isec
INFORMATION SECURITY
lectures by Ferucio Laurentiu Tiplea

Electronic Mail Security

Prof.dr. Ferucio Laurențiu Țiplea

Fall 2023

Department of Computer Science
“Alexandru Ioan Cuza” University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Introduction to electronic mail

Electronic mail

Electronic mail, or e-mail/email/E-mail/Email, is a method of exchanging digital messages across the Internet or other computer networks.

An electronic mail system consists of:

1. Standards for defining addressing methods and message formatting;
2. A number of protocols that play different functions in implementing email messaging.

A brief history of email systems

The evolution of computer-based messaging between users can be traced back to the early 1960s. Here are a few milestones:

- Early 1960s: SNDMSG, an electronic mail program, allowed a user to compose, address, and send a message to other users' mailboxes on the same computer;
- 1971: Raymond Tomlinson modified SNDMSG so that it could send messages between different machines connected via ARPANET. He also proposed the use of the character "@" as a way to distinguish the username from the host name of the destination computer;
- Beginning 1973: ARPANET defined conventions for using the File Transfer Protocol (FTP) for electronic mail;
- Sept 1980: it was proposed to replace FTP with Mail Transfer Protocol;

A brief history of email systems

- August 1982: RFC 821 by Jon Postel, proposing the Simple Mail Transfer Protocol (SMTP);
- January 1st, 1983 (the "flag day"): TCP/IP became the standard for the ARPANET, replacing the earlier Network Control Protocol. The Internet is growing rapidly, and with it the SMTP protocol;
- Oct 1984: Post Office Protocol (POP) (RFC 918, RFC 1939 POP3);
- July 1988: Internet Message Access Protocol (IMAP) (RFC 1064 IMAP2, RFC 9051 IMAP4rev2).

The email system built around SMTP and its extensions became the most commonly used Internet mail system, although others have existed before and since SMTP became popular.

Internet mail architecture

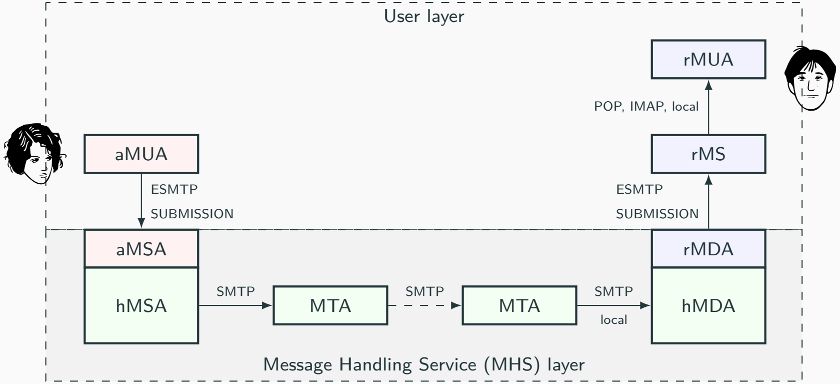
Email architectural layers

An email system usually involves three distinct architectural layers, each providing its type of store-and-forward service of data:

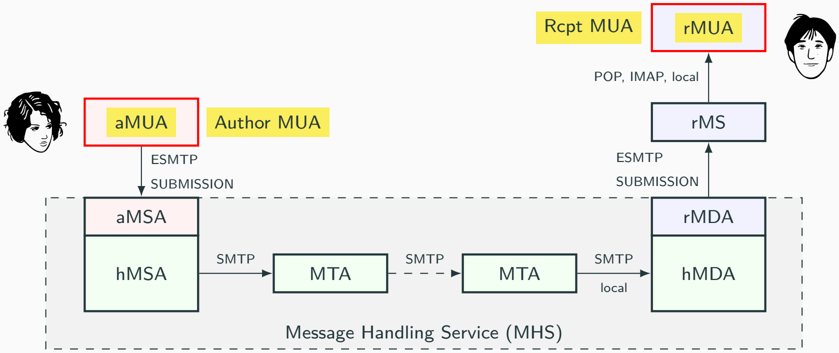
- User layer;
- Message Handling Service (MHS) layer;
- Packet Switches layer (this is the Internet's IP service).

In what follows, we will focus on services from the first two layers. These are performed by Message User Agents (MUAs) for the user level, and Message Transfer Agents (MTAs) for the MHS level. A brief discussion will follow on the following slides (for details, we indicate Crocker (2009)).

Email architectural layers



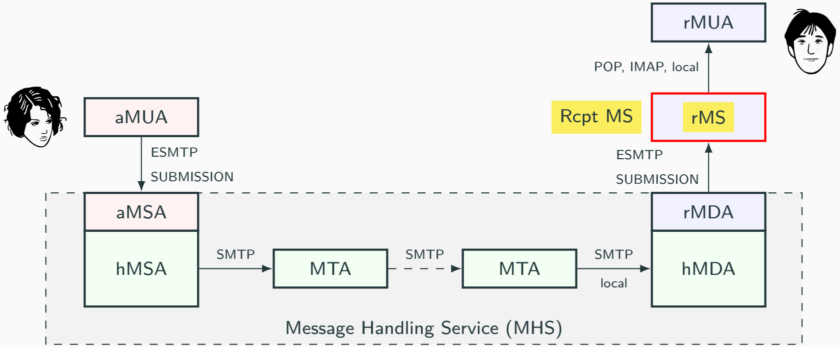
Message User Agents



Message User Agents (MUAs), otherwise known as email clients, are software applications (or web interfaces) used by individuals to send, receive, and manage their e-mails.

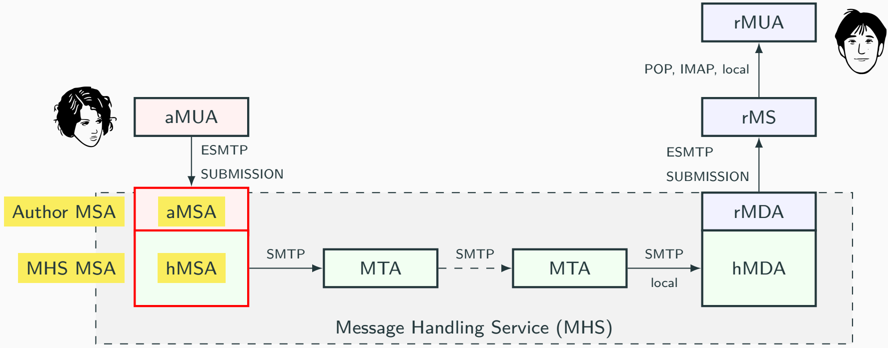
Examples: web-based interfaces like Gmail and desktop clients like Microsoft Outlook.

Message Store



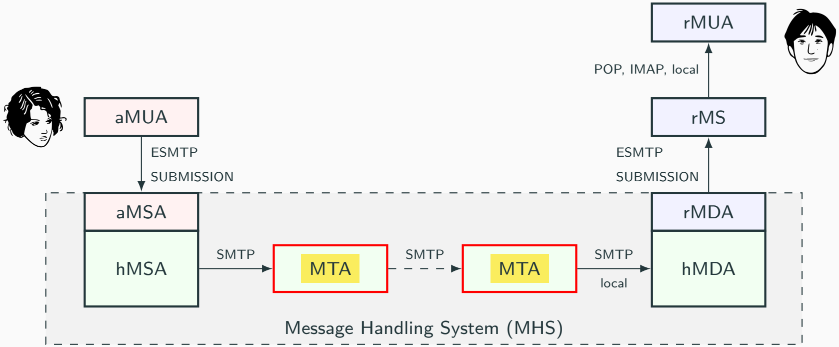
A Message Store (MS) is a system that can hold email messages for a user whose normal means of dealing with incoming email is not currently active. A message store may exist for both Authors (aMS) and recipients (rMS). An MS can be located on a remote server or on the same machine as the MUA.

Message Submission Agents



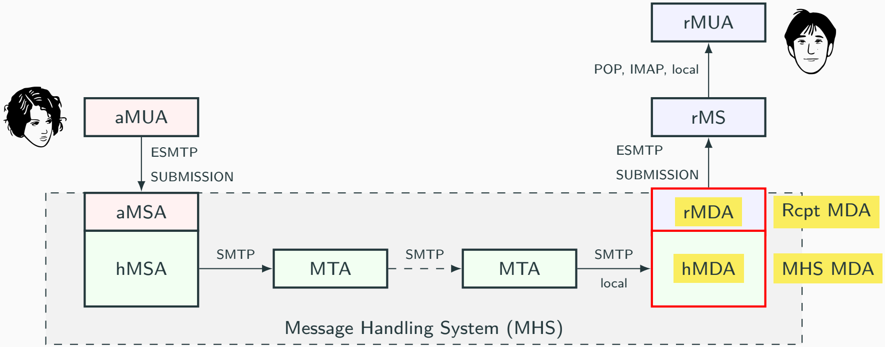
Message Submission Agents (MSAs) (i) perform final message preparation for submission and (ii) effect the transfer of responsibility to the MHS, via the hMSAs. The amount of preparation depends upon the local implementations.

Message Transfer Agents



Message Transfer Agents (MTAs) relay mail for one application-level "hop". Relaying is performed by a sequence of MTAs until the message reaches a destination MDA.

Message Delivery Agents



Message Delivery Agents (MDAs) transfer the message from the MHS to the recipient's MS. Like an MSA, an MDA serves two roles, Recipient's and MHS's. The MHS portion (hMDA) primarily functions as a server SMTP engine. The job of the Recipient portion of the MDA (rMDA) is to perform any delivery actions that the Recipient specifies.

Email system protocols

There are two types of protocols used in an email system:

- Message transfer protocols, such as:
  - Simple Mail Transfer Protocol (SMTP) (RFC 821, RFC 2821, RFC 5321). The term Extended SMTP (ESMTP) is often used to refer to the later versions of SMTP;
  - SUBMISSION protocol (RFC 6409);
  - Local Message Tramsfer Protocol (LMTP) (RFC 2033);
- Message access protocols, such as:
  - Post Office Protocol (POP) (RFC 918, RFC 1939 POP3);
  - Internet Message Access Protocol (IMAP) (RFC 1064 IMAP2, RFC 9051 IMAP4rev2).

Simple Mail Transfer Protocol

The Simple Mail Transfer Protocol (SMTP) is a protocol used for message transfer between MTAs, from aMUA to MSA, and from rMDA to rMS. Some facts about it are in order:

- It was standardized in RFC 821, the current version being RFC 5598 (Klensin (2008));
- SMTP servers both send and receive email. The device sending email act as a client, while the one receiving email acts as a server;
- All SMTP communication is done using TCP;
- SMTP servers generally must be kept running and connected to the Internet continuously;
- SMTP servers listen continuously on the SMTP server port (the well-known port number 25) for any TCP connection requests from other SMTP servers.

SMTP connection and session establishment

An SMTP server that wishes to send email does as follows:

- First, it begins with a DNS lookup of the MX record corresponding to the domain name of the intended recipient email address in order to obtain the name of the appropriate SMTP server;
- This name is then resolved to an IP address;
- The SMTP sender then establishes a SMTP session with the SMTP receiver;
- Once the session is established, email transactions can be performed: the SMTP sender provides the return address, the recipient address, and transfers the message;
- When the SMTP server is done, it terminates the connection.

Message SUBMISSION protocol

SMTP is a message transfer protocol. It is also widely used to submit from an aMUA to an MSA, although it is not very suitable for such a job:

1. Authentication and authorization of initial submission have become increasingly important, but SMTP usually does not support them;
2. Messages being submitted are, in some cases, finished (complete) messages and, in other cases, are unfinished (incomplete).

Separating message submission (aMUA –> MSA) from message transfer (MTA –> MTA) allows developers and network administrators to add enhanced submission services.

The protocol currently prefered for initial submission is SUBMISSION (Klensin and Gellens (2011)), which derives from SMTP, uses a distinct TCP port, and imposes several requirements, such as access authorization.

Local message transfer protocol

SMTP and its service extensions (ESMTP) provide a mechanism for transferring mail reliably and efficiently. The design of SMTP requires the server to manage a mail delivery queue.

In some limited circumstances, it is desirable to implement a system where a mail receiver does not manage a queue.

The Local Message Transfer Protocol (LMTP) (Myers (1996)) is a protocol derived from SMTP (it uses with a few changes the syntax and semantics of ESMTP) that does not require the server to manage a mail delivery queue. LMTP should be used only by specific prior arrangement and configuration, and it must not be used on TCP port 25.

Models for email access and retrieval

1. On-line access model. In this model, every machine is always connected to the Internet running an SMTP server. Therefore, users have constant, direct on-line access to their mailboxes;
2. Off-line access model. In this model, a user establishes a connection to a server where his mailbox is located, downloads the received messages, and then deletes them from the server mailbox. The received messages are manipulated off-line;
3. Disconnected access model. This is a hybrid model: the messages are downloaded (but not deleted) from the server, and are manipulated off-line. When the user connects back with the server, all changes made on the local device are synchronized with the mailbox on the server.

Protocols and methods for email access and retrieval

1. Post Office Protocol (POP): Implements the off-line access model;
2. Internet Message Access Protocol (IMAP): Can implement all three of the access models, although it is primarily used in the on-line and disconnected models;
3. Direct server email access: Based on establishing direct access to the server where the mailbox (which is just a file) is located. So, it can be accessed in several ways, such as: by the SMTP server directly, file sharing access (using a protocol such as NFS), dial-up remote server access, telnet remote server access;
4. Web-based email access. This technique exploits the flexibility of HTTP to informally "tunnel" email from a mailbox server to the client (a Web browser).

Internet message format

The structure of email messages is established by the following RFCs:

- Internet Mail Format (IMF) (RFC 733, RFC 822, RFC 2822, RFC5322);
- Multipurpose Internet Mail Extensions (MIME) (RFC 2045 – 2049).

Email structure

An email has the following general structure:

- Envelope. The envelope encapsulates the message and contains all the information needed for transporting the message by message transport agents, such as: the destination address, priority, and security level;
- Message, which has two parts:
  - Header, which contains control information for the user agents;
  - Body, which is the message itself.

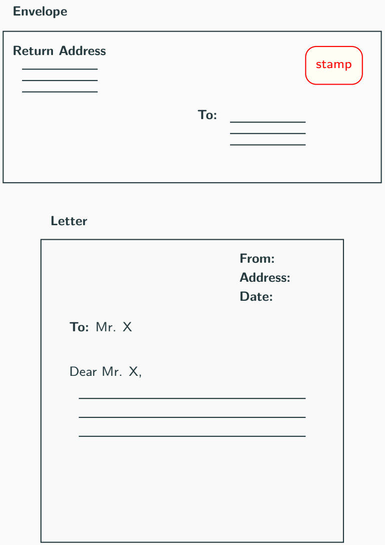
The distinction between envelope and email message is important from a technical point of view. The envelope is not the same the email message header!

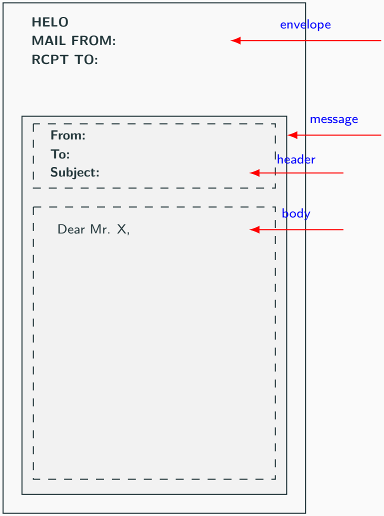
However, email software can process and interpret the message to construct the necessary envelope for transport agents to transport the message.

Email structure

Physical mail

Electronic mail





RFC 822 message header format

A header line has the following structure:

$$\begin{array}{lcl} \langle\text{header name}\rangle & : & \langle\text{header value part 1}\rangle \\ \langle\text{white space}\rangle & & \cdots \\ \langle\text{white space}\rangle & & \langle\text{header value part k}\rangle \end{array}$$

where ⟨white space⟩ means at least one space or tab character.

The most frequently used header names are From, To, Subject, and Date.

A small number of header lines are mandatory. Some are not mandatory but usually present, and others are automatically included when needed.

RFC 822 allows users to invent new header names, provided that these header names start with the string "X-". For instance,

X-Weather-Forecast : Light rain

RFC 822 message body format

An RFC 822 message body is separated from the header by an empty line.

The body consists of several lines of text subjected to the following technical constraints due to the technology of the years when RFC 822 was created:

- Consists of 7-bit ASCII characters only;
- Each line ends with a CR character (13, in decimal), followed by an LF character (10, in decimal) (this combination is called “CRLF”);
- Each line should be (recommendation) 78 characters or less (not including CRLF), and must not be more than 998 characters (not including CRLF);
- The characters CR and LF must not appear by themselves within the text line.

RFC 822 email format limitations

By the late 1980s, it was quite clear that the RFC 822 email format must be extended in order to support:

- Messages in languages with accents (e.g., French), in non-Latin alphabets (e.g., Russian), or in languages without alphabets (e.g., Chinese);
- Non-text information (e.g., graphic files, multimedia);
- Arbitrary binary data, including executable programs (binary data = arbitrary sequences of octets).

These are mainly dictated by the fact that SMTP cannot transmit texts with special characters, executable files, or other types of binary objects.

Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extensions (MIME), proposed in RFC 1341 and RFC 1342 (current version: RFC 2045 – 2049), came as a solution to the limitations of the RFC 822 email format.

MIME's basic idea is to:

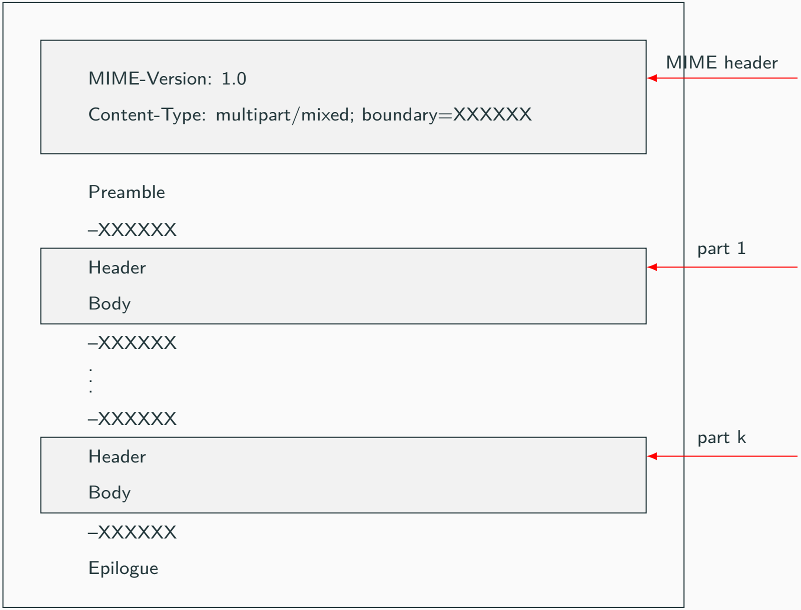
- Add structure to the message body. This means to consider the MIME body as composed of several parts, each part with a specific but unitary identity.;
- Encode non-ASCII data as ASCII text.

MIME header

In order to provide information about the structure of a MIME message, new message header names have been introduced:

- MIME-version: Identifies the MIME version;
- Content-Description: Tells what is in the message (it is optional);
- Content-ID: Unique identifier;
- Content-Type: Describes the nature of the data that is encoded in the MIME entity (e.g., “text/html” says that the message contains an html document);
- Content-Transfer-Encoding: Specifies the encoding method for each part of the body.

MIME multipart message structure



MIME: Content-Type values

Content-Type values as defined in RFC 2046 (other values have also been introduced. See, for instance, RFC 1847, RFC 2387, RFC 2388, RFC 2616, RFC 6522).

	type	sub-type	notes
handled by non-MIME mechanisms	text	plain	plain (unformatted) text
	image	jpeg	image in jpeg format
		gif	image in gif format
	audio	basic	basic audio
	video	mpeg	mpeg standard video
	application	postscript	ps file
		octet-stream	arbitrary binary data
handled using MIME mechanisms	message	rfc822	message according RFC 822
		partial	a fragment of a larger entity
		external body	actual body data not included
	multipart	mixed	independent parts in a specified order
		digest	each part is a complete RFC 822 message
		alternative	same message in different formats
		parallel	parts must be viewed simultaneously

MIME: Content-Transfer-Encoding

MIME encoding methods of the body (RFC 2045):

- 7bit: This indicates 7-bit ASCII format (RFC 822 standard);
- 8bit: Encoding using 8-bit characters (8bit data is defined as 7bit data is but with the difference that octets with decimal values greater than 127 may be used);
- binary: any sequence of octets;
- quoted-printable: This is used when most of the data is ASCII text and only very few characters are non-ASCII;
- base64 (also called Radix-64 or ASCII armor): Encodes data by mapping 6-bit blocks of input to 8-bit blocks of outputs, all of which are printable ASCII characters.

Email security threats

Email security threats

Security threats to the canonical functions of an email service can be classified as follows:

1. Integrity-related threats to the email system – could result in unauthorized access to an enterprises' email system, or spoofed email used to initiate an attack;
2. Confidentiality-related threats to email – could result in unauthorized disclosure of sensitive information;
3. Availability-related threats to the email system – could prevent end users from being able to send or receive email.

Integrity-related threats to the email system

Possible sources of this type of security threat:

1. Unauthorized email senders within an organization's IP address block;
2. Unauthorized email receivers within an organization's IP address block;
3. Unauthorized email messages from a valid DNS domain;
4. Tampering/Modification of email content from a valid DNS domain;
5. DNS Cache Poisoning;
6. Phishing and spear phishing.

Confidentiality-related threats to email

Possible sources of this type of security threat:

1. (Unauthorized) access to the packets that make up the email message as they move over a network, in the form of a passive wiretapping or eavesdropping attack;
2. Software may be installed on a MTA that makes copies of email messages and delivers them to the adversary. For example, the adversary may have modified the target's email account so that a copy of every received message is forwarded to an email address outside the organization;
3. The valid receiver's mail servers may be spoofed (or DNS MX reply spoofed), to get the sender to connect to a MTA controlled by an attacker.

Availability-related threats to the email system

Possible sources of this type of security threat:

1. Email bombing;
2. Unsolicited Bulk Email (UBE), also called “Spam”;
3. Availability of email servers.

Security threats mitigation

Pretty Good Privacy

Pretty Good Privacy (PGP) is a standard that provides authentication and confidentiality for data communication. Brief history:

- 1991: PGP was created by Philip Zimmermann, and distributed across the Internet together with its complete source code for free non-comercial use (Zimmermann (1999));
- Feb 1993 – 1996: Zimmermann became the formal target of a criminal investigation by the US Government. The investigation ended without filing criminal charges against Zimmermann;
- Beginning 1997: OpenPGP Working Group was formed in the IETF (OpenPGP_WG (2023));
- Nov 1998: OpenPGP standard (see OpenPGP_WG (2023)).

Check OpenPGP_WG (2023) web-site for a collection of OpenPGP related RFCs and Internet drafts!

PGP services

1. PGP authentication (applied to x): $(x, sig(x))$, where $sig(x)$ is sender's signature on x.
2. PGP confidentiality (applied to x): $(\{K\}_{K^e}, \{x\}_K)$, where K is a fresh symmetric encryption key and K^e is receiver's public key.
3. PGP authentication and confidentiality are obtained by applying confidentiality to $x' = (x, sig(x))$.
4. Compression. PGP recommends compressing the message after applying the signature but before encryption.
5. Radix-64 conversions. PGP provides Radix-64 encoding for transferring data.

Please see Finney et al. (2007) for more details regarding PGP services.

PGP key management

PGP key management includes:

- Session key generation;
- Key storage (a user may have multiple asymmetric keys that need to be stored, as well as a list of public keys of other users):
    - PGP associates identifiers to public keys. Then, asymmetric keys are stored as table-like structures, called key rings, that may be indexed by user ID or key ID;
    - A PGP key identifier (key ID) associated to a public-key K^e consists of its least significant 64 bits;
    - PGP stores two key rings as two files on the hard disk of the host that runs PGP; one for private keys, and one for public keys;
- Public key certification.

PGP key rings

A private-key ring at a host that runs PGP looks like this:

Timestamp	Key ID	Public Key	Enc Private Key	User ID
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
t	$K_U^e \ mod\ 2^{64}$	K_U^e	$\{K_U^d\}_{H(P_U)}$	$ID(U)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

P_U is a password of U and $H(P_U)$ is a key derived by H from P_U. This key ring is indexed by key ID and user ID.

A public-key ring at a host that runs PGP contains the fields "Timestamp", "Key ID", "Public Key", and "User ID" as in a private-key ring, together with four new fields, "Owner Trust", "Key Legitimacy", "Signature(s)", and "Signature(s) Trust".

The new fields have trust flag values indicating to what extent a public key is trusted as a valid public key of the corresponding user.

Using the key rings to sign and encrypt

Using PGP key rings:

- User U signing a message:
  - PGP retrieves U's encrypted private key from the private-key ring using $ID(U)$;
  - PGP prompts U for the password to recover U's private key;
  - The signature is constructed;
- User U encrypting a message:
  - PGP generates a session key and encrypts the message;
  - PGP retrieves the recipient's public key from the public-key ring using his/her ID;
  - The session key is encrypted by the recipient's public key.

Using the key rings to decrypt and verify signatures

Using PGP key rings:

- User U verifying a signature:
    - PGP retrieves the sender's public key from the public-key ring using the key ID (which is attached to the message he received);
    - PGP verifies the signature;
- User U decrypting a message:
    - PGP retrieves U's encrypted private key from the public-key ring using his/her ID;
    - PGP prompts U for the password to recover U's private key;
    - PGP decrypts the message.

Secure/Multipurpose Internet Mail Extensions

Secure/Multipurpose Internet Mail Extensions (S/MIME) is a standard to send and receive secure MIME data:

- S/MIME had been proposed by RSA Data Security Inc. in 1995. The current standard is S/MIME v4.0 (Schaad et al. (2019));
- S/MIME provides the same functions as OpenPGP, authentication (through digital signatures), confidentiality, compression, and Radix-64 conversion for data transfer. Due to using digital signatures, message integrity and non-repudiation of origin are also obtained;
- S/MIME can be used by traditional MUA to add/interpret cryptographic security services.

Secure/Multipurpose Internet Mail Extensions

S/MIME can be used with any transport mechanism that transports MIME data. To this, S/MIME makes use of several new MIME content types:

	type	sub-type	s-mime type (optional)	extension
handled by non-MIME mechanisms	application	pkcs7-mime	enveloped-data	.p7m
			signed-data	.p7m
			authEnveloped-data	.p7m
			certs-only	.p7c
			compressed-data	.p7z
		pkcs7-signed		.p7s

Data processing is done similarly to OpenPGP. For details, refer to Schaad et al. (2019).

S/MIME examples

Example 1:

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
   name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

```
MIIBHgYJKoZIhvcNAQcDoIIBDzCCAQsCAQAxgcAwgb0CAQAwJjASMRAwDgYDVQQDEw
dDYXJsUlNBAhBGNGvHgABWvBHTbi7NXXHQMA0GCSqGSIb3DQEBAQUABIGAC3EN5nGI
iJi2lsGPcP2iJ97a4e8kbKQz36zg6Z2i0yx6zYC4mZ7mX7FBs3IWg+f6KgCLx3M1eC
bWx8+MDFbbpXadCDgO8/nUkUNYeNxJtuzubGgzoyEd8Ch4H/dd9gdzTd+taTEgS0ip
dSJuNnkVY4/M652jKKHRLFf02hosdR8wQwYJKoZIhvcNAQcBMBQGCCqGSIb3DQMHBA
gtaMXpRwZRNYAgDsiSf8Z9P43LrY4OxUk660cu1lXeCSFOSOpOJ7FuVyU=
```

Example 2:

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
   name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

```
MIIDmQYJKoZIhvcNAQcCoIIDijCCA4YCAQExCTAHBgUrDgMCGjAtBgkqhkiG9w0BBw
GgIAQeDQpUaGlzIGlzIHNvbWUgc2FtcGxlIGNvbnRlbnQuoIIC4DCCAtwwggKboAMC
AQICAgDIMAkGByqGSM44BAMwEjEQMA4GA1UEAxMHQ2FybERTUzAeFw05OTA4MTcwMT
```

OpenPGP and S/MIME in practice

1. OpenPGP and S/MIME are two of the most widely used email security standards;
2. S/MIME is commonly used in corporate and government environments. It benefits from its ability to integrate into PKIs and that most widely-used email clients support it by default;
3. OpenPGP often requires the installation of additional software;
4. They are not compatible with each other;
5. The integration of S/MIME or OpenPGP for e-mail security depends on circumstances and needs, which makes one preferable to the other (depending on the situation);
6. In recent years, attacks have been reported that work on both standards, which led to their analysis and improvement. Wide attention was given to OpenPGP, which currently benefits from modern techniques and updated encryption algorithms.

OpenPGP and S/MIME in practice

Both S/MIME and OpenPGP are protocols that facilitate signing and encryption. So, care should be taken to the open distribution of public keys!

Two recent DNS-Based Authentication of Named Entities (DANE; RFC 6698) protocols have been proposed to approach this problem:

1. SMIMEA (RFC 8162);
2. OPENPGPKEY (RFC 7929).

Both of them specify new DNS RR types for storing end user key material in the DNS. Email clients and encryption software can query the DNS records associated with a domain to get the necessary public key information for encryption and verification.

“Envelope FROM” vs “Header FROM”

Two email addresses are used when sending an email via SMTP:

- **MAIL FROM** or **envelope-from** or **return-path** address (similar to the “from” address on the physical envelope), used by the SMTP server to generate a “non-delivery report” (NDR);
- **Header FROM** or **sender** address (similar to the “from” address on the letter inside the envelope), used by the email client to display information in the “From” field of the header message.

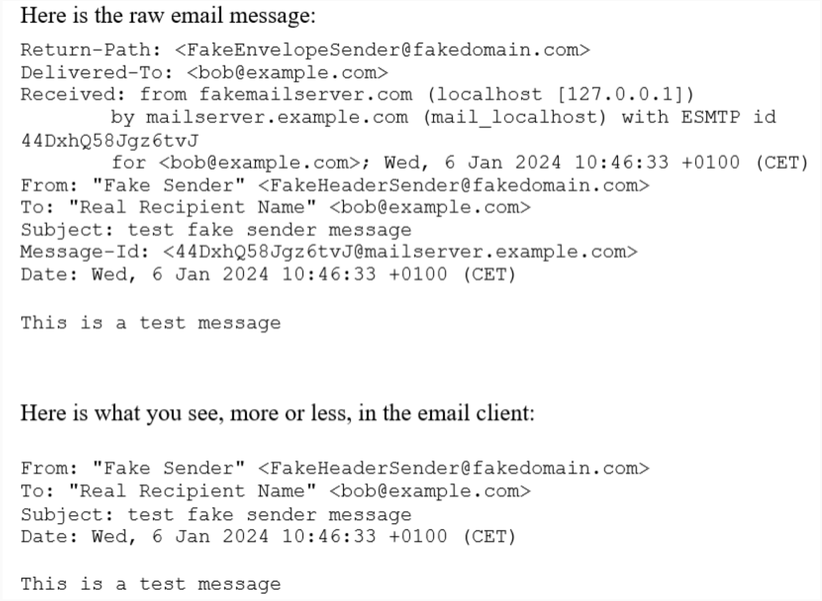
Important facts:

1. The sender email address (header FROM) can be different from the envelope MAIL FROM;
2. The MAIL FROM address can be forged by a sending MTA;
3. The Header FROM address can be forged.

Example of sender address forgery

	Client connects to port 25	
S:	220 mailserver.example.com	mailserver.example.com is ready
C:	HELO Fakemailserver.com	email envelope (blue lines) with a fake envelope sender and a real recipient
S:	250 OK	
C:	MAIL FROM: <FakeEnvelopeSender@fakedomain.com>	
S:	250 Ok	
C:	RCPT TO: <bob@example.com>	
S:	250 Ok	
C:	DATA	Client asks for permission to transfer the mail data
S:	354 End data with <CR><LF>.<CR><LF>	Permission granted
C:	From: ”Fake Sender” <FakeHeaderSender@fakedomain.com>	email header with a fake sender and real recipient
C:	To: ”Real Recipient Name” <bob@example.com>	
C:	Subject: test fake sender message	
C:		email body (the first line must be empty, and the last must contain a dot)
C:	This is a test message	
C:	.	
S:	250 Ok: queued as 12345	email received by server
C:	QUIT	Request to terminate
S:	221 Bye	
	Server closes the connection	

Example of sender address forgery


Here is the raw email message:

```
Return-Path: <FakeEnvelopeSender@fakedomain.com>
Delivered-To: <bob@example.com>
Received: from fakemailserver.com (localhost [127.0.0.1])
        by mailserver.example.com (mail_localhost) with ESMTP id
44DxhQ58Jgz6tvJ
        for <bob@example.com>; Wed, 6 Jan 2024 10:46:33 +0100 (CET)
From: "Fake Sender" <FakeHeaderSender@fakedomain.com>
To: "Real Recipient Name" <bob@example.com>
Subject: test fake sender message
Message-Id: <44DxhQ58Jgz6tvJ@mailserver.example.com>
Date: Wed, 6 Jan 2024 10:46:33 +0100 (CET)

This is a test message
```

Here is what you see, more or less, in the email client:

```
From: "Fake Sender" <FakeHeaderSender@fakedomain.com>
To: "Real Recipient Name" <bob@example.com>
Subject: test fake sender message
Date: Wed, 6 Jan 2024 10:46:33 +0100 (CET)

This is a test message
```

Conclusions

1. The **envelope sender address**, also called the **return-path**, is used by SMTP to return the message to sender in the case of a delivery failure.
   Mail clients do not usually display it to users;
2. The **header sender address** of an email message is what is displayed to the user by mail clients.
   Generally, mail servers do not care about the header sender address when delivering a message;
3. Both envelope and header sender addresses can be faked.

Can the two addresses be authenticated?

Sender Policy Framework (SPF)

Sender Policy Framework (SPF) is an email authentication protocol that allows you to specify the servers authorized to send mail from your domain (RFC 4408, RFC 7208).

How SPF works:

1. The domain owner includes a TXT resource record (also called a SPF record) in the DNS with information about the servers authorized to send email for the domain;
2. When the email arrives, the receiving mail server checks to see if the IP address from which the mail was sent is listed in your SPF record. If it is, the message passes SPF; otherwise, it does not.

The receiving mail server looks up the SPF record for the domain in the envelope-FROM field, which may not be the same as the domain in the "From" field! So, header sender address may be spoofed!

DomainKeys Identified Mail (DKIM)

DomainKeys Identified Mail (DKIM) is an email authentication protocol that allows you to digitally sign mail messages (header and body).

How DKIM works:

1. The sending MTA generates a signature over the message (header, body, or portions of them) and includes it in the message header, along with information for signature validation;
2. The public key for signature valiation is stored as an TXT RR in DNS (also called a DKIM record);
3. When the receiving MTA gets the message, it checks the signature (issuing first a DNS query for the verification key).

The signature is no longer verified if the signed email components change in transit!

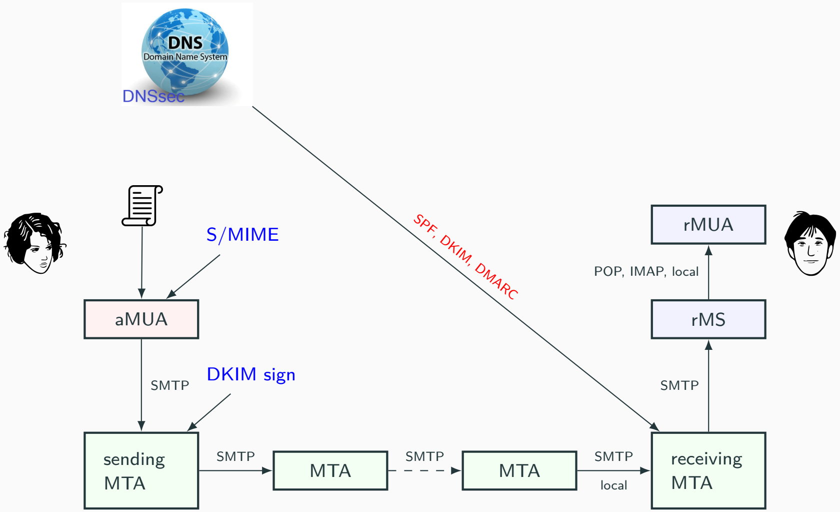
Domain-based Message Authentication, Reporting, and Conformance (DMARC)

Domain-based Message Authentication, Reporting, and Conformance (DMARC) is an email authentication protocol that works in conjunction with SPF and DKIM to verify that the sender of the email is genuine.

Here is how the three protocols work:

1. SPF verifies that the email has been sent from authorized email server;
2. DKIM cryptographically verifies that the sender address and the message body have not been changed in transit;
3. DMARC ensures that the domain in DKIM and SPF checks matches the sender's domain in the message header (the "From" field). It also specifies how email servers should handle a message that fails both DKIM and SPF (accept, reject, or spam).

DNSsec, S/MIME, SPF, DKIM, and DMARC



STARTTLS

A few facts about SMTP:

1. SMTP is not an end-to-end protocol but instead a protocol that sends mail messages through a series of hops (MUA, MSA, MTAs);
2. There is no way to signal that message submission must be secure;
3. There is no way to signal that any hop in the transmission should be secure.

STARTTLS (RFC 3207) is an extension of SMTP that allows an SMTP client and server to use TLS to provide private, authenticated communication across the Internet.

STARTTLS

How it works:

- If the client initiates the connection over a TLS-enabled port, the server advertises that the STARTTLS option is available to connecting clients;
- The client can then issue the STARTTLS command in the SMTP command stream;
- The two parties proceed to establish a secure TLS connection.

Similar mechanisms are available for running TLS over IMAP and POP.

References

Crocker, D. (2009). Internet Mail Architecture. RFC 5598.

Finney, H., Donnerhacke, L., Callas, J., Thayer, R. L., and Shaw, D. (2007). OpenPGP Message Format. RFC 4880.

Klensin, D. J. C. (2008). Simple Mail Transfer Protocol. RFC 5321.

Klensin, D. J. C. and Gellens, R. (2011). Message Submission for Mail. RFC 6409.

Myers, J. G. (1996). Local Mail Transfer Protocol. RFC 2033.

OpenPGP_WG (2023). Openpgp. `https://www.openpgp.org/`.

Schaad, J., Ramsdell, B. C., and Turner, S. (2019). Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. RFC 8551.

Zimmermann, P. (1999). Why I wrote PGP. `https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html`.