Envelope

Posílání e-mailů intuitivně

Edvard Rejthar • edvard.rejthar@nic.cz • 12. 11. 2020

Spolufinancováno Evropskou unií
Nástroj pro propojení Evropy

CZ.NIC | SPRÁVCE DOMÉNY CZ

Envelope – overview

- email

```
email = EmailMessage()
email["Subject"] = "Encrypted message"  # real subject should be revealed when decrypted
email.set_type("multipart/encrypted")
email.set_param("protocol", "application/pgp-encrypted")
email.set_payload(text)
```

Envelope – overview

- email
- smtplib

```
email = EmailMessage()
email["Subject"] = "Encrypted message"  # real subject should be revealed when decrypted
email.set_type("multipart/encrypted")
email.set_param("protocol", "application/pgp-encrypted")
email.set_payload(text)

smtp = smtplib.SMTP(self.host, self.port)
if self.security == "starttls":
  smtp.starttls()
smtp.login(self.user, self.password)
```

Envelope – overview

- email
- smtplib
- python-gnupg

```
email = EmailMessage()
email["Subject"] = "Encrypted message"  # real subject should be revealed when decrypted
email.set_type("multipart/encrypted")
email.set_param("protocol", "application/pgp-encrypted")
email.set_payload(text)

smtp = smtplib.SMTP(self.host, self.port)
if self.security == "starttls":
  smtp.starttls()
smtp.login(self.user, self.password)

gpg = gnupg.GPG(gnupghome=self._get_gnupg_home(), options=["--trust-model=always"])
gpg.sign(message,
         extra_args=["--textmode"],
         keyid=sign if sign and sign is not True else None,
         passphrase=self._passphrase if self._passphrase else None,
         detach=True if send is not None else None,
        )
```

Envelope – overview

- email
- smtplib
- python-gnupg
- M2Crypto
- magic

```
email = EmailMessage()
email["Subject"] = "Encrypted message"  # real subject should be revealed when decrypted
email.set_type("multipart/encrypted")
email.set_param("protocol", "application/pgp-encrypted")
email.set_payload(text)

smtp = smtplib.SMTP(self.host, self.port)
if self.security == "starttls":
  smtp.starttls()
smtp.login(self.user, self.password)

gpg = gnupg.GPG(gnupghome=self._get_gnupg_home(), options=["--trust-model=always"])
gpg.sign(message,
         extra_args=["--textmode"],
         keyid=sign if sign and sign is not True else None,
         passphrase=self._passphrase if self._passphrase else None,
         detach=True if send is not None else None,
        )
```

Envelope – e-mail structure


```
From: person@example.com
To: person@example.cz
Subject: Personal subject

Message text.
```

Envelope – e-mail structure

MAIL
- headers
- subject
- text

Envelope – PGP inline signed

SIGNED MAIL
headers
subject
text

SIGNATURE
PGP ASCII armoured data

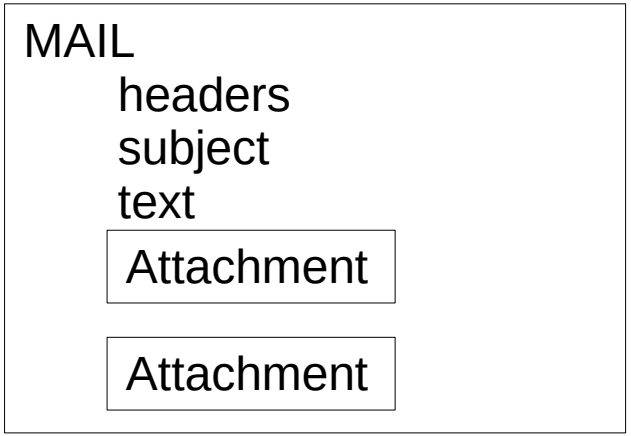
Envelope – PGP inline signed

SIGNED MAIL
- headers
- subject
- text

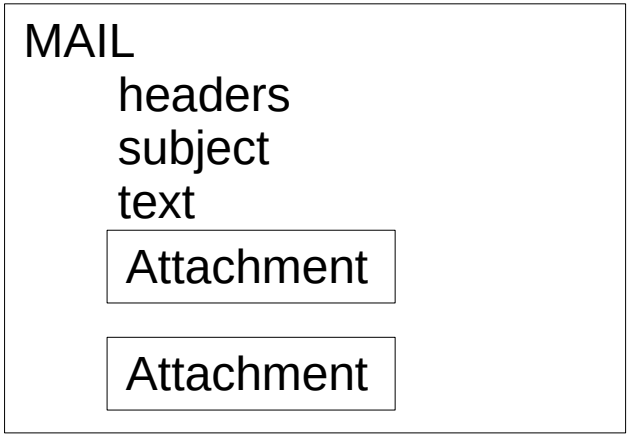
SIGNATURE
PGP ASCII armoured data

```
From: Emil Natrhbuřt <emil.natrhburt@example.com>
To: Nemyl Sešil <nemyl.sesil@example.com>
Subject: Předmět e-mailu
Content-Type: text/plain; charset=us-ascii

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Ahoj,
Tohle je můj podepsaný text.
Emil

-----BEGIN PGP SIGNATURE-----
Version: GnuPG2
QWhvaiBFbWlsZSB0YWR5IGplIG5lamFreSBzb25ldCBPIGhvdyBtdWNoIG1vcmUg
ZG90aCBiZWF1dHkgYmVhdXRlb3VzIHNlZW0gQnkgdGhhdCBzd2VldCBvcm5hbWVu
dCB3aGljaCB0cnV0aCBkb3RoIGdpdmUgVGhlIHJvc2UgbG9va3MgZmFpciBidXQg
-----END PGP SIGNATURE-----
```

Envelope – PGP signed



Envelope – PGP signed



SIGNATURE
PGP ASCII armoured data

Envelope – PGP signed

SIGNED MESSAGE

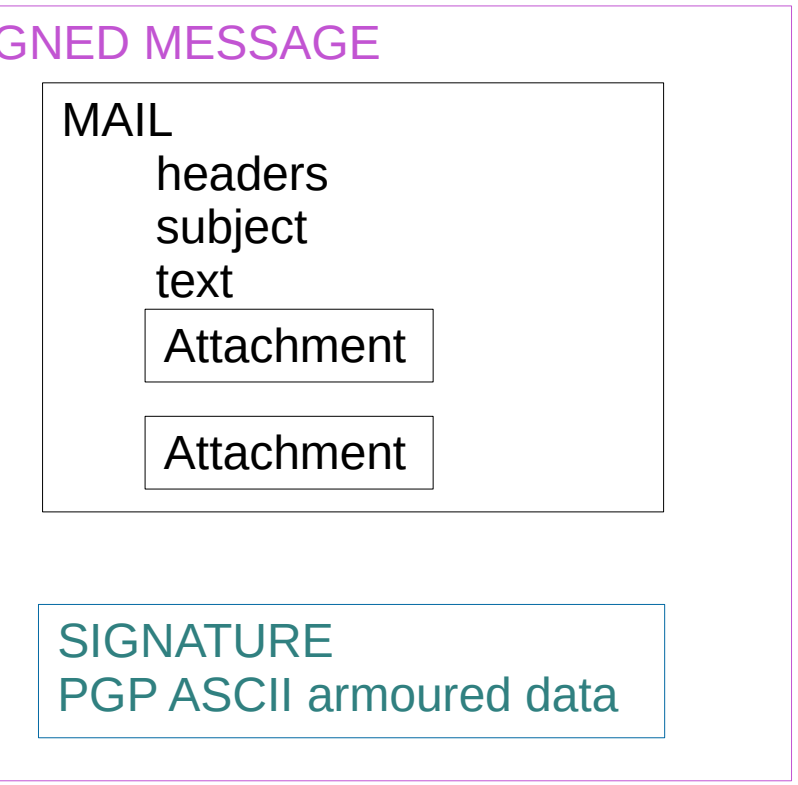
MAIL
- headers
- subject
- text

Attachment

Attachment

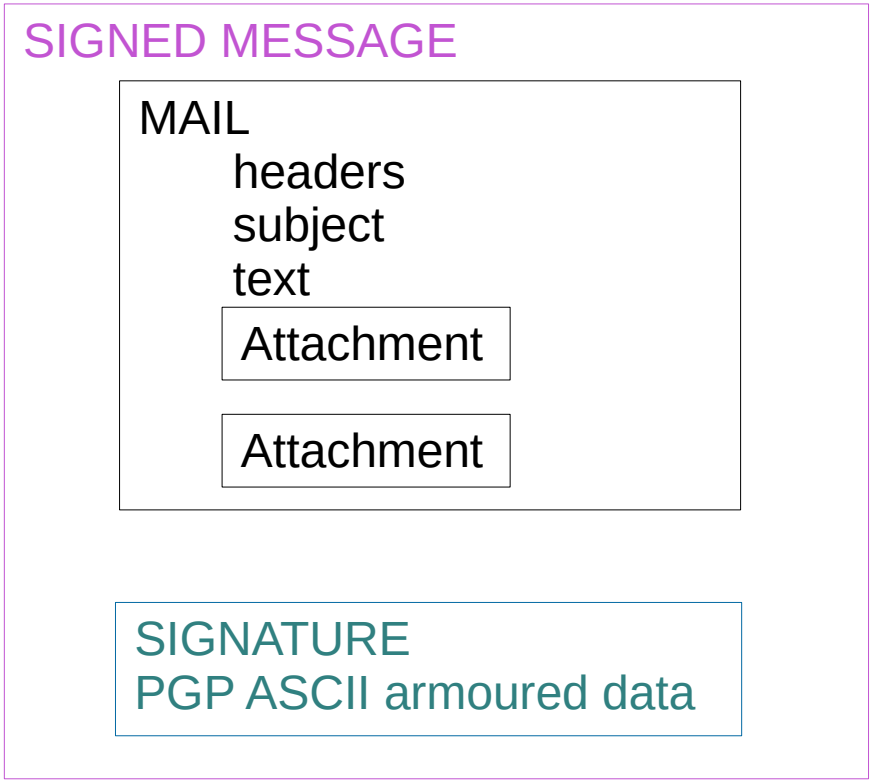
SIGNATURE
PGP ASCII armoured data

Envelope – PGP signed



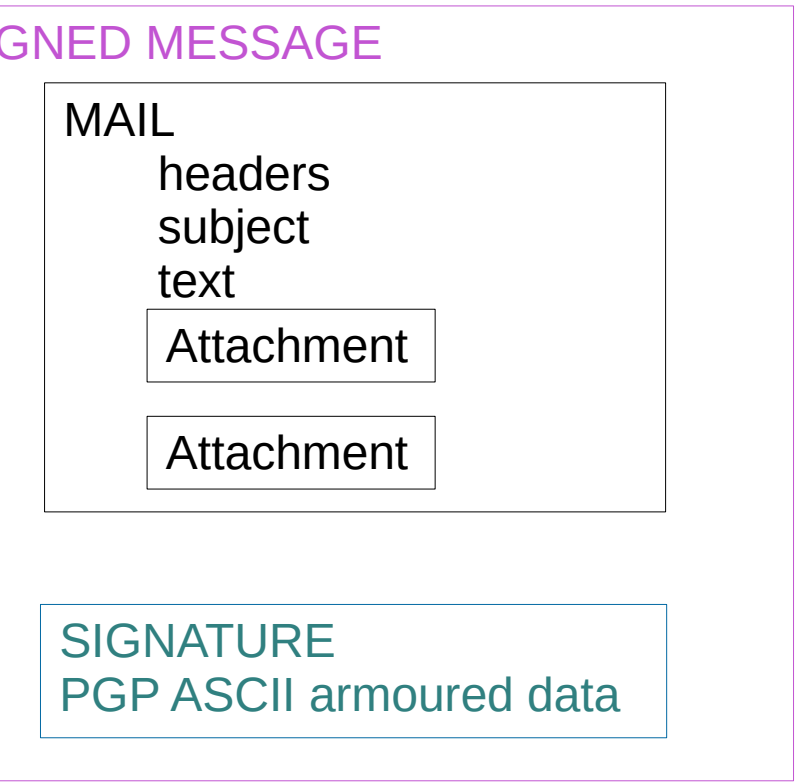

```
From: Emil Natrhbuřt <emil.natrhburt@example.com>
To: Nemyl Sešil <nemyl.sesil@example.com>
Subject: Předmět e-mailu
Content-Type: multipart/signed; protocol="application/pgp-
signature"; micalg="pgp-sha512"; boundary="==ZLOM=="

--==ZLOM==
Content-Type: text/plain

Ahoj,
Tohle je můj podepsaný text.
Emil

--==ZLOM==
Content-Type: application/pgp-signature; name="signature.asc"
Content-Description: OpenPGP digital signature
Content-Disposition: attachment; filename="signature.asc"

----BEGIN PGP SIGNATURE-----
Version: GnuPG2
QWhvaiBFbWlsZSB0YWR5IGplG5lamFreSBzb25ldCBPIGhvdyBtdWNoIG1vcmUg
ZG90aCBiZWF1dHkgYmVhdXRlb3VzIHNlZW0gQnkgdGhhdCBzd2VldCBvcm5hbWVu
dCB3aGljaCB0cnV0aCBkb3RoIGdpdmUgVGhlIHJvc2UgbG9va3MgZmFpciBidXQg
-----END PGP SIGNATURE-----
--==ZLOM==--
```

Envelope – PGP signed



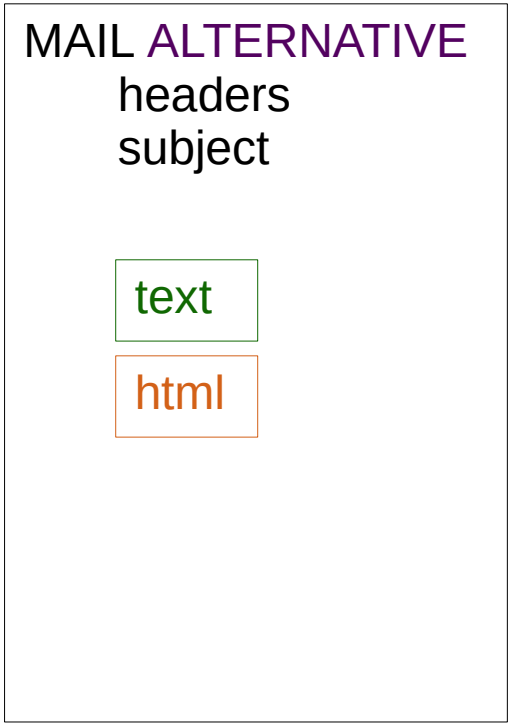

```
From: Emil Natrhbuřt <emil.natrhburt@example.com>
To: Nemyl Sešil <nemyl.sesil@example.com>
Subject: Předmět e-mailu
Content-Type: multipart/signed; protocol="application/pgp-
signature"; micalg="pgp-sha512"; boundary="==ZLOM=="

--==ZLOM==
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: base64

QWhvaiwgVG9obGUgamUgbcWvaiBwb2RlcHNhbsO9IHRleHQuIEVtaWwK


--==ZLOM==
Content-Type: application/pgp-signature; name="signature.asc"
Content-Description: OpenPGP digital signature
Content-Disposition: attachment; filename="signature.asc"

----BEGIN PGP SIGNATURE-----
Version: GnuPG2
QWhvaiBFbWlsZSB0YWR5IGplG5lamFreSBzb25ldCBPIGhvdyBtdWNoIG1vcmUg
ZG90aCBiZWF1dHkgYmVhdXRlb3VzIHNlZW0gQnkgdGhhdCBzd2VldCBvcm5hbWVu
dCB3aGljaCB0cnV0aCBkb3RoIGdpdmUgVGhlIHJvc2UgbG9va3MgZmFpciBidXQg
-----END PGP SIGNATURE-----
--==ZLOM==--
```

Envelope – PGP signed

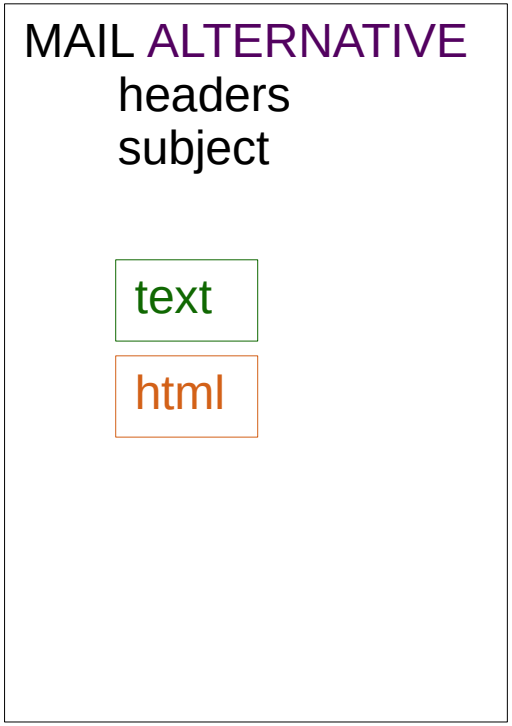



```
From: Emil Natrhbuřt <emil.natrhburt@example.com>
To: Nemyl Sešil <nemyl.sesil@example.com>
Subject: Předmět e-mailu
Content-Type: multipart/signed; protocol="application/pgp-
signature"; micalg="pgp-sha512"; boundary="==ZLOM=="

--==ZLOM==
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable

Ahoj,
Tohle je m=C5=AFj podepsan=C3=BD text.
Emil
--==ZLOM==
Content-Type: application/pgp-signature; name="signature.asc"
Content-Description: OpenPGP digital signature
Content-Disposition: attachment; filename="signature.asc"

----BEGIN PGP SIGNATURE-----
Version: GnuPG2
QWhvaiBFbWlsZSB0YWR5IGplG5lamFreSBzb25ldCBPIGhvdyBtdWNoIG1vcmUg
ZG90aCBiZWF1dHkgYmVhdXRlb3VzIHNlZW0gQnkgdGhhdCBzd2VldCBvcm5hbWVu
dCB3aGljaCB0cnV0aCBkb3RoIGdpdmUgVGhlIHJvc2UgbG9va3MgZmFpciBidXQg
-----END PGP SIGNATURE-----
--==ZLOM==--
```

Envelope – plain text alternative

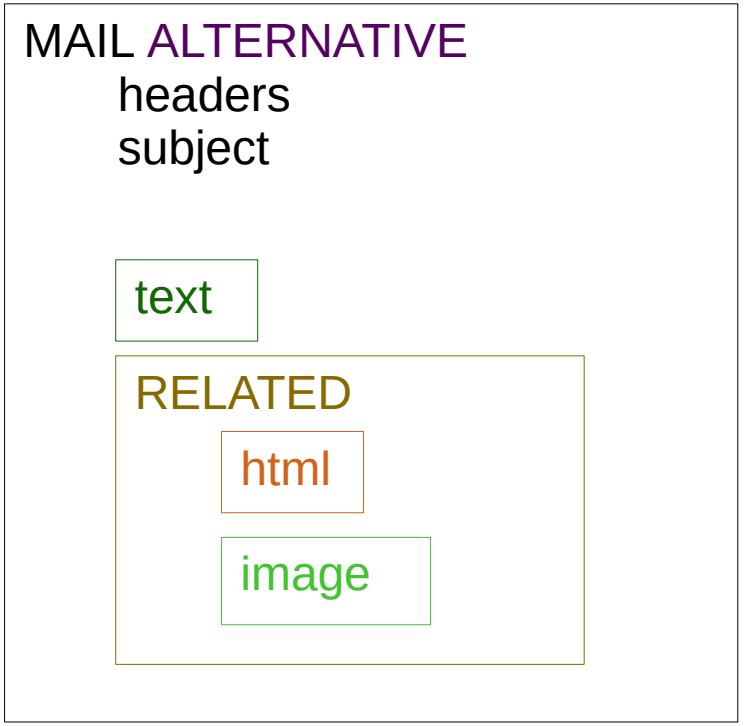


Envelope – plain text alternative



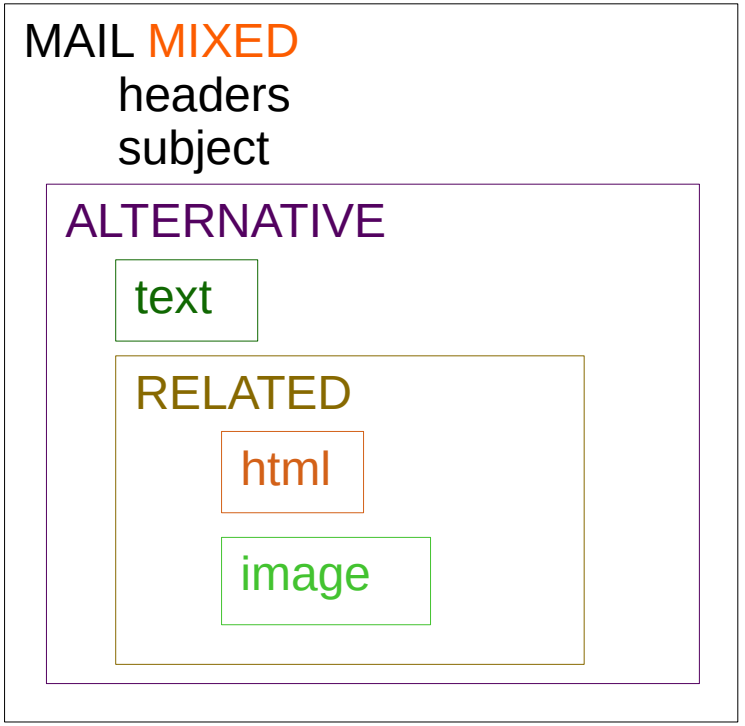

```
From: Emil Natrhbuřt <emil.natrhburt@example.com>
To: Nemyl Sešil <nemyl.sesil@example.com>
Subject: Předmět e-mailu
Content-Type: multipart/alternative; boundary="==ZLOM=="

--==ZLOM==
Content-Type: text/plain

Ahoj,
Tohle je můj nepodepsaný text.
Emil

--==ZLOM==
Content-Type: text/html

Ahoj,
Tohle je můj <b>nepodepsaný</b> text.
<strike>kamarád</strike> Emil

--==ZLOM==--
```

Envelope – inline images

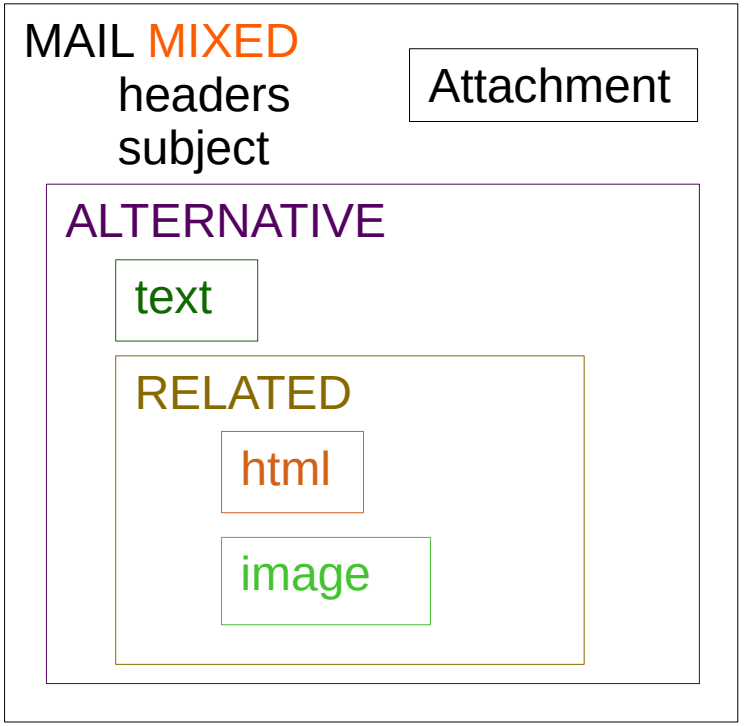



```
--==ZLOM==
Content-Type: multipart/related; boundary="==CHRUP=="

Ahoj,
Tohle je můj nepodepsaný text.
Emil

--==CHRUP==
Content-Type: text/html

Ahoj,
Tohle je můj <img src="cid:pruduch.png" />
<strike>kamarád</strike> Emil

--==CHRUP==
Content-Type: text/plain
Content-Transfer-Encoding: base64
Content-ID: <pruduch.png>
Content-Disposition: inline

L2hvbWUvZWR2YXJkL2VkdmFyZC93d3cvZW52ZWxvcGUvdGVzdHMvZW1sL2ltYWdl
LmdpZg==

--==CHRUP==--

--==ZLOM==
```

Envelope – inline images, plain alternative...

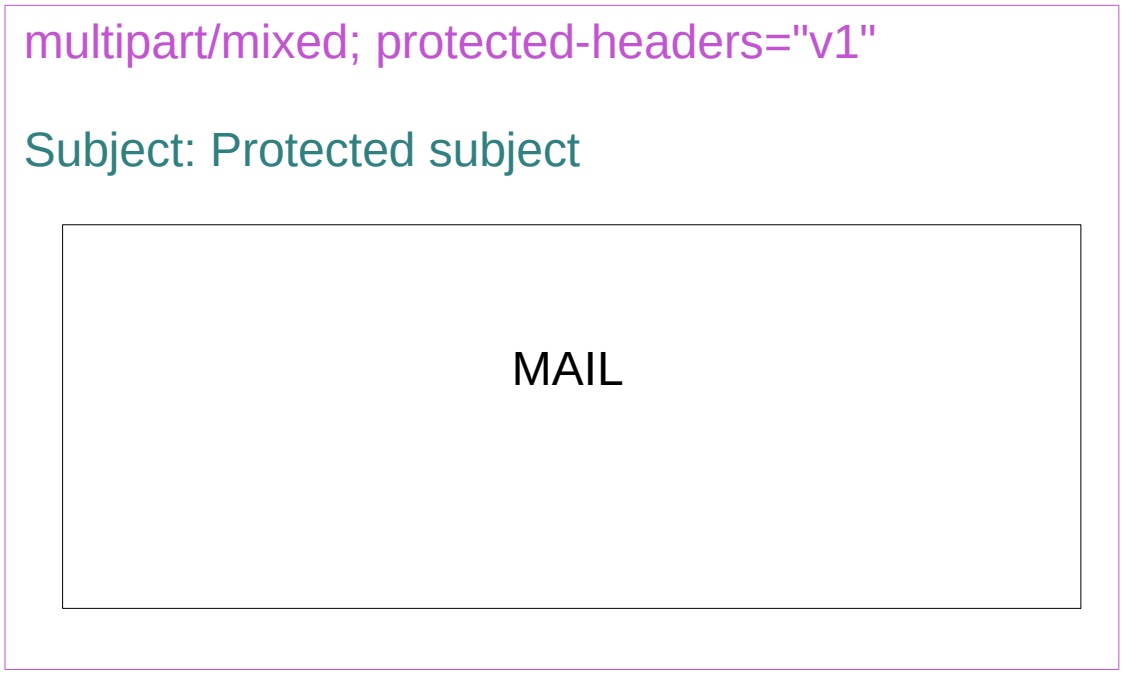


Envelope – … and attachments

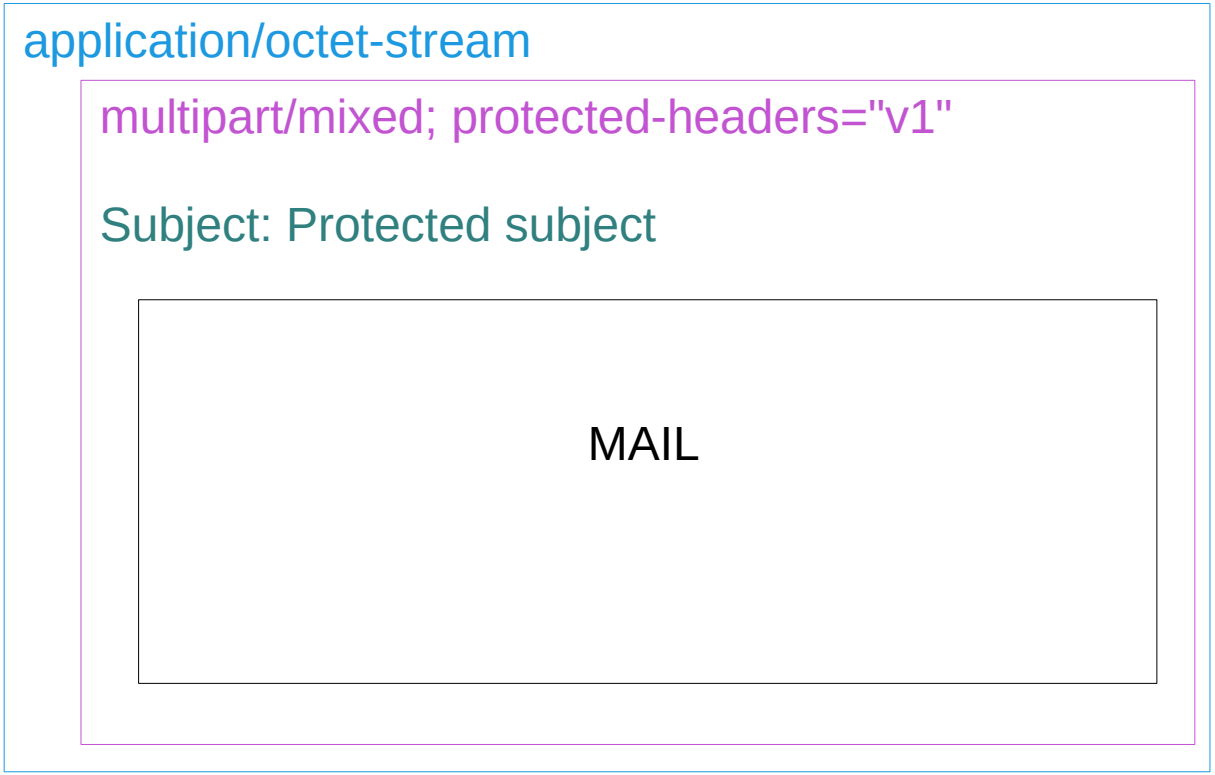


Envelope – sign and encrypt?

Envelope – e-mail structure (RFC-3156)



Envelope – e-mail structure (RFC-3156)



Envelope – e-mail structure (RFC-3156)

application/octet-stream

multipart/mixed; protected-headers="v1"

Subject: Protected subject

MAIL

application/pgp-encrypted

Version: 1

Envelope – PGP encrypted

multipart/encrypted

application/pgp-encrypted

Version: 1

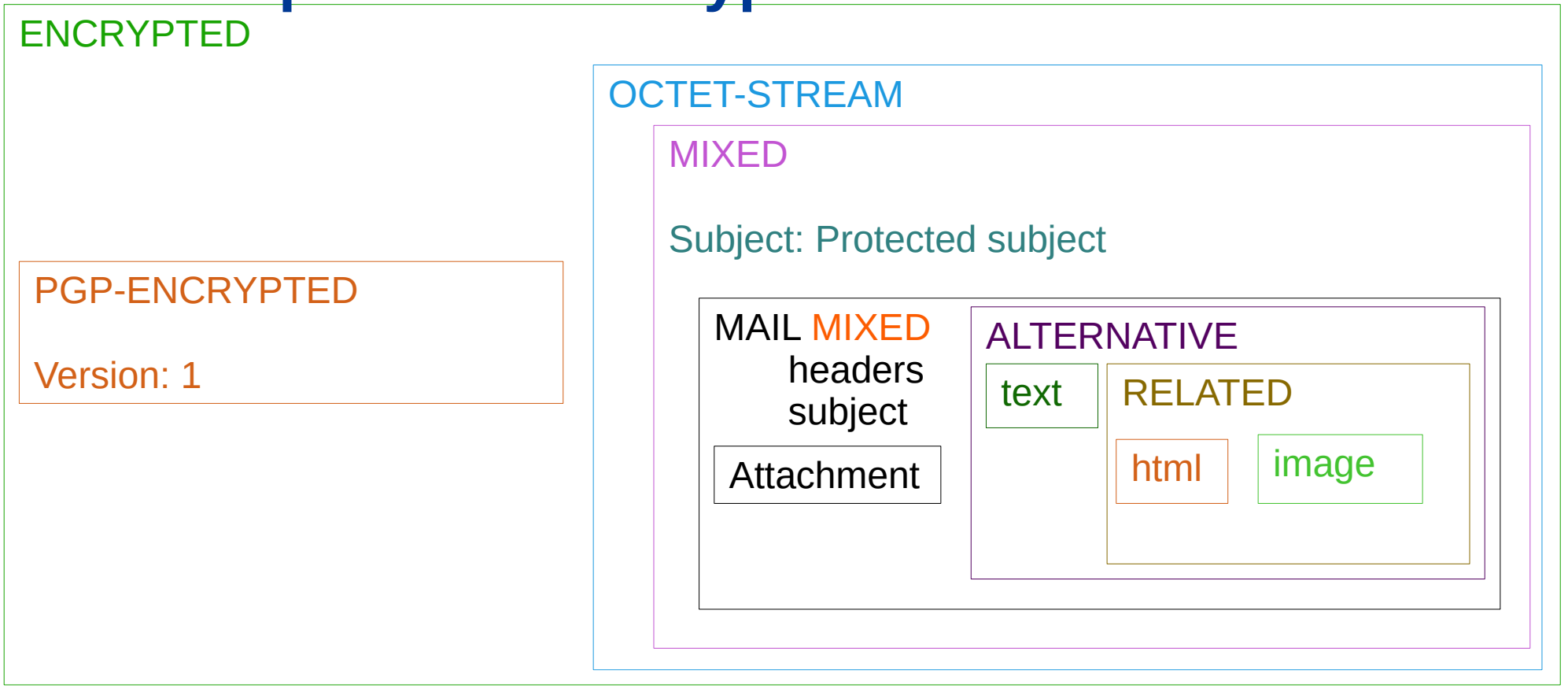
application/octet-stream

multipart/mixed; protected-headers="v1"

Subject: Protected subject

MAIL

Envelope – PGP encrypted



Envelope – interfaces

- One-liner
- Fluent
- CLI

```
Envelope("my message")
    .subject("hello world")
    .to("example@example.com")
    .attach(file_contents, filename="attached-file.txt")
    .smtp("localhost", 587, "user", "pass", "starttls")
    .signature()
    .send()
```

Envelope – interfaces

FLUENT

```
from envelope import Envelope
Envelope().message("Hello world")
    .output("/tmp/output_file")
    .sender("me@example.com")
    .to("remote_person@example.com")
    .encrypt(key_path="/tmp/remote_key.asc")
```

Envelope – interfaces

FLUENT

```
from envelope import Envelope
Envelope().message("Hello world")
    .output("/tmp/output_file")
    .sender("me@example.com")
    .to("remote_person@example.com")
    .encrypt(key_path="/tmp/remote_key.asc")
```

ONE-LINER

```
from envelope import Envelope
Envelope(message="Hello world",
         output="/tmp/output_file",
         sender="me@example.com",
         to="remote_person@example.com",
         encrypt="/tmp/remote_key.asc")
```

Envelope – interfaces

FLUENT

```
from envelope import Envelope
Envelope().message("Hello world")
    .output("/tmp/output_file")
    .sender("me@example.com")
    .to("remote_person@example.com")
    .encrypt(key_path="/tmp/remote_key.asc")
```

```
envelope --message "Hello world"
         --output "/tmp/output_file"
         --sender "me@example.com"
         --to "remote_person@example.com"
         --encrypt-path "/tmp/remote_key.asc"
```

CLI

```
from envelope import Envelope
Envelope(message="Hello world",
         output="/tmp/output_file",
         sender="me@example.com",
         to="remote_person@example.com",
         encrypt="/tmp/remote_key.asc")
```

ONE-LINER

Envelope – e-mail addresses

```
Envelope
    .to("Person Erson <person@example.com>")
```

Envelope – e-mail addresses

```
Envelope
    .to("Person Erson <person@example.com>")
    .to()
```

Envelope – e-mail addresses

```
a = Envelope
        .to("Person Erson <person@example.com>")
        .to()[0]
```

Envelope – e-mail addresses

```
a = Envelope
        .to("Person Erson <person@example.com>")
        .to()[0]



print(a)  # Person Erson <person@example.com>
```

Envelope – e-mail addresses

```
a = Envelope
        .to("Person Erson <person@example.com>")
        .to()[0]



isinstance(a, str)  # True
type(a)  # envelope.utils.Address
print(a)  # Person Erson <person@example.com>
a.address  # 'person@example.com'
a.name  # 'Person Erson'
```

Envelope – e-mail addresses

```
Envelope()
        .to("person1@example.com")
        .to("person1@example.com, John <person2@example.com>")
        .to(["person3@example.com"])
        .to()  # ["person1@example.com", "John <person2@example.com>",
"person3@example.com"]
```

Envelope – e-mail addresses

```
Envelope()
        .to("person1@example.com")
        .to("person1@example.com, John <person2@example.com>")
        .to(["person3@example.com"])
        .to()  # ["person1@example.com", "John <person2@example.com>",
"person3@example.com"]

[x.get(address=False) for x in e.to()]  # ["", "John", ""]

[x.get(name=True) for x in e.to()]      # ["person1@example.com",
"John", "person3@example.com"]
                                        # return an address if no name
given
[x.get(address=True) for x in e.to()]   # ["person1@example.com",
"person2@example.com", "person3@example.com"]
                                        # addresses only
```

Envelope – e-mail addresses

```
Envelope()
        .to("person1@example.com")
        .to("person1@example.com, John <person2@example.com>")
        .to(["person3@example.com"])
        .to()  # ["person1@example.com", "John <person2@example.com>",
"person3@example.com"]

[x.get(address=False) for x in e.to()]  # ["", "John", ""]

[x.get(name=True) for x in e.to()]      # ["person1@example.com",
"John", "person3@example.com"]
                                        # return an address if no name
given
[x.get(address=True) for x in e.to()]   # ["person1@example.com",
"person2@example.com", "person3@example.com"]
                                        # addresses only
```

Envelope – e-mail addresses

```
Envelope()
        .to("person1@example.com")
        .to("person1@example.com, John <person2@example.com>")
        .to(["person3@example.com"])
        .to()  # ["person1@example.com", "John <person2@example.com>",
"person3@example.com"]

[x.get(address=False) for x in e.to()]  # ["", "John", ""]

[x.get(name=True) for x in e.to()]      # ["person1@example.com",
"John", "person3@example.com"]
                                        # return an address if no name
given
[x.get(address=True) for x in e.to()]   # ["person1@example.com",
"person2@example.com", "person3@example.com"]
                                        # addresses only
```

Envelope – any attainable contents

```
e = Envelope()
e.message("text of the message")
```

CZ.NIC | SPRÁVCE DOMÉNY CZ

Envelope – any attainable contents

- Text
- Bytes
- Stream (ex: from open())
- pathlib.Path
- CLI flags

```
e = Envelope()
e.message("text of the message")
```

Envelope – any attainable contents

- Text
- Bytes
- Stream (ex: from open())
- pathlib.Path
- CLI flags

```
e = Envelope()
e.message("text of the message")

with open("/tmp/file.txt") as f:
    e.message(f)
```

Envelope – any attainable contents

- Text
- Bytes
- Stream (ex: from open())
- pathlib.Path
- CLI flags

```
e = Envelope()
e.message("text of the message")

with open("/tmp/file.txt") as f:
    e.message(f)

e.message(Path("/tmp/file.txt"))
```

Envelope – any attainable contents

- Text
- Bytes
- Stream (ex: from open())
- pathlib.Path
- CLI flags

```
e = Envelope()
e.message("text of the message")

with open("/tmp/file.txt") as f:
    e.message(f)

e.message(Path("/tmp/file.txt"))

e.message(path="/tmp/file.txt")
```

Envelope – any attainable contents

- Text
- Bytes
- Stream (ex: from open())
- pathlib.Path
- CLI flags

```
e = Envelope()
e.message("text of the message")

with open("/tmp/file.txt") as f:
    e.message(f)

e.message(Path("/tmp/file.txt"))

e.message(path="/tmp/file.txt")

e.message("He<b>llo</b>")
e.message("Hello", alternative="plain")
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
Envelope()
    .attach(path="/tmp/file.txt")
    .attach(path="/tmp/another-file.txt")
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
Envelope()
    .attach(path="/tmp/file.txt")
    .attach(path="/tmp/another-file.txt")

e.attach("text of the attachment")

with open("/tmp/file") as f:
    e.attach(f)

e.attach(Path("/tmp/file"))
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
Envelope()
    .attach(path="img.png", name="me.png")
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
Envelope()
    .attach(path="img.png", name=“me.png“,
inline=True)
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
Envelope()
    .attach(path="img.png", inline=“me.png“)
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
Envelope()
    .attach(path="img.png", inline=“me.png“)
    .message("<img src='cid:me.png' />")
```

Envelope – attachments

```
def attach(self, attachment=None, mimetype=None, name=None, inline=None, *, path=None)
```

```
MIME-Version: 1.0
Content-Type: multipart/related
Subject:
Date: Thu, 12 Nov 2020 14:48:58 +0100

--==ZLOM==
Content-Type: text/html; charset="utf-8"
Content-Transfer-Encoding:

<img src=‘cid:me.png‘ />

--==ZLOM==
Content-Type: image/png
Content-Transfer-Encoding: base64
Content-ID: <me.png>
MIME-Version: 1.0
Content-Disposition: inline

R0lGODlhAwADAKEDAAIJAvz9
--==ZLOM==--
```

```
print(Envelope()
    .attach(path="img.png", inline=“me.png“)
    .message("<img src='cid:me.png' />"))
```

Envelope – load

```
def attachments(self, name=None, inline=None)
```

```
Envelope.load(path="message.eml")
    .attachments()
```

Envelope – load

```
Envelope.load(path="message.eml")
    .recipients()
```

Envelope – load

```
Envelope.load("Subject: testing message").subject()
```

Envelope – load

```
Envelope.load("Subject: testing message").subject()
```

```
$ echo "Subject: testing message" | envelope --subject
```

```
$ envelope --load message.eml --subject
```

Envelope – load

```
Envelope.load(path="tests/eml/smime_encrypt.eml",
    key='key.pem', cert='cert.pem')
```

Envelope – related affairs

- SMTP
- GPG, S/MIME
- SPF, DKIM, DMARC
- check

```
$ docker run --network=host --restart always -d bytemark/smtp
        # starts open port 25 on localhost
```

```
$ envelope --message "SMTP test" --from [your e-mail] --to [your e-mail]
        --smtp localhost 25 --send
```

Envelope – related affairs

- SMTP
- GPG, S/MIME
- SPF, DKIM, DMARC
- check

```
$ ls -l $(tty)  # see current TTY owner
$ sudo chown www-data $(tty)  # if creating the key for a different user and generation fails (...)
$ GNUPGHOME=/var/www/.gnupg sudo -H -u www-data gpg --full-generate-key
$ GNUPGHOME=/var/www/.gnupg sudo -H -u www-data gpg --list-secret-keys
$ GNUPGHOME=/var/www/.gnupg sudo -H -u www-data gpg --send-keys [key ID] # now the world is able to pull the key (…)
(...)
```

Envelope – related affairs

- SMTP
- GPG, S/MIME
- SPF, DKIM, DMARC
- check

```
# Either: Do you have private key?
$ openssl req -key YOUR-KEY.pem -nodes -x509 -days 365 -out
certificate.pem  # will generate privkey.pem alongside

# Or: Do not you have private key?
$ openssl req -newkey rsa:1024 -nodes -x509 -days 365 -out
certificate.pem  # will generate privkey.pem alongside
```

Envelope – related affairs

- SMTP
- GPG, S/MIME
- SPF, DKIM, DMARC
- check

```
# Check your domain on DKIM:

$ dig -t TXT [selector]._domainkey.example.com

# You can obtain the selector from an e-mail message you received.
(...)
```

Envelope – related affairs

- SMTP
- GPG, S/MIME
- SPF, DKIM, DMARC
- check

```
$ envelope --smtp localhost 25 --sender me@example.com --check
SPF found on the domain example.com: v=spf1 -all
See: dig -t SPF example.com && dig -t TXT example.com
DKIM found: ['v=DKIM1; g=*; k=rsa; p=...']
Could not spot DMARC.
Trying to connect to the SMTP...
Check succeeded.
```

Envelope – test before send

```
$ envelope --to "user@example.org" --message "Hello world" --send 0
********************************************************************************
Have not been sent from  to user@example.org

Content-Type: text/html; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
Subject:
From:
To: user@example.org
Date: Mon, 07 Oct 2019 16:13:37 +0200
Message-ID: <157045761791.29779.5279828659897745855@...>

Hello world
```

Thanks

https://github.com/CZ-NIC/envelope

pip3 install envelope

Edvard Rejthar • edvard.rejthar@nic.cz

Spolufinancováno Evropskou unií
Nástroj pro propojení Evropy

CZ.NIC | SPRÁVCE DOMÉNY CZ