - ENVID is used for one message posting until that message is delivered. A re-posting of the message, such as by a MTA, does not reuse that ENVID.

HEADER

The header contain the following mandatory fields:

- From - Sender's email address of the sender.
- To - Email recipients. Multiple email addresses can be included.
- Date - The date and time when the message was sent. It follows a specific format (RFC 5322).
- Message-ID
- Subject - A brief summary of the content of the email.

Note that the To: field is not necessarily related to the addresses to which the message is delivered. The actual delivery list is supplied separately to SMTP, which may or may not originally have been extracted from the header content. In the same way, the "From:" field does not have to be the real sender of the email message.

HEADER

Other headers are optional:

- Cc (Carbon Copy) - Additional recipients who will receive a copy of the email.
- Bcc (Blind Carbon Copy) - Same as Cc, but these recipients are hidden.
- Reply-To - Specifies the email address to which replies should be sent if it's different from the "From" address.
- In-Reply-To - Contains the Message-ID of the email to which this message is a response. This is used for threading in email clients.

HEADER

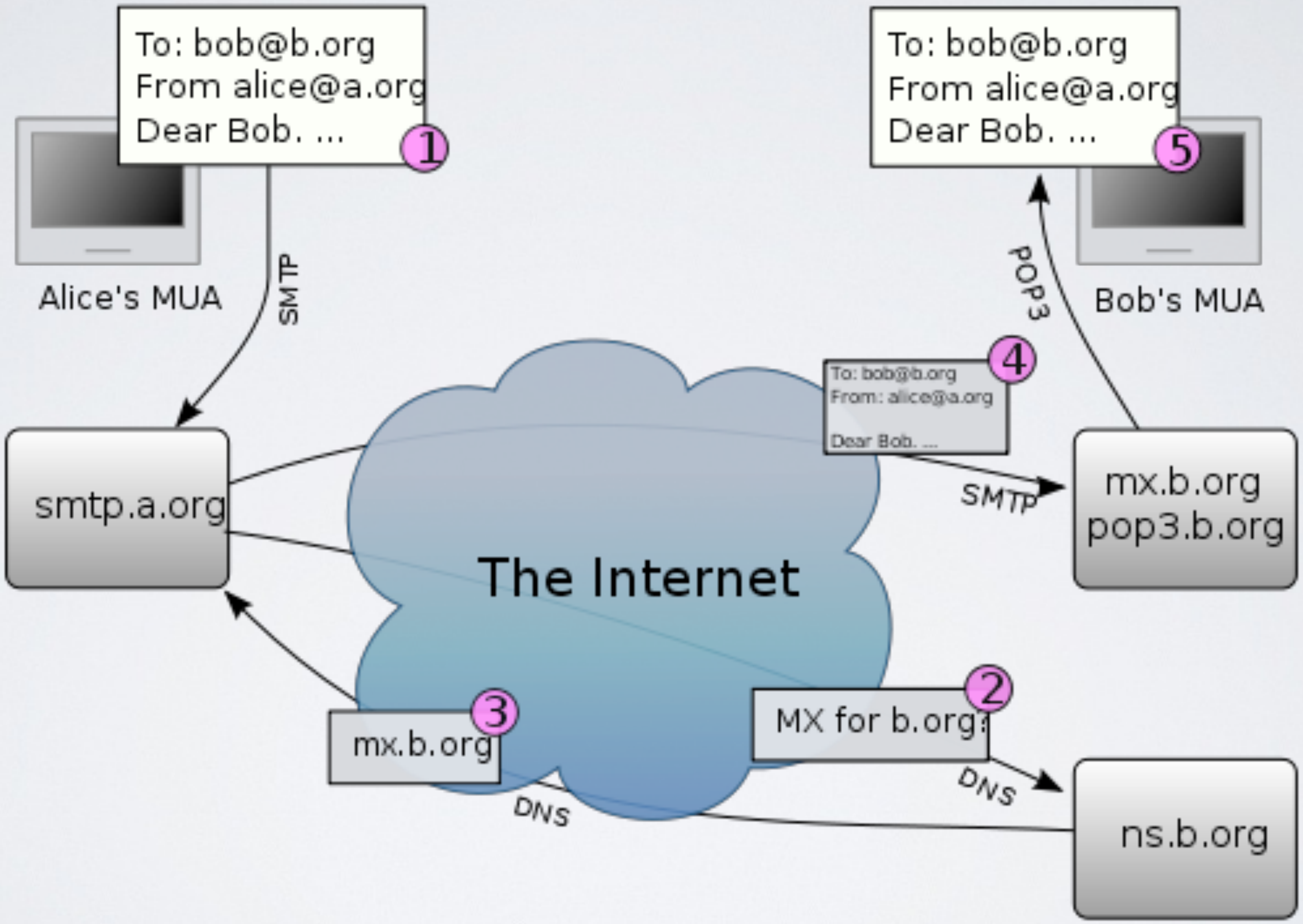
Other headers are optional:

- References - Lists Message-IDs of previous emails in the thread to maintain conversation history.
- Sender - Used when the person sending the email is different from the one listed in the "From" field.
- Return-Path - Specifies the address that will receive bounce-back messages if the email cannot be delivered.
- Received - This field is added by each mail server that processes the message, tracking the route of the email. Each hop adds a new "Received" field.
  - Example: `Received: from mail.example.com by smtp.example.org with ESMTP; Fri, 1 Oct 2024 10:22:33 -0400`

E-MAIL EXCHANGE

- MUAs (client mail applications) only use SMTP for sending messages to a mail server for relaying
- To access their mail box accounts, MUAs usually use:
  - the **Post Office Protocol (POP)** for mail downloading and offline usage
  - Internet **Message Access Protocol (IMAP)** for online mail reading
  - Proprietary protocols (such as in Microsoft Exchange or Lotus Notes/Domino)

OPERATION OVERVIEW



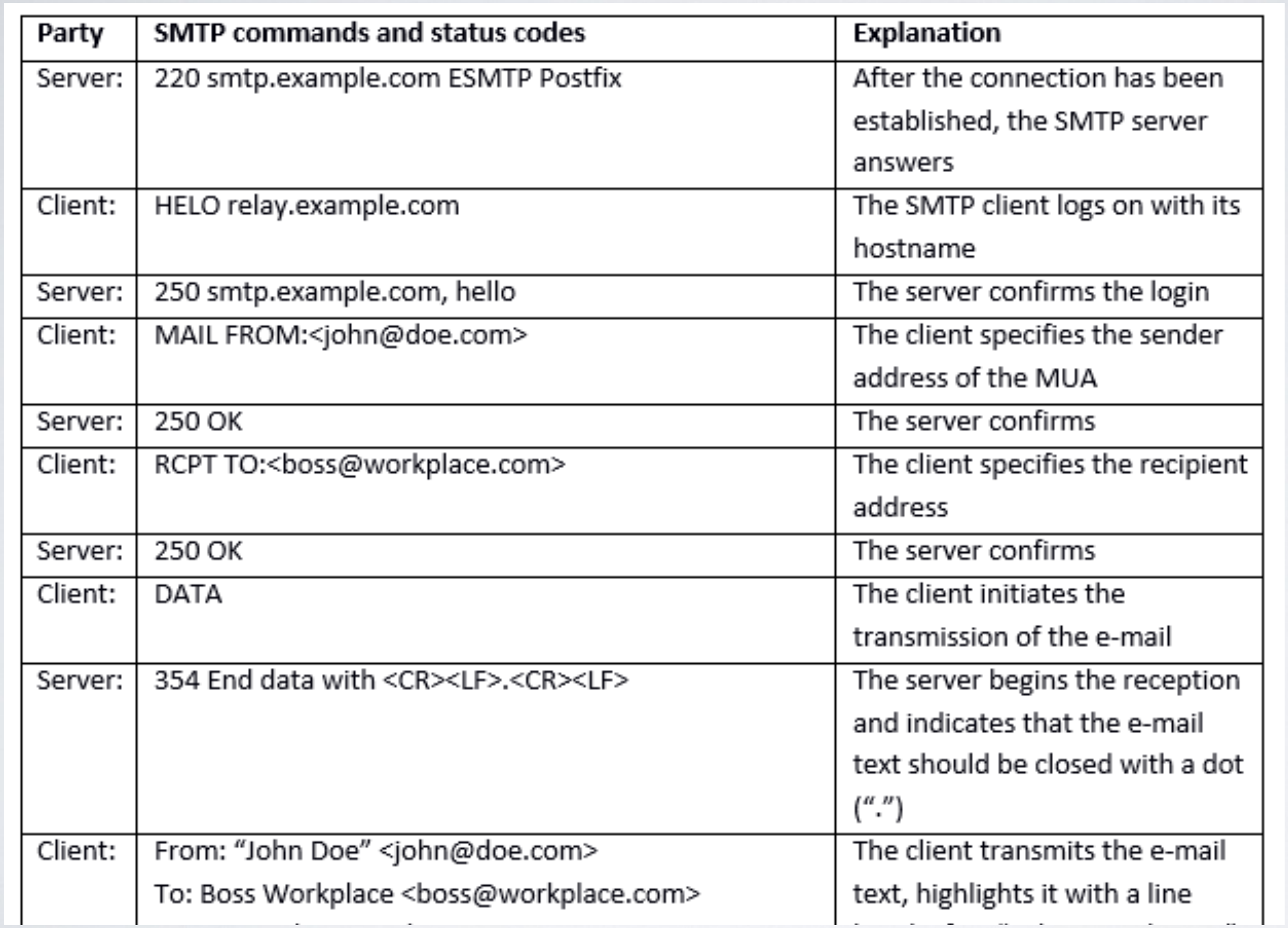
OPERATION OVERVIEW

1. Alice composes a message using her MUA; she enters the e-mail address of her correspondent, and hits the "send" button
2. The MUA formats the message in email format and uses the Submission Protocol (variant of SMTP, see RFC 6409) to send the message to the local MSA
3. The MSA looks at the destination address provided in the SMTP protocol and resolves a domain name to determine the fully qualified domain name of the mail exchange server
4. the DNS server for the b.org domain responds with any MX records listing the mail exchange servers for that domain, in this case mx.b.org, a MTA server run by Bob's ISP

OPERATION OVERVIEW

5. smtp.a.org sends the message to mx.b.org using SMTP
   - this server may need to forward the message to other MTAs before the message reaches the final message delivery agent (MDA), which delivers it to the mailbox of Bob
6. Bob presses the "get mail" button in his MUA, which picks up the message using either the Post Office Protocol (POP3) or the Internet Message Access Protocol (IMAP4)

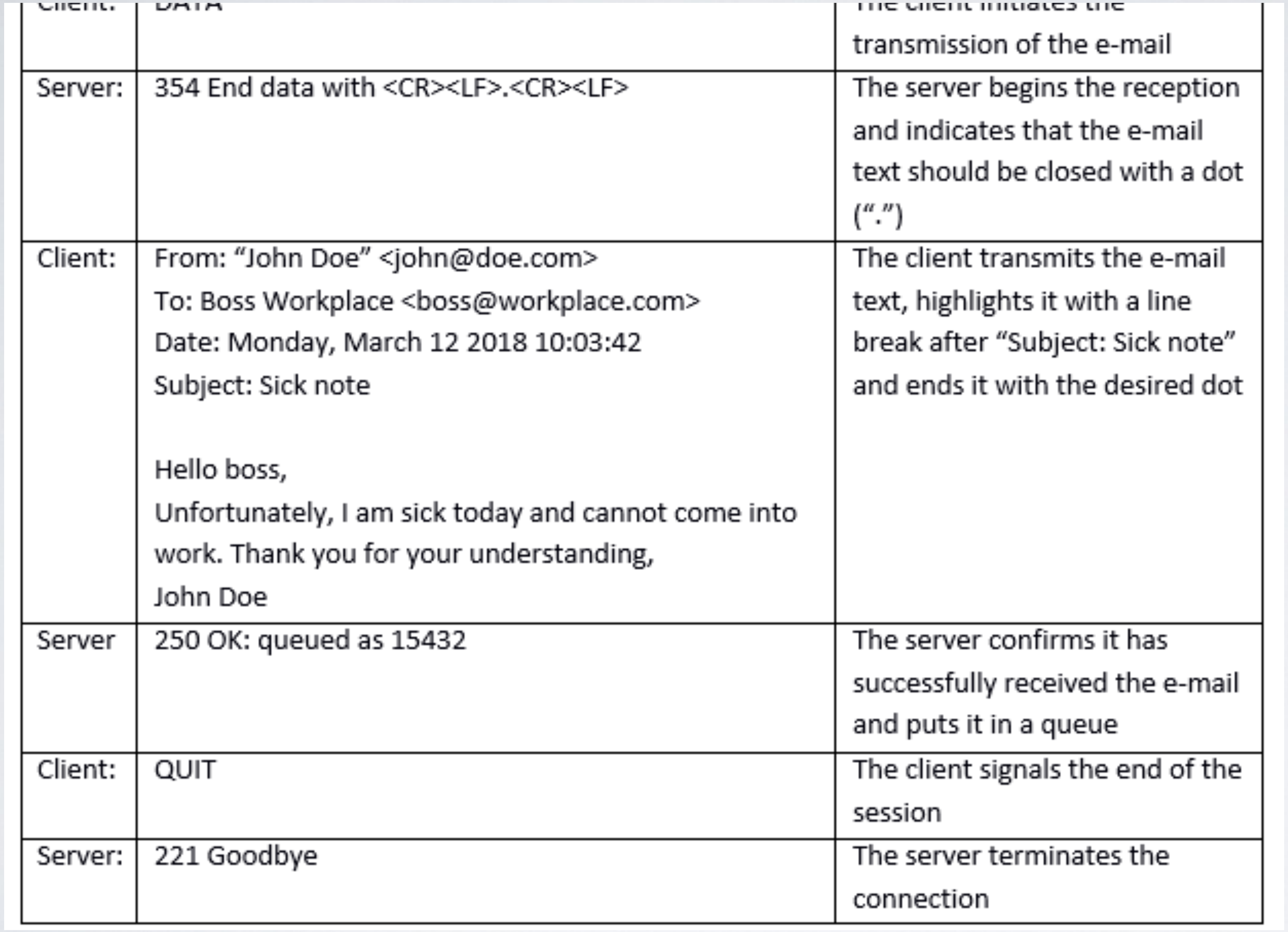
SAMPLE SMTP INTERACTION



Party	SMTP commands and status codes	Explanation
Server:	220 smtp.example.com ESMTP Postfix	After the connection has been established, the SMTP server answers
Client:	HELO relay.example.com	The SMTP client logs on with its hostname
Server:	250 smtp.example.com, hello	The server confirms the login
Client:	MAIL FROM:<john@doe.com>	The client specifies the sender address of the MUA
Server:	250 OK	The server confirms
Client:	RCPT TO:<boss@workplace.com>	The client specifies the recipient address
Server:	250 OK	The server confirms
Client:	DATA	The client initiates the transmission of the e-mail
Server:	354 End data with <CR><LF>.<CR><LF>	The server begins the reception and indicates that the e-mail text should be closed with a dot (“.”)
Client:	From: “John Doe” <john@doe.com> To: Boss Workplace <boss@workplace.com>	The client transmits the e-mail text, highlights it with a line

as of
RCF 821

SAMPLE SMTP INTERACTION



Client:	DATA	The client initiates the transmission of the e-mail
Server:	354 End data with <CR><LF>.<CR><LF>	The server begins the reception and indicates that the e-mail text should be closed with a dot (".")
Client:	From: "John Doe" <john@doe.com> To: Boss Workplace <boss@workplace.com> Date: Monday, March 12 2018 10:03:42 Subject: Sick note Hello boss, Unfortunately, I am sick today and cannot come into work. Thank you for your understanding, John Doe	The client transmits the e-mail text, highlights it with a line break after "Subject: Sick note" and ends it with the desired dot
Server	250 OK: queued as 15432	The server confirms it has successfully received the e-mail and puts it in a queue
Client:	QUIT	The client signals the end of the session
Server:	221 Goodbye	The server terminates the connection

as of RCF 821

ESMTP

RFC 821 (1982) was obsoleted by RFC 5321 (2008), where ESMTP, an extended version of SMPT, was introduced.

Software agents should stick to ESMTP but for backward compatibility reason a client connecting by SMTP will be also served

The greeting command for ESMTP is EHLO, which gets a (possibly multiline) response listing the supported extended commands

OTHER ESMTP COMMANDS

- 8BITMIME — 8 bit data transmission, RFC 6152
- ATRN — Authenticated TURN for On-Demand Mail Relay, RFC 2645
- AUTH — Authenticated SMTP, RFC 4954
- CHUNKING — Chunking, RFC 3030
- DSN — Delivery status notification, RFC 3461 (See Variable envelope return path)
- ETRN — Extended version of remote message queue starting command TURN, RFC 1985
- HELP — Supply helpful information, RFC 821
- PIPELINING — Command pipelining, RFC 2920
- SIZE — Message size declaration, RFC 1870
- STARTTLS — Transport layer security, RFC 3207 (2002)
- SMTPUTF8 — Allow UTF-8 encoding in mailbox names and header fields, RFC 6531
- UTF8SMTP — Allow UTF-8 encoding in mailbox names and header fields, RFC 5336 (deprecated)

MIME (MULTIPURPOSE INTERNET MAIL EXTENSIONS)

Internet standard that extends the format of email to support:

- Text in character sets other than ASCII
- Non-text attachments
- Message bodies with multiple parts
- Header information in non-ASCII character sets

MIME's use has grown beyond describing the content of email to describe content type in general (web, storage)

Virtually all human-written Internet email and a fairly large proportion of automated email is transmitted via SMTP in MIME format

MIME is specified in six linked RFC memoranda

MIME

Important RFCs

- RFC-822 Standard for the format for ARPA Internet text messages
- RFC-2045 MIME Part 1: Format of Internet Message Bodies
- RFC-2046 MIME Part 2: Media Types
- RFC-2047 MIME Part 3: Message Header Extensions
- RFC-2048 MIME Part 4: Registration Procedure
- RFC-2049 MIME Part 5: Conformance Criteria

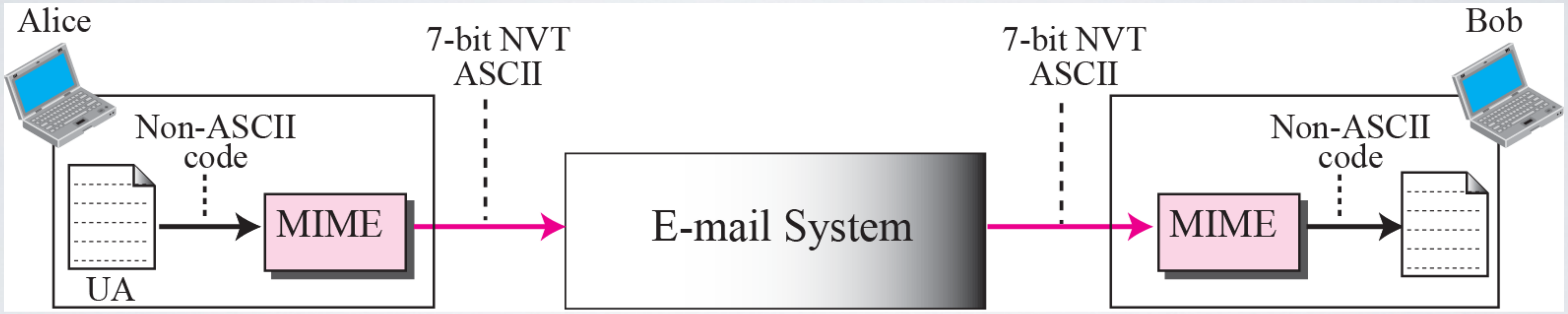
MIME – WHAT IS IT?

- MIME refers to an official Internet standard that specifies how messages must be formatted so that they can be exchanged between different email systems.
- MIME permits the inclusion of virtually any type of file or document in an email message.
- Specifically, MIME messages can contain
  - text
  - images
  - audio
  - video
  - application-specific data
    - spreadsheets
    - word processing documents

MIME FEATURES

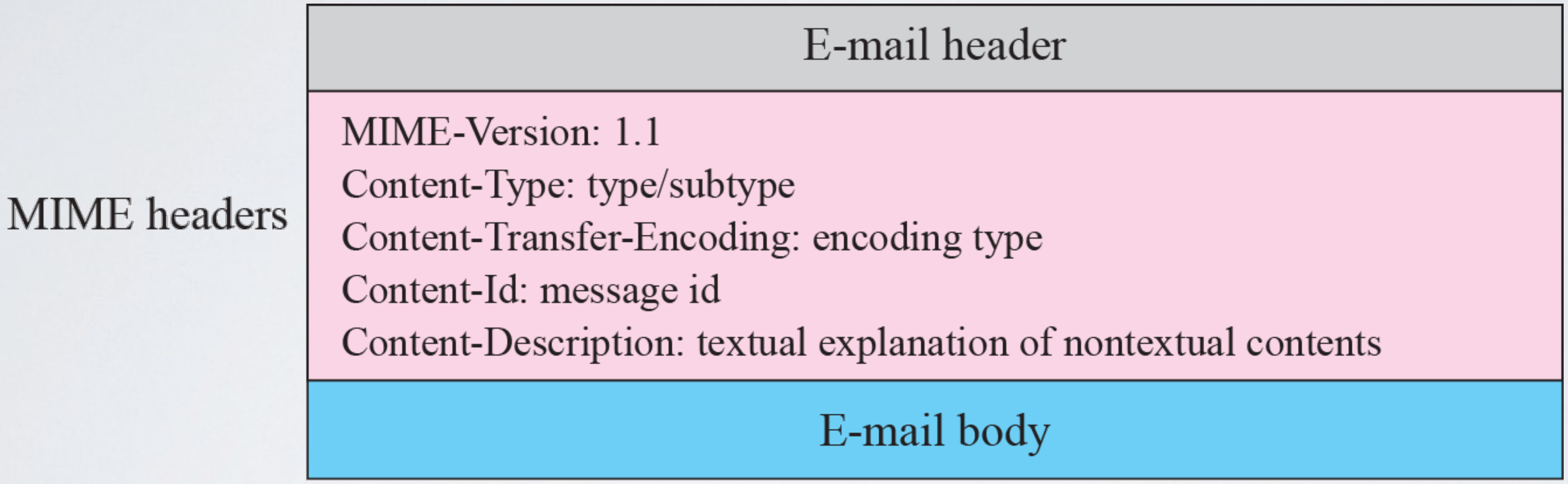
- Support of character sets other than ASCII
- Content type labeling system
- Support of non-text content in e-mail messages
- Support for compound documents

MIME SCHEME



NVT = network virtual terminal

MIME HEADERS


MIME headers

E-mail header

```
MIME-Version: 1.1
Content-Type: type/subtype
Content-Transfer-Encoding: encoding type
Content-Id: message id
Content-Description: textual explanation of nontextual contents
```

E-mail body

NON-ASCII CHARACTER SET SUPPORT

Message header

- content-type field
  - put in the header by the client program creating the e-mail for use by the client program used to display the received message
  - charset= optional parameter
    - if absent ASCII is assumed

Content-Type: text/plain; charset="ISO-8859-1"

- ISO-8859-1 extends the basic character set of ASCII to include many of the accented characters used in languages such as Spanish, French, German and Italian.
- US-ASCII is the standard character set used in the US

CONTENT LABELING

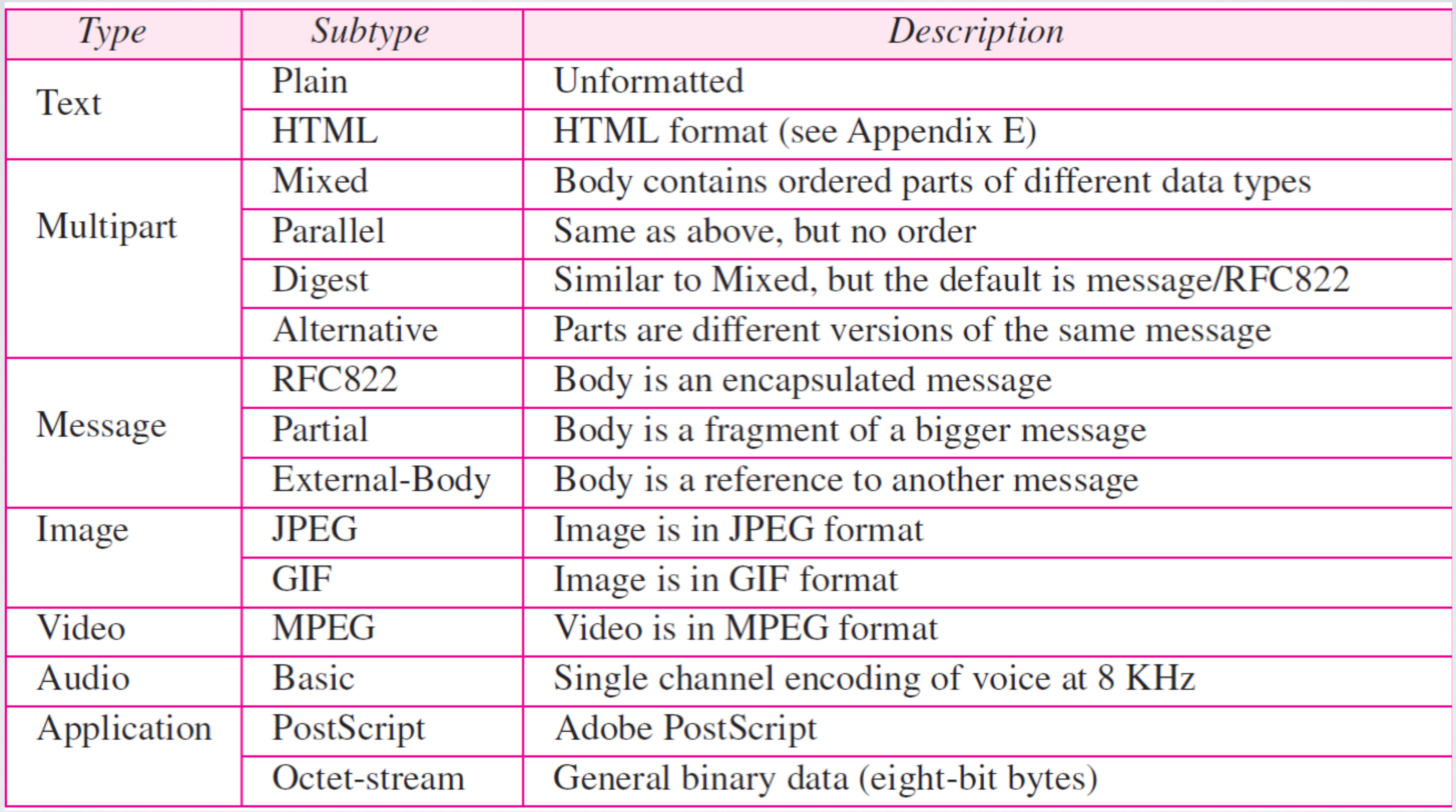
A set of registered MIME Types that map to specific file types

- MIME Types consist of :
  - a primary type
  - a sub type separated by a / (as text/html)

Common Mime Types:

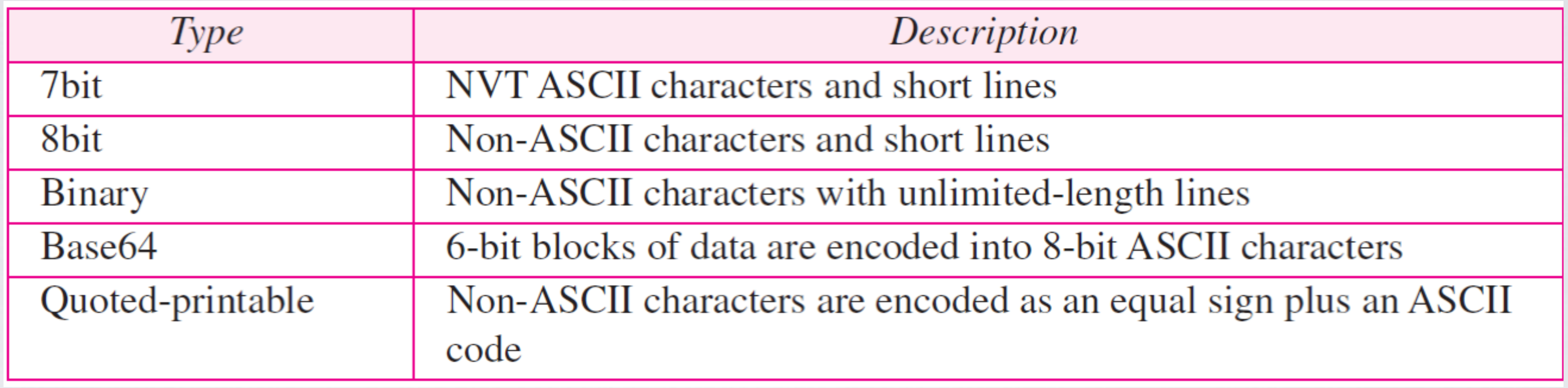
FileExtension	MIME Type	Description
.txt	text/plain	Plain text
.htm	text/html	Styled text in HTML format
.jpg	image/jpeg	Picture in JPEG format
.gif	image/gif	Picture in GIF format
.wav	audio/x-wave	Sound in WAVE format
.mp3	audio/mpeg	Music in MP3 format
.mpg	video/mpeg	Video in MPEG format
.zip	application/zip	Compressed file in PK-ZIP format

MIME TYPES/SUBTYPES



Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Appendix E)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to Mixed, but the default is message/RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 KHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (eight-bit bytes)

CONTENT-TRANSFER-ENCODING



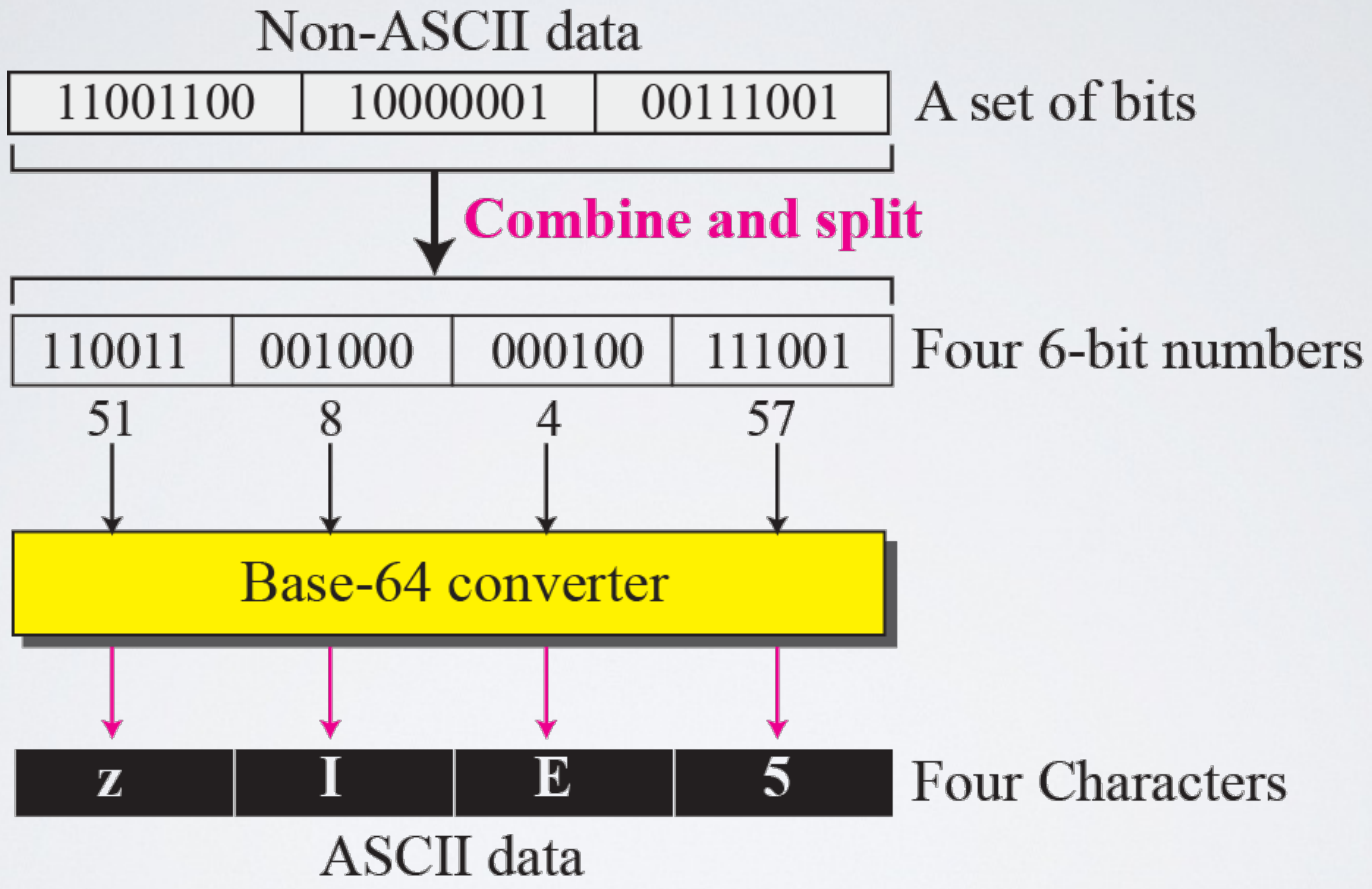
Type	*Description*
7bit	NVT ASCII characters and short lines
8bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data are encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters are encoded as an equal sign plus an ASCII code

NON-TEXT CONTENT

Non-textual content is encoded in ASCII for transmission and decoded back to its original format for display upon receipt

- MIME uses base 64 encoding (RFC 2045)
  - binary to text encoding scheme
  - targets A-Z, a-z, 0-9, +,/
- scheme:
  - take 3 bytes of data, put into a 24 bit buffer
  - extract 4 six-bits values
  - use each value as an index into: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
  - this yields 4 ASCII characters
  - use zero, one or two = symbols for padding (at the end)

BASE-64 ENCODING SCHEME



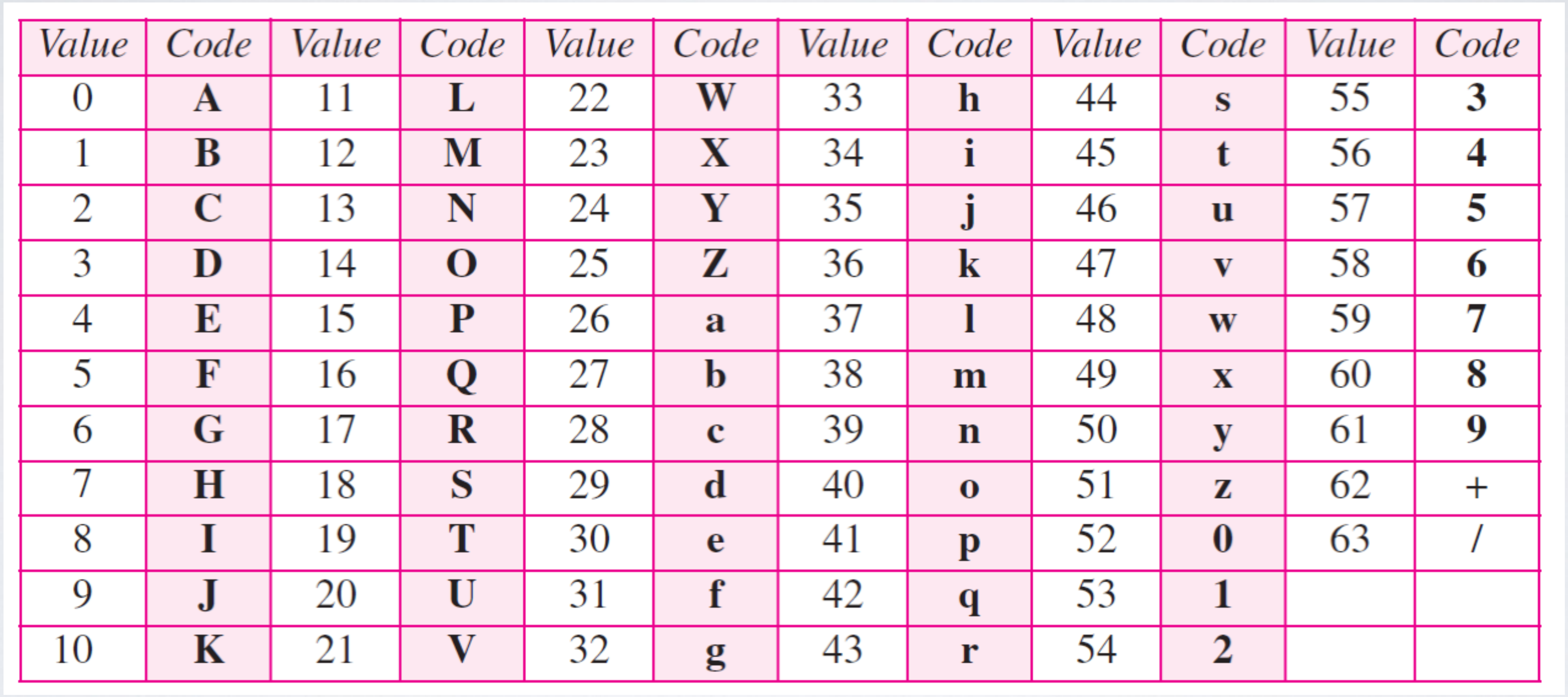
BASE-64 ENCODING EXAMPLE

Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure.

base64 encoded:

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0
aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1
c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0
aGUgY29udGludWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdl
LCBleGNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=

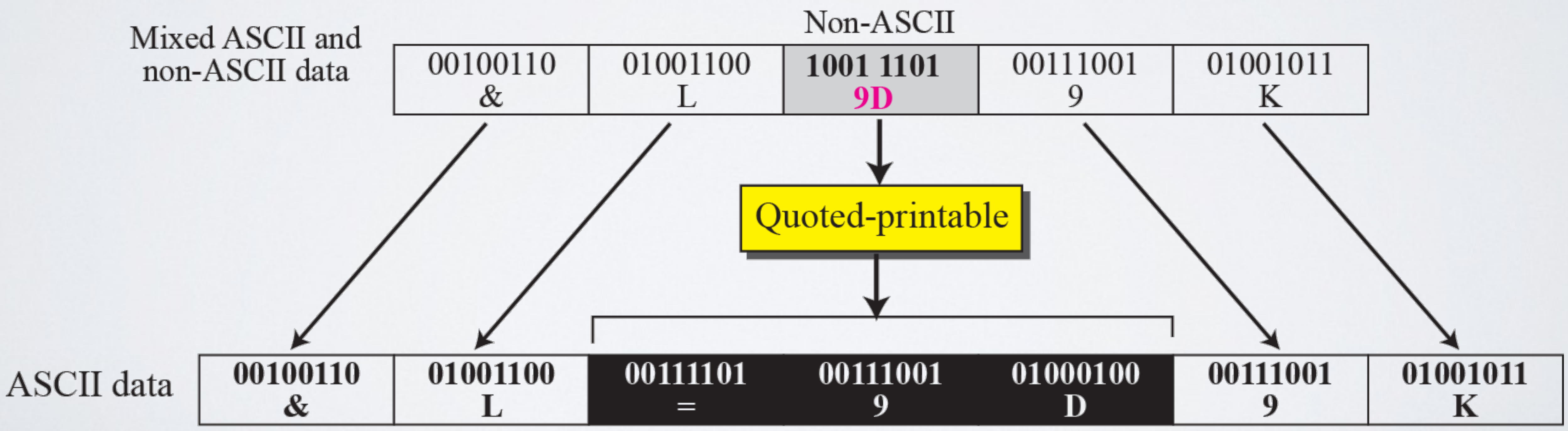
BASE-64 CONVERTING TABLE



Value	Code	Value	Code	Value	Code	Value	Code	Value	Code	Value	Code
0	**A**	11	**L**	22	**W**	33	**h**	44	**s**	55	**3**
1	**B**	12	**M**	23	**X**	34	**i**	45	**t**	56	**4**
2	**C**	13	**N**	24	**Y**	35	**j**	46	**u**	57	**5**
3	**D**	14	**O**	25	**Z**	36	**k**	47	**v**	58	**6**
4	**E**	15	**P**	26	**a**	37	**l**	48	**w**	59	**7**
5	**F**	16	**Q**	27	**b**	38	**m**	49	**x**	60	**8**
6	**G**	17	**R**	28	**c**	39	**n**	50	**y**	61	**9**
7	**H**	18	**S**	29	**d**	40	**o**	51	**z**	62	+
8	**I**	19	**T**	30	**e**	41	**p**	52	**0**	63	/
9	**J**	20	**U**	31	**f**	42	**q**	53	**1**		
10	**K**	21	**V**	32	**g**	43	**r**	54	**2**		

QUOTED-PRINTABLE ENCODING

- Any 8-bit byte value may be encoded with 3 characters: an '=' followed by two hexadecimal digits (0–9 or A–F) representing the byte's numeric value
- Non 8-bit byte values are ASCII chars from 33 to 126 (excluded 61, the '=' sign)
- special cases for SPACE and TAB



MULTIPART SUBTYPES

- Mixed - For sending files with different "Content-Type" headers.
- Digest - To send multiple text messages.
- Message - Contains any MIME email message, including any headers
- Alternative - Each part is an "alternative" version of the same (or similar) content (e.g., text + HTML)

- more subtypes…

MIME TYPES/SUBTYPES

```
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--
```

MIME TYPES/SUBTYPES

```
MIME-Version: 1.0
X-Mailer: MailBee.NET 8.0.4.428
Subject: test subject
To: kevinm@datamotion.com
Content-Type: multipart/mixed;
       boundary="XXXXboundary text"

--XXXXboundary text
Content-Type: multipart/alternative;
       boundary="XXXXboundary text"

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

This is the body text of a sample message.
--XXXXboundary text
Content-Type: text/html;
       charset="utf-8"
Content-Transfer-Encoding: quoted-printable
<pre>This is the body text of a sample message.</pre>

--XXXXboundary text
Content-Type: text/plain;
name="log_attachment.txt"
Content-Disposition: attachment;
filename="log_attachment.txt"
Content-Transfer-Encoding: base64

TUlNRS1WZXJzaW9uOiAxLjANClgtTWFpbGVyOiBNYWlsQmVlLk5FVCA4LjAuNC40MjgNClN1Ympl
```

PLAIN TEXT AND HTML

- modern graphic email clients allow use of HTML for the body
  - HTML email messages often include a plain text copy as well
- HTML messages should have an additional header: "Content-type: text/html". Most email programs insert this header automatically
- advantages of HTML include the ability to include in-line links and images, etc.
- disadvantages include the increased size of the email

SUBADDRESSING

Local subaddressing (or "+ subaddressing"):

- Provides support for "tags" in the local part of the email address
- Tags are defined following a separator character that is oftentimes "+"
- The address is an alias of a mailbox defined by the prefix preceding the separator.
  - querzoni+mastercourses@diag.uniroma1.it => querzoni@diag.uniroma1.it
- RFC5233

Domain subaddressing

- The local name can be used as domain subaddress
- Local address can be defined at will
  - mastercourses@querzoni.diag.uniroma1.it => querzoni@diag.uniroma1.it

EMAIL EXAMPLE

```
Delivered-To: querzoni@diag.uniroma1.it
Received: by 2002:a59:cb63:0:b0:49d:7dd7:7af1 with SMTP id c3csp711565vqv;
        Fri, 11 Oct 2024 15:05:46 -0700 (PDT)
X-Google-Smtp-Source: AGHT+IGIGOiU4RVgtBjHbzt4mT54JLoQYHfs5UosD6BSJBz6PgBjOqzYVEmmQFbBj
Return-Path: <crisis2024@easychair.org>
Received: from easychair.org (easychair.org. [213.136.76.235]) by mx.google.com with ESMTPS id
   5b1f17b1804b1-43111acd015si33937595e9.60.2024.10.11.15.05.46 for
   <querzoni@diag.uniroma1.it> (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
   Fri, 11 Oct 2024 15:05:46 -0700
Received: from easychair.org (m5801.contaboserver.net [213.136.76.235]) by easychair.org
   (8.15.2/8.15.2/Debian-18) with ESMTP id 49BM5kCe2465090 for <querzoni@diag.uniroma1.it>;
   Sat, 12 Oct 2024 00:05:46 +0200
Message-Id: <202410112205.49BM5kCe2465090@easychair.org>
Content-Type: text/plain; charset="UTF-8"
MIME-Version: 1.0
Date: Sat, 12 Oct 2024 00:05:46 +0200
From: Crisis 2024 <crisis2024@easychair.org>
To: Leonardo Querzoni <querzoni@diag.uniroma1.it>
Subject: Instructions for Final Paper Submission - CRiSIS 2024
Sender: crisis2024@easychair.org

Dear Leonardo Querzoni,
```

SECURITY CHALLENGES

The email system suffers from several well-known threats:

- Spam
- Phishing Attacks
- Malware/Ransomware Distribution
- Email Spoofing
- Lack of traceability
- Data Leakage
- Man-in-the-Middle (MITM) Attacks
- Business Email Compromise (BEC)
- Email Bombing

SIMPLE SPOOFING EXAMPLE

```
Delivered-To: leonardo.querzoni@uniroma1.it
Received: by 2002:a05:6520:2e05:b0:2a1:68a6:a6c0 with SMTP id df5csp164156lkb;
        Tue, 17 Sep 2024 04:50:45 -0700 (PDT)
Return-Path: <alessandro.lazzaro@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id2adb3069b0e04-5368707b163sor1622092e87.11.2024.09.17.04.50.42
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security); Tue, 17 Sep
        2024 04:50:42 -0700 (PDT)
In-Reply-To: <CAMSOQ=mLT1Vx5H5h8oVQnMz4nzFEkQ=UeMS0boAhf3v02tQLyw@mail.gmail.com>
Reply-To: servicecepol@gmail.com
From: Polizia criminale <alessandro.lazzaro@uniroma1.it>
Date: Tue, 17 Sep 2024 13:50:27 +0200
Message-ID: <CAMSOQ=kfzPNKa7gF3MAD_7hEKauRxxZVitjUSGC3nwsAkoGOGQ@mail.gmail.com>
Subject: ⭕RICHIESTA DI PROTEZIONE PERSONALE⭕
To: polizia-cr@info.it
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/mixed; boundary="00000000000070a7ab06224f4cad"

--00000000000070a7ab06224f4cad
```

SIMPLE SPOOFING EXAMPLE

```
Delivered-To: leonardo.querzoni@uniroma1.it
Received: by 2002:a05:6520:2e05:b0:2a1:68a6:a6c0 with SMTP id df5csp164156lkb;
        Tue, 17 Sep 2024 04:50:45 -0700 (PDT)
Return-Path: <alessandro.lazzaro@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id2adb3069b0e04-5368707b163sor1622092e87.11.2024.09.17.04.50.42
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security); Tue, 17 Sep
        2024 04:50:42 -0700 (PDT)
In-Reply-To: <CAMSOQ=mLT1Vx5H5h8oVQnMz4nzFEkQ=UeMS0boAhf3v02tQLyw@mail.gmail.com>
Reply-To: servicecepol@gmail.com
From: Polizia criminale <alessandro.lazzaro@uniroma1.it>
Date: Tue, 17 Sep 2024 13:50:27 +0200
Message-ID: <CAMSOQ=kfzPNKa7gF3MAD_7hEKauRxxZVitjUSGC3nwsAkoGOGQ@mail.gmail.com>
Subject: ⭕RICHIESTA DI PROTEZIONE PERSONALE⭕
To: polizia-cr@info.it
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/mixed; boundary="00000000000070a7ab06224f4cad"

--00000000000070a7ab06224f4cad
```

SIMPLE SPOOFING EXAMPLE

```
Delivered-To: leonardo.querzoni@uniroma1.it
Received: by 2002:a05:6520:2e05:b0:2a1:68a6:a6c0 with SMTP id df5csp164156lkb;
        Tue, 17 Sep 2024 04:50:45 -0700 (PDT)
Return-Path: <alessandro.lazzaro@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id2adb3069b0e04-5368707b163sor1622092e87.11.2024.09.17.04.50.42
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security); Tue, 17 Sep
        2024 04:50:42 -0700 (PDT)
In-Reply-To: <CAMSOQ=mLT1Vx5H5h8oVQnMz4nzFEkQ=UeMS0boAhf3v02tQLyw@mail.gmail.com>
Reply-To: servicecepol@gmail.com
From: Polizia criminale <alessandro.lazzaro@uniroma1.it>
Date: Tue, 17 Sep 2024 13:50:27 +0200
Message-ID: <CAMSOQ=kfzPNKa7gF3MAD_7hEKauRxxZVitjUSGC3nwsAkoGOGQ@mail.gmail.com>
Subject: ⭕RICHIESTA DI PROTEZIONE PERSONALE⭕
To: polizia-cr@info.it
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/mixed; boundary="00000000000070a7ab06224f4cad"

--00000000000070a7ab06224f4cad
```

SIMPLE SPOOFING EXAMPLE 2

```
Delivered-To: querzoni@dis.uniroma1.it
Received: by 2002:a59:c26b:0:b0:48e:d422:933f with SMTP id c11csp2768947vqr;
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <leonardo.querzoni+caf_=querzoni=dis.uniroma1.it@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 2adb3069b0e04-537a86516a3sor234209e87.25.2024.09.24.00.18.04
        for <querzoni@dis.uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <gaia.fiore@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 41be03b00d2f7-7e6b7c4586fsor419004a12.4.2024.09.24.00.18.02
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:03 -0700 (PDT)
MIME-Version: 1.0
From: "*SAPIENZAN*" <gaia.fiore@uniroma1.it>
Date: Tue, 24 Sep 2024 09:17:51 +0200
Message-ID: <CAHkmBVPVrn2rHkcg3EhhOfoA+1EHMLmUYYQ+G0KBmYM2ZBSPzA@mail.gmail.com>
Subject: QUESTA AZIONE È OBBLIGATORIA
To: undisclosed-recipients:;
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/alternative; boundary="0000000000004e74070622d84ed1"

--0000000000004e74070622d84ed1
```

SIMPLE SPOOFING EXAMPLE 2

```
Delivered-To: querzoni@dis.uniroma1.it
Received: by 2002:a59:c26b:0:b0:48e:d422:933f with SMTP id c11csp2768947vqr;
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <leonardo.querzoni+caf_=querzoni=dis.uniroma1.it@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 2adb3069b0e04-537a86516a3sor234209e87.25.2024.09.24.00.18.04
        for <querzoni@dis.uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <gaia.fiore@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 41be03b00d2f7-7e6b7c4586fsor419004a12.4.2024.09.24.00.18.02
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:03 -0700 (PDT)
MIME-Version: 1.0
From: "*SAPIENZAN*" <gaia.fiore@uniroma1.it>
Date: Tue, 24 Sep 2024 09:17:51 +0200
Message-ID: <CAHkmBVPVrn2rHkcg3EhhOfoA+1EHMLmUYYQ+G0KBmYM2ZBSPzA@mail.gmail.com>
Subject: QUESTA AZIONE È OBBLIGATORIA
To: undisclosed-recipients:;
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/alternative; boundary="0000000000004e74070622d84ed1"

--0000000000004e74070622d84ed1
```

SIMPLE SPOOFING EXAMPLE 2

```
Delivered-To: querzoni@dis.uniroma1.it
Received: by 2002:a59:c26b:0:b0:48e:d422:933f with SMTP id c11csp2768947vqr;
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <leonardo.querzoni+caf_=querzoni=dis.uniroma1.it@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 2adb3069b0e04-537a86516a3sor234209e87.25.2024.09.24.00.18.04
        for <querzoni@dis.uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <gaia.fiore@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 41be03b00d2f7-7e6b7c4586fsor419004a12.4.2024.09.24.00.18.02
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:03 -0700 (PDT)
MIME-Version: 1.0
From: "*SAPIENZAN*" <gaia.fiore@uniroma1.it>
Date: Tue, 24 Sep 2024 09:17:51 +0200
Message-ID: <CAHkmBVPVrn2rHkcg3EhhOfoA+1EHMLmUYYQ+G0KBmYM2ZBSPzA@mail.gmail.com>
Subject: QUESTA AZIONE È OBBLIGATORIA
To: undisclosed-recipients:;
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/alternative; boundary="0000000000004e74070622d84ed1"

--0000000000004e74070622d84ed1
```

SIMPLE SPOOFING EXAMPLE 2

```
Delivered-To: querzoni@dis.uniroma1.it
Received: by 2002:a59:c26b:0:b0:48e:d422:933f with SMTP id c11csp2768947vqr;
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <leonardo.querzoni+caf_=querzoni=dis.uniroma1.it@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 2adb3069b0e04-537a86516a3sor234209e87.25.2024.09.24.00.18.04
        for <querzoni@dis.uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:05 -0700 (PDT)
Return-Path: <gaia.fiore@uniroma1.it>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id 41be03b00d2f7-7e6b7c4586fsor419004a12.4.2024.09.24.00.18.02
        for <leonardo.querzoni@uniroma1.it> (Google Transport Security);
        Tue, 24 Sep 2024 00:18:03 -0700 (PDT)
MIME-Version: 1.0
From: "*SAPIENZAN*" <gaia.fiore@uniroma1.it>
Date: Tue, 24 Sep 2024 09:17:51 +0200
Message-ID: <CAHkmBVPVrn2rHkcg3EhhOfoA+1EHMLmUYYQ+G0KBmYM2ZBSPzA@mail.gmail.com>
Subject: QUESTA AZIONE È OBBLIGATORIA
To: undisclosed-recipients:;
Bcc: leonardo.querzoni@uniroma1.it
Content-Type: multipart/alternative; boundary="0000000000004e74070622d84ed1"

--0000000000004e74070622d84ed1
```

SIMPLE SPOOFING EXAMPLE 3

```
Return-Path: <yanting@united.com.sg>
Delivered-To: admin@malware-traffic-analysis.net
Received: from united.com.sg (unknown [71.19.248.52])
        by (information removed] (Postfix) with ESMTP id 4DZZ0g5hnCz5vMF
        for < admin@malware-traffic-analysis.net>; Tue, 9 Feb 2021 07:15:10 +0000 (UTC)
From: "Yan Ting"<yanting@united.com.sg>
To: admin@malware-traffic-analysis.net
Subject: united scientific equipment
Date: 08 Feb 2021 23:15:11 -0800
Message-ID: <20210208231511.B2A19DA7B4F9872F@united.com.sg>
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="--=_NextPart_000_0012_16021DE4.1EB30607"

This is a multi-part message in MIME format.

_=_NextPart_000_0012_16021DE4.1EB30607
Content-Type: text/html;
        charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<! DOCTYPE HTML PUBLIC "-//W3C//DID HTML 4.01 Iransitional//EN" "http://www.=
w3.org/TR/htm14/loose.dtd">

<HTML xmlns:0 =3D "urn: schemas-microsoft-com:office:office" xmlns:v =3D "ur=
n:schemas-microsoft-com:vm]"><HEAD>
```

SECURITY CHALLENGES

Most of these threats stem from the lack of adequate security guarantees in the standard email architecture and its protocols

- No sender authentication means no way to trace the origin of an email
- No end-to-end encryption means no way to enforce content confidentiality

SENDER AUTHENTICATION

Sender Policy Framework (SPF)

- Verifies if the sending server is authorized to send emails on behalf of the domain - RFC 4408

DomainKeys Identified Mail (DKIM)

- Provides a cryptographic signature to ensure the email's integrity and authenticity - RFC 4871

Domain-based Message Authentication, Reporting & Conformance (DMARC)

- Enforces policies based on SPF and DKIM and sends reports back to domain owners on authentication failures.

Authenticated Receiver Chain (ARC)

- Allows for maintaining the validity of DMARC policies across intermediaries.

SENDER POLICY FRAMEWORK

- **SPFv1 (or SPF Classic)** protects the sender address (in envelope) by allowing the owner of a domain to specify a mail sending policy, namely which mail servers are authorized to send mail from the domain, using special DNS records (SPF, type 99)
- If server accepts the sender, recipients and body of message, it should insert a Return-Path field in the message header in order to save the sender address
    - While the address in the Return-Path often matches other originator addresses in the mail header such as From or Sender, this is not necessarily the case, and SPF does not prevent forgery of these other addresses
- Guarntees Sender Server Authorization against mail spoofing.

SPF BY EXAMPLE

- Bob owns domain *example.net*.
- He also sometimes sends mail through his GMail account and contacted GMail's support to identify the correct SPF record for GMail.
- Since he often receives bounces about messages he didn't send, he decides to publish an SPF record in order to reduce the abuse of his domain in e-mail envelopes:

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

- Check spf record

```
❯ dig +noall +answer uniroma1.it txt
```

SPF BY EXAMPLE

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

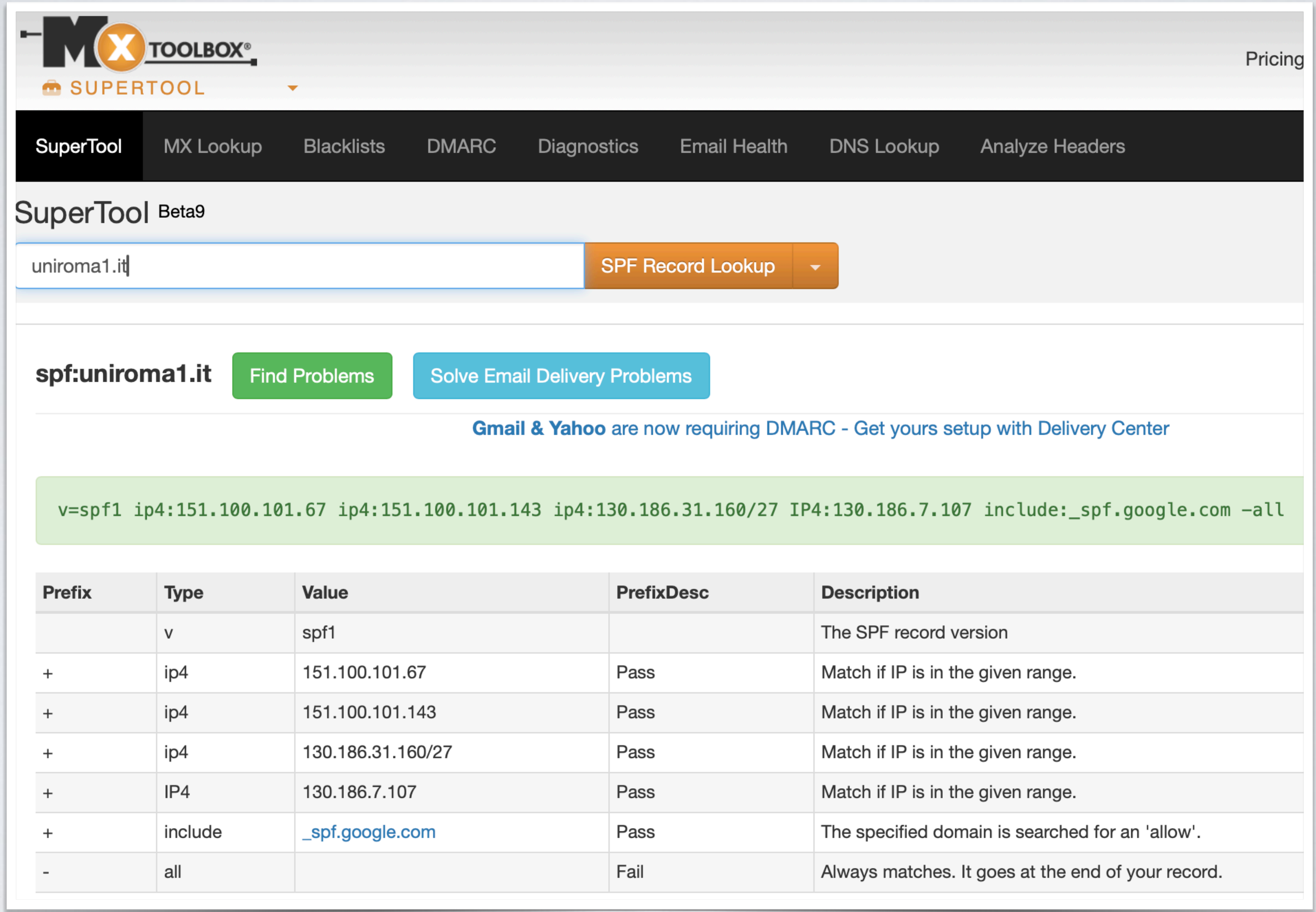
SPF record item	Description
`v=spf1`	SPF version 1
`mx`	The incoming mail servers (MXes) of the domain are authorized to also send mail for example.net
`a:pluto.example.net`	the machine pluto.example.net is authorized, too
`include:aspmx.googlemail.com`	every mail server considered legitimate by gmail.com is legitimate for example.net
`~all`	all other servers are not authorized, and email they generate should be marked as suspicious

SPF BY EXAMPLE

- When an email is sent from the domain, the receiving mail server queries the DNS for the domain's SPF record to verify if the IP address of the sending server is listed in the SPF record.
  - **Pass**: If the sending server's IP matches one of the authorized IPs in the SPF record, the email passes the SPF check.
  - **Fail**: If the sending IP is not listed, the email fails the SPF check. The receiving server can then choose to reject, flag, or accept the message based on local policy.

```
<XXXX.YYYY@gmail.com>: host gmail-smtp-in.l.google.com[173.194.78.26]
said: 550-5.7.1 [aa.bb.cc.dd] The IP you're using to send mail is not
authorized to 550-5.7.1 send email directly to our servers. Please use the
SMTP relay at your 550-5.7.1 service provider instead. Learn more at 550
5.7.1 http://support.google.com/mail/bin/answer.py?answer=10336
fl4si3665795wib.12 - gsmtp (in reply to end of DATA command)
```

SPF BY EXAMPLE



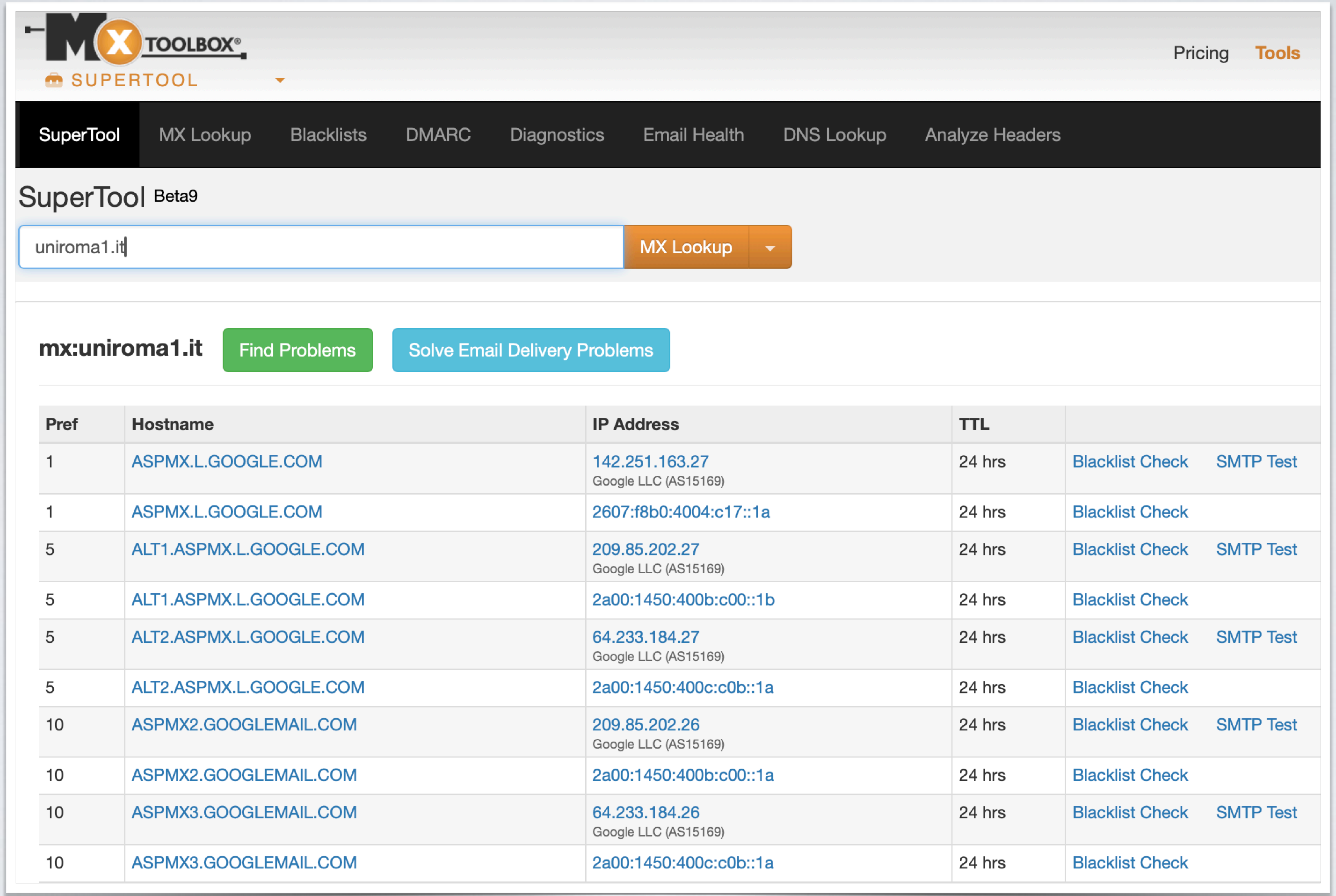
spf:uniroma1.it

Gmail & Yahoo are now requiring DMARC - Get yours setup with Delivery Center

```
v=spf1 ip4:151.100.101.67 ip4:151.100.101.143 ip4:130.186.31.160/27 IP4:130.186.7.107 include:_spf.google.com -all
```

Prefix	Type	Value	PrefixDesc	Description
	v	spf1		The SPF record version
+	ip4	151.100.101.67	Pass	Match if IP is in the given range.
+	ip4	151.100.101.143	Pass	Match if IP is in the given range.
+	ip4	130.186.31.160/27	Pass	Match if IP is in the given range.
+	IP4	130.186.7.107	Pass	Match if IP is in the given range.
+	include	_spf.google.com	Pass	The specified domain is searched for an 'allow'.
-	all		Fail	Always matches. It goes at the end of your record.

SPF BY EXAMPLE



mx:uniroma1.it

Pref	Hostname	IP Address	TTL		
1	ASPMX.L.GOOGLE.COM	142.251.163.27 Google LLC (AS15169)	24 hrs	Blacklist Check	SMTP Test
1	ASPMX.L.GOOGLE.COM	2607:f8b0:4004:c17::1a	24 hrs	Blacklist Check	
5	ALT1.ASPMX.L.GOOGLE.COM	209.85.202.27 Google LLC (AS15169)	24 hrs	Blacklist Check	SMTP Test
5	ALT1.ASPMX.L.GOOGLE.COM	2a00:1450:400b:c00::1b	24 hrs	Blacklist Check	
5	ALT2.ASPMX.L.GOOGLE.COM	64.233.184.27 Google LLC (AS15169)	24 hrs	Blacklist Check	SMTP Test
5	ALT2.ASPMX.L.GOOGLE.COM	2a00:1450:400c:c0b::1a	24 hrs	Blacklist Check	
10	ASPMX2.GOOGLEMAIL.COM	209.85.202.26 Google LLC (AS15169)	24 hrs	Blacklist Check	SMTP Test
10	ASPMX2.GOOGLEMAIL.COM	2a00:1450:400b:c00::1a	24 hrs	Blacklist Check	
10	ASPMX3.GOOGLEMAIL.COM	64.233.184.26 Google LLC (AS15169)	24 hrs	Blacklist Check	SMTP Test
10	ASPMX3.GOOGLEMAIL.COM	2a00:1450:400c:c0b::1a	24 hrs	Blacklist Check	

SPF LIMITATIONS

Email forwarding

- When an email is forwarded, the SPF check can fail.
  - The forwarder's IP address may not be included in the original sender's SPF record

Shared or third-party email services

- Organizations often use third-party services (like CRM platforms, marketing tools, or cloud services) to send emails on their behalf. These services may not be listed in the domain's SPF record.

Mailing lists

- Emails sent through mailing lists often pass through multiple servers before reaching recipients. Mailing list software sometimes modifies the email headers, including the sender's information, which can break SPF validation.

SPF LIMITATIONS

SPF record size limitations

- The size limit for an SPF record is 450 characters minus the length of the domain name and the length of any other TXT record value. This is the upper exclusive limit recommended in RFC 7208, the document that specifies SPF.

No protection for email body

- SPF only authenticates the sending server's IP address and does not protect the email's content or prevent modification of the message body or attachments.

No full sender verification

- SPF only verifies the sending server's IP address against the envelope sender (the "MAIL FROM" address), not the From: header that the user sees.

SPF LIMITATIONS

Misconfiguration

- Incorrectly configured SPF records, such as syntax errors or forgetting to include authorized sending IPs, can cause legitimate emails to fail SPF checks.

Complex mail routing

- In some scenarios with complex email routing (e.g., email relays, hybrid systems), SPF checks can fail because the originating IP doesn't match the IP specified in the SPF record.

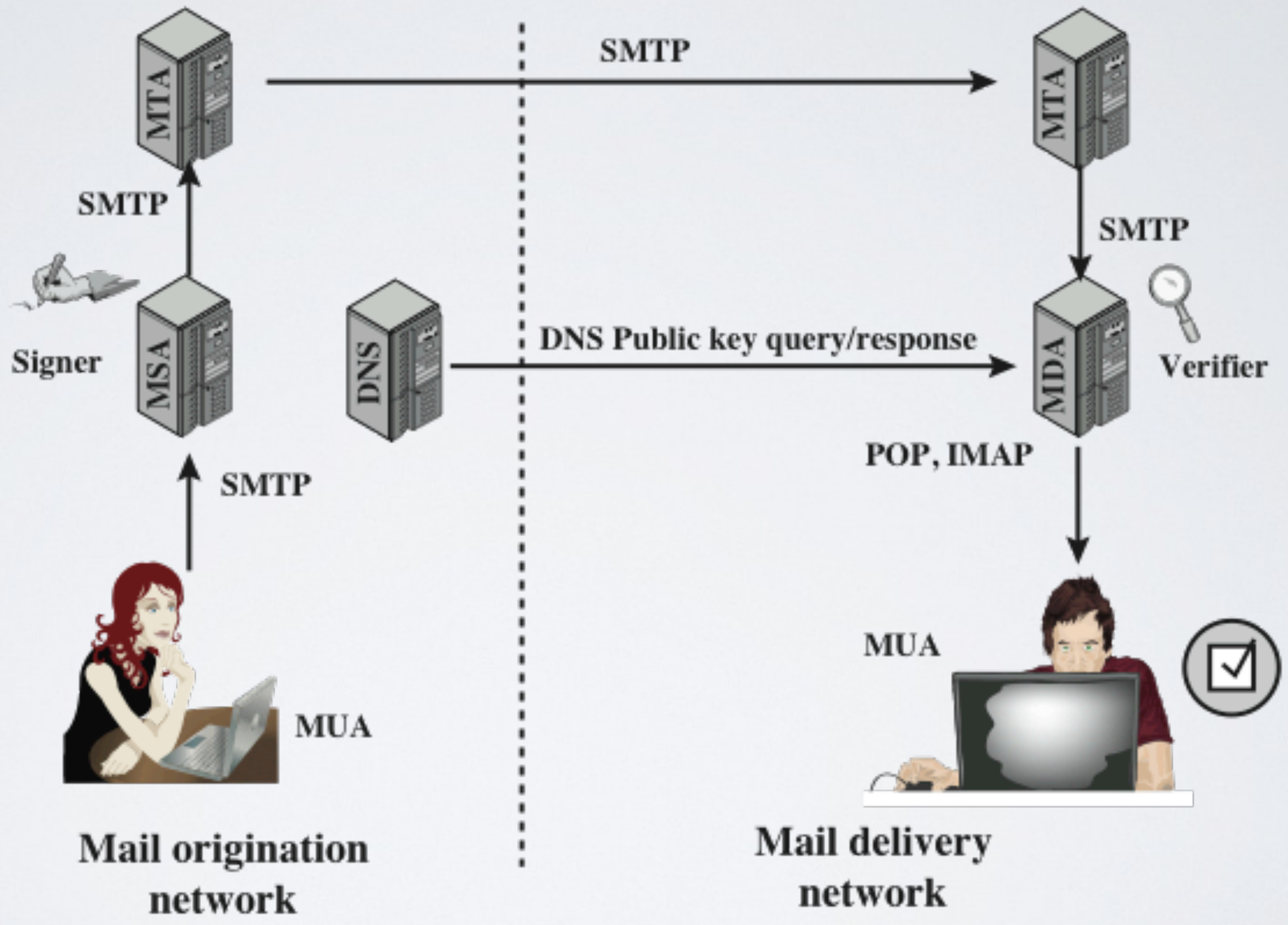
Outbound-Only Security

- SPF is mainly used to authenticate outbound email servers, but it doesn't offer any protection or guarantee about the inbound email a domain receives.

DKIM

- **DomainKeys Identified Mail (DKIM)** - RFC 4871 - is a specification for cryptographically signing e-mail messages, permitting a signing domain to claim responsibility for a message.
- Message recipients (or agents acting in their behalf) can verify the signature by querying the signer's domain to retrieve the public key
- What DKIM guarantees:
  - Email content integrity
  - Domain authentication
  - Non repudiation

POSSIBLE DKIM DEPLOYMENT



HOW DOES DKIM WORK

1 - Generating the Signature (Signing Process):

- When a domain sends an email, the sender's mail server (or an intermediary) generates a cryptographic signature based on specific parts of the message (e.g., the body and certain headers, like the subject or the From field). The signing is done using a private key that only the domain owner controls.
- The signature is added to the email as a new header: the DKIM-Signature header. This signature contains information like:
  - The hashing algorithm used.
  - The domain that is taking responsibility for the email.
  - The email parts that were signed (headers and body).
  - The signature itself (a base64-encoded value).

HOW DOES DKIM WORK

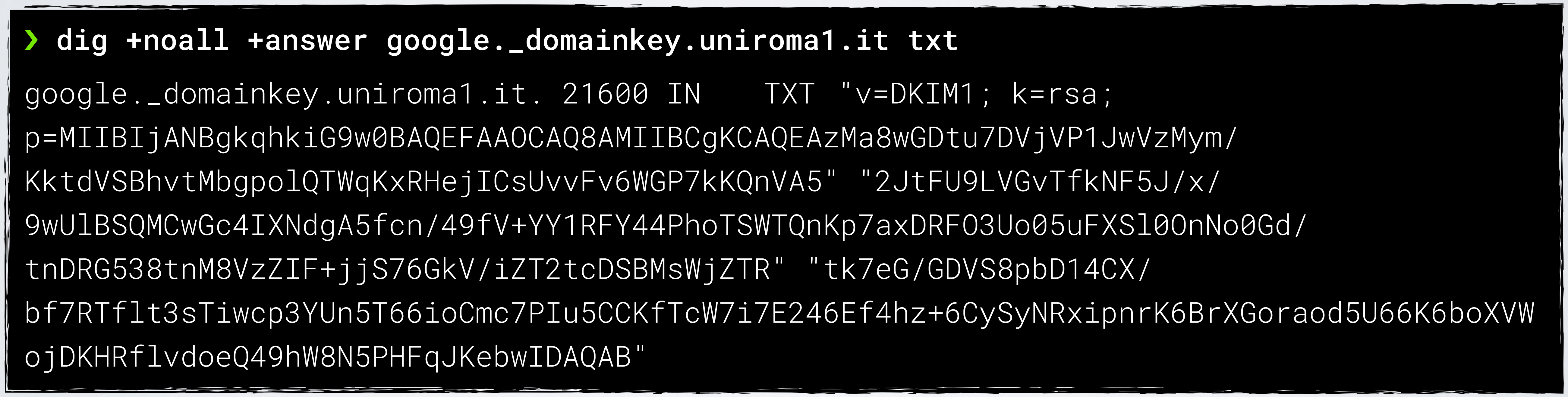
1 - Generating the Signature (Signing Process):

```
DKIM-Signature:
       v=1;
       a=rsa-sha256;
       c=relaxed/relaxed;
       d=diag.uniroma1.it;
       s=google;
       t=1728828422;
       x=1729433222;
       dara=google.com;
       h=to:subject:message-id:date:from:mime-version:from:to:cc:subject:date:message-
       id:reply-to;
       bh=t6DYHnvgeJvZ02sgmWVU/4X9LTVieRyKbl+FRWPu3Co=;
       b=gx0VlcD55ZtEOTnE2FJJSSgt8Padr87pkhbkWw7FbuIyWY2O0QvKy52DD6DsuiT2oj
         q5/jSGM4y1b0XKHM7CU1Fhk+ScKi16hj8HdezlpnFOZbbiKS43uysV8lLfbv5SOaDEFv
         REmsa4Az8duRs1fYEsQ9ixRu5RPOLRgCBcxb0=
```

HOW DOES DKIM WORK

2 - Public Key Published in DNS:

- The domain that sends the email publishes the corresponding public key in the Domain Name System (DNS) as a TXT record. The public key is used to verify the authenticity of the signature.
- The DNS record also specifies the selector (a prefix to differentiate between multiple keys) and the policy the sender wants to use for DKIM.



```
❯ dig +noall +answer google._domainkey.uniroma1.it txt
google._domainkey.uniroma1.it. 21600 IN    TXT "v=DKIM1; k=rsa;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzMa8wGDtu7DVjVP1JwVzMym/
KktdVSBhvtMbgpolQTWqKxRHejICsUvvFv6WGP7kKQnVA5" "2JtFU9LVGvTfkNF5J/x/
9wUlBSQMCwGc4IXNdgA5fcn/49fV+YY1RFY44PhoTSWTQnKp7axDRFO3Uo05uFXSl0OnNo0Gd/
tnDRG538tnM8VzZIF+jjS76GkV/iZT2tcDSBMsWjZTR" "tk7eG/GDVS8pbD14CX/
bf7RTflt3sTiwcp3YUn5T66ioCmc7PIu5CCKfTcW7i7E246Ef4hz+6CySyNRxipnrK6BrXGoraod5U66K6boXVW
ojDKHRflvdoeQ49hW8N5PHFqJKebwIDAQAB"
```

SELECTORS

To support multiple concurrent public keys per signing domain, key namespace is subdivided using selectors

- for example selectors might indicate the names of office locations, the signing date, or even the individual user

Selectors are useful to implement some important use cases

- domains that want to delegate signing capability for a specific address for a given duration to a partner, such as an advertising provider or other outsourced function
- domains that want to allow frequent travelers to send messages locally without the need to connect with a particular MSA.
- "affinity" domains (e.g., college alumni associations) that provide forwarding of incoming mail, but that do not operate a MSA for outgoing mail

HOW DOES DKIM WORK

3 - Verification Process:

- When an email is received, the recipient's mail server looks at the DKIM signature header to see which domain signed the message and which selector to use. It retrieves the corresponding public key from the domain's DNS.
- The server then verifies the digital signature by comparing it with the hash of the received message's content. If the signature matches, it confirms that the email:
  - Hasn't been altered in transit (message integrity).
  - Is indeed authorized by the sender's domain (authenticity).

HOW DOES DKIM WORK

3 - Passing or Failing the DKIM Check:

- **Pass**: If the signature is valid, the message is authenticated, and the recipient server knows that the message came from an authorized source and hasn't been tampered with.
- **Fail**: If the signature doesn't match (due to changes in the content during transit) or if the public key in DNS doesn't match, the message fails the DKIM check.
  - The final action (e.g., rejection, quarantine) depends on other factors (e.g. DMARC policy).

DKIM AND SPF

How does DKIM compares to SPF?

- SPF authenticates the sending server's IP address against the domain's SPF record.
- DKIM authenticates the email content by verifying the signature against a public key stored in DNS.

Limitations

- DKIM doesn't authenticate the visible From address. It only verifies the email's authenticity from the domain that signed the message.
- Email forwarding can break DKIM if the forwarding server alters the content (e.g., adding disclaimers).
- It doesn't provide encryption or privacy—only authenticity and integrity.

CANONICALIZATION

- e-mail servers and relay systems may modify email in transit, potentially invalidating a signature
- headers are subjected to a canonicalization algorithm
  - **relaxed** (tolerating) or **simple** (strict)
- bodies are also subjected to a canonicalization algorithm
  - choices for header/body are independent
- see RFC 4871 for details

DKIM EXAMPLE

```
DKIM-Signature:
    v=1;
    a=rsa-sha256;
    c=relaxed;
    d=example.com;
    s=mail;
    h=From:To:Subject:Date;
    bh=MTIzNDU2Nzg5MDEyMzQ1Njc4OQ==;
    b=abcdefghijklmnopqrstuvwxyz1234
    567890abcdefghijklmnopqrstuvwxyz
```

a = Hash/signing algorithm

q = Algorithm for getting public key

d = Signing domain

i = Signing identity

s = Selector

c = Canonicalization algorithm (*relaxed/simple*)

t = Signing time (seconds since 1/1/1970)

x = Expiration time

h = List of headers included in signature; *DKIM-Signature* is implied

b = The signature itself

bh = The hash of the canonicalized body part of the message

v = DKIM version

DMARC

- **Domain-based Message Authentication, Reporting, and Conformance**, is a technical standard (RFC 7489) that helps protect email senders and recipients from spam/spoofing/phishing.
- DMARC allows an organization to publish a policy that defines its email authentication practices and provides instructions to receiving mail servers for how to enforce them.
- Specifically, DMARC establishes a method for a domain owner to:
  - Publish its email authentication practices
  - State what actions should be taken on mail that fails authentication checks
  - Enable reporting of these actions taken on mail claiming to be from its domain
- Puts together SPF and DKIM

HOW DOES DMARC WORK?

1) A domain administrator publishes the policy defining its email authentication practices and how receiving mail servers should handle mail that violates this policy.

- This DMARC policy is listed as part of the domain's overall DNS records.
- A DMARC record is included in an organization's DNS database. It is a specially-formatted version of a standard DNS TXT record with a particular name: `_dmarc.mydomain.com`

HOW DOES DMARC WORK?

DMARC record example

```
❯ dig +noall +answer _dmarc.uniroma1.it txt
_dmarc.uniroma1.it.   21600    IN   TXT  "v=DMARC1; p=reject; pct=10;
rua=mailto:lxoyu6mk@ag.eu.dmarcadvisor.com,mailto:dmarc-ar@uniroma1.it;
ruf=mailto:lxoyu6mk@fr.eu.dmarcadvisor.com,mailto:dmarc-f@uniroma1.it;"
```

- **`v=DMARC1`** specifies the DMARC version
- **`p=reject`** specifies the preferred treatment, or DMARC policy
- **`rua=mailto:…`** is the mailbox to which aggregate reports should be sent
- **`ruf=mailto:…`** is the mailbox to which forensic reports should be sent
- **`pct=10`** is the percentage of mail to which the domain owner would like to have its policy applied

HOW DOES DMARC WORK?

DMARC record example

```
❯ dig +noall +answer _dmarc.uniroma1.it txt
_dmarc.uniroma1.it.   21600    IN   TXT  "v=DMARC1; p=reject; pct=10;
rua=mailto:lxoyu6mk@ag.eu.dmarcadvisor.com,mailto:dmarc-ar@uniroma1.it;
ruf=mailto:lxoyu6mk@fr.eu.dmarcadvisor.com,mailto:dmarc-f@uniroma1.it;"
```

The DMARC specification provides three choices for domain owners to specify their preferred treatment of mail that fails DMARC validation checks. These policies are:

- **none**: treat the mail the same as it would be without any DMARC validation
- **quarantine**: accept the mail but place it somewhere other than the recipient's inbox (typically the spam folder)
- **reject**: reject the message outright

HOW DOES DMARC WORK?

2) When an inbound mail server receives an incoming email, it uses DNS to look up the DMARC policy for the domain contained in the message's "From" (RFC 5322) header.

The inbound server then checks the message for three key factors:

- Does the message's DKIM signature validate?
- Did the message come from IP addresses allowed by the sending domain's SPF records?
- Do the headers in the message show proper "domain alignment"?

HOW DOES DMARC WORK?

2) When an inbound mail server receives an incoming email, it uses DNS to look up the DMARC policy for the domain contained in the message's "From" (RFC 5322) header.

"Domain alignment" is a concept in DMARC that expands the domain validation intrinsic to SPF and DKIM. 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= domain must match

HOW DOES DMARC WORK?

3)With this information, the server is ready to apply the sending domain's DMARC policy to decide whether to accept, reject, or otherwise flag the email message.

4) After using DMARC policy to determine the proper disposition for the message, the receiving mail server will report the outcome to the sending domain owner.

HOW DOES DMARC WORK?

DMARC reports are generated by inbound mail servers as part of the DMARC validation process. There are two formats of DMARC reports:

- **Aggregate reports**, which are XML documents showing statistical data about the messages received that claimed to be from a particular domain. Data reported includes authentication results and message disposition. Aggregate reports are designed to be machine-readable.
- **Forensic reports**, which are individual copies of messages which failed authentication, each enclosed in a full email message using a special format called AFRF. Forensic report can be useful both for troubleshooting a domain's own authentication issues and for identifying malicious domains and web sites.

DMARC CHECK OUTCOMES

Outcomes from protocol checks are reported in the mail headers

```
Authentication-Results: mx.google.com;
       dkim=pass header.i=@diag.uniroma1.it header.s=google header.b=gx0VlcD5;
       spf=pass (google.com: domain of querzoni@diag.uniroma1.it designates
      209.85.220.41 as permitted sender) smtp.mailfrom=querzoni@diag.uniroma1.it;
       dmarc=pass (p=NONE sp=NONE dis=NONE) header.from=diag.uniroma1.it;
       dara=pass header.i=@gmail.com
```

DMARC LIMITATIONS

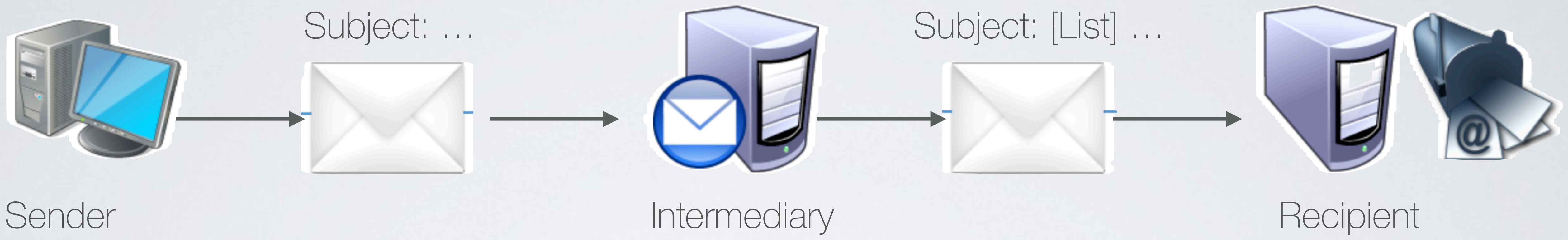
Domains with strict DMARC policies (p=reject) may see legitimate messages blocked if they go through indirect mailflows such as mailing lists, forwarding, or filtering services

- Forwarding causes SPF to fail even if origin was legit
- Forwarders often alter messages, breaking DKIM
  - Disclaimers and footers
  - Virus scan results
  - Removed attachments
  - Mailing list subject tags

Some text for ARC-related slides is from The Trusted Domain Project

DMARC LIMITATIONS

Example:



- Intermediary sends the message from a new IP address, causing SPF to fail to verify for Sender's domain
- Intermediary changes the message contents, causing Sender's DKIM signature to fail to verify

AUTHENTICATED RECEIVED CHAIN

ARC helps preserve the authentication status of an email that passes through multiple intermediaries or forwarding services.

- **Preserves Authentication**: ARC ensures that if an email passes SPF, DKIM, and DMARC checks at the original source, this authenticated status can be carried forward to subsequent servers or recipients.
- **Accountability for Intermediaries**: Each intermediary in the email flow adds its own ARC authentication results, creating a verifiable chain of who has handled the message and whether they altered the original email.
- **Improved Email Deliverability**: ARC can improve deliverability for emails sent through mailing lists, email forwarding services, or resending platforms by helping them pass DMARC checks that would otherwise fail.

ARC does not replace DMARC but works alongside it. ARC is concerned with **preserving the email's authentication chain**.

AUTHENTICATED RECEIVED CHAIN

What ARC does not do:

- Does not say anything about "trustworthiness" of the message sender or intermediaries
- Says nothing about the contents of the message
- Intermediaries might still inject bad content
- Intermediaries might remove some or all ARC headers

AUTHENTICATED RECEIVED CHAIN

ARC introduces three header fields:

`ARC-Authentication-Results:`(AAR) Archived copy of Authentication-Results:. Records the results of any authentication checks (like SPF or DKIM) performed by the intermediary.

`ARC-Seal:`(AS) Includes a DKIM-style cryptographic signature that verifies the authenticity of the ARC chain up to this point, confirming that no one has tampered with the previous ARC headers.

`ARC-Message-Signature:`(AMS) A DKIM-style signature of the entire message except ARC-Seal: headers. Helps preserving the integrity of the message as it passes through intermediaries.

AUTHENTICATED RECEIVED CHAIN

The `ARC-Seal:` header contains the following fields:

- `b=` is a signature of all ARC headers; no non-ARC headers
- `a=/d=/s=` fields match the corresponding DKIM tags
  - Same key format and DNS records as for DKIM
  - Can use your DKIM keys for ARC
  - Can use separate keys per local policy or preference
- `cv=` indicates whether ARC chain validated as received by the reporting intermediary
- `i=` tag is a sequence number for ARC header sets

AUTHENTICATED RECEIVED CHAIN

To sign a modification, an intermediate server performs the following steps:

1. The content of `Authentication-Results:` content is copied into a new `ARC-Authentication-Results:` header, prefixed to the message
2. Calculates the `ARC-Message-Signature:` for the message, including the latest AAR header, and prefixed to the message
   - Must not include any `ARC-Seal:` headers
3. `ARC-Seal:` is calculated and prefixed

ARC headers are prefixed per common practice, but order of appearance is not critical for validation

NOTE: a new ARC header must be inserted ONLY if the intermediary makes changes that may break DMARC checks.

AUTHENTICATED RECEIVED CHAIN

The `i=` sequence tag is used to order the ARC headers for various operations:

- Allows multiple ARC header sets to be grouped easily and correctly
- Eliminates reliance on the order of headers being inserted – or not being altered
- Compare with order of insertion of various authentication, content scanning, or `Received:` headers

AUTHENTICATED RECEIVED CHAIN

Steps to check an ARC chain:

- Verify the `ARC-Seal:` — Ensure the cryptographic signature in the seal matches the domain and selector used by the intermediate server.
- Verify the `ARC-Message-Signature:` — Check that the signature in the ARC-Message-Signature correctly matches the email's original content and headers.
- Check `Authentication Results:` — Review the ARC-Authentication-Results to see if SPF, DKIM, and other checks passed for the original email.
- Evaluate the Chain — Continue verifying the chain of ARC headers through all the intermediaries. Each hop adds another ARC-Seal, which must be validated to maintain the integrity of the chain.

AUTHENTICATED RECEIVED CHAIN

Example of ARC Headers:

```
ARC-Seal: i=3; a=rsa-sha256; t=1608562589; cv=pass;
      d=google.com; s=arc-20160816;
      b=eDQJjq5aDJrTBzWbJU0dt46mObilIR3UFyYJcr6lYC0g3n3EosZUStgpmuSXSV1[…]

ARC-Message-Signature: i=3; a=rsa-sha256; c=relaxed/relaxed; d=google.com;
      s=arc-20160816;
      h=list-unsubscribe:list-archive:list-help:list-post:list-id […]
      bh=C+v3YGJvRDYuOojq0nvM9h7NPqS2uE+OtdOmVd6jGOc=;
      b=cWvAcmX8L3q3iNVeBQKsgYT97hqqbMHFDK/E1cSrjjdIX2yMTJ7m5aYo0ldvECo […]

ARC-Authentication-Results: i=3; mx.google.com;
      dkim=pass header.i=@ceitnet.it header.s=google header.b="JVoNyZ/W";
      arc=pass (i=2 spf=pass spfdomain=persiplast.com.br dkim=pass
      dkdomain=persiplast.com.br);
       spf=pass (google.com: domain of delivery.of.????@ceitnet.it designates
          209.85.220.69 as permitted sender) smtp.mailfrom=delivery.of.????@ceitnet.it
```

END-TO-END SECURITY

For a fully-trustable email system we would like the following guarantees to hold for ANY message:

- confidentiality of message content
- authentication of message sender
- message integrity
- sender non-repudiation

SECURING E-MAIL BY PGP

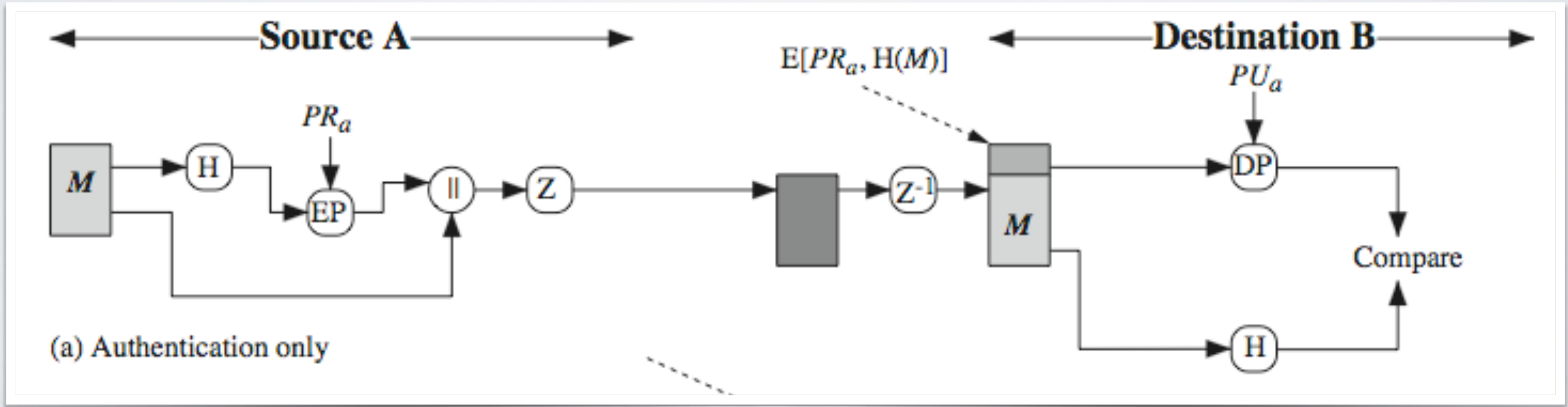
- Pretty Good Privacy is a standard created by Phil Zimmermann in 1991
  - "PGP empowers people to take their privacy into their own hands. There has been a growing social need for it. That's why I wrote it." See Why I wrote PGP https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html
- The slides on PGP are inspired to the well-known textbook Cryptography and Network Security, 5/e, by William Stallings, Chapter 18 – "Electronic Mail Security"

PRETTY GOOD PRIVACY (PGP)

- Well known and widely used since the 90s
- Using best available crypto algorithms
- Integrated into a single program
  - Linux/Unix, PC, Macintosh and other systems
- Originally free, now owned by Symantec (www.pgp.com)
- open version (OpenPGP) standardized in RCF 4880
  - several implementations, e.g., Gnu Privacy Guard (www.gnupg.org)

PGP AUTHENTICATION

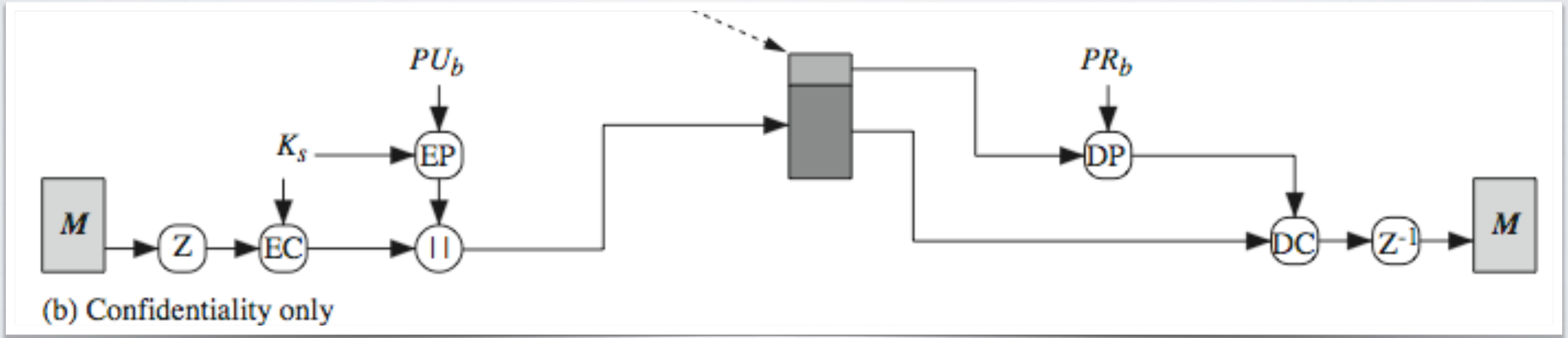
1. sender creates message
2. make SHA-1 160-bit hash of message
3. attached RSA signed hash to message
4. receiver decrypts & recovers hash code
5. receiver verifies received message hash



(a) Authentication only

PGP CONFIDENTIALITY

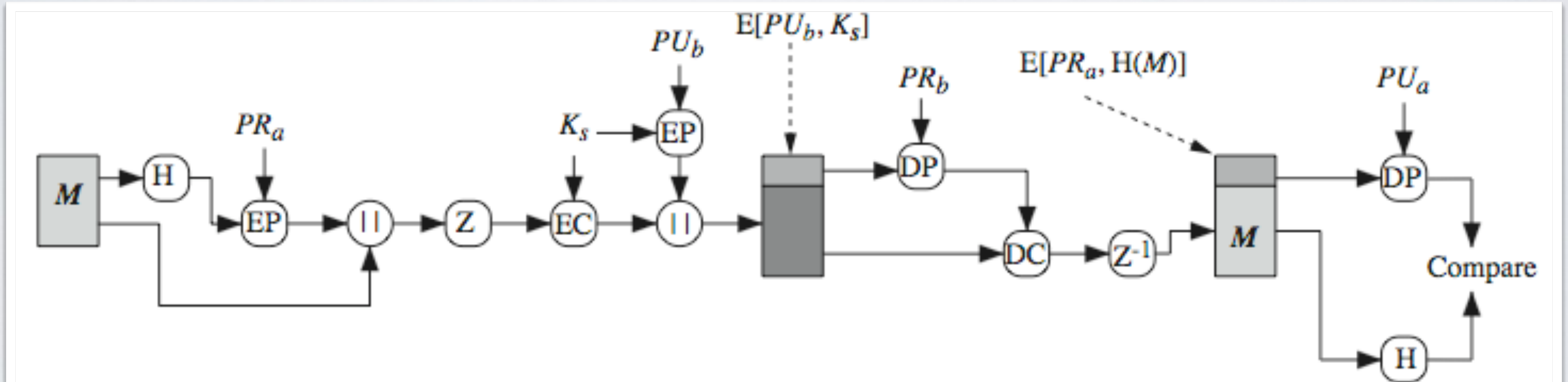
1. sender forms 128-bit random session key
2. encrypts message with session key
3. attaches session key encrypted with RSA
4. receiver decrypts & recovers session key
5. session key is used to decrypt message



(b) Confidentiality only

CONFIDENTIALITY & AUTHENTICATION

- can use both services on same message
  - create signature & attach to message
  - encrypt both message & signature
  - attach RSA/El-Gamal encrypted session key



(c) Confidentiality and authentication

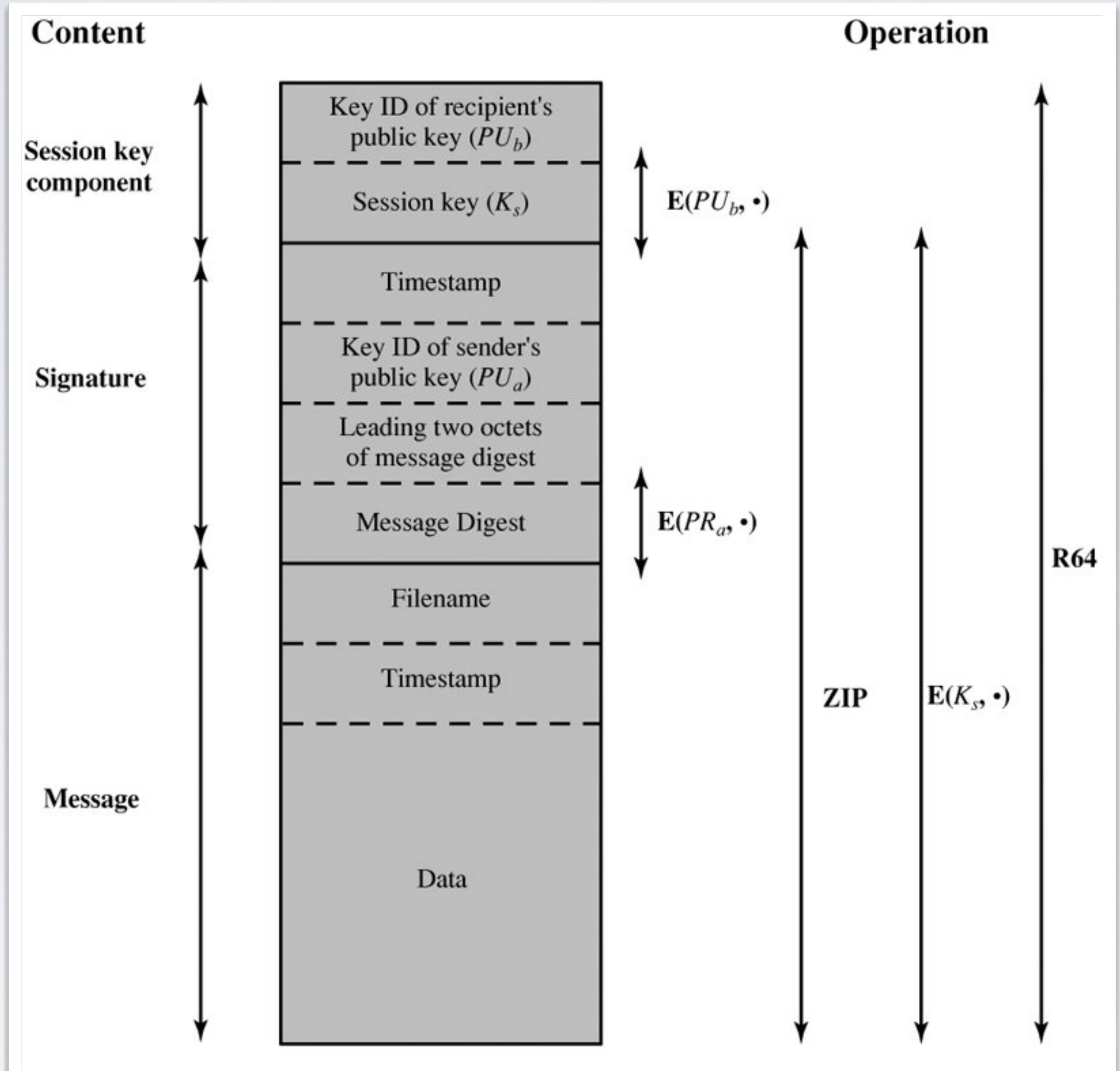
COMPRESSION

- by default PGP compresses message after signing but before encrypting
  - so can store uncompressed message & signature for later verification
  - because compression is non deterministic (if verification requires compression it may fail on legitimate messages)
- uses ZIP compression algorithm

PGP PUBLIC & PRIVATE KEYS

- since many public/private keys may be in use (by one user), need to identify which is actually used to encrypt session key in a message
  - could send full public-key with every message
  - but this is inefficient
- rather use a key identifier (ID) based on key
  - least significant 64-bits of the key
  - will very likely be unique
- also use key ID in signatures

PGP MESSAGE FORMAT



Notation:

$E(PU_b, \bullet)$	= encryption with user b's public key
$E(PR_a, \bullet)$	= encryption with user a's private key
$E(K_s, \bullet)$	= encryption with session key
ZIP	= Zip compression function
R64	= Radix-64 conversion function

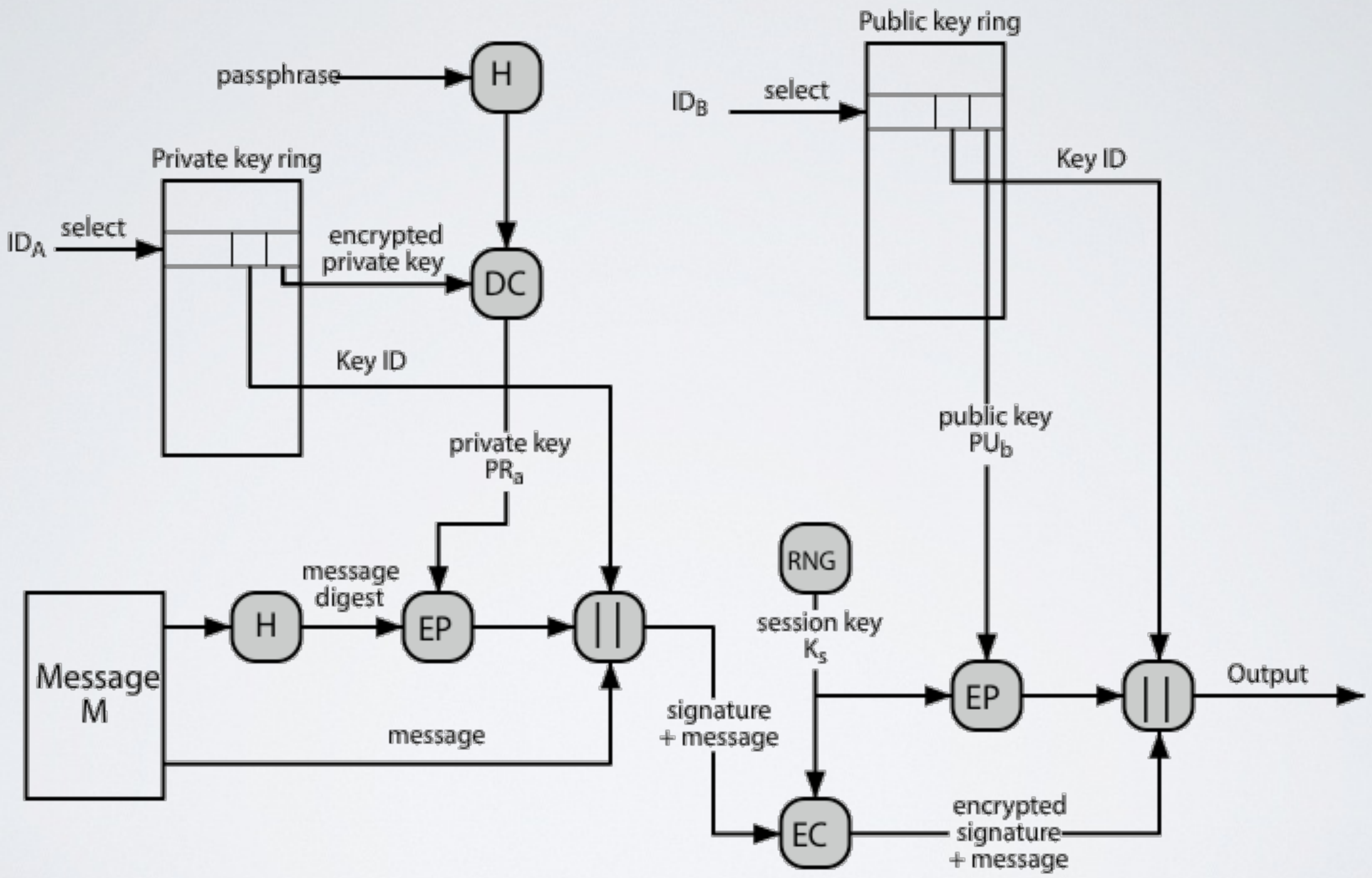
PGP KEY RINGS

Each PGP user has a pair of keyrings:

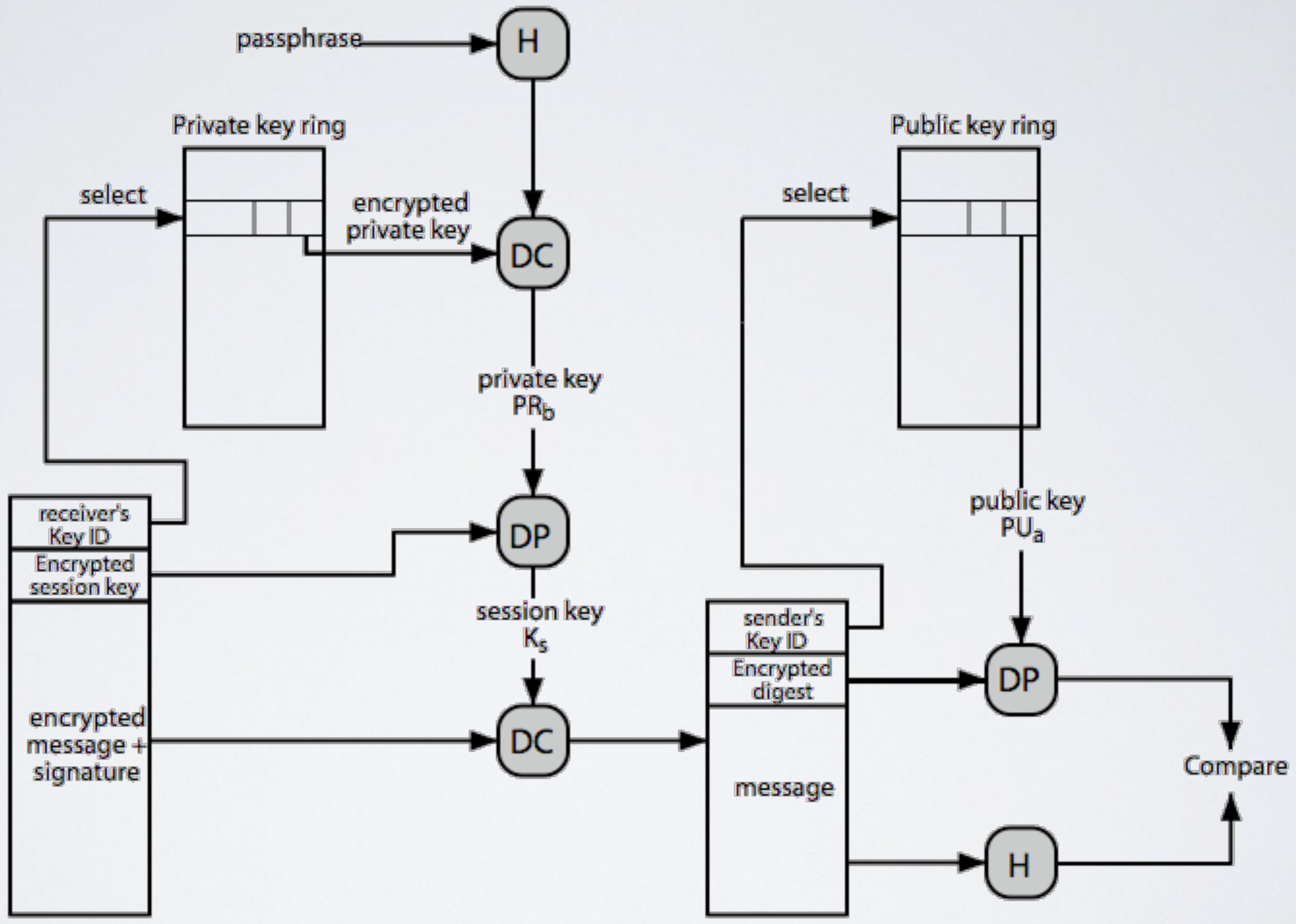
- **public-key ring** contains all the public-keys of other PGP users known to this user, indexed by key ID
- **private-key ring** contains the public/private key pair(s) for this user, indexed by key ID & encrypted keyed from a hashed passphrase

Security of private keys thus depends on the passphrase security

PGP MESSAGE GENERATION



PGP MESSAGE RECEPTION



PGP KEY MANAGEMENT

Rather than relying on certificate authorities in PGP every user is his own CA

- can sign keys for users they know directly

The idea is to create a "web of trust"

- trust keys are signed
- can trust keys others have signed if have a chain of signatures to them

Key ring includes trust indicators

- users can also revoke their keys

a possible key-sign procedure http://herrons.com/keysigning-party-guide/

WEB OF TRUST (ZIMMERMANN)

"As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys."

S/MIME

- Introduces MIME extension for e-mail security
- Provides the same guarantees as PGP
- Different model that uses a centralised PKI for key management
- Uses X.509 certificates instead of plain public/private key pairs
- Less user control
- Better interoperability