Automated use of GnuPG through GPGME

Andre Heinecke

OpenPGP-Conf Cologne – 9.9.16

Outline

- What is the Problem?
- Advantages
- Concepts and Usage
- Disadvantages
- Language Bindings
- Questions

What is the Problem?

- GnuPG is a tool not a library
- Changes to machine interface break things
- Example:

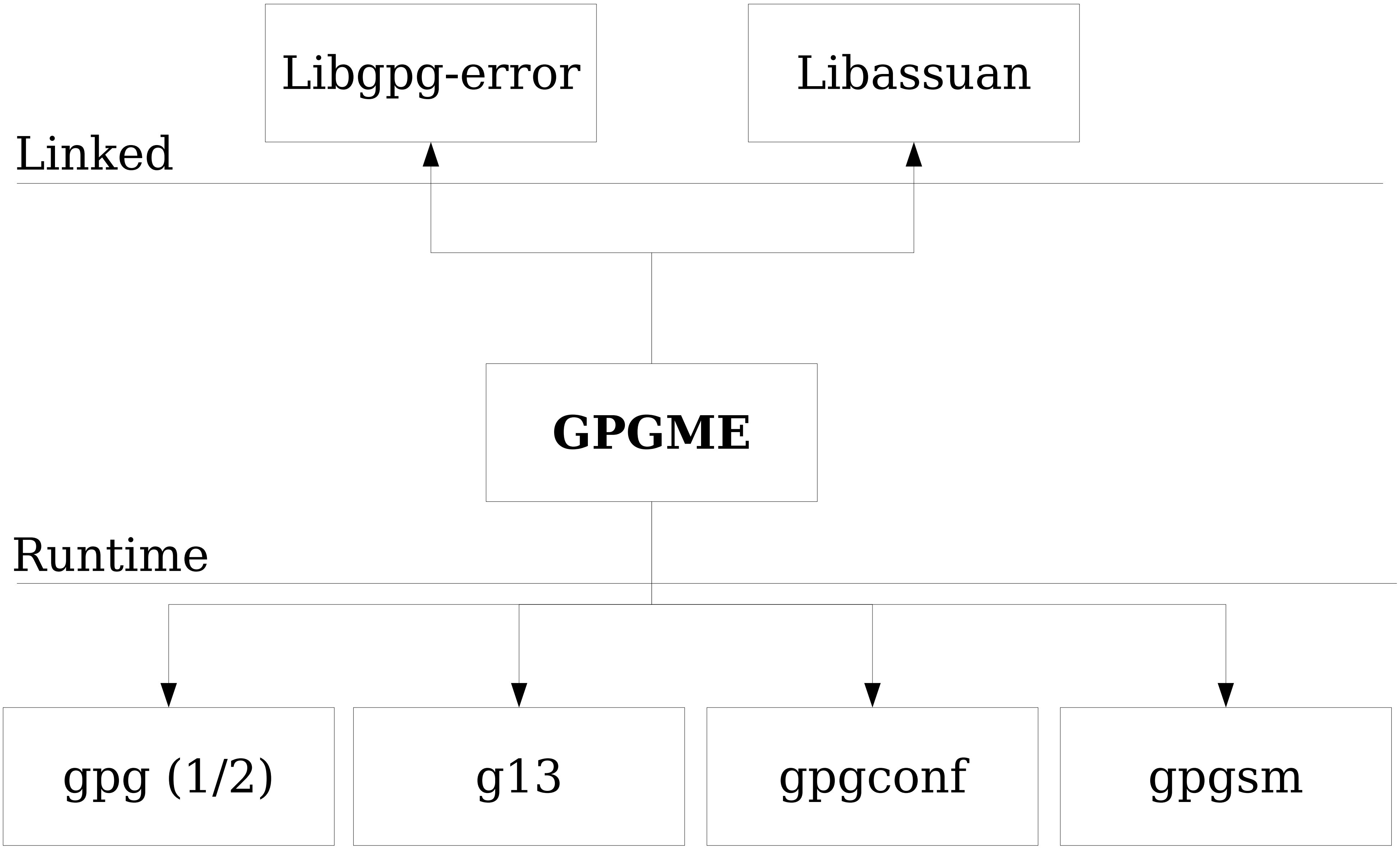
```
pub:f:2048:17:F2AD85AC1E42B367:1199118275:1546232400::f:::scESC:
pub:f:2048:17:F2AD85AC1E42B367:1199118275:1546232400::f:::scESC:::::::
```

- Usage of outdated versions
- Wrong usage / status handling

GnuPG Made Easy

- Make the tool into a library
- Stable API / ABI
- Reference parser implementation
- Works with all GnuPG Versions
- You can update it and you can update GnuPG

Dependencies



Libgpg-error

Libassuan

Linked

GPGME

Runtime

gpg (1/2)

g13

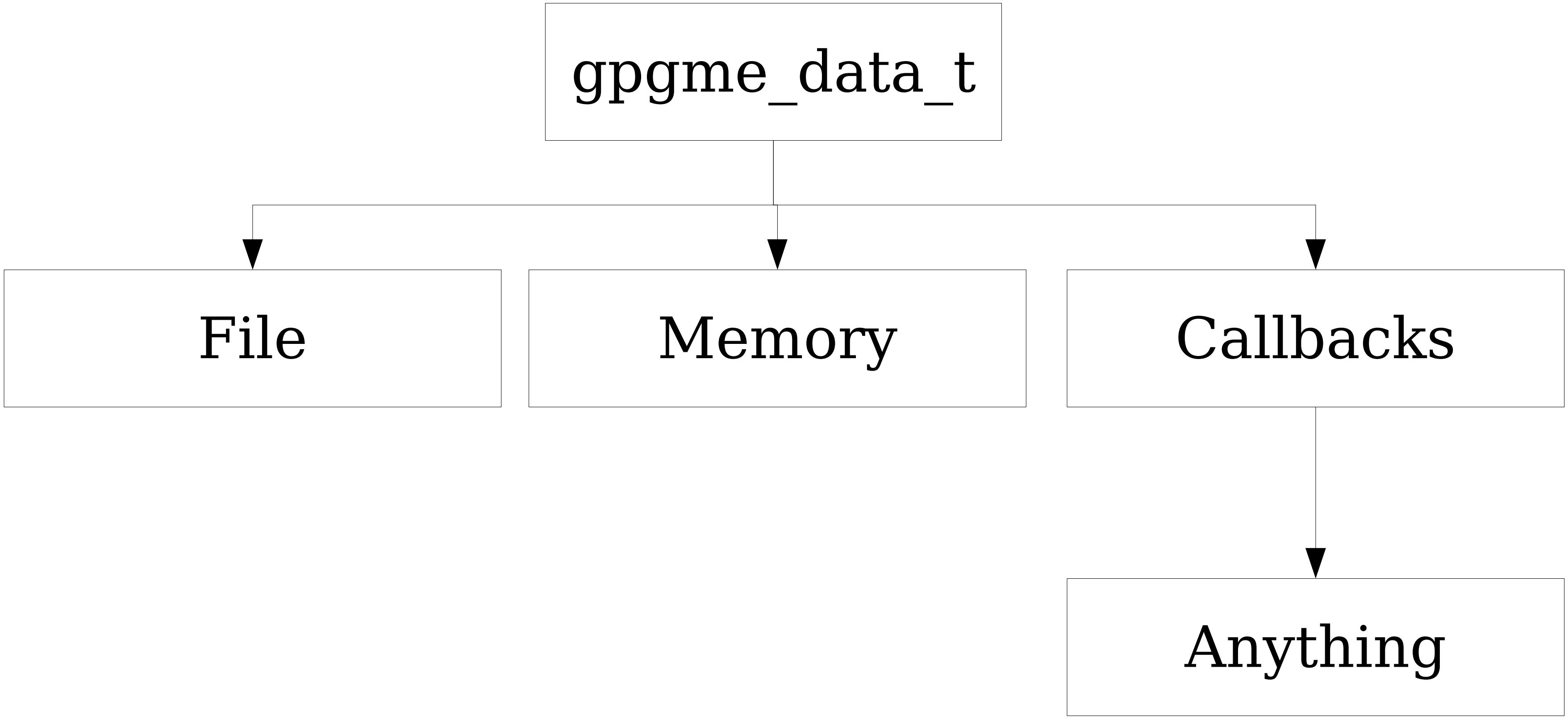
gpgconf

gpgsm

Advantages

- No need for DETAILS
- Decent Documentation
- Version Abstraction
- Data Abstraction
- Platform Abstraction
- Protocol Abstraction (CMS / OpenPGP)
- Convenience functions
- Lots of Tests that can serve as examples
- Maintained by the GnuPG Project

Data abstraction



Concepts and Usage

- Operations controlled by a Context
- Context binds the ressources
- Workflow
  - Create Context for your Protocol
  - Set up data for input / output
  - Set options on a Context
  - Run an Operation (Async or Sync)
  - Handle the result

Engine Abstraction

- Think of an Engine as a distinct tool
  - OpenPGP
  - CMS
  - GpgConf
  - Assuan (can be used for dirmngr or scdaemon)
  - G13 (VFS Container)
  - Spawn (Platform independent process control)

Example - core use

```
gpgme_error_t err;
gpgme_ctx_t ctx;
gpgme_key_t key;
gpgme_keylist_result_t result;

gpgme_check_version (NULL);

gpgme_new (&ctx);
gpgme_set_protocol (ctx,
                              GPGME_PROTOCOL_OpenPGP);

gpgme_op_keylist_start (ctx, "foo@bar.baz", 0);
while (!(err = gpgme_op_keylist_next (ctx, &key)))
    {
        ...
    }
```

Example – context modification

- Look for keys on Keyserver:

```
gpgme_set_keylist_mode(ctx,
                       GPGME_KEYLIST_MODE_EXTERN);
```

- Look for keys with "locate-keys" (if supported)

```
gpgme_set_keylist_mode(ctx,
                       GPGME_KEYLIST_MODE_EXTERN |
                       GPGME_KEYLIST_MODE_LOCAL);
```

Disadvantages

- Low Level API
- Not complete
- Edit Key / Smartcard Edit still tricky
  - 98 Status codes,.. (few are relevant)
- Can make it harder to debug problems
- It's written in C

Language Bindings

- New in GpgME 1.7.0
- Can provide high level functions
- Make it actually easy for their language
    - Python
    - Common Lisp
    - C++
    - Qt

GpgMEpp

- Formerly part of KDE
- Uses C++ patterns for resource management
- Object Oriented
- A bit more high level. E.g. Edit-Interactors
- Manually written / maintained
- Bad documentation

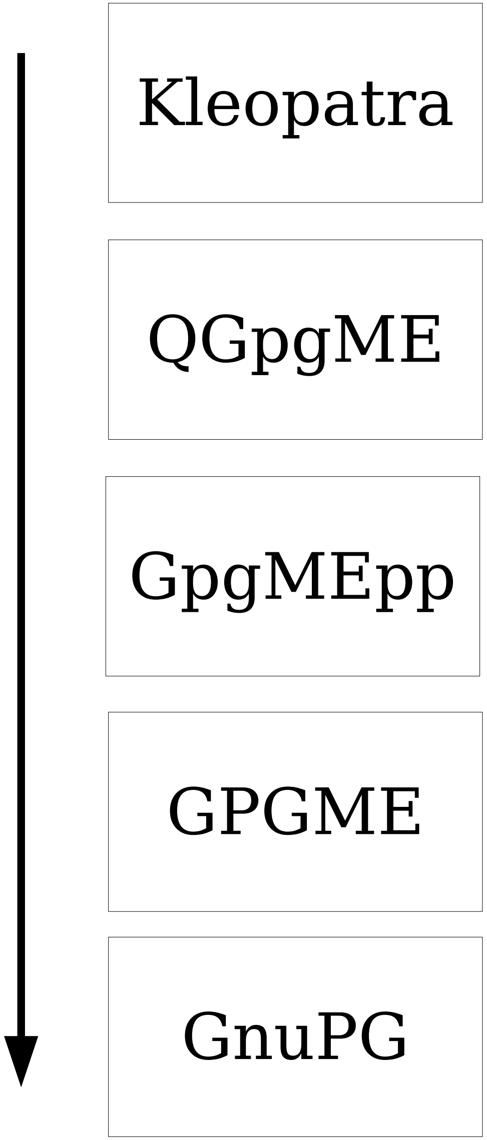
GpgMEpp Example

```
Context *ctx = Context::CreateForProtocol(OpenPGP);

EditInteractor *ei = new
          GpgSetExpiryTimeEditInteractor(std::string("2016-09-08T15:35:37+0200")
Data data;
const Error err = ctx->edit(key, std::unique_ptr<EditInteractor> (ei), data);
delete ctx;
```

QGpgME

- Formerly part of KDE
- Uses itself GpgMEpp
- Convienient API
- Everything is a Job (Async)
- Handles Qt Data Types
- Manually written / maintained



Kleopatra

QGpgME

GpgMEpp

GPGME

GnuPG

QGpgME Example

```
ChangeOwnerTrustJob *job = openpgp()->changeOwnerTrustJob();
connect(job, &ChangeOwnerTrustJob::result, this, [this](Error e)
{
    /* Do something */
});
job->start(key, Key::Ultimate);
```

Python

- Based on PyME (SWIG)

Example:

```
# Init
support.init_gpgme(constants.PROTOCOL_OpenPGP)
c = core.Context()
# Set options
c.set_armor(True)

#Setup Data
text = core.Data(test_text1)
sig = core.Data(test_sig1)

#Verify
c.op_verify(sig, text, None)
result = c.op_verify_result()
```

Python Example

Example:

```
with tempfile.TemporaryFile() as source, \
         tempfile.TemporaryFile() as signed, \
         core.Context() as c:
        source.write(b"Hello world\n")
        source.seek(0, os.SEEK_SET)
        c.op_sign(source, signed, constants.SIG_MODE_NORMAL)
```

Use & Contribute

- Use it, report use cases that are missing
- It's LGPL
- Contribute to it :-)
- More language bindings (C#, Java, etc.)

Questions?

Andre Heinecke <aheinecke@intevation.de>

Mail: gnupg-devel@gnupg.org

xmpp: gnupg-devel@conference.jabber.gnupg.org