Email Security

Ferucio Laurențiu Țiplea

Department of Computer Science
"Al.I.Cuza" University of Iași
Iași 700506, Romania
e-mail: fltiplea@info.uaic.ro

January 10, 2017

Electronic mail

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

An email system is a system which consists of:

- standards for defining addressing methods and message formatting, and
- a number of protocols that play different functions in implementing email messaging.

Email systems

- File transfer protocol based email systems.
  The first email systems simply consisted of file transfer protocols. The first line of a message (file) in such systems should contained the recipient's address.
  These systems have many inconveniences:
  - sending a message to a group of recipients is inconvenient;
  - messages do not have an internal structure;
  - the sender never knows if a message has been received or not;
  - messages cannot be handled by another party (a secretary, for instance) if the recipient is away for some time;
  - impossibility of creating messages containing text, drawings, facsimile, and voice.

Email systems

- ARPANET email system (RFC 822-based email system, Internet email system).
  In 1982, ARPANET has proposed an email system published in RFC 821 (transmission protocol) and RFC 822 (message format) (revisions in RFC 2821 and RFC 2822);
- X.400 email system.
  In 1984, CCITT (Comité Consultatif International Téléphonique et Télégraphique) - now known as ITU-T, has proposed a suite of recommendations (X.400) that define standards for Data Communication Networks for Message Handling Systems (MHS).

X.400 was poorly designed and very complex that nobody could implement it well.

Email system functions

As time went on, the RFC 822-based email system became widely used. From now on we will refer to it as the Internet email system.

Typically, an email system should support five basic functions:

- composition
- transfer
- reporting
- displaying
- disposition

Advanced features: create and manage mailboxes, mailing lists, create (blind) carbon copies, set priorities, encrypt and sign messages and so on.

Email system components

An email system has two main components (sub-systems):

- user agents. An user agent is a local program (sometimes called a mail reader) that is responsible for composing, receiving, and replying to messages, as well as for manipulating mailboxes;
- message transfer agents. A message transfer agent is typically a system daemon, i.e., a computer program that runs in the background, rather than under the direct control of a user. Daemons are usually initiated as background processes at boot time.
  The role of a message transfer agent is to transfer messages from originator to recipient.

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.

RFC 822 Based Email System

The main TCP/IP email standard was defined in:

- RFC 821, which defines the Simple Mail Transport Protocol (SMTP) as the primary transport agent for delivering email;
- RFC 822, which defines the email message format.

RFC 822 does not clearly distinguish the envelope fields from the header fields.

RFC 822 message format

An RFC 822 message consists of several lines of text called header lines (which define the header), followed by an empty line which marks the end of the header, and then followed by another several lines of text which define the message body.

A line of text is subjected to the following constraints:

- consists of 7-bit ASCII characters only;
- each line ends with a CR character, followed by an LF character (this combination is called "CRLF");
- each line should be 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 message format

A header line has the following structure:

$$\langle \text{header name} \rangle : \langle \text{header value} \rangle$$

The header value depends on the header name. If it exceeds the maximum length of a line, then it can be "folded" onto multiple lines:

$$\begin{aligned} &\langle \text{header name} \rangle : \langle \text{header value part 1} \rangle \\ &\langle \text{white space} \rangle \quad \langle \text{header value part 2} \rangle \\ &\langle \text{white space} \rangle \quad \cdots \\ &\langle \text{white space} \rangle \quad \langle \text{header value part k} \rangle \end{aligned}$$

where $\langle \text{white space} \rangle$ means at least an one space or tab character.

RFC 822 message format

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

is a valid header line.

RFC 822 message format

Remarks on line of text limitations in RFC 822 messages:

- the 998 character limit is due to limitations in many implementations which send, receive, or store Internet Message Format messages that simply cannot handle more than 998 characters on a line;
- the more conservative 78 character recommendation is to accommodate the many implementations of user interfaces that display these messages which may truncate, or disastrously wrap, the display of more than 78 characters per line;

RFC 822 message format

Remarks on line of text limitations in RFC 822 messages:

- the limitation on 7-bit ASCII characters is due to the fact that, when the RFC 822 was defined, the networks were small, the number of users was small, the computing capabilities of network hosts were small, the capacity of network connections was small and so on. Moreover, most computer inputs and outputs were text-based.
  As a result of this, the requirements for email messages were to limit them to text messages.

RFC 822 message processing sequence

Main events in the lifetime of an email message:

- composition. The email originator writes the message body and also tells the email client program the values to use for certain important header fields;
- sender client processing. The email client processes the message and creates the envelope for transmission of the message by SMTP. It may also insert certain header fields to the message;
- SMTP server processing. SMTP servers may add new headers, especially trace headers such as *Received* and *Return-path*, as they transport the message;

RFC 822 message processing sequence

Main events in the lifetime of an email message:

- recipient client processing. When the message arrives at its destination, the recipient client may add new header fields to indicate the date and time when the message was received;
- recipient access. Display the message in a way meaningful to the user (and allow message manipulation).

Multipurpose Internet Mail Extensions

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 files (including executable programs).

A solution was proposed in RFC 1341 and RFC 1342 (June 1992), called Multipurpose Internet Mail Extensions (MIME).

Multipurpose Internet Mail Extensions

The basic idea to MIME is to:

- add structure to the message body;
- encode non-ASCII data as ASCII text.

A structured body allows multiple different media to be contained in a single email message.

In order to be able to structure a message, information about the type of each media should be provided. Therefore, MIME defines five new message header names.

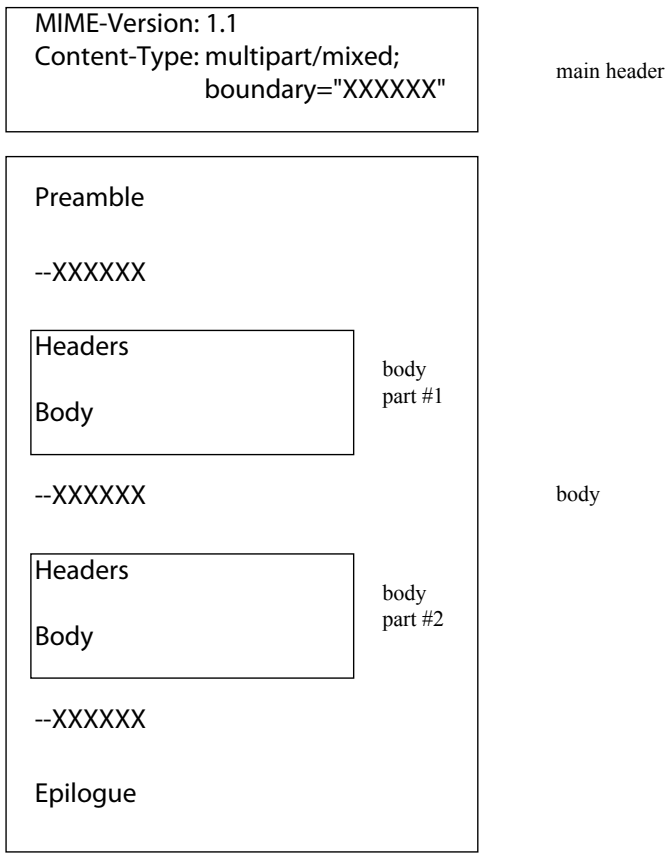
Multipurpose Internet Mail Extensions

New MIME header names:

- 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.

Multipurpose Internet Mail Extensions

General view of a MIME multipart message structure:



Multipurpose Internet Mail Extensions

A body part of the main MIME body may have a type like this:

- text/enriched. Text that may include formatting information;
- image/jpeg. An image in JPEG format;
- audio/mpeg. MPEG standard audio;
- video/dv. Digital video;
- model/vrml. A Virtual Reality Modeling Language model;
- application/pdf. A PDF file;
- message/rfc822. Indicates that the body of this body part contains an encapsulated email (which may be a MIME message) formatted according to RFC 822 standard.

Multipurpose Internet Mail Extensions

A MIME multipart message may have a type like this:

- multipart/mixed. Independent parts in the specified order;
- multipart/alternative. Same message in different formats;
- multipart/parallel. Parts must be viewed simultaneously;
- multipart/digest. Each part is a complete RFC 822 message.

Multipurpose Internet Mail Extensions

MIME encoding methods of the body:

- 7bit. This indicates 7-bit ASCII format (RFC 822 standard);
- 8bit/binary. Encoding using 8-bit characters;
- quoted-printable. This is used when most of the data is ASCII text and only very few characters are non-ASCII;
- base64. Encodes data by mapping 6-bit blocks of input to 8-bit blocks of outputs, all of which are printable ASCII characters.

base64 encoding is also called Radix-64 or ASCII armor.

RFC 2046 defines two other encodings, ietf-token and x-token, to allow new encoding types to be defined in the future.

Multipurpose Internet Mail Extensions

Quoted-printable example:

- Studenții de la Iași învață securitatea informației
- Studen=C5=A3ii de la Ia=C5=9Fi =C8=8Bnva=C5=A3=C7=8E securitatea informa=C5==A3iei

Radix-64 example:

- Studenții de la Iași învață securitatea informației
- U3R1ZGVuY2lpIGRlIGxhIElhX2kgC252YWPO IHNlY3VyaXRhdGVhIGluZm9ybWFjaWVp

Multipurpose Internet Mail Extensions

MIME encoding methods of the header:

- the above encoding methods deals with the encoding of ASCII or non-ASCII data into the body of an RFC 822 message;
- a MIME header which contains non-ASCII characters is encoded as follows:

  $$=?\,\langle\text{charset}\rangle\,?\;\langle\text{encoding}\rangle\,?\;\langle\text{encoded-text}\rangle\,?=$$

  where:
  - "charset" stands for the character set used (e.g., GB2312 stands for Chinese);
  - "encoding" may be "B" (base64) or "Q" (quoted-printable);
  - "encoded-text" is the encoded text by the encoding type indicated.

Simple Mail Transfer Protocol

The Simple Mail Transfer Protocol (SMTP) is a protocol used for the delivery of email between TCP/IP systems and users (but not for the final retrieval by an email recipient).

SMTP was derived from an the earlier Mail Transfer Protocol (MTP) and standardized in RFC 821.

RFC 2821 (April 2001) is the current base standard for SMTP. One of the most important update is the incorporation of DNS features (when RFC 821 was defined, there was no DNS !).

SMTP servers both send and receive email. The device sending email act as a client for that transaction (called SMTP sender), while the one receiving it acts as a server (called SMTP receiver).

SMTP connection and session establishment

Basic facts:

- all SMTP communication is done using the 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;
- when the SMTP server is done, it terminates the connection.

Email access and retrieval

SMTP is responsible for email transfer from the SMTP sender to the SMTP receiver. In order to access the mailbox on the local SMTP server, the recipient should use special protocols and methods designed specifically for email access and retrieval.

SMTP cannot be used for email access because:

- SMTP was designed for email transport only;
- flexibility.

Email access and retrieval

Models for email access and retrieval (RFC 1733):

- 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;
- 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;
- 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.

Email access and retrieval

Protocols and methods for email access and retrieval:

- Post Office Protocol (POP). Implements the off-line access model. The current version of POP is version 3, and the protocol is usually abbreviated POP3 for that reason;
- 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;

Email access and retrieval

Protocols and methods for email access and retrieval:

- Direct server email access. Based on establishing direct access to the server where the mailbox is located. The mailbox is just a file that can be manipulated by email client programs. Methods: using the SMTP server directly, file sharing access (using a protocol such as NFS), dial-up remote server access, telnet remote server access.
- 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):
  - a Web browser is opened and given a URL for a special Web server document that accesses the user's mailbox;
  - the Web server reads information from the mailbox and sends it to the Web browser which displays it to the user.

Pretty Good Privacy

Pretty Good Privacy (PGP) is a computer program that provides cryptographic privacy and authentication for data communication.

History of PGP:

- PGP was created by Philip Zimmermann in 1991, and distributed across the Internet together with its complete source code for free non-comercial use;
- in February 1993, Zimmermann became the formal target of a criminal investigation by the US Government for "munitions export without a license" (as PGP found its way outside the United States);
- the Federal criminal investigation ended in 1996 without filing criminal charges against Zimmermann;

PGP

- In 1996, Zimmermann and his team joined Viacrypt, which then changed its name to PGP Incorporated;
- in July 1997, PGP Inc. proposed to the IETF (Internet Engineering Task Force) which develops and promotes Internet Standards, a new standard called OpenPGP;
- OpenPGP is on the Internet Standards Track and is under active development since Nov 1998 (RFC 2440). The current specification is RFC 4880 (Nov 2007);
- In December 1997, PGP Inc. was acquired by Network Associates Inc.;
- In August 2002, several ex-PGP team members formed a new company, PGP Corporation;
- In July 2010, Symantec Corp. acquired PGP Corp.

PGP

PGP offers two cryptographic services:

- authentication (by signatures);
- confidentiality (by encryption),

and three auxiliary services:

- compression (which is applied after signing a message but before encryption);
- email compatibility (base64 conversion);
- segmentation (messages longer than a given maximum length are broken up into smaller segments).

PGP authentication and confidentiality

PGP authentication (applied to x):

- $A \rightarrow B : Z(x, sig_A(x))$

where $sig_A(x)$ is A's signature on x, and Z denotes Zip-compression. A's signature can be an RSA or DSS signature on $SHA\text{-}1(x)$.

PGP confidentiality (applied to x):

- $A \rightarrow B : (\{K\})\ \{Z(x)\}_K, \{K\}_{K_B^e}$

where the symmetric encryption is by CAST-128, IDEA, or 3DES, all in the CFB mode, and the asymmetric encryption is by RSA or ElGamal.

PGP authentication and confidentiality are obtained by applying confidentiality to $x' = (x, sig_A(x))$.

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);
- public key certification (methods to certify the authenticity of a public key).

PGP key management

PGP session key generation works as follows:

- it uses the pseudorandom number generator (PRNG) ANSI X9.17 with CAST-128 instead of two-key 3DES in EDE mode;
- the encryption key (128 bits) is either chosen randomly or it is a session key previously used;
- the initial input to the PRNG consists of two 64-bit blocks, one of them representing the current date and time and the other one being chosen randomly.

PGP key management

PGP key storage is based on associating 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 (that is, $K^e \ mod \ 64$).

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.

PGP key management

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

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

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 so that a private key can be recovered either by means of its corresponding public key or by means of its proprietary.

PGP key management

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.

PGP key management

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.

PGP key management

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 for public key encryption and signing of MIME data.

History:

- S/MIME had been proposed by RSA Data Security Inc. in 1995;
- the original specification used the IETF MIME specification with the standard PKCS#7 for secure message format;
- the development of S/MIME is now coordinated by the IETF S/MIME Working Group;
- S/MIME v2 (RFC 2311-2315), S/MIME v3 (RFC 2630-2634 and RFC 5035), S/MIME v3.1 (RFC 2634, RFC 3850-3852, and RFC 5035), S/MIME v3.2 (RFC 5751,5752).

Secure/Multipurpose Internet Mail Extensions

S/MIME provides the following functions:

- envelop data. Encrypt data and then base64-encode it;
- sign data. Sign data and then base64-encode it;
- clear-sign data. Sign data and then base64-encode the signature only;
- sign and envelope data. Signed and encrypted data are nested.

S/MIME content types

S/MIME makes use of a number of new MIME content types in order to deal with cryptographic services:

- application/pkcs7-mime. This can be used with four S/MIME types: signed-data, encrypted-data, compressed-data, and degenerate-signed-data;
- application/pkcs7-signature. This can be used with signed-data to specify clear-signatures;
- application/pkcs10-mime.

S/MIME enveloped data

S/MIME envelopes x as follows:

- $A \rightarrow B : (\{K\})\ R64(ID_{PKC}, ID_{EA}, \{K\}_{K_B^e}, \{x\}_K)$

where:

- ID_{PKC} is an identifier of the recipient's public-key certificate (which certifies K_B^e);
- ID_{EA} is an identifier of the encryption algorithm used to encrypt the session key K;
- $R64$ denotes the base64 encoding.

S/MIME signed data

S/MIME signs x as follows:

- $A \rightarrow B : R64(ID_{PKC}, ID_{MDA}, ID_{EA}, sig_{K_A^d}(x), x)$

where:

- ID_{PKC} is an identifier of the signer's public-key certificate (which certifies the public key needed to verify the signature);
- ID_{MDA} is an identifier of the message digest algorithm used to obtain a message digest of x;
- ID_{EA} is an identifier of the encryption algorithm used to encrypt the message digest of x;
- $R64$ denotes the base64 encoding.

S/MIME clear signing

S/MIME clear-signs x as follows:

- $A \rightarrow B : R64(ID_{PKC}, ID_{MDA}, ID_{EA}, sig_{K_A^d}(x')), x'$

where:

- x' is obtained by encoding x by base64 or quoted-printable encoding in case that x is not 7-bit ASCII;
- all the other elements are as in the case of S/MIME signed data.

In case of a clear signature, the message can be viewed by anyone who have MIME capabilities, although he cannot check the signature.

S/MIME – multiple recipients and multiple operations

Remarks:

- If an encrypted message is to be sent to a group of recipients, the sender must send an encrypted message to each member of the group;
- the signed and enveloped MIME formats can be nested.

S/MIME certificate processing

- S/MIME uses public-key certificates that conform to X.509;
- S/MIME managers and/or users must configure each client with a list of trusted keys.