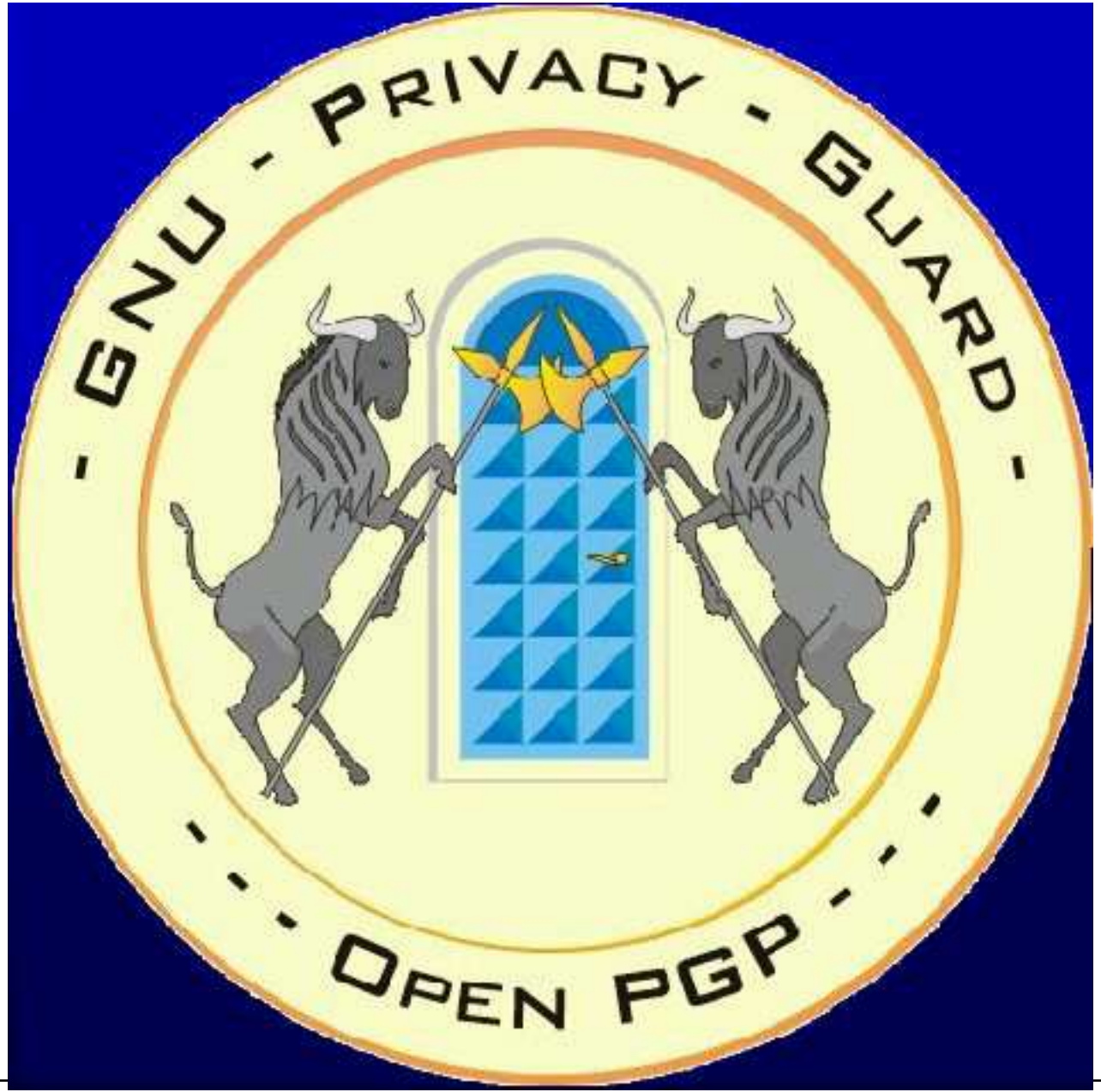
# GNU Crypto Software

(CCC Congress Dec 27th, 2000)



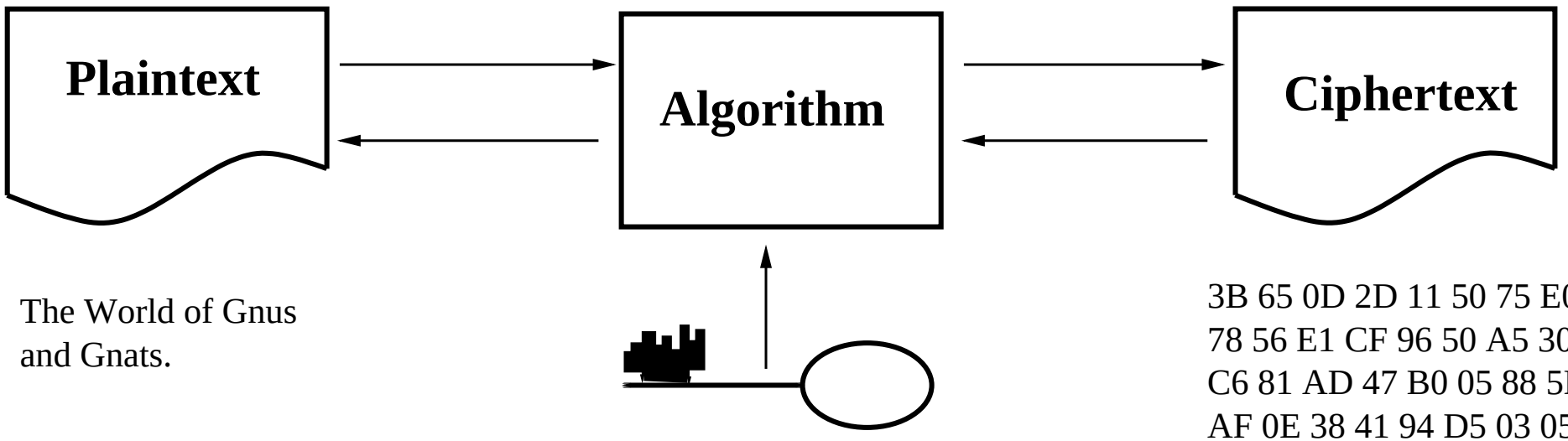
# History

- DH patent expired in 1997.
- Started with GnuPG in late 97.
- OpenPGP WG founded at the same time.
- RFC2440 published in Nov 1998.
- V 1.0 released September last year.

# Are we Arms Trafficker?

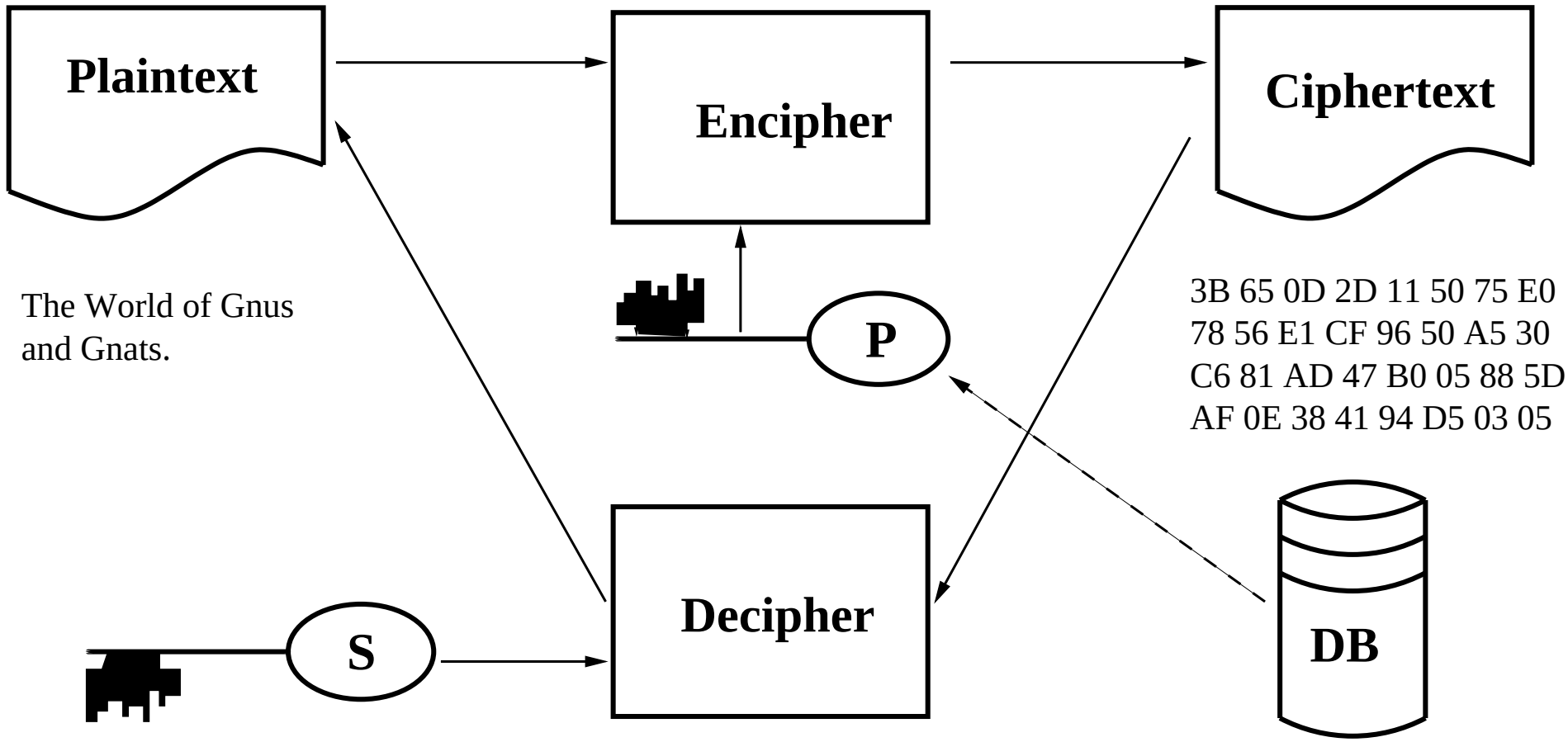
- Good News:  NO

# Symmetric Encryption




- Methods are known for some thausand years.
- Very secure and quite fast.
- Keymanagement is awesome.

# Asymmetric Encryption




- 2 different keys.
- Only one must be kept secret.
- Keymanagement is somewhat easier.

# Trapdoor Algorithms

For the use of public key cryptography a hard mathematic problem is required:

- Knapsack problem
- Factoring problem  (e.g. RSA)
- Discrete logarithm problem  (e.g. DSA)

# RSA

Create a key:

$p, q$ = two secret large primes
$e$ = an arbitrary number relatively prime to $(p-1)(q-1)$.
$n = p * q$
$d = e^{-1} \bmod (p-1)(q-1)$

$n, e$ is the public key and $d$ is the secret key.

Encrypting or signature verification:

$$c = m^e \bmod n$$

Decrypting or signature creation:

$$m = c^d \bmod n$$

# ElGamal

Key:

$p$ = a large prime
$g$ = a generator for $p$
$y = g^x \bmod p$
$x$ = an arbitrary number less than $p$

$p, g, y$ is the public key and $x$ is the secret key.

Encrypt:

$$r = g^k \bmod p$$
$$s = m * y^k \bmod p$$

Decrypt:

$$m = s/r^x \bmod p$$

# ElGamal Signatures

Signature creation:

$$r = g^k \bmod p$$
$$s = (m - x * r) * k^{-1} \bmod (p - 1)$$

Signature verification:

$$1 = g^{-m} * y^r * r^s \bmod p$$

where $k$ is a random number relatively prime to $p - 1$.

# Digital Signature Algorithm (DSA)

Key:

p = a prime of size 512 to 1024 (in 64 bit steps)
q = 160 bit prime factor of $p - 1$
g = $h^{(p-1)/q} \bmod p$
h = any number less than q which yields a $g > 1$
x = any number less than q
y = $g^x \bmod p$

$p, q, g, y$ are the public key and $x$ is the secret key.

Create a signature:

$$r = g^k \bmod p \bmod q$$
$$s = k^{-1}(m + xr) \bmod q$$

Where $k$ is a random number less than $q$ and $r, s$ is the signature.

Verify a signature:

$$w = s^{-1} \bmod q$$
$$u_1 = m * w \bmod q$$
$$u_2 = rw \bmod q$$
$$v = g^{u_1} * g^{u_2} \bmod p \bmod q$$
$$v = r$$

# Standards

- RFC1991 describes the PGP2 data formats.
- PGP 5 data formats are based on PGP 2.
- IETF OpenPGP is based on PGP 5
- RFC2440 released in November 1998.
- Should go to draft standard soon.

# And what about S/MIME?

- X.509 needs a complete infrastructure

  Major interoperability problems

- The PGP model is more oriented to practise.

- Did we learn something from OSI?

- So we don’t use it.

# Problems with PGP

- PGP 2 still needs a patented algorithms.
- PGP 2 is not easy to extend and maintain.
- PGP 5 is proprietary software.
- PGP was subject to U.S. export control rules.
- Would you trust a major U.S. Gov. contractor ??

# What is GnuPG?

- An encryption and signing tool.
  - Offline communication.
  - Data storage.
- A key management tool.
- An implementation of OpenPGP.
- A free replacement of PGP.

# Advantages of GnuPG

- Free, GPLed and GNU software.
- Does not swap secrets to the disk.
- Fast and secure key generation.
- Easy to extend with new algorithms.
- Easy to parse output and error messages.
- A lot of configurable parameters.
- Runs on most POSIX platforms.

# Supported Systems

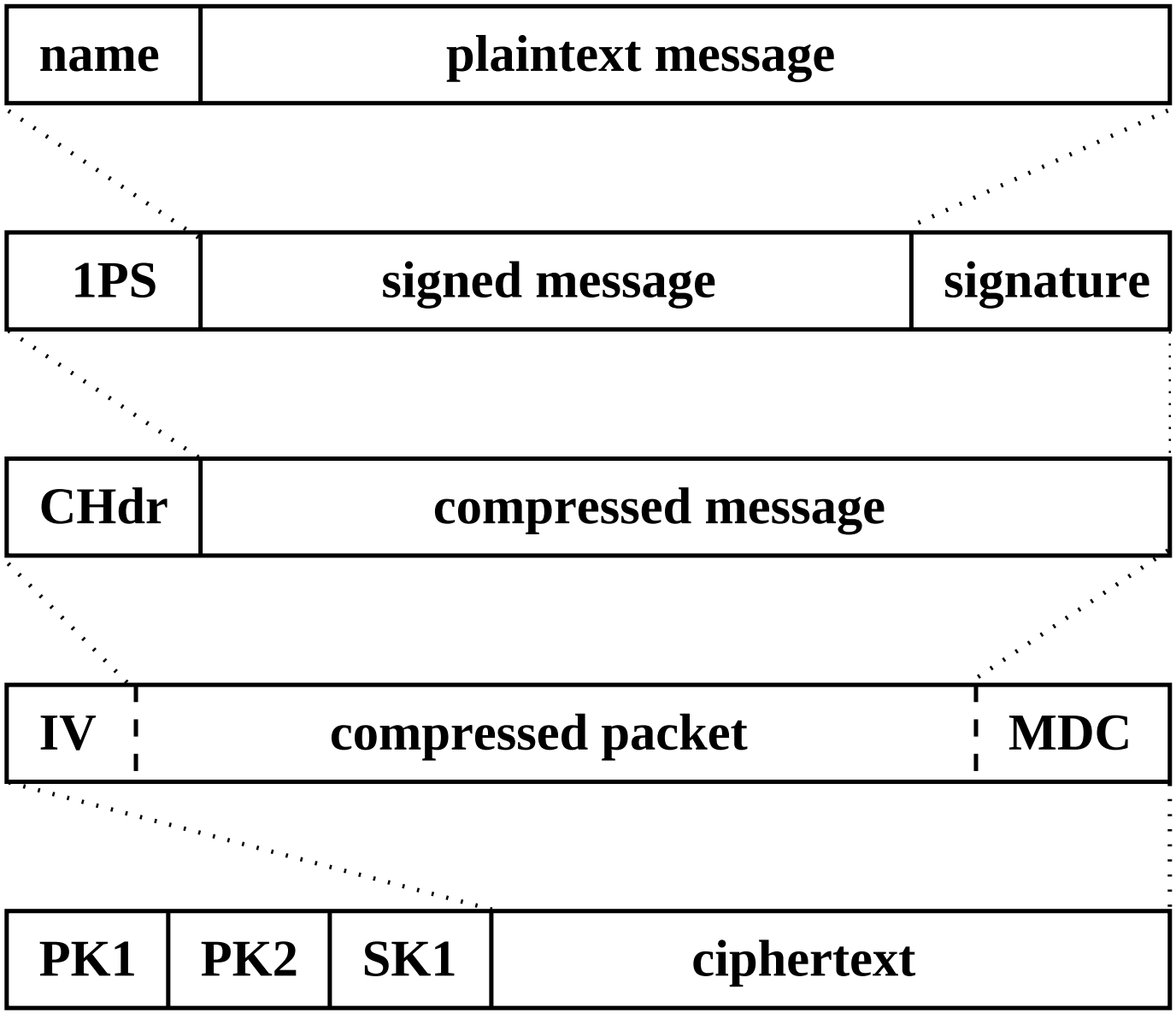
Autoconf & Co enables the software to run on many different platforms:

- GNU (Linux and Hurd kernels)
- Most other Unices
- MS-Windows (W32 API)
- OS/2
- VMS (native)

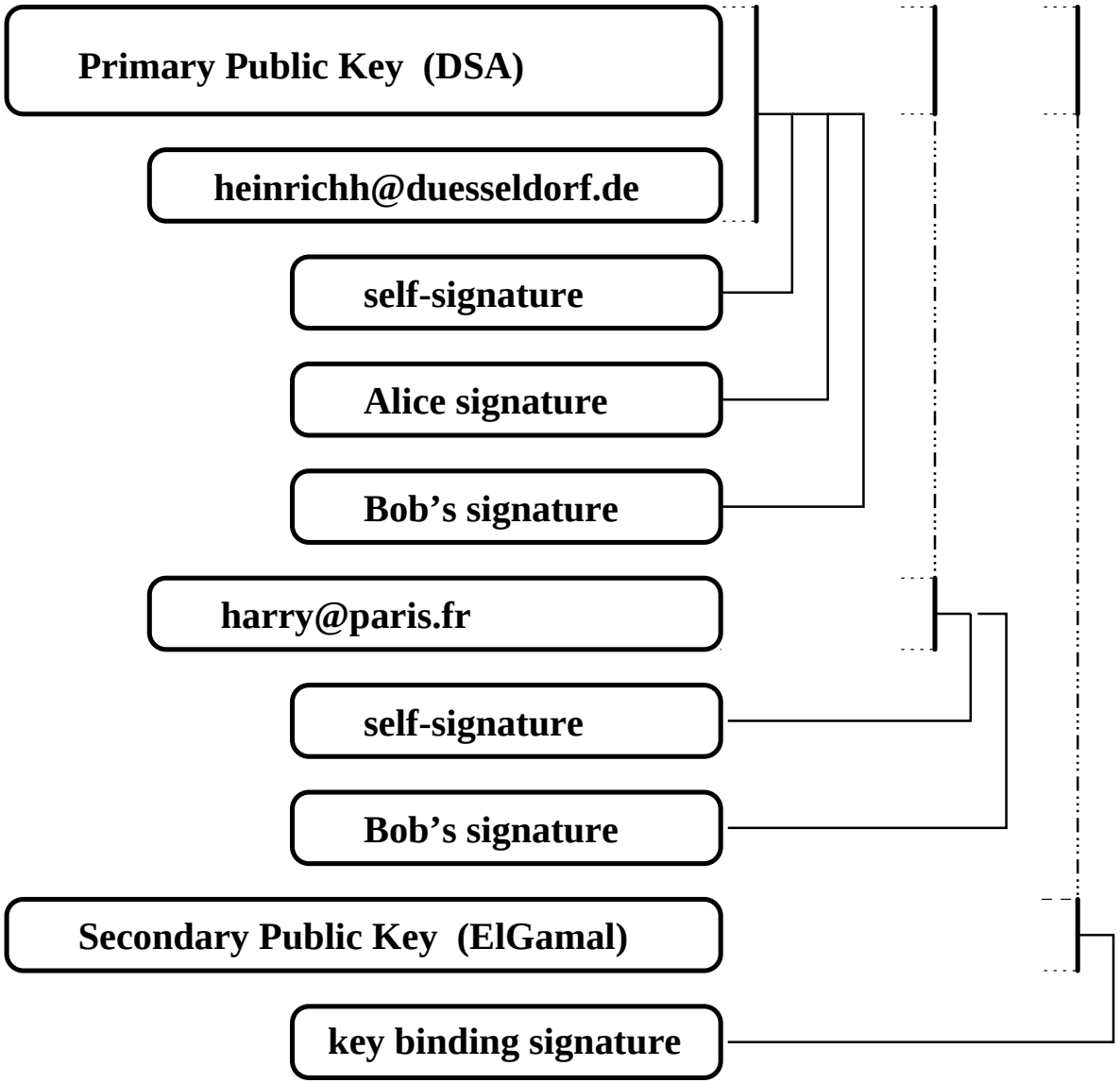
- Assembler optimized math due to GMP.

- RNG is a little bit problematic.

# An OpenPGP Message



# An OpenPGP Key



# Implementation Details

- Subsystem oriented design.
- Simple open/read/write/close logic.
- Pushable filters to cope with nested data structures.
- Easy to read source.
- Makes auditing easier.

# Secure Memory

- Needed to protect secrets from swapping to disk.
- Sets aside a fixed block of memory and does a mlock().
- Math and cipher functions know how to cope with it.
- Simple interface.
- Only secrets are put into secure memory.

# Math Library

- Closely based on GMP.
- Development of GMP has more or less stopped.
- Removed all but the integer stuff.
- Implements the secure memory design.
- Some special functions needed for cryptography.

# Getting Random

- Use of a CSPRNG.
- Different entropy sources.
  - /dev/random
  - Other system info
  - Running system utilities.
- A slow and a fast poll function.
- 2 Pools are used.

# Tool Overview

GnuPG - The Crypto Engine

GPAPA - Wrapper Library

GPA/Seahorse  - Key Management Tools

GEAM - MTA based encryption

GPGME - GnuPG library

WinPT - Windows Privacy Tool

LIBGCRYPT - The GNU Crypto Library

# Future Development (gpg 1.1)

- Protect the secret keys with a hardware token.
- Move the handling of secret keys entirely into a secure perimeter.
- Implement different trust models.
- Have a faster way to store public keys.
- Redesign the current network of keyservers.