PGP Command Line - Freeware

User’s Guide

Version 6.5

PGP*, Version 6.5.2

9-99. Printed in the United States of America.

LIMITED WARRANTY

Limited Warranty.   Network Associates warrants that for sixty (60) days from the date of original purchase the media (for example diskettes) on which the Software is contained will be free from defects in materials and workmanship.

Customer Remedies. Network Associates' and its suppliers' entire liability and your exclusive remedy shall be, at Network Associates' option, either (i) return of the purchase price paid for the license, if any, or (ii) replacement of the defective media in which the Software is contained with a copy on nondefective media.   You must return the defective media to Network Associates at your expense with a copy of your receipt. This limited warranty is void if the defect has resulted from accident, abuse, or misapplication. Any replacement media will be warranted for the remainder of the original warranty period. Outside the United States, this remedy is not available to the extent Network Associates is subject to restrictions under United States export control laws and regulations.

Warranty Disclaimer. To the maximum extent permitted by applicable law, and except for the limited warranty set forth herein, THE SOFTWARE IS PROVIDED ON AN “AS IS” BASIS WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. WITHOUT LIMITING THE FOREGOING PROVISIONS, YOU ASSUME RESPONSIBILITY FOR SELECTING THE SOFTWARE TO ACHIEVE YOUR INTENDED RESULTS, AND FOR THE INSTALLATION OF, USE OF, AND RESULTS OBTAINED FROM THE SOFTWARE. WITHOUT LIMITING THE FOREGOING PROVISIONS, NETWORK ASSOCIATES MAKES NO WARRANTY THAT THE SOFTWARE WILL BE ERROR-FREE OR FREE FROM INTERRUPTIONS OR OTHER FAILURES OR THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NETWORK ASSOCIATES DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT WITH RESPECT TO THE SOFTWARE AND THE ACCOMPANYING DOCUMENTATION. SOME STATES AND JURISDICTIONS DO NOT ALLOW LIMITATIONS ON IMPLIED WARRANTIES, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU. The foregoing provisions shall be enforceable to the maximum extent permitted by applicable law.

LICENSE AGREEMENT

Network Associates, Inc.
3965 Freedom Circle
Santa Clara, CA 95054
http://www.nai.com
info@nai.com

(408) 988-3832 main

Table of Contents

Preface ix

Organization of this guide ix

Conventions used in this guide ix

How to contact Network Associates x

Customer service x

Year 2000 compliance x

Network Associates training x

Comments and feedback x

Recommended reading x

Chapter 1. Introducing PGP 13

Using PGP 13

A quick overview 13

Basic steps for using PGP 14

Chapter 2. Getting Started 17

Starting PGP 17

Location of PGP files 17

PGPPATH: Set the pathname for PGP 18

Making PGP compatible with PGP 2.6.2 18

Making and exchanging keys 19

Key concepts 19

Making a key pair 20

Protecting your keys 22

Distributing your public key 23

Summary of key server commands 23

Creating a passphrase that you will remember 24

PGP’s command line options 25

Entering PGP configuration parameters on the command line 27

Common PGP functions 27

Creating, disabling, reenabling, and revoking a key 27

Encrypting and decrypting messages 28

Wiping your disk 29
Signing messages 30
Specifying file types 30
Key maintenance commands 31
Creating signature certificates 32
Summary of commands 32
Cancelling an operation 32

Chapter 3. Advanced Topics 33
Identifying your home directory: HOME 33
Using PGP non-interactively from UNIX shell scripts or MSDOS batch files 33
Suppressing unnecessary questions: BATCHMODE 33
Eliminating confirmation questions: FORCE 34
Understanding PGP exit status codes 34
Using PGP as a UNIX-style filter 34
Encrypting and transmitting binary data 35
Sending binary data files in ASCII-armored format without encryption or signature 35
Decrypting ASCII-armored messages 36
Sending a public key in ASCII-armored format 36
Sending ASCII text files to different machine environments 36
Managing signature certificates 37
Creating separate signature certificate and text files 37
Receiving separate signature certificate and text files 37
File management commands 38
Decrypting a message and viewing plaintext output on your screen 38
Decrypting a message and renaming the plaintext filename output 38
Decrypting a message and recovering the original plaintext filename 38
Deleting a key from the key server 38
Encrypting for viewing by recipient only 39
Storing signed files: Signing a file without encrypting 39
Wiping your disk 39

Key management commands 40
Editing your user ID or passphrase, or making an existing key your default signing key 40
Editing the trust parameters for a public key 40
Verifying the contents of your public key ring 41
Verifying a public key over the phone 41
Selecting keys using the key ID 42
PGPPASS: Store your passphrase 42
PGPPASSFD 43

Chapter 4. PGP’s Configuration File 45
Learning about PGP’s configuration file: pgp.cfg 45
ADKKEY: Encrypt with Additional Decryption Key (ADK) 46
ARMOR: ASCII-armor output 46
ARMORLINES: Size of ASCII armor multipart files 47
CERT_DEPTH: Depth of introducers be nested 47
CLEARSIG: Signed message readable with human eyes 47
COMMENT: ASCII armor comment 48
COMPATIBLE: Enable user-interface compatibility with PGP 2.6.2 48
COMPLETES_NEEDED: Number of completely trusted introducers needed 49
COMPRESS: Compression before encryption 49
CIPHERNUM 49
ENCRYPTTOSELF: Encrypt to self 49
ENFORCEADK: Enforce encrypting to an ADK 49
FASTKEYGEN: Fast key generation 50
HASHNUM 50
INTERACTIVE: Confirmation for key adds 50
KEYSERVER_URL 50
MARGINALS_NEEDED: Number of marginally trusted introducers needed 51
MYNAME: Default user ID for signatures 51
PAGER: Shell command to display plaintext output 51
PGP_MIME 51
PGP_MIMEPARSE 51

PUBRING: Filename for your public keyring 52
RANDOMDEVICE 52
RANDSEED: Filename for random number seed 52
SECRING: Filename for your secret keyring 52
SHOWPASS: Echo passphrase to user 53
TMP: Directory pathname for temporary files 53
TEXTMODE: Assume plaintext is a text file 53
TZFIX: Timezone adjustment 54
VERBOSE: Quiet, normal, or verbose messages 54

Appendix A. Exit And Error Codes 55

Index 57

Preface

Organization of this guide

This Guide is divided into the following chapters:

- Chapter 1, "Introducing PGP" This chapter provides an introduction to using PGP Command Line software.
- Chapter 2, "Getting Started" This chapter describes how to start and stop PGP, how to make and exchange keys, and how to perform common PGP functions from the command line.
- Chapter 3, "Advanced Topics" This chapter describes how to use PGP non-interactively from UNIX shell scripts and MSDOS batch files, how to use PGP as a UNIX-style filter, and how to encrypt and transmit binary data.
- Chapter 4, "PGP's Configuration File" This chapter introduces you to PGP's configuration file and the configuration parameters in that file.

Conventions used in this guide

The following describes the conventions used in this guide:

Bold	Menus, fields, options, and buttons are in bold typeface. An example follows: Select the **Clear** option from the **Edit** menu.
Sans-serif font	Pathnames, filenames, icon names, screen text, and special keys on the keyboard are shown in a sans-serif font.
Keystrokes	Keystrokes that you enter are shown in bold sans-serif type.
Variables	Command-line text for which you must supply a value is shown in italic sans-serif type.

How to contact Network Associates

Customer service

To order products or obtain product information, contact the Network Associates Customer Service department at (408) 988-3832 or write to the following address:

Network Associates, Inc.
McCandless Towers
3965 Freedom Circle
Santa Clara, CA 95054-1203
U.S.A.

Year 2000 compliance

Information regarding NAI products that are Year 2000 compliant and its Year 2000 standards and testing models may be obtained from NAI's website at http://www.nai.com/y2k. For further information, email y2k@nai.com.

Network Associates training

For information about scheduling on-site training for any Network Associates product, call (800) 338-8754.

Comments and feedback

Network Associates appreciates your comments and feedback, but incurs no obligation to you for information you submit. Please address your comments about PGP product documentation to: Network Associates, Inc., 3965 Freedom Circle Santa Clara, CA 95054-1203 U.S.A.. You can also email comments to tns_documentation@nai.com.

Recommended reading

Non-Technical and beginning technical books

- Whitfield Diffie and Susan Eva Landau, "Privacy on the Line," *MIT Press*; ISBN: 0262041677
  This book is a discussion of the history and policy surrounding cryptography and communications security. It is an excellent read, even for beginners and non-technical people, but with information that even a lot of experts don't know.

- David Kahn, “The Codebreakers” *Scribner*; ISBN: 0684831309
  This book is a history of codes and code breakers from the time of the Egyptians to the end of WWII. Kahn first wrote it in the sixties, and there is a revised edition published in 1996. This book won't teach you anything about how cryptography is done, but it has been the inspiration of the whole modern generation of cryptographers.

- Charlie Kaufman, Radia Perlman, and Mike Spencer, “Network Security: Private Communication in a Public World,” *Prentice Hall;* ISBN: 0-13-061466-1
  This is a good description of network security systems and protocols, including descriptions of what works, what doesn't work, and why. Published in 1995, so it doesn't have many of the latest advances, but is still a good book. It also contains one of the most clear descriptions of how DES works of any book written.

Intermediate books

- Bruce Schneier, “Applied Cryptography: Protocols, Algorithms, and Source Code in C,” *John Wiley & Sons*; ISBN: 0-471-12845-7
  This is a good beginning technical book on how a lot of cryptography works. If you want to become an expert, this is the place to start.

- Alfred J. Menezes, Paul C. van Oorschot, and Scott Vanstone, “Handbook of Applied Cryptography,” *CRC Press;* ISBN: 0-8493-8523-7
  This is the technical book you should get after Schneier. There is a lot of heavy-duty math in this book, but it is nonetheless usable for those who do not understand the math.

- Richard E. Smith, “Internet Cryptography,” *Addison-Wesley Pub Co*; ISBN: 020192480
  This book describes how many Internet security protocols. Most importantly, it describes how systems that are designed well nonetheless end up with flaws through careless operation. This book is light on math, and heavy on practical information.

- William R. Cheswick and Steven M. Bellovin, “Firewalls and Internet Security: Repelling the Wily Hacker” *Addison-Wesley Pub Co*; ISBN: 0201633574
  This book is written by two senior researchers at AT&T Bell Labs, about their experiences maintaining and redesigning AT&T's Internet connection. Very readable.

Advanced books

- Neal Koblitz, “A Course in Number Theory and Cryptography”
  Springer-Verlag; ISBN: 0-387-94293-9
  An excellent graduate-level mathematics textbook on number theory and cryptography.

- Eli Biham and Adi Shamir, “Differential Cryptanalysis of the Data Encryption Standard,” *Springer-Verlag*; ISBN: 0-387-97930-1
  This book describes the technique of differential cryptanalysis as applied to DES. It is an excellent book for learning about this technique.

Introducing PGP 1

Welcome to PGP Command Line, a member of the PGP product family. PGP products bring easy-to-use, strong encryption and authentication services to your enterprise across a broad range of platforms and applications. With PGP, you can protect your data by encrypting it so that only intended co-workers and business partners can read it. You can also digitally sign data, which ensures it's authenticity and that it has not been altered along the way.

Using PGP

This command line version of PGP is designed for two broad types of applications: transferring information securely between batch servers and integration into automated processes.

- A financial institution can use PGP to securely transfer files from one office to another. Files are encrypted to the receiving server's key and ftp to a directory on a remote server. The remote server periodically examines its receiving directory. When the remote server identifies newly transferred files, it decrypts the files and sends them to their final destination.
- UNIX and Windows developers can use this product to secure financial transactions that users make on the internet. For example, if you sell products on your website, you can include PGP in your scripts to automatically encrypt a customer's order and credit card information for storage or transfer to a secure machine.

The term *MSDOS batch files* refers to a Windows NT command prompt. The term *MSDOS* means the command prompt window that exists in Windows NT.

A quick overview

PGP is based on a widely accepted encryption technology known as *public key cryptography* in which two complementary keys, called a *key pair*, are used to maintain secure communications. One of the keys is designated as a *private key* to which only you have access and the other is a *public key* which you freely exchange with other PGP users. Both your private and your public keys are stored in keyring files.

For a comprehensive overview of PGP encryption technology, refer to *"An Introduction to Cryptography,"* which is included with the product.

Basic steps for using PGP

This section takes a quick look at the procedures you normally follow in the course of using PGP. For details concerning any of these procedures, refer to the appropriate chapters in this book.

1. Install PGP on your computer. Refer to the documentation included with PGP for complete installation instructions.

2. Create a private and public key pair.

   Before you can begin using PGP, you need to generate a key pair. A PGP key pair is composed of a private key to which only you have access and a public key that you can copy and make freely available to everyone with whom you exchange information.

   You can create a new key pair any time after you have finished the PGP installation procedure.

   For more information about creating a private and public key pair, refer to "Making a key pair" on page 20.

3. Exchange public keys with others.

   After you have created a key pair, you can begin corresponding with other PGP users. You will need a copy of their public key and they will need yours. Your public key is just a block of text, so it's quite easy to trade keys with someone. You can include your public key in an email message, copy it to a file, or post it on a public or corporate key server where anyone can get a copy when they need it.

   For more information about exchanging public keys, refer to and "Making and exchanging keys" on page 19 and "Distributing your public key" on page 23.

4. Validate public keys.

   Once you have a copy of someone's public key, you can add it to your public keyring. You should then check to make sure that the key has not been tampered with and that it really belongs to the purported owner. You do this by comparing the unique *fingerprint* on your copy of someone's public key to the fingerprint on that person's original key.

   You can also accept a key as valid based on the presence of a signature from a trusted introducer. PGP users often have other trusted users sign their public keys to further attest to their authenticity. For instance, you might send a trusted colleague a copy of your public key with a request that he or she certify and return it so you can include the signature when you post your key on a public key server. Using PGP, when someone gets a copy of your public key, they don't have to check the key's authenticity themselves, but can instead rely on how well they trust the person(s) who signed your key. PGP provides the means for establishing this level of validity for each of the public keys you add to your public keyring. This means that when you get a key from someone whose key is signed by a trusted introducer, you can be fairly sure that the key belongs to the purported user.

Your Security Officer can act as a trusted introducer, and you may then trust any keys signed by the corporate key to be valid keys. If you work for a large company with several locations, you may have regional introducers, and your Security Officer may be a meta-introducer, or a trusted introducer of trusted introducers.

When you are sure that you have a valid public key, you sign it to indicate that you feel the key is safe to use. In addition, you can grant the owner of the key a level of trust indicating how much confidence you have in that person to vouch for the authenticity of someone else's public key.

5. Encrypt and sign your email and files.

   After you have generated your key pair and have exchanged public keys, you can begin encrypting and signing email messages and files.

6. Decrypt and verify your email and files.

   When someone sends you encrypted data, you can decrypt the contents and verify any appended signature to make sure that the data originated with the alleged sender and that it has not been altered.

7. Wipe files.

   When you need to permanently delete a file, you can use the wipe command to ensure that the file is unrecoverable. The file is immediately overwritten so that it cannot be retrieved using disk recovery software.

Getting Started 2

This chapter covers the following topics:

- Starting and quitting PGP
- Making and exchanging key pairs
- Performing common PGP functions from the command line
- Viewing PGP's online User Guide

Starting PGP

To start PGP, enter the following at the command line:

pgp

You can perform all PGP functions from the command line.

Location of PGP files

In UNIX:

The first time you start PGP, the software checks to see if the environment variable PGPPATH is defined.

- If PGPPATH is defined, the software looks for *pgp.cfg* in the directory specified by PGPPATH. If *pgp.cfg* does not exist in that directory, the software creates *pgp.cfg* in that directory.
- If PGPPATH is not defined, the software looks for *pgp.cfg* in the user's home directory, as defined by the environment variable HOME. If *pgp.cfg* does not exist, the software creates *pgp.cfg* in that directory.

The software creates the keyring files (*pubring* and *secring*), PGP preferences, and the random seed file in the "./pgp" directory off of the user's home directory.

In Windows:

The first time you start PGP, the software checks to see if the environment variable PGPPATH is defined. If PGPPATH is defined, the software puts the pgp.cfg in the %PGPPATH% directory.

If PGPPATH is not defined, the software checks to see if the environment variable USERPROFILE is defined. If USERPROFILE is defined, the software puts the pgp.cfg file in the %USERPROFILE%\Application Data\pgp directory.

If USERPROFILE is not defined, the software puts the pgp.cfg file in %SYSTEMROOT%\pgp.

The preference file is placed in the %USERPROFILE%\Application Data\pgp directory, and the preference file identifies where the default keyrings are placed (normally in the same directory, %USERPROFILE%\Application Data\pgp).

The randseed file is always placed in the %SYSTEMROOT% directory.

PGPPATH: Set the pathname for PGP

This parameter identifies the location of specific PGP files:

```
SET PGPPATH=<PGPpathname>
```

For example:

```
SET PGPPATH=C:\PGP
```

PGP needs to know where the following files are located:

- Your key ring files pubring.pkr and secring.skr
- The random number seed file randseed.rnd
- The PGP configuration file pgp.cfg (or .pgprc)

These files can be kept in any directory. Use the PGPPATH parameter to identify their location.

Making PGP compatible with PGP 2.6.2

This version of PGP includes a compatible switch that enables user-interface compatibility with PGP 2.6.2. You may require this feature for interoperation with scripts that parse the output or otherwise interact with PGP dialogues.

To activate this feature, add the following line to the configuration file, pgp.cfg:

```
COMPATIBLE=on
```

You can also enter **+COMPATIBLE** on the command line.

Making and exchanging keys

This section describes how to generate the public and private key pair that you need to correspond with other PGP users. It also explains how to distribute your public key and obtain the public keys of others so that you can begin exchanging private and authenticated email.

Key concepts

PGP is based on a widely accepted and highly trusted *public key encryption* system, as shown in Figure 2-1, by which you and other PGP users generate a key pair consisting of a private key and a public key. As its name implies, only you have access to your private key, but in order to correspond with other PGP users you need a copy of their public key and they need a copy of yours. You use your private key to sign the email messages and file attachments you send to others and to decrypt the messages and files they send to you. Conversely, you use the public keys of others to send them encrypted email and to verify their digital signatures.

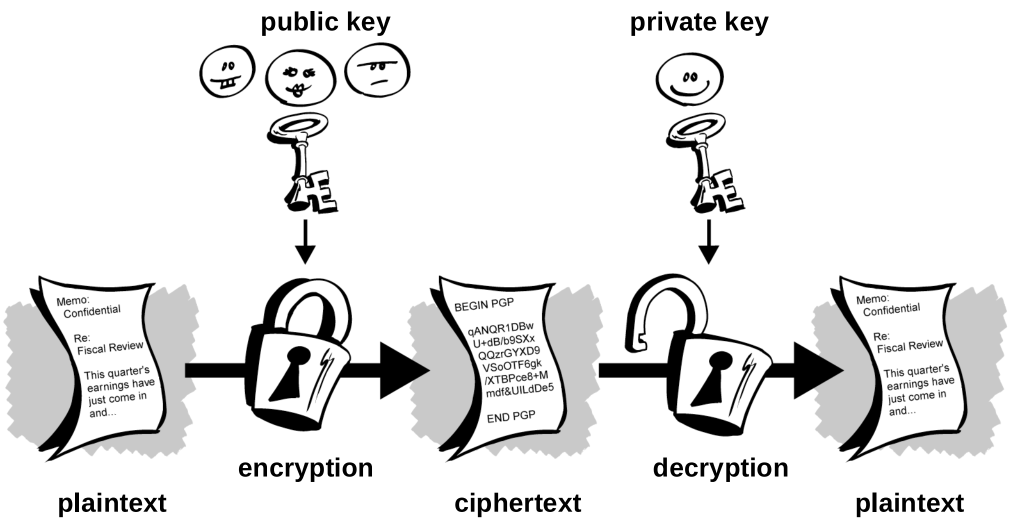

Figure 2-1. Public Key Cryptography diagram

Making a key pair

Unless you have already done so while using another version of PGP, the first thing you need to do before sending or receiving encrypted and signed email is create a new key pair. A key pair consists of two keys: a private key that only you possess and a public key that you freely distribute to those with whom you correspond. You generate a new key pair from the PGP command line.

NOTE: If you are upgrading from an earlier version of PGP, you have probably already generated a private key and have distributed its matching public key to those with whom you correspond. In this case you don't have to make a new key pair (as described in the next section). Instead, use the PGPPATH environment variable to identify the location of your existing keyrings. For more information, see "PGPPATH: Set the pathname for PGP" on page 18.

To create a new key pair

1. Enter the following at the command line:

   pgp -kg

2. For DSS/DH enabled version, go to Step 4.

   For RSA enabled versions, choose the key type:

   a. DSS/DH

   b. RSA

   Go to Step 4.

3. For DSS/DH enabled versions, select either a new signing key or add a new encryption subkey to an existing DSS key.

4. Select the key size you want to generate. A larger key size may take a long time to generate, depending on the speed of the computer you are using.

   The key size corresponds to the number of bits used to construct your digital key. A larger key is stronger. However, when you use a larger key, it takes more time to encrypt and decrypt. You need to strike a balance between the convenience of performing PGP functions quickly with a smaller key and the increased level of security provided by a larger key. Unless you are exchanging extremely sensitive information that is of enough interest that someone would be willing to mount an expensive and time-consuming cryptographic attack in order to read it, you are safe using a key composed of 1024 bits.

5. Enter your user ID. It's not absolutely necessary to enter your real name or even your email address. However, using your real name makes it easier for others to identify you as the owner of your public key. For example:

   Robert M. Smith <rms@xyzcorp.com>

   If you do not have an email address, use your phone number or some other unique information that would help ensure that your user ID is unique.

6. For RSA enabled versions, go to Step 7.

   If you selected a new signing key, enter **y** to create an encryption key, then select the size.

   If you do not want to create an encryption key, enter **n** to generate a new signing key only.

7. Enter a passphrase, a string of characters or words you want to use to maintain exclusive access to your private key. For more information, see "Creating a passphrase that you will remember" on page 24.

   NOTE: Your passphrase should contain multiple words and may include spaces, numbers, and punctuation characters. Choose something that you can remember easily but that others won't be able to guess. The passphrase is case sensitive, meaning that it distinguishes between uppercase and lowercase letters. The longer your passphrase, and the greater the variety of characters it contains, the more secure it is. Strong passphrases include upper and lowercase letters, numbers, punctuation, and spaces but are more likely forgotten.

   ---

8. The software asks you to enter some random text to help it accumulate some random bits to create the keys. Enter keystrokes that are reasonably random in their timing.

9. The generated key pair is placed on your public and secret key rings.

   Use the **-kx** command option to copy your new public key from your public key ring and place it in a separate public key file suitable for distribution to your friends. The public key file can be sent to your friends for inclusion in their public key rings. For more information, see "Distributing your public key" on page 23

Protecting your keys

Once you have generated a key pair, it is wise to put a copy of them in a safe place in case something happens to the originals.

Your private keys and your public keys are stored in separate keyring files, which you can copy just like any other files to another location on your hard drive or to a floppy disk. By default, the private and public keyring s (pubring.pkr and secring.skr) are stored along with the other program files in the directory identified by the PGPPATH environment variable, but you can save your backups in any location you like. For more information, see "PGPPATH: Set the pathname for PGP" on page 18.

Besides making backup copies of your keys, you should be especially careful about where you store your private key. Even though your private key is protected by a passphrase that only you should know, it is possible that someone could discover your passphrase and then use your private key to decipher your email or forge your digital signature. For instance, somebody could look over your shoulder and watch the keystrokes you enter or intercept them on the network or even over the airwaves.

To prevent anyone who might happen to intercept your passphrase from being able to use your private key, you should store your private key only on your own computer. If your computer is attached to a network, you should also make sure that your files are not automatically included in a system-wide backup where others might gain access to your private key. Given the ease with which computers are accessible over networks, if you are working with extremely sensitive information, you may want to keep your private key on a floppy disk, which you can insert like an old-fashioned key whenever you want to read or sign private information.

As another security precaution, consider assigning a different name to your private keyring file and then storing it somewhere other than in the default PGP folder where it will not be so easy to locate.

Distributing your public key

After you create your keys, you need to make them available to others so that they can send you encrypted information and verify your digital signature. You have three alternatives for distributing your public key:

- Make your public key available through a public key server.
- Include your public key in an email message.
- Export your public key or copy it to a text file.

Your public key is basically composed of a block of text, so it is quite easy to make it available through a public key server, include it in an email message, or export or copy it to a file. The recipient can then use whatever method is most convenient to add your public key to their public keyring.

Summary of key server commands

To extract a key from your keyring and send it to the key server:

```
pgp -kx <userid> <keyfile> <URL>
```

To get a key from the key server and put the key on your keyring (requires two commands):

```
pgp -kx <userid> <keyfile> <URL>

pgp -ka <keyfile>
```

To remove a key from your keyring or key server

```
pgp -kr <userid> <URL>
```

To display keys that match a specific userid on the key server:

```
pgp -kv <userid> <URL>
```

Note that the environment variable KEYSERVER_URL identifies the URL of the default key server, for example, ldap://certserver.pgp.com.

Creating a passphrase that you will remember

Encrypting a file and then finding yourself unable to decrypt it is a painful lesson in learning how to choose a passphrase you will remember. Most applications require a password between three and eight letters. A single word password is vulnerable to a dictionary attack, which consists of having a computer try all the words in the dictionary until it finds your password. To protect against this manner of attack, it is widely recommended that you create a word that includes a combination of upper and lowercase alphabetic letters, numbers, punctuation marks, and spaces. This results in a stronger password, but an obscure one that you are unlikely to remember easily. We do not recommend that you use a single-word passphrase.

A passphrase is less vulnerable to a dictionary attack. This is accomplished easily by using multiple words in your passphrase, rather than trying to thwart a dictionary attack by arbitrarily inserting a lot of funny non-alphabetic characters, which has the effect of making your passphrase too easy to forget and could lead to a disastrous loss of information because you can’t decrypt your own files. However, unless the passphrase you choose is something that is easily committed to long-term memory, you are unlikely to remember it verbatim. Picking a phrase on the spur of the moment is likely to result in forgetting it entirely. Choose something that is already residing in your long-term memory. Perhaps a silly saying you heard years ago that has somehow stuck in your mind all this time. It should not be something that you have repeated to others recently, nor a famous quotation, because you want it to be hard for a sophisticated attacker to guess. If it’s already deeply embedded in your long-term memory, you probably won’t forget it.

Of course, if you are reckless enough to write your passphrase down and tape it to your monitor or to the inside of your desk drawer, it won't matter what you choose.

PGP's command line options

The following table identifies and describes PGP's command line options used to encrypt, decrypt, and manage files and keys. The next section, "Common PGP functions" on page 27 tells you how to use these options from the command line.

Option	Description
-a	When used with other options such as encryption or signing, converts a file to ASCII-armored format (creates a .asc file).
-c	Encrypt conventionally.
-e	Encrypt using public key encryption.
-f	Use UNIX-style filter mode to read from standard input and write to standard output
-g	Display help on group options. See table below for -g combinations.
-h	Display summary of commands
-k	Display help on key options. See table below for -k combinations.
-m	Display plaintext output on your screen.
-o	When used with other options such as encryption, decryption, checking signatures, and filter mode, specifies the output filename.
-p	Recover the original plaintext filename.
-s	Sign
-t	Identifies the input file as a text file.
-u	Identifies the key to use for signing.
-w	Instructs PGP to wipe the file.
-z	Identifies the passphrase on the command line.

The **-k** option displays help on key options. It is also used in combination with other option. The following table lists and describes these combinations.

Options	Description
-k	Display help on key options
-kg	Generate a key
-ka	Add keys to the keyring
-kc	Check signatures
-ke	Edit userid or passphrase for your secret key, or make an existing key your default signing key
-kr	Remove keys from the keyring or key server
-krs	Remove signatures attached to keys on the keyring
-ks	Sign keys on the keyring
-kd	Revoke or disable keys on the keyring
-kds	Revoke signatures attached to keys on the keyring
-kx	Extract keys from the keyring and send to key server
-kv	View keys on the keyring
-kvc	View the fingerprints of a set of keys
-kvv	View keys and signatures on the keyring

The **-g** option is always used in combination with another option. The following table lists these combinations and describes how they are used.

Options	Description
-g	Display help on group options.
-ga	Add items to a group.
-gr	Remove items from a group.
-gv	View a group.
-gvv	View a group and the keys it contains. Default is view all groups and their constituent keys.

Entering PGP configuration parameters on the command line

Note that any of the PGP configuration parameters described in Chapter 4, “PGP’s Configuration File” can be entered as long options on the command line (for example, +fastkeygen, +passthrough, or +ADKKEY=”John Doe”).

Common PGP functions

This section describes common PGP functions in the following categories:

- Creating, disabling, reenabling, and revoking a key
- Encrypting and decrypting messages
- Wiping out text
- Signing messages
- Specifying file types
- Key maintenance commands

Note that [brackets] denote an optional field; do not type the brackets.

Creating, disabling, reenabling, and revoking a key

Create a key pair

To create your own unique public and secret key pair, enter the following at the command line:

pgp -kg

Revoke your key

To permanently revoke your own key, issue a key revocation certificate:

pgp -kd <*your_userid*>

Disable or reenable a key

To disable or reenable a public key on your own public key ring:

pgp -kd <*userid*>

Encrypting and decrypting messages

Decrypt a message, or check the signature integrity of a signed file

pgp <*ciphertext_filename*> [-o *plaintext_filename*]

Decrypt a message and recover the original plaintext filename

pgp -p <*ciphertext_filename*>

For more information, see "Decrypting a message and recovering the original plaintext filename" on page 38.

Decrypt a message and view plaintext output on your screen

pgp -m <*ciphertext_filename*>

Output is similar to the UNIX-style "more" command. Output is not written to a file. For more information, see "Decrypting a message and viewing plaintext output on your screen" on page 38.

Decrypt an ASCII-armored message

pgp <*ASCII-armored_message*>

This command decrypts an ASCII-armored message. PGP converts the message to binary, producing a ".pgp" ciphertext file in binary form, then creates the output file in plaintext. For more information, see "Decrypting ASCII-armored messages" on page 36.

Decrypt a message, read from standard input and write to standard output

pgp -feast <*recipients_userid*> <<*input_filename*> ><*output_filename*>

For more information, see "Using PGP as a UNIX-style filter" on page 34.

Encrypt a plaintext file with conventional cryptography only

pgp -c <*plaintext_filename*>

Encrypt a plaintext file with the recipient's public key

pgp -e <*plaintext_filename*> <*recipients_userid*>

Encrypt a message for any number of recipients

pgp -e <*textfile-filename*> <*userid1*> <*userid2*> <*userid3*>...

Encrypt a message for viewing by recipient only

Use this command to specify that the recipient’s decrypted plaintext be shown only on the recipient’s screen and cannot be saved to disk.

pgp -sem <*message.txt*> <*recipients_userid*>

For more information, see “Encrypting for viewing by recipient only” on page 39.

Encrypt with Additional Decryption Key (ADK)

Use this command to add an ADK equal to **ADKKEY** to all generated keys; all outgoing email encrypted to the user’s key is also encrypted to the ADK key identified by this parameter.

pgp +ADKKEY=<*userid*>

or

pgp +ADKKEY=<*keyid*>

For example, ADKKEY=“John Doe” or ADKKEY=“0xAB12C34”

For more information, see “ADKKEY: Encrypt with Additional Decryption Key (ADK)” on page 46.

Enforce encrypting to an ADK

Use this command to set the enforce ADK bit on generated keys.

pgp +ENFORCEADK=on

For more information, see “ENFORCEADK: Enforce encrypting to an ADK” on page 49.

Wiping your disk

Wipe out original plaintext file

pgp -ew <*message.txt*> <*recipients_userid*>

PGP wipes out the plaintext file after producing the ciphertext file.

- Add the **-w** (wipe) option when encrypting.
- Add the **-m** (more) option when decrypting.

For more information, see “Wiping your disk” on page 39.

Signing messages

Sign a plaintext file with your secret key and encrypt it with the recipient's public key

pgp -es <*plaintext filename*> <*recipients_userid*> [-u *your_userid*]

Sign a plaintext file with your secret key

pgp -s <*plaintext_filename*> [-u *your_userid*]

Sign a plaintext ASCII text file

pgp -sta <*plaintext_filename*> [-u *your_userid*]

PGP signs a plaintext ASCII text file with your secret key, producing a signed plaintext message suitable for email.

Specifying file types

Create a ciphertext file in ASCII-armored-64 format

pgp -sea <*plaintext_filename*> <*recipients_userid*>

or

pgp -kxa <*userid*> <*keyfile*> [*keyring*]

The generated file can be uploaded into a text editor through 7-bit channels for transmission as normal email.

Add the **-a** option when encrypting or signing a message or extracting a key. For more information, see "Encrypting and transmitting binary data" on page 35.

Create a plaintext ASCII file

pgp -seat <*message.txt*> <*recipients_userid*>

The file is converted to the recipient's local text line conventions.

Add the **-t** (text) option to other options.

Key maintenance commands

Add a public or secret key file's contents to your public or secret key ring

pgp -ka <*keyfile*> [*keyring*]

Copy a key from your public or secret key ring

pgp -kx <*userid*> <*keyfile*> [*keyring*]

or:

pgp -kxa <*userid*> <*keyfile*> [*keyring*]

Get a key from the key server and put the key on your keyring (requires two commands)

pgp -kx <*userid*> <*keyfile*> <*URL*>

pgp -ka <*keyfile*>

An example of a URL: ldap://certserver.pgp.com

Display the contents of your public key ring

pgp -kv[v] [*userid*] [*keyring*]

Display all certifying signatures attached to each key

pgp -kvv [*userid*] [*keyring*]

Display the fingerprint of a public key

pgp -kvc [*userid*] [*keyring*]

PGP displays the "fingerprint" of a public key, to help verify it over the telephone with the key's owner. To learn more about fingerprints, see "Verifying a public key over the phone" on page 41.

Display the contents of your public key ring and check the certifying signatures

pgp -kc [*your_userid*] [*keyring*]

To learn more, see "Verifying the contents of your public key ring" on page 41.

Display all the keys in a specific key ring filename

pgp <*keyring_filename*>

PGP displays all the keys in a specific key ring filename. When you use this command, PGP lists all the keys in keyfile.pgp, and also attempts to add them to your key ring if they are not already on your key ring.

Edit the userid or passphrase for your secret key, or to make an existing key your default signing key

pgp -ke <*userid*> [*keyring*]

Edit the trust parameters for a public key

pgp -ke <*userid*> [*keyring*]

To learn more, see "Editing the trust parameters for a public key" on page 40.

Remove a key or a userid from your public key ring

pgp -kr <*userid*> [*keyring*]

If you specify a keyring file, PGP tries to open that file and the corresponding public or private keyring file. If the userid that you want to delete pertains to a key with both a public and private key, PGP asks you if you want to delete the private key as well. If you answer No, PGP does not delete anything.

Remove selected signatures from a userid on a keyring

pgp -krs <*userid*> [*keyring*]

Sign and certify someone else's public key on your public key ring

pgp -ks <*recipients_userid*> [-u *your_userid*] [*keyring*]

Creating signature certificates

Create a signature certificate that is detached from the document

pgp -sb <*plaintext_filename*> [-u *your_userid*]

For more information, see "Creating separate signature certificate and text files" on page 37.

Summary of commands

To display a quick command usage summary of PGP, enter the following at the command line:

pgp -h

Cancelling an operation

To cancel the current operation, enter **Ctrl-C** at any prompt.

To cancel a long running operation, enter **Ctrl-C** at any time.

Advanced Topics 3

This chapter describes advanced PGP topics and commands:

- Identifying your home directory.
- Using PGP non-interactively from UNIX shell scripts or MSDOS batch files
- Using PGP as a UNIX-style filter
- Encrypting and transmitting binary data
- Sending ASCII files to different machine environments

Identifying your home directory: HOME

UNIX only. This environment variable identifies the users home directory.

Using PGP non-interactively from UNIX shell scripts or MSDOS batch files

NOTE: MSDOS refers to the Windows NT command prompt.

Suppressing unnecessary questions: BATCHMODE

When the BATCHMODE flag is enabled on the command line, PGP does not ask any unnecessary questions or prompt for alternate filenames:

```
pgp +batchmode <ciphertext_filename>
```

This variable is useful when you run PGP from shell scripts or batch files. When BATCHMODE is on, some key management commands still need user interaction (for example, if a passphrase is required on a key), so shell scripts may need to avoid them.

You can also enable BATCHMODE to check the validity of a signature on a file:

- If there was no signature on the file, the exit code is 1.
- If there was a good signature on the file, the exit code is 0.

Eliminating confirmation questions: FORCE

When you instruct PGP to overwrite an existing file or remove a key from a keyring (the -kr command), PGP requires confirmation.

To run PGP non-interactively from a UNIX shell script or MSDOS batch file, use the FORCE option to instruct PGP to assume a “yes” response each time PGP requires confirmation:

pgp +force <*ciphertext_filename*>

or:

pgp -kr +force <*your_userid*>

Understanding PGP exit status codes

When you run PGP in “batch” mode (for example, from an MSDOS “.bat” file or from a UNIX shell script), PGP returns an error exit status to the shell.

- A zero exit status code signifies a normal exit.
- A non-zero exit status code tells you that an error occurred. Different error exit conditions return different exit status codes to the shell.

Using PGP as a UNIX-style filter

UNIX uses pipes to make two applications work together. The output of one application can be directly fed through a pipe to be read as input to another application. For this to work, the applications must be capable of reading the raw material from “standard input” and writing the finished output to “standard output.”

To use PGP’s UNIX-style filter mode, reading from standard input and writing to standard output, add the -f option:

pgp -feast <recipients_userid> <<*input_filename*> ><*output_filename*>

This feature makes it easier to use PGP with email applications.

When you use PGP’s filter mode to decrypt a ciphertext file, you may find the PGPPASS environmental variable useful. This variable holds the passphrase so that PGP does not prompt you for this information. For more information, see “PGPPASS: Store your passphrase” on page 42.

Encrypting and transmitting binary data

Many email systems only allow messages that contain ASCII text. As a result, PGP supports an ASCII-armored format for ciphertext messages (similar to MIME).

This format, which represents binary data using only printable ASCII characters, enables you to transmit binary encrypted data through 7-bit channels, or to send binary encrypted data as normal email text. PGP's ASCII-armored format acts as a form of "transport armor," protecting the message against corruption as it travels through intersystem gateways on the Internet. PGP also appends a CRC to detect transmission errors.

ASCII-armored format converts the plaintext by expanding groups of 3 binary 8-bit bytes into 4 printable ASCII characters. As a result, the file expands by about 33%. However, this expansion is offset by the compression that occurs before encryption.

To produce an ASCII-armored formatted file, enter the following command:

pgp -ea <*plaintext_filename*> <*recipients_userid***>**

This command instructs PGP to produce a ciphertext file in ASCII-armored format called message.asc. This file contains data in a MIME-like ASCII-armored format. You can upload the file into a text editor through 7-bit channels and transmit as normal email.

Most email facilities prohibit messages that are more than 50000 or 65000 bytes. Larger messages are broken into smaller files. If you request ASCII-armored format for a large file, PGP breaks the file into smaller files named with extensions ".as1", ".as2", ".as3", and so on.

Sending binary data files in ASCII-armored format without encryption or signature

Use PGP's -a option to convert a file into ASCII-armored format. No encryption or signing is involved, so neither sender or recipient requires a key. When you use the -a option, PGP breaks big files up into smaller files that can be sent via email, attempts to compress the data before converting it to ASCII-armored format, and appends a CRC error detection code to each of the smaller files. Use the command as follows:

pgp -a <*binary_filename***>**

This command instructs PGP to produce an ASCII-armored file called "filename.asc". The recipient uses the -p option to unwrap the message and restore the sender's original filename.

Decrypting ASCII-armored messages

To decrypt an ASCII-armored message, enter the following command:

pgp <*ASCII-armored_filename*>

PGP recognizes that the file is in ASCII-armored format, converts the file back to binary (creating a .pgp ciphertext file in binary form), and creates an output file in normal plaintext form.

If the original message was large and sent in a number of smaller files, you must concatenate the files in their proper order into one file before decrypting the message. When PGP is decrypting the message, it ignores an extraneous text in mail headers that are not enclosed in the ASCII-armored message blocks.

Sending a public key in ASCII-armored format

To send a public key to someone else in ASCII-armored format, add the -a option while extracting the key from your keyring.

If you forgot to use the -a option when you made a ciphertext file or extracted a key, you can convert the binary file into ASCII-armored format by using the -a option (do not specify encryption). PGP converts the file to a ".asc" file.

Sending ASCII text files to different machine environments

PGP encrypts any plaintext file, binary 8-bit data, or ASCII text. The most common use of PGP is for email, which is ASCII text.

ASCII text is represented differently on different machines. For example, on an MSDOS system, all lines of ASCII text are terminated with a carriage return followed by a linefeed. On a UNIX system, all lines end with just a linefeed. On a Macintosh, all lines end with just a carriage return.

Normal unencrypted ASCII text messages are often automatically translated to some common "canonical" form when they are transmitted from one machine to another. Canonical text has a carriage return and a linefeed at the end of each line of text.

Encrypted text cannot be automatically converted by a communication protocol, because the plaintext is hidden by encipherment. To remedy this problem, PGP's "t" option lets you specify that the plaintext be treated as ASCII text and converted to canonical text before encryption. When the message is received, the decrypted plaintext is automatically converted to the appropriate text form for the local environment.

To use this feature, enter the “t” option when encrypting or signing a message:

pgp -et <*plaintext_filename*> <*recipients_userid*>

If PGP detects non-text binary data in the plaintext file, PGP ignores the “t” option.

PGP includes an environment variable that corresponds to the “t” option, TEXTMODE. If you consistently receive plaintext files rather than binary data, set TEXTMODE=ON.

Managing signature certificates

Creating separate signature certificate and text files

In most cases, signature certificates are physically attached to the text they sign. This makes it convenient to verify signatures. You can, however, create a separate, detached signature certificate file, and then send both files (the text file and the signature certificate file) to the recipient. This feature is useful when more than one party must sign a document such as a legal contract, without nesting signatures. Each person’s signature is independent.

To create a separate, detached signature certificate file, combine the ‘b’ (break) option with the ‘s’ (sign) option. Enter the following command:

pgp -sb <*plaintext_filename*> [-u <*your_userid*>]

This instructs PGP to produce an separate, detached signature certificate in a file named letter.sig. The contents of letter.sig are not appended to <letter.txt>.

Receiving separate signature certificate and text files

When you attempt to process a signature certificate file, PGP asks you to identify the corresponding text file. Once the text file is identified, PGP checks the signature integrity.

If you know that a signature is detached from a text file, you can specify both filenames on the command line:

pgp <*letter.sig*> <*letter.txt*>

or

pgp <*letter*> <*letter.txt*>

File management commands

Decrypting a message and viewing plaintext output on your screen

To view decrypted plaintext output on your screen (similar to the UNIX-style “more” command), without writing the output to a file, use the -m (more) option when you decrypt:

pgp -m <*ciphertext_filename*>

This command instructs PGP to display the decrypted plaintext on your screen, one screen at a time.

Decrypting a message and renaming the plaintext filename output

When PGP encrypts a plaintext file, it saves the original filename and attaches it to the plaintext before it is compressed and encrypted. When PGP decrypts the ciphertext file, it names the plaintext output file with a name similar to the input ciphertext filename, but drops the extension.

Use the -o option on the command line to specify a more meaningful plaintext filename for the output:

pgp -o <*new_plaintext_filename*> <*ciphertext_filename*>

Decrypting a message and recovering the original plaintext filename

As stated in the previous section, when PGP encrypts a plaintext file, it saves the original filename and attaches it to the plaintext before it is compressed and encrypted. Use the “-p” option to instruct PGP to preserve the original plaintext filename and use it as the name of the decrypted plaintext output file:

pgp -p <*ciphertext_filename*>

Deleting a key from the key server

pgp -kr <*userid*> <*URL*>

An example of a URL: ldap://certserver.pgp.com

Encrypting for viewing by recipient only

To specify that the recipient's decrypted plaintext be shown only on the recipient's screen and not saved to disk, add the -m option:

pgp -sem <*message.txt*> <*recipients_userid*>

When the recipient decrypts the ciphertext with their secret key and passphrase, the plaintext is displayed on the recipient's screen but is not saved to disk. The text is displayed as it would if the recipient used the UNIX "more" command, one screen at a time. If the recipient wants to read the message again, they must decrypt the ciphertext a second time.

This feature is the safest way for you to prevent your sensitive message from being inadvertently left on the recipient's disk.

Note that this feature does not prevent a clever and determined person from finding a way to save the decrypted plaintext to disk -- it is designed to help prevent a casual user from doing it inadvertently.

Storing signed files: Signing a file without encrypting

If you sign a plaintext file without specifying encryption, PGP compresses the file after you sign it. This makes the file unreadable to the casual human observer. This is a suitable way to store signed files in archival applications.

Wiping your disk

After PGP produces a ciphertext file for you, you can request PGP to automatically overwrite and delete the plaintext file, leaving no trace of plaintext on the disk. Use the "w" when a plaintext file contains sensitive information; it prevents someone from recovering the file with a disk block scanning utility.

Use the "w" option when you encrypt and sign a message:

pgp -ew <*message.txt*> <*recipients_userid*>

This instructs PGP to create a ciphertext file "message.pgp", and to destroy the plaintext file "message.txt".

Note that this option will not wipe out any fragments of plaintext that your word processor might have created on the disk while you were editing the message before running PGP. Most word processors create backup files, scratch files, or both.

PGP overwrites the file 26 times.

Key management commands

Editing your user ID or passphrase, or making an existing key your default signing key

You may need to change your passphrase, perhaps because someone looked over your shoulder while you typed it on the keyboard. You may need to change your user ID, because you changed your name or your email address. You may need to add a second or third user ID to your key, because you are known by more than one name, email address, or job title. PGP lets you attach more than one user ID to your key, any one of which can be used to look up your key on the key ring. You may also need to make an existing key your default signing key.

To edit your userid or passphrase for your secret key, or to make an existing key your default signing key, use the following command:

pgp -ke <*your_userid*> [*keyring*]

PGP prompts you for a new user ID or a new passphrase.

If you edit your user ID, PGP actually adds a new user ID, without deleting the old one. If you want to delete an old user ID, you must do that in a separate operation.

If you elect to use the key as an ultimately-trusted introducer, you can make the key your default signing key.

The optional [keyring] parameter, if specified, must be a public keyring, not a secret keyring. The userid field must be your own userid, which PGP knows is yours because it appears on both your public keyring and your secret keyring. Both keyrings are updated, even though you only specified the public keyring.

You can also use the -ke command to edit the trust parameters for a public key. For details, see "Editing the trust parameters for a public key" on page 40.

Editing the trust parameters for a public key

To edit the trust parameters for a public key on your public key ring, enter the following command:

pgp -ke <*userid*> [*keyring*]

The optional [keyring] parameter, if specified, must be a public keyring, not a secret keyring.

Verifying the contents of your public key ring

PGP automatically checks any new keys or signatures on your public key ring and updates all the trust parameters and validity scores. In theory, it keeps all the key validity status information up-to-date as material is added to or deleted from your public key ring.

At some point, however, you may want to explicitly force PGP to perform a comprehensive analysis of your public key ring, checking all the certifying signatures, checking the trust parameters, updating all the validity scores, and checking your own ultimately-trusted key against a backup copy on a write-protected floppy disk. It may be a good idea to do this hygienic maintenance periodically to make sure nothing is wrong with your public key ring.

To force PGP to perform a full analysis of your public key ring, use the -kc (key ring check) command:

pgp -kc

You can also use the following command to make PGP check all the signatures for a single selected public key:

pgp -kc <*your_userid*> [*keyring*]

For information on how to check the backup copy of your own key, see "CERT_DEPTH: Depth of introducers be nested" on page 47.

Verifying a public key over the phone

If you receive a public key from someone that is not certified by anyone you trust, how can you tell if it's really their key? If you know the key's owner and would recognize their voice on the phone, call them and verify the key's fingerprint over the telephone. To do so, both you and the key's owner use the -kvd command to view the key's fingerprint:

pgp -kvc <*userid*> [*keyring*]

This command instructs PGP to display the key with the 32 character digest of the public key components (Diffie-Hellman keys have 40 character fingerprints). Read the fingerprint to the key's owner to see if the fingerprints match.

Using this procedure, you can verify and sign each other's keys with confidence. This is a safe and convenient way to get the key trust network started for your circle of friends.

Note that sending a key fingerprint via email is not the best way to verify the key, because email can be intercepted and modified. It is best to use a different channel than the one that was used to send the key itself. A good combination is to send the key via email, and the key fingerprint via a voice telephone conversation. Some people distribute their key fingerprint on their business cards.

Selecting keys using the key ID

In most cases you enter a user ID or the fragment of a user ID to select a key. However, you can also use the hexadecimal key ID to select a key. To do so, enter the key ID, with a prefix of “0x”, instead of the user ID:

```
pgp -kv 0x67F796C2
```

This command instructs PGP to display all keys that have 67F796C2 in their key IDs.

This feature is particularly useful if you have two different keys from the same person, with the same user ID. You can pick the correct key by specifying the specific key ID.

PGPPASS: Store your passphrase

When PGP needs a passphrase to unlock a secret key, PGP prompts you to enter your passphrase. Use the PGPPASS environment variable, entered on the command line, to store your passphrase. When PGP requires a passphrase, it attempts to use the stored passphrase. If the stored passphrase is incorrect, PGP recovers by prompting you for the correct passphrase.

```
SET PGPPASS=zaphod beeblebrox for president
```

The above example would eliminate the prompt for the passphrase if the passphrase was “zaphod beeblebrox for president”.

This feature is convenient if you regularly receive a large number of incoming messages addressed to your secret key, eliminating the need for you to repeatedly type in your passphrase.

The safest way to use this feature is to enter the command each time you boot your system, and erase it or turn off your machine when you are done. Do not use this feature in an environment where someone else may have access to your machine.

Passing your passphrase from another application

PGP includes a command line option, -z, that you can use to pass your passphrase into PGP from another application. This option is designed primarily to invoke PGP from inside an email package.

The passphrase follows the -z option on the command line. Use this feature with caution.

PGPPASSFD

The passphrase file descriptor. If this environment variable is set to zero (0), PGP uses the first text line from stdin as the password.

PGP's Configuration File 4

Learning about PGP's configuration file: pgp.cfg

PGP stores a number of user-defined parameters in the configuration text file, *pgp.cfg*. A configuration file enables you to define flags and parameters (also called environment variables) for PGP, eliminating the need to define these parameters in the PGP command line.

Use these configuration parameters to perform the following tasks as well as many others:

- Control where PGP stores its temporary scratch files.
- Adjust PGP's level of skepticism when it evaluates a key's validity based on the number of the key's certifying signatures.

Configuration parameters may be assigned integer values, character string values, or on / off values; the type of values depends on the type of parameter. PGP includes a sample configuration file for your review.

The following rules apply to the configuration file:

- Blank lines are ignored.
- Characters that follow the comment character, #, are ignored.
- Keywords are not case-sensitive.

The following is a short sample fragment of a typical configuration file:

```
# TMP is the directory for PGP scratch files, such as a RAM disk.
TMP = "e:\"     # Can be overridden by environment variable TMP.
Armor = on      # Use -a flag for ASCII armor whenever applicable.
# CERT_DEPTH is how deeply introducers may introduce introducers.
cert_depth = 3
```

Under the following conditions, PGP uses default values for the configuration parameters:

- Configuration parameters are not defined.
- Configuration file does not exist.
- PGP cannot find the configuration file.

Note that it is also possible to set these same configuration parameters directly from the PGP command line, by preceding the parameter setting with a “+” (plus) character. For example, the following two PGP commands produce the same effect:

pgp -e +armor=on message.txt smith

pgp -ea message.txt smith

For the location of pgp.cfg, please refer to “Location of PGP files” on page 17.

The remainder of this chapter summarizes PGP’s configuration parameters. Parameters appear in alphabetical order.

ADKKEY: Encrypt with Additional Decryption Key (ADK)

ADKKEY = <*userid*> or ADKKEY = <*keyid*>

For example, **ADKKEY = “John Doe”** or **ADKKEY = “0xAB12C34”**

When this parameter is used, all generated keys have an ADK equal to the value of **ADKKEY**, and all outgoing email encrypted to the user’s key is also encrypted to the ADK key identified by this parameter.

If this parameter is used and the encryption key was generated with **ENFORCEADK** set to on, the data is encrypted to the ADK if the key is available. If the key is not available, a warning is displayed.

This parameter can also be set on the command line with the + prefix, for example, +ADKKEY=”John Doe”.

ARMOR: ASCII-armor output

Default setting: ARMOR = off

The configuration parameter ARMOR is equivalent to the -a command line option. If enabled, this parameter causes PGP to emit ciphertext or keys in ASCII-armored format suitable to transport through email channels. Output files are named with the ".asc" extension.

If you intend to use PGP primarily for email purposes, you should turn this parameter on (ARMOR=ON).

ARMORLINES: Size of ASCII armor multipart files

Default setting: ARMORLINES = 0

Most email facilities prohibit messages that are more than 50000 or 65000 bytes. As a result, PGP restricts the number of lines to a file to 720. When PGP creates a large “.asc” ASCII-armored file, the file is broken into smaller multipart files so that it can be sent through email utilities. The smaller files are named with suffixes “.as1”, “.as2”, “.as3”, and so on.

The configuration parameter ARMORLINES specifies the maximum number of lines in each of the smaller files in a multipart “.asc” file sequence. If you set ARMORLINES to zero, PGP does not break the large file into smaller files.

CERT_DEPTH: Depth of introducers be nested

Default setting: CERT_DEPTH = 4

The configuration parameter CERT_DEPTH identifies how many levels deep you can nest introducers to certify other introducers to certify public keys on your public key ring.

For example, If CERT_DEPTH is set to 1, there can only be one layer of introducers below your own ultimately-trusted key. If that is the case, you are required to directly certify the public keys of all trusted introducers on your key ring. If you set CERT_DEPTH to 0, you could have no introducers at all, and you would have to directly certify each and every key on your public key ring to use it. The minimum CERT_DEPTH is 0, the maximum is 8.

CLEARSIG: Signed message readable with human eyes

Default setting: CLEARSIG = on

Use the CLEARSIG parameter to generate a signed message that can be read with human eyes, without the aid of PGP. The recipient must still use PGP to verify the signature.

Unencrypted PGP signed messages have a signature certificate prepended in binary form. The signed message is compressed, rendering the message unreadable to human eyes, even though the message is not encrypted.

To send this binary data through a 7-bit email channel, PGP applies ASCII-armor (see the ARMOR parameter). Even if PGP did not compress the message, the ASCII armor renders the message unreadable to human eyes. The recipient must first use PGP to strip the armor off the message, and then decompress the message before reading it.

If the original plaintext message is in text, not binary form, you can use the CLEARSIG parameter to send a signed message through an email channel; the signed message is not compressed, and the ASCII armor is applied to the binary signature certificate, but not to the plaintext message. The CLEARSIG parameter makes it possible to generate a signed message that can be read with human eyes, without the aid of PGP (again, the recipient still needs PGP to verify the signature).

The CLEARSIG flag is preset to "on". To enable the full CLEARSIG behavior, the ARMOR and TEXTMODE flags must also be turned on. Set ARMOR=ON (or use the -a option), and set TEXTMODE=ON (or use the -t option). If CLEARSIG is set to off in your configuration file, you can turn it back on again directly on the command line:

```
pgp -sta +clearsig=on message.txt
```

Note that since this method only applies ASCII armor to the binary signature certificate, and not to the message text itself, there is some risk that the unarmored message may suffer some accidental molestation while enroute. This can happen if it passes through an email gateway that performs character set conversions, or in some cases extra spaces may be added to or stripped from the ends of lines. If this occurs, the signature will fail to verify, which may give a false indication of intentional tampering.

When PGP calculates the signature for text in CLEARSIG mode, trailing blanks are ignored on each line.

COMMENT: ASCII armor comment

Default setting: COMMENT = ""

ASCII Armor Comment appears in all armored output as a Comment header just beneath the Version header.

COMPATIBLE: Enable user-interface compatibility with PGP 2.6.2

Default setting: COMPATIBLE=off

The configuration parameter COMPATIBLE enables user-interface compatibility with PGP 2.6.2. You may require this feature for interoperation with scripts that parse the output or otherwise interact with PGP dialogues.

To activate this feature, add the following line to the configuration file, pgp.cfg:

```
COMPATIBLE=on
```

COMPLETES_NEEDED: Number of completely trusted introducers needed

Default setting: COMPLETES_NEEDED = 1

The configuration parameter COMPLETES_NEEDED identifies the minimum number of completely trusted introducers required to fully certify a public key on your public key ring.

COMPRESS: Compression before encryption

Default setting: COMPRESS = on

The configuration parameter COMPRESS enables or disables data compression before encryption. It is used mainly to debug PGP. Under normal circumstances, PGP attempts to compress the plaintext before it encrypts it. Do not change this setting.

CIPHERNUM

Use to specify the symmetric cipher to use. Values areas follows:

kPGPCipherAlgorithm_IDEA = 1

kPGPCipherAlgorithm_3DES = 2

kPGPCipherAlgorithm_CAST5 = 3

This is specified so that the application does not need to know the values coded into the SDK. There may be more algorithms added in future releases.

ENCRYPTTOSELF: Encrypt to self

Default setting: ENCRYPTTOSELF = off

Use this variable to instruct PGP to add MYNAME to recipients.

ENFORCEADK: Enforce encrypting to an ADK

Default setting: ENFORCEADK = off

Use in conjunction with the **ADDKEY** parameter (encrypting to a key with an Additional Decryption Key (ADK)).

Set **ENFORCEADK** to on if you want generated keys to have the enforce ADK bit set.

When encrypting a message to a user (for example, User1), If that user's key has an ADK associated with it (for example, User2), *PGP Command Line* will try to encrypt to the ADK key (that is, User2) as well as to User1's key. If the ADK key isn't on the keyring, User1's key enforce ADK bit is set, and if **ENFORCEADK** = on, *PGP Command Line* generates an error message.

When **ENFORCEADK** is set to off and the ADK key is not present on the user's keyring, *PGP Command Line* displays a warning message.

This parameter can also be set on the command line using the + prefix, for example, **+ENFORCEADK=on**.

FASTKEYGEN: Fast key generation

Default setting: FASTKEYGEN = on

Use to specify fast key generation.

HASHNUM

A number that describes the hash algorithm used. Values are of type PGPHashAlgorithm:

kPGPHashAlgorithm_MD5 = 1

kPGPHashAlgorithm_SHA = 2

kPGPHashAlgorithm_RIPEMD160 = 3

This is specified so that the application does not need to know the values coded into the SDK. There may be more algorithms added to future releases.

INTERACTIVE: Confirmation for key adds

Default setting: INTERACTIVE = off

Use this variable to instruct PGP to ask for confirmation when you add a key file with multiple keys to your key ring. When this variable is set to "on", PGP asks for confirmation for each key in the key file before adding it to your key ring.

KEYSERVER_URL

Default setting: KEYSERVER_URL = ""

Identifies the URL of the default key server, for example, `ldap://certserver.pgp.com`.

MARGINALS_NEEDED: Number of marginally trusted introducers needed

Default setting: MARGINALS_NEEDED = 2

The configuration parameter MARGINALS_NEEDED identifies the minimum number of marginally trusted introducers required to fully certify a public key on your public key ring.

MYNAME: Default user ID for signatures

Default setting: MYNAME = “”

The configuration parameter MYNAME specifies the default user ID to use to select the secret key for making signatures. If MYNAME is not defined, PGP uses the most recent secret key you installed on your secret key ring. You can override this setting by using the -u option to specify a user ID on the PGP command line.

PAGER: Shell command to display plaintext output

Default setting: PAGER = “”

PGP’s **-m** option lets you view decrypted plaintext output on your screen, one screen at a time, without writing the output to a file.

PGP includes a built-in page display utility. If you prefer to use a different page display utility, use the PAGER parameter to identify the utility. The PAGER parameter specifies the shell command PGP uses to display a file.

Note that if the sender specified that a file is for your eyes only, PGP always uses its own built-in display function.

For further details, see “Decrypting a message and viewing plaintext output on your screen” on page 38.

PGP_MIME

Default setting: PGP_MIME = off

Use to specify compatibility with PGP-MIME.

PGP_MIMEPARSE

Default setting: PGP_MIMEPARSE = off

Use to instruct PGP to try to parse MIME body parts.

PUBRING: Filename for your public keyring

Default setting: PUBRING = "%PGPPATH%/pubring.pkr" on UNIX

%USERPROFILE%\Application Data\pgp\pubring.pkr on NT

You may want to keep your public keyring in a directory separate from your PGP configuration file (that is, the directory specified by your PGPPATH environment variable). Use the PUBRING parameter to identify the full path and filename for your public keyring.

You can also use this feature on the command line to specify an alternative keyring.

RANDOMDEVICE

Default setting: RANDOMDEVICE = /dev/random

UNIX only. Identifies the system entropy pool, /dev/random. PGP tries to open this device to acquire entropy, and if that fails, will try to acquire entropy from user keystrokes. Not applicable to Windows NT.

RANDSEED: Filename for random number seed

Default setting: RANDSEED = "%PGPPATH%/randseed.rnd" on UNIX

"%SYSTEMROOT% /randseed.rnd" on Windows NT

The random number seed file, randseed.rnd, is used to generate session keys. You may want to keep your random number seed file in a more secure directory or device (this file generally resides in the directory specified by your PGPPATH environmental variable). Use the RANDSEED parameter to identify the full path and filename for your random seed file.

SECRING: Filename for your secret keyring

Default setting: SECRING = "%PGPPATH%/secring.skr"

You may want to keep your secret keyring in a directory separate from your PGP configuration file (that is, the directory specified by your PGPPATH environmental variable). Use the PUBRING parameter to identify the full path and filename for your secret keyring.

SHOWPASS: Echo passphrase to user

Default setting: SHOWPASS = off

PGP does not let you see your passphrase as you type it. This makes it harder for someone to look over your shoulder while you type and learn your passphrase. However, you may have problems typing your passphrase without seeing what you are typing. In addition, you may be typing in the privacy of your own homes.

The configuration parameter SHOWPASS enables PGP to echo your typing during passphrase entry.

TMP: Directory pathname for temporary files

Default setting: TMP = “”

The configuration parameter TMP specifies what directory PGP uses for temporary scratch files. If TMP is undefined, the temporary files are written in the current directory. If the shell environmental variable TMP is defined, PGP stores temporary files in the named directory.

TEXTMODE: Assume plaintext is a text file

Default setting: TEXTMODE = off

The configuration parameter TEXTMODE is equivalent to the -t command line option. If enabled, this parameter causes PGP to assume the plaintext is a text file, not a binary file, and converts the plaintext to “canonical text” before encrypting it. Canonical text has a carriage return and a linefeed at the end of each line of text.

This parameter is automatically turned off if PGP detects that the plaintext file contains non-text binary data. If you intend to use PGP primarily for email purposes, you should turn TEXTMODE=ON.

For further details, see “Sending ASCII text files to different machine environments” on page 36.

TZFIX: Timezone adjustment

Default setting: TZFIX = 0

UNIX only. PGP includes timestamps for keys and signature certificates in Greenwich Mean Time (GMT). When PGP asks the system for the time of day, the system should give the time in GMT. However, on some improperly configured systems, the system time is returned in US Pacific Standard Time time plus 8 hours.

The configuration parameter TZFIX specifies the number of hours to add to the system time function to get GMT. If your operating system does not give time in GMT, use TZFIX to adjust the system time to GMT.

For Los Angeles:	SET TZ=PST8PDT
For Denver:	SET TZ=MST7MDT
For Arizona:	SET TZ=MST7

(Arizona does not use 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
For Moscow:	SET TZ=MSK-3MSD
For Auckland:	SET TZ=NZT-13

VERBOSE: Quiet, normal, or verbose messages

Default setting: VERBOSE = 1

The VERBOSE variable controls the amount of detail you receive from PGP diagnostic messages. The settings are as follows:

0 - Displays only queries and errors (that is, prompts the user for input and displays errors when they occur).

1 - Normal default setting. Displays a reasonable amount of detail in diagnostic or advisory messages.

2 - Displays maximum information, usually to help diagnose problems in PGP. Not recommended for normal use.

Exit And Error Codes A

The tables in this appendix identify PGP’s exit and error codes.

General Errors

Error	Description
0	Exit OK, no error
1	invalid file
2	file not found
3	unknown file
4	batchmode error
5	bad argument
6	process interrupted
7	out of memory error

Keyring Errors

Error	Description
10	key generation error
11	non-existing key error
12	keyring add error
13	keyring extract error
14	keyring edit error
15	keyring view error
16	keyring removal error
17	keyring check error
18	key signature error or key signature revoke error
19	key signature removal error

Encode Errors

Error	Description
20	signature error
21	public key encryption error
22	encryption error
23	compression error

Decode Errors

Error	Description
30	signature check error
31	public key decryption error
32	decryption error
33	decompression error

Index

Symbols

.asc file 25

A

-a 25

add a public or secret key file's contents to your public or secret key ring 31

add items to a group 26

add keys to the keyring 26

ADKKEY 29, 46

ARMOR 46

ARMORLINES 47

ASCII armor comment 48

ASCII-armor output 46

ASCII-armored format 25, 35

assume plaintext is a text file 53

attackers

    protecting against 22

B

BAKRING 47

BATCHMODE 33

binary data files 35

C

-c 25

CERT_DEPTH 47

certifying

    public keys 14

check signatures 26

check the signature integrity of a signed file 28

CIPHERNUM 49

CLEARSIG 47

COMMENT 48

COMPATIBLE 18, 48

COMPLETES_NEEDED 49

COMPRESS 49

compression before encryption 49

confirmation for key adds 50

copy a key from your public or secret key ring 31

create a ciphertext file in ASCII-armored-64 format 30

create a key pair 27

create a plaintext ASCII file 30

create a signature certificate that is detached from the document 32

creating

    key pairs 20

creating separate signature certificate and text files 37

Customer service

    contacting x

D

decrypt a message 28

decrypt a message and recover the original plaintext filename 28

decrypt a message and view plaintext output on your screen 28

decrypt a message, read from standard input and write to standard output 28

decrypt an ASCII-armored message 28

decrypting

    email 15

decrypting a message and renaming the plaintext filename output 38

decrypting a message and viewing plaintext output on your screen 38

decrypting ASCII-armored messages 36

default signing key 40
default user ID for signatures 51
depth of introducers be nested 47
directory pathname for temporary files 53
disable a key 27
display all certifying signatures attached to each key 31
display all the keys in a specific key ring filename 31
display plaintext output 25
display the contents of your public key ring 31
display the contents of your public key ring and check the certifying signatures 31
display the fingerprint of a public key 31
distributing
    public keys 14
    your public keys 23

E

-e 25
echo passphrase to user 53
edit a set of keys 26
edit the trust parameters for a public key 32
edit the userid or passphrase for your secret key 32
editing the trust parameters for a public key 40
editing your user ID or passphrase 40
eliminating confirmation questions 34
email
    decrypting 15
    encrypting 15
    signing 15
    verifying 15
encrypt a message for any number of recipients 28
encrypt a message for viewing by recipient only 29
encrypt a plaintext file with conventional cryptography only 28
encrypt a plaintext file with the recipient's public key 28
encrypt conventionally 25
encrypt to self 49
encrypt using public key encryption 25
encrypt with ADK 29, 46
encrypting
    email 15
encrypting binary data 35
ENCRYPTTOSELF 49
enforce encrypting to an ADK 49
ENFORCEADK 49
error codes 55
exchanging
    public keys 14
exit codes 55
extract keys from the keyring 26

F

-f 25
fast key generation 50
FASTKEYGEN 50
file management commands 38
filename for random number seed 52
filename for your public keyring 52
filename for your secret keyring 52
filtering 34
FORCE 34

G

-g 25 to 26
generate a key 26
generating
    key pairs 20
get a key from the key server and put the key on your keyring 31

H

-h 25

HASHNUM 50

help on group options 25 to 26

help on key options 25 to 26

HOME 50

I

INTERACTIVE 50

K

-k 25 to 26

key management commands 40

key pairs
- creating 14, 20
- description of 20
- generating 20
- making 20

keyrings
- overview of 13

keys
- backing up 22
- distributing 23
- generating 20
- overview of 19
- protecting 22
- saving 22

KEYSERVER_URL 50

-kx command 21

M

-m 25

make key default signing key 40

making
- key pairs 20

managing signature certificates 37

MARGINALS_NEEDED 51

MYNAME 51

N

Network Associates
- contacting
  - Customer Service x
- training x

number of completely trusted introducers needed 49

number of marginally trusted introducers needed 51

O

-o 25

overviews
- key concepts 19
- keyrings 13
- private keys 13

P

-p 25

PAGER 51

passing your passphrase from another application 43

passphrases
- suggestions for 21

PGP exit status codes 34

pgp -h 32

pgp -kd 27

PGP Key Wizard
- using to create key pairs 20

pgp -kg 20, 27

PGP 2.6.2 18

pgp.cfg 45

PGP_MIME 51

PGP_MIMEPARSE 51

PGPkeys window
- creating key pairs with 20

PGPPASS 42

PGPPASSFD 43

PGPPATH 18

private and public key pairs
  creating 14
private keys
  creating 14
    key pairs 14
  overview 13
  protecting 22
  storing 22
protecting
  your keys 22
public keys
  certifying 14
  creating 14
    key pairs 14
  distributing your 23
  exchanging with other users 14
  giving to other users 14
  protecting 22
  storing 22
  trading with other users 14
  validating 14
PUBRING 52
pubring.pkr 22

R

RANDOMDEVICE 52
RANDSEED 52
receiving separate signature certificate and text files 37
reenable a key 27
remove a key or a userid from your public key ring 32
remove items from a group 26
remove keys from the keyring 26
remove selected signatures from a userid on a keyring 32
remove signatures attached to keys on the keyring 26
revoke or disable keys on the keyring 26
revoke signatures attached to keys on the keyring 26
revoke your key 27

S

-s 25
saving
  keys 22
SECRING 52
secring.skr 22
selecting keys using the key ID 42
sending a public key in ASCII-armored format 36
sending ASCII text files to different machine environments 36
sending binary data files in ASCII-armored format without encryption or signature 35
set pathname for PGP 18
shell command to display plaintext output 51
SHOWPASS 53
sign 25
sign a plaintext ASCII text file 30
sign a plaintext file with your secret key 30
sign a plaintext file with your secret key and encrypt it with the recipient's public key 30
sign and certify someone else's public key on your public key ring 32
sign keys on the keyring 26
signature certificates 37
signed message readable with human eyes 47
signing
  email 15
signing a file without encrypting 39
size of ASCII armor multipart files 47
starting PGP 17
store your passphrase 42
storing
  keys 22
storing signed files 39
summary of commands 25, 32
suppressing unnecessary questions 33

T

-t 25

tampering

protecting your keys against 22

TEXTMODE 53

TMP 53

training for Network Associates products x

scheduling x

transmitting binary data 35

U

-u 25

ultimately-trusted introducer 40

UNIX-style filter 34

UNIX-style filter mode 25

V

validating

public keys 14

VERBOSE 54

verifying

email 15

verifying a public key over the phone 41

verifying the contents of your public key ring 41

view a group 26

view a group and the keys it contains 26

view keys and signatures on the keyring 26

view keys on the keyring 26

view the fingerprints of a set of keys 26

W

-w 25

wipe 25

wipe out original plaintext file 29

wiping your disk 39

Z

-z 25