E-mail security

Diana Gratiela Berbecaru
diana.berbecaru@polito.it

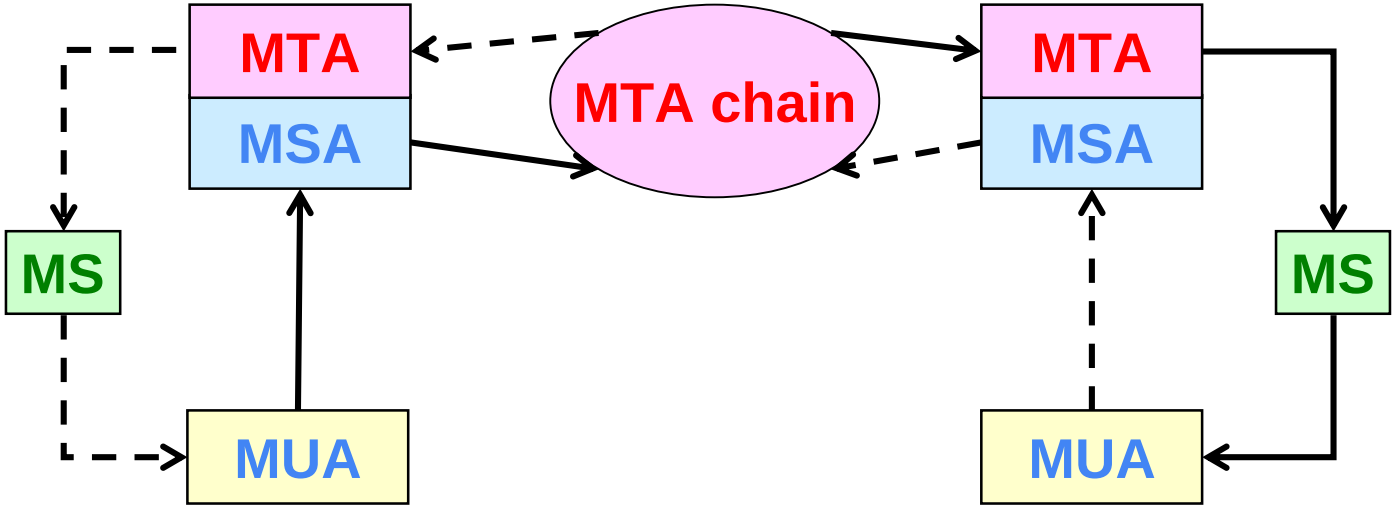
Politecnico di Torino
Dip. Automatica e Informatica

AY. 2023 - 2024

Acknowledgment

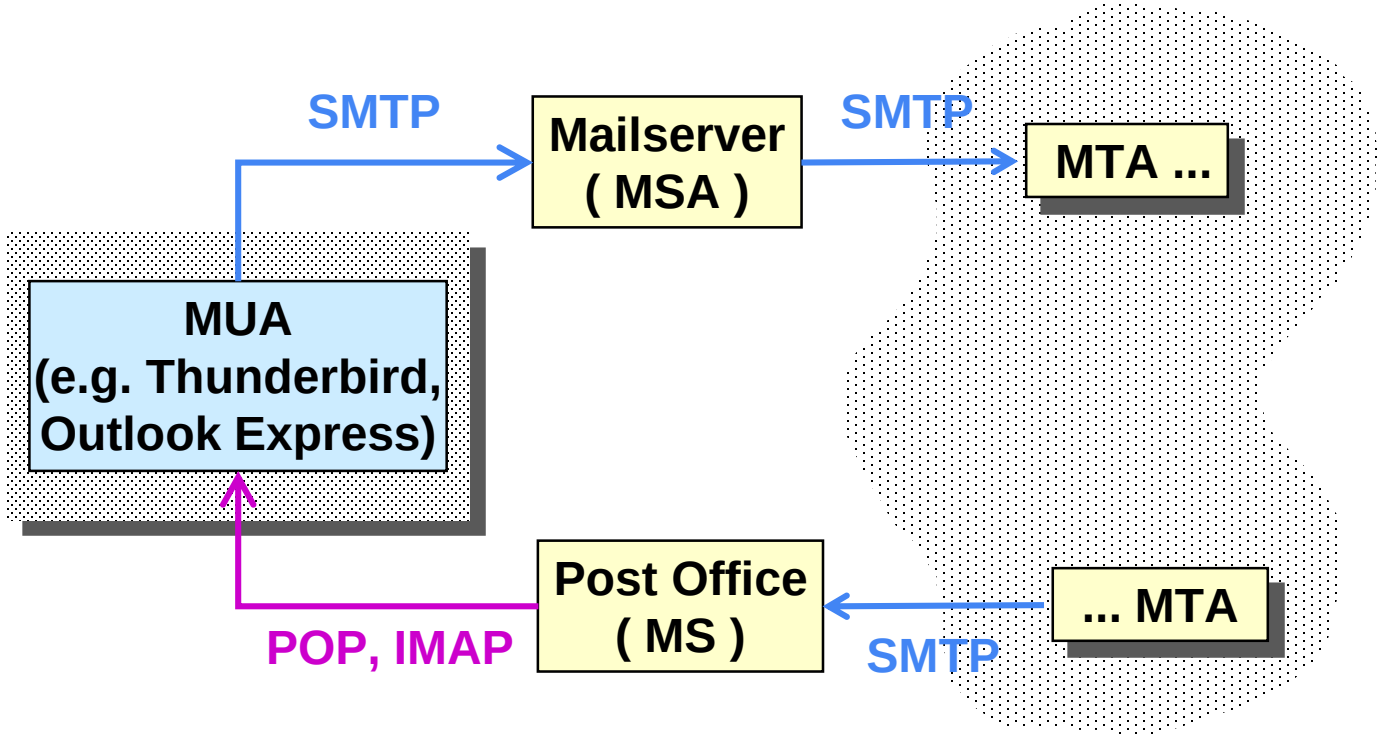
- **Slides content has been prepared by Prof. Antonio Lioy for the course Information Systems Security (2005 - 2022)**
  - minor modifications applied
- **… so this set of slides is entirely compatible with the course of the previous year(s)**

MHS (Message Handling System)

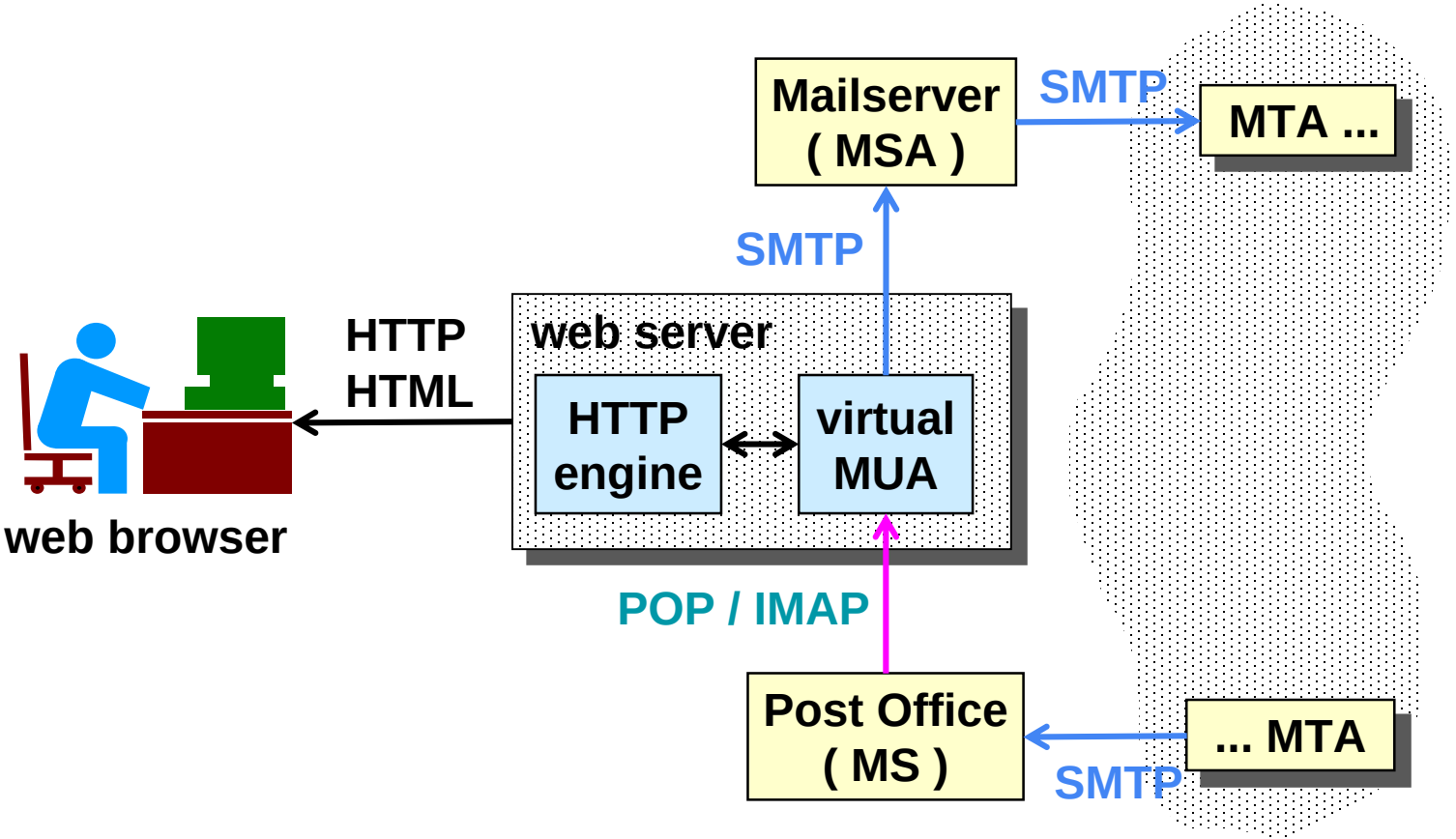


- MUA (Message User Agent)
- MSA (Message Submission Agent)
- MTA (Message Transfer Agent)
- MS (Message Store)

E-mail in client-server mode



Webmail


Mailserver
(MSA)
SMTP
MTA ...
SMTP
web server
HTTP
HTML
HTTP
engine
virtual
MUA
web browser
POP / IMAP
Post Office
(MS)
... MTA
SMTP

Protocols, ports, and formats

- **SMTP (Simple Mail Transfer Protocol)**
  - 25/tcp (MTA)
  - 587/tcp (MSA)
- **POP (Post Office Protocol)**
  - 110/tcp
- **IMAP (Internet Message Access Protocol)**
  - 143/tcp
- **“RFC-822”**
  - message format (pure text body)
- **MIME**
  - multimedia extension of RFC-822

RFC-822 messages

- **only US-ASCII characters**
  - encoded on 7 bits
  - MSB once used for error control (‘parity’ bit)
- **lines terminated by <CR> <LF>**
- **messages composed by header + body**
- **header**
  - keywords at the beginning of the line
  - continuation lines start with a space
- **body**
  - separated from the header by an empty line
  - contains the message

Header RFC-822

From:	sender (logical)
Sender:	sender (operational)
Organization:	organization of the sender
To:	destination
Subject:	subject
Date:	date and hour of sending
Received:	intermediate steps
Message-Id:	sending ID
CC:	copy to
Bcc:	copy (hidden) to
Return-Receipt-To:	return receipt to

An SMTP / RFC-822 example

telnet duke.colorado.edu 25
Trying .....
Connected to duke.colorado.edu
Escape character is '^]'
220 duke.colorado.edu ...
HELO leonardo.polito.it
250 Hello leonardo.polito.it ... Nice to meet you!
MAIL FROM: cat
250 cat ... Sender ok
RCPT TO: franz
250 franz ... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself

An SMTP / RFC-822 example (cont.)

From: cat@athena.polito.it (Antonio Lioy)
To: franz@duke.colorado.edu
Subject: vacation

Hello Francesco,
I renew my invitation to come to my place during your vacation in Italy. Let me know when you arrive.

Antonio

.

250 Ok

QUIT

221 duke.colorado.edu closing connection
connection closed by foreign host

Problems in securing e-mail

- **connectionless system (store-and-forward, also because of MX records)**
- **untrusted MTA's**
- **security of MS**
- **mailing-list encryption**
- **compatibility with what is already installed**
- **concurrent solutions:**
  - Internet = PGP, PEM, MOSS, S/MIME
  - OSI = X.400

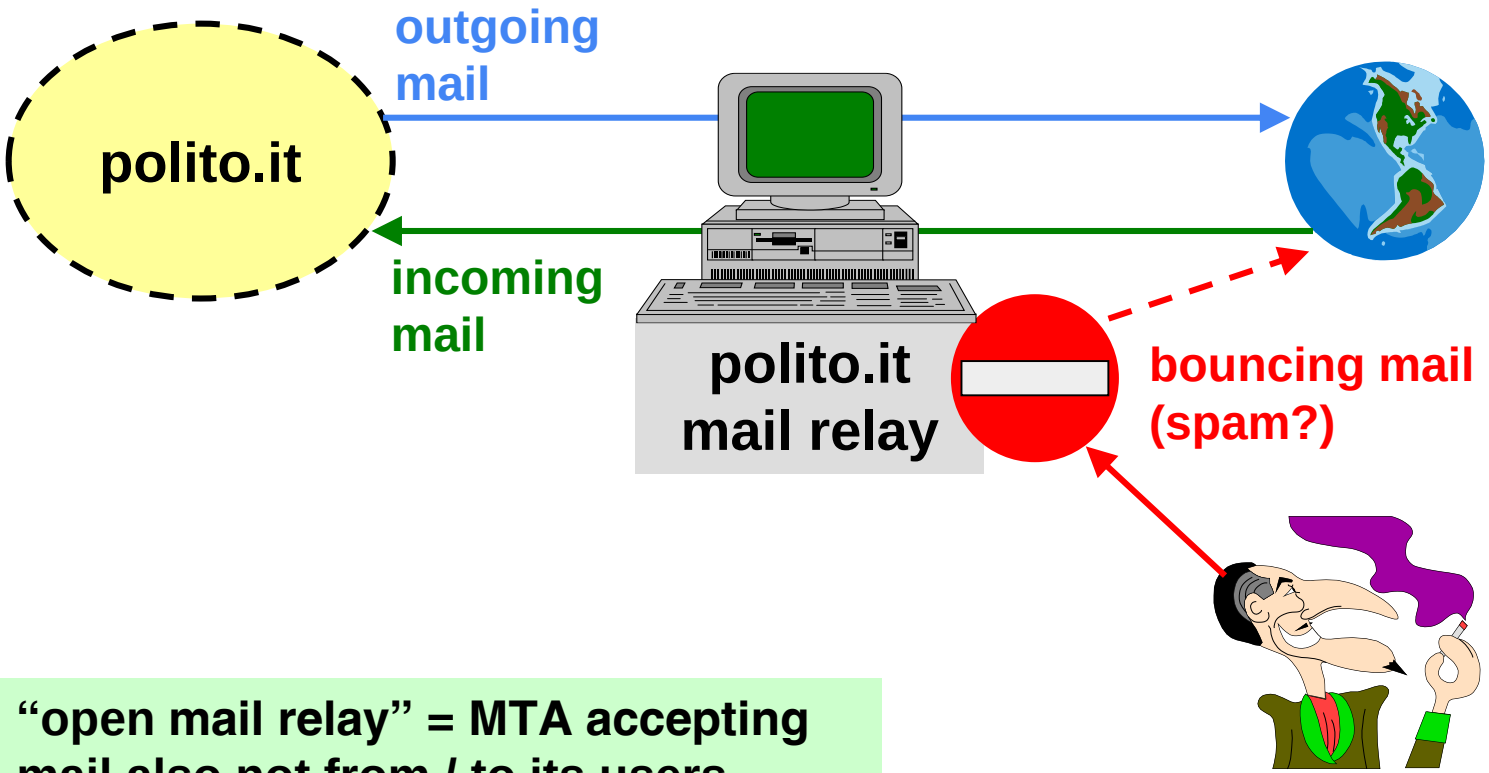
Mail spamming

- **also named UBE (Unsolicited Bulk Email) or UCE (Unsolicited Commercial E-mail)**
- **sending of unwanted messages:**
  - unauthorised advertisement
  - attacks (malware, phishing, …)
- **today it is nearly 50% of the total e-mail traffic (54% at mar'20)**
  - heavy load on servers and network channels
  - heavy annoyance to the users
- **canned pork meat and Monty Python**
- **the opposite of "spam" is "ham" (term used by identification and filtering applications)**

Spamming strategies

- **hide the real sender**
  - … but use a valid sender
- **send spam via special MTA**
  - open mail relay
  - zombie or botnet
  - with variable or phantom IP address
- **content obfuscation**
  - deliberate mistakes (e.g. Vi@gr@)
  - image rather than text
  - Bayesian poisoning (e.g. text from a book)
  - inside an error message

(Open) mail relay



“open mail relay” = MTA accepting mail also not from / to its users

Anti-spam for MSA

- **do not configure your own MSA as an “open relay” but restrict its use only to authorized users**
- **authenticate the users of our MSA:**
  - IP address of the MUA
    - problem with mobile nodes, IP spoofing and malware (at valid nodes)
  - value of the field From
    - can be easily tricked with a fake mail
  - SMTP authentication
    - secure authentication methods?

ESMTP

- **Extended SMTP, defined in RFC-1869 and subsequently incorporated (with SMTP) in RFC-2821**
- **the base protocol and the communication channel is the same**
- **the ESMTP clients must identify themselves to the communicating parties with:**
  EHLO *hostname*
- **if the receiving server speaks ESMTP, it must declare the extensions that it supports, one per line, in its response to EHLO**

SMTP-Auth

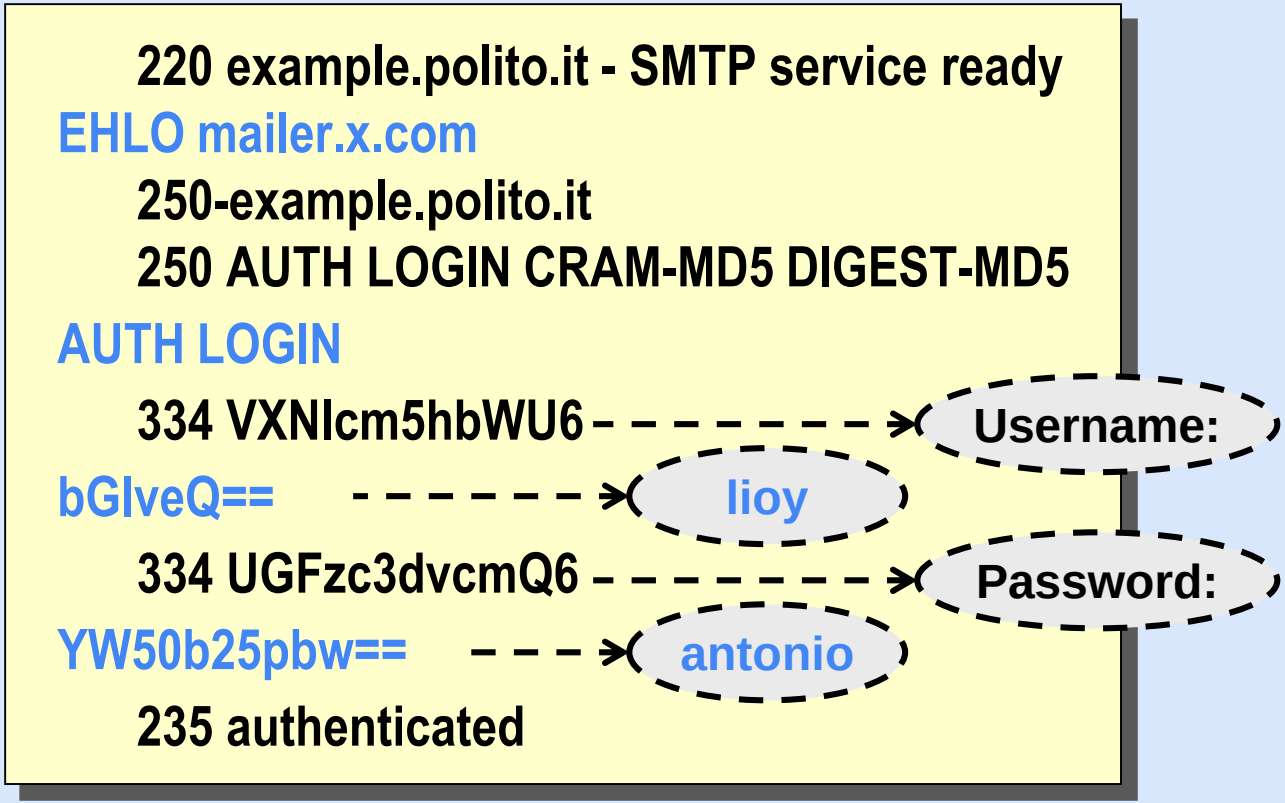
- **extension of ESMTP defined in RFC-4954**
- **command AUTH + options of MAIL FROM**
- **to authenticate a client …**
- **… before accepting messages from it!!!**
- **useful against spamming:**
  - after the EHLO command the server sends the authentication mechanisms supported
  - the client chooses one
  - the authentication protocol is executed
  - if the authentication fails, the communication channel is closed

Negative AUTH example

- **the mailer does not know (or does not accept) the authentication method proposed by the client:**

```
        220 example.polito.it - SMTP service ready
EHLO mailer.x.com
        250-example.polito.it
        250 AUTH LOGIN CRAM-MD5 DIGEST-MD5
AUTH PLAIN
        504 Unrecognized authentication type
```

AUTH: LOGIN method



AUTH: PLAIN method

- **syntax (RFC-2595):**
  AUTH PLAIN *id_pwd*$_{BASE64}$
- **id_pwd is defined as:**
  [authorize_id] \0 authentication_id \0 pwd

```
        220 example.polito.it - SMTP service ready
EHLO mailer.x.com
        250-example.polito.it
        250 AUTH LOGIN PLAIN
AUTH PLAIN bGIveQBsaW95AGFudG9uaW8=
        235 authenticated
```

lioy \0 lioy \0 antonio

AUTH: challenge-response methods

- **CRAM–MD5**
  - RFC-2195
  - challenge = base64 (nonce)
  - response = base64 (
    usr SP hmac-md5(pwd, nonce)$_{LHEX}$)
- **DIGEST–MD5**
  - RFC-2831
  - similar to HTTP/1.1 digest-authentication
  - declared obsolete in RFC-6331 (2011) and replaced with SCRAM

AUTH: CRAM-MD5 method

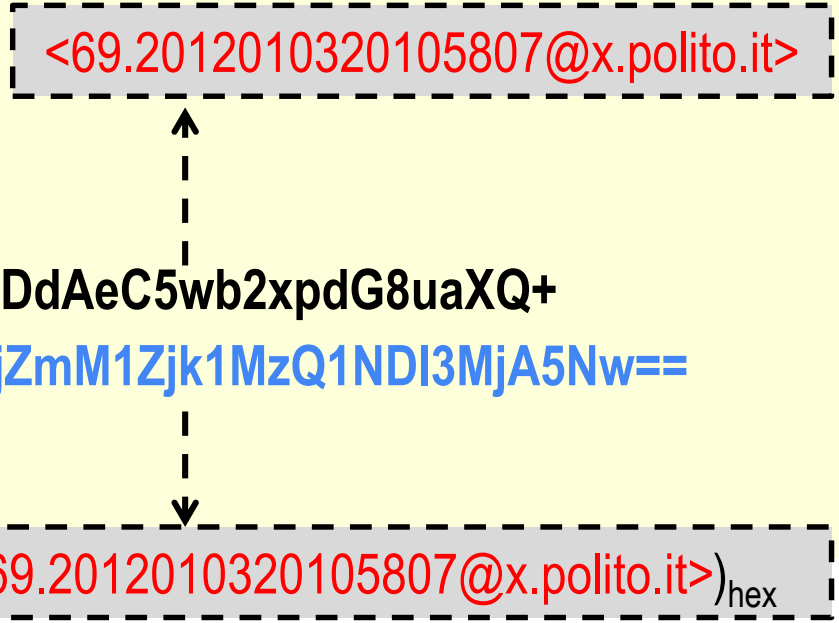
220 x.polito.it - SMTP service ready

EHLO mailer.x.com

250-x.polito.it

250 AUTH CRAM-MD5 DIGEST-MD5

AUTH CRAM-MD5



Analysis of CRAM-MD5

- **advantages:**
  - client authentication (password)
  - no replay (challenge = rnd + timestamp + FQDN)
  - resistant to sniffing (hash is not invertibile)
- **disadvantages:**
  - no server authentication (but OK if used over TLS which always provides server authentication)
  - cleartext storage of the pwd, unless the intermediate steps of HMAC are stored (i.e. K' ⊕ opad and K' ⊕ ipad)
  - dictionary attack still possible if pwd copied
  - possible MITM (channel takeover after CRAM)
    - but this is a general problem of peer authentication …

Protection of SMTP with TLS

- **RFC-2487 “SMTP Service Extension for Secure SMTP over TLS”**
- **STARTTLS**
  - option of EHLO
  - command to start TLS negotiation inside an SMTP channel
- **if the TLS negotiation is successful, the protocol status is reset (starts again from EHLO and the extensions supported can be different)**
- **if the negotiated security level is insufficient:**
  - the client sends immediately QUIT and closes the connection
  - the server responds to each command with code 554 (refused due to low security)

Protection of SMTP with TLS: example

220 example.polito.it - SMTP service ready

EHLO mailer.x.com

250-example.polito.it

250-8BITMIME

250-STARTTLS

250 DSN

STARTTLS

220 Go ahead

… TLS negotiation is started between client and server

… and new ESMTP connection is negotiated (because extensions may be different over a secure channel)

Security services for e-mail messages

- **integrity (without direct communication):**
  - the receiver will detect if the message was modified in transit
- **authentication**
  - the receiver can exactly identify the sender
- **non repudiation**
  - the sender cannot deny of having sent the e-mail message
- **confidentiality (optional):**
  - e-mail messages are not readable both in transit, on intermediate MTAs, and when stored in the mailbox

E-mail security – main ideas (I)

- **no modification to the present MTA**
  - messages encoded to avoid problems when passing through gateways or MTA non 8BITMIME
- **no modification to the present UA**
  - inconvenient user interface
- **with modification to the present UA**
  - better user interface

E-mail security – main ideas (II)

- **symmetric algorithms**
  - for the encryption of messages
  - with message key
- **asymmetric algorithms**
  - to encrypt and exchange the symmetric key
  - for digital signature
- **use public key certificates (e.g. X.509v3) for non-repudiation**
- **If the algorithms are strong and the keys are of adequate length, then the message security is based only on the security of the MUA of the recipient, not on the security of MTA (that are not trusted)**

Types of secure messages (I)

- **clear-signed**
  - msg in clear (so that anybody is able to read it) + digital signature (as an attachment or inside the msg)
  - only who has a secure MUA can verify the signature
- **signed**
  - [msg + dsig] encoded (e.g. base64, uuencode)
  - only who has a secure MUA (or performs operations manually) can decode and verify the signature

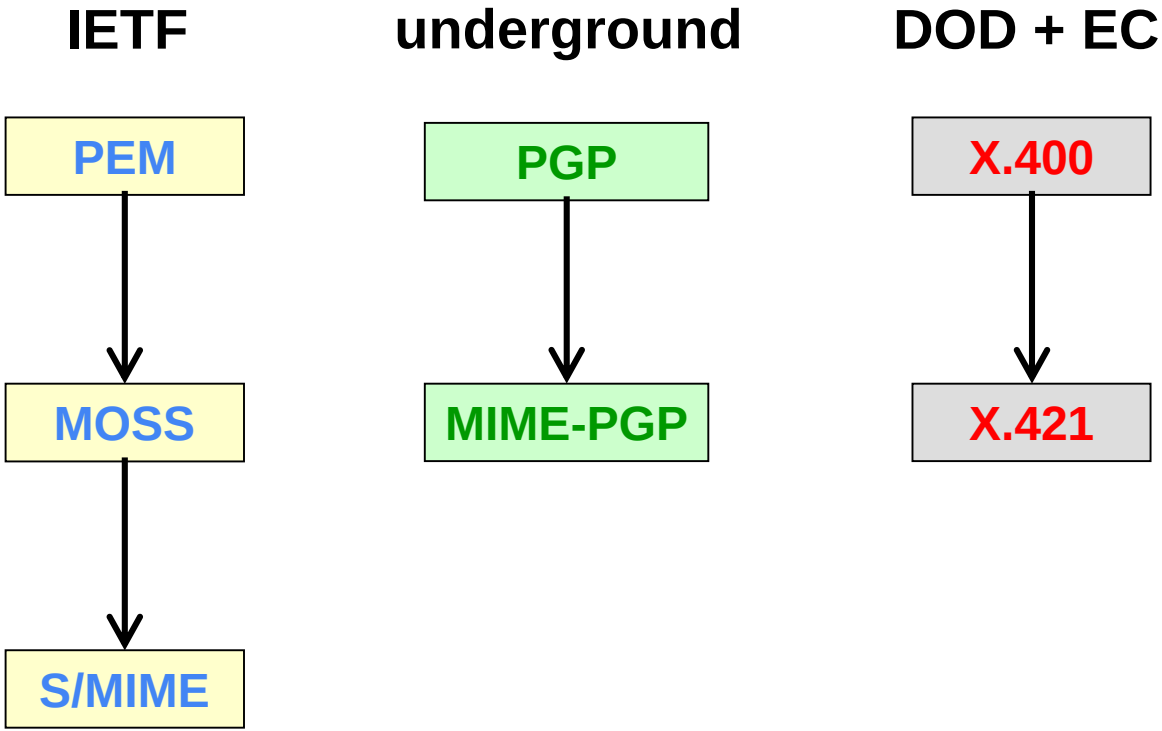
Types of secure messages (II)

- **encrypted / enveloped**
  - [encrypted msg + encrypted keys] encoded
  - only who has a secure MUA (and the keys!) can decrypt the message
- **signed and enveloped**

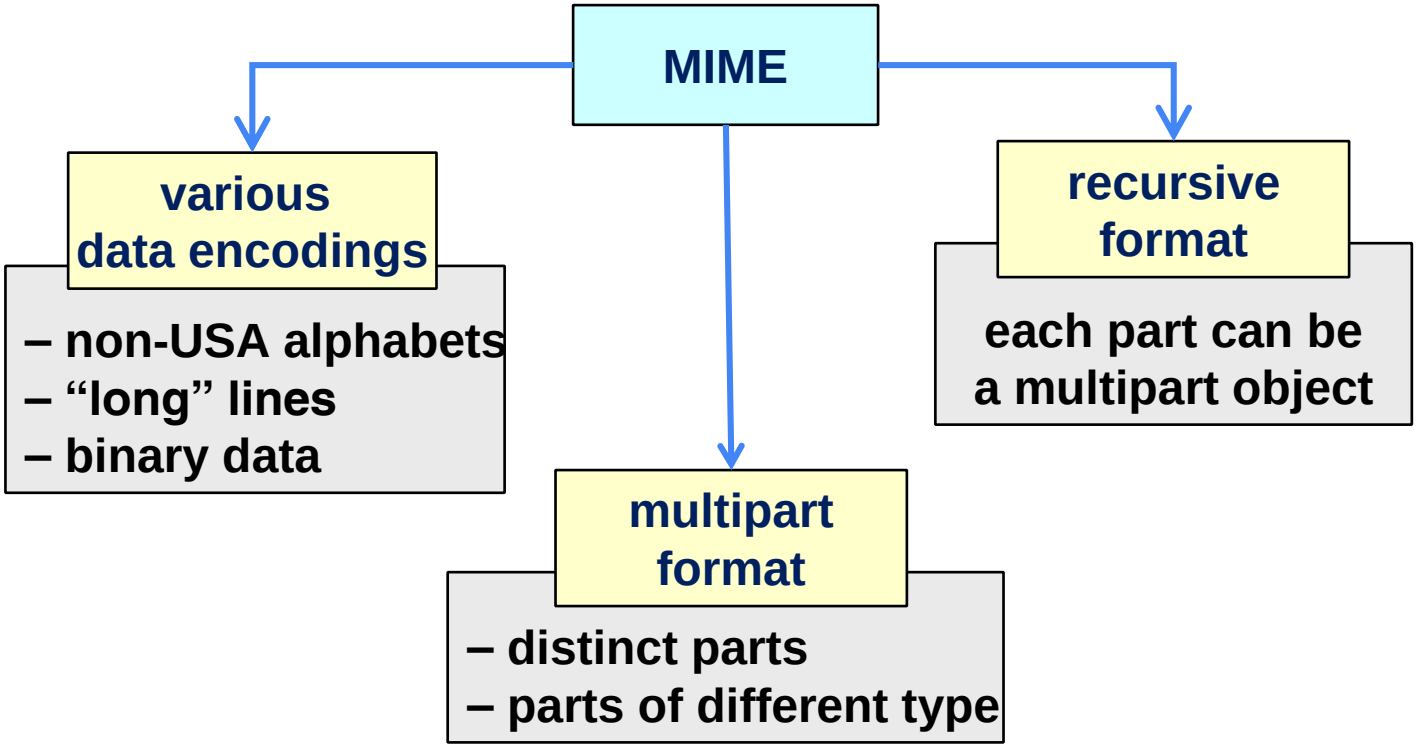
Secure messages: creation

- **transform in canonical form**
  - standard format, independent from OS / host / net
- **MIC (Message Integrity Code)**
  - integrity and authentication
  - typically: msg + { h(msg) } Kpri_sender (+ Cert (Kpub_sender))
- **encryption**
  - confidentiality
  - typically: { msg } K_M + { K_M } Kpub_receiver1 + …
- **encoding**
  - to avoid modification by the MTA
  - typically: base64, uuencode, binhex

Secure electronic mail formats

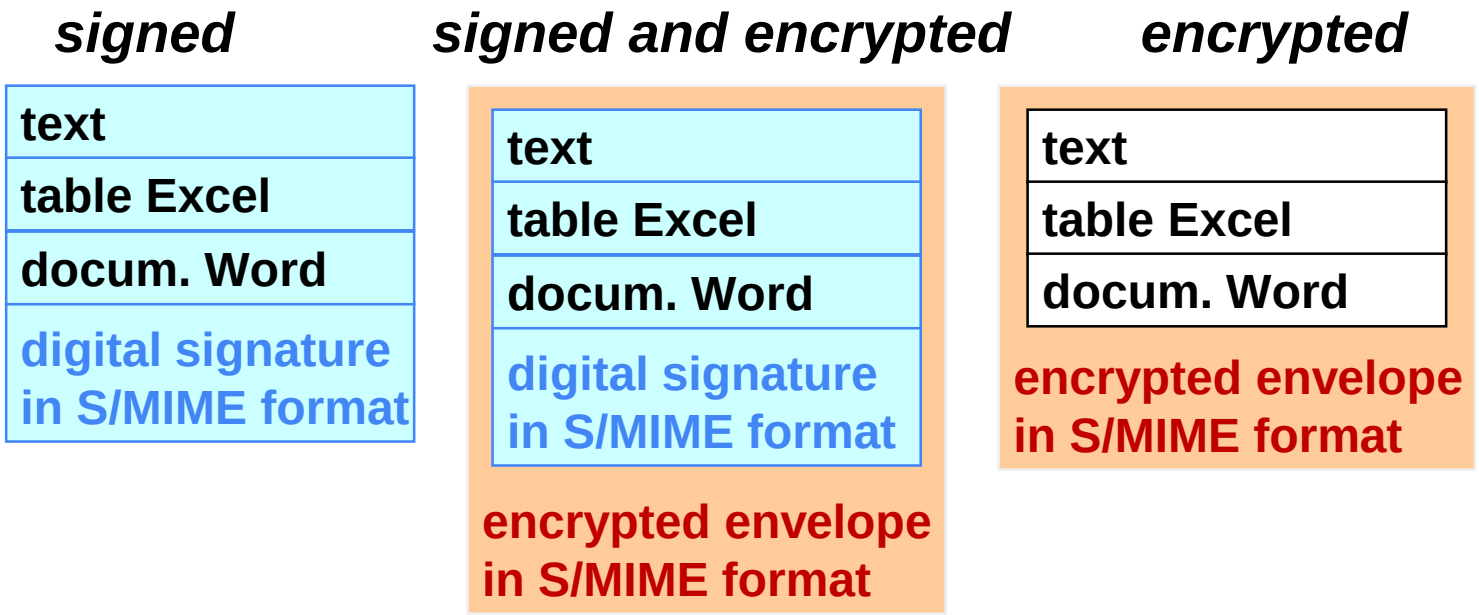


MIME (Multipurpose Internet Mail Extensions)



Secure multimedia electronic mail (S-MIME)

- **digital signature/encryption with X.509 certificates**
- **protection of MIME messages**



RFC-1847

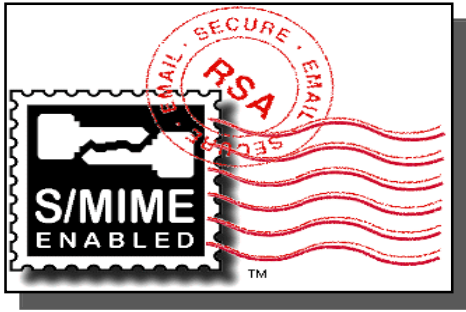
- **MIME extensions for message security**
- **for digital signature:**

  Content-Type: multipart/signed;
  protocol="TYPE/STYPE";
  micalg="...";
  boundary="..."

- **with N body parts:**
  - the first N-1 ones are those to be protected (content-type: ...)
  - the last one contains the digital signature (content-type: TYPE/STYPE)

S/MIME

- **security of MIME messages**
- **promoted by RSA**
- **v2 published as a series of informational RFC:**
  - RFC-2311 “S/MIME v2 message specification”
  - RFC-2312 “S/MIME v2 certificate handling”
  - RFC-2313 “PKCS-1: RSA encryption v.1-5”
  - RFC-2314 “PKCS-10: certification request syntax v.1-5”
  - RFC-2315 “PKCS-7: cryptographic message syntax v.1-5”



S/MIME v3, v4

- **proposed standard IETF**
- **S/MIME v3 (jun'99) then v3.1 (jul'04) and v3.2 (jan'10)**
  - RFC-2633, "S/MIME v3 message specification"
  - RFC-2632, "S/MIME v3 certificate handling"
  - RFC-2634, "Enhanced Security Services for S/MIME"
- **S/MIME v4 (apr'19)**
  - RFC-8551, "S/MIME v4 message specification"
  - RFC-8550, "S/MIME v4 certificate handling"

S/MIME architecture

- **Architecturally based on:**
  - PKCS-7 (S/MIME v2)
    CMS (since S/MIME v3)
    specifies the cryptographic characteristics and the message types (equivalent to PEM)
  - PKCS-10
    format of certificate request
  - X.509v3
    format of public key certificates

S/MIME v4.0 - algorithms

- **digital signature:**
  - (MUST) ECDSA with curve P-256 and SHA-256
  - (MUST) ECDSA with curve 25519
  - (MUST--) RSA with SHA-256
  - (SHOULD) RSASSA-PSS with SHA-256
- **Key exchange:**
  - (MUST) ECDH with curve P-256
  - (MUST) ECDH with curve X25519 with HKDF-256
  - (MUST--) RSA encryption
  - (SHOULD+) RSAES-OAEP

S/MIME v4.0 - algorithms

- **confidentiality:**
  - (MUST) AES-128-GCM and AES-256-GCM
  - (MUST--) AES-128-CBC
  - (SHOULD+) Chacha20-Poly1305
- **micalg (depends also upon digital signature):**
  - SHA-256
  - SHA-512

MIME type

- **application/pkcs7-mime, used for:**
  - msg. encrypted (envelopedData)
  - msg. signed (signedData) addressed only to S/MIME users because it is encoded in base64
  - msg. that contain only a public key (= certificate, in a degenerate signedData body)
  - standard extension: **.p7m**
  - always base64-enconded

MIME type

- **multipart/signed**
  - signed messages addressed also to users not supporting S/MIME
  - the message is in clear
  - the last MIME part is the signature (per RFC-1847) and its base64-encoded
  - standard extension for the signature: **.p7s**

- **application/pkcs10**
  - used to send a certification request to a CA
  - base64-encoded

S/MIME examples

- **encrypted**
  - B64(P7_enveloped(msg))
- **signed (only for S/MIME users)**
  - B64(P7_signed(msg))
- **clear-signed (for generic users)**
  - MIME(msg) + B64(P7_signed_detached(msg))
- **signed and encrypted**
  - B64(P7_enveloped(P7_signed(msg)))
  - B64(P7_signed(P7_enveloped(msg)))
- **note: msg is the RFC-822 body of the message**

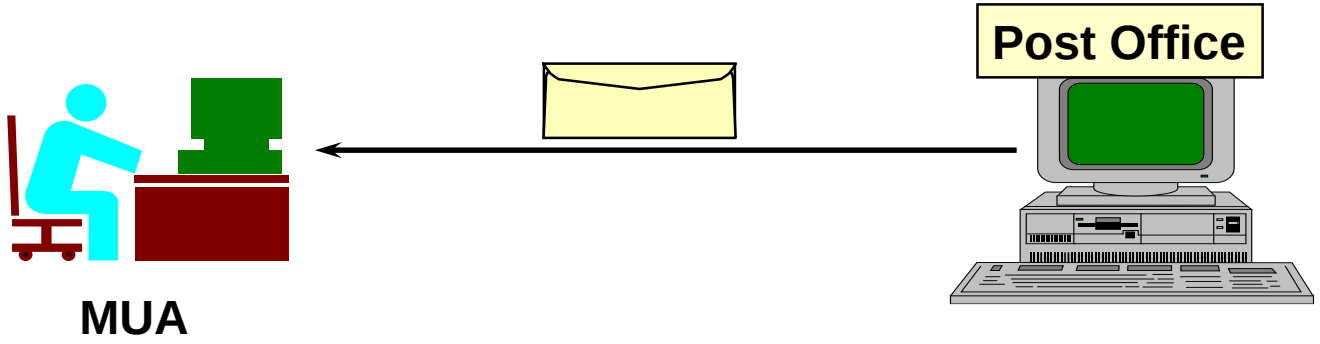
S/MIME: signature example

```
Content-Type: multipart/signed;
 protocol="application/pkcs7-signature";
 micalg=sha1;
 boundary="-----aaaaa"

-----aaaaa
Content-Type: text/plain
Content-Transfer-Encoding: 7bit

Hello!
-----aaaaa
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: base64

MIIN2QasDDSdwe/625dBxgdhdsf76rHfrJe65a4f
fvVSW2Q1eD+SfDs543Sdwe6+25dBxfdER0eDsrs5
-----aaaaa-
```

Naming in S/MIME

- **used for:**
  - selecting the certificate
  - verifying the sender's address
- **S/MIMEv2 uses the Email= or E= fields in the DN of the X.509v3 certificate, but it is possible to use the extension subjectAltName with rfc822 encoding**
- **S/MIMEv3 mandates the use of the subjectAltName extension with rfc822 encoding**

Client-server e-mail services



- authentication of the user
- authentication of the server
- confidentiality/integrity of mail messages
  - on the server
  - while in transit

Client - server e-mail services

- **POP (Post-Office Protocol)**
  - POP-2 (RFC-937), POP-3 (RFC-1939)
    user authentication by means of a password in clear (!!!)
  - APOP
    user authentication via symmetric challenge-response
  - K-POP
    mutual authentication by means of tickets
- **IMAP (Internet Mail Access Protocol)**
  - username and password in clear
  - can use OTP, Kerberos or GSS-API

POP-3 example

telnet pop.polito.it 110

+OK POP3 server ready <7831.84549@pop.polito.it>

USER lioy

+OK password required for lioy

PASS antonio

+OK lioy mailbox locked and ready

STAT

+OK 2 320

.....

QUIT

+OK POP3 server signing off

RFC-2595 (TLS for POP / IMAP)

- **RFC-2595**
  “Using TLS with IMAP, POP3 and ACAP”
- **first the communication channel is opened then the security characteristics are negotiated by means of a dedicated command:**
  - STARTTLS for IMAP and ACAP
  - STLS for POP3
- **client and server must allow to be configured to reject *user* and *password***
- **client compares the identity in the certificate with the identity of the server**

©2023 by Diana Berbecaru. Permission to make digital or hard copies of part or all of this set of slides is currently granted *only for personal or classroom use.*