Wissenswertes über GnuPG 2.1

Werner Koch

GUUG FFG 2015 — Stuttgart, 26. Marz 2015

Outline

What's new in 2.1

A replacement for the SSH Agent

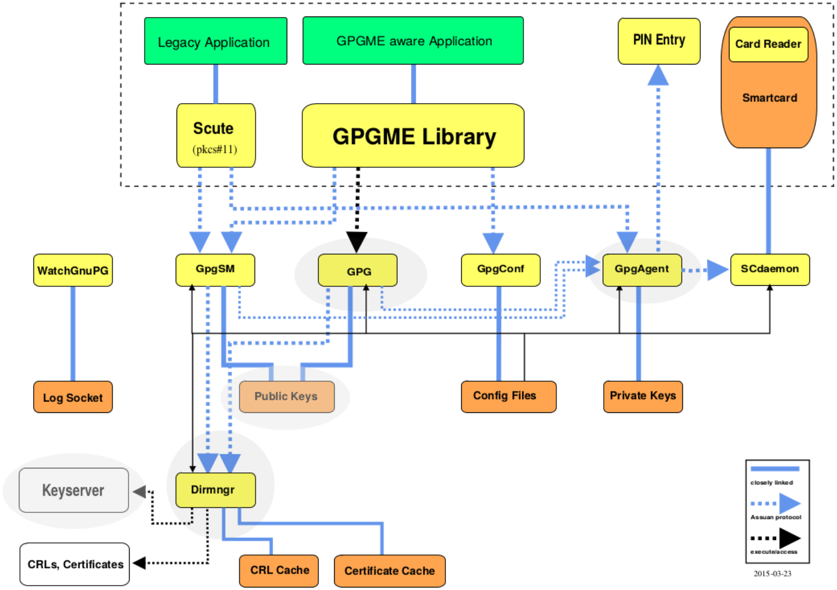
Remote Use

Elliptic Curve Cryptography

Questions

Major changes in 2.1

- *secring.gpg* has gone.
- gpg-agent now takes care of all private keys.
- A faster key store module is the default.
- Stateful keyserver access.


Legacy Application
GPGME aware Application
PIN Entry
Card Reader
Smartcard
Scute
(pkcs#11)
GPGME Library
WatchGnuPG
GpgSM
GPG
GpgConf
GpgAgent
SCdaemon
Log Socket
Public Keys
Config Files
Private Keys
Keyserver
Dirmngr
CRLs, Certificates
CRL Cache
Certificate Cache
closely linked
Assuan protocol
execute/access
2015-03-23

What is this

- Most know the ssh-agent (or Pageant).
- Problem: The keys are not persistant. Need to ssh-add them all the time.
- gpg-agent already takes care of private keys
- Add the ssh-agent protocol to gpg-agent and use those keys.
- Migrate existing keys using ssh-add.
- Automagically make use of smartcards.
- Flawless integration of ssh into the GnuPG system.
- Supports all algorithms (RSA, DSA, ECC and Ed25519)

Ssh-agent with GnuPG demo

Let's see how this works in practise. . . .

What and Why

- Run gpg on the server,
- but keep the private key on the local box
- or, use a smartcard.

Use cases:

- Restore a backup on a server
- Sign large files on the server
- Read mail on remote machine using local keys.

How does it work

- Unix Domain Socket Forwarding in OpenSSH 6.7
- gpg-agent creates a second socket on the client
- OpenSSH forwards a socket from the server to the new socket on the client.
- Done.

Extra socket required to prevent the server from learning too much about the client.

Live demo

```
sockclient="/home/ed/.gnupg/S.gpg-agent-extra"
sockserver="/home/esnowden/.gnupg/S.gpg-agent"
ssh -R $sockserver:$sockclient esnowden@pullach.nsa.gov
```

(To avoid the usual Internet connection problems during demos, we actually ssh to another account on the local machine)
We also put this into sshd_config:

```
StreamLocalBindUnlink
```

Why ECC

- Keys are smaller than keys for RSA.
- Much faster at the same security level
- Well researched for 30 years
- Used for 20 years with smartcards
- Private Keys can be written down
  or even memorized (32 bytes).
- Google and Yahoo end-to-end encryption will default to ECC.
- OpenSSH already uses it as well as a lot of other protocols.

What curves

- There are many different curves. Commonly described by the Weierstrass function:
  $y^2 = x^3 + ax + b$
- Use for Public-Key Cryptography suggested by both, Koblitz and Miller in 1985.
- The problem is based on the dificulty to compute a discrete logarithm ($b^k = g \bmod p$).
- Curve25519 ($y^2 = x^3 + 486662x^2 + x$) developed by DJB in 2006 for encryption. Very clear description of the domain parameter selection process.
- Bernstein et al. developed the EdDSA algorithm based on a variant of the above curve (($-x^2 + y^2 = 1 + dx^2y^2$) for digital signatures.

What to use in GnuPG

- RFC-6637 demands use of the 3 NIST curves
- but allows for arbitary other curves, e.g. Brainpool.
- GnuPG supports this since 2.1.0beta2 (2011-03-08)
- People don't like NIST or Brainpool curves.
  $\Rightarrow$ They won't be the default.
- GnuPG 2.1 supports Ed25519 for signatures. OpenSSH supports the same algorithm.
- Curve25519 for encryption will be supported soon (need to agree on the on-wire point format)
- Stronger non-NIST curves will eventually be supported.

Q&A

https://gnupg.org/faq/whats-new-in-2.1.html
https://wiki.gnupg.org

What questions may I try to answer now?

Danke schön.