PGP®
for Personal Privacy

For
Macintosh

User's Guide

Pretty Good Privacy, Inc.

Copyright © 1997 Pretty Good Privacy, Inc. All rights reserved.
9-97. Printed in the United States of America.
PGP for Personal Privacy, Version 5.5.1 for Macintosh

Record the serial number from your License Agreement in the space provided below:

PRETTY GOOD PRIVACY, INC.
2121 South El Camino Real, Suite 902
San Mateo, CA 94403
(650) 631-1747 voice
(650) 572-1932fax
info@pgp.com
http://www.pgp.com

LIMITED WARRANTY

Limited Warranty. Pretty Good Privacy, Inc. ("PGP") warrants that the Software Product will perform substantially in accordance with the accompanying written materials for a period of sixty (60) days from the date of original purchase. To the extent allowed by applicable law, implied warranties on the Software Product, if any, are limited to such sixty (60) day period. Some jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

Customer Remedies. PGP's and its suppliers' entire liability and your exclusive remedy shall be, at PGP's option, either (a) return of the purchase price paid for the license, if any or (b) repair or replacement of the Software Product that does not meet PGP's limited warranty and which is returned at your expense to PGP with a copy of your receipt. This limited warranty is void if failure of the Software Product has resulted from accident, abuse, or misapplication. Any repaired or replacement Software Product will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. Outside the United States, neither these remedies nor any product support services offered by PGP are available without proof of purchase from an authorized international source and may not be available from PGP to the extent they subject to restrictions under U.S. export control laws and regulations.

NO OTHER WARRANTIES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AND EXCEPT FOR THE LIMITED WARRANTIES SET FORTH HEREIN, THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" AND PGP AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, CONFORMANCE WITH DESCRIPTION, TITLE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM JURISDICTION TO JURISDICTION.

LIMITATION OF LIABILITY. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL PGP OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL OR EXEMPLARY DAMAGES OR LOST PROFITS WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE PRODUCT OR THE FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF PGP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, PGP'S CUMULATIVE AND ENTIRE LIABILITY TO YOU OR ANY OTHER PARTY FOR ANY LOSS OR DAMAGES RESULTING FROM ANY CLAIMS, DEMANDS OR ACTIONS ARISING OUT OF OR RELATING TO THIS AGREEMENT SHALL NOT EXCEED THE PURCHASE PRICE PAID FOR THIS LICENSE. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Preface

This book describes how to use PGP for Personal Privacy, Version 5.5 for Macintosh. PGP Version 5.5 has some new features, which are described in Chapter 1.

Conventions used in this document

Types of notes:

NOTE: Notes provide additional information on using PGP—for example, if you are not using PGP/MIME, you must encrypt any files you want to send as attachments from the Finder before sending your message

TIP: Tips provide guidelines for using PGP effectively—for example, how to create a useful passphrase.

ALERT Alerts provide information to prevent loss of data—for example, if you are sending your key to colleagues who are using PCs, enter a name of up to eight characters and three additional characters for the file type extension (for example, e-mail.txt).

For more information

There are several ways to find out more about PGP and its products.

PGP Web site

PGP provides information about our products, the PGP organization, product updates, and related topics, such as Privacy Matters, at the PGP Web site. Visit us at www.pgp.com.

Registration information

Users are asked to register online. This information is for our internal use only and will enable us to serve you better in the future. This data will remain confidential -- we respect your privacy (that's why we do what we do). See our registration page at the following URL:

http://www.pgp.com/products/online-register.cgi

Support

To get Technical Support for your PGP product, see our Technical Support web site :http://www.pgp.com/service/ or e-mail us at PGPSupport@pgp.com.

Service is available for PGP products by sending e-mail to PGPservice@pgp.com or by contacting the PGP Web site at http://www.pgp.com/service/

Service is available for the following issues:

- Product Returns
- Download Problems
- Export Control Issues
- Marketing Questions
- Product Literature Fulfillment
- Order Tracking and Information

Recommended Introductory Readings

Bacard, Andre *Computer Privacy Handbook*, Peachpit Press, 1995.

Garfinkel, Simson *Pretty Good Privacy,* O'Reilly & Associates, 1995.

Schneier, Bruce *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition,* John Wiley & Sons, 1996.

Schneier, Bruce *E-mail Security,* John Wiley & Sons, 1995.

Stallings, William *Protect Your Privacy,* Prentice-Hall, 1994.

Other Readings:

Lai, Xuejia, "On the Design and Security of Block Ciphers," Institute for Signal and Information Processing, ETH-Zentrum, Zurich, Switzerland, 1992.

Lai, Xuejia, Massey, James L., and Murphy, Sean " Markov Ciphers and Differential Cryptanalysis," Advances in Cryptology—EUROCRYPT'91.

Rivest, Ronald "The MD5 Message Digest Algorithm" *MIT Laboratory for Computer Science*, 1991.

Wallich, Paul "Electronic Envelopes" *Scientific American*, Feb. 1993, page 30.

Zimmermann, Philip "A Proposed Standard Format for RSA Cryptosystems" *Advances in Computer Security*, Vol. III, edited by Rein Turn Artech House, 1988.

Your feedback is welcome

We continually enhance PGP and welcome customer feedback as we design new versions. We appreciate your interest in PGP and your thoughts on product content and functionality, especially as we plan features to enhance the products for corporate settings. Feedback like yours help us to develop richer and easier-to-use software and services. While we cannot incorporate all suggestions, but we will give your input serious consideration as we develop future products.

If you would like to provide input, please send e-mail to mac-doc@pgp.com.

Contents

Chapter 1

Introducing PGP for Personal Privacy

Welcome to PGP! With PGP® for Personal Privacy, you can easily and securely protect the privacy of your e-mail messages and file attachments by encrypting them so that only the intended recipients can read them. You can also digitally sign messages and files, which ensures their authenticity. A signed messages verifies that the information in it has not been tampered with in any way.

What's new in PGP Version 5.5

PGP Version 5.5 includes these new features:

- You can create recipient groups, in which you select a group of people's keys and encrypt mail to all of them simultaneously.
- New key server integration capabilities that you can use to automatically store, search, and synchronize keys.
- A new Key Search Window that you can use to locate keys on remote servers with the same user interface that you use to search your keyring.
- The PGP Version 1.0 Policy Management Agent for SMTP, which enforces your organization's e-mail encryption policy. The PGP SMTP Policy Agent works in conjunction with a standard SMTP mail server to ensure that incoming and outgoing e-mail adheres to the policies enforced by a given site.

- A PGP Wipe feature that overwrites files so that they cannot be recovered with software tools
- A configurable View menu that provides information about the keys on your keyring.
- New key-signing capabilities that allow you to choose whether the owner of the key that you are signing is a trusted introducer or a meta-introducer. You can also choose whether your signature is exportable, which means it is exported with your key, or nonexportable.

What’s new in PGP Version 5.5 documentation

This guide contains information about PGP Version 5.5. The PGP documentation has the following new features:

- The manuals for user products contain conceptual information and procedures for using PGP. Online help for PGP user products is written specifically for each plug-in, and contains more detailed information on how to use PGP with each plug-in.
- PGP Version 5.5 for Macintosh includes online help, and an electronic copy of the documentation in .pdf format is available on the PGP Version 5.5 CD ROM.

Using PGP

One of the most convenient ways to use PGP is through one of the popular e-mail applications supported by the PGP plug-ins. With these applications, you can encrypt and sign as well as decrypt and verify your messages while you are composing and reading your mail with a simple click of a button.

If you are using an e-mail application that is not supported by the plug-ins, you can easily encrypt the text of the message using PGPmenu. In addition, if you need to encrypt or decrypt file attachments, you can do so directly from the Finder by choosing the appropriate menu option. You can also use PGP to encrypt and sign files on the hard disk of your computer for secure storage, and to securely wipe files from your hard disk so that sensitive data can’t be retrieved with disc recovery software.

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* that you freely exchange with other PGP users. Both your private and your public keys are stored in keyring files, which are accessible from the PGPkeys window. It is from this window that you perform all your key management functions.

To send someone a private e-mail message, you use a copy of that person's public key to encrypt the information, which only they can decipher by using their private key. Conversely, when someone wants to send you encrypted mail, they use a copy of your public key to encrypt the data, which only you can decipher by using a copy of your private key. You can also use PGP to encrypt or sign files that are stored on your computer, to authenticate that they have not been altered.

You also use your private key to sign the e-mail you send to others or to sign files to authenticate them. The recipients can then use their copy of your public key to determine if you really sent the e-mail and whether it has been altered while in transit. When someone sends you e-mail with their digital signature, you use a copy of their public key to check the digital signature and to make sure that no one has tampered with the contents.

With the PGP program you can easily create and manage your keys and access all of the functions for encrypting and signing as well as decrypting and verifying your e-mail messages, files, and file attachments.

The rest of 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.

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 have the option of creating a new key pair immediately after you have finished the PGP installation procedure, or you can do so at any time by opening the PGPkeys application.

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 e-mail 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.

Validate your 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. 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.

Encrypt and sign your e-mail and files

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

- If you are using an e-mail application supported by the plug-ins, you can encrypt and sign your messages by selecting the appropriate options from your application's tool bar.
- If your e-mail application is not supported by the plug-ins, you can copy the message to the clipboard and perform the appropriate functions from there. You can also encrypt and sign files from the Finder before attaching them to your e-mail; encrypt files to store them securely on your computer; and sign files to verify that they have not been tampered with.

Decrypt and verify your e-mail and files

When someone sends you encrypted e-mail, you can unscramble 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.

- If you are using an e-mail application that is supported by the plug-ins, you can decrypt and verify your messages by selecting the appropriate options from your application’s tool bar.
- If your e-mail application is not supported by the plug-ins, you can copy the message to the clipboard and perform the appropriate functions from there. If you want to decrypt and verify file attachments, you can do so from the Finder. You can also decrypt encrypted files stored on your computer, and verify signed files to ensure that they have not been tampered with.

Wipe files

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

Chapter 2

Getting Started

This chapter explains how to run PGP and provides a quick overview of the procedures you will normally follow in using the product. It also contains a table of the icons used with PGPkeys.

System requirements

These are the system requirement for installing PGP Version 5.5:

- Macintosh IIci or later model with 68030 or higher
- System software 7.5.3 or later
- 8 MB RAM
- 10 MB hard disk space
- 68K Macs must be running Apple's CFM 68K 4.0 or higher. The PGP installer installs this if necessary.

Compatibility with other versions

PGP has gone through many revisions since it was released by Phil Zimmermann as a freeware product in 1991, and it is estimated that there are now over 4 million copies in circulation. Although this version of PGP represents a significant rewrite of the original program and incorporates a completely new user interface, it has been designed to be compatible with earlier versions of PGP. This means that you can exchange secure e-mail with people who are still using these older versions of the product:

- PGP 2.6 (Released by MIT)

- PGP 2.7.1 for the Macintosh (Released by ViaCrypt)
- PGP 4.0 (Released by ViaCrypt)
- PGP 4.5 (Released by PGP, Inc.)
- PGP for Personal Privacy, Version 5.0
- PGP for Business Security Version 5.5

About PGP for Personal Privacy

PGP for Personal Privacy Version 5.5 supports the use of two types of keys: RSA and Diffie-Hellman. Using PGP Version 5.5 for Personal Privacy, you can only create new Diffie-Hellman keys. You can use existing RSA keys but you cannot generate new RSA keys.

Before PGP Version 5.0, RSA keys were the only option. PGP now offers keys based on Diffie-Hellman encryption and DSS digital signature technologies. The DSS portion of the key is used for signing and the Diffie-Hellman part is used for encryption.

However, if you need to exchange e-mail or files with others who are using keys created using RSA-encryption algorithms, you will use an existing set of RSA keys. If you are exchanging e-mail or files with someone who is using an older version of PGP, they should upgrade to one of newer PGP versions to take advantage of the better user interface and features.

If you are encrypting e-mail or files to multiple recipients, some of whom have RSA keys and others of whom have DSS/Diffie-Hellman keys, the e-mail is encrypted using the appropriate key for each individual. However, in order for users of older versions of PGP to be able to decipher or verify this e-mail, they will first need to upgrade to one of the patched versions that remove this limitation. These previous versions are available as free upgrades from PGP.

PGP for Business Secrity Version 5.5 contains some features of which you may become aware if you correspond with people who are using it. PGP for Business Security offers a Corporate Signing key and Incoming and Outgoing Additional Decryption Keys.

A Corporate Signing Key is a public key that is designated as the organization-wide key that all users can trust to sign other keys.

Additional Decryption Keys are keys that allows security officers in organizations, under certain circumstances, to decrypt messages that have been sent to or from people within their organization. The Incoming Additional Decryption Key causes encrypted mail sent to people in an organization to also be encrypted to the Incoming Additional Decryption Key. The Outgoing Additional Decryption Key causes encrypted mail sent from people in an organization to also be encrypted to the Outgoing Additional Decryption Key. You can use Key Properties to determine if a key has an Additional Decryption Key associated with it and you are alerted if you are encrypting to a key with an Additional Decryption Key.

Using PGP/MIME

PGP/MIME is a standard for some of the plug-ins that integrate PGP functions directly into popular e-mail applications. If you are using an e-mail application that is supported by one of the plug-ins offering PGP/MIME, you will be able to encrypt and sign as well as decrypt and verify your e-mail messages and file attachments automatically when you send or receive e-mail.

However, check before sending PGP/MIME e-mail to be sure that your recipients are using an e-mail application that supports this standard, or they may have difficulty decrypting and verifying your messages.

You can also encrypt messages and files without using PGP/MIME.

Upgrading from a previous version

If you are upgrading from a previous version of PGP (from either PGP, Inc. or ViaCrypt), you may want to remove the old program files before installing PGP to free up some disk space. However, you should be careful not to delete the private and public keyring files used to store any keys you have created or collected while using the previous version. When you install PGP, you are given the option of retaining your existing private and public keyrings, so you don’t have to go to the trouble of importing all of your old keys. To upgrade from a previous version, follow the appropriate steps listed next.

To upgrade from PGP Version 2.7.1 or 2.6.2

1. Make sure that you have exited all programs currently running on your computer.
2. Make backups of your old PGP keyrings on another volume. Your public keys are stored in “pubring.pgp” and your private keys are stored in “secring.pgp”.

NOTE: You may want to make two separate backups of your keyrings onto two different floppy disks just to be safe. Be especially careful not to lose your private keyring; otherwise you will never be able to decrypt any e-mail messages or file attachments encrypted with the lost keys. Store the keyrings in a secure place where only you have access to them.

3. When you have successfully backed up your old keyrings, remove or archive the (old) PGP software. You have two options here:
   - Manually delete the entire old PGP folder and all of its contents; or
   - Manually delete the old PGP program and archive the remaining files, especially the configuration and keyring files.

NOTE: If you obtain a copy of the newly patched PGP264 version from MIT, your old software will be able to read the RSA keys on the new 5.5 keyrings and will not fail when it encounters the new Diffie-Hellman / DSS format keys.

4. Install PGP 5.5 using the provided Installer.
5. When the Installer asks if you have existing keyrings, click Yes, locate your old keyrings, and follow the instructions to copy those keys to your new PGP 5.5 keyrings.
6. Restart your computer.

Installing PGP Version 5.5

Here are the ways to install PGP 5.5:

- From a CD ROM
- From PGP's web site
- From your company's file server

To install PGP from a CD ROM

1. Start your Macintosh.
2. Insert the CD ROM.
3. Run the Installer.
4. Follow the on-screen prompts.

To install PGP from PGP's Web site

1. Download the PGP program onto your computer's hard drive.
2. Double-click the PGP installation program icon.
3. Follow the on-screen prompts.

Running PGP

PGP works on the data generated by other applications. Therefore the appropriate PGP functions are designed to be immediately available to you based on the task you are performing at any given moment. There are four primary ways to use PGP:

- From PGPmenu
- From within supported e-mail applications
- From the PGPtools window
- From PGPcontextmenu (for Mac OS8 users)

Using PGP from the PGPmenu

You can perform most PGP functions from the Finder or from within most applications by choosing options from PGPmenu, which appears as an icon in the menu bar of the Finder and any other applications you have selected. This feature provides immediate access to the PGP functions regardless of which application you are using and is especially useful if you are using an e-mail application that is not supported by the PGP plug-ins.

While using e-mail or other text-based applications to which you have added PGPmenu, you can encrypt and sign and decrypt and verify text by choosing options from PGPmenu. While in the Finder, you can encrypt, sign, decrypt, verify, and wipe files.

(If you cannot find PGPmenu in one of your applications, you need to add the application from the PGPmenu pane of the Preferences dialog box in the PGPkeys application.)

Opening the PGPkeys application

When you choose PGPkeys from PGPmenu or from the PGP 5.5 folder, the PGPkeys application opens, showing the private and public key pairs that you have created for yourself as well as any public keys of other users that you have added to your public keyring. (If you have not already created a new key pair, the PGP Key Generation Wizard leads you through the necessary steps. However, before going through the process of creating a new key pair, you may wish to read Chapter 3 for complete details about the various options.)

From the PGPkeys window you can create new key pairs and manage all of your other keys. For instance, this is where you examine the attributes associated with a particular key, specify how confident you are that the key actually belongs to the alleged owner, and indicate how well you trust the owner of the key to vouch for the authenticity of other users' keys. For a complete explanation of the key management functions you perform from the PGPkeys window, see Chapter 6.

Setting PGP preferences

When you choose Preferences from the Edit menu in the PGPkeys application, the Preferences dialog box appears, in which you specify settings that affect how the PGP program functions based on your computing environment.

By clicking the appropriate tab, you advance to the preference settings you want to modify. For a complete explanation of these settings, see Chapter 6.

Getting help

PGPkeys supports Apple Guide help, which can be accessed from the Help menu in Mac OS 8 and the iconic Question M ark menu in Mac OS 7.

Using PGP from supported e-mail applications

If you have one of the popular e-mail applications supported by the PGP plug-ins, you can access the necessary PGP functions by clicking the appropriate buttons in your application’s toolbar. For example, you click the lock icon to indicate that you want to encrypt your message and the quill icon to indicate that you want to sign. Some applications have an icon of both a lock and quill, which lets you do both at once.

When you receive e-mail from another PGP user, you decrypt the message and verify the person’s digital signature by clicking the opened envelope, or by selecting “Decrypt / Verify” from the menu.

Some plug-ins support a key and envelope button, which adds any keys included in the message to your keyring. You can also access the PGPkeys window at any time while composing or retrieving your mail by clicking the double keys button in some plug-ins.

If you are using an e-mail application with one of the plug-ins that supports the PGP / MIME standard, and you are communicating with another user whose e-mail application also supports this standard, both of you can automatically encrypt and decrypt your e-mail messages and any attached files when you send or retrieve your e-mail. All you have to do is turn on the PGP / MIME encryption and signatory functions from the PGP Preferences dialog box.

When you receive e-mail from someone who uses the PGP / MIME feature, the mail arrives with an attached icon in the message window indicating that it is PGP / MIME encoded.

To decrypt the text and file attachments in PGP / MIME encapsulated e-mail and to verify any digital signatures, you simply double-click the opened envelope icon. Attachments are still encrypted if PGP / MIME is not used, but the decryption process is usually more manual for the receiver.

Using PGP from the PGPtools application

If you are using an e-mail application that is not supported by the plug-ins, or if you want to perform PGP functions from within other applications, you can encrypt and sign, decprypt and verify, or securely wipe messages and files directly from the PGPtools window. You can open the PGPtools window in the following ways:

- Open the PGP folder and double-click the PGPtools icon.
- Store an alias of PGPtools in the Apple menu, and choose PGPtools from that menu. You can also store an alias on your desktop.

When the PGPtools window opens, you can begin your encryption work.

If you are working with text, you perform your encryption / decryption, signature / verification, and wiping functions by selecting the text and then dragging it onto the appropriate button in the PGPtools window.

If you are working with files, you can simply drag them to the appropriate button.

Selecting recipients

When you send e-mail to someone whose e-mail application is supported by the PGP plug-ins, the recipient's e-mail address determines which keys to use when encrypting the contents. However, if you enter a user name or e-mail address that does not correspond to any of the keys on your public keyring, or if you are encrypting from PGPmenu or from PGPtools, you must manually select the recipient's public key from the PGP Key Selection dialog box.

To select a recipient's public key, drag the icon representing the key into the Recipients list box and then click OK.

For complete instructions on how to encrypt and sign and decrypt and verify e-mail, see Chapter 4. If you want to encrypt files to store on your hard disk or to send as attachments, see Chapter 5.

Taking shortcuts

Although you will find that PGP is quite easy to use, a number of shortcuts are available to help you accomplish your encryption tasks even quicker. For example, you can perform most PGP functions on files or volumes on your disk using PGPcontextmenu or PGPmenu (for Mac OS 7 users), or by dragging the file or volume and dropping it onto one of the PGPtools icons.

To access PGP functions using PGPcontextmenu

1. Point t-o the file or volume, either in a window or on the desktop.
2. Click once while holding down the Control key.

   The contextual menu appears. PGP appears among the menu options.
3. Choose an option from the PGP menu.

   To close the menu without choosing a command, click outside the menu.

This feature is available on machines running MacOS8 or later and is equivalent to launching and using the features within PGPtools.

PGPmenu works similarly for MacOS7 users.

PGPkeys icon definitions

The following table shows all of the mini-icons used in the PGPkeys window, along with a description of what they represent.

ICON	WHAT IT REPRESENTS
🔑	A pair of gold modern keys represents your Diffie-Hellman/DSS key pair, which consists of your private key and your public key.
🔑	A single gold modern key represents a Diffie-Hellman/DSS public key.
🔑	A pair of blue skeleton keys represents your RSA key pair, which consists of your private key and your public key.
🔑	A single blue skeleton key represents an RSA public key.
🔑	When a key or key pair is dimmed, the keys are temporarily unavailable for encrypting and signing. You can disable a key from the PGPkeys window, which prevents seldom-used keys from cluttering up the Key Selection dialog box.
🔑	A key with a red line through it indicates that the key has been revoked. Users revoke their keys when they are no longer valid or have been compromised in some way. A key with a red X through it represents a corrupted or damaged key.
🔑	A key with a clock indicates that the key has expired. A key’s expiration date is established when the key is created.
✉	An envelope represents the owner of the key and lists the user names and e-mail addresses associated with the key.
○	An empty circle indicates that the key is invalid.

ICON	WHAT IT REPRESENTS
●	A filled circle indicates that they key is valid (green) or has a CMRK (red).
◆	A diamond indicates that they key is valid (green) or has a CMRK (red).
✎ ✒	A pencil or fountain pen indicates the signatures of the PGP users who have vouched for the authenticity of the key. A signature with a red line through it indicates a revoked signature. A signature with a red X through it indicates a bad or invalid signature. A signature with a blue arrow next to it indicates that it is exportable.
▭	An empty bar indicates an invalid key or an untrusted user.
◧	A half-filled bar indicates a marginally valid key or marginally trusted user.
▬	A full bar indicates a completely valid key or a completely trusted user.

Chapter 3

Making and Exchanging Keys

This chapter describes how to generate the public and private key pairs 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 e-mail.

Key concepts

PGP is based on a widely accepted and highly trusted *public key encryption* system 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 e-mail 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 e-mail and to verify their digital signatures.

NOTE: Without going into too much technical detail, you might be interested to know that it is not actually the content of the e-mail that is encrypted using the public key encryption scheme. Instead, the data is encrypted using a much faster single-key algorithm, and it is this single key that is actually encrypted using the recipient's public key. The recipient then uses his or her private key to decrypt this key, which allows the recipient to decipher the encrypted data.

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 certified e-mail 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 PGPkeys window using the PGP Key Generation Wizard, which guides you through the process.

ALERT: 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, you specify the location of your keys when you run the PGPkeys application. You can go to the Key Files pane of the Preferences dialog box and enter the correct path to your existing keys at any time.

To create a new key pair

1. Choose PGPkeys from PGPmenu.

The PGPkeys applucation opens.

2. Choose New from the Keys menu.

The PGP Key Generation Wizard provides some introductory information on the first screen.

3. When you are through reading this information, click Next to advance to the next dialog box.

The PGP Key Generation Wizard asks you to enter your name and e-mail address.

4. Enter your name on the first line and your e-mail address on the second line.

It's not absolutely necessary to enter your real name or even your e-mail address. However, using your real name makes it easier for others to identify you as the owner of your public key. Also, by using your correct e-mail address, you and others can take advantage of the plug-in feature that automatically looks up the appropriate key on your current keyring when you address mail to a particular recipient.

5. Click Next to advance to the next dialog box.

The PGP Key Generation Wizard asks you to specify a size for your new keys.

6. Select a key size from 768 to 3072 bits, or enter a custom key size from 512 to 4096 bits.

NOTE: A custom key size may take a long time to generate, depending on the speed of the computer you are using. Very old 68020 Macintoshes have been known to take over a day to generate a 4096 bit key. The time required to generate a key is non-deterministic, but should not take more than a few minutes using standard key sizes.

The key size corresponds to the number of bits used to construct your digital key. The larger the key, the less chance that someone will be able to crack it, but the longer it takes to perform the decryption and encryption process. 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.

NOTE: When creating a Diffie-Hellman/DSS key, the size of the DSS portion of the key is increased in fixed increments, is less than or equal to the size of the Diffie-Hellman portion of the key, and is limited to a maximum size of 1024 bits. The strength of a DSS 1024 bit signing key is roughly equivalent to a 2048 bit RSA key.

7. Click Next to advance to the next dialog box.

The PGP Key Generation Wizard asks you to indicate when the key pair should expire.

8. Indicate when you want your keys to expire. You can either use the default selection, which is Never, or you can enter a specific date after which the keys will expire.

Once you create a key pair and have distributed your public key to the world, you will probably continue to use the same keys from that point on. However, under certain conditions you may want to create a special pair of keys that you plan to use for only a limited period of time. In this case, when the public key expires, it can no longer be used by someone to encrypt mail for you but it can still be used to verify your digital signature. Similarly, when your private key expires, it can still be used to decrypt mail that was sent to you before your public key expired but can no longer be used to sign mail for others.

9. Click Next to advance to the next dialog box.

The PGP Key Generation Wizard asks you enter a *passphrase*.

10. In the Passphrase dialog box, enter the string of characters or words you want to use to maintain exclusive access to your private key. To confirm your entry, press the Tab key to advance to the next line, then enter the same passphrase again.

11. Normally, as an added level of security, the characters you enter for the passphrase do not appear on the screen. However, if you are sure that no one is watching, and you would like to see the characters of your passphrase as you type, clear the Hide Typing check box.

TIP: Your passphrase should contain multiple words and may include spaces, numbers, and punctuation. 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 upper and lowercase letters. Strong passphrases include upper and lowercase letters, numbers, punctuation, and spaces. The longer your passphrase, and the greater the variety of characters it contains, the more secure it is. Try to include equal quantities of upper and lowercase alphabetic characters, numbers, punctuation marks, and so on. The quality bar attempts to rate the strength of your passphrase compared to the strength of the key being generated. A full bar means that they are roughly equivalent.

12. Click Next to begin the key generation process.

The PGP Key Generation Wizard indicates that it is busy generating your key.

If you have entered an inadequate passphrase, a warning message appears before the keys are generated and you have the choice of accepting the bad passphrase or entering a more secure one before continuing.

If there is not enough random information upon which to build the key, the PGP Random Data dialog box appears. As instructed in the dialog box, move your mouse around and enter a series of random keystrokes until the progress bar is completely filled in. Your mouse movements and keystrokes generate random information that is needed to create a unique key pair.

NOTE: PGP Versions 5.0 and later are contantly gathering random data from many sources on the system, including mouse positions, timings, and keystrokes. If the Random Data dialog box does not appear, it indicates that PGP has already collected all the random data that it needs to create the key pair.

After the key generation process begins, it may take a while to generate the keys. In fact, if you specify a size other than the default values for a Diffie-Hellman / DSS key, the fast key generation option is not used and it may take hours to generate your key at larger sizes. Eventually the PGP Key Generation Wizard indicates that the key generation process is complete.

13. Click Next to advance to the next dialog box.

The PGP Key Generation Wizard indicates that you have successfully generated a new key pair and asks if you want to send your public key to a key server.

14. Specify whether you want your new public key to be sent to the appropriate server for the e-mail domain you entered and then click Next.

When you send your public key to the key server, anyone who has access to that key server can get a copy of your key when they need it. For complete details, see “Distributing your public key,” later in this chapter.

When the key generation process is complete, the final dialog box appears.

15. Click Finish.

A pair of keys representing your newly created keys appears in the PGPkeys window. You will notice that the older RSA keys are blue and the newer Diffie-Hellman / DSS keys are yellow. At this point you can examine your keys by checking their properties and the attributes associated with the keys; you may also want to add other e-mail addresses that belong to you.   See Chapter 6 for details on how to add a new user name.

Protecting your keys

Once you have generated a key pair, it is wise to create a spare pair and put them in a safe place in case something happens to the originals. PGP prompts you to save a backup copy when you close the PGPkeys application after creating a new key pair.

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 keyring (PGP Private Keys) and the public keyring (PGP Public Keys) are stored along with the other program files in the “PGP Keyrings” folder in the “PGP 5.5” folder, but you can save your backups in any location you like.

When you specify that you want to save a backup copy of your keys, the Save As dialog box appears asking you to specify the location of the backup private and public keyring files that are to be created.

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 e-mail 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 get hold of your passphrase from being able to use your private key, you should only store it 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 your private mail.

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. You use the Keys pane of the PGPkeys Preferences dialog box to specify a name and location for your private and public keyring files.

Distributing your public key

After you create your keys, you need to make them available to others so that they can send you encrypted e-mail 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 e-mail 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 e-mail 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.

Making your public key available through a key server

The best method for making your public key available is to place it on a public key server where anyone can access it. That way, people can send you e-mail without having to explicitly request a copy of your key. It also relieves you and others from having to maintain a large number of public keys that you rarely use. There are a number of key servers worldwide, including those offered by PGP, Inc., where you can make your key available for anyone to access.

If you are using PGP for Personal Privacy, your Security Officer will usually pre-configure your keyserver settings so that everything works correctly for your site.

To send your public key to a key server

1. Connect to the Internet.
2. Open the PGPkeys window.
3. Select the icon that represents the public key you want to post on the key server.
4. Choose For Domain from the Send to Server submenu of the Keys menu.

After you place a copy of your public key on a key server, you can tell people who want to send you encrypted mail or to verify your digital signature to get a copy of your key from the server. Even if you don't explicitly point them to your public key, they can get a copy by searching the key server for your name or e-mail address. Many people include the Web address for their public key at the end of their e-mail messages; in most cases the recipient can just double-click the address to access a copy of your key on the server.

If you ever need to change your e-mail address, or if you acquire new signatures, all you have to do to replace your old key is send a new copy to the server; the information is automatically updated. However, you should keep in mind that public key servers are only capable of updating new information and will not allow removal of user names or signatures from your key. If your key is ever compromised, you can revoke it, which tells the world to no longer trust that version of your key. See Chapter 6 for more details on how to revoke a key.

Including your public key in an e-mail message

Another convenient method of delivering your public key to someone is to include it along with an e-mail message.

To include your public key in an e-mail message

1. Open the PGPkeys window.
2. Select your key pair and then choose Copy from the Edit menu.
3. Open the editor you use to compose your e-mail messages, place the cursor in the desired area, and then choose Paste from the Edit menu. In newer e-mail applications, you can simply drag your key from the PGPkeys window into the text of your e-mail message to transfer the key information.

When you send someone your public key, be sure to sign the e-mail. That way, the recipient can verify your signature and be sure that no one has tampered with the information along the way. Of course, if your key has not yet been signed by any trusted introducers, recipients of your signature can only truly be sure the signature is from you by verifying the fingerprint on your key.

Exporting your public key to a file

Another method of distributing your public key is to copy it to a file and then make this file available to the person with whom you want to communicate. There are three ways to copy your public key to a file:

- Select the icon representing your key pair from the PGPkeys window, then choose Export from the Keys menu and enter the name of the file where you want the key to be saved.
- Drag the icon representing your key pair from the PGPkeys window and drop it where you want the key to be saved.
- Select the icon representing your key pair in the PGPkeys window, choose Copy from the Edit menu, and then choose Paste to insert the key information into a text document.

ALERT: If you are sending your key to colleagues who are using PCs, enter a name of up to eight initial characters and three additional characters for the file type extension (for example, e-mail.txt).

Obtaining the public keys of others

Just as you need to distribute your public key to those who want to send you encrypted mail or to verify your digital signature, you need to obtain the public keys of others so you can send them encrypted mail or verify their digital signatures. You have three alternatives for obtaining someone's public key:

- Get the key from a public key server.
- Add the public key directly from an e-mail message.
- Import the public key from a file.

Public keys are really just blocks of text, so they are quite easy to add to your keyring by importing them from a file or by copying them from an e-mail message and then pasting them into your public keyring.

Getting public keys from a key server

If the person to whom you want to send encrypted mail is an experienced PGP user, chances are that they have placed a copy of their public key on a key server. This makes it very convenient for you to get a copy of their most up-to-date key whenever you want to send them mail and also relieves you from having to store a lot of keys on your public keyring.

With PGP Version 5.5, you can search for keys on a key server using these methods:

- User ID
- Key ID
- Key Type (Diffie-Hellman or RSA)
- Creation date
- Expiration date
- Revoked keys
- Disabled keys
- Key size
- Keys signed by a particular key

The inverse of most of these operations is also available. For example, you may search using “User ID is not Bob” as your criterion.

There are a number of public key servers, such as the one maintained by PGP, Inc., where you can locate the keys of most PGP users. If the recipient has not pointed you to the Web address where their public key is stored, you can access any key server and do a search for the user’s name or e-mail address, because all key servers are regularly updated to include the keys stored on all the other servers. With PGP Version 5.5, you can quickly locate a specific user’s key when you are sending e-mail or managing your keys from the PGPkeys window.

To get someone’s public key from a key server

1. Open the PGPkeys application from the PGP menu or by double-clicking the PGPkeys icon.
2. Choose Search Server from the Keys menu.

   The Search dialog box appears.
3. Select the location or server you want to search from the Search For Keys On menu.
4. Enter search criteria to use to locate the user’s public key. To narrow your search, click More Choices to specify additional criteria.

   If a public key for the specified user is found, you are asked whether you want to add it to your public keyring. When you add a public key to your keyring, the key shows up in the PGPkeys window, where you can examine it to make sure that it is valid.

Adding public keys from e-mail messages

One convenient way to get a copy of someone’s public key is to have that person include it in an e-mail message. If you have an e-mail application that is supported by the PGP plug-in, then you can add the sender’s public key to your public keyring by simply clicking a button. For example, when an e-mail message arrives with a block of text containing someone’s public key, click the key and envelope button to have the key stored on your public keyring.

If you are using an e-mail application that is not supported by the plug-ins, you can add the public key to the keyring by copying the block of text that represents the public key and pasting it into the PGPkeys window.

Importing a public key from a file

Another method of obtaining someone’s public key is to have that person save it to a file from which you can import it or copy and paste it into your public keyring. There are three methods of extracting someone’s public key and adding it to your public keyring.

- Choose Import from the Keys menu and then enter the name of the file where the public key is stored.
- Drag the file containing the public key onto the PGPkeys window.
- Open the text document where the public key is stored, select the block of text representing the key, and then choose Copy from the Edit menu. Go to the PGPkeys window and choose Paste from the Edit menu to copy the key. The key then shows up as an icon in the PGPkeys window.

Verifying the authenticity of a key

When you exchange keys with someone, it is sometimes hard to tell if the key really belongs to that person. PGP provides a number of safeguards that allow you to check a key’s authenticity and to certify that the key belongs to a particular owner. The PGP program also warns you if you attempt to use a key that is not valid and also defaults to warn you when you are about to use a marginally valid key.

One of the major vulnerabilities of public key encryption systems is the ability of some eavesdroppers to mount a “man-in-the-middle” attack by replacing someone’s public key with one of their own. In this way they can intercept any encrypted e-mail intended for that person, decrypt it using their own key, then encrypt it again with the person’s real key and send it on to them as if nothing had ever happened. In fact, this could all be done automatically through a sophisticated computer program that stands in the middle and deciphers all of your correspondence.

Based on this scenario, you and those with whom you exchange e-mail need a way to determine whether you do indeed have legitimate copies of each others' keys. The best way to be completely sure that a public key actually belongs to a particular person is to have the owner copy it to a floppy disk and then physically hand it to you. However, you are seldom close enough to personally hand a disk to someone; you generally exchange public keys via e-mail or get them from a public key server.

Even though these are somewhat less secure methods of exchanging tamper-proof keys, you can still determine if a key really belongs to a particular person by checking its digital fingerprint, a unique series of numbers generated when the key is created. By comparing the fingerprint on your copy of someone's public key to the fingerprint on their original key, you can be absolutely sure that you do in fact have a valid copy of their key.

The most definitive way to check a key's fingerprint is to call the person and have them read their fingerprint to you over the phone.

Once you are absolutely convinced that you have a legitimate copy of someone's public key, you can then sign their key. By signing someone's public key with your private key, you are certifying that you are sure the key belongs to the alleged user. For instance, when you create a new key, it is automatically certified with your own digital signature, since it is a reasonably safe assumption that the person creating the key is in fact the true owner. The reason for signing your own key is to prevent anyone from modifying it, which would immediately invalidate your signature. By default, signatures you make on other keys are not exportable, which means they apply only to the key when it is on your local keyring.

Getting keys via trusted introducers

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 they certify and return it so you can include their 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 and shows the level of trust and validity associated with each key in the PGPkeys window. 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. For details on how to sign keys and validate users, see “Signing someone’s public key” in Chapter 6.

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.

Chapter 4

Sending and Receiving Secure E-mail

This chapter explains how to encrypt and sign the e-mail you send to others and decrypt and verify the e-mail others send to you.

Encrypting and signing e-mail

The quickest and easiest way to encrypt and sign e-mail is with an application supported by the PGP plug-ins. Although the procedure varies slightly between different e-mail applications, you perform the encryption and signing process by clicking the appropriate buttons in the application's toolbar.

If you are using an e-mail application that is not supported by the PGP plug-ins, you can encrypt and sign your e-mail messages via PGPmenu, which is compatible with most popular text-based applications. When accessing this menu from the Finder, you can encrypt and sign or decrypt and verify files and even entire folders.

As an alternative to the other interfaces, you can also use the PGPtools window to encrypt and sign text and files. When using this interface to encrypt and sign text, you copy the text to the clipboard, perform the desired operation by choosing the appropriate button and then paste the contents back to your application. You can also encrypt and / or sign a selected portion of text or even files by dragging them to the appropriate button.

NOTE: If you don't send your e-mail immediately but instead store it in your outbox, you should be aware that when using some e-mail applications, the information is not encrypted until the e-mail is actually transmitted. Before queuing encrypted messages you should check to see if your application does in fact encrypt the messages in your outbox. If it does not, you might want to consider encrypting your messages via PGPmenu before queuing it in the outbox.

If you do not have one of the e-mail applications that is supported by PGP, see Chapter 5, for information on how to encrypt and sign files.

Encrypting and signing with supported e-mail applications

When you encrypt and sign with an e-mail application that is supported by the PGP plug-ins, you have two choices, depending on what type of e-mail application the recipient is using. If you are communicating with other PGP users who have an e-mail application that supports the PGP / MIME standard, you can take advantage of a PGP / MIME feature to encrypt and sign your e-mail messages and any file attachments automatically when you send them. If you are communicating with someone who does not have a PGP / MIME-compliant e-mail application, you should encrypt your e-mail with PGP / MIME turned off to avoid any compatibility problems.

To encrypt and sign with supported e-mail applications

1. Use your e-mail application to compose your e-mail message just as you normally would.
2. When you have finished composing the text of your e-mail message, specify whether you want to encrypt and sign the text of your message by clicking the lock and quill buttons.

NOTE: If you know that you are going to use PGP / MIME regularly, you can leave this turned on by selecting the appropriate settings from the e-mail pane of the Preferences dialog box.

3. Send your message as you normally do.

   If you have elected to sign the encrypted data, the Passphrase dialog box appears requesting your passphrase before the mail is sent. (This dialog box appears earlier in some applications.)

4. Enter your passphrase and then click OK.

   If you have a copy of the public keys for every one of the recipients, the appropriate keys are used. However, if you specify a recipient for whom there is no corresponding public key, the PGP Key Selection dialog box appears so that you can specify the correct key.

5. Drag the public keys for those who are to receive a copy of the encrypted e-mail message into the Recipients list box. You can also double-click any of the keys to move them from one area of the screen to the other.

   The Validity icon indicates the minimum level of confidence that the public keys in the Recipient list are valid. This validity is based on the signatures associated with the key, and the trust indicates how well you can rely on the owner of the key to vouch for the authenticity of another users key. See Chapter 6 for details.

6. Click OK to send your mail.

Encrypting e-mail to groups of recipients

You can use PGP to create recipient groups. For example, if you want to send encrypted mail to 10 people at engineering@xyz.com, you could create a group with that name. The View menu contains a Groups item that toggles the display of the Groups section of the PGPkeys window.

Using the groups feature, you can create a list of people to whom you want to send encrypted e-mail.

To create a group

1. Choose New Group from the Group menu.

2. Enter a name for the Group. Optionally, enter a group description.

3. For example, you can name a group “everyone@company.com” with a description of “All employees.”

4. Click OK to create the group.

To add members to a group

1. In the PGPkeys window, select the users or groups you want to add to your group.

2. Drag the users from the PGPkeys window to the Group pane.

NOTE: Groups can be members of other groups.

To delete group members

1. Select the group member to be deleted.

2. Press the Delete key.

PGP asks you to confirm your choice.

To delete a group

1. Select the group from the Groups pane of the PGPkeys windows.

2. Press the Delete key or choose Delete Groups from the pop-up menu.

To add a group to another group

1. Select the group that you want to place within another group.

2. Drag the group to the group into which you want to include it.

3. Select the Paste into Group from the pop-up menu.

To send encrypted e-mail to groups

1. Address the mail to your mail group

The name of your encryption group must correspond to the name of the e-mail group.

2. Encrypt the message.

3. Send the message.

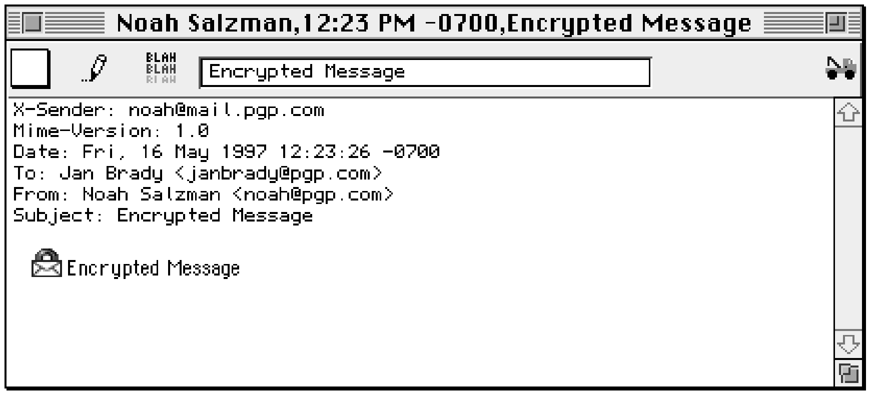
Decrypting and verifying e-mail

The quickest and easiest way to decrypt and verify the e-mail sent to you is with an application supported by the PGP plug-ins. Although the procedure varies slightly between different e-mail applications, when you are using an e-mail application supported by the plug-ins, you can perform the decryption and verification process by clicking the envelope icon in the message or your application's toolbar. In some cases you may need to select Decrypt / Verify from the menu in your e-mail application. In addition, if you are using an application that supports the PGP / MIME standard, you can decrypt and verify your e-mail messages as well as any file attachments by clicking an icon attached to your message.

If you are using an e-mail application that is not supported by the PGP plug-ins, you will decrypt and verify your e-mail messages via PGPmenu. In addition, if your e-mail includes encrypted file attachments, you must decrypt them separately via PGPtools, PGPmenu in the Finder, or PGPcontextmenu for MacOS8 users.

Decrypting and verifying from supported e-mail applications

If you are communicating with other PGP users, and they have encrypted and signed their mail using the PGP / MIME standard, a locked envelope icon appears when you open your e-mail.



```
X-Sender: noah@mail.pgp.com
Mime-Version: 1.0
Date: Fri, 16 May 1997 12:23:26 -0700
To: Jan Brady <janbrady@pgp.com>
From: Noah Salzman <noah@pgp.com>
Subject: Encrypted Message
```

In that case, you can decrypt and verify the message and any attached files by simply double-clicking this icon.

If you are receiving e-mail from someone who is not using a PGP/MIME-compliant e-mail application, you will decrypt the e-mail messages by clicking the opened envelope icon in your application's toolbar or by selecting Decrypt/Verify from the Plugins menu. Also, if there are any encrypted file attachments, you will decrypt them from PGPtools, or PGPmenu in the Finder, or PGPcontextmenu for MacOS8 users.

To decrypt and verify from supported e-mail applications

1. Open your e-mail message just as you normally do.

You will see a block of unintelligible ciphertext in the body of your e-mail message.

2. To decrypt and verify the contents of the e-mail message, click the opened envelope button in your application's toolbar.

NOTE: If you are using Eudora, you will have to decrypt the text by choosing the PGP Decrypt option from the Message Plug-In menu. A convenient shortcut is to hold down the command key and type the number 5

The PGP Enter Passphrase dialog box appears asking you to enter your passphrase.

3. Enter your passphrase and then click OK.

The message is decrypted. If it has been signed, a message appears indicating whether the signature is valid.

4. At this point you can save the message in its decrypted state, or you can save the original encrypted version so that it remains secure.

Chapter 5

Using PGP for Secure File Storage

This chapter describes how to use PGP functions without using e-mail plug-ins. It describes how to use PGP to encrypt and decrypt, and how to sign and verify files from the Finder using PGPtools or PGPmenu so that you can encrypt and sign files either for e-mail or for secure storage on your computer. It also describes the PGP Secure Wipe function, which deletes files by erasing their contents completely from your computer.

Using PGP to encrypt and decrypt files

You can use PGP to encrypt and sign files without using an e-mail plug-in so that you can send encrypted or signed files as e-mail attachments. You can also use the techniques described in this chapter to encrypt and sign files that you store on your computer.

Encrypting and signing using PGPmenu

If you plan to send an encrypted file as an attachment with your e-mail message, or if you want to encrypt a file to protect it on your computer, you do so using PGPmenu.

To encrypt and sign using PGPmenu

1. In the Finder, select the file or files that you want to encrypt.

2. Choose the desired option from PGPmenu or from PGPcontextmenu, which MacOS8 users can access by holding down the Control key while you select a file.

   The PGP Key Selection dialog box appears. You can select the recipient's public keys for the file you are encrypting or signing.

   You have the following encryption options:

   - **Text Output** When sending files as attachments with some e-mail applications, you may need to select the Text Output check box to save the file as ASCII text. This is sometimes necessary in order to send a binary file using older e-mail applications. Selecting this option increases the size of the file by about 30 percent.
   - **Conventional Encrypt** Select this check box to rely on a common passphrase rather than on public key cryptography. The file is encrypted using a session key, which encrypts using a passphrase that you will be asked to choose.
   - **Wipe Original** Select this check box to overwrite the original document that you are encrypting or signing, so that your sensitive information is not readable by anyone who can access your hard disk.

   If you have signed the files, you are asked to supply your passphrase.

3. Select the public keys by dragging them to the Recipients list and then click OK.

   If you look in the folder where the original file was located, you will find a file with the specified name represented by one of two icons:

.pgp	.asc
encrypted with standard output	encrypted with text output

To encrypt and sign using the PGP tools

1. In the Finder, select the file or files that you want to encrypt.

You can select multiple files, but you must encrypt and sign each of them individually.

2. Drag the file(s) onto the Encrypt, Sign, or Encrypt and Sign button in the PGPtools window.

The PGP Key Selection dialog box appears. You can select the recipient's public keys for the file you are encrypting or signing. The options available are described in "To decrypt and verify using PGPmenu".

3. Select the public keys by dragging them to the Recipients list and then click OK.

The file is encrypted and / or signed and the resulting file appears in the folder with the original.

Decrypting and verifying using PGPmenu

If the e-mail you receive has file attachments, and you are not using a PGP / MIME-compliant e-mail application, you must decrypt them from the Finder.

First, make sure that PGPmenu is available.

To decrypt and verify using PGPmenu

1. In the Finder, select the file or files that you want to decrypt and verify.

You can select multiple files, but you must decrypt and verify each file individually.

2. Choose Decrypt / Verify from PGPmenu, or hold down the Control key and select the file to open PGPcontextmenu, then choose Decrypt / Verify from the menu.

The passphrase dialog box appears.

3. Enter your passphrase and then click OK.

If the file has been signed, a message appears indicating whether the signature is valid.

4. Click OK.

   The decrypted file is saved in the same location as the original file.

To decrypt and verify using PGPtools

1. In the Finder, select the file or files that you want to decrypt.

   You can select multiple files, but you must decrypt and verify each of them individually.

2. Drag the file onto the Decrypt/Verify button in the PGPtools window.

   The PGP Enter Passphrase dialog box appears asking you to enter your passphrase.

3. Enter your passphrase and then click OK.

   If the message has been signed, a message appears indicating whether the signature is valid.

4. Click OK.

   The decrypted file is saved in the same location as the original file.

Using PGP Secure Wipe to delete files

The Secure Wipe item on PGPmenu deletes files and their contents. The Wipe feature is a secure way of permanently removing a file and its contents from the hard drive of your computer. When you delete a file normally by placing it in the Trash, the name of the file has been removed from the file directory, but the data in the file is still there. Secure Wipe removes all traces of a file's data so that no one can use a software tool to recover the file.

To permanently delete a file

1. Select the file you want to delete.
2. Select Wipe from PGPmenu.

   A confirmation dialog box appears.
3. Click OK to permanently erase the file

ALERT Even on systems with virtual memory, PGP correctly writes over all the contents of the file. It is worth noting that application programs that may have saved the file prior to encrypting the file, may have left fragments of the file on your disk in locations which are no longer considered part of the file. See “Swap files or virtual memory” in Chapter 8. Also, be aware that many programs automatically save files in progress, so there may be backed up copies of the file that you want to delete.
You may wish to use a third-party utility to wipe all free space on your disk to solve this problem. PGP does not currently have this feature.

Chapter 6

Managing Keys and Setting Preferences

This chapter explains how to examine and manage the keys stored on your digital keyrings. It also describes how to set your preferences to suit your particular computing environment.

Managing your keys

The keys you create, as well as those you collect from others, are stored on digital keyrings, which are essentially files stored on your hard drive or on a floppy disk. Normally your private keys are stored in a file named PGP Private Keys and your public keys are stored in another file named PGP Public Keys. These files are usually located in the PGP Keyrings folder. .

NOTE: If you are not comfortable storing your keys in the usual place, you can choose a different filename or location. For details, see “Setting your preferences,” later in this chapter.

Occasionally, you may want to examine or change the attributes associated with your keys. For instance, when you obtain someone’s public key, you might want to identify its type (either RSA or Diffie-Hellman / DSS), check its fingerprint, or determine its validity based on any digital signatures included with the key. You may also want to sign someone’s public key to indicate that you believe it is valid, assign a level of trust to the key’s owner, or change a passphrase for your private key. You may want to search a key server for someone’s key. You perform all of these key-management functions from the PGPkeys window.

The PGPkeys window

To open the PGPkeys application, choose PGPkeys from PGPmenu or double-click the application icon in the program folder.

The PGPkeys window displays the keys you have created for yourself, as well as any public keys you have added to your public keyring.

Double keys represent the private and public key pairs you have created for yourself, and single keys represent the public keys you have collected from others. If you have more than one type of key, you will notice that RSA-type keys are blue skeleton keys and Diffie-Hellman / DSS keys are yellow modern keys.

By clicking on the Disclosure Triangle at the left side of the key icon, you can expand the entries to reveal the user ID and e-mail address for the owner of the key as represented by the figure icons. By clicking an envelope icon, you can see the signatures of any users who have certified the key, as represented by one of four icons). If you don’t want to expand each key individually, simply select the keys of interest and then choose Expand All from the Edit menu.

PGPkeys attribute definitions

Along the top of the window are labels that correspond to the attributes associated with each key.

Name	Shows an iconic representation of the key along with the user name and e-mail address of the owner, and the names of the key’s signers.
Validity	Indicates the level of confidence that the key actually belongs to the alleged owner. The validity is based on who has signed the key and how well you trust the signer(s) to vouch for the authenticity of a key. The public keys you sign yourself have the highest level of validity, based on the assumption that you only sign someone’s key if you are totally convinced that it is valid. The validity of any other keys, which you have

not personally signed, depends on the level of trust you have granted to any other users who have signed the key. If there are no signatures associated with the key, then it is not considered valid, and a message indicating this fact appears whenever you use the key.

Validity is indicated by either diamonds or striped bars, indicating Implicit trust, and circles or half-filled bars for Marginal trust. When you click a diamond or a circle, the Key Signing dialog box appears, in which you can sign other user's keys. See "Signing someone's public key," later in this chapter.

Validity is represented by either dot and diamond shapes or bar shapes. The diamonds represent implicit validity, filled circles represent complete validity (or marginal also if the Treat Marginally Valid Keys as Invalid check box is deselected in the Advanced tab of the Preferences dialog box). Empty circles represent no validity or marginal validity if the Treat Marginally Valid Keys as Invalid check box is selected.

Trust Indicates the level of trust you have granted to the owner of the key to serve as an introducer for the public keys of others. This trust comes into play when you are unable to verify the validity of someone's public key for yourself and instead rely on the judgment of other users who have signed the key. When you create a pair of keys, they are considered implicitly trustworthy, as shown by the striping in the trust and validity bars, or by a diamond validity indicator.

When you receive a public key from someone that has been signed by another of the user's keys on your public keyring, the level of authenticity is based on the trust you have granted to the signer of that key. You assign a level of trust, either Complete, Marginal, or Untrusted, in the Key Properties dialog box.

Trust is not displayed initially in the PGP keys display. You can display the Trust column by choosing Columns from the View menu. Click the Validity button in the main window to open the Sign Key dialog box. The Validity button changes to a diamond when the key is axiomatic.

If Implicit Trust is set to on, trust is displayed with a diamond. You can change trust through the Key Properties menu. Trust options are None, Marginal, and Complete.

Creation Shows the date when the key was originally created. You can sometimes make an assumption about the validity of a key based on how long it has been in circulation. If the key has been in use for a while, it is less likely that someone will try to replace it because there are many other copies in circulation. Never rely on creation dates as the sole indicator of validity.

Expiration Shows the date when the key will expire. Most keys are set to Never; however, there may be instances when you want to use a key for a fixed period of time.

Corporate Message Recovery Key
Shows whether the key has an associated Corporate Message Recovery Key. (See Chapter 2 for a definition.)

Size Shows the number of bits used to construct the key. Generally, the larger the key, the less chance that it will ever be compromised. However, larger keys require slightly more time to encrypt and decrypt data than do smaller keys. When you create a Diffie-Hellman / DSS key, there is one number for the Diffie-Hellman portion and another number for the DSS portion. The DSS portion is used for signing, and the Diffie-Hellman portion for encryption.

Examining a key’s properties

In addition to the general attributes shown in the PGPkeys window, you can also examine and change other key properties. To access the properties for a particular key, select the desired key and then choose Info from the Keys menu.

Key ID	A unique identifying number associated with each key. This identification number is useful for distinguishing between two keys that share the same user name and e-mail address.
Created	The date when the key was created.
Key Type	The key type, either RSA or Diffie-Hellman/DSS.
Expires	The date when the key expires. Owners specify this date when they create their keys, and the value is usually set to Never. However, some keys are set to expire on a particular date if the owner wants them to be used for a limited period of time.
Key Size	The size of the key.
Cipher	CAST, Triple DES, or IDEA. This is the “preferred” cipher by which the owner of the key requests that you encrypt to his orher key. If this algorithm is allowed in your Advanced preferences, it will be used whenever encrypting to this key.
Trust Model	Indicates the validity of the key based on its certification and the level of trust you have in the owner to vouch for the authenticity of someone else’s public key. You set the trust level by sliding the bar to the appropriate level (Complete, Marginal, or Untrusted). The bar is disabled for revoked, expired, and implicitly trusted keys.
Enabled	Indicates whether the key is currently enabled. When a key is disabled, it is dimmed in the PGPkeys window and is not available for performing any PGP functions except Decrypt and Verify. However, the key remains on your keyring and you can enable it again at any time. To enable or disable a key, select or clear the Enabled check box. (The check box is not visible for

implicitly trusted keys.) This feature is useful for preventing seldom-used keys from cluttering up the Key Selection dialog box when you are sending encrypted e-mail.

Fingerprint A unique identification number that is generated when the key is created. This is the primary means by which you can check the authenticity of a key. One good way to check a fingerprint is to have the owner read their fingerprint to you over the phone so that you can compare it with the fingerprint shown for your copy of their public key. You can also check the authenticity of someone's key by comparing the fingerprint on your copy of their public key to the one listed on a key server, since it is assumed that the owner periodically checks to make sure that it remains valid.

Change Passphrase

Changes the passphrase for a private key. If you ever think that your passphrase is no longer a secret (perhaps you caught someone looking over your shoulder), click this button to enter a new passphrase.

It is a good idea to change your passphrase every 6 months or so. For instructions on changing your passphrase, see "Changing your Passphrase," later in this chapter.

Specifying a default key pair

When you sign a message or someone's public key, your default key pair is used. If you have more than one pair of keys, you may want to specifically designate one pair as your default pair. The current default key pair is displayed in bold type to distinguish these keys from your other keys.

To specify your default key pair

1. Select the key pair you want designated as your default pair.

2. Choose Set Default from the Keys menu.

The selected key pair is displayed in bold type, indicating that it is now designated as your default key pair.

Adding a new user name or address

You may have more than one user name or e-mail address for which you want to use the same set of keys. After creating a new set of keys, you can add alternate names and addresses to the key. You can only add a new user name or e-mail address if you have both the private and public keys.

To add a new user name or address to an existing key

1. Select the key pair for which you want to add another user name or address.
2. Choose Add Name from the Keys menu.

   The PGP New User Name dialog box appears.
3. Enter the new name and e-mail address in the appropriate fields, and then Click OK.

   The PGP Enter Passphrase dialog box appears.
4. Enter your passphrase and then click OK.

   The new name is added to the end of the user name list associated with the key. If you want to set the new user name and address as the primary identifier for your key, select the name and address and then choose Set Primary Name from the Keys menu.

Checking a key's fingerprint

It is often difficult to know for sure that a key belongs to a particular individual unless that person physically hands the key to you on a floppy disk. Exchanging keys in this manner is not usually practical, especially for users who are located many miles apart, but you can rely on the unique fingerprint associated with each key to verify that a key does indeed belong to the alleged owner. There are several ways to check a key's fingerprint, but the safest is to call the person and have them read the fingerprint to you over the phone. It is highly unlikely that someone would be able to intercept this random call and imitate the person you expect to hear on the other end. You can also compare the fingerprint on your copy of someone’s public key to the fingerprint listed for their original key on a public server.

To check a key’s fingerprint

1. Select the key for the fingerprint you want to check.
2. Choose Info from the Keys menu.
3. Note the fingerprint and compare it to the original.

Signing someone’s public key

When you create a set of keys, they are automatically signed using your public key. Similarly, once you are sure that a key belongs to the proper individual, you can sign that person’s public key, indicating that you are sure it is a valid key.

To sign someone’s public key

1. Select the key you want to sign.
2. Choose Sign from the Keys menu.

   The PGPkeys alert box appears.
3. Click OK to indicate your certainty that the key does indeed belong to the purported owner.

   The Enter Passphrase dialog box appears.
4. Enter your passphrase and then click OK.

   An icon associated with your user name is now included with the public key that you just signed.

   If you would like to send the key with your signature to a key server, select the Send Signature to Keyserver check box. The public key on the server is updated to reflect the inclusion of your signature. Most users prefer to use their discretion when allowing others to sign their keys, so it’s always a good idea to check with the owner before you add your signature to their key on the server.

   You are then asked to enter the passphrase for your default key pair.

5. Enter your passphrase and then click OK. If you have another key pair that you want to sign with, click the down arrow and select the desired key.

   A check box is displayed that says “Allow signature to be exported. Others may rely on your signature.”

6. Select this check box if you want your signature to be exported.

   An exportable signature is one that is allowed to be sent to servers and travels with the key whenever it is exported, such as by dragging it to an e-mail message.

   When you sign someone’s public key, an icon associated with your user name is shown for that key.

Granting trust for key validations

Besides certifying that a key belongs to someone, you can assign a level of trust to the user of the keys indicating how well you trust them to act as an introducer to others whose keys you may get in the future. This means that if you ever get a key from someone that has been signed by an individual whom you have designated as trustworthy, the key is considered valid even though you have not done the check yourself.

To grant trust for a key

1. Select the key for which you want to change the trust level.

2. Choose Info from the Keys menu.

   The Information dialog box appears.

3. Use the Trust Level sliding bar to choose the appropriate level of trust for the key: Never, Marginal, or Complete.

4. Close the dialog box to accept the new setting.

Disabling and enabling keys

Sometimes you may want to temporarily disable a key. The ability to disable keys is useful when you want to retain a public key for future use, but you don't want it cluttering up your recipient list every time you send mail.

To disable a key

1. Select the key you want to disable.
2. Choose Info from the Keys menu.

   The Information dialog box appears.
3. Clear the Enabled check box. (The Enabled check box does not appear for implictly trusted keys.)
4. Close the dialog box to accept the new setting.

The key is dimmed and is temporarily unavailable for use.

To enable a key

1. Select the key you want to enable.
2. Choose Info from the Keys menu.

   The Information dialog box appears.
3. Select the Enabled check box.
4. Close the dialog box to accept the new setting.

The key becomes visible and can be used as before.

Deleting a key or signature

At some point you may want to remove a key, a signature, or a user ID associated with a particular key.

To delete a key, signature, or user ID

1. Select the key, signature, or user ID you want to delete.
2. Choose Clear from the Edit menu or press the Delete key.

Changing your Passphrase

It’s a good idea to change your passphrase periodically.

To change your passphrase

1. Select the key pair for which you want to change the passphrase.

2. Choose Info from the Keys menu.

The Information dialog box appears.

3. Click Change Passphrase.

The Change Passphrase dialog box appears.

4. Enter your old passphrase in the top field and then press the Tab key to advance to the next field.

5. Enter your new passphrase in the center dialog box and then press the Tab key to advance to the bottom field

6. Confirm your entry by entering your new passphrase again.

7. Click OK.

Importing and Exporting Keys

Although you often distribute your public key and obtain the public keys of others by cutting and pasting the raw text from a public or corporate key server, you can also exchange keys by importing and exporting them as separate text files. For instance, someone could hand you a disk containing their public key, or you might want to make your public key available over an FTP server.

To import a key from a file

1. Select the key you want to import.

2. Choose Import from the Keys menu.

The Import dialog box appears.

3. Select the file that contains the key you want to import and then click Open.

The imported key appears in the PGPkeys window, where you can use it to encrypt data or to verify someone's digital signature.

To add a key from an e-mail message

If a colleague sends you an e-mail message with their key enclosed (as a block of text) you can add it to your keyring.

1. With the e-mail message window open, open the PGPkeys window.
2. Tile the two windows so that you can see part of the PGPkeys window behind the message window.
3. Select the key text, including the START BLOCK and END BLOCK texts. Drag the text onto the PGPkeys window.
4. The new key or keys appear in the PGPkeys window.

To export a key to a file

1. Select the key you want to export to a file.
2. Choose Export from the Keys menu.

   The Export dialog box appears.
3. Enter the name of the file to which you want the key to be exported and then click Save.

   The exported key is saved to the named file in the specified folder location.

Revoking a key

If the situation ever arises that you no longer trust your personal key pair, you can issue a revocation to the world telling everyone to stop using your public key. The best way to circulate a revoked key is to place it on a public key server.

To revoke a key

1. Select the key pair to revoke.
2. Choose Revoke from the Keys menu.

The Revocation Confirmation dialog box appears.

3. Click OK to confirm your intent to revoke the selected key.

   The PGP Enter Passphrase dialog box appears.

4. Enter your passphrase and then click OK.

   When you revoke a key, it is crossed out with a red line to indicate that it is no longer valid.

5. Send the revoked key to the server so everyone will know not to use your old key.

It is possible that you might forget your passphrase someday. In that case, you would never be able to use your key again, and you would have no way of revoking your old key when you create a new one. To safeguard against this possibility, you can create a revocation key by making a copy of your private key, revoking the copy, and saving the revoked copy in a safe place. You can then send the revoked copy to a public key server if you ever forget your password. However, you should be very careful about where you store the revoked version of your key. If someone were to get hold of the revoked key, they could revoke your key without your approval.

Setting your preferences

PGP is configured to accommodate the needs of most users, but you have the option of adjusting some of the settings to suit your particular computing environment. You specify these settings through the Preferences dialog box, which you can access by choosing Preferences from the PGPkeys Edit menu.

General preferences

You specify general encryption settings from the General pane.

Always Encrypt to Default Key

When this setting is selected, all the e-mail messages and file attachments you encrypt with a recipient's public key are also encrypted to you using your default public key. It is useful to leave this setting turned on so that you have the option of decrypting the contents of any e-mail or files you have previously encrypted.

Cache Decryption Passphrase

This setting specifies the amount of time (in hours: minutes: seconds) that your encryption passphrase is stored in your computer's memory. If you regularly compose or read several e-mail messages in succession, you may want to increase the amount of time your passphrase is cached so you don't have to enter it over and over again to get through all of your mail. However, remember that the longer your passphrase is stored in your computer's memory, the more time a sophisticated snooper has to get hold of this highly compromising bit of information. The default setting is 2 minutes, which is probably sufficient to perform most of your PGP chores without having to enter your passphrase too many times, but not long enough for you to walk away and an attacker to find it in your computer's memory.

Cache Signing Passphrase

This setting specifies the amount of time (in hours: minutes: seconds) that your signing passphrase is stored in your computer's memory. If you regularly compose or read several e-mail messages in succession, you may want to increase the amount of time your passphrase is cached so you don't have to enter it over and over again to get through all of your mail.

NOTE: Using your passphrase for signing is considered a greater threat in some cases because the attacker can impersonate you. This is the reason why the two caching options are handled separately. The cache times begin and reset every time you sign a message and erases the passphrase from memory immediately when the timer expires. This cache defaults to off due to the potentially complex security implications.

Faster Key Generation

When this setting is selected, less time is required to generate a new Diffie-Hellman / DSS key pair. This process is speeded up by using a previously calculated set of prime numbers rather than going through the time-consuming process of creating them from scratch each time a new key is generated. However, remember that fast key generation is only implemented for the fixed key sizes above 1024 and below 4096 provided as options when you create a key, and is not used if you enter some other value. Although it would be just about impossible for anyone to crack your key based on their knowledge of these canned prime numbers, some may want to spend the extra time to create a key pair with the maximum level of security.

The general belief in the cryptographic community is that using canned primes provides no decrease in security for the Diffie-Hellman / DSS algorithms. If this feature makes you uncomfortable, you may turn it off. For more information, read the FAQ located on the PGP website.

Warn Before Wiping Files

When this setting is selected, a dialog box appears before you wipe a file to give you one last chance to change your mind before PGP securely overwrites the contents of the file and deletes it from your computer.

Files preferences

Click the Files tab to advance to the pane in which you specify the location of the keyrings used to store your private and public keys.

Set Public Keyring Location

Shows the current location and name of the file where the PGP program expects to find your public keyring file. If you plan to store your public keys in a file with a different name or in some other location, you specify this information here. You can use the Browse button to search through your files rather than having to type the path. The location you specify will be used also to store all automatic backups of the public keyring.

Set Private Key Ring Location

Shows the current location and name of the file where the PGP program expects to find your private keyring file. If you plan to store your private keys in a file with a different name or in some other location, you specify this information here. Some users like to keep their private keyring on a floppy disk, which they insert like a key whenever they need to sign or decrypt mail. The location you specify will be used also to store all automatic backups of the public keyring.

Set Random Seed Location

Shows the location of the Random Seed file. Some users may wish to keep their Random Seed file in a secure location to prevent tampering. This attack is very difficult and PGP has many different protections against it.

E-mail preferences

Click the E-mail tab to advance to the pane in which you specify preferences that affect the way PGP functions are implemented for your particular e-mail application. Remember that not all of the selections may apply to your particular e-mail application.

Use PGP/MIME Encryption

If you are using Eudora and you turn this setting on, all of your e-mail messages and file attachments are automatically encrypted to the intended recipient. This setting has no effect on other encryptions you perform from PGPmenu and should not be used if you plan to send e-mail to recipients who use e-mail applications that are not supported by the PGP/MIME standard. Using Eudora, attachments will always be encrypted regardless of this setting, but if the recipient does not have PGP/MIME, the decryption provess will be more manual.

Use PGP/MIME Signing

If you are using Eudora and you turn this setting on, all of your e-mail messages and file attachments automatically include your digital signatures. This setting has no effect on other signatures you add from PGPmenu and should not be used if you plan to send e-mail to recipients who are using applications that do not support the PGP / MIME standard.

Word wrap clear-signed messages at column [ ]

This setting specifies the column number where a hard carriage return is used to wrap the text in your digital signature to the next line. This feature is necessary because not all applications handle word wrapping in the same way, which could cause the lines in your digitally signed messages to be broken up in a way that cannot be easily read. The default setting is 70, which prevents problems with most applications.

ALERT If you change the word-wrap setting in PGP, make sure that it is less than the word-wrap settings in your e-mail application. If you set it to be the same or a greater length, carriage returns may be added that invalidate your PGP signature.

PGPmenu preferences

Click the PGPmenu tab to advance to the pane where you add and remove PGPmenu for various applications.

Add Use this option to add the PGP icon to the menu bar of the applications you select. For example, click the Add button and add SimpleText to the application list. The PGP icon is added to the SimpleText menu bar, so that you can sign, encrypt, decrypt, and verify the selected text in the document.

The PGP icon is automatically available on the Finder menu bar, so that you can encrypt entire folders while using the Finder. Simply select the folder you want to encrypt and then choose Encrypt from PGPmenu.

	Double-clicking an application name in the PGPmenu preferences brings up an Advanced PGPmenu Preferences dialog for that particular application, which contains settings that may help if you experience any compatibility problems using PGPmenu with a particular application
Remove	Use this option to remove the PGP icon from the menu bar of applications you have previously selected.

Server preferences

Click the Server tab to advance to the pane where you specify settings for the key server you are using.

Domain	Specifies the Internet domain (such as “company.com”) for the public key server that is used by PGP to send and retrieve public keys. When sending keys to a server, this domain is used to look up the appropriate server based on the key’s domain.
Port	The port address for the public key server. You must enter the address in full URL format, such as “http://pgpkeys.mit.edu:11371”. The LDAP protocol is also supported.
Set Default	Specifies which server to use if no server can be found for the domain of the key.

Advanced preferences

Click the Advanced tab to advance to the pane in which you make the following choices.

Encryption algorithm

You can select the encryption algorithm for your PGP keys: CAST (the default), IDEA, or Triple-DES. If you want to use IDEA or Triple-DES, you must make the selection before you generate your keys.

CAST is a new algorithm that PGP and other cryptographers have very high confidence in, and Triple-DES is a U.S. Government algorithm that has withstood the test of time. IDEA is the algorithm traditionally used in PGP. For more information about these algorithms, see “The PGP symmetric algorithms” in Chapter 8.

There are two reasons why PGP gives you the option to change encryption algorithms:

- When using conventional encryption, the cipher selected here is used to encrypt.
- When creating a key, the preferred cipher is recorded as part of the key so that other people use that algorithm when encrypting to you.

ALERT Use the CAST, IDEA, and Triple-DES check boxes only if you have suddenly decided that a particular algorithm is insecure. For example, if you become aware that Triple-DES has been broken, you can deselect that check box and all new keys you generate will have a record that Triple-DES may not be used when encrypting to you.

Display Marginal Validity Level

Use this check box to specify whether to display marginally valid keys or simply to show validity as on or off. Green circles indicate that a key is valid; gray indicates that the key has not been validated, has not been signed by either a trusted introducer or by you.

Treat marginally valid keys as invalid

Use this check box to specify whether to treat all marginally valid keys as invalid. Selecting this option causes the Key Selection dialog box to appear whenever you encrypt to marginally valid keys.

Searching for a key

You can search for keys on local keyrings and remote key servers.

To search for a user's key

1. Open the PGPkeys window.
2. Choose Search from the Keys menu.

   The PGPkeys Search window appears.
3. Choose the location you wish to search from the Search for Keys On menu.
4. Specify your search criteria:

   The default is User ID, but you can click the drop-down menu to select Key ID, Key Type, Creation Date, Expiration Date, Key Status, or Key Size. For example, you might search for all keys with the User ID of Fred.
5. Specify the condition for which you are searching.

   You can use any of the following conditions:

   - Is
   - Is Not
   - Contains
   - Does Not Contain
   - Is Signed By
   - Is Not Signed By
   - Is On (for Creation or Expiration Dates)
   - Is On or Before (for Creation or Expiration Dates)
   - Is On or After (for Creation or Expiration Dates)
6. Enter the value you want to search for.
7. Click More Choices to add additional criteria to your search; for example, Key IDs with the name Fred created on or before October 6, 1997.

8. To begin the search, click Search.

A progress bar shows that the search is being conducted.

NOTE: To cancel a search in progress, click Stop Search

The results of the search appear in the window.

9. Click Clear Search to empty the search window and clear your search criteria.

Chapter 7

Troubleshooting PGP

This chapter presents information about potential problems and suggests solutions.

Error	Cause	Solution
No secret keys could be found on your keyring.	There are no secret keys on your keyring.	Generate your own pair of keys in PGPkeys.
The evaluation time for PGP encrypting and signing has passed. Operation aborted.	The product evaluation time has expired.	Download the freeware version or buy the commercial version of the product.
The PGP library has run out of memory.	The operating system has run out of memory.	Close other running programs. If that doesn't work, you may need more memory in your machine.
There was an error opening or writing the keyring or the output file.	A file that was needed couldn't be opened.	Make sure the settings in your PGP Preferences is correct. If you've recently deleted files in the directory that you installed PGP, you may need to re-install the product.
Unable to perform operation because this file is read-only or otherwise protected. If you store your keyring files on removable media the media may not be inserted.	A file that was needed is set to read-only or is being used by another program.	Close other programs that may be accessing the same files as the program you are running. If you keep your keyring files on a floppy disk, check and make sure that the floppy disk is in the floppy drive.

Error	Cause	Solution
There was an error during the writing of the keyring or the exported file.	The program failed to write data to a certain file.	Your hard drive may be full, or if the file is on a floppy, the floppy is not present in the floppy drive.
The action could not be completed due to an invalid file operation.	The program failed to read or write data in a certain file.	The file is probably corrupt. Try altering your PGP Preferences to use a different file, if possible.
The passphrase you entered does not match the passphrase on the key.	The passphrase you entered is incorrect.	You may have the CAPS LOCK on, or you simply may have mis-typed the passphrase. Try again.
The keyring contains a bad (corrupted) PGP packet.	The PGP message that you are working with has been corrupted, or your keyring has been corrupted.	Ask the sender to re-send the message if it's a message that you're working with. If it's your keyring, try restoring from your backup keyring.
The specified user ID was not added because it already exists on the selected key.	You can't add a User ID to a key if there is one just like it already on the key.	Try adding a different user ID, or delete the matching one first.
The keyring contains a bad (corrupted) PGP packet.	The PGP message that you are working with has been corrupted, or your keyring has been corrupted.	Ask the sender to re-send the message if it's a message that you're working with. If it's your keyring, try restoring from your backup keyring.
The specified user ID was not added because it already exists on the selected key.	You can't add a User ID to a key if there is one just like it already on the key.	Try adding a different user ID, or delete the matching one first.
This key is already signed by the specified signing key.	You can't sign a key that you have already signed.	You may have accidentally picked the wrong key. If so, choose a different key to sign.

Error	Cause	Solution
The specified key could not be found on your keyring.	The key needed to decrypt the current message is not on your keyring.	Ask the sender of the message to resend the message and make sure they encrypt the message to your public key.
The message/data contains a detached signature.	The signature for the message/file is located in a separate file.	Double-click on the detached signature file first.
Could not encrypt to specified key because it is a sign-only key.	The selected key can only be used for signing.	Choose a different key, or generate a new key that can encrypt data.
Could not sign with specified key because it is an encrypt-only key.	The selected key can only be used for encrypting.	Choose a different key, or generate a new key that can sign data.
There is not enough random data currently available.	The random number generator needs more input in order to generate good random numbers.	When prompted, move the mouse around, or press random keys, in order to generate input.
The specified input file does not exist.	The file name typed in does not exist.	Use the Explorer to find the exact name and path of the file you want.
Cannot perform the requested operation because the output buffer is too small.	The output is larger than the internal buffers can handle.	If you are encrypting or signing, you may have to break up the message and encrypt/sign smaller pieces at a time. If you are decrypting or verifying, ask the sender to encrypt/sign smaller pieces and re-send them to you.
The keyring file is corrupt.	The program failed to read or write data in a certain file.	There is a file that is probably corrupt or missing. It may or may not be the keyring file. Try using a different file name or path, if possible.

Chapter 8

Security Features and Vulnerabilities

This chapter contains introductory and background information about cryptography written by Phil Zimmermann.

"Whatever you do will be insignificant, but it is very important that you do it." —Mahatma Gandhi.

Why I wrote PGP

It’s personal. It’s private. And it’s no one’s business but yours. You may be planning a political campaign, discussing your taxes, or having a secret romance. Or you may be communicating with a political dissident in a repressive country. Whatever it is, you don’t want your private electronic mail (e-mail) or confidential documents read by anyone else. There’s nothing wrong with asserting your privacy. Privacy is as apple-pie as the Constitution.

The right to privacy is spread implicitly throughout the Bill of Rights. But when the United States Constitution was framed, the Founding Fathers saw no need to explicitly spell out the right to a private conversation. That would have been silly. Two hundred years ago, all conversations were private. If someone else was within earshot, you could just go out behind the barn and have your conversation there. No one could listen in without your knowledge. The right to a private conversation was a natural right, not just in a philosophical sense, but in a law-of-physics sense, given the technology of the time.

But with the coming of the information age, starting with the invention of the telephone, all that has changed. Now most of our conversations are conducted electronically. This allows our most intimate conversations to be exposed without our knowledge. Cellular phone calls may be monitored by anyone with a radio. Electronic mail, sent across the Internet, is no more secure than cellular phone calls. E-mail is rapidly replacing postal mail, becoming the norm for everyone, not the novelty it was in the past. And e-mail can be routinely and automatically scanned for interesting keywords, on a large scale, without detection. This is like driftnet fishing.

Perhaps you think your e-mail is legitimate enough that encryption is unwarranted. If you really are a law-abiding citizen with nothing to hide, then why don't you always send your paper mail on postcards? Why not submit to drug testing on demand? Why require a warrant for police searches of your house? Are you trying to hide something? If you hide your mail inside envelopes, does that mean you must be a subversive or a drug dealer, or maybe a paranoid nut? Do law-abiding citizens have any need to encrypt their e-mail?

What if everyone believed that law-abiding citizens should use postcards for their mail? If a nonconformist tried to assert his privacy by using an envelope for his mail, it would draw suspicion. Perhaps the authorities would open his mail to see what he's hiding. Fortunately, we don't live in that kind of world, because everyone protects most of their mail with envelopes. So no one draws suspicion by asserting their privacy with an envelope. There's safety in numbers. Analogously, it would be nice if everyone routinely used encryption for all their e-mail, innocent or not, so that no one drew suspicion by asserting their e-mail privacy with encryption. Think of it as a form of solidarity.

Until now, if the government wanted to violate the privacy of ordinary citizens, they had to expend a certain amount of expense and labor to intercept and steam open and read paper mail. Or they had to listen to and possibly transcribe spoken telephone conversation, at least before automatic voice recognition technology became available. This kind of labor-intensive monitoring was not practical on a large scale. It was only done in important cases when it seemed worthwhile.

Senate Bill 266, a 1991 omnibus anticrime bill, had an unsettling measure buried in it. If this non-binding resolution had become real law, it would have forced manufacturers of secure communications equipment to insert special "trap doors" in their products, so that the government could read anyone’s encrypted messages. It reads, “It is the sense of Congress that providers of electronic communications services and manufacturers of electronic communications service equipment shall ensure that communications systems permit the government to obtain the plain text contents of voice, data, and other communications when appropriately authorized by law.” It was this bill that led me to publish PGP electronically for free that year, shortly before the measure was defeated after vigorous protest by civil libertarians and industry groups.

The 1994 Digital Telephony bill mandated that phone companies install remote wiretapping ports into their central office digital switches, creating a new technology infrastructure for “point-and-click” wiretapping, so that federal agents no longer have to go out and attach alligator clips to phone lines. Now they’ll be able to sit in their headquarters in Washington and listen in on your phone calls. Of course, the law still requires a court order for a wiretap. But while technology infrastructures can persist for generations, laws and policies can change overnight. Once a communications infrastructure optimized for surveillance becomes entrenched, a shift in political conditions may lead to abuse of this new-found power. Political conditions may shift with the election of a new government, or perhaps more abruptly from the bombing of a federal building.

A year after the 1994 Digital Telephony bill passed, the FBI disclosed plans to require the phone companies to build into their infrastructure the capacity to simultaneously wiretap 1 percent of all phone calls in all major U.S. cities. This would represent more than a thousandfold increase over previous levels in the number of phones that could be wiretapped. In previous years, there were only about a thousand court-ordered wiretaps in the United States per year, at the federal, state, and local levels combined. It’s hard to see how the government could even employ enough judges to sign enough wiretap orders to wiretap 1 percent of all our phone calls, much less hire enough federal agents to sit and listen to all that traffic in real time. The only plausible way of processing that amount of traffic is a massive Orwellian application of automated voice recognition technology to sift through it all, searching for interesting keywords or searching for a particular speaker’s voice. If the government doesn’t find the target in the first 1 percent sample, the wiretaps can be shifted over to a different 1 percent until the target is found, or until everyone’s phone line has been checked for subversive traffic. The FBI says they need this capacity to plan for the future. This plan sparked such outrage that it was defeated in Congress, at least this time around, in 1995. But the mere fact that the FBI even asked for these broad powers is revealing of their agenda. And the defeat of this plan isn’t so reassuring when you consider that the 1994 Digital Telephony bill was also defeated the first time it was introduced, in 1993.

Advances in technology will not permit the maintenance of the status quo, as far as privacy is concerned. The status quo is unstable. If we do nothing, new technologies will give the government new automatic surveillance capabilities that Stalin could never have dreamed of. The only way to hold the line on privacy in the information age is strong cryptography.

You don’t have to distrust the government to want to use cryptography. Your business can be wiretapped by business rivals, organized crime, or foreign governments. The French government, for example, is notorious for using its signals intelligence apparatus against U.S. companies to help French corporations get a competitive edge. Ironically, the United States government’s restrictions on cryptography have weakened U.S. corporate defenses against foreign intelligence and organized crime.

The government knows what a pivotal role cryptography is destined to play in the power relationship with its people. In April 1993, the Clinton administration unveiled a bold new encryption policy initiative, which had been under development at the National Security Agency (NSA) since the start of the Bush administration. The centerpiece of this initiative is a government-built encryption device, called the Clipper chip, containing a new classified NSA encryption algorithm. The government has been trying to encourage private industry to design it into all their secure communication products, such as secure phones, secure faxes, and so on. AT&T has put Clipper into its secure voice products. The catch: At the time of manufacture, each Clipper chip will be loaded with its own unique key, and the government gets to keep a copy, placed in escrow. Not to worry, though—the government promises that they will use these keys to read your traffic only “when duly authorized by law.” Of course, to make Clipper completely effective, the next logical step would be to outlaw other forms of cryptography.

The government initially claimed that using Clipper would be voluntary, that no one would be forced to use it instead of other types of cryptography. But the public reaction against the Clipper chip has been strong, stronger than the government anticipated. The computer industry has monolithically proclaimed its opposition to using Clipper.

FBI director Louis Freeh responded to a question in a press conference in 1994 by saying that if Clipper failed to gain public support, and FBI wiretaps were shut out by non-government-controlled cryptography, his office would have no choice but to seek legislative relief. Later, in the aftermath of the Oklahoma City tragedy, Mr. Freeh testified before the Senate Judiciary Committee that public availability of strong cryptography must be curtailed by the government (although no one had suggested that cryptography was used by the bombers).

The Electronic Privacy Information Center (EPIC) obtained some revealing documents under the Freedom of Information Act. In a briefing document titled "Encryption: The Threat, Applications and Potential Solutions," and sent to the National Security Council in February 1993, the FBI, NSA, and Department of Justice (DOJ) concluded that "Technical solutions, such as they are, will only work if they are incorporated into all encryption products. To ensure that this occurs, legislation mandating the use of Government-approved encryption products or adherence to Government encryption criteria is required."

The government has a track record that does not inspire confidence that they will never abuse our civil liberties. The FBI's COINTELPRO program targeted groups that opposed government policies. They spied on the antiwar movement and the civil rights movement. They wiretapped the phone of Martin Luther King Jr. Nixon had his enemies list. And then there was the Watergate mess. Congress now seems intent on passing laws curtailing our civil liberties on the Internet. At no time in the past century has public distrust of the government been so broadly distributed across the political spectrum, as it is today.

If we want to resist this unsettling trend in the government to outlaw cryptography, one measure we can apply is to use cryptography as much as we can now while it's still legal. When use of strong cryptography becomes popular, it's harder for the government to criminalize it. Therefore, using PGP is good for preserving democracy.

If privacy is outlawed, only outlaws will have privacy. Intelligence agencies have access to good cryptographic technology. So do the big arms and drug traffickers. But ordinary people and grassroots political organizations mostly have not had access to affordable "military grade" public-key cryptographic technology. Until now.

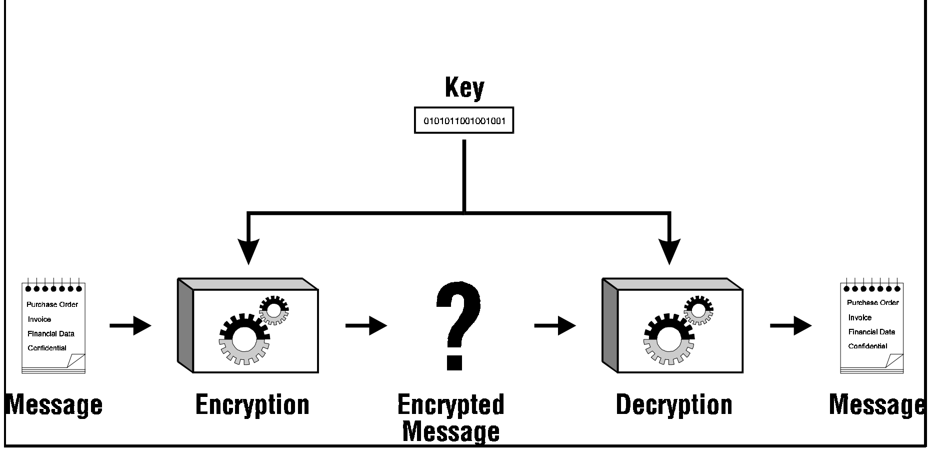
PGP empowers people to take their privacy into their own hands. There's a growing social need for it. That's why I created it.

Encryption basics

First, some elementary terminology. Suppose you want to send a message to a colleague, whom we'll call Alice, and you don't want anyone but Alice to be able to read it. As shown in Figure 1, you can encrypt, or encipher, the message, which means scrambling it up in a hopelessly complicated way, rendering it unreadable to anyone except you and Alice. You supply a cryptographic key to encrypt the message, and Alice must use the same key to decipher or decrypt it. At least that's how it works in conventional "secret key" encryption.

A single key is used for both encryption and decryption. This means that this key must be initially transmitted via secure channels so that both parties can know it before encrypted messages are sent over insecure channels. This may be inconvenient. If you have a secure channel for exchanging keys, then why do you need cryptography in the first place?

Figure 1



How public key cryptography works

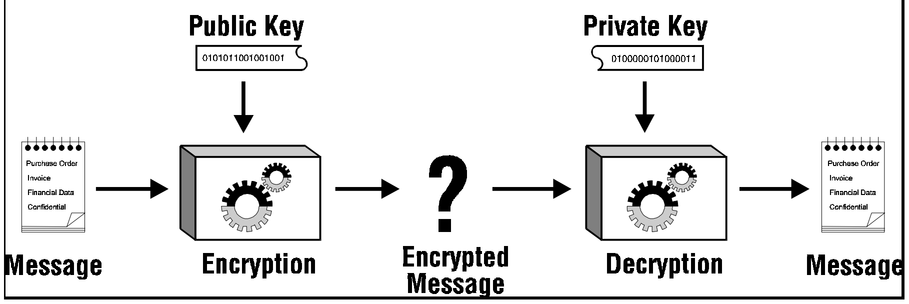
In public key cryptography, as shown in Figure 2, everyone has two related complementary keys, a public key and a private key. Each key unlocks the code that the other key makes. Knowing the public key does not help you deduce the corresponding private key. The public key can be published and widely disseminated across a communications network.

This protocol provides privacy without the need for the same kind of secure channels that conventional secret key encryption requires.

Anyone can use a recipient’s public key to encrypt a message to that person, and that recipient uses her own corresponding private key to decrypt that message. No one but the recipient can decrypt it, because no one else has access to that private key. Not even the person who encrypted the message with the recipient’s public key can decrypt it.

Figure 2

Public Key Cryptography



How your files and messages are encrypted

Because the public key encryption algorithm is much slower than conventional single-key encryption, encryption is better accomplished by using the process shown in Figure 3.

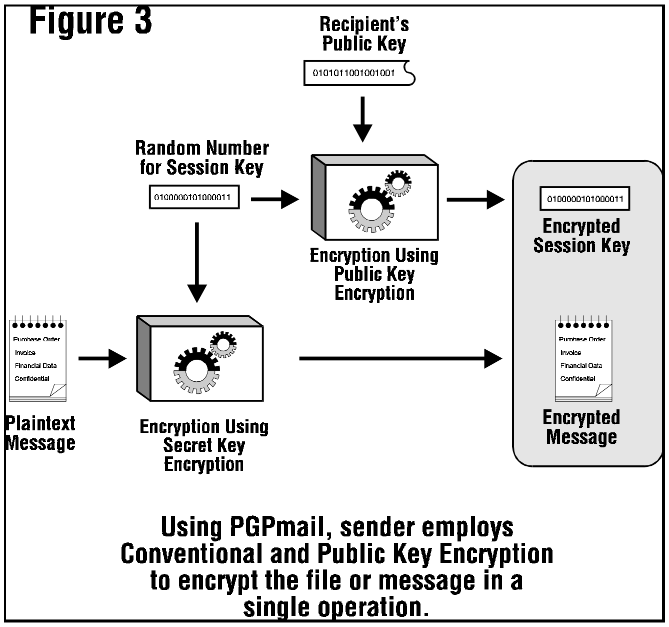


Figure 3
Recipient's Public Key
0101011001001001
Random Number for Session Key
0100000101000011
Encryption Using Public Key Encryption
0100000101000011
Encrypted Session Key
Purchase Order
Invoice
Financial Data
Confidential
Plaintext Message
Encryption Using Secret Key Encryption
Encrypted Message
Using PGPmail, sender employs Conventional and Public Key Encryption to encrypt the file or message in a single operation.

A high-quality fast conventional secret key encryption algorithm is used to encipher the message. This original unenciphered message is called "plaintext." In a process invisible to the user, a temporary random key, created just for this one "session," is used to conventionally encipher the plaintext file. Then the recipient's public key is used to encipher this temporary random conventional key. This public-key-enciphered conventional "session" key is sent along with the enciphered text (called "ciphertext") to the recipient.

The PGP symmetric algorithms

PGP offers a selection of different secret key algorithms to encrypt the actual message. By secret key algorithm, we mean a conventional, or symmetric, block cipher that uses the same key to both encrypt and decrypt. The three symmetric block ciphers offered by PGP are CAST, Triple-DES, and IDEA. They are not "home-grown" algorithms. They were all developed by teams of cryptographers with distinguished reputations.

For the cryptographically curious, all three ciphers operate on 64-bit blocks of plaintext and ciphertext. CAST and IDEA have key sizes of 128 bits, while Triple-DES uses a 168-bit key. Like Data Encryption Standard (DES), any of these ciphers can be used in cipher feedback (CFB) and cipher block chaining (CBC) modes. PGP uses them in 64-bit CFB mode.

I included the CAST encryption algorithm in PGP because it shows promise as a good block cipher with a 128-bit key size, it's very fast, and it's free. Its name is derived from the initials of its designers, Carlisle Adams and Stafford Tavares of Northern Telecom (Nortel). Nortel has applied for a patent for CAST, but they have made a commitment in writing to make CAST available to anyone on a royalty-free basis. CAST appears to be exceptionally well designed, by people with good reputations in the field. The design is based on a very formal approach, with a number of formally provable assertions that give good reasons to believe that it probably requires key exhaustion to break its 128-bit key. CAST has no weak or semiweak keys. There are strong arguments that CAST is completely immune to both linear and differential cryptanalysis, the two most powerful forms of cryptanalysis in the published literature, both of which have been effective in cracking DES. CAST is too new to have developed a long track record, but its formal design and the good reputations of its designers will undoubtedly attract the attentions and attempted cryptanalytic attacks of the rest of the academic cryptographic community. I'm getting nearly the same preliminary gut feeling of confidence from CAST that I got years ago from IDEA, the cipher I selected for use in earlier versions of PGP. At that time, IDEA was also too new to have a track record, but it has held up well.

The IDEA (International Data Encryption Algorithm) block cipher is based on the design concept of "mixing operations from different algebraic groups." It was developed at ETH in Zurich by James L. Massey and Xuejia Lai, and published in 1990. Early published papers on the algorithm called it IPES (Improved Proposed Encryption Standard), but they later changed the name to IDEA. So far, IDEA has resisted attack much better than other ciphers such as FEAL, REDOC-II, LOKI, Snefru and Khafre. And IDEA is more resistant than DES to Biham and Shamir's highly successful differential cryptanalysis attack, as well as attacks from linear cryptanalysis. As this cipher continues to attract attack efforts from the most formidable quarters of the cryptanalytic world, confidence in IDEA is growing with the passage of time. Sadly, the biggest obstacle to IDEA's acceptance as a standard has been the fact that Ascom Systec holds a patent on its design, and unlike DES and CAST, IDEA has not been made available to everyone on a royalty-free basis.

As a hedge, PGP includes three-key Triple-DES in its repertoire of available block ciphers. The DES was developed by IBM in the mid-1970s. While it has a good design, its 56-bit key size is too small by today's standards. Triple-DES is very strong, and has been well studied for many years, so it might be a safer bet than the newer ciphers such as CAST and IDEA. Triple-DES is the DES applied three times to the same block of data, using three different keys, except that the second DES operation is run backwards, in decrypt mode. While Triple-DES is much slower than either CAST or IDEA, speed is usually not critical for e-mail applications. Although Triple-DES uses a key size of 168 bits, it appears to have an effective key strength of at least 112 bits against an attacker with impossibly immense data storage capacity to use in the attack. According to a paper presented by Michael Weiner at Crypto96, any remotely plausible amount of data storage available to the attacker would enable an attack that would require about as much work as breaking a 129-bit key. Triple-DES is not encumbered by any patents.

PGP public keys that were generated by PGP Version 5.0 or later have information embedded in them that tells a sender what block ciphers are understood by the recipient's software, so that the sender's software knows which ciphers can be used to encrypt. DSS / Diffie-Hellman public keys accept CAST, IDEA, or Triple-DES as the block cipher, with CAST as the default selection. At present, for compatibility reasons, RSA keys do not provide this feature. Only the IDEA cipher is used by PGP to send messages to RSA keys, because older versions of PGP only supported RSA and IDEA.

Data compression

PGP normally compresses the plaintext before encrypting it, because it's too late to compress the plaintext after it has been encrypted; encrypted data is not compressible. Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit redundancies found in the plaintext to crack the cipher. Data compression reduces this redundancy in the plaintext, thereby greatly enhancing resistance to cryptanalysis. It takes extra time to compress the plaintext, but from a security point of view it's worth it.

Files that are too short to compress, or that just don't compress well, are not compressed by PGP. In addition, the program recognizes files produced by most popular compression programs, such as PKZIP, and does not try to compress a file that has already been compressed.

For the technically curious, the program uses the freeware ZIP compression routines written by Jean-Loup Gailly, Mark Adler, and Richard B. Wales. This ZIP software uses compression algorithms that are functionally equivalent to those used by PKWare's PKZIP 2.x. This ZIP compression software was selected for PGP mainly because it has a really good compression ratio and because it's fast.

About the random numbers used as session keys

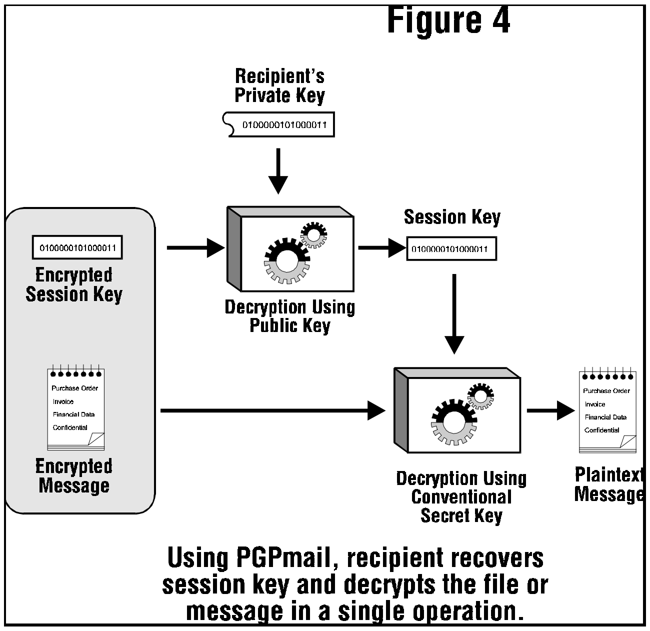
PGP uses a cryptographically strong pseudo-random-number generator for creating temporary session keys.If this random seed file does not exist, it is automatically created and seeded with truly random numbers derived from your random events gathered by the PGP program from the timing of your keystroke and mouse movements.

This generator reseeds the seed file each time it is used, by mixing in new material partially derived from the time of day and other truly random sources. It uses the conventional encryption algorithm as an engine for the random number generator. The seed file contains both random seed material and random key material used to key the conventional encryption engine for the random generator.

This random seed file should be protected from disclosure, to reduce the risk of an attacker deriving your next or previous session keys. The attacker would have a very hard time getting anything useful from capturing this random seed file, because the file is cryptographically laundered before and after each use. Nonetheless, it seems prudent to try to keep it from falling into the wrong hands. If possible, make the file readable only by you. If this is not possible, don't let other people indiscriminately copy disks from your computer.

How decryption works

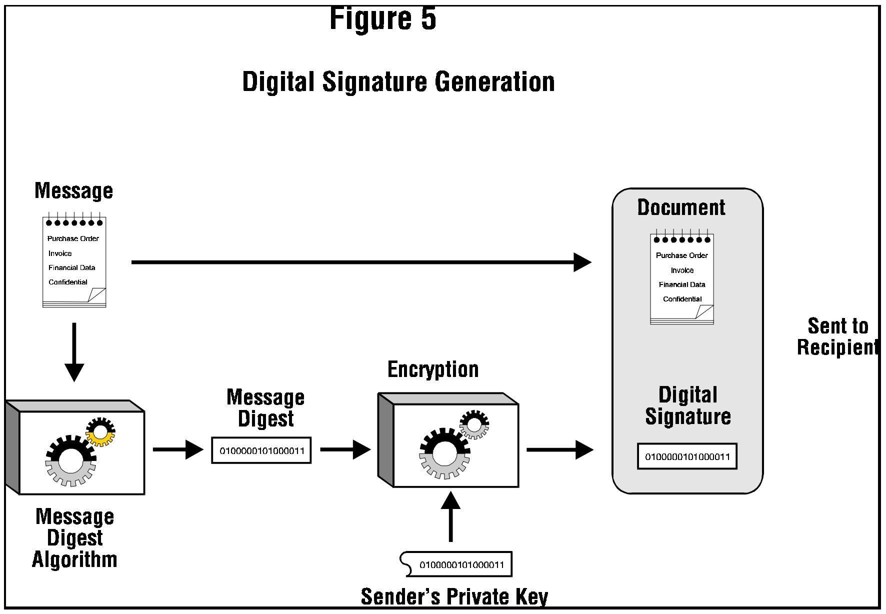
As shown in Figure 4, the decryption process is just the reverse of encryption. The recipient's private key is used to recover the temporary session key, and then that session key is used to run the fast conventional secret key algorithm to decipher the large ciphertext message.



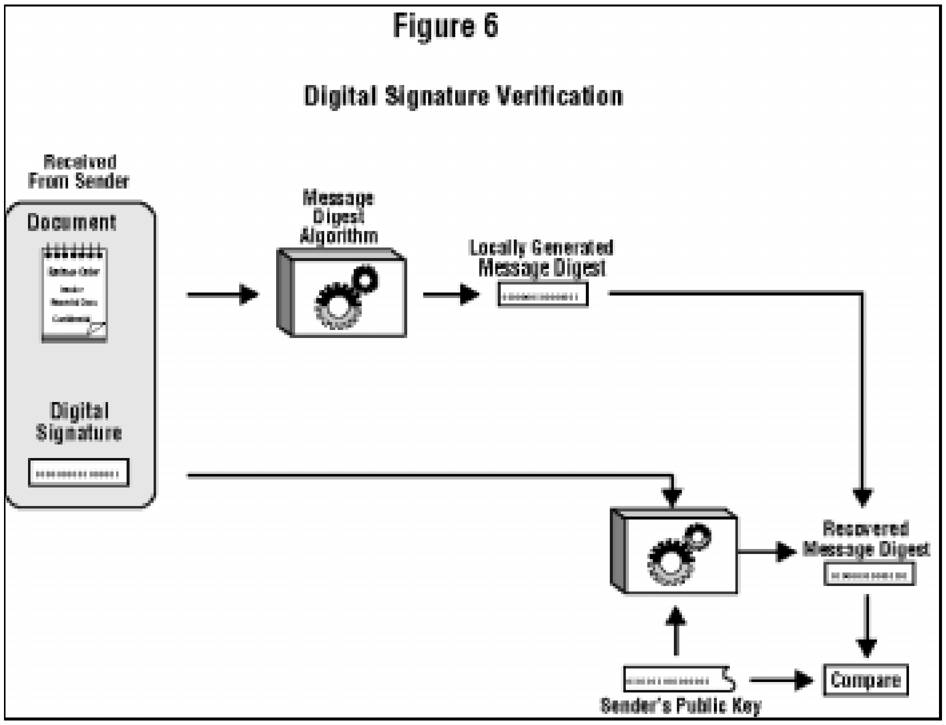
How digital signatures work

PGP uses digital signatures to provide message authentication. The sender's own private key can be used to encrypt a message digest, thereby "signing" the message. A *message digest* is a 160-bit or a 128-bit cryptographically strong one-way hash function. It is somewhat analogous to a "checksum" or CRC error checking code, in that it compactly represents the message and is used to detect changes in the message. Unlike a CRC, however, it is believed to be computationally infeasible for an attacker to devise a substitute message that would produce an identical message digest. The message digest gets encrypted by the sender's private key, creating a digital signature of the message.

Figure 5 shows how a digital signature is generated.



The recipient (or anyone else) can verify the digital signature by using the sender's public key to decrypt it, as shown in Figure 6. This proves that the sender was the true originator of the message, and that the message has not been subsequently altered by anyone else, because the sender alone possesses the private key that made that signature. Forgery of a signed message is not feasible, and the sender cannot later disavow his signature.



About the message digest

The message digest is a compact (160-bit or 128-bit) “distillate” of your message or file checksum. You can also think of it as a “fingerprint” of the message or file. The message digest “represents” your message, in such a way that if the message were altered in any way, a different message digest would be computed from it. This makes it possible to detect any changes made to the message by a forger. A message digest is computed using a cryptographically strong one-way hash function of the message. It should be computationally infeasible for an attacker to devise a substitute message that would produce an identical message digest. In that respect a message digest is much better than a checksum, because it is easy to devise a different message that would produce the same checksum. But like a checksum, you can’t derive the original message from its message digest.

The message digest algorithm now used in PGP (Version 5.0 and later) is called SHA, which stands for Secure Hash Algorithm, designed by the NSA for the National Institute of Standards and Technology (NIST).

SHA is a 160-bit hash algorithm. Some people might regard anything from the NSA with suspicion, because the NSA is in charge of intercepting communications and breaking codes. But keep in mind that the NSA has no interest in forging signatures, and the government would benefit from a good unforgeable digital signature standard that would preclude anyone from repudiating their signatures. That has distinct benefits for law enforcement and intelligence gathering. Also, SHA has been published in the open literature and has been extensively peer reviewed by most of the best cryptographers in the world who specialize in hash functions, and the unanimous opinion is that SHA is extremely well designed. It has some design innovations that overcome all the observed weaknesses in message digest algorithms previously published by academic cryptographers. All new versions of PGP use SHA as the message digest algorithm for creating signatures with the new DSS keys that comply with the NIST Digital Signature Standard. For compatibility reasons, new versions of PGP still use MD5 for RSA signatures, because older versions of PGP used MD5 for RSA signatures.

The message digest algorithm used by older versions of PGP is the MD5 Message Digest Algorithm, placed in the public domain by RSA Data Security, Inc. MD5 is a 128-bit hash algorithm. In 1996, MD5 was all but broken by a German cryptographer, Hans Dobbertin. Although MD5 was not completely broken at that time, it was discovered to have such serious weaknesses that no one should keep using it to generate signatures. Further work in this area might completely break it, allowing signatures to be forged. If you don’t want to someday find your PGP digital signature on a forged confession, you might be well advised to migrate to the new PGP DSS keys as your preferred method for making digital signatures, because DSS uses SHA as its secure hash algorithm.

How to protect public keys from tampering

In a public key cryptosystem, you don’t have to protect public keys from exposure. In fact, it’s better if they are widely disseminated. But it’s important to protect public keys from tampering, to make sure that a public key really belongs to the person to whom it appears to belong. This may be the most important vulnerability of a public key cryptosystem. See “Protecting your keys” in Chapter 3 for procedures. Let’s first look at a potential disaster, then describe how to safely avoid it with PGP.

Suppose you want to send a private message to Alice. You download Alice's public key certificate from an electronic bulletin board system (BBS). You encrypt your letter to Alice with this public key and send it to her through the BBS's e-mail facility.

Unfortunately, unbeknownst to you or Alice, another user named Charlie has infiltrated the BBS and generated a public key of his own with Alice's user ID attached to it. He covertly substitutes his bogus key in place of Alice's real public key. You unwittingly use this bogus key belonging to Charlie instead of Alice's public key. All looks normal because this bogus key has Alice's user ID. Now Charlie can decipher the message intended for Alice because he has the matching private key. He may even reencrypt the deciphered message with Alice's real public key and send it on to her so that no one suspects any wrongdoing. Furthermore, he can even make apparently good signatures from Alice with this private key because everyone will use the bogus public key to check Alice's signatures.

The only way to prevent this disaster is to prevent anyone from tampering with public keys. If you got Alice's public key directly from Alice, this is no problem. But that may be difficult if Alice is a thousand miles away or is currently unreachable.

Perhaps you could get Alice's public key from a mutually trusted friend, David, who knows he has a good copy of Alice's public key. David could sign Alice's public key, vouching for the integrity of Alice's public key. David would create this signature with his own private key.

This would create a signed public key certificate, and would show that Alice's key had not been tampered with. This requires that you have a known good copy of David's public key to check his signature. Perhaps David could provide Alice with a signed copy of your public key also. David is thus serving as an "Introducer" between you and Alice.

This signed public key certificate for Alice could be uploaded by David or Alice to the BBS, and you could download it later. You could then check the signature via David's public key and thus be assured that this is really Alice's public key. No impostor can fool you into accepting his own bogus key as Alice's because no one else can forge signatures made by David.

A widely trusted person could even specialize in providing this service of "introducing" users to each other by providing signatures for their public key certificates. This trusted person could be regarded as a

"Certifying Authority." Any public key certificates bearing the Certifying Authority's signature could be trusted as truly belonging to the person to whom they appear to belong to. All users who wanted to participate would need a known good copy of just the Certifying Authority's public key, so that the Certifying Authority's signatures could be verified. In some cases, the Certifying Authority may also act as a key server, allowing users on a network to look up public keys by asking the key server, but there is no reason why a key server must also certify keys.

A trusted centralized Certifying Authority is especially appropriate for large impersonal centrally controlled corporate or government institutions. Some institutional environments use hierarchies of Certifying Authorities.

For more decentralized environments, allowing all users to act as trusted introducers for their friends would probably work better than a centralized key certification authority.

One of the attractive features of PGP is that it can operate equally well in a centralized environment with a Certifying Authority or in a more decentralized environment where individuals exchange personal keys.

This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. It is the "Achilles heel" of public key cryptography, and a lot of software complexity is tied up in solving this one problem.

You should use a public key only after you are sure that it is a good public key that has not been tampered with, and that it actually belongs to the person with whom it purports to be associated. You can be sure of this if you got this public key certificate directly from its owner, or if it bears the signature of someone else that you trust, from whom you already have a good public key. Also, the user ID should have the full name of the key's owner, not just her first name.

No matter how tempted you are, you should *never* give in to expediency and trust a public key you downloaded from a bulletin board, unless it is signed by someone you trust. That uncertified public key could have been tampered with by anyone, maybe even by the system administrator of the bulletin board.

If you are asked to sign someone else’s public key certificate, make certain that it really belongs to the person named in the user ID of that public key certificate. This is because your signature on her public key certificate is a promise by you that this public key really belongs to her. Other people who trust you will accept her public key because it bears your signature. It can be ill-advised to rely on hearsay—don’t sign her public key unless you have independent first-hand knowledge that it really belongs to her. Preferably you should sign it only if you got it directly from her.

In order to sign a public key, you must be far more certain of that key’s ownership than if you merely want to use that key to encrypt a message. To be convinced of a key’s validity enough to use it, certifying signatures from trusted introducers should suffice. But to sign a key yourself, you should require your own independent first-hand knowledge of who owns that key. Perhaps you could call the key’s owner on the phone and read the key fingerprint to her, to confirm that the key you have is really her key—and make sure you really are talking to the right person.

Bear in mind that your signature on a public key certificate does not vouch for the integrity of that person, but only vouches for the integrity (the ownership) of that person’s public key. You aren’t risking your credibility by signing the public key of a sociopath, if you are completely confident that the key really belongs to him. Other people would accept that key as belonging to him because you signed it (assuming they trust you), but they wouldn’t trust that key’s owner. Trusting a key is not the same as trusting the key’s owner.

It would be a good idea to keep your own public key on hand with a collection of certifying signatures attached from a variety of “introducers,” in the hope that most people will trust at least one of the introducers who vouch for the validity of your public key. You could post your key with its attached collection of certifying signatures on various electronic bulletin boards. If you sign someone else’s public key, return it to them with your signature so that they can add it to their own collection of credentials for their own public key.

Make sure that no one else can tamper with your own public keyring. Checking a newly signed public key certificate must ultimately depend on the integrity of the trusted public keys that are already on your own public keyring. Maintain physical control of your public keyring, preferably on your own personal computer rather than on a remote time-sharing system, just as you would do for your private key. This is to protect it from tampering, not from disclosure. Keep a trusted backup copy of your public keyring and your private key on write-protected media.

Since your own trusted public key is used as a final authority to directly or indirectly certify all the other keys on your keyring, it is the most important key to protect from tampering. You may want to keep a backup copy on a write-protected floppy disk.

PGP generally assumes that you will maintain physical security over your system and your keyrings, as well as your copy of PGP itself. If an intruder can tamper with your disk, then in theory he can tamper with the program itself, rendering moot the safeguards the program may have to detect tampering with keys.

One somewhat complicated way to protect your own whole public keyring from tampering is to sign the whole ring with your own private key. You could do this by making a detached signature certificate of the public keyring.

How does PGP keep track of which keys are valid?

Before you read this section, you should read the previous section, “How to protect public keys from tampering.”

PGP keeps track of which keys on your public keyring are properly certified with signatures from introducers that you trust. All you have to do is tell PGP which people you trust as introducers, and certify their keys yourself with your own ultimately trusted key. PGP can take it from there, automatically validating any other keys that have been signed by your designated introducers. And of course you can directly sign more keys yourself.

There are two entirely separate criteria that PGP uses to judge a public key’s usefulness—don’t get them confused:

1. Does the key actually belong to the person to whom it appears to belong? In other words, has it been certified with a trusted signature?
2. Does it belong to someone you can trust to certify other keys?

PGP can calculate the answer to the first question. To answer the second question, you must tell PGP explicitly. When you supply the answer to question 2, PGP can then calculate the answer to question 1 for other keys signed by the introducer you designated as trusted.

Keys that have been certified by a trusted introducer are deemed valid by PGP. The keys belonging to trusted introducers must themselves be certified either by you or by other trusted introducers.

PGP also allows for the possibility of your having several shades of trust for people to act as introducers. Your trust for a key's owner to act as an introducer does not just reflect your estimation of their personal integrity—it should also reflect how competent you think they are at understanding key management and using good judgment in signing keys. You can designate a person as untrusted, marginally trusted, or completely trusted to certify other public keys. This trust information is stored on your keyring with their key, but when you tell PGP to copy a key off your keyring, PGP does not copy the trust information along with the key, because your private opinions on trust are regarded as confidential.

When PGP is calculating the validity of a public key, it examines the trust level of all the attached certifying signatures. It computes a weighted score of validity—for example, two marginally trusted signatures are deemed to be as credible as one fully trusted signature. The program's skepticism is adjustable—for example, you can tune PGP to require two fully trusted signatures or three marginally trusted signatures to judge a key as valid.

Your own key is "axiomatically" valid to PGP, needing no introducer's signature to prove its validity. PGP knows which public keys are yours by looking for the corresponding private keys on the private key. PGP also assumes that you completely trust yourself to certify other keys.

As time goes on, you will accumulate keys from other people whom you may want to designate as trusted introducers. Everyone else will choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

This unique grass-roots approach contrasts sharply with standard public key management schemes developed by government and other monolithic institutions, such as Internet Privacy Enhanced Mail (PEM), which are based on centralized control and mandatory centralized trust. The standard schemes rely on a hierarchy of Certifying Authorities who dictate who you must trust. The program's decentralized probabilistic method for determining public key legitimacy is the centerpiece of its key management architecture. PGP lets you alone choose who you trust, putting you at the top of your own private certification pyramid. PGP is for people who prefer to pack their own parachutes.

Note that while this decentralized, grass-roots approach is emphasized here, it does not mean that PGP does not perform equally well in the more hierarchical, centralized public key management schemes. Large corporate users, for example, will probably want a central figure or person who signs all the employees' keys. PGP handles that centralized scenario as a special degenerate case of PGP's more generalized trust model.

How to protect private keys from disclosure

Protect your own private key and your passphrase very carefully. If your private key is ever compromised, you'd better get the word out quickly to all interested parties before someone else uses it to make signatures in your name. For example, someone could use it to sign bogus public key certificates, which could create problems for many people, especially if your signature is widely trusted. And of course, a compromise of your own private key could expose all messages sent to you.

To protect your private key, you can start by always keeping physical control of it. Keeping it on your personal computer at home is OK, or keep it in your notebook computer that you can carry with you. If you must use an office computer that you don't always have physical control of, then keep your public and private keyrings on a write-protected removable floppy disk, and don't leave it behind when you leave the office. It wouldn't be a good idea to allow your private key to reside on a remote timesharing computer, such as a remote dial-in UNIX system. Someone could eavesdrop on your modem line and capture your passphrase and then obtain your actual private key from the remote system. You should only use your private key on a machine that is under your physical control. See Chapter 6 for additional information.

Don’t store your passphrase anywhere on the computer that has your private key file. Storing both the private key and the passphrase on the same computer is as dangerous as keeping your PIN in the same wallet as your Automatic Teller Machine bank card. You don’t want somebody to get their hands on your disk containing both the passphrase and the private key file. It would be most secure if you just memorize your passphrase and don’t store it anywhere but your brain. If you feel you must write down your passphrase, keep it well protected, perhaps even better protected than the private key file.

And keep backup copies of your private key—remember, you have the only copy of your private key, and losing it will render useless all the copies of your public key that you have spread throughout the world.

The decentralized noninstitutional approach that PGP supports for management of public keys has its benefits, but unfortunately it also means that you can’t rely on a single centralized list of which keys have been compromised. This makes it a bit harder to contain the damage of a private key compromise. You just have to spread the word and hope that everyone hears about it.

If the worst case happens—your private key and passphrase are both compromised (hopefully you will find this out somehow)—you will have to issue a “key compromise” certificate. This kind of certificate is used to warn other people to stop using your public key. You can use PGP to create such a certificate by using the Revoke command from the PGPkeys menu. Then you must somehow send this compromise certificate to everyone else on the planet, or at least to all your friends and their friends, et cetera. Their own PGP software installs this key compromise certificate on their public keyrings and automatically prevents them from accidentally using your public key ever again. You can then generate a new private / public key pair and publish the new public key. You could send out one package containing both your new public key and the key compromise certificate for your old key.

What if you lose your private key?

Normally, if you want to revoke your own private key, you can use the Revoke command from the PGPkeys menu to issue a revocation certificate, signed with your own private key.

But what can you do if you lose your private key, or if your private key is destroyed? You can't revoke it yourself, because you must use your own private key to revoke it, and you don't have it anymore. You ask each person who signed your key to retire his or her certification. Then anyone attempting to use your key based on the trust of one of your introducers will know not to trust your public key.

Beware of snake oil

When examining a cryptographic software package, the question always remains, why should you trust this product? Even if you examined the source code yourself, not everyone has the cryptographic experience to judge the security. Even if you are an experienced cryptographer, subtle weaknesses in the algorithms could still elude you.

When I was in college in the early seventies, I devised what I believed was a brilliant encryption scheme. A simple pseudorandom number stream was added to the plaintext stream to create ciphertext. This would seemingly thwart any frequency analysis of the ciphertext, and would be uncrackable even to the most resourceful government intelligence agencies. I felt so smug about my achievement.

Years later, I discovered this same scheme in several introductory cryptography texts and tutorial papers. How nice. Other cryptographers had thought of the same scheme. Unfortunately, the scheme was presented as a simple homework assignment on how to use elementary cryptanalytic techniques to trivially crack it. So much for my brilliant scheme.

From this humbling experience I learned how easy it is to fall into a false sense of security when devising an encryption algorithm. Most people don't realize how fiendishly difficult it is to devise an encryption algorithm that can withstand a prolonged and determined attack by a resourceful opponent. Many mainstream software engineers have developed equally naive encryption schemes (often even the very same encryption scheme), and some of them have been incorporated into commercial encryption software packages and sold for good money to thousands of unsuspecting users.

This is like selling automotive seat belts that look good and feel good, but snap open in the slowest crash test. Depending on them may be worse than not wearing seat belts at all. No one suspects they are bad until a real crash. Depending on weak cryptographic software may cause you to unknowingly place sensitive information at risk when you might not otherwise have done so if you had no cryptographic software at all. Perhaps you may never even discover that your data has been compromised.

Sometimes commercial packages use the Federal Data Encryption Standard (DES), a fairly good conventional algorithm recommended by the government for commercial use (but not for classified information, oddly enough—Hmmm). There are several “modes of operation” that DES can use, some of them better than others. The government specifically recommends not using the weakest simplest mode for messages, the Electronic Codebook (ECB) mode. But they do recommend the stronger and more complex Cipher Feedback (CFB) and Cipher Block Chaining (CBC) modes.

Unfortunately, most of the commercial encryption packages I’ve looked at use ECB mode. When I’ve talked to the authors of a number of these implementations, they say they’ve never heard of CBC or CFB modes, and don’t know anything about the weaknesses of ECB mode. The very fact that they haven’t even learned enough cryptography to know these elementary concepts is not reassuring. And they sometimes manage their DES keys in inappropriate or insecure ways. Also, these same software packages often include a second faster encryption algorithm that can be used instead of the slower DES. The author of the package often thinks his proprietary faster algorithm is as secure as DES, but after questioning him I usually discover that it’s just a variation of my own brilliant scheme from college days. Or maybe he won’t even reveal how his proprietary encryption scheme works, but assures me it’s a brilliant scheme and I should trust it. I’m sure he believes that his algorithm is brilliant, but how can I know that without seeing it?

In fairness I must point out that in most cases these terribly weak products do not come from companies that specialize in cryptographic technology.

Even the really good software packages, that use DES in the correct modes of operation, still have problems. Standard DES uses a 56-bit key, which is too small by today's standards, and can now be easily broken by exhaustive key searches on special high-speed machines. The DES has reached the end of its useful life, and so has any software package that relies on it.

There is a company called AccessData (87 East 600 South, Orem, Utah 84058, phone 1-800-658-5199) that sells a package for $185 that cracks the built-in encryption schemes used by WordPerfect, Lotus 1-2-3, MS Excel, Symphony, Quattro Pro, Paradox, MS Word, and PKZIP. It doesn't simply guess passwords—it does real cryptanalysis. Some people buy it when they forget their password for their own files. Law enforcement agencies buy it too, so they can read files they seize. I talked to Eric Thompson, the author, and he said his program only takes a split second to crack them, but he put in some delay loops to slow it down so it doesn't look so easy to the customer.

In the secure telephone arena, your choices look bleak. The leading contender is the STU-III (Secure Telephone Unit), made by Motorola and AT&T for $2,000 to $3,000, and used by the government for classified applications. It has strong cryptography, but requires some sort of special license from the government to buy this strong version. A commercial version of the STU-III is available that is watered down for NSA's convenience, and an export version is available that is even more severely weakened. Then there is the $1,200 AT&T Surity 3600, which uses the government's famous Clipper chip for encryption, with keys escrowed with the government for the convenience of wiretappers. Then, of course, there are the analog (nondigital) voice scramblers that you can buy from the spy-wannabe catalogs, that are really useless toys as far as cryptography is concerned, but are sold as "secure" communications products to customers who just don't know any better.

In some ways, cryptography is like pharmaceuticals. Its integrity may be absolutely crucial. Bad penicillin looks the same as good penicillin. You can tell if your spreadsheet software is wrong, but how do you tell if your cryptography package is weak? The ciphertext produced by a weak encryption algorithm looks as good as ciphertext produced by a strong encryption algorithm. There's a lot of snake oil out there. A lot of quack cures. Unlike the patent medicine hucksters of old, these software implementors usually don't even know their stuff is snake oil. They may be good software engineers, but they usually haven't even read any of the academic literature in cryptography. But they think they can write good cryptographic software. And why not? After all, it seems intuitively easy to do so. And their software seems to work OK.

Anyone who thinks they have devised an unbreakable encryption scheme either is an incredibly rare genius or is naive and inexperienced. Unfortunately, I sometimes have to deal with would-be cryptographers who want to make “improvements” to PGP by adding encryption algorithms of their own design.

I remember a conversation with Brian Snow, a highly placed senior cryptographer with the NSA. He said he would never trust an encryption algorithm designed by someone who had not “earned their bones” by first spending a lot of time cracking codes. That made a lot of sense. I observed that practically no one in the commercial world of cryptography qualifies under this criterion. “Yes,” he said with a self-assured smile, “And that makes our job at NSA so much easier.” A chilling thought. I didn’t qualify either.

The government has peddled snake oil too. After World War II, the United States sold German Enigma ciphering machines to third-world governments. But they didn’t tell them that the Allies cracked the Enigma code during the war, a fact that remained classified for many years. Even today many UNIX systems worldwide use the Enigma cipher for file encryption, in part because the government has created legal obstacles against using better algorithms. They even tried to prevent the initial publication of the RSA algorithm in 1977. And they have for many years squashed essentially all commercial efforts to develop effective secure telephones for the general public.

The principal job of the United States government’s National Security Agency is to gather intelligence, principally by covertly tapping into people’s private communications (see James Bamford’s book, *The Puzzle Palace*). The NSA has amassed considerable skill and resources for cracking codes. When people can’t get good cryptography to protect themselves, it makes NSA’s job much easier. NSA also has the responsibility of approving and recommending encryption algorithms. Some critics charge that this is a conflict of interest, like putting the fox in charge of guarding the hen house. In the 1980s, NSA had been pushing a conventional encryption algorithm that they designed (the COMSEC Endorsement Program), and they won’t tell anybody how it works because that’s classified. They wanted others to trust it and use it. But any cryptographer can tell you that a well-designed encryption algorithm does not have to be classified to remain secure. Only the keys should need protection. How does anyone else really know if NSA's classified algorithm is secure? It's not that hard for NSA to design an encryption algorithm that only they can crack, if no one else can review the algorithm. And now with the Clipper chip, the NSA is pushing SKIPJACK, another classified cipher they designed. Are they deliberately selling snake oil?

There are three main factors that have undermined the quality of commercial cryptographic software in the United States.

- The first is the virtually universal lack of competence of implementors of commercial encryption software (although this is starting to change since the publication of PGP). Every software engineer fancies himself a cryptographer, which has led to the proliferation of really bad crypto software.
- The second is the NSA deliberately and systematically suppressing all the good commercial encryption technology, by legal intimidation and economic pressure. Part of this pressure is brought to bear by stringent export controls on encryption software which, by the economics of software marketing, has the net effect of suppressing domestic encryption software.
- The third principle method of suppression comes from the granting of all the software patents for all the public key encryption algorithms to a single company, affording a single choke point to suppress the spread of this technology (although this crypto patent cartel broke up in the fall of 1995).

The net effect of all this is that before PGP was published, there was almost no highly secure general purpose encryption software available in the United States.

I'm not as certain about the security of PGP as I once was about my brilliant encryption software from college. If I were, that would be a bad sign. But I don't think PGP contains any glaring weaknesses (although I'm pretty sure it contains bugs). I have selected the best algorithms from the published literature of civilian cryptologic academia. For the most part, these algorithms have been individually subject to extensive peer review. I know many of the world's leading cryptographers, and have discussed with some of them many of the cryptographic algorithms and protocols used in PGP. It's well researched, and has been years in the making. And I don’t work for the NSA. But you don’t have to trust my word on the cryptographic integrity of PGP, because source code is available to facilitate peer review.

One more point about my commitment to cryptographic quality in PGP: Since I first developed and released PGP for free in 1991, I spent three years under criminal investigation by United States Customs for PGP’s spread overseas, with risk of criminal prosecution and years of imprisonment. By the way, you didn’t see the government getting upset about other cryptographic software—it’s PGP that really set them off. What does that tell you about the strength of PGP? I have earned my reputation on the cryptographic integrity of my products. I will not betray my commitment to our right to privacy, for which I have risked my freedom. I’m not about to allow a product with my name on it to have any secret back doors.

Vulnerabilities

No data security system is impenetrable. PGP can be circumvented in a variety of ways. In any data security system, you have to ask yourself if the information you are trying to protect is more valuable to your attacker than the cost of the attack. This should lead you to protect yourself from the cheapest attacks, while not worrying about the more expensive attacks.

Some of the discussion that follows may seem unduly paranoid, but such an attitude is appropriate for a reasonable discussion of vulnerability issues.

“If all the personal computers in the world—260 million—were put to work on a single PGP-encrypted message, it would still take an estimated 12 million times the age of the universe, on average, to break a single message.” William Crowell, Deputy Director, National Security Agency, March 20, 1997.

Compromised passphrase and private key

Probably the simplest attack comes if you leave the passphrase for your private key written down somewhere. If someone gets it and also gets your private key file, they can read your messages and make signatures in your name.

Here are some recommendations for protecting your passphrase:

1. Don't use obvious passphrases that can be easily guessed, such as the names of your kids or spouse.
2. Use spaces and a combination of numbers and letters in your passphrase. If you make your passphrase a single word, it can be easily guessed by having a computer try all the words in the dictionary until it finds your password. That's why a passphrase is so much better than a password. A more sophisticated attacker may have his computer scan a book of famous quotations to find your passphrase.
3. Be creative. Use an easy to remember but hard to guess passphrase; you can easily construct one by using some creatively nonsensical sayings or obscure literary quotes.

Public key tampering

A major vulnerability exists if public keys are tampered with. This may be the most crucially important vulnerability of a public key cryptosystem, in part because most novices don't immediately recognize it. The importance of this vulnerability, and appropriate hygienic countermeasures, are detailed in "How to protect public keys from tampering," earlier in this chapter.

To summarize: When you use someone's public key, make certain it has not been tampered with. A new public key from someone else should be trusted only if you got it directly from its owner, or if it has been signed by someone you trust. Make sure no one else can tamper with your own public keyring. Maintain physical control of both your public keyring and your private key, preferably on your own personal computer rather than on a remote timesharing system. Keep a backup copy of both keyrings.

Not Quite Deleted Files

Another potential security problem is caused by how most operating systems delete files. When you encrypt a file and then delete the original plaintext file, the operating system doesn't actually physically erase the data. It merely marks those disk blocks as deleted, allowing the space to be reused later. It's sort of like discarding sensitive paper documents in the paper recycling bin instead of the paper shredder. The disk blocks still contain the original sensitive data you wanted to erase, and will probably be overwritten by new data at some point in the future. If an attacker reads these deleted disk blocks soon after they have been deallocated, he could recover your plaintext.

In fact, this could even happen accidentally, if something went wrong with the disk and some files were accidentally deleted or corrupted. A disk recovery program may be run to recover the damaged files, but this often means that some previously deleted files are resurrected along with everything else. Your confidential files that you thought were gone forever could then reappear and be inspected by whoever is attempting to recover your damaged disk. Even while you are creating the original message with a word processor or text editor, the editor may be creating multiple temporary copies of your text on the disk, just because of its internal workings. These temporary copies of your text are deleted by the word processor when it's done, but these sensitive fragments are still on your disk somewhere.

The only way to prevent the plaintext from reappearing is to somehow cause the deleted plaintext files to be overwritten. Unless you know for sure that all the deleted disk blocks will soon be reused, you must take positive steps to overwrite the plaintext file, and also any fragments of it on the disk left by your word processor. You can take care of any fragments of the plaintext left on the disk by using PGP's Secure Wipe feature.

Viruses and Trojan horses

Another attack could involve a specially tailored hostile computer virus or worm that might infect PGP or your operating system. This hypothetical virus could be designed to capture your passphrase or private key or deciphered messages and to covertly write the captured information to a file or send it through a network to the virus's owner. Or it might alter PGP's behavior so that signatures are not properly checked. This attack is cheaper than cryptanalytic attacks.

Defending against this kind of attack falls into the category of defending against viral infection generally. There are some moderately capable antiviral products commercially available, and there are hygienic procedures to follow that can greatly reduce the chances of viral infection. A complete treatment of antiviral and antiworm countermeasures is beyond the scope of this document. PGP has no defenses against viruses, and assumes that your own personal computer is a trustworthy execution environment. If such a virus or worm actually appeared, hopefully word would soon get around warning everyone.

A similar attack involves someone creating a clever imitation of PGP that behaves like PGP in most respects, but that doesn't work the way it's supposed to. For example, it might be deliberately crippled to not check signatures properly, allowing bogus key certificates to be accepted. You should make an effort to get your copy of PGP directly from Pretty Good Privacy.

There are other ways to check PGP for tampering, using digital signatures. You could use another trusted version of PGP to check the signature on a suspect version of PGP. But this won't help at all if your operating system is infected, nor will it detect if your original copy of pgp.exe has been maliciously altered in such a way as to compromise its own ability to check signatures. This test also assumes that you have a good trusted copy of the public key that you use to check the signature on the PGP executable.

Swap files or virtual memory

PGP was originally developed for MS-DOS, a primitive operating system by today's standards. But as it was ported to other more complex operating systems, such as Microsoft Windows and the Macintosh OS, a new vulnerability emerged. This vulnerability stems from the fact that these fancier operating systems use a technique called *virtual memory.*

Virtual memory allows you to run huge programs on your computer that are bigger than the space available in your computer's semiconductor memory chips. This is handy because software has become more and more bloated since graphical user interfaces became the norm and users started running several large applications at the same time. The operating system uses the hard disk to store portions of your software that aren't being used at the moment. This means that the operating system might, without your knowledge, write out to disk some things that you thought were kept only in main memory—-things like keys, passphrases, and decrypted plaintext. PGP does not keep that kind of sensitive data lying around in memory for longer than necessary, but there is some chance that the operating system could write it out to disk anyway.

The data is written out to some scratchpad area of the disk, known as a *swap file*. Data is read back in from the swap file as needed, so that only part of your program or data is in physical memory at any one time. All this activity is invisible to the user, who just sees the disk chattering away. Microsoft Windows swaps chunks of memory, called *pages*, using a Least Recently Used (LRU) page-replacement algorithm. This means pages that have not been accessed for the longest period of time are the first ones to be swapped to the disk. This approach suggests that in most cases the risk is fairly low that sensitive data will be swapped out to disk, because PGP doesn't leave it in memory for very long. But we don't make any guarantees.

This swap file can be accessed by anyone who can get physical access to your computer. If you are concerned about this problem, you may be able to solve it by obtaining special software that overwrites your swap file. Another possible cure is to turn off your operating system's virtual memory feature. Microsoft Windows allows this, and so does the Mac OS. Turning off virtual memory may mean that you need to have more physical RAM chips installed in order to fit everything in RAM.

Physical security breach

A physical security breach may allow someone to physically acquire your plaintext files or printed messages. A determined opponent might accomplish this through burglary, trash-picking, unreasonable search and seizure, or bribery, blackmail, or infiltration of your staff. Some of these attacks may be especially feasible against grass-roots political organizations that depend on a largely volunteer staff.

Don't be lulled into a false sense of security just because you have a cryptographic tool. Cryptographic techniques protect data only while it's encrypted—direct physical security violations can still compromise plaintext data or written or spoken information.

This kind of attack is cheaper than cryptanalytic attacks on PGP.

Tempest attacks

Another kind of attack that has been used by well-equipped opponents involves the remote detection of the electromagnetic signals from your computer. This expensive and somewhat labor-intensive attack is probably still cheaper than direct cryptanalytic attacks. An appropriately instrumented van can park near your office and remotely pick up all of your keystrokes and messages displayed on your computer video screen. This would compromise all of your passwords, messages, and so on. This attack can be thwarted by properly shielding all of your computer equipment and network cabling so that it does not emit these signals. This shielding technology, known as “Tempest,” is used by some government agencies and defense contractors. There are hardware vendors who supply Tempest shielding commercially.

Protecting against bogus timestamps

A somewhat obscure vulnerability of PGP involves dishonest users creating bogus timestamps on their own public key certificates and signatures. You can skip over this section if you are a casual user and aren’t deeply into obscure public-key protocols.

There’s nothing to stop a dishonest user from altering the date and time setting of his own system’s clock, and generating his own public key certificates and signatures that appear to have been created at a different time. He can make it appear that he signed something earlier or later than he actually did, or that his public/private key pair was created earlier or later. This may have some legal or financial benefit to him, for example by creating some kind of loophole that might allow him to repudiate a signature.

I think this problem of falsified timestamps in digital signatures is no worse than it is already in handwritten signatures. Anyone can write any date next to their handwritten signature on a contract, but no one seems to be alarmed about this state of affairs. In some cases, an “incorrect” date on a handwritten signature might not be associated with actual fraud. The timestamp might be when the signator asserts that he signed a document, or maybe when he wants the signature to go into effect.

In situations where it is critical that a signature be trusted to have the actual correct date, people can simply use notaries to witness and date a handwritten signature. The analog to this in digital signatures is to get a trusted third party to sign a signature certificate, applying a trusted timestamp. No exotic or overly formal protocols are needed for this. Witnessed signatures have long been recognized as a legitimate way of determining when a document was signed.

A trustworthy Certifying Authority or notary could create notarized signatures with a trustworthy timestamp. This would not necessarily require a centralized authority. Perhaps any trusted introducer or disinterested party could serve this function, the same way real notary publics do now. When a notary signs other people's signatures, it creates a signature certificate of a signature certificate. This would serve as a witness to the signature in the same way that real notaries now witness handwritten signatures. The notary could enter the detached signature certificate (without the actual whole document that was signed) into a special log controlled by the notary. Anyone could read this log. The notary's signature would have a trusted timestamp, which might have greater credibility or more legal significance than the timestamp in the original signature.

There is a good treatment of this topic in Denning's 1983 article in IEEE Computer. Future enhancements to PGP might have features to easily manage notarized signatures of signatures, with trusted timestamps.

Exposure on multi-user systems

PGP was originally designed for a single-user PC under your direct physical control. If you run PGP at home on your own PC, your encrypted files are generally safe, unless someone breaks into your house, steals your PC and persuades you to give them your passphrase (or your passphrase is simple enough to guess).

PGP is not designed to protect your data while it is in plaintext form on a compromised system. Nor can it prevent an intruder from using sophisticated measures to read your private key while it is being used. You will just have to recognize these risks on multiuser systems, and adjust your expectations and behavior accordingly. Perhaps your situation is such that you should consider only running PGP on an isolated single-user system under your direct physical control.

Traffic analysis

Even if the attacker cannot read the contents of your encrypted messages, he may be able to infer at least some useful information by observing where the messages come from and where they are going, the size of the messages, and the time of day the messages are sent. This is analogous to the attacker looking at your long-distance phone bill to see who you called and when and for how long, even though the actual content of your calls is unknown to the attacker. This is called traffic analysis. PGP alone does not protect against traffic analysis. Solving this problem would require specialized communication protocols designed to reduce exposure to traffic analysis in your communication environment, possibly with some cryptographic assistance.

Cryptanalysis

An expensive and formidable cryptanalytic attack could possibly be mounted by someone with vast supercomputer resources, such as a government intelligence agency. They might crack your RSA key by using some new secret factoring breakthrough. But civilian academia has been intensively attacking it without success since 1978.

Perhaps the government has some classified methods of cracking the IDEA conventional encryption algorithm used in PGP. This is every cryptographer's worst nightmare. There can be no absolute security guarantees in practical cryptographic implementations.

Still, some optimism seems justified. The IDEA algorithm's designers are among the best cryptographers in Europe. It has had extensive security analysis and peer review from some of the best cryptanalysts in the unclassified world. It appears to have some design advantages over DES in withstanding differential cryptanalysis.

Besides, even if this algorithm has some subtle unknown weaknesses, PGP compresses the plaintext before encryption, which should greatly reduce those weaknesses. The computational workload to crack it is likely to be much more expensive than the value of the message.

If your situation justifies worrying about very formidable attacks of this caliber, then perhaps you should contact a data security consultant for some customized data security approaches tailored to your special needs.

In summary, without good cryptographic protection of your data communications, it may be practically effortless and perhaps even routine for an opponent to intercept your messages, especially those sent through a modem or e-mail system. If you use PGP and follow reasonable precautions, the attacker will have to expend far more effort and expense to violate your privacy.

If you protect yourself against the simplest attacks, and you feel confident that your privacy is not going to be violated by a determined and highly resourceful attacker, then you’ll probably be safe using PGP. PGP gives you Pretty Good Privacy.

Appendix A

Transferring Files Between the MacOS and Windows using PGP

Transferring files to and from MacOS is a classic problem in using almost any kind of data exchange software, such as e-mail applications, FTP, compression utilities, and PGP. This appendix is intended to document how this problem is finally solved by PGP Version 5.5, and to discuss how to communicate with previous versions of PGP.

The MacOS stores files differently from other operating systems. Even the text file format of the MacOS is different. MacOS files are really two files consisting of a Data segment and a Resource segment. In order to send a file from MacOS to Windows without losing data, the two segments must be merged into one. The standard method by which a MacOS file is converted into a single file so that it can be transferred to another Macintosh or PC without losing either of its halves is called MacBinary.

The problem is that, without special software, Windows and other platforms cannot inherently understand the MacBinary format. If a situation occurs where the receiving software fails to convert a MacBinary format file into a Windows file, the resulting file is unusable. Third-party utilities exist on Windows to convert it after the fact into a usable file, but that can be rather inconvenient.

Previous versions of PGP and most utilities available on the market today generally try to ignore this problem as much as possible and leave all decisions up to the user as to whether or not to encode a file with MacBinary when sending from MacOS. This places the burden of deciding to send with MacBinary, and not risk losing any data, or send without MacBinary, with hope that no important data will be lost on the user, who often has no idea what the correct decision is. The decision should generally be based on whether the file is being sent to Windows or MacOS. But what about if you're sending to both at the same time? There is no good solution to that problem with older versions of PGP and many other utilities. This has resulted in great confusion and inconvenience for users.

The reverse, sending a file from Windows to the MacOS, has also been a major problem. Windows uses filename extensions, such as .doc, to identify the type of a file. This is meaningless on MacOS. These files are sent to a Macintosh computer without any file type or creator information. The process of making them readable after receipt generally involves various arcane motions in the Open dialog of the creator application, or in many cases requires the user to understand MacOS lore of creator and type codes by setting them manually in a third-party utility.

Fortunately, PGP Version 5.5 finally leads the way out of this confusion. If all PGP users were to use PGP Version 5.5, no one would have to think about how to send files from MacOS to Windows and vice versa.

Sending from the MacOS to Windows

On the MacOS, there are three options in PGP 5.5 when encrypting or signing a file on MacOS:

- MacBinary: Yes
- MacBinary: No
- MacBinary: Smart

MacBinary: Yes

This is the recommended option for all encryptions when sending to another user of PGP Version 5.5 or above on any platform. This means that MacOS users will receive the exact file that was intended, and the Windows version will automatically decode the MacBinary and even append the appropriate file extension, such as .doc for Microsoft Word or .ppt for Microsoft PowerPoint. PGP includes information on most popular application filename extensions and Macintosh-creator codes. In cases where the type is unknown or known to be a MacOS-only file such as a MacOS application, the file remains in MacBinary format so that it can later be forwarded to a Macintosh fully intact.

MacBinary: No

If you are communicating with users who have an older version of PGP, the decision of whether to send with MacBinary generally ends up in the sender's hands as in most other programs and in previous versions of PGP for MacOS. When sending to a PC using an older version, if you know that the file you are sending can be read by Windows applications when no MacBinary is used, select this option. This includes most files that are generally cross-platform such as those created by the Microsoft Office applications, graphics files, compressed files, and many others. The sender or the recipient will have to manually rename the file to have the correct filename extension on Windows. This is required because the Windows recipient does not have the creator information normally encoded with MacBinary.

MacBinary: Smart

There are some very limited cases where this option can be useful when communicating with users who are not using Version 5.5. This option makes a decision as to whether to encode with MacBinary based on an analysis of the actual data in the file. If the file is one of the following types, it will not be encoded with MacBinary, thereby making it readable on a PC with any version of PGP:

- * PKzip compressed file
- * Lempel-Ziv compressed file
- * MIDI music format file

- * PackIt compressed file
- * GIF graphics file
- * StuffIt_ compressed file
- * Compactor compressed file
- * Arc compressed file
- * JPEG graphics file

As shown, only a limited selection of files will result in a readable file by old versions of PGP on other platforms using the Smart option. Any other file received on a PC with an older version of PGP will be unreadable without stripping the MacBinary encoding with a third-party utility. Also, the file will not have the correct filename extension on the PC unless that extension was manually added by the user on the sending side. Using Smart mode, the resulting file may not be the same as the original when sent to a Macintosh, because it may lose its creator and type codes. This mode remains in the product mostly due to the fact that it was in PGP Version 5.0 and some users may only have a need to send the above file types. This option is not recommended in most cases.

In summary, if you are sending only to Versions 5.5 or above, always select MacBinary: Yes (the default). Thus, no thought is required if your environment is using PGP Version 5.5 exclusively. When sending to users with older versions, you should select MacBinary: No for cross-platform file types and MacBinary: Yes for files which simply wouldn't be readable to PC users anyway (such as a MacOS application).

PGP Version 5.0 Note: PGP Version 5.0 did not have a MacBinary: No option. In order to send file types without MacBinary, which are not included in the MacBinary: Smart list to a PC using 5.0, the file must be manually set to one of the creator and type codes on the Smart list before sending.

Receiving Windows files on the MacOS

When decrypting, PGP Version 5.5 automatically attempts to translate filename extensions for non-MacBinary files into MacOS creator and type information. For example, if you receive a file from Windows with an extension of .doc, the file will be saved as a Microsoft Word document. The same list of applications used when adding filename extensions upon receipt of a MacBinary file on Windows is used to translate filename extensions back into the MacOS equivalent when received on a Macintosh computer. In almost all cases, this results in files which are immediately readable and double-clickable on MacOS.

Previous versions of PGP for MacOS do not have this feature. The user will have to manually determine that a file named “report.doc” is a Microsoft Word file. After determining the creator application, in the case of Microsoft Word, one can simply use the Open dialog to open the file by selecting Show All Files from the popup menu. Many other applications also have this feature, but some don't. If the document cannot be opened from within the application, the user must find out what the appropriate Macintosh creator and type codes are for the file and manually set them with a third-party utility. There are many free utilities to do this. Upgrading to Version 5.5 is probably the easiest option in this case, as it eliminates this problem.

Supported Applications

The following list of major applications produce documents which are automatically translated by PGP 5.5 when sent from Windows to MacOS and vice versa. At this time, there is no way for a user to add or change these conversions, however, we may add such functionality in the future.

- PhotoShop(GIF, native Photoshop documents, TGA, JPEG)
- PageMaker(Versions 3.X, 4.X, 5.X, 6.X)
- Microsoft Project(project and template files)
- FileMaker Pro
- Adobe Acrobat
- Lotus 123
- Microsoft Word (text, RTF, templates)
- PGP
- Microsoft PowerPoint
- StuffIt
- QuickTime

- Corel WordPerfect
- Microsoft Excel(many different types of files)
- Quark XPress

The following general filename extensions are also converted:

cvs	.arj	.ima	.eps	.mac	.cgm
.dl	.fli	.ico	.iff	.img	.lbm
msp	.pac	pbm	.pcs	.pcx	.pgm
plt	.pm	.ppm	.rif	.rle	.shp
spc	.sr	.sun	.sup	.wmf	.flc
.gz	.vga	hal	lzh	.Z	.exe
.mpg	dvi	tex	.aif,	zip	.au
mod,	.svx	.wav	.tar	.pct	pic
.pit	.txt	mdi	pak	.tif,	.eps

A Glossary of Terms

ASCII-armored text: Binary information that has been encoded using a standard, printable, 7-bit ASCII character set, for convenience in transporting the information through communication systems. In the PGP program, ASCII armored text files are given the default filename extension, and they are encoded and decoded in the ASCII radix-64 format.

authentication: The determination of the origin of encrypted information through the verification of someone’s digital signature or someone’s public key by checking its unique fingerprint.

certify: To sign another person's public key.

certifying authority: One or more trusted individuals who are assigned the responsibility of certifying the origin of keys and adding them to a common database.

conventional encryption: Encryption that relies on a common passphrase instead of public key cryptography. The file is encrypted using a session key, which encrypts using a passphrase that you will be asked to choose

decryption: A method of unscrambling encrypted information so that it becomes legible again. The recipient's private key is used for decryption.

digital signature: See signature.

encryption: A method of scrambling information to render it unreadable to anyone except the intended recipient, who must decrypt it to read it.

fingerprint: A uniquely identifying string of numbers and characters used to authenticate public keys. This is the primary means for checking the authenticity of a key.

introducer: A person or organization who is allowed to vouch for the authenticity of someone's public key. You designate an introducer by signing their public key.

key: A digital code used to encrypt and sign and decrypt and verify e-maile-mail messages and files. Keys come in key pairs and are stored on keyrings.

key escrow: A practice where a user of a public key encryption system surrenders their private key to a third party thus permitting them to monitor encrypted communications.

key fingerprint: A uniquely identifying string of numbers and characters used to authenticate public keys. For example, you can telephone the owner of a public key and have him or her read the fingerprint associated with their key so you can compare it with the fingerprint on your copy of their public key to see if they match. If the fingerprint does not match, then you know you have a bogus key.

key ID: A legible code that uniquely identifies a key pair. Two key pairs may have the same user ID, but they will have different Key IDs.

key pair: A public key and its complimentary private key. In public-key cryptosystems, like the PGP program, each user has at least one key pair.

keyring: A set of keys. Each user has two types of keyrings: a private keyring and a public keyring.

message digest: A compact "distillate" of your message or file checksum. It represents your message, such that if the message were altered in any way, a different message digest would be computed from it

passphrase: A series of keystrokes that allow exclusive access to your private key which you use to sign and decrypt e-mail messages and file attachments.

plaintext: Normal, legible, unencrypted, unsigned text.

private key: The secret portion of a key pair-used to sign and decrypt information. A user's private key should be kept secret, known only to the user.

private keyring: A set of one or more private keys, all of which belong to the owner of the private keyring.

public key: One of two keys in a key pair-used to encrypt information and verify signatures. A user's public key can be widely disseminated to colleagues or strangers. Knowing a person's public key does not help anyone discover the corresponding private key.

public keyring: A set of public keys. Your public keyring includes your own public key(s).

public-key cryptography: Cryptography in which a public and private key pair is used, and no security is needed in the channel itself.

sign: To apply a signature.

signature: A digital code created with a private key. Signatures allow authentication of information by the process of signature verification. When you sign a message or file, the PGP program uses your private key to create a digital code that is unique to both the contents of the message and your private key. Anyone can use your public key to verify your signature.

text: Standard, printable, 7-bit ASCII text.

trusted: A public key is said to be trusted by you if it has been certified by you or by someone you have designated as an introducer.

trusted introducer: Someone who you trust to provide you with keys that are valid. When a trusted introducer signs another person’s key, you trust that their keys are valid, and you do not need to verify their keys before using them.

user ID: A text phrase that identifies a key pair. For example, one common format for a user ID is the owner's name and e-mail address. The user ID helps users (both the owner and colleagues) identify the owner of the key pair.

verification: The act of comparing a signature created with a private key to its public key. Verification proves that the information was actually sent by the signer, and that the message has not been subsequently altered by anyone else.

web of trust:: A distributed trust model used by PGP to validate the ownership of a public key where the level of trust is cumulative, based on the individuals’ knowledge of the introducers.

Index