Internet infrastructure

Prof. dr. ir. André Mariën
Mail message formats
Mail message formats

• RFC (2)822
• MIME
• S/MIME
RFC822 structure

Rfc822 message

From: x@y.com
To: i@j.org
Cc: u@v.net

Hello,
RTFM
Goodbye

(c) A. Mariën -
RFC 2822

• Standard for Internet Message Format
• Distinguished formalized header and body
• Header format: <field name> “:” <field body>
• Field names:
  – "Return-Path", "Received", "Date", "From", "Subject", "Sender", "To", “Cc”, “Bcc”
  – "Message-ID", "References”, "Keywords”, "Subject”
Characteristics

• Shortcomings:
  – Messages consist of lines of text
  – No significant consideration to
    • data compression
    • transmission and storage efficiency

• MIME and S/MIME address many of these shortcomings
SMTP envelop – RFC822 headers

SMTP envelope:
  RCPT TO: d@x.com
  MAIL FROM: f@y.comp

RFC822 message
  From: u@i.com
  To: d@x.com
  Subject: all is discovered!
  Run while you can...
Bcc - email groups

• Bcc: blind carbon copy
  – Rfc822 headers do not contain Bcc: for other destinations
  – SMTP envelope contains delivery address

• Mail groups
  – Expansion by sending agent
    • User defined groups
  – Expansion by SMTP agent
    • Corporate supported mail groups
  – Expansion by receiving agent
    • External addressable alias
    • Expansion on reception
MIME

Multipurpose Internet Mail Extensions
MIME Header Fields

- A number of new RFC 822 header fields to describe the content of a MIME entity.

- Header fields occur in at least two contexts:
  - As part of a regular RFC 822 message header.
  - In a MIME body part header within a multipart construct.

New Headers

- MIME-Version: 1.0
- Content-type
- Content-Transfer-Encoding
Content Type

Syntax:
- `<type>/<subtype>`
  *(`;<attribute>=<value>)`
  - `type := discrete-type / composite-type`
  - `discrete-type := "text" / "image" / "audio" / "video" / "application"`
  - `composite-type := "message" / "multipart"`
- The subtype specification is MANDATORY
- The type, subtype, and parameter names are not case sensitive

Parameters:
- `text/plain; charset= "US-ASCII"
  - the "charset" parameter is applicable to any subtype of "text"
- `multipart/mixed; boundary="=aaaaaa ="
  - the "boundary" parameter is required for any subtype of the "multipart" media type.
Content-Type

• completely equivalent:
  – Content-type: text/plain; charset=us-ascii (Plain text)
  – Content-type: text/plain; charset="us-ascii"

• Content-Type Defaults
  – no Content-Type: plain text US-ASCII character set
  – Content-type: text/plain; charset="us-ascii"
Encoding
Encoding

• "Content-Transfer-Encoding" Header Field
• RFC 821 (SMTP) restricts mail messages
  – to 7bit US-ASCII data
  – lines <= 1000 characters including any trailing CRLF line separator.
• standard encoding mechanisms: 7bit, 8bit, binary, quoted-printable, base64
• provides two pieces of information:
  – sort of encoding transformation
  – specifies what the domain of the result is
Defined encodings

- **encodings**: identity, the "quoted-printable" encoding, "base64" encoding
- **domains**: "binary", "8bit", "7bit".
  - "7bit", "8bit", "binary": identity encoding (= NO encoding, just stating the content as 7 bit or ...)
- **encodings** "quoted-printable", "base64":
  - quoted-printable: +- readable encoding
  - Base64: uniform encoding
Example encoding:

• headers
  – Content-Type: text/plain; charset=ISO-8859-1
  – Content-transfer-encoding: base64

• meaning
  – the body is a base64 US-ASCII encoding
  – data was originally in ISO-8859-1
"Quoted-Printable"

• What:
  – Content-Transfer-Encoding: quoted-printable

• When:
  – data that largely consists of printable characters (also) in US-ASCII set

• How: Quoted-printable: rules
  – General 8bit representation:
    • $c \Rightarrow \"=\" + 2$ two digit hexadecimal representation
    • example: US-ASCII \"=\", dec 61 \Rightarrow \"=3D\"
  – Literal representation:
    • $33 \leq c \leq 60 \&\& 62 \leq c \leq 126 \Rightarrow$ corresponding US-ASCII characters

(c) A. Mariën -
Quoted-printable:

• rules
  – White Space:
    • c == 9 => US-ASCII TAB, c == 32 US-ASCII SPACE
  – Line Breaks:
    • a CRLF sequence in the text=> (RFC 822) line break: CRLF
    • Sequences like "=0D", "=0A", "=0A=0D" and "=0D=0A" may appear in data
• Soft line breaks
  – lines <= 76 characters long
  – longer lines: use "soft" line breaks: "=" as last character on line
  – example:
    Listen carefully, I will only say this once.
    Listen carefully, =
    I will only=
    say this once.
"Base64" Content-Transfer-Encoding

• Objective
  – To represent arbitrary sequences of octets in a widely, safely, transportable format
    • It need not be humanly readable
Base64: encoding process

- 3 byte input => 4 encoded characters (bytes)
  - Encoded is about 33 percent larger than un-encoded
- How:
  - 3 8bit inputs => 24 bits => 4 * 6bit groups
  - each: single digit in the base64 alphabet
  - A 65-character subset of US-ASCII is used: 64 + special ("=")
    - 0-25  => A-Z
    - 26-51 => a-z
    - 52-61 => 0-9
    - 62  => +
    - 63  => /
  - ", is used to signify a special processing function
- represented identically in all versions of ISO 646, including US-ASCII & EBCDIC.
Encoding/decoding

F(bbbbb)  F(bbbbb)  F(bbbbb)  F(bbbbb)
Encoding/decoding: end
MIME: Media Types
MIME: Media Types

- "text": textual information
- subtype: "plain": text containing no formatting commands or directives of any sort
  
  Content-type: text/plain; charset=iso-8859-1

- charsets:
  
  - "US-ASCII": ANSI X3.4-1986
MIME: Media types

• application: octet-stream, postscript, pdf
• audio: basic, mpeg
• image: jpeg, gif
• message: rfc822, news
• model: vrml
• multipart: form-data, signed, mixed, alternative
• text: plain, html, xml
• video: mpeg, quicktime
MIME: Media Type: "application"

• some other kind of data, typically either uninterpreted binary data or information to be processed by an application

• subtype "octet-stream": uninterpreted binary data
Composite top-level media types

• multipart: multiple entities of independent data types.

• Multipart subtypes:
  – "mixed": a generic mixed set
  – "alternative": same data in multiple formats
  – "parallel": to be viewed simultaneously
  – "digest": default type of each part is "message/rfc822".
Multipart body

must contain body parts:

- each one preceded by a boundary line
- followed by a closing boundary line
- Content-Type: multipart/xxx; boundary="<somestring>"
- The boundary delimiter line:
  - CRLF--<somestring><CRLF>
- Final boundary delimiter line:
  - CRLF--<somestring>--<CRLF>
Each part in multipart

- initial boundary delimiter line
  - header area, blank line, body area
- RFC 822 syntax, different meaning.
- NO header fields are actually required in body parts
- relevant header fields: "Content-*".
- no Content-Type, defaults:
  - "multipart/digest": "message/rfc822"
  - Otherwise: "text/plain"
Preamble and epilogue

• Problem in multipart specification: two unused areas
• In general: not used: no proper semantics
• many implementations insert an explanatory note for recipients who read the message with MIME-challenged software
Example: multipart message

Content-type: multipart/mixed; boundary="simple"

This is the preamble.
--simple
Content-type: text/plain; charset=us-ascii

message

--simple--
This is the epilogue. It is also to be ignored.
Multipart: "message": encapsulated message

- subtype: "rfc822"
- type "message/rfc822": body contains an encapsulated message, with RFC 822 syntax
- a "message/rfc822" message could well be a News article or a MIME message.
MIME typing

• Content handled by application
• File recognition
  – Magic numbers
  – File extension
  – Fingerprinting content
• Configuration
  – Application – MIME type – file extension(s)
References

• RFC 2045: headers used to describe the structure of MIME messages
• RFC 2046: general structure of the MIME media typing system and defines an initial set of media types
• RFC 2047: to allow non-US-ASCII text data in header fields.
• RFC 2049: conformance criteria & examples, acknowledgements, bibliography.
S/MIME
S/MIME

- Secure MIME
S/MIME

• RFC 2311, 2633: S/MIME Version 2&3 Message Specification
• S/MIME provides the following cryptographic security services:
  – authentication
  – message integrity
  – non-repudiation of origin (using digital signatures)
  – privacy
  – data security (using encryption).
S/MIME: not restricted to mail

- can be used with any transport mechanism that transports MIME data
- can be used in systems that use cryptographic security services that do not require (or support) any human intervention

© A. Mariën -
Message security system

• Connection security
  – Negotiation phase
    • Protocols to be used
    • Keys to be used
    • Mutual authentication

• Message security
  – Sender provides all elements
Message security: signing

- Actual data to be sent
- Digest of data
  - Which digest algorithm?
- Signature of digest
  - Which signing algorithm?
Message security: additional info

• Also provide response information
  – Add own certificate
  – Signing time
  – Add info on preferred
    • Public key system
    • Key encryption system
    • Symmetric key system
    • Hashing algorithm
PKCS standards

- "PKCS #1: RSA Encryption"
- "PKCS #7: Cryptographic Message Syntax"
- "PKCS #10: Certification Request Syntax"
Message structure: signed

• Single PKCS#7 signedData format
  – MIME type:
    • application/pkcs7-mime;smime-type=signed-data

• Composite message
  – MIME type: multipart/signed
  – Part1: text/plain, application/msword, ...
  – Part2: application/pkcs7-signature

(c) A. Mariën -
PKCS #7 content types:

- **“data”**
  - The data to be secured
- **“signedData”**
  - to apply a digital signature to a message
  - to convey certificates
- **“envelopedData”**
  - To provide confidentiality
  - Does not provide authentication
S/MIME provides

- one format for enveloped-only data
- several formats for signed-only data
- several formats for signed and enveloped data
Preparing the MIME Entity

• Needed for both Signing or Enveloping

• S/MIME: secures MIME entities

• A MIME entity:
  – sub-part(s) of a message
  – whole message with all its sub-parts.

• whole message: does not include the RFC-822 headers (why not?)
Transfer Encoding

• transfer encoding for all MIME entities
• primary reason for the 7-bit requirement: Internet mail transport infrastructure cannot guarantee transport of 8-bit or binary data
application/pkcs7-mime

• MIME type used to carry PKCS #7 objects: envelopedData, signedData

• general characteristics
  – always carries a single PKCS #7 object.
  – must always be BER encoding of the ASN.1 syntax describing the object
Sidenote: BER DER

- BER: Basic Encoding Rules
- DER: Distinguished Encoding Rules
- defined in the ITU-T X.690 standard
- How to convert an ASN.1 object into a byte stream
- Recursive:
  - Type length data
S/Mime type

• The application/pkcs7-mime content type defines the optional "smime-type" parameter.
• The intent of this parameter: to convey details about the security applied (signed or enveloped) along with information about the contained content.
• S/mime-types:
  – enveloped-data: EnvelopedData
  – signed-data: SignedData
  – certs-only: SignedData
Creating an Enveloped-only Message

- the format for enveloping a MIME entity without signing it.

1. The MIME entity to be enveloped is prepared
2. The MIME entity and other required data is processed into a PKCS #7 object of type envelopedData.
3. The PKCS #7 object is inserted into an application/pkcs7-mime MIME entity.
Format for Signed-only Messages

• Choice:
  – multipart/signed format can always be viewed by the receiver whether they have S/MIME software or not
  – signedData format: cannot be viewed by a recipient unless they have S/MIME facilities
PKCS#7: SignedData

• ASN.1 structure for the SignedData content type
  – version
  – digestAlgorithms
  – contentInfo (or empty)
  – certificates (OPTIONAL)
  – crls (OPTIONAL)
  – signerInfos (SET OF): multiple signers
PKCS#7: SignerInfo

- ASN.1 structure for the SignerInfo type
  - version
  - issuerAndSerialNumber
  - digestAlgorithm
  - authenticatedAttributes
  - digestEncryptionAlgorithm
  - encryptedDigest (= actual signature)
  - unauthenticatedAttributes

(c) A. Mariën
multipart/signed: parameters

• two required parameters:
  – the protocol parameter:
    • "application/pkcs7-signature"
  – the micalg (message integrity check algorithm) parameter:
    • md5, sha1, unknown
Example message

Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF4676hIGfHfYT64VQpfyF4676hIGfHfYT6jH77n8HHGghyHhHUUujhJhj756tbB9HGTrfvdjvbnj8HHGTrfvhJhjH776tbB9HG4VQbnj75676hIGfHfYT6ghyHhHUujpfyF476hIGfHfYT64VQbnj756

--boundary42--
BOTH Signing and Encrypting

• signing and enveloping: signed-only - encrypted-only
• may be nested
• options:
  – sign a message first
  – to envelope the message first.
• It is up to the implementer and the user to choose.
Choice of order

• Signing first:
  – the signatories are then securely obscured by the enveloping.

• Enveloping first:
  – the signatories are exposed
  – It is possible to verify signatures without removing the enveloping
Certificate Processing

• A receiving agent MUST provide some certificate retrieval mechanism in order to gain access to certificates for recipients of digital envelopes.
  • Example: LDAP look-up
• Receiving and sending agents SHOULD also provide a mechanism to allow a user to "store and protect" certificates for correspondents in such a way so as to guarantee their later retrieval.
  • Example: certificate stores (why protect?)
References

• RFC 1321: The MD5 Message Digest Algorithm
• RFC 2045: MIME Part 1: Format of Internet Message Bodies
• RFC 2046: MIME Part 2: Media Types
• RFC 1847: Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted
• RFC 2313: PKCS #1: RSA Encryption Version 1.5
References

• RFC 2315: PKCS #7: Cryptographic Message Syntax Version 1.5
• RFC 2314: PKCS #10: Certification Request Syntax Version 1.5