Pretty Good Privacy version 2.0 - Installation Guide

How to Install PGP

The first question is, what platform are you on?

PGP 2.0 runs on several varieties of Unix, MS-DOS, VAX VMS, Ataris, Amigas, and possibly other operating systems. Naturally, installation instructions differ depending on your hardware. Separate instructions are provided here for MSDOS and Unix.

No matter what the machine you are on, though, do this...

STEP 1:
READ THE DOCUMENTATION. At least read Volume I of the PGP User's Guide. Cryptography software is easy to misuse, and if you don't use it properly much of the security you could gain by using it will be lost! You might also be unfamiliar with the concepts behind public key cryptography; the manual explains these ideas. Even if you are already familiar with Public Key Cryptography, it is important that you understand the various security issues associated with using PGP. It may not be important to read the fine print on a box of breakfast cereal, but it may be crucial to read the label of a prescription drug. Cryptography software is like pharmaceuticals--so read the manual!

See the section below for your system's particular installation instructions.

If you do not have any of these systems, you will either have to port the sources to your machine or find someone who has already done so.

##
For MSDOS:

PGP is distributed in a compressed archive format, which keeps all the relevant files grouped together, and also saves disk space and transmission time.

The current version, 2.0, is archived with the PKZIP utility, and the PGP executable binary release system is in a file named PGP20.ZIP. This contains the executable program, the user documentation, and a few keys and signatures. There is also a second file available containing the C and assembly source code, called PGP20SRC.ZIP; unless you are a programmer interested in cryptography, it is probably of little interest to you. It may or may not be available from the source from which you get PGP20.ZIP; if not, and you want it, see the Licensing and Distribution section of the PGP User's Guide.

You will need PKUNZIP version 1.1 or later to uncompress and split the PGP20.ZIP archive file into individual files. PKUNZIP is shareware and is widely available on MSDOS machines.

Create a directory for the PGP files. For this description, let's use the directory C:\PGP as an example, but you should substitute your own disk and directory name if you use something different. Type these commands to make the new directory:

```
c:
md \pgp
cd \pgp
```

Uncompress the distribution file PGP20.ZIP to the directory. For this example, we will assume the file is on floppy drive A - if not, substitute your own file location.

```
pkunzip a:pgp20
```

Setting the Environment

Next, you can set an MSDOS "environment variable" to let PGP know where to find its special files, in case you use it from other than the default PGP directory. Use your favorite text editor to add the following lines to your AUTOEXEC.BAT file (usually on your C: drive):

```
SET PGPPATH=C:\PGP
SET PATH=C:\PGP;%PATH%
```

Substitute your own directory name if different from "C:\PGP".

Another environmental variable you should set in MSDOS is "TZ", which tells MSDOS what time zone you are in, which helps PGP create GMT timestamps for its keys and signatures. If you properly define TZ in AUTOEXEC.BAT, then MSDOS gives you good GMT timestamps, and will handle daylight savings time adjustments for you. Here are some sample lines to insert into AUTOEXEC.BAT, depending on your time zone:

```
For Colorado:    SET TZ = MST7MDT
For Arizona:     SET TZ = MST7
   (Arizona never uses daylight savings time)
For Chicago:     SET TZ = CST6CDT
For New York:    SET TZ = EST5EDT
For London:      SET TZ = GMT0BST
For Amsterdam:   SET TZ = MET-1DST
```

Now reboot your system to run AUTOEXEC.BAT and set up PGPPATH and TZ for you.

Generating Your First Key

One of the first things you will want to do to really use PGP (other than to test itself) is to generate your own key. This is described in more detail in the "RSA Key Generation" section of PGP User's Guide. Remember that your key becomes something like your written signature or your bank card code number or even a house key - keep it secret and keep it secure! Use a good, unguessable pass phrase and remember it. Right after you generate a key, put it on your key rings and copy your secret keyring (SECRING.PGP) to a blank floppy and write protect the floppy.

After you generate your own key pair, you can add a few more public keys to your key ring. A collection of sample public keys is provided with the release in the file KEYS.ASC. To add them to your public key ring, see the PGP User's Guide, in the section on adding keys to your key ring.

##

For UNIX:

You likely will have to compile PGP for your system; to do this, first make sure the unpacked files are in the correct unix textfile format (the files in pgp20src.zip are in MSDOS CRLF format, so for unix you must unpack with "unzip -a"). Then copy the file "makefile.unx" in the distribution to "Makefile".

If you don't have an ANSI C compiler you will need the unproto package written by Wietse Venema. unproto was posted on comp.sources.misc and can be obtained from the various sites that archive this newsgroup (volume 23: v23i012 and v23i013) or ftp.win.tue.nl file: /pub/programming/unproto4.shar.Z Read the file README in the unproto distribution for instructions on how to use unproto. The unix makefile for pgp (makefile.unx) contains a few targets for compliling with unproto, these assume you have unpacked unproto in a subdirectory "unproto" in the pgp "src" directory.

Then...

type:

```
  "make sungcc"   for Sun with GNU gcc
  "make suncc"    for Sun with cc and unproto
  "make sysv_386" for SVR4 386 with asm primitives
  "make x286"     for XENIX/286 with asm primitives and unproto
  "make ultrix"   for DEC 4.2BSD Ultrix with gcc
  "make rs6000"   for RS6000 AIX
```

If your system doesn't have a target in makefile.unx you will have to edit the makefile, make sure you compile for the correct byte order for your system: define HIGHFIRST if your system is big-endian (eg. Motorola 68030).

If all goes well, you will end up with an executable file called "pgp".

Before you install pgp, run these tests:
(do not create your real public key yet, this is just for testing pgp)

- create a public/secret key pair (enter "test" as userid/password):

```
pgp -kg
```

- add the sample keys from the file "keys.asc" to the public keyring:

```
pgp -ka keys.asc
```

  pgp will ask if you want to sign the keys you are adding, answer yes for at least one key.

- do a keyring check:

```
pgp -kc
```

- encrypt pgpdoc1.txt:

```
pgp -e pgpdoc1.txt test -o testfile.pgp
```

- decrypt this file:

```
pgp testfile.pgp
```

this should produce the file "testfile" compare this file with pgpdoc1.txt

If everything went well, install pgp in a bin directory.

Place the documentation, pgpdoc1.txt and pgpdoc2.txt somewhere where you can reasonably read it; since it's for you, not the software, the location doesn't really matter.

Place the man page (pgp.1) in an appropriate spot. If you don't know anything about how man pages work, you can make the man page look human readable yourself by typing "nroff -man pgp.1 >pgp.man" and reading "pgp.man".

Create a subdirectory somewhere in your home directory hierarchy to hold your public and private key rings and anything else pgp might need (like the language.txt file). You must set the environment variable "PGPPATH" to point to this place before you use the system. Copy the files "language.txt", "config.txt", and the ".hlp" files from the distribution into this subdirectory.

IMPORTANT: THIS DIRECTORY CANNOT BE SHARED! IT WILL CONTAIN YOUR PERSONAL PRIVATE KEYS!

IMPORTANT: PLEASE READ THE SECTIONS IN THE MAN PAGE AND MANUAL ABOUT VULNERABILITIES BEFORE EVEN THINKING ABOUT USING THIS SOFTWARE ON A MULTI-USER MACHINE!!!!

Now, if you haven't done so yet, GO READ THE MANUAL.

##
For VMS:

Read the file readme.vms in the vms subdirectory

##
For Amiga:

[This space intentionally left blank]

##
For Atari:

[This space intentionally left blank]
##