PGP5 User manual

Chris Page

COLLABORATORS			
	TITLE : PGP5 User manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Chris Page	August 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PGP5 User manual

1.1 Main

PGP User Manual PGP

Converted to AmigaGuide format by Chris Page

1.2 About this document

This document is based on the ansi converted man pages provided by Stefan Zakarias <stef@amitar.com.au> as part of the Amiga pgp 5 archive. In order to make this document more relevant to the Amiga I HAVE been forced to make some minor alterations to some sections and instructions found in the original documentation. These changes are due to the implementation differences between UNIX and Amiga OS as well as the standard of documentation of the original freeware UNIX version. These documents should be treated as Amiga specific extensions to the original UNIX documents - do not attempt to transfer the instructions contained here to other platforms. This document does not contain the 'pgp-integration.7.ansi' file because it is not likely to be of great interest to the casual user. Developers interested in integrating pgp5 into their applications should refer to the pgp-integration.7.ansi file for more details.

Chris

dasoft@zetnet.co.uk

1.3 Introduction to PGP 5

There are two files in this package, but several additional modes of operation are available via symbolic links [Amiga note: DO NOT use symbolic links to create the additional pgps and pgpv files, copy the pgpe binary as described in the installation section]

- pgpe is executed to encrypt, or encrypt and sign, files..
- pgps is executed to only sign files. It is a copy of pgpe.
- pgpv is executed to only verify or decrypt signed or encrypted files. It is a copy of pgpe.
- pgpk is the key management application, which is used to generate, retrieve and send keys, as well as manage trust.

Public key cryptography must be fully understood by the user to be useful. A successful PGP user must be familiar with public key cryptography in general, and some PGP-specific concepts (such as the web of trust). If you feel comfortable with your own level of knowledge on this subject, your first step is probably going to be to invoke pgpk to generate a key. Additionally, a page by Phil Zimmermann on the importance of cryptography is included in pgp-intro.

1.4 Converting your old pgp 2.n setup to PGP 5

Users migrating from earlier versions of PGP will need to manually migrate the following configuration files:

- config.txt is now pgp.cfg This file may be copied manually. If not copied, internal defaults will be used. This file is largely unchanged in 5.0. See pgp.cfg for more information on this file.
- pubring.pgp is now pubring.pkr You may copy your old public keyring, or allow 5.0 to generate a new keyring for you.
- secring.pgp is now secring.skr You may copy your old private keyring. Even if you do this, you are encouraged to generate a new DSS/Diffie-Hellman key to allow communication with all 5.0 users.
- language.txt is now language50.txt This file should not be copied from your previous installation, as it is completely different in 5.0. If this file is not present, internal defaults will be used.

1.5 pgp.cfg - Format of the configuration file

The location of this file must be specified in the environmental variable PGPPATH. To do this open a shell window and type:

```
SetEnv PGPPATH PGP:
Copy ENV:PGPPATH ENVARC:
```

This assumes that "pgp.cfg" is installed and exists in the PGP: assigned directory.

All PGP applications accept these options as command-line arguments, as well. Command line arguments always override the configuration file. Options are specified with a double-dash (--) or plus (+), followed by an equal sign (=) and the value, if appropriate. For example, to specify the public keyring to use on an encryption operation:

```
pgpe -r foo@bar.baz.com --pubring=~/fooring.pkr
```

To turn on text mode, in an encryption:

```
pgpe -r foo@bar.baz.com --textmode
```

Options

The following configuration options are supported, both in the pgp.cfg file and on the command line of PGP applications. Case is not important in specifying the variable names, but may be in specifying filenames, depending on your operating environment. Immediately following each option is a description of its type: Boolean is either "1" or "on" or "0" or "off"; String is a string; and Integer is a number.

Armor [Boolean]
: Turns ASCII armoring on or off. The default is off.

ArmorLines [Integer]
: Specifies the maximum number of lines that may be contained in an ASCII armored message. Messages longer than this number of lines will be broken up into multi-part ASCII armored messages. A value of zero indicates an unlimited number of lines. The default is zero.

AutoServerFetch [Boolean]
: If on, when adding keys with pgpk(1), if a specified key file is not found, it is assumed to be the name of a key to retrieve from your default key server (see HTTPKeyServerHost and HTTPKeyServerPort, below). Some intelligence is attempted; even if this is on, if it is obvious a file was intended, a fetch will not be made.
The default is on.

Compress [Boolean]
: Specifies whether messages should be compressed prior to encryption. The default is on.

EncryptToSelf [Boolean]
: If on, automatically encrypts all messages to your default key, as well as to the intended recipient. Intended to allow you to read encrypted mail you've sent. The default is off.

FastKeyGen [Boolean]
: Utilizes pre-generated prime numbers for certain initial operations on DSS/Diffie-Hellman key generation. This only works for "standard" key sizes (1024, 1596 and 2048). The default is on. To turn this off,

```
    use: pgpk +fastkeygen=0 -g

HTTPKeyServerHost [String]
    The name of a keyserver to use in default lookups (see
    AutoServerFetch, above). The default is pgpkeys.mit.edu.

HTTPKeyServerPort [Integer]
    The port to use for connections to the default key server as specified
    in HTTPKeyServer-Host, above. The default is 11371, the default port
    for the Horowitz Key Protocol.

Language [String]
    The country code of the language you wish to use. The default is "us."

LanguageFile [String]
    The file from which to load runtime strings. The default is
    "language50.txt" in your PGP: directory. If this file does not exist,
     PGP will use internal (English) defaults.

MyName [String]
    The key ID you wish to be default for signing operations and the
    EncryptToSelf option, above. There is no default; however, if none is
    specified, the first secret key on your secret keyring will be used
    for most operations.

NoBatchInvalidKeys [Boolean]
    If True, batch mode will fail if the user attempts to encrypt to a key
    that is invalid or not completely valid. If false, batch mode will
    use invalid keys. Note that this option only applies to batch mode;
    normally, PGP will ask the user to confirm prior to encrypting to in
    an invalid key. The default is on.

PubRing [String]
    Your public keyring file. The default is "pubring.pkr" in the current
    directory.

RandomDevice [String]
    Entropy-generating device. If present, it will be used in favor of
    asking the user for keyboard input. [No effect on Amiga - Do not use.]

SecRing [String]
    Your secret keyring file. The default is "secring.skr" in the current
    directory.

WarnOnMixRSADiffieHellman [Boolean]
    Warns the user if encrypting to multiple keys, one or more of which is
    RSA and one or more of which is DSS/Diffie-Hellman. The reason for
    this warning is that most RSA key owners will still be using 2.6.2,
    which will not properly decrypt such messages. The default is on.

WarnOnRSARecipAndNonRSASigner [Boolean]
    Warns the user if encrypting to an RSA key, but signing with a
    DSS/Diffie-Hellman key. The reason for this warning is that most RSA
    key owners will still be using 2.6.2, which will not properly decrypt
    such messages. The default is on.
```

1.6 pgpk - Public and Private key management for PGP.

```
NAME
  pgpk - Public and Private key management for PGP.

SYNOPSIS
  pgpk  [-a  keyfile | -c  [userid]] | -d <userid> | -e <userid> |
        -g | -l[l] [userid] | --revoke[s] <userid> | -r[u|s] <userid> |
        -s <userid> [-u  <yourid>] | -x <userid>] [-o <outfile>] [-z]


DESCRIPTION

      pgpk manages public and private keys for PGP. Unlike other PGP
  applications, pgpk is stream based and not file based; if no files are
  specified, stdin and stdout are used.

OPTIONS

      All configuration options can be controlled from the command line. See
  pgp.cfg for a complete list and how to specify them.


  -a [keyfile]

      Adds the contents of [keyfile] to your keyring. If [keyfile] is not
  specified, input is taken from stdin. [Keyfile] may also be an URL; the
  supported protocols are hkp (Horowitz Key Protocol), http and finger. To
  add foo@bar.baz.com's key to your keyring from PGP, Inc's server, for
  example, enter:

  mpgpk -a hkp://keys.pgp.com/foo@bar.baz.com

  If foo@bar.baz.com has his key in his finger information, you could add
  that with:

  pgpk -a finger://bar.baz.com/foo

  If foo@bar.baz.com has his key on his web page, you could add that with:

  pgpk -a http://www.baz.com/foo/DSSkey.html

      If the Keyfile is not obviously a filename and it doesn't exist as a
  readable file, an attempt will be made to fetch it from your default
  keyserver using the Horowitz Key Protocol.  See pgp.cfg for information
  on setting your default keyserver). For example, if there is no file named
  foo@bar.baz.com readable in the current directory,

  pgpk -a foo@bar.baz.com

  will extract foo@bar.baz.com's key from your default keyserver. Some
  people consider this a security risk (as it could potentially leak
  information about the files on your system if you make a typing error -
  I think this relates to UNIX platforms... Stef.)  Use the
  GetNotFoundKeyFiles configuration option to disable this behavior.
```

```
-c [userid]

    Checks the signatures of all keys on your public keyring. If [userid]
is specified, only the signatures on that key are checked. This command
performs pgpk -ll on all specified keys, then outputs an explicit listing
of trust and validity for each key. Trust is the amount of trust placed in
each key as an introducer. Validity is the certainty that the key and user
ID belong together. Both this command and the long listing function output
a leading column which succinctly describes the condition of the key.

The possible leading columns can have the following first three character
values:

pub A public key
ret A revoked key
sec A secret key
sub A sub-key (in 5.0, this  is  always  a  Diffie-Hellman key)
SIG A signature issued by a public key to which you have the
      corresponding private key (i.e., your key)
sig A signature issued by a public key to which you do NOT have the
      corresponding private key (i.e., someone else's key)
uid A user ID

Following this column is a single character which describes other
attributes of the object:

%  The object is not valid (it does not have enough trusted signatures)
?  No information is available about the object generally because it is a
   signature from a key that is not on your keyring)
!  The object has been checked
*  The object has been tried
@  The object is disabled
+  The object is axiomatically trusted (i.e., it's your key)


-d <userid>

    Toggles  the  disablement of <userid>'s key on your public keyring.


-e <userid>

    Edits <userid>'s key. If this is your key, it allows you to edit your
userid(s) and passphrase. If it is someone else's key, it allows you to
edit the trust you have in that person as an introducer.


-g

    Generate a public/private key pair.


-l[l] [userid]

    Lists information about a key. -ll lists more information about a key.
If [userid] is specified, that key is listed. Otherwise, all keys are
```

listed. See -c, above, for more information about the long format.

-o outfile

Specifies that output should go to outfile. If not specified, output goes to stdout. If the output file is from a key extraction (see -x, below), you may specify an hkp (Horowitz Key Protocol) URL. For example:

pgpk -x foo@bar.baz.com -o hkp://keys.pgp.com

would send foo@bar.baz.com's key to the PGP, Inc. public key server.

--revoke <userid>

Permanately revokes the key specified. There is no way to undo this, so don't play with it if you don't mean it.

--revokes <userid>

Permanently revokes your signature (if any) on the key specified.

-r <userid>

Removes <userid>'s key from your public keyring, and your private as well, if it's there.

-ru <userid>

Removes the given userid from your public and private keyrings.

-rs <userid>

Removes the given signature from your public keyring.

-s <userid> [-u <yourid>]

Signs <userid>'s key with your default signing key. If -u is specified, uses that key, instead.

-x <userid>

Extracts the specified key in ASCII-armored form.

-z

Batch mode. See pgp-integration for a discussion of integrating pgp support into your application.

1.7 pgpe - Encrypts and signs messages

```
NAME
  pgpe - Encrypts and signs messages


SYNOPSIS
  pgpe -r recipient> [-s [-u <myid>]] [-aftz] [-o <outfile>] file ...

  pgpe -c [-aftz] [-o outfile] file ...


DESCRIPTION

      pgpe encrypts and signs files using public key cryptography, or
  encrypts files using conventional cryptography.


OPTIONS

      All configuration options can be controlled from the com- mand line.
  See pgp.cfg for a complete list and how to specify them.


  -a, --armor

      Turn on "ASCII Armoring." This outputs a text-only version of your
  encrypted text. This makes the result safe for mailing, but about 30%
  larger.


  -c

      Conventional encrypting mode. This uses IDEA to encrypt your message.
  As IDEA is a symmetric cipher, no public-key related arguments (-r, -s
  and -u) are accepted with this argument.


  -f

      Stream mode. Accepts input on stdin and places output on stdout. If no
  files are specified as arguments, pgp executes in this mode by default.


  -o outfile

      Specifies that output should go to outfile. If not specified, output
  goes to the default filename. The default filename for each input file is
  the input filename with ".pgp" appended, unless ASCII Armoring is turned
  on, in which case it is ".asc". It is an error to specify multiple input
  files with this option.


  -s

      Sign the document as well as encrypting it. If you wish to sign only,
```

use pgps. This will use your default signing key, or the ID of the key specified with the -u option.

-t

Turns on text mode. This causes PGP to convert your input message to a platform-independent form. It is primarily for use when moving files from one operating system to another.

-u

Sets the ID of the key used for signing. This may be a user ID (such as foo@bar.baz.com) or a Key ID (such as 0x12345678). Specifying this switch without specifying -s (sign) does nothing.

-z

Batch mode. See pgp-integration for a discussion of integrating pgp support into your application.

1.8 pgps - Signs messages

NAME
pgps - Signs messages

SYNOPSIS
pgps [-u <userid>] [-abftv] [-z|-zs] [-o <outfile>] file ...

DESCRIPTION
pgps signs files using public key cryptography.

OPTIONS

All configuration options can be controlled from the command line. See pgp.cfg for a complete list and how to specify them.

-a, --armor

Turn on "ASCII Armoring." This outputs a text-only version of your encrypted text. This makes the result safe for mailing, but about 30% larger. Unlike previous versions of PGP, this also implicitly turns on clearsigning (wrapping the signature around your message, leaving your message in a readable form).

-b

```
    Detached signature. Creates a detached signature file (by default,
  file.sig) instead of combining the signature with the message in the same
  file.


  -f

    Stream mode. Accepts input on stdin and places output on stdout. If no
  files are specified as arguments, pgp executes in this mode by default.


  -o outfile

    Specifies that output should go to outfile. If not specified, output
  goes to the default filename. The default filename for each input file is
  the input filename with ".pgp" appended, unless ASCII Armoring is turned
  on, in which case it is ".asc". If -b (detached signature) is specified,
  the output name for the signature is the input file with ".sig" appended.
  It is an error to specify multiple input files with this option.


  -t

    Turns on text mode. This causes PGP to convert your input message to a
  platform-independent form. It is primarily for use when moving files from
  one operating system to another.


  -u

    Sets the ID of the key used for signing. This may be a user ID (like
 ~foo@bar.baz.com) or a Key ID (like 0x12345678). If not specified, your
  default signing key is used. See also the MyName directive in pgp.cfg.


  -z

    Batch mode. See pgp-integration for a discussion of integrating pgp
  support into your application.
```

1.9 pgpv - Decrypts and Verifies messages.

```
NAME
  pgpv - Decrypts and Verifies messages.


SYNOPSIS
  pgpv [-fmqv] [-z|-zs] [-o <outfile>] file ...


DESCRIPTION
  pgpv Decrypts and verifies files encrypted and/or signed by PGP.


OPTIONS
```

```
    All configuration options can be controlled from the command line.
See pgp.cfg for a complete list and how to specify them.


-f

    Stream mode. Accepts input on stdin and places output on stdout. If no
files are specified as arguments, pgp executes in this mode by default.


-o outfile

    Specifies that output should go to outfile. If not specified, output
goes to the default filename. The default filename for each input file is
the input filename with ".pgp" appended, unless ASCII Armoring is turned
on, in which case it is ".asc". If -b (detached signature) is specified,
the output name for the signature is the input file with ".sig" appended.
It is an error to specify multiple input files with this option.


-m

    More mode. Displays message output with PGP's internal pager, or the
Pager specified in your pgp.cfg file.


-z

    Batch mode. See pgp-integration for a discussion of integrating pgp
support into your application.
```

1.10 installation

```
Taken from 'AMIGA.README' by Stefan Zakarias <stef@amitar.com.au>


Note that PGP5 can co-exist with a previous installation of PGP 2.6.3.
In fact I recommend it!

    1.  Make sure you have "ixemul.library" and "ixnet.library"
        version 47.2 located in your system default "LIBS:" directory.

    2.  Make 2 copies of "PGPE" - One named "PGPV", the other one
        named "PGPS".

    3.  Copy "PGPE", "PGPV", "PGPS" and "PGPK" to your C: directory.
        DO NOT CHANGE THE NAME OF THESE FILES OR THEY MAY NOT WORK!!!

    4.  If you don't have an assignment to "PGP:"...
        a)  Make a directory named "PGP" somewhere on your hard-drive.
        b)  Remember the complete path to this directory.
        c)  Open up your favourite text editor and load in your
            "S:User-StartUp" file.
        d)  Somewhere in the "User-StartUp" file, put the following
```

line, where "xxx" is the path to the PGP directory you created in step 4a.

```
C:Assign PGP: xxx:PGP
```

e) Reboot your computer and continue from here...

5. Copy "pgp.cfg" to your assigned "PGP:" directory.

6. Copy "language50.txt" to your assigned "PGP:" directory. If you do not put "language50.txt" in the PGP: directory, then the default is to use the built-in language which is English.

7. Make a new drawer named "Keys50" in your assigned "PGP:" directory.

8. An environment variable has to be set for PGP5. It's called "PGPPATH" and must be set to the directory you have installed the PGP5 "pgp.cfg" file. You'll have open up a Shell window and type:

```
SetEnv PGPPATH PGP:
Copy ENV:PGPPATH ENVARC:
```

This assumes that "pgp.cfg" is installed and exists in the "PGP:" assigned directory.

9a. If you have never used PGP before and do NOT have a Secret or Public keyrings from a previous version of PGP (EG: PGP 2.6.3), then you'll have to read the docs about "PGPK -g" and create a new pair of keys for yourself!

9b. If you DO have keyrings from a previous version of PGP (E.G: PGP 2.6.3), then make copies of them, naming the new copies as follows:

```
"pubring.pgp" is now "pubring.pkr"
"secring.pgp" is now "secring.skr"
```

Copy the new keyrings, "pubring.pkr" amd "secring.skr", to the "Keys50" directory you created in step 7.

10. When you have the keyrings in the "Keys50" drawer, you should read the documentation named "pgp.cfg.ansi" for details on how to modify the "pgp.cfg" file to suit your purposes. If you've used a previous version of PGP, then you'll notice that the "pgp.cfg" file is almost the same as the old "config.txt" configuration file with just a few extra "enhancements" for the new PGP5 operations.

11. I hope this is enough to get you going... The "xxx.n.ansi" documents should help you do the rest. Just remember that the filenames and paths mentioned in the docs are UNIX orientated. Appropriate conversions to AMIGADOS path specs will have to be made! Look in "pgp.cfg" for examples.

NOTES
=====

* Resist the urge to make PGPS and PGPV a hard or soft link to PGPE!!!

Only COPY PGPE as PGPS and PGPV! It'll take up some extra space on
your hard drive but it is the safest way to do a "link". The reason
is that making links to files on the AMIGA is still NOT very stable.
If you are using AmiFileSafe it is disasterous - you'll more than
likely corrupt the partition the link is on if you try to remove
the link! It WILL force you to reformat that partition to recover.