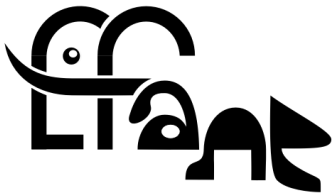
GnuPG

Andreas Enge

LFANT project-team
INRIA Bordeaux–Sud-Ouest
andreas.enge@inria.fr
http://www.math.u-bordeaux.fr/~aenge

Midi de la bidouille
20 mars 2018



GnuPG

Public key encryption

Alice

$K_{\text{priv, A}}$

$K_{\text{pub, A}}$

Bob

m

$c = E(m, K_{\text{pub, A}})$

c ←

$m = D(c, K_{\text{priv, A}})$

Hybrid encryption

Alice

$K_{\text{priv, A}}$

$K_{\text{pub, A}}$

Bob

m

$K \leftarrow \text{random}$

$c_1 = E(K, K_{\text{pub, A}}),\ c_2 = e_K(m)$

$\xleftarrow{(c_1, c_2)}$

$K = D(c_1, K_{\text{priv, A}})$

$m = e_K^{-1}(c_2)$

Signing for integrity and authenticity

Alice

Bob

$K_{\text{priv, A}}$

$K_{\text{pub, A}}$

m

$s = S(m, K_{\text{priv, A}})$

$\xrightarrow{(m,s)}$

$V(m, s, K_{\text{pub, A}}) \stackrel{?}{=}$ yes

GnuPG

Signature verification

- `wget https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz`
- `wget https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz.sig`

- `gpg mpc-1.1.0.tar.gz.sig`

Signature verification

- `wget https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz`
- `wget https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz.sig`
- `gpg --recv-keys AD17A21EF8AED8F1CC02DBD9F7D5C9BF765C61E3`
- `gpg --verify mpc-1.1.0.tar.gz.sig`

GnuPG

Creating a key pair

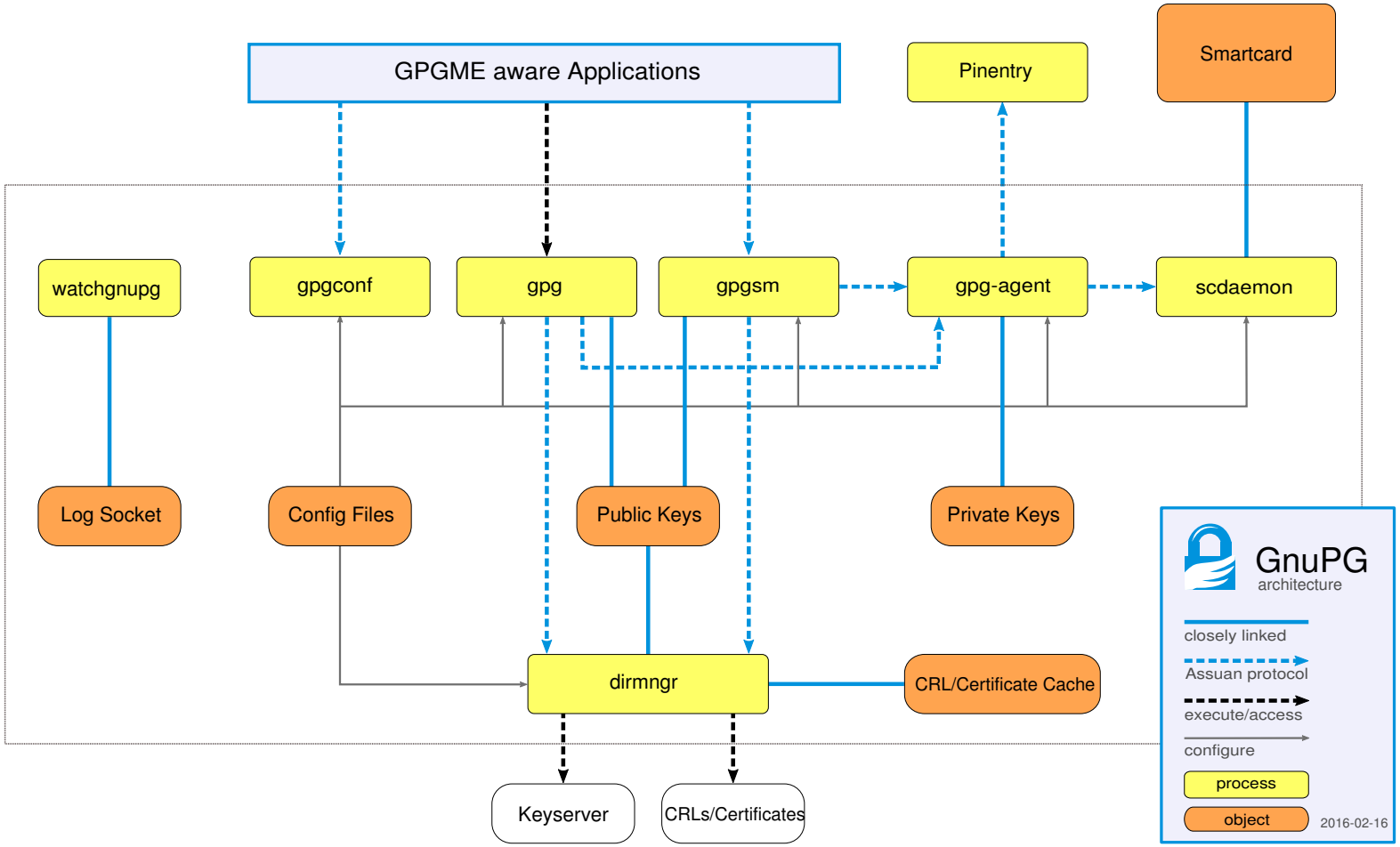
- `gpg --generate-key`

Creating a key pair

- `gpg --generate-key`

More theory...

GnuPG manual p. 178: How the whole thing works internally



(GPLv3+)

Creating a key pair

- `killall -v gpg-agent`
- `vim .gnupgp/gpg-agent.conf`

```
default-cache-ttl 300
max-cache-ttl 3600
pinentry-program /home/enge/.guix-profile/bin/pinentry-curses
```

- `gpg --expert --full-generate-key`
- `gpg --list-keys`
- `gpg --export -a` *KEYID* $>$ *mykey.txt*
  KEYID can be a name or an e-mail address
- `gpg --import` *theirkey.txt*

Encrypting and decrypting a file

- gpg `--encrypt --recipient` *KEYID file.txt*
- gpg `--encrypt -a --recipient` *KEYID file.txt*

- gpg `--decrypt` *file.txt.gpg*
- gpg *file.txt.gpg*

Signing a file

Choose a largish file (at least a few kB).

- gpg `--sign` *file.txt*
- gpg *file.txt.gpg*

- gpg `--detach-sign` *file.txt*
- gpg *file.txt.sig*

- gpg `--detach-sign -a` *file.txt*
- gpg *file.txt.asc*

Signing, then encrypting a file

- gpg `-e -s -r` *KEYID file.txt*
- gpg *file.txt.gpg*

GnuPG

Man in the middle

Alice

Bob

$K_{\text{priv, A}}$

$K_{\text{pub, A}}$

m_A

$s = S(m_A, K_{\text{priv, A}})$

$\xrightarrow{K_{\text{pub, A}}}$

$\xrightarrow{(m_A, s)}$

$V(m_A, s, K_{\text{pub, A}}) \stackrel{!}{=} \text{yes}$

Man in the middle

Alice	Eve		Bob
			
$K_{\text{priv, A}}$	$K_{\text{priv, E}}$		
$K_{\text{pub, A}}$	$K_{\text{pub, E}}$	$\xrightarrow{K_{\text{pub, E}}}$	
m_A	m_B		
$s = S(m_A, K_{\text{priv, A}})$	$s = S(m_E, K_{\text{priv, E}})$		
		$\xrightarrow{(m_E, s)}$	
			$V(m_E, s, K_{\text{pub, E}}) \stackrel{!}{=}$ yes

Trust management

- Maybe necessary: trust thyself!
  `gpg --edit-key` *MYKEYID*
  `trust`
- `gpg --list-keys`
- `gpg --list-sigs`
- `gpg --update-trustdb`

Key signing

- Verify your partner's identity (passport)!
- Import your partner's key
  `gpg --receive-key` *KEYID*
- Let your partner check the key finger print.
  `gpg --fingerprint` *KEYID*
- `gpg --sign-key` *KEYID*
- `gpg --list-sigs`
- `gpg --update-trustdb`
- `gpg --export -a` *KEYID* $>$ *keysig.txt*
- Encrypt *keysig.txt* and send it to your partner.
- Let them execute
  `gpg --import` *keysig.txt*
  `gpg --send-key` *MYKEYID*

Advanced tools: pius

```
pius -e -s MYKEYID
     -H smtp.inria.fr -P 587 -u USER -m FROM
     KEYID
```

Advanced tools: `signing-party`

- `caff`
- `gpg-key2ps` *MYKEYID* > *paperkey.ps*
- More tools for organising key signing parties.

Integration into mail programs

- mutt: edit `.muttrc`, see file `dotmuttrc`

```
set pgp_use_gpg_agent
set pgp_decode_command="gpg --status-fd=2 %?p?--passphrase-fd 0? --no-verbose --quiet --batch --output -
set pgp_verify_command="gpg --status-fd=2 --no-verbose --quiet --batch --output - --verify %s %f"
set pgp_decrypt_command="gpg --status-fd=2 %?p?--passphrase-fd 0? --no-verbose --quiet --batch --output
set pgp_sign_command="gpg --no-verbose --batch --quiet --output - %?p?--passphrase-fd 0? --armor --detac
set pgp_clearsign_command="gpg --no-verbose --batch --quiet --output - %?p?--passphrase-fd 0? --armor --
set pgp_encrypt_only_command="pgpewrap gpg --batch --quiet --no-verbose --output - --encrypt --textmode
set pgp_encrypt_sign_command="pgpewrap gpg %?p?--passphrase-fd 0? --batch --quiet --no-verbose --textmod
...
```

- emacs + gnus: « Cela marche tout seul ! »

Attention: The remainder is not tested.

- pine: ez-pine-gpg, `http://business-php.com/opensource/ez-pine-gpg/`
- Thunderbird: Enigmail
- MacOS X: GPG Suite, `https://gpgtools.org/`
- Android: K-9 Mail + OpenKeychain, `https://f-droid.org/`