E-mail security

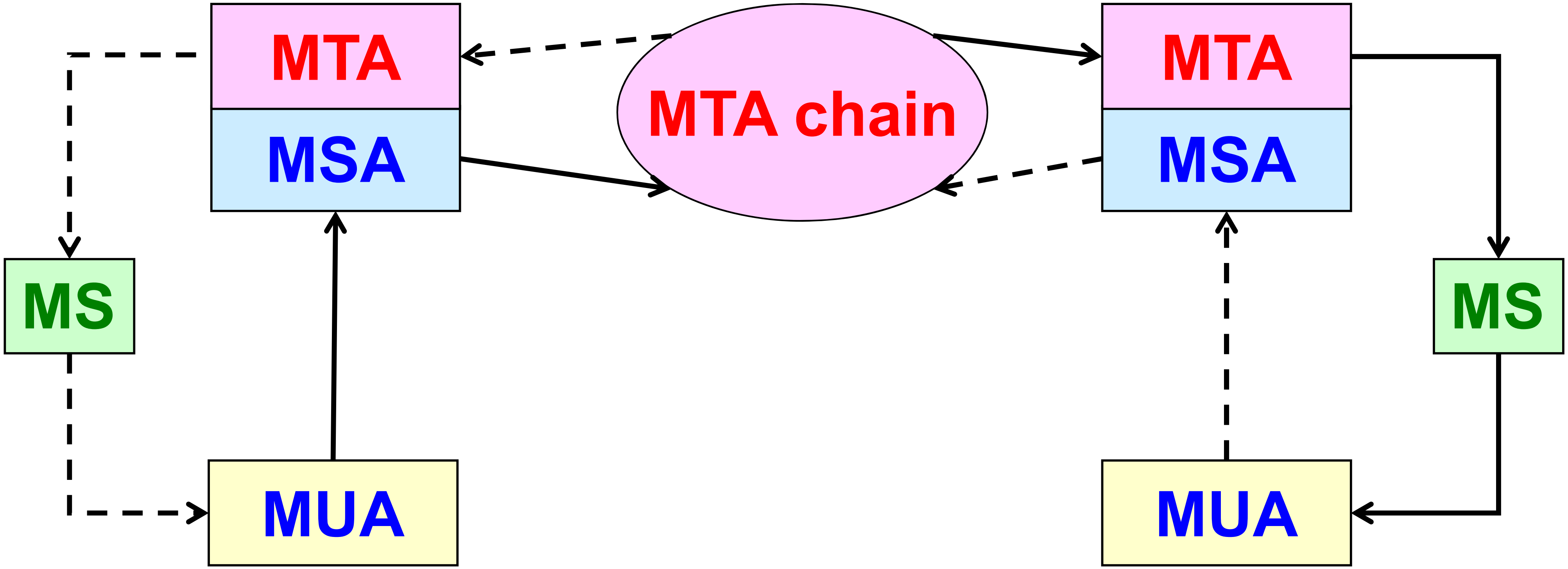
Antonio Lioy

< lioy @ polito.it>

Politecnico di Torino

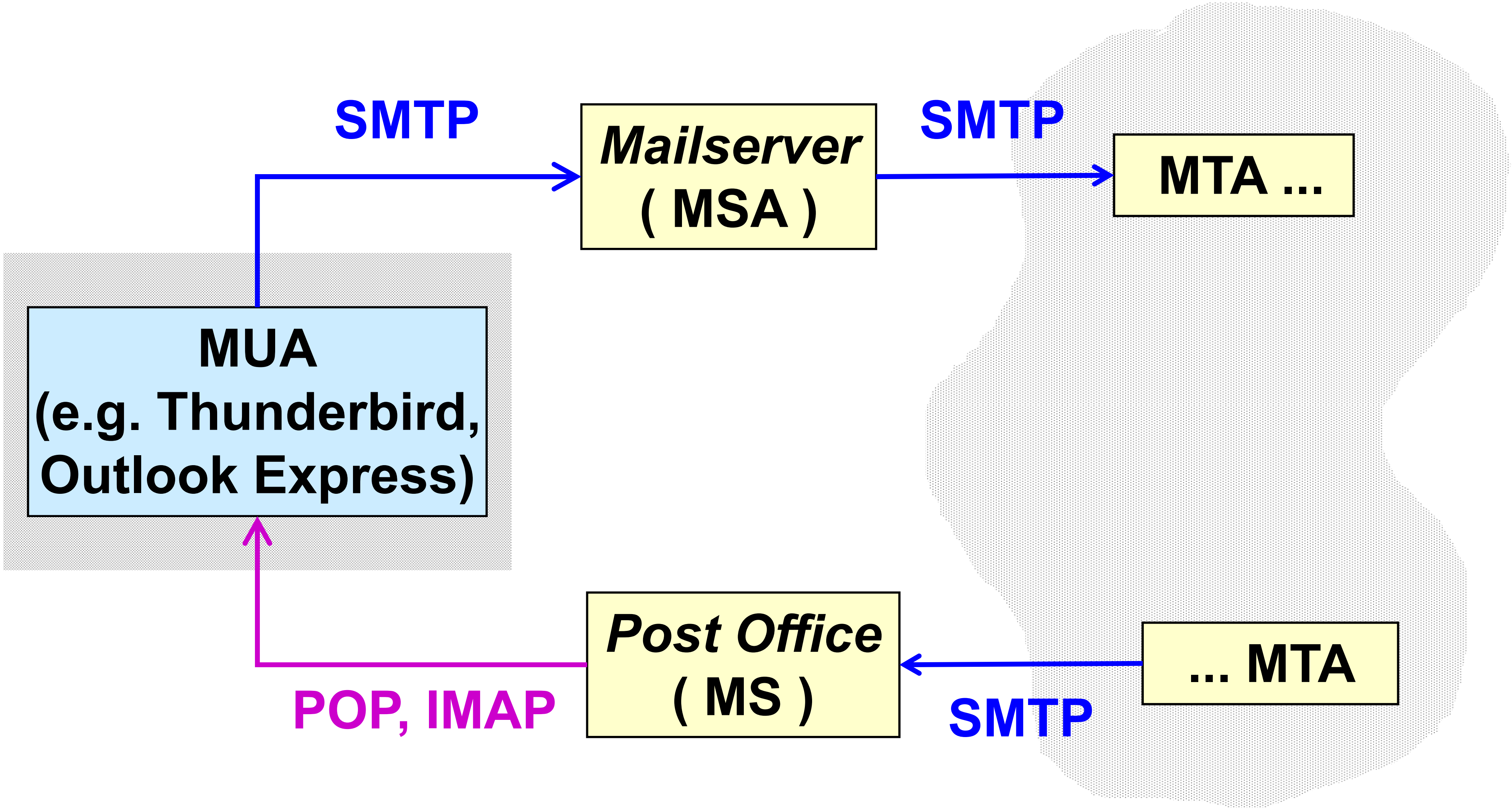
Dip. Automatica e Informatica

MHS (Message Handling System)

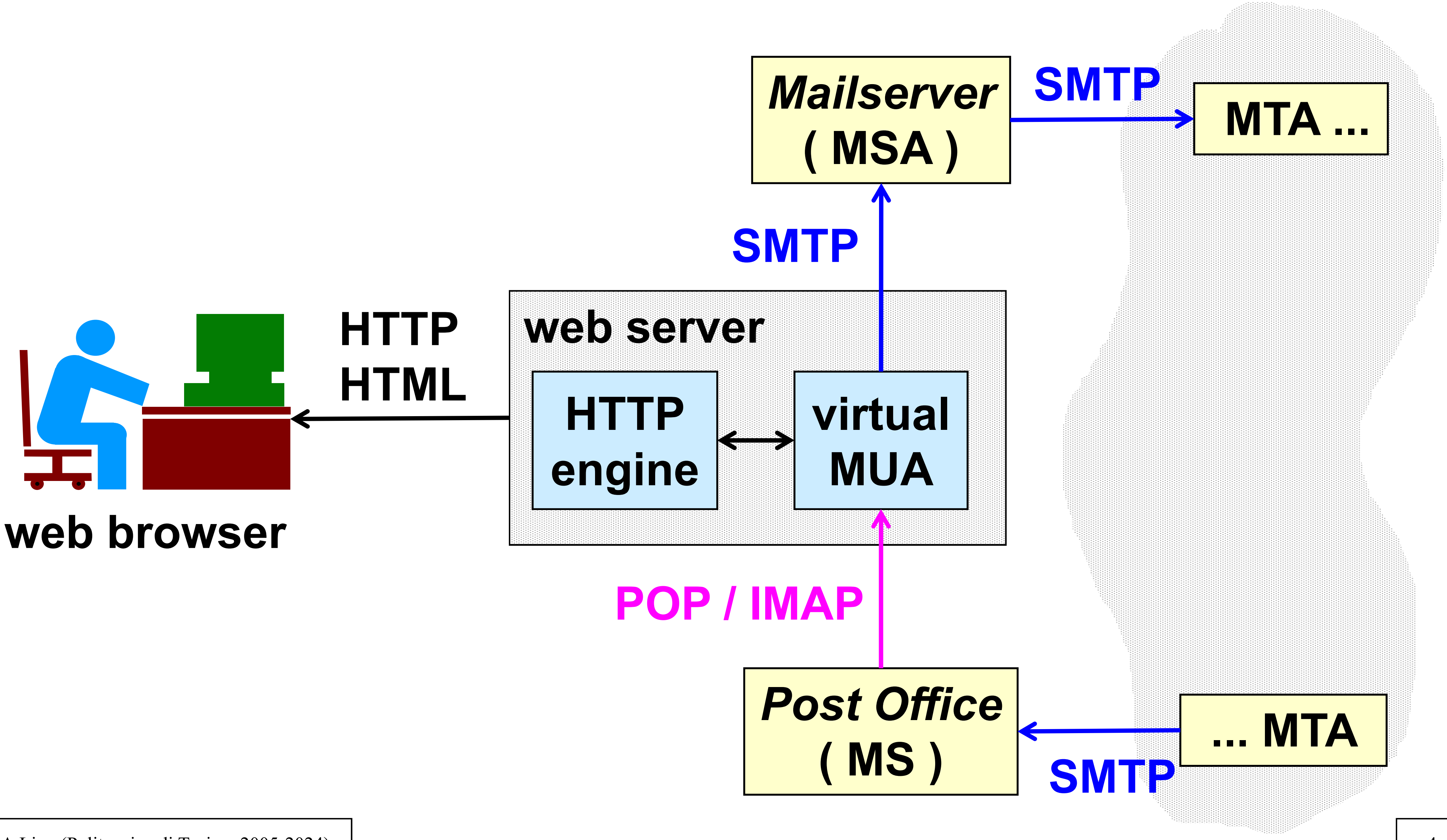


MUA = Message User Agent
MSA = Message Submission Agent
MTA = Message Transfer Agent
MS = Message Store

E-mail in client-server mode



Webmail


Mailserver
(MSA)
SMTP
MTA ...
SMTP
HTTP
HTML
web server
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
```

```
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 = (PEM, MOSS), S/MIME, PGP
  - OSI = X.400

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

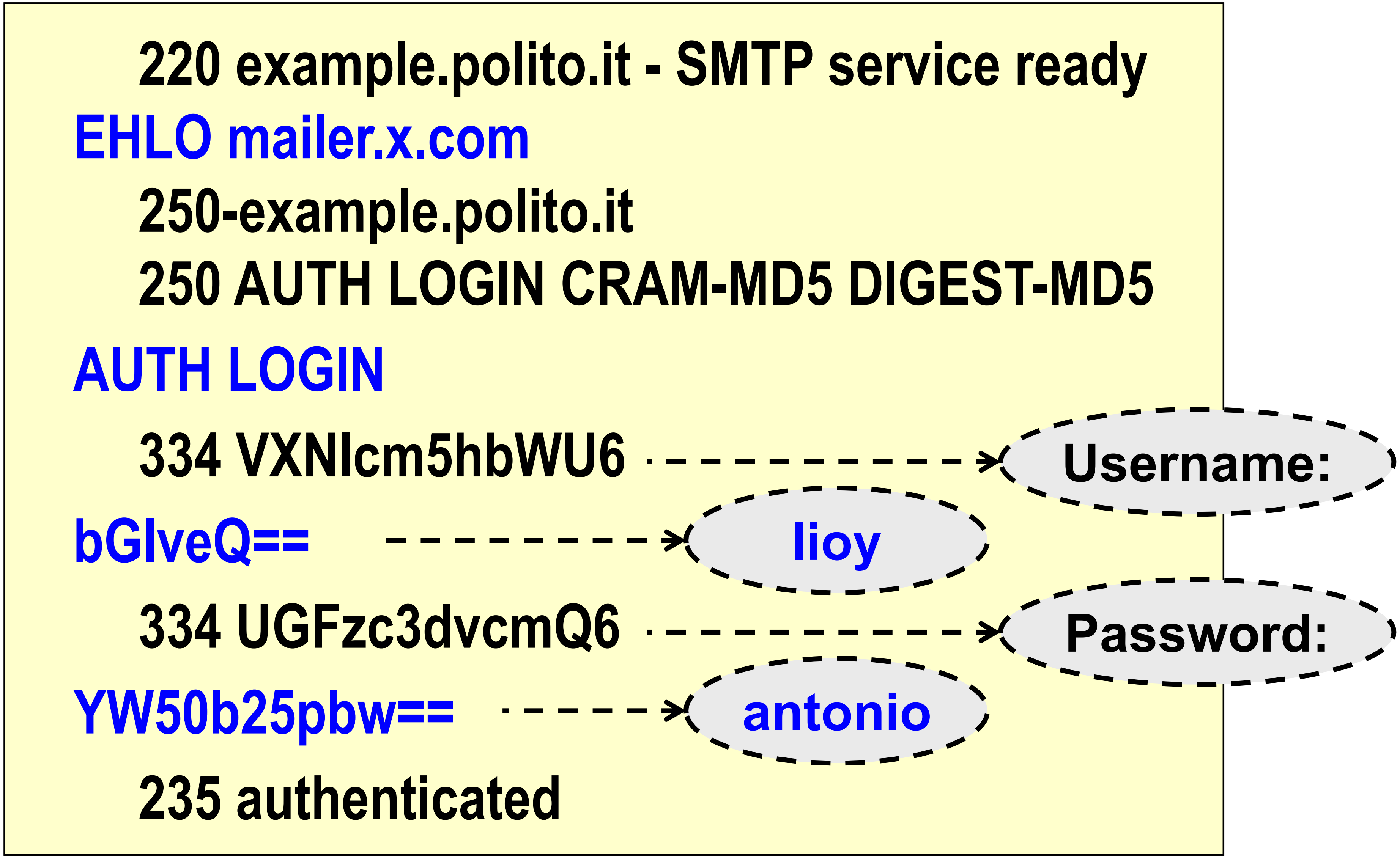
- **extension of ESMTP defined in RFC-4954**
- **command AUTH + options of MAIL FROM**
- **to authenticate a client …**
  - … before accepting messages from it!!!
  - typically used by MSA
- **authentication process:**
  - after the EHLO command the server sends the authentication mechanisms supported
  - 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



```
    220 example.polito.it - SMTP service ready
EHLO mailer.x.com
    250-example.polito.it
    250 AUTH LOGIN CRAM-MD5 DIGEST-MD5
AUTH LOGIN
    334 VXNlcm5hbWU6                    Username:
bGlveQ==                   lioy
    334 UGFzc3dvcmQ6                    Password:
YW50b25pbw==               antonio
    235 authenticated
```

AUTH: PLAIN method

- syntax (RFC-2595):
  AUTH PLAIN *id_pwd*$_{\text{BASE64}}$
- 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 bGlveQBsaW95AGFudG9uaW8=
    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)$_{\text{LHEX}}$)
- **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
  334 PDY5LjIwMTIwMTAzMjAxMDU4MDdAeC5wb2xpdG8uaXQ+
bGlveSA1MGUxNjJiZDc5NGZjNDNjZmM1Zjk1MzQ1NDI3MjA5Nw==
  235 Authentication successful
```

<69.2012010320105807@x.polito.it>

lioy hmac-md5(antonio,<69.2012010320105807@x.polito.it>)$_{hex}$

Analysis of CRAM-MD5

- **advantages:**
  - client authentication (password)
  - no replay (challenge = rnd + timestamp + FQDN)
  - resistant to sniffing (hash is not invertible)
- **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 a 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-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 mail
- **confidentiality (optional):**
  - messages are not readable both in transit and when stored in the mailbox

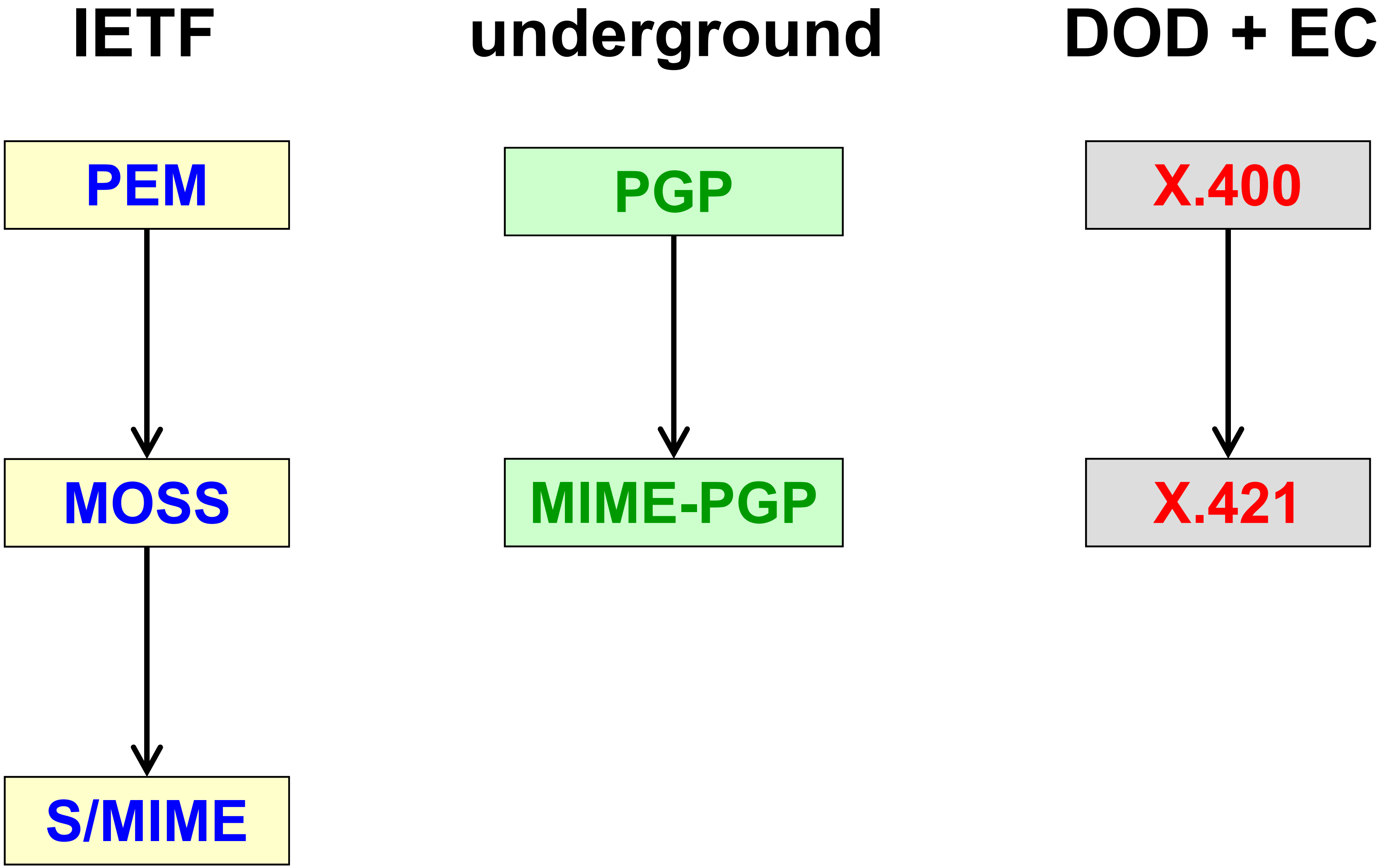
Types of secure messages

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

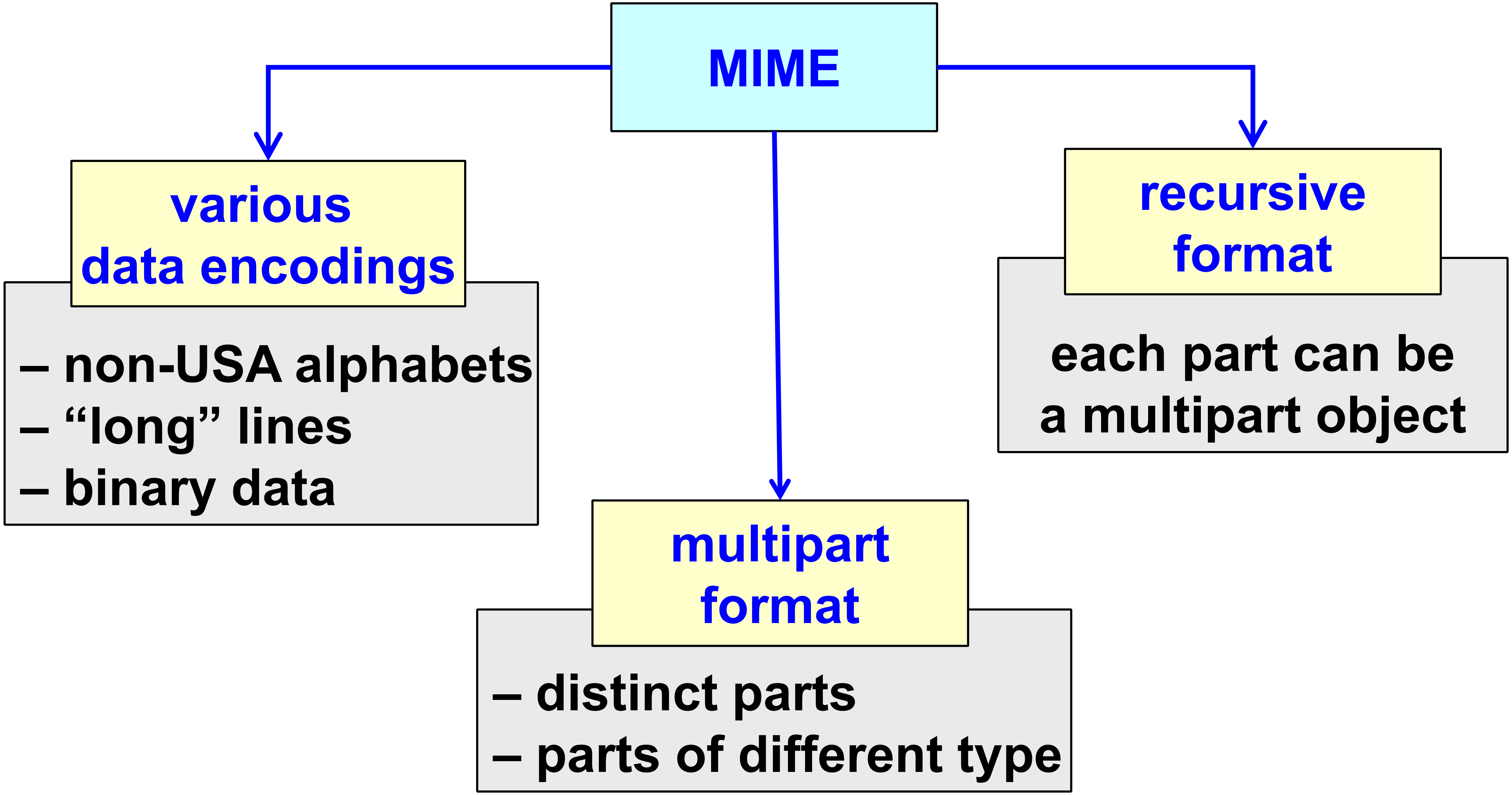
Secure messages: creation

- **transform in canonical form**
  - standard format, independent from OS / host / net
- **MIC (Message Integrity Code)**
  - integrity and authentication
  - typically: msg + { h(msg) } SK_sender
- **encryption**
  - confidentiality
  - typically: { msg } K_M + { K_M } PK_receiver1 + …
- **encoding**
  - to avoid modification by the MTA
  - typically: base64, uuencode, binhex

Secure electronic mail formats



MIME
(Multipurpose Internet Mail Extensions)



MIME

various data encodings

- non-USA alphabets
- “long” lines
- binary data

multipart format

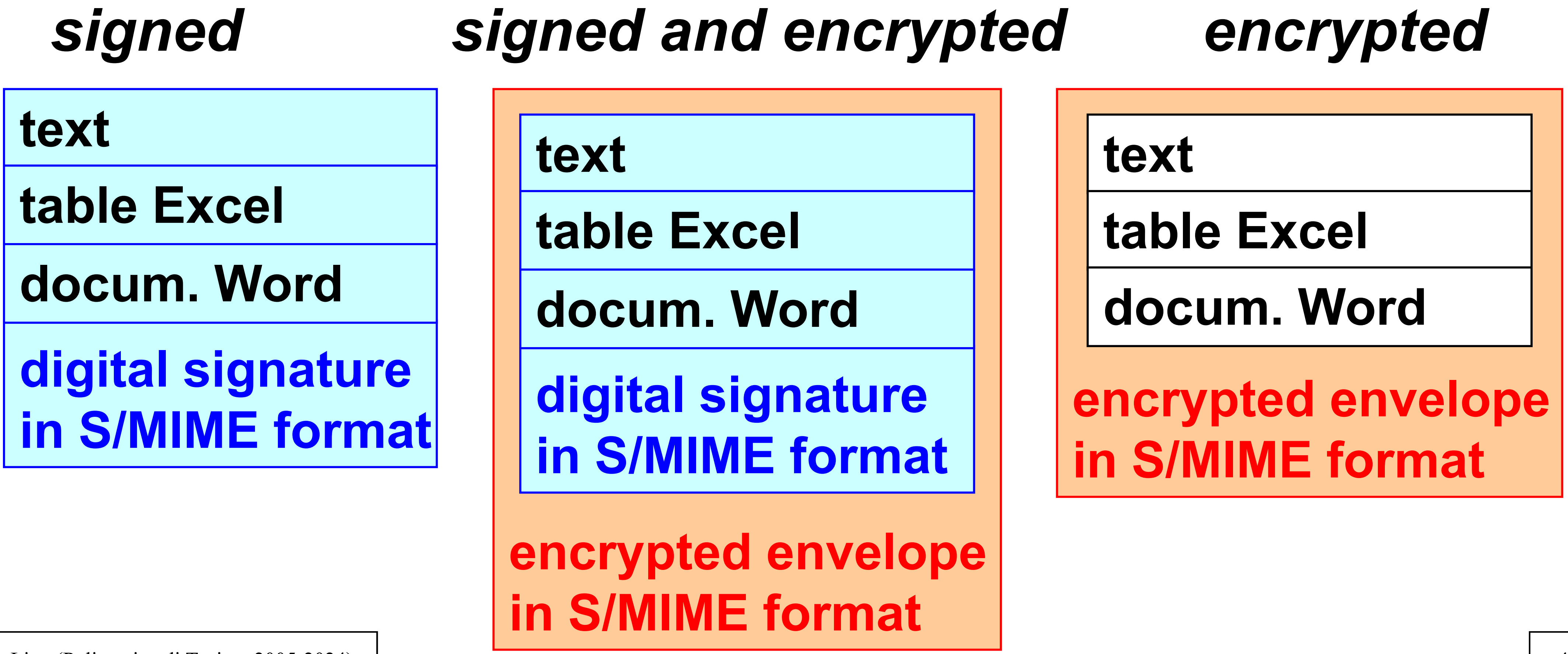
- distinct parts
- parts of different type

recursive format

each part can be a multipart object

Secure multimedia electronic mail (MOSS or S-MIME)

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



signed

text

table Excel

docum. Word

digital signature in S/MIME format

signed and encrypted

text

table Excel

docum. Word

digital signature in S/MIME format

encrypted envelope in S/MIME format

encrypted

text

table Excel

docum. Word

encrypted envelope in S/MIME format

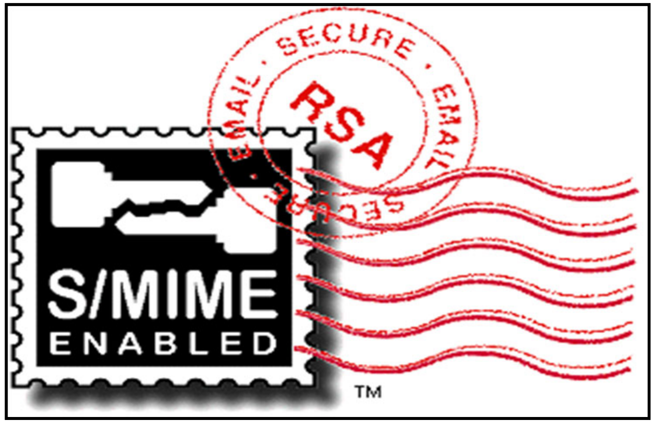
RFC-1847

- **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**
- **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.509
  format of public key certificates

S/MIME v4.0 – algorithms

- **digital signature:**
  - (MUST) ECDSA with curve P-256 and SHA-256
  - (MUST) EdDSA 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 e 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-encoded

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))
- **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**

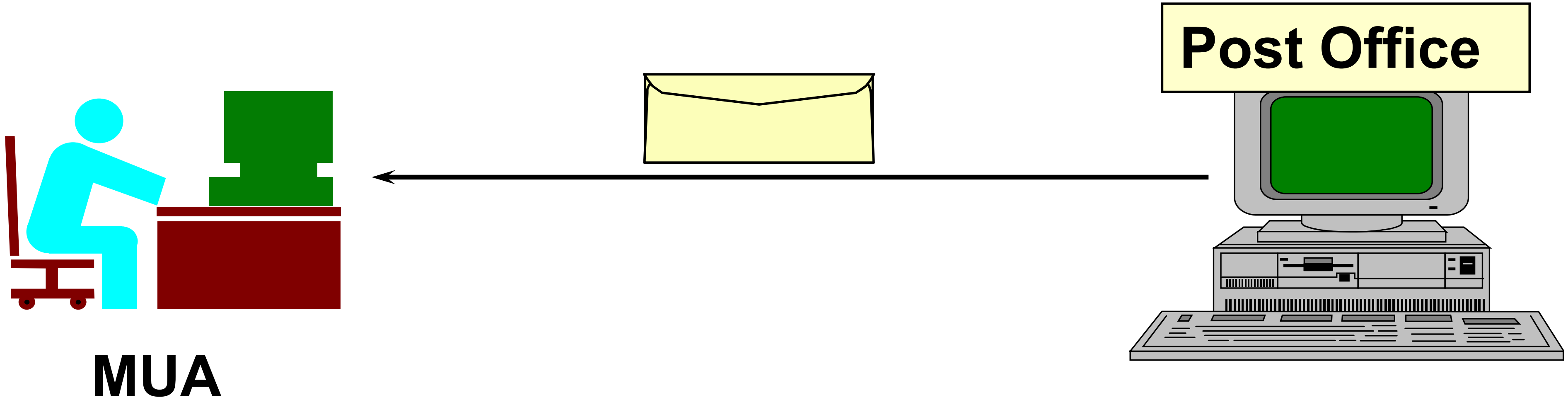
S/MIME: signature example

```
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.509 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 per 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**

Separate ports for SSL/TLS?

- **discouraged by IETF due to the following reasons:**
  - involve different URLs (e.g. http and https)
  - involve an incorrect secure / insecure model (e.g. is 40-bit SSL secure? is it insecure an application without TLS but with SASL?)
  - not easy to implement “use TLS if available”
  - doubles the number of necessary ports
- **… but it presents some advantages:**
  - simple to filter traffic on packet-filter firewalls
  - TLS with client-authentication allows not to expose the applications to attacks