Post-quantum OpenPGP with Claws Mail on Fedora 45 KDE

Stavros Kousidis

Published 2026-09-28
Tested on Fedora 45 KDE Plasma · Claws Mail 4.4.0 · claws-mail-plugins-pgp 4.4.0 · GPGME 2.0.1-6.rfc9980.1.fc45 · Sequoia Chameleon GnuPG 0.13.1

This guide covers the Claws Mail-specific part of the Fedora 45 RFC 9980 proof of concept. Complete the shared setup first, including the patched GPGME installation, Chameleon command selection, and disposable test certificate import and authorization.

Claws Mail uses its PGP plugins through GPGME:

```
Claws Mail
  -> PGP/Core + PGP/MIME
  -> patched GPGME
  -> Sequoia Chameleon
  -> Sequoia OpenPGP / Keystore
```

Unlike Evolution, Claws Mail is therefore part of the GPGME-based test path.

Install Claws Mail and the PGP plugins

Install the packages on Fedora 45 KDE:

```
sudo dnf install -y \
  claws-mail \
  claws-mail-plugins-pgp
```

Verify the installed package versions:

```
rpm -q claws-mail claws-mail-plugins-pgp gpgme
```

The tested baseline is:

```
claws-mail-4.4.0-8.fc45.x86_64
claws-mail-plugins-pgp-4.4.0-8.fc45.x86_64
gpgme-2.0.1-6.rfc9980.1.fc45.x86_64
```

Start Claws Mail and verify the runtime stack

Use the same terminal in which you set the compatibility `PATH` in the shared setup. Close any existing Claws Mail instance before continuing.

Confirm the selected OpenPGP command and patched GPGME package:

```
command -v gpg
gpgconf --list-components | grep -E '^(gpg:|gpgsm:)'
rpm -q gpgme
```

Start Claws Mail once from this terminal and keep this instance open for the remaining steps:

```
claws-mail >/tmp/rfc9980-claws.log 2>&1 &
CLAWS_PID=$!
sleep 5
kill -0 "$CLAWS_PID"
```

If the last command fails, inspect `/tmp/rfc9980-claws.log` and check whether an earlier Claws Mail instance was still running.

Complete the first-run wizard

On a fresh Claws Mail installation, complete the setup wizard before loading the PGP plugins.

On **About You**, enter:

```
Your name:          RFC9980 PoC
Your email address: rfc9980-poc@example.invalid
Organization:       [leave blank]
```

On **Receiving mail**, select **Local mbox file**. Leave the automatically selected local mailbox unchanged. Normally this is `/var/mail/<username>`, the local system mailbox for the current user. This test only queues outgoing messages, so receiving mail configuration is minimal.

On **Sending mail**, use:

```
SMTP server address: localhost
Use authentication:  off
Use TLS:             off
```

No working SMTP server is required; the goal is to let Claws construct and queue the complete MIME message locally.

On **Saving mail on disk**, leave the mailbox name as:

```
Mail
```

This creates the local MH mailbox at `$HOME/Mail`, which the verification steps use for the queue. Finish the wizard.

Load the PGP plugins

Open **Configuration → Plugins** and load:

```
PGP/Core
PGP/MIME
```

Then inspect the libraries loaded by the Claws Mail process:

```
grep -E \
  'pgpcore\.so|pgpmime\.so|libgpgme\.so' \
  "/proc/$CLAWS_PID/maps" \
  | awk '{print $6}' \
  | sort -u
```

The tested runtime loaded:

```
/usr/lib64/claws-mail/plugins/pgpcore.so
/usr/lib64/claws-mail/plugins/pgpmime.so
/usr/lib64/libgpgme.so.45.0.1
```

Together with the package and `gpgconf` checks, this establishes the GPGME-based Chameleon path used by the test.

Configure the test account

Open **Configuration → Edit accounts...**. The account created by the first-run wizard may not yet display as **RFC9980 PoC**.

On **Basic**, use:

```
Name of account:    RFC9980 PoC
Set as default:     enabled
Full name:          RFC9980 PoC
Mail address:       rfc9980-poc@example.invalid
Protocol:           Local mbox file
Local mailbox:      [keep the wizard-created value]
SMTP server (send): localhost
```

The local mailbox is normally `/var/mail/<username>`; keep the value created by the wizard.

On **Privacy**, use:

```
Default privacy system:                                    PGP/MIME
Always sign messages:                                      disabled
Always encrypt messages:                                   disabled
Always sign messages when replying to a signed message:  enabled
Always encrypt messages when replying to an encrypted message: enabled
Encrypt sent messages with your own key in addition to recipient's: disabled
Save sent encrypted messages as clear text:              disabled
```

Leaving **Always sign messages** and **Always encrypt messages** disabled allows you to create the unsigned, signed-only, and encrypted-and-signed variants separately.

Under **Plugins → GPG**, in **Sign key**, select **Select key by your email address**. Do not select **Use default GnuPG key** or **Specify key manually** for this test.

The **Plugins → S/MIME** page requires no changes for this OpenPGP test.

Choose **OK** to save the account settings, then close **Edit accounts...**.

Create and locate the three messages

Create three messages to:

```
rfc9980-poc@example.invalid
```

Configure and queue them as follows:

```
Subject             PGP/MIME signing   PGP/MIME encryption
Unsigned            off                off
Signed only         on                 off
Encrypted + signed  on                 on
```

Use the compose window's PGP/MIME signing and encryption controls for each variant. **Queue** each message; do not send it. The verification steps inspect the locally queued messages.

The tested configuration used Claws Mail's local MH mailbox at `$HOME/Mail`. If your account uses a different location, adjust accordingly.

For the tested layout, set:

```
MAILBOX="$HOME/Mail"
QUEUE="$MAILBOX/queue"
test -d "$QUEUE"
```

List the queue files in modification-time order:

```
find "$QUEUE" -maxdepth 1 -type f -printf '%T@ %p\n' \
  | sort -nr
```

Do not assume numeric queue filenames. Match the test messages by their subjects and Claws-specific headers:

```
find "$QUEUE" -maxdepth 1 -type f -print0 |
while IFS= read -r -d '' MESSAGE; do
  printf '\n=== %s ===\n' "$MESSAGE"
  grep -E '^(Subject:|X-Claws-Sign:|X-Claws-Encrypt:)' "$MESSAGE"
done
```

For the three messages created above, identify the entries with subjects `Unsigned`, `Signed only`, and `Encrypted + signed`. The unsigned queue file has `X-Claws-Sign:0`; the signed-only queue file has `X-Claws-Sign:1` and `X-Claws-Encrypt:0`; the encrypted-and-signed queue file has both set to `1`.

Record the full paths of the signed-only and encrypted-and-signed queue files and create one verification directory:

```
SIGNED="SIGNED_QUEUE_FILE"
ENCRYPTED="ENCRYPTED_QUEUE_FILE"
WORK="$HOME/rfc9980-mail-test/claws-verify"
export SIGNED ENCRYPTED WORK
```

```
rm -rf "$WORK"
mkdir -p "$WORK"
```

Claws queue files contain private queue headers before the RFC 5322/MIME message. Strip those headers before MIME parsing:

```
python3 - <<'PY'
from pathlib import Path
import os

work = Path(os.environ["WORK"])
for name, env in [("signed.eml", "SIGNED"), ("encrypted.eml", "ENCRYPTED")]:
    raw = Path(os.environ[env]).read_bytes()
    marker = b"X-Claws-End-Special-Headers: 1"
    pos = raw.find(marker)
    assert pos >= 0, f"{env}: Claws special-header terminator not found"
    pos = raw.find(b"\n", pos)
    assert pos >= 0, f"{env}: malformed queue headers"
    work.joinpath(name).write_bytes(raw[pos + 1:].lstrip(b"\r\n"))
PY
```

Confirm the resulting MIME types:

```
grep -m1 '^Content-Type:' "$WORK/signed.eml"
grep -m1 '^Content-Type:' "$WORK/encrypted.eml"
```

The expected outer types are `multipart/signed` and `multipart/encrypted` respectively.

Independent verification

Extract and verify the signed-only PGP/MIME message:

```
python3 - <<'PY'
from email import policy
from email.parser import BytesParser
from pathlib import Path
import os

work = Path(os.environ["WORK"])
m = BytesParser(policy=policy.SMTP).parsebytes(work.joinpath("signed.eml").read_bytes())
parts = list(m.iter_parts())
assert len(parts) >= 2, "Signed message does not have expected MIME parts"
work.joinpath("signed-content.txt").write_bytes(parts[0].as_bytes(policy=policy.SMTP))
work.joinpath("signed-signature.asc").write_bytes(parts[1].get_payload(decode=True))
PY

gpg --list-packets "$WORK/signed-signature.asc"

gpg --status-fd=1 \
  --verify "$WORK/signed-signature.asc" "$WORK/signed-content.txt"
```

Confirm a signature packet using algorithm `30`, followed by `GOODSIG` and `VALIDSIG ... 30 ...` with exit status `0`.

Extract the encrypted OpenPGP payload:

```
python3 - <<'PY'
from email import policy
from email.parser import BytesParser
from pathlib import Path
import os

work = Path(os.environ["WORK"])
```

```
m = BytesParser(policy=policy.SMTP).parsebytes(work.joinpath("encrypted.eml").read_bytes())
parts = list(m.iter_parts())
assert len(parts) >= 2, "Encrypted message does not have expected MIME parts"
work.joinpath("encrypted.pgp").write_bytes(parts[1].get_payload(decode=True))
PY

gpg --list-packets "$WORK/encrypted.pgp"

gpg --status-fd=1 \
  --output "$WORK/decrypted.eml" \
  --decrypt "$WORK/encrypted.pgp"
```

Confirm a version-6 public-key encrypted session key packet using algorithm `35` and a successful `DECRYPTION_OKAY`.

The decrypted MIME entity is signed. Extract and verify its inner signature:

```
python3 - <<'PY'
from email import policy
from email.parser import BytesParser
from pathlib import Path
import os

work = Path(os.environ["WORK"])
m = BytesParser(policy=policy.SMTP).parsebytes(work.joinpath("decrypted.eml").read_bytes())
parts = list(m.iter_parts())
assert len(parts) >= 2, "Decrypted message is not multipart/signed"
work.joinpath("inner-content.txt").write_bytes(parts[0].as_bytes(policy=policy.SMTP))
work.joinpath("inner-signature.asc").write_bytes(parts[1].get_payload(decode=True))
PY

gpg --list-packets "$WORK/inner-signature.asc"

gpg --status-fd=1 \
  --verify "$WORK/inner-signature.asc" "$WORK/inner-content.txt"
```

Confirm another algorithm-`30` signature with `GOODSIG`, `VALIDSIG`, and exit status `0`.

Result

The procedure exercises this path:

```
Claws Mail
  -> PGP/Core + PGP/MIME
  -> patched GPGME
  -> Sequoia Chameleon
  -> Sequoia OpenPGP / Keystore
```

with:

```
Signing:
  ML-DSA-65+Ed25519
  OpenPGP algorithm 30
  independent verification successful

Encryption:
  ML-KEM-768+X25519
  OpenPGP algorithm 35
  version-6 PKESK
  independent decryption successful
  inner signature independently valid
```

This proof of concept requires no Claws Mail source patch.