OpenPGP:SDK

"PGP is not just for email"

Ben Laurie and Rachel Willmer
Nominet Ltd
EuroOSCON 2005

Overview

- The OpenPGP:SDK
- What PGP does
- History
- Current state of play
- PGP not just for email
- Other applications
- The OpenPGP:SDK

OpenPGP:SDK

- New open source library for OpenPGP
- Developed from scratch
- Apache/BSD licence
- C
- Portable
  - BSD/Linux/Solaris known to work
- http://openpgp.nominet.org.uk

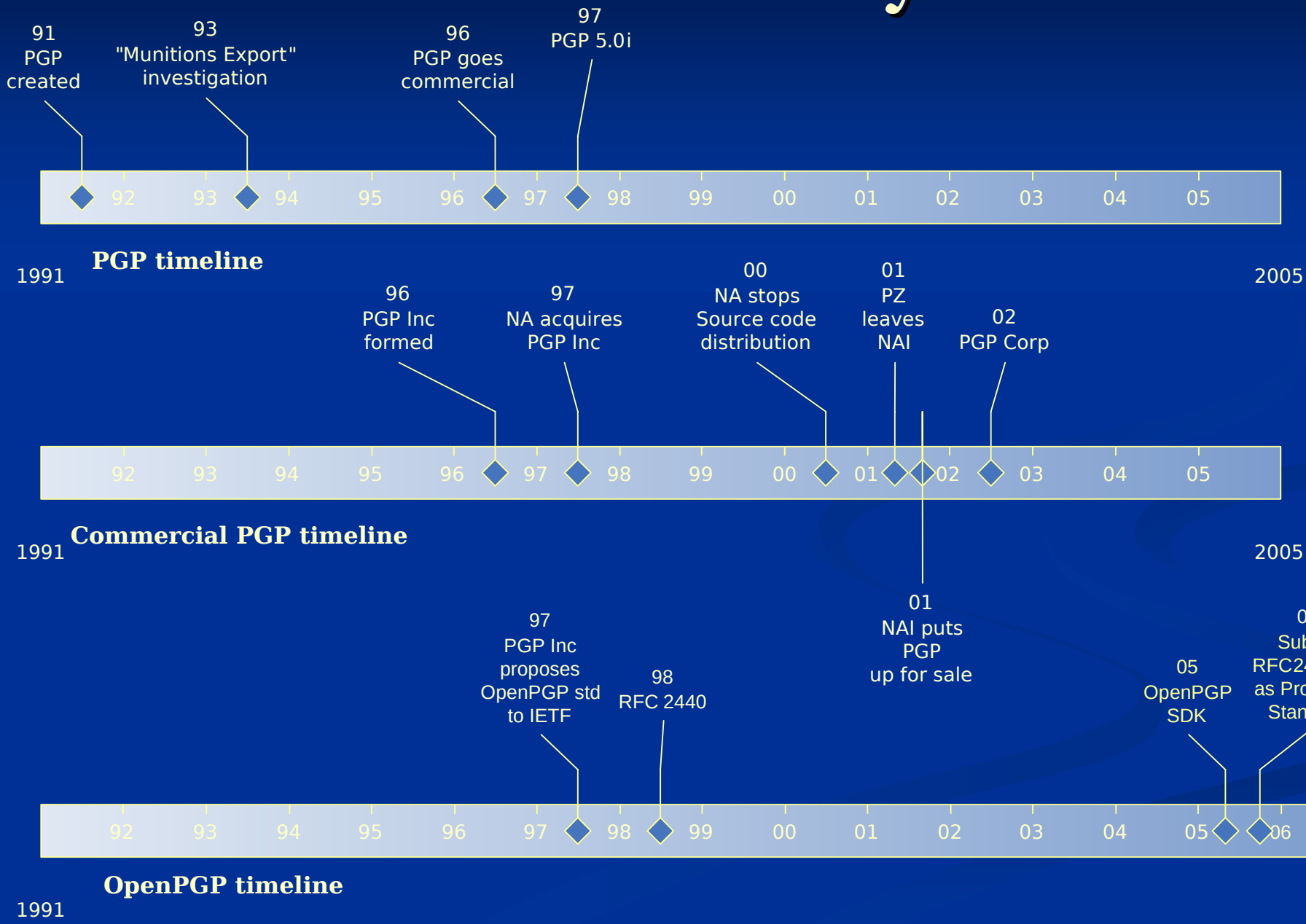
What PGP does

- “OpenPGP software uses a combination of strong public-key and symmetric cryptography to provide security services for electronic communications and data storage”. [Source: RFC 2440]

- Provides:
  - Confidentiality via Encryption
  - Authentication via Digital Signatures
  - Key Management

- Common Usages:
  - Public-key Email Encryption and Signing
  - Secure Disk storage
  - Software Signing

History


91
PGP
created
93
"Munitions Export"
investigation
96
PGP goes
commercial
97
PGP 5.0i
92 93 94 95 96 97 98 99 00 01 02 03 04 05
PGP timeline
1991
2005
96
PGP Inc
formed
97
NA acquires
PGP Inc
00
NA stops
Source code
distribution
01
PZ
leaves
NAI
02
PGP Corp
92 93 94 95 96 97 98 99 00 01 02 03 04 05
Commercial PGP timeline
1991
2005
01
NAI puts
PGP
up for sale
97
PGP Inc
proposes
OpenPGP std
to IETF
98
RFC 2440
05
OpenPGP
SDK
05
Submit
RFC2440bis
as Proposed
Standard
92 93 94 95 96 97 98 99 00 01 02 03 04 05 06
OpenPGP timeline
1991
2006

PGP not just for email

- Main use is email
- Secondary use is secure disk storage
- Software signing

Other Applications

- Authorisation for use of automated services
- X.509 Client certification using PGP
- Notarisation
  - Medical research
  - Copyright

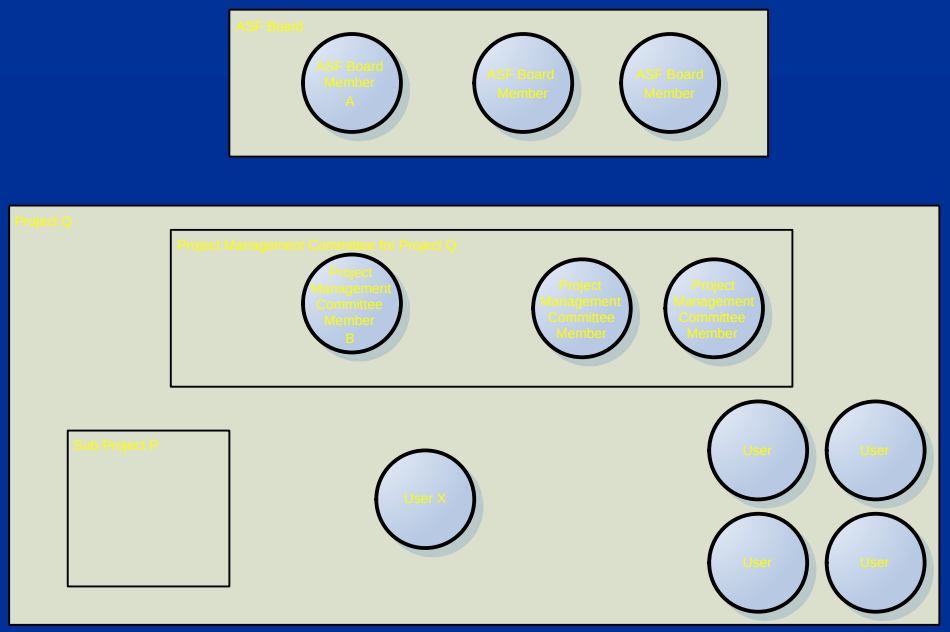
Example: Automated Authenticated Services

- Network Solutions use PGP for authentication of DNS changes
- Also Nominet (who have funded the OpenPGP:SDK) and RIPE
- Apache Software Foundation trialling PGP in their new CA project

DNS

- Network Solutions, Nominet
- Simple system:
  - Domain owner registers key
  - Domain changes sent by PGP-signed email
  - Registrar checks request is signed by registered key

Apache CA



- A signs “B owns project Q”
- B signs “P is subproject of project Q”
- B signs “User X has access to project P”

Validation steps

To check that user X has subversion access to project P:

- Validate that A is ASF board member
  - Build keyring, check signature
- Validate that A has signed B's right to control project Q
  - Build keyring, check signature
- Validate that B has signed project P as subproject of project Q
  - Build keyring, check signature
- Validate that B has signed user X as member of project P
  - Build keyring, check signature

- **8 invocations of command line tool for 1 user validation**

- SDK gives you the ability to do fine-grain operations within a single process

Example: X.509/PGP certificates for online authentication

- What's wrong with X.509 certificates?
  - Hierachical structure
  - Command line tool complicated
  - Long-winded process to get certificate
- What's right with X.509 certificates?
  - Leverage existing browser/server infrastructure
- Solution: combine X.509 with PGP web of trust

X.509/PGP – how it would work

- Server wants access control with Public Key cryptography
- Setup:
  - User sends PGP key to server for signing
  - Utility to create X.509 client cert with PGP public key embedded (New)
  - (X.509 cert can be self-signed, signature is irrelevant)
  - User installs X.509 cert in browser
- Usage:
  - Browser provides cert to server
  - Server ignores X.509 signature and checks PGP key
  - Is PGP key valid and signed by server?
  - If yes, access granted
- Benefit of this approach:
  - Usability: Uses existing browser/server auth mechanism
  - Uses PGP “web of trust”
  - X.509 certificate merely conduit for PGP key exchange

Example: Notarisation

- “Real World” example: copyright protection
- Online equivalent: can do today with existing tools with manual process
- Automated online process would benefit from library
- Applications: copyright, medical research
- Blind Notarisation

The OpenPGP:SDK

- A low-level C API for OpenPGP
- Based around C structures for each OpenPGP data structure
- Can parse OpenPGP packets to produce structures...
- ...or, use structures to construct packets

The Parser

- You provide:
  - A Reader: A function that will read data (e.g. from a file, a socket or some memory)
  - A Callback: which consumes parsed packets
- We provide:
  - Some standard readers (e.g. from file desciptor, from memory)
  - Stackable readers (e.g. read armoured data or compressed packets)
  - A basic parser
  - Stackable parsers (e.g. accumulate a keyring)

The Reader

- Trivial interface – given a buffer and a length, reads as much as it can
- Is not expected to…
  - Buffer
  - Seek
  - Skip data
- Can stack on top of another reader

The Callback

- Called for each parsed packet
  - Handed the structure corresponding to the packet
- Also called for errors
- Indefinite length packets are chunked (e.g. signed cleartext)
- Callbacks can be stacked (e.g. chunked packets could be consolidated in a stacked callback)

Support Library

- Low-level functions
  - Hashes
  - Encryption
  - Signatures
  - Compression
  - Big Number operations
  - Mostly provided by OpenSSL, but pluggable

Support Library

- High-level functions (using OpenPGP:SDK data structures)
  - Check OpenPGP signature
    - On key, subkey, data, cleartext…
  - Generate OpenPGP signature
  - Decrypt encrypted packets
  - Generate encrypted packets
  - Etc…

Packet Construction

- (At least) one API per packet type
- Completely freedom to construct all valid packets in any order
- Packets are constructed from C data structures
- Packets are constructed in memory – then you do what you want with them
- We may provide higher-level APIs to construct standard sequences of packets

Example – Read a Keyring

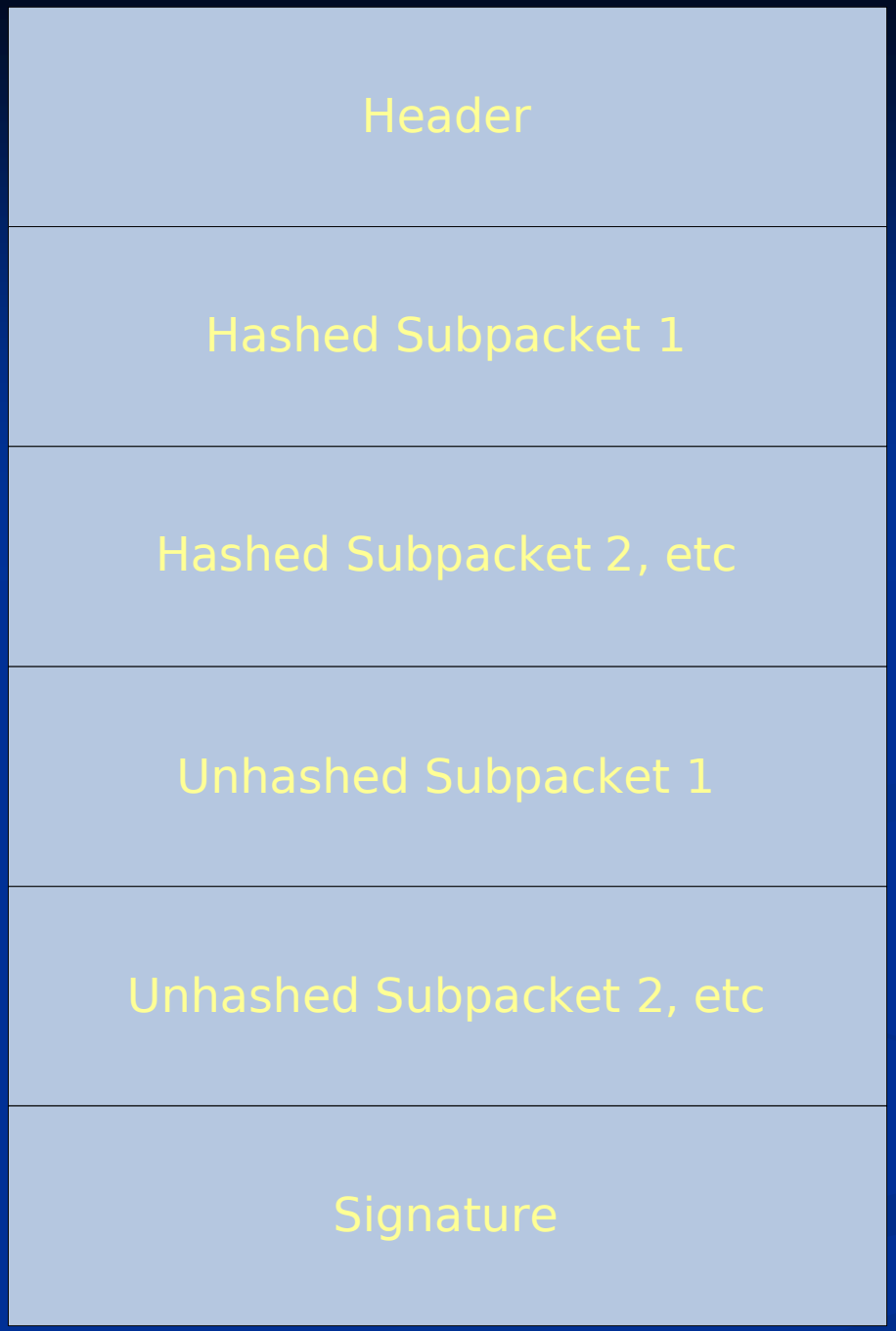
```
memset(&keyring,'\0',sizeof keyring);
ops_parse_options_init(&opt);

arg.fd=open(keyfile,O_RDONLY);
if(arg.fd < 0)
    // Error handling…
opt.reader_arg=&arg;
opt.reader=ops_reader_fd;

ops_parse_and_accumulate(&keyring,&opt);

close(arg.fd);
```

Example – Verify Cleartext Sig I

```
case OPS_PTAG_CT_SIGNED_CLEARTEXT_HEADER:
    free(signed_data);
    signed_data=NULL;
    length=0;
    break;

case OPS_PTAG_CT_SIGNED_CLEARTEXT_BODY:
    signed_data=realloc(signed_data,
                length+content->signed_cleartext_body.length);
    memcpy(signed_data+length,
      content->signed_cleartext_body.data,
      content->signed_cleartext_body.length);
    length+=content->signed_cleartext_body.length;
    break;

case OPS_PTAG_CT_SIGNED_CLEARTEXT_TRAILER:
    signed_hash=content->signed_cleartext_trailer.hash;
    return OPS_KEEP_MEMORY;
```

Example – Verify Cleartext Sig II

```
case OPS_PTAG_CT_SIGNATURE:
    signer=ops_keyring_find_key_by_id(&keyring,
                                      content->signature.signer_id);
    if(!signer)
        {
        fprintf(stderr,"SIGNER UNKNOWN!!!\n");
        exit(2);
        }

    if(ops_check_hash_signature(signed_hash,&content->signature,
                                ops_get_public_key_from_data(signer)))
        {
        puts("Good signature...\n");
        fputs(signed_data,stdout);
        free(signed_data);
        length=0;
        }
    else
        {
        fprintf(stderr,"BAD SIGNATURE!!!\n");
        exit(1);
        }
    break;
```


Header
Hashed Subpacket 1
Hashed Subpacket 2, etc
Unhashed Subpacket 1
Unhashed Subpacket 2, etc
Signature

Example – Write Self-Signed Key

```
ops_write_struct_public_key(&skey.public_key,&opt);

ops_fast_create_user_id(&id,user_id);
ops_write_struct_user_id(&id,&opt);

ops_signature_start(&sig,&skey.public_key,&id,
                    OPS_CERT_POSITIVE);
ops_signature_add_creation_time(&sig,time(NULL));

ops_keyid(keyid,&skey.public_key);
ops_signature_add_issuer_key_id(&sig,keyid);

ops_signature_add_primary_user_id(&sig,ops_true);

ops_signature_hashed_subpackets_end(&sig);

ops_write_signature(&sig,&skey.public_key,&skey,&opt);
```

The OpenPGP:SDK

http://openpgp.nominet.org.uk/