

Pretty Good Privacy™
PGP for Personal Privacy, Version 5.0

For
Windows® 95
Windows NT

User's Guide

PGP™, Inc.

© 1997 by Pretty Good Privacy, Inc. All rights reserved.
5-97. Printed in the United States of America.

PGP for Personal Privacy, Version 5.0

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

Copyright © [1990], 1997 by Pretty Good Privacy, Inc. All Rights Reserved.

PGP, Pretty Good, and Pretty Good Privacy are registered trademarks of Pretty Good Privacy, Inc. All other trademarks and registered trademarks are the property of their respective owners.

Pretty Good Privacy, Inc. may have patents and/or pending patent applications covering subject matter in this document. The furnishing of this document or the software does not give you any license to these patents.

PGP uses public key algorithms described in U.S. Patent numbers 4,200,770, 4,218,582, 4,405,829, and 4,424,414, licensed exclusively by Public Key Partners.

PGP uses the IDEA cryptographic cipher described in U.S. Patent number 5,214,703 and is licensed from Ascom Tech AG. IDEA is a trademark of Ascom Tech, AG.

The compression code in PGP is by Mark Adler and Jean-loup Gailly, taken with permission from the free Info-ZIP implementation.

The software provided with this documentation is licensed to you for your individual use under the terms of the End User License Agreement and Limited Warranty provided with the software. The information in this document is subject to change without notice. Pretty Good Privacy, Inc. does not warrant that the information meets your requirements or that the information is free of errors. The information may include technical inaccuracies or typographical errors. Changes may be made to the information and incorporated in new editions of this document, if and when made available by Pretty Good Privacy, Inc.

Export of this software and documentation may be subject to compliance with the rules and regulations promulgated from time to time by the Bureau of Export Administration, United States Department of Commerce, which restrict the export and re-export of certain products and technical data.

PRETTY GOOD PRIVACY, INC.
2121 South El Camino Real, Suite 902
San Mateo, CA 94403
(415) 631-1747
(415) 572-1932 fax
info@pgp.com
<http://www.pgp.com>

LIMITED WARRANTY. Pretty Good Privacy, Inc. warrants that the Software will perform substantially in accordance with the written materials in this package for a period of 90 days from the date of original purchase. Pretty Good Privacy, Inc.'s entire liability and your exclusive remedy shall be, at Pretty Good Privacy, Inc.'s option, either (a) return of the purchase price paid for the license or (b) repair or replacement of the Software that does not meet Pretty Good Privacy, Inc.'s limited warranty and which is returned at your expense to Pretty Good Privacy, Inc. with a copy of your receipt. This limited warranty is void if failure of the Software has resulted from accident, abuse, or misapplication. Any repaired or replacement Software will be warranted for the remainder of the original warranty period or 30 days, whichever is longer.

IF THE SOFTWARE IS EXPORT CONTROLLED (SEE BELOW), THESE REMEDIES ARE NOT AVAILABLE OUTSIDE THE UNITED STATES OF AMERICA. NO OTHER WARRANTIES. EXCEPT FOR THE WARRANTIES SET FORTH HEREIN, THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" AND PRETTY GOOD PRIVACY, INC. DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, CONFORMANCE WITH DESCRIPTION, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE TO STATE. LIMITATION OF LIABILITY. PRETTY GOOD PRIVACY, INC.'S CUMULATIVE 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 THE LICENSE. IN NO EVENT SHALL PRETTY GOOD PRIVACY, INC. 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 OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE, EVEN IF PRETTY GOOD PRIVACY, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

***This book was written by Mike Iannamico
special thanks to Gail Kesner Haspert***

Table of Contents

| | |
|--|----------|
| Table of Contents | v |
| Chapter 1: Introducing PGP for Personal Privacy | 1 |
| A Quick Overview | 2 |
| Create a Private and Public Key Pair | 3 |
| Exchange Public Keys with Others | 3 |
| Certify and Validate Your Keys | 3 |
| Encrypt and Sign Your Email | 4 |
| Decrypt and Verify Your Email | 4 |
| About This Manual | 5 |
| Chapter 1 <i>Introducing PGP for Personal Privacy</i> | 5 |
| Chapter 2 <i>Getting Started</i> | 5 |
| Chapter 3 <i>Making and Exchanging Keys</i> | 5 |
| Chapter 4 <i>Sending and Receiving Private E-mail</i> | 5 |
| Chapter 5 <i>Managing Keys And Setting Preferences</i> | 5 |
| Chapter 6 <i>Security Features and Vulnerabilities</i> | 5 |

| | |
|---|-----------|
| Chapter 2: Getting Started | 7 |
| System Requirements | 7 |
| Compatibility with Other Versions | 7 |
| Upgrading from a Previous Version | 8 |
| Upgrading from PGP 2.6.2 (MIT Freeware) | 9 |
| Upgrading from PGPmail 4.0 | 10 |
| Upgrading from PGPmail 4.5 | 10 |
| Upgrading from previous beta versions of PGPmail 5.0: | 11 |
| Installing PGP | 12 |
| To Install PGP from a CD ROM: | 12 |
| To Install PGP from PGP's Web site: | 12 |
| Running PGP | 12 |
| Using PGP from the System tray | 13 |
| Performing PGP functions from the Clipboard | 13 |
| Opening the PGPkeys Window | 14 |
| Setting Preferences | 15 |
| Getting Help | 15 |
| Quitting PGP | 15 |
| Using PGP from Supported E-mail Applications | 16 |
| Using PGP from the Windows Explorer | 17 |
| Selecting Recipients | 18 |
| Taking Shortcuts | 18 |
| Chapter 3: Making and Exchanging Keys | 21 |
| Key Concepts | 21 |
| Making a Key Pair | 22 |
| To create a new key pair | 23 |

| | |
|---|-----------|
| Protecting Your Keys | 33 |
| Distributing Your Public Key | 35 |
| Making your Public Key Available Through a Key Server | 35 |
| To send your public key to a key server | 36 |
| Including your Public Key in an E-mail Message | 37 |
| To include your public key in an e-mail message | 37 |
| Exporting your Public Key to a File | 37 |
| Obtaining the Public Keys of Others | 38 |
| Getting Public Keys from a Key Server | 38 |
| To get someone's public key from a key server | 39 |
| Adding Public Keys from E-mail Messages | 39 |
| Importing a Public Key from a File | 40 |
| Verifying the Authenticity of a Key | 40 |
| Chapter 4: Sending and Receiving Private E-mail. | 43 |
| Encrypting and Signing E-mail | 43 |
| Encrypting and Signing with Supported E-mail Applications | 43 |
| To encrypt and sign with supported e-mail applications | 44 |
| Encrypting and Signing Via the Clipboard | 46 |
| To encrypt and sign via the clipboard | 46 |
| Encrypting and Signing from the Windows Explorer | 48 |
| To encrypt and sign from the Windows Explorer | 48 |
| Decrypting and Verifying E-mail | 51 |
| Decrypting and Verifying from Supported e-mail Applications | 51 |
| To decrypt and verify with supported e-mail applications | 52 |
| Decrypting and Verifying Via the Clipboard | 53 |
| To decrypt and verify using the clipboard | 53 |
| Decrypting and Verifying from the Windows Explorer | 54 |
| To decrypt and verify from the Windows Explorer | 54 |

| | |
|--|-----------|
| Chapter 5: Managing Keys And Setting Preferences. | 57 |
| Managing Your Keys | 57 |
| The PGPkeys Window | 58 |
| PGPkeys Attribute Definitions | 59 |
| PGPkeys Icon Definitions | 60 |
| Examining a Key's Properties | 62 |
| Specifying a Default Key Pair | 63 |
| To specify your default key pair | 64 |
| Adding a New User Name or Address | 64 |
| To add a new user name or address to an existing key | 64 |
| Checking a Key's Fingerprint | 65 |
| To check a key's fingerprint | 65 |
| Signing Someone's Public Key | 66 |
| To sign someone's public key | 66 |
| Granting Trust for Key Validations | 67 |
| To grant trust for a key | 67 |
| Disabling and Enabling Keys | 68 |
| To disable a key | 68 |
| To enable a key | 69 |
| Deleting a Key or Signature | 69 |
| To delete a key, signature or user ID | 69 |
| Changing your Passphrase | 69 |
| To change your passphrase | 69 |
| Importing and Exporting Keys | 71 |
| To import a key from a file | 71 |
| To export a key to a file | 72 |
| Revoking a Key | 72 |
| To revoke a key | 72 |
| Setting Your Preferences | 74 |
| General Preferences | 74 |

| | |
|---|-----------|
| Key Files Preferences | 76 |
| E-mail Preferences | 77 |
| Key Server Preferences | 79 |
| Chapter 6: Security Features and Vulnerabilities | 81 |
| Why I wrote PGP | 81 |
| Encryption Basics | 86 |
| How Public Key Cryptography Works | 86 |
| How Your Files and Messages are Encrypted | 87 |
| The PGP Symmetric Algorithms | 88 |
| Data Compression | 90 |
| About the Random Numbers used as Session Keys | 91 |
| How Decryption Works | 91 |
| How Digital Signatures Work | 92 |
| About the Message Digest | 94 |
| How to Protect Public Keys from Tampering | 95 |
| How Does PGP Keep Track of Which Keys are Valid? | 99 |
| How to Protect Private Keys from Disclosure | 101 |
| What If You Lose Your Private Key? | 102 |
| Beware of Snake Oil | 103 |
| Vulnerabilities | 108 |
| Compromised passphrase and Private Key | 108 |
| Public Key Tampering | 109 |
| Not Quite Deleted Files | 109 |
| Viruses and Trojan Horses | 110 |
| Swap Files or Virtual Memory | 111 |
| Physical Security Breach | 112 |
| Tempest Attacks | 112 |
| Protecting Against Bogus Timestamps | 112 |
| Exposure on Multi-user Systems | 114 |
| Traffic Analysis | 114 |
| Cryptanalysis | 114 |

| | |
|---|------------|
| Recommended Introductory Readings | 115 |
| Other Readings: | 116 |
| A Glossary of Terms | 117 |
| Index | 121 |

Introducing PGP for Personal Privacy

With PGP™ for Personal Privacy, you can easily protect the privacy of your email messages and file attachments by encrypting them so that only those with the proper authority can decipher the information. You can also digitally sign the messages and files you exchange, which ensures that they have come from the person who allegedly sent them and that the information has not been tampered with in any way while in transit.

The most convenient way to use PGP is through one of the popular email applications supported by the plug-ins. This allows you to encrypt and sign as well as decrypt and verify your messages while you are composing and reading your e-mail. In addition, if you are communicating with another PGP user who is using an email application that adheres to the PGP/MIME standard, you can perform all of the PGP functions on both your messages and any file attachments by simply clicking a button when sending or receiving your e-mail.

If you are using an email application that is not supported by the plug-ins, you can easily transfer the text of your email messages to the clipboard and perform the necessary functions from there. In addition, if you need to encrypt or decrypt entire file attachments, you can do so directly from the Windows Explorer by choosing the appropriate menu option.

Here are some of the features offered by PGP:

- Widely-trusted encryption and decryption incorporating maximum-strength cryptographic technologies
- Digital signature and verification for certifying messages and files
- Quick access to all functions from easily selectable menu items

- Integrated plug-in support for popular email applications
- Implementation of PGP/MIME for quick encryption and decryption of messages and file attachments when sending and receiving email
- Simple key generations with up to 4096-bit keys and support for multiple key formats (RSA and DSS/Diffie-Hellman)
- Sophisticated key management with graphical representations of key properties
- Integrated support for distributing and retrieving keys from public key servers

NOTE:

If you are running the DSS/Diffie-Hellman version of PGP for Personal Privacy, it does not generate keys using the RSA algorithm nor does it encrypt, decrypt, sign, or verify using RSA keys. If you find that you need to generate keys or otherwise use the RSA algorithm, see the vendor from whom you bought your PGP product.

A Quick Overview

PGP is based on a widely accepted encryption technology known as “public key cryptography” in which two complementary keys are used to maintain secure communications. One of the keys is a private key to which only you have access and the other is a public key which you freely exchange with other PGP users. Both, your private and your public keys are stored in keyring files which are accessible from the PGPkeys window in which you perform all your key management functions.

To send someone a private email 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 also use your private key to sign the email you send to others. The recipients can then use their copy of your public key to determine if you really sent the email and whether it has been altered while in transit. When someone sends you email 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 email messages and file attachments.

The following section provides a quick run-through of 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 where they are fully explained.

Create a Private and Public Key Pair

Before you can begin using PGP, you need to generate a key pair consisting 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 email. 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 window.

Exchange Public Keys with Others

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

Certify and 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 their 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 them to vouch for the authenticity of someone else's public key.

Encrypt and Sign Your Email

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

- If you are using an email application supported by the plug-ins, you can encrypt and sign your messages by selecting the appropriate options from your application's tool bar. In addition, if you are communicating with other PGP users who are using a version which adheres to the PGP/MIME standard, you can encrypt and sign messages as well as file attachments automatically when you send your mail.
- If your email 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 include any file attachments, you can encrypt and sign them from the Windows Explorer before attaching them to your email.

Decrypt and Verify Your Email

When someone sends you encrypted email, 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 email 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. In addition, if your email application supports the PGP/MIME standard, you can decrypt and verify messages and file attachments sent using this format by clicking on an icon when reading your mail.
- If your email 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 Windows Explorer.

About This Manual

This manual is organized in the following manner:

Chapter 1 *Introducing PGP for Personal Privacy*

Describes the purpose of the program, delves into the concept of public key encryption and digital signatures and provides a quick overview of how you will use the program.

Chapter 2 *Getting Started*

Runs through the steps needed to install and run the PGP program with a brief discussion of the main components and primary functions.

Chapter 3 *Making and Exchanging Keys*

Explains how to generate your private and public key pair and describes the methods for exchanging, protecting and authenticating keys.

Chapter 4 *Sending and Receiving Private E-mail*

Explains how to send and receive email messages and file attachments depending on the type of email application you and the recipients of your email are using.

Chapter 5 *Managing Keys And Setting Preferences*

Explains how to examine and alter a key's attributes and how to establish preferences for the PGP program.

Chapter 6 *Security Features and Vulnerabilities*

This chapter is provided by Phil Zimmermann and describes the basic concepts behind public key encryption and elaborates on some of the vulnerabilities.

Getting Started

This chapter explains how to run PGP and provides a quick overview of the procedures you will normally follow in the course of using the product. Based on this information, you will have a fairly good understanding of how to use PGP, which should be especially appreciated by those who don't want to read through the entire manual before beginning to use the product.

System Requirements

Windows 95 or NT

8 MB RAM

15 MB hard disk space

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 2 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 those who are still using these older versions of the product:

PGP 2.6 (Released by MIT)

PGP 4.0 (Released by Viacrypt)

PGP 4.5 (Released by PGP, Inc.)

Along with the new user interface and other improvements, one of the distinct differences between this version of PGP and its predecessors is the ability to generate a new type of key. In addition to the RSA keys used by previous versions, PGP gives you the option of using keys based on the DSS/Diffie-Hellman encryption and digital signature technologies. Although the DSS/Diffie-Hellman keys are provided as an alternative to the traditional RSA keys, you can take advantage of these newer keys only if you are exchanging e-mail with another user who is using a version of PGP which is capable of recognizing these new keys. Considering that it will take a while before the DSS/Diffie-Hellman keys gain widespread use in the user community, you may want to reserve a set of RSA keys so that you can continue to communicate with those who have earlier versions of PGP.

If you are encrypting e-mail to multiple recipients, where some have RSA keys and others have DSS/Diffie-Hellman keys, the e-mail will be encrypted using the appropriate type 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 which remove this limitation.

Another improvement in this version of PGP is the implementation of the PGP/MIME 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 which 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, sending PGP/MIME e-mail to those who are not using an e-mail application that supports this standard may be less convenient for them to decrypt and verify.

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 won't have to go through the trouble of importing all of your old keys. To upgrade from a previous version, follow the appropriate steps listed below:

Upgrading from PGP 2.6.2 (MIT Freeware)

1. Make sure that you have exited all programs currently running on your computer.
2. Locate and 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. You must 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 no-one but you has access to them.

3. Once you have successfully backed up your old keyrings, remove or archive the (old) PGP 2.6.2 software from/on your hard disk. You have two options here:

Manually delete the entire old "PGP262" directory and all of its contents.

Manually delete the "pgp.exe" (262) program and archive the remaining files, especially the config.txt and keyring files.

NOTE: If you obtain a copy of the newly-patched PGP264 version from MIT, your old 2.6.x software will be able to read the RSA keys on the new 5.0 keyrings and will not fail when it encounters the new DSS/Diffie-Hellman format keys.

4. Install PGP 5.0 using the provided InstallShield executable.
5. When the installer asks you if you have existing keyrings click on "yes," locate your old 262 keyrings and follow the instructions to copy those keys to your new PGP 5.0 keyrings.
6. Restart your computer.

Upgrading from PGPmail 4.0

This process is the same as with PGP 2.6.2 (ViaCrypt PGP must be manually removed and/or archived). Be sure to keep backups of your keyrings. See also the ReadMe file for PGP 4.0.1 for UNIX and DOS, which describes the patched version for reading PGP 5.0 keyrings with the old ViaCrypt software.

Upgrading from PGPmail 4.5

1. Make sure that you have exited all programs/processes currently running on your computer.
2. End the PGP Enclyptor process ("enclrypt_32.exe") that may be running (so it can be uninstalled).

To determine if the Enclyptor is running, look for its floating palette or its minimized item in the Taskbar (name = "The Enclyptor"). As an alternate method you can use Control+Alt+Delete to bring up the Task Manager; select the process named "The Enclyptor" and click the "End Task" button.

3. Open the Start menu's Settings/Control Panel item.
4. Double-click **Add/Remove Programs**.
5. Select the "PGPmail 4.5" item.
6. Click the **Add/Remove** button. Allow the Uninstall utility to automatically delete all the necessary files for you, and clean up your Registry file.

NOTE: If you are asked about whether to delete any ".dll" files during the uninstall process, it's safe to remove them. The new PGP 5.0 software will install newer versions of those files.

7. Click on **OK** to complete the removal and dismiss the Add/Remove panel when complete.
8. Install the new PGP 5.0 software using the provided InstallShield utility. It's recommended, but not mandatory, to direct the installer to the default install directory.

9. When the installer asks you if you have existing keyrings click on "yes," locate your old PGP 4.5 keyrings and follow the instructions to copy those keys to your new PGP 5.0 keyrings.
10. Restart your computer.

Upgrading from previous beta versions of PGPmail 5.0:

1. Make sure that you have exited all programs/processes currently running on your computer.
2. End the PGPTray task that may be running (so it can be uninstalled).

To determine if "PGPTray.exe" is running, check in the Tray area of the Taskbar for a small PGP "envelope" icon: if present, PGPTray is running. To end the task, click on the PGP tray icon and select the "Quit PGPTray" command at the bottom of the menu. You can also use Control+Alt+Delete to bring up the Task Manager, select the "pgpTray" process and click the "End Task" button.

3. Open the Start menu's Settings/Control Panel item.
4. Double-click on **Add/Remove Programs**.
5. Select the "PGP 5.0bNN" item.
6. Click the **Add/Remove** button.

NOTE: If you are asked whether to delete any ".dll" files during the uninstall process, it's safe to remove them. The new PGP 5.0 software you install will install the newer versions of those files for you.

7. Click on **OK** to complete the uninstallation and dismiss the Add/Remove panel when complete.
8. Install the new PGP 5.0 software using the provided InstallShield utility. It's recommended, but not mandatory, to direct the installer to the default install directory
9. When the installer asks you if you have existing keyrings click on "yes," locate your old PGP 5.0 keyrings and follow the instructions to copy those keys to your new PGP 5.0 keyrings.

10. Restart your computer.

You can now run the new PGP 5.0 software!

Installing PGP

To Install PGP from a CD ROM:

1. Start Windows.
2. Insert the CD ROM.
3. Run the Setup program.
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.

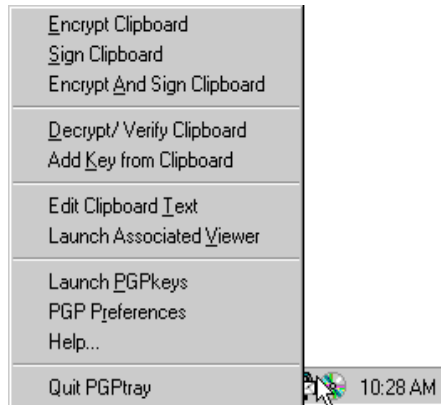
Running PGP

PGP works on the data generated by other applications. As such, 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 three primary ways to use PGP:

- From the System tray
- From within supported e-mail applications
- From the Windows Explorer **File** menu

Using PGP from the System tray

You can access many of the main PGP functions by clicking the lock and key icon, that is normally located in the System tray, and then selecting the appropriate menu item. (If you cannot find this icon in your System tray, you need to run PGP from the **Start** menu).



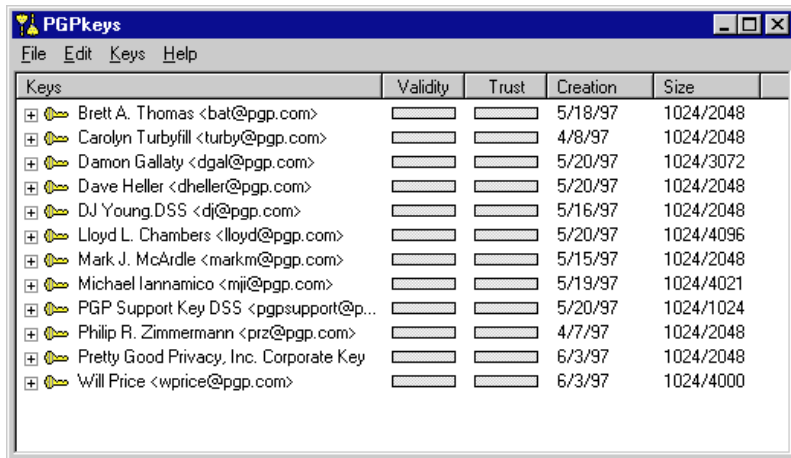
Performing PGP functions from the Clipboard

You will notice that many of the options on this menu refer to PGP functions that you perform from the Windows clipboard. If you are using an e-mail application that is not supported by the PGP plug-ins or you are working with text generated by some other application, you perform your encryption/decryption and signature/verification functions via the Windows clipboard.

For instance, to encrypt or sign text, you copy it from your application to the clipboard, encrypt and sign it using the appropriate PGP functions, then paste it back into your application before sending it to the intended recipient(s). When you receive an encrypted or signed e-mail message, you simply reverse the process and copy the ciphertext from your application to the clipboard, decrypt and verify the information, and then view the contents. After you view the decrypted message, you can decide whether to save the information or retain it in its encrypted form.

Opening the PGPkeys Window

By choosing the **Launch PGPkeys** option from the PGP pop-up menu, you open the PGPkeys window that shows the private and public key pairs you have created for yourself as well as any public keys you have added to your public keyring. (If you have not already created a new key pair, the PGP Key Wizard leads you through the steps necessary to create a new key pair. However, before going through the process of creating a new key pair, you should see Chapter 3 for complete details regarding the various options.

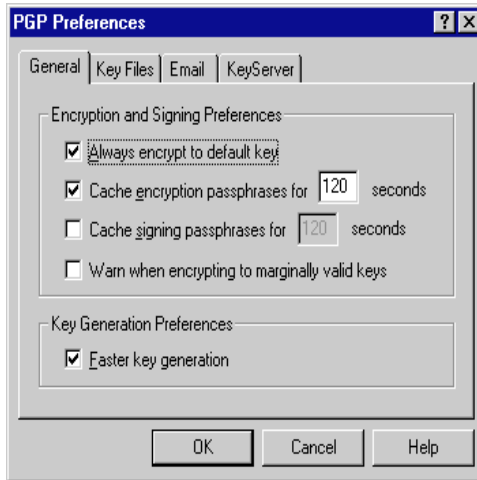


| Keys | Validity | Trust | Creation | Size |
|---|--------------------------|--------------------------|----------|-----------|
| Brett A. Thomas <bat@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/18/97 | 1024/2048 |
| Carolyn Turbyfill <turby@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 4/8/97 | 1024/2048 |
| Damon Gallaty <dgal@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/20/97 | 1024/3072 |
| Dave Heller <dheller@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/20/97 | 1024/2048 |
| DJ Young.DSS <dj@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/16/97 | 1024/2048 |
| Lloyd L. Chambers <lloyd@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/20/97 | 1024/4096 |
| Mark J. McArdle <markm@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/15/97 | 1024/2048 |
| Michael Iannamico <mji@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 5/19/97 | 1024/4021 |
| PGP Support Key DSS <pgpsupport@p... | <input type="checkbox"/> | <input type="checkbox"/> | 5/20/97 | 1024/1024 |
| Philip R. Zimmermann <prz@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 4/7/97 | 1024/2048 |
| Pretty Good Privacy, Inc. Corporate Key | <input type="checkbox"/> | <input type="checkbox"/> | 6/3/97 | 1024/2048 |
| Will Price <wprice@pgp.com> | <input type="checkbox"/> | <input type="checkbox"/> | 6/3/97 | 1024/4000 |

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 user's keys. For a complete explanation of the key management functions you perform from the PGPkeys window, see Chapter 5.

Setting Preferences

By choosing the **PGP Preferences** option from the PGP pop-up menu, you can access the PGP Preferences dialog box where you specify settings which affect how the PGP program functions based on your computing environment.



By clicking on the appropriate tab, you can advance to the preference settings you want to modify. For a complete explanation of these settings, see Chapter 5.

Getting Help

By choosing the **Help** option from the PGP pop-up menu, you can access the PGP help system which provides a general overview and instructions for all of the procedures you are likely to perform. Many of the dialog boxes also have context-sensitive help which you can use by simply clicking the question mark in the right corner of the window and then pointing to the area of interest on the screen for a short explanation.

Quitting PGP

By default, the PGP program runs whenever you start your computer as indicated by the lock and key icon which is displayed in the System tray. If for some reason you need to quit running PGP from the System tray, you can do so by choosing the **Quit PGP** option from the PGP pop-up menu.

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 it.



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.



The key and envelope button will add any keys included in the message onto your keyring. You can also access the PGPkeys window at any time while composing or retrieving your mail by clicking the double keys button.

To make things even simpler, if you are using an e-mail application with one of the plug-ins that adheres to the PGP/MIME standard, and you are communicating with another user who's 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 indicating that it is PGP/MIME encoded.

```
X-Sender: mji@mail.pgp.com (Unverified)
X-Mailer: QUALCOMM Windows Eudora Pro Version 3.0.2 b4 (32)
Date: Wed, 21 May 1997 10:02:32 -0700
To: mji@pgp.com
From: Mike Iannamico <mji@pgp.com>
Subject: test
```

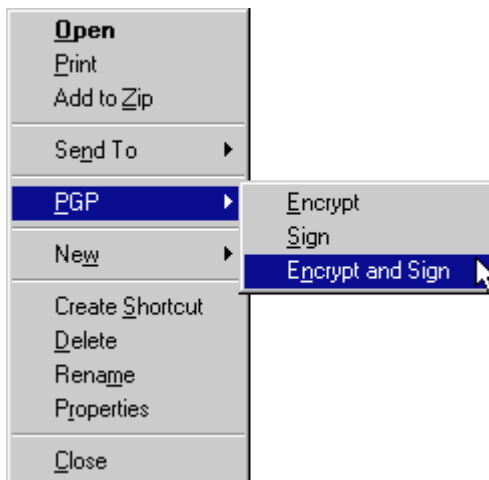


Decrypt PGP/MIME Message

All you need to do in order to decrypt the text and file attachments in PGP/MIME encapsulated e-mail and to verify any digital signatures is to double-click the opened envelope icon.

Using PGP from the Windows Explorer

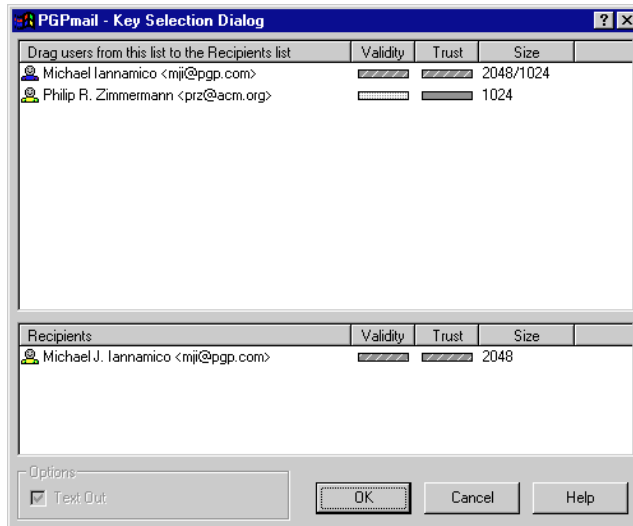
You can encrypt and sign or decrypt and verify files such as word processing documents, spreadsheets and video clips directly from the Windows Explorer. If you are not using an e-mail application such as Eudora that adheres to the PGP/MIME standard, you must use this method to attach files you want to send along with your e-mail messages. In some cases, you might even want to encrypt and decrypt files that you store on your own computer to prevent others from accessing them. To access PGP functions from the Windows Explorer, choose the appropriate option from the **PGP** submenu of the **File** menu.



The options that appear depend on the current state of the file you have selected. If the file has not yet been encrypted or signed, then the options for performing these functions appear on the menu. If the file is already encrypted or signed, then options for decrypting and verifying the contents of the file are displayed.

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 the clipboard or the Windows Explorer, you must manually select the recipient's public key from the PGP Key Selection Dialog box.



All you need do to select a recipient's public key is to drag the icon representing their key into the Recipient's list box and then click **OK**. For complete instructions on how to encrypt and sign and decrypt and verify e-mail, see Chapter 4.

Taking Shortcuts

While 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 instance, while you are managing your keys in the PGPkeys window, you can click the right mouse button to perform all of the necessary PGP functions rather than accessing them from the menu bar. You can also drag a file containing a key into the PGPkeys window to add it to your key ring. Keyboard shortcuts are also available for most menu operations

which allow you to instigate a function by holding down the Ctrl key and some other key. These keyboard shortcuts are shown on all of the PGP menus and the other shortcuts are described in their proper context throughout this manual.

Making and Exchanging Keys

This chapter describes how to generate the private and public 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 certified 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 your public key. 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 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 recipients public key. The recipient then uses their private key to decrypt this key, which allows them to decipher the encrypted data.

Your private key is also used to sign the contents of a given e-mail message or file attachment. Anyone who has a copy of your public key can check your digital signature to confirm that you are the originator of the mail and that the contents have not been altered in any way during transit. In the same way, if you want to verify somebody else's digital signature or check the integrity of the e-mail they send to you, then you need a copy of their public key to do so.

This version of PGP supports two distinct types of keys—the traditional RSA key used in older versions of PGP and a new type of key called DSS/Diffie-Hellman which is based on the latest advancements in cryptographic technologies. If you plan to exchange e-mail with someone who has PGP for Personal Privacy, Version 5.0 or later, then you can take advantage of the new DSS/Diffie-Hellman keys. However, if you are corresponding with someone who is using a previous version of PGP, you have to use the traditional RSA keys to communicate with them.

NOTE: If you are upgrading from an earlier version of PGP, you have probably already generated a private key and have distributed its matching public key to those with whom you correspond. In this case you don't have to make a new key pair (as described in the next section). Instead, you should have specified the location of your keys during the installation process and your keys will show up when you open the PGPkeys window. If you have existing keys and did not specify their location during the installation, you can go to the "Key Files" pane of the Preferences dialog box and enter the correct path to your existing keys.

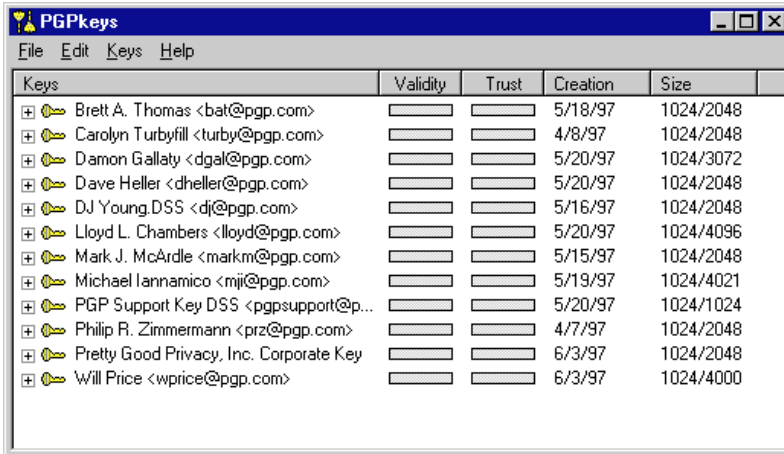
Making a Key Pair

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 Wizard which guides you through the process.

To create a new key pair

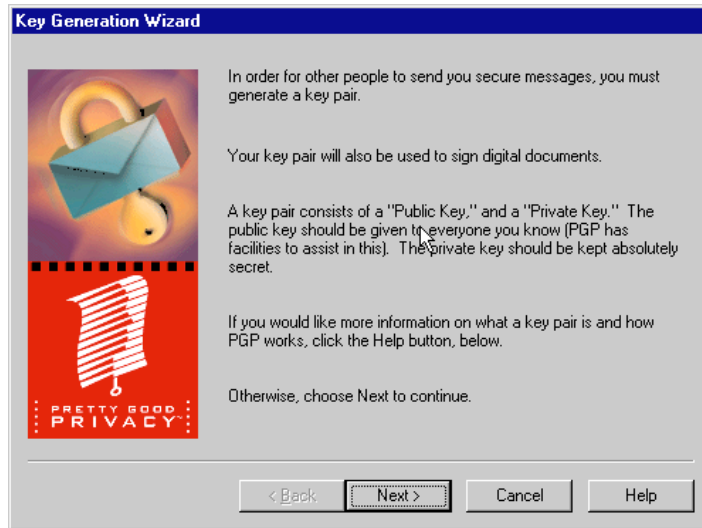
1. Either click the **Start** button and then choose **PGPkeys** from the **PGP** submenu of the **Programs** menu or click the lock and key icon in the System tray and choose **Launch PGPkeys**. You can also open this window by clicking the double keys icon located in your e-mail application's toolbar.

The PGPkeys window opens.



2. Choose **New Key** option from the **Keys** menu.

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

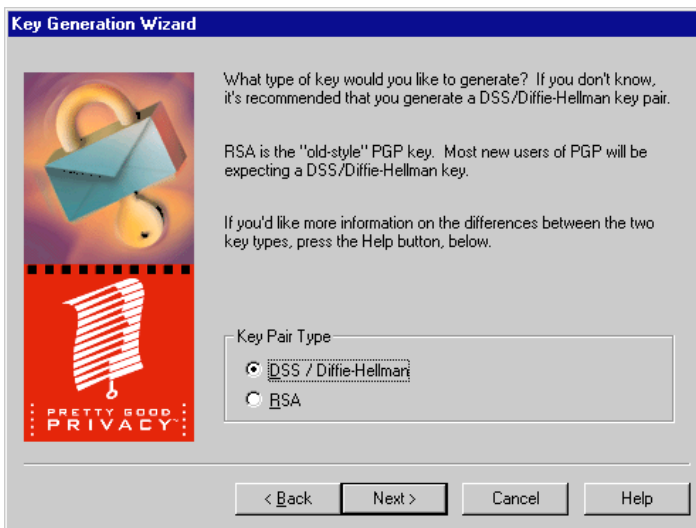


4. Enter your name on the first line and your e-mail address on the second line.

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 one of a 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 Key Generation Wizard then asks you to choose a key type.



6. Select a key type, either DSS/Diffie-Hellman or RSA.

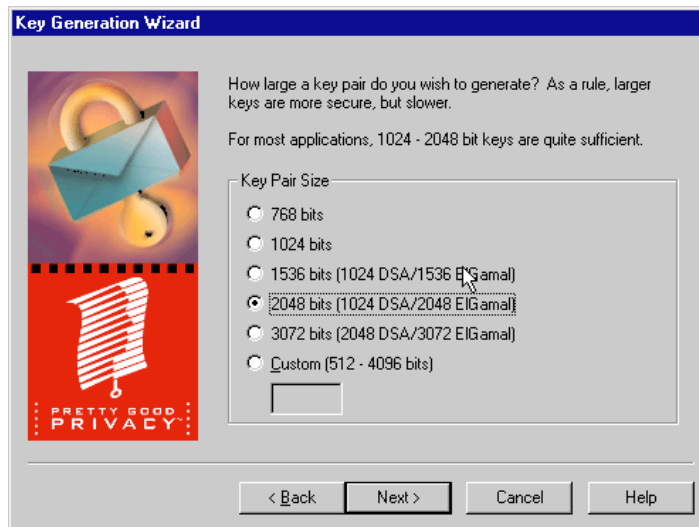
Earlier versions of PGP use an older technology referred to as RSA to generate keys. Beginning with this version of PGP, you have the option of creating a new type of key based on the newer DSS/Diffie-Hellman technology.

- If you plan to correspond with individuals who are still using the older RSA keys, you will probably want to generate an RSA key pair that is compatible with older versions of the program.

- If you plan to correspond with individuals who have the latest version of PGP, you can take advantage of the new technology and generate a pair of DSS/Diffie-Hellman keys.
- If you want to be able to exchange e-mail with all PGP users, you should make a pair of RSA keys and a pair of DSS/Diffie-Hellman keys and then use the appropriate pair depending on the version of PGP used by the recipient with whom you are communicating.

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

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



8. Select a key size (from 768 to 3072) or enter any key size from (from 512 to 4096). Note that RSA keys are limited to 2048 bits in order to maintain compatibility with older versions of PGP.

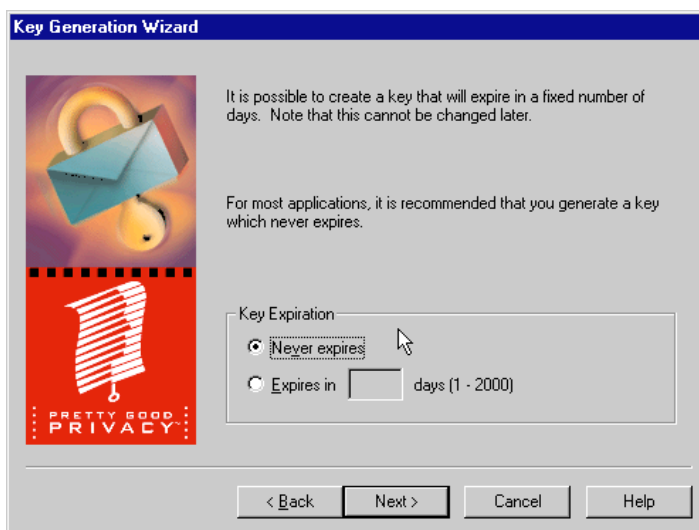
The key size corresponds to the number of bits used to construct your digital key. The larger the key, the less chance that someone will ever be able to crack it, but the longer it will take to perform the decryption and encryption process. You will need to strike a balance between the convenience of 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 DSS/Diffie-Hellman keys, the size of the DSS portion of the key is increased in fixed increments and is less than the size of the Diffie-Hellman portion of the key and is limited to a maximum size of 1024 bits.

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

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



10. Indicate when you want your keys to expire. You can either go with the default selection which is “never”, or you can enter a specific number of days 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 set 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.

11. Click **Next** to advance to the next dialog box.

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



12. In the “Passphrase” entry 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.

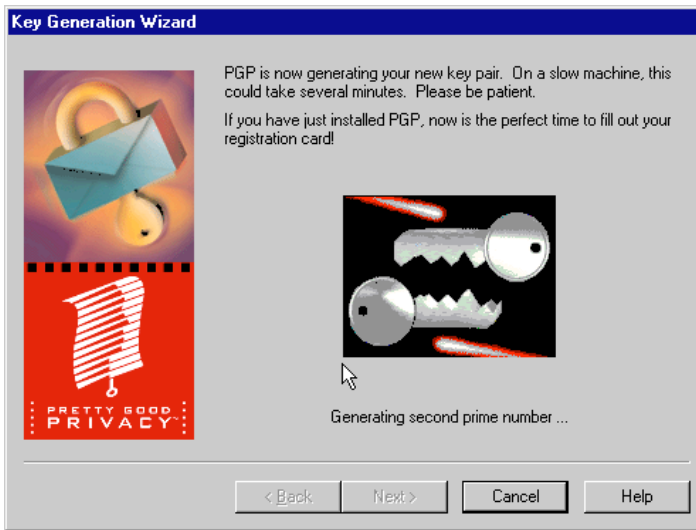
Normally, as an added level of security, the characters you enter for the passphrase do not appear on the screen. However, if you are sure that no one is watching over your shoulder, and you would like to see the characters of your passphrase as you type, clear the “Hide Typing” check box.

TIP:

Your passphrase should contain multiple words and may include spaces, numbers, and other printable characters. Choose something that you can remember easily but that others won't be able to guess, and keep in mind that the passphrase is case sensitive. The longer your passphrase, and the wider 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.

13. Click **Next** to begin the key generation process.

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



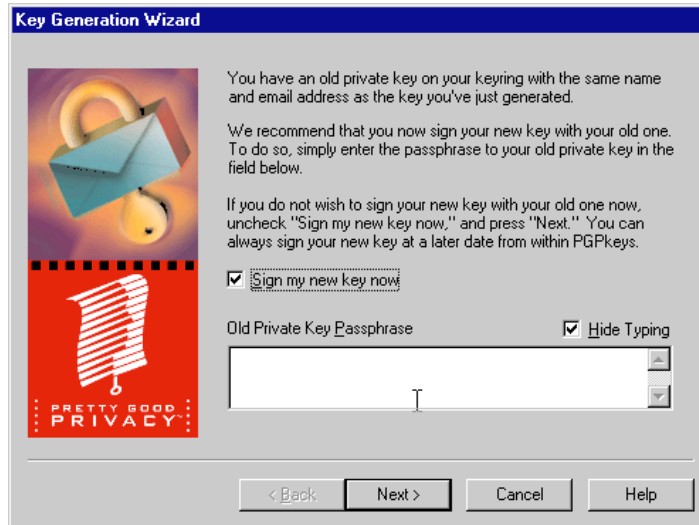
If you have entered an inadequate passphrase, a warning message appears before the keys are generated and you have the choice of accepting the bad passphrase or entering a more secure one before continuing.

If there is not enough random information upon which to build the key, the PGP Random Data dialog box appears. As instructed on the screen, move your mouse around and enter a series of random keystrokes until the progress bar in the dialog box is completely filled in. Your mouse movements and keystrokes generate random information that is needed to create a unique 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 DSS/Diffie-Hellman key, the fast key generation feature is not used and it could take hours to generate your key. Eventually the Key Generation Wizard indicates that the key generation process has completed.

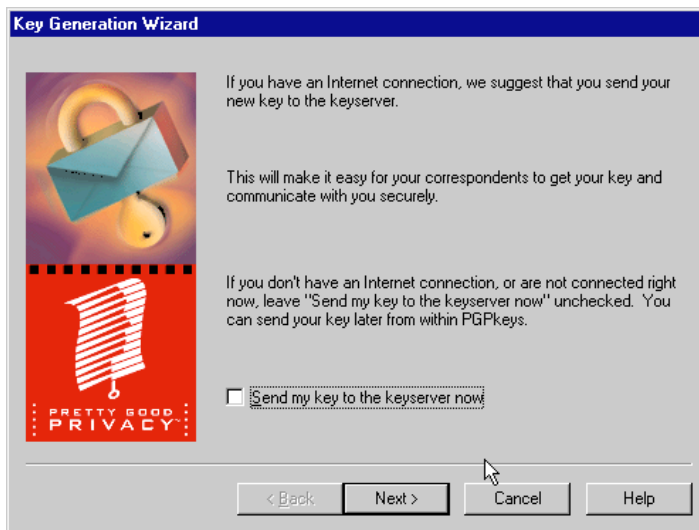
14. Click **Next** to advance to the next dialog box.

If you are creating a key with the same user name or e-mail address as a previous key, you are given the opportunity to sign the new key with your older key. This will endow the new key with same level of validity and trust as your older key when someone adds it to their keyring. The validity is based on those who have signed the key in the past but it does not include the signatures from your old key.



15. If applicable, sign your new key with the older key and enter the passphrase for the old key, then click **Next**.

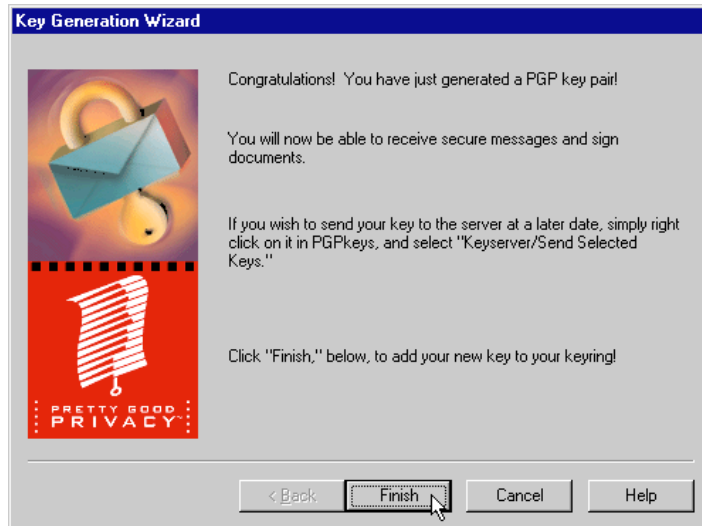
The Key Generation Wizard indicates that you have successfully generated a new key pair and asks you if you want to send your public key to the public key server.



16. Specify whether you want your new public key to be sent to the key server and then click **Next**.

By sending your public key to the key server, anyone will be able to get a copy of your key when they need it. For complete details, see the section, “Distributing Your Public Key,” later in this chapter.

When the Key Generation process completes, the final dialog box appears.



17. 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 DSS/Diffie-Hellman 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 user names or e-mail addresses. See Chapter 5 for complete details on how to add a new user name.

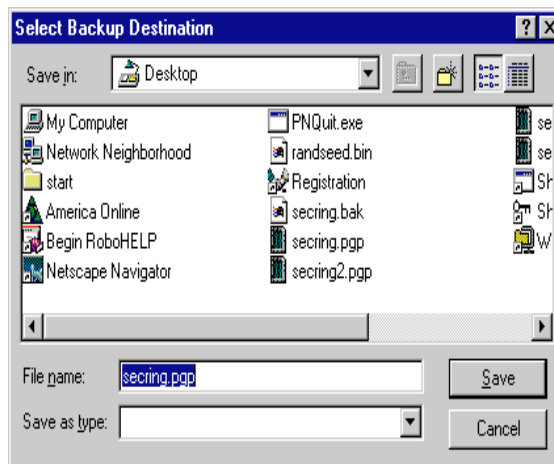
Protecting Your Keys

Once you have generated a key pair, it is wise to create a spare set and put them in a safe place in case something happens to the originals. In fact, when you close the PGPkeys window after creating a new key pair, you are prompted to save a backup copy.



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 (`secring.pgp`) and the public keyring (`pubring.pgp`) are stored along with the other program files in the PGP file directory, 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 “Select Backup Destination” dialog box appears asking you to specify the location of the private and public keyring files that are to be backed up.



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 today’s 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 file directory 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 key ring 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 several 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

Since your public key is basically composed of a block of text, it is really 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

Probably the best long-term and hassle-free method for making your public key available is to place it on a public key server where anyone can access it. By storing your public key on a key server, 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, such as those offered by PGP, Inc. where you can make your public key available for anyone to access. It doesn't really matter which key server you use to initially submit your public key, because once you submit your key to one server it is automatically propagated to all the other major servers in the world.

Each site provides a slightly different interface for submitting a public key, but the procedure basically requires you to copy the text content of your key and then paste it into the proper place on the key server. However, when using PGP, you can post your public key to a public key server automatically whenever you create a new key or at any time thereafter from within the PGP keys window.

To send your public key to a key server

1. Open the PGPkeys window by clicking on the lock and key icon in the Win95 tray, or click the **Start** button and choose **PGPkeys** from the **PGP** submenu of the **Programs** menu.
2. Select the icon that represents the public key you want to post on the key server.
3. Choose **Send Selected Keys** from the **Keyserver** submenu of the **Keys** menu. As an alternative, you can click the right mouse button and select this option from the pop-up menu.

After placing a copy of your public key on a key server, you can tell those who want to send you encrypted mail or verify your digital signature to get a copy of your key from the server. Even if you don't explicitly point someone 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 in the footer of their e-mail messages; in many 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 you acquire new signatures, all you have to do to replace your old key is send a new copy to the server and 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 be updated to reflect user names or signatures which have been removed from your key. If your key is ever compromised, you can revoke your key which tells the world to no longer trust that version of your key. See Chapter 5 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 your e-mail message.

To include your public key in an e-mail message

1. Open the PGPkeys window by clicking on the lock and key icon in the Win95 tray, or click the **Start** button and choose **PGPkeys** from the **PGP** submenu of the **Programs** menu.
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.

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 several ways to copy your public key to a file:

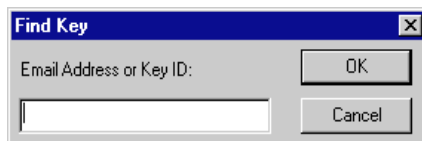
- Select the icon representing your key pair from the PGPkeys window, then choose **Export** from the **Keys** menu and enter the name of the file where you want the key to be saved.
- Drag the icon representing your key pair from the PGPkeys window and drop it into the desired location in the Windows Explorer window.
- 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.

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, since all key servers are regularly updated to include the keys stored on all the other servers. However, with this version of PGP, don't have to go through this old-fashioned method but can instead 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 window by clicking on the lock and key icon in the Win95 tray, or click the **Start** button and choose **PGPkeys** from the **PGP** submenu of the **Programs** menu.
2. Choose **Find New Key** from the **Keyserver** submenu of the **Keys** menu.

The "Find Key" dialog box appears.



3. Enter the e-mail address or user name to locate the users public key.
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 will show 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 them include it when they send you encrypted e-mail. If you have an e-mail applications that is supported by the PGP plug-in, then adding the senders public key to your public key ring can be accomplished by simply clicking a button. For example, when a 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 copy the block of text that represents the public key and paste it into the PGPkeys window and thus add the key to your public keyring.

Importing a Public Key from a File

Another method of obtaining someone's public key is to have them save it to a file from which you can import it or copy and paste it into your public keyring. There are several 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 from the Windows Explorer window onto the PGPkeys window.
- Open the text document where the public key is stored, select the block of text representing the key, then choose **Copy** from the **Edit** menu. Then, go to the PGPkeys window and choose **Paste** from the **Edit** menu to copy the key. The key will then show 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 which allow you to check a key's authenticity and to certify that the key belongs to a particular owner. The PGP program will also warn you when you attempt to use a key that is not valid and can be set to optionally warn you when you are about to use a marginally trusted key.

One of the major vulnerabilities of public key encryption systems is the ability of some eavesdropper 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 diskette and then physically hand it to you. Since you are not always within close enough proximity to personally hand a disk to someone, you will 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 against 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 over the phone. When you get a key from a public key server, you don't have to go through this exercise but can instead access the fingerprint information for the key while you are on-line. Of course, you do this with the expectation that the person periodically checks their key to make sure that no one has switched their key.

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 signifying to the world 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.

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. Now, when someone gets a copy of your public key, they don't necessarily have to check the key's authenticity themselves, but can instead rely on how well they trust the person who signed your key. PGP provides the means for establishing this level of trust for each of the

public keys you add to your public keyring and shows the level of trust 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 Chapter 5.

Sending and Receiving Private 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. In addition, if you are using an application which supports the PGP/MIME standard, you can encrypt and sign your e-mail messages as well as any file attachments when you send or receive your e-mail.

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 the Windows clipboard by selecting the appropriate option from the lock and key icon located in the System Tray. To include any file attachments, you encrypt the files from the Windows Explorer before attaching them.

Encrypting and Signing with Supported E-mail Applications

When you are encrypting and signing 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. The drawback with this method is that you will have to separately encrypt any file attachments you want to send with the e-mail.

NOTE: If you do not send your e-mail immediately but instead temporarily store it in your outbox, you should be aware that when using some e-mail applications, the information will not be 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 the message via the clipboard before queuing it in the outbox.

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.



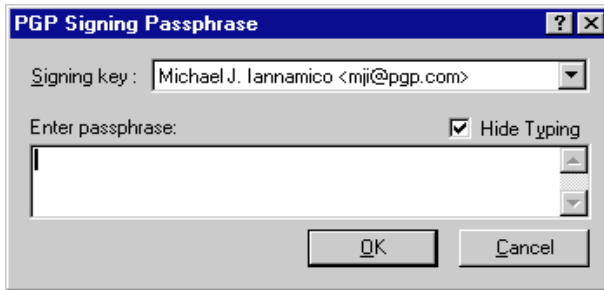
If you are communicating with another PGP user who is using an e-mail application that adheres to the PGP/MIME standard, you will want to click on the PGP/MIME button.

When you click one of these buttons, they remain indented to indicate the operations you want to perform.

NOTE: If you know that you are going to either encrypt and/or sign or you are going to use PGP/MIME on a regular basis, you can leave these operations turned on by selecting the appropriate settings from the e-mail pane of the Preferences dialog box.

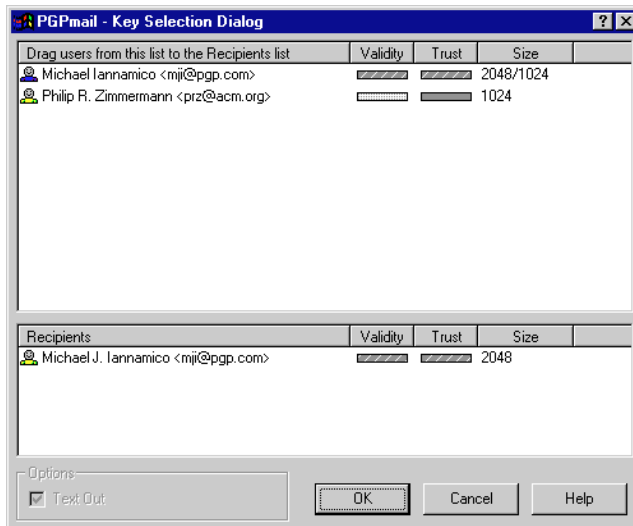
3. Send your e-mail message as you normally do.

If you have elected to sign the encrypted data, the Passphrase dialog box appears requesting your passphrase before the mail is sent.



4. Enter your passphrase and then click **OK**.

As long as 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 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 on any of the keys to move them from one area of the screen to the other.

The “Validity” bar 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 5 for more details).

NOTE: If you are not using PGP/MIME, you must encrypt any files you want to send as attachments from the Windows Explorer before sending them.

6. Click **OK** to send your mail.

Encrypting and Signing Via the Clipboard

If you are using an e-mail application that is not yet supported by the PGP plug-ins, you must encrypt and sign your e-mail via the Windows clipboard. You do this by clicking on the lock and key icon located in the System Tray and then choosing the appropriate option. Essentially, you copy the contents of your message to the clipboard, encrypt and/or sign its contents, then paste it into your e-mail editor before sending it. If you plan to attach any files with your message, you must encrypt them from the Windows Explorer before attaching them.

To encrypt and sign via the clipboard

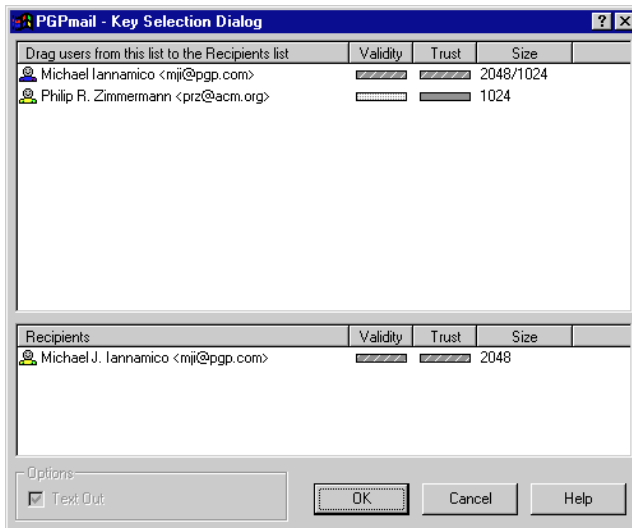
Here is the procedure for encrypting and signing an e-mail message using the clipboard:

1. Use the editor supplied with your e-mail application or your favorite word processing program to compose the message you want to send.
2. When you are ready to send the message, select the area of text you want to encrypt or choose **Select All** from the **Edit** menu available in most applications.
3. Choose **Copy** from the **Edit** menu to copy the contents of your message to the clipboard.

You should note that any time you copy or cut text in your application, it is temporarily stored on the clipboard.

- Click the lock and key icon in the System tray and choose either **Encrypt Clipboard**, **Sign Clipboard** or **Encrypt And Sign Clipboard** depending on the operations you want to perform.

If you indicate that you want to encrypt the contents of the clipboard, the PGP Key Selection Dialog box appears:

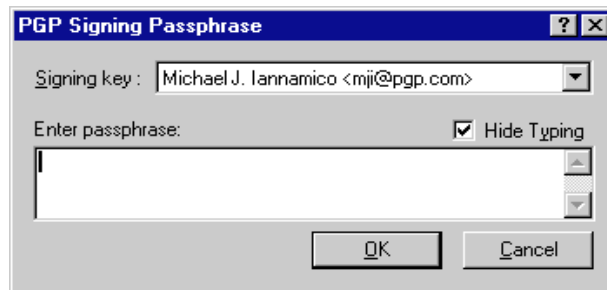


- Drag the public keys for those who are to receive a copy of the encrypted e-mail message into the “Recipients” list box.

The “Validity” bar 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 5 for more details.

- Click **OK**.

If you have elected to sign the message, the PGP Signing Passphrase dialog box appears requesting your personal passphrase for your default private key. If you have other key pairs and you want to use one of those instead, you can click on the arrow and select the appropriate key.



7. Enter your passphrase and then click **OK**.
8. Return to your e-mail application and choose the **Paste** command from the **Edit** menu. This will copy the encrypted message back into your e-mail application.
9. Send your e-mail to the intended recipient(s).

Encrypting and Signing from the Windows Explorer

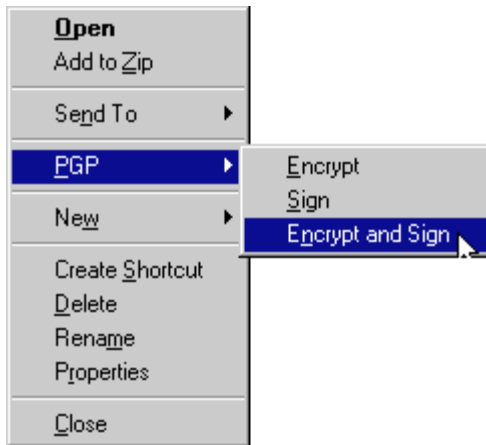
If you plan to send an encrypted file as an attachment with your e-mail message, or if you just want to encrypt a file to protect it on your own computer, you do so from the Windows File Explorer.

To encrypt and sign from the Windows Explorer

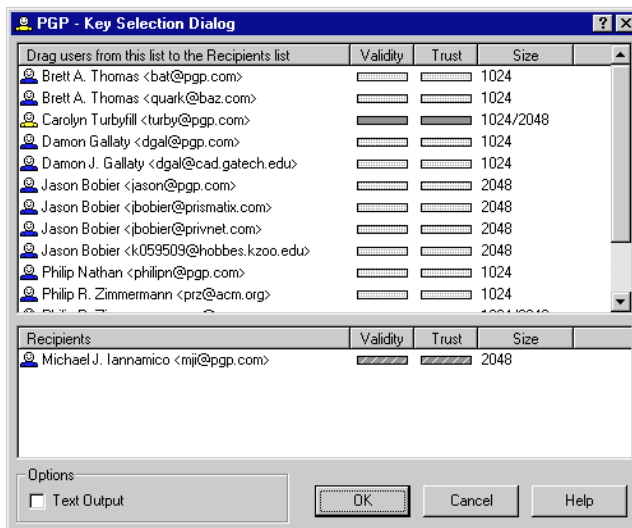
1. Open the Windows Explorer from the **Start** menu.
2. Select the file or files that you want to encrypt.

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

3. Choose the desired option from the **File** menu or from the pop-up menu, which you access by pressing the right mouse button.



The “Key Selection Dialog” box appears where you can select the recipient’s public keys for the file you are encrypting or signing.



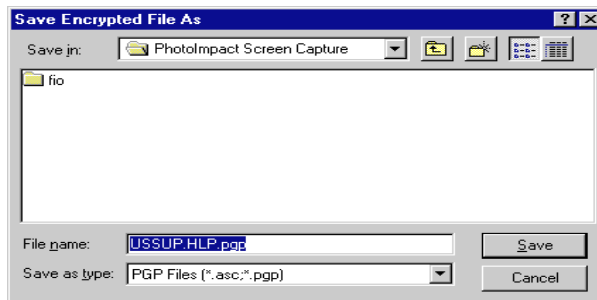
When sending files as attachments with some e-mail applications, you may need to check the “Text Output” check box to have the file saved as ASCII text. This is sometimes necessary in order to send a binary file using older e-mail applications.

If you have elected to sign the files, you will be asked to supply your passphrase.

If you are adding your signature to the encrypted file and would like the signature stored in a separate file, select the “Separate Signature File” check box.

- If you want the output of the encrypted file saved in a text format that can be handled by all e-mail applications, select the **Text Out** checkbox. You should note that selecting this option increases the size of the file by about 30 percent.
4. Select the public keys by dragging them to the recipients list and then click **OK**.

The “Save Encrypted File As” dialog box appears:



5. Specify the location and enter the name of the file where you want to save the encrypted version of the file. The `.pgp` extension is automatically appended to the file name unless you have turned on the ASCII Armor option in which case the `.asc` extension is used.
6. Click **Save** to save the file to the specified location.

If you look in the directory where you saved the file, you will find a file with the specified name represented by one of two icons:



`.pgp`

encrypted with standard output



`.asc`

encrypted with text output

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 plugins, you can perform the decryption and verification process by clicking a button in your application's toolbar. 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 just clicking an icon attached to your e-mail.

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 the Windows clipboard. In addition, if your e-mail includes encrypted file attachments, you must decrypt them separately from the Windows Explorer.

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, an opened envelope icon will appear when you open your e-mail.

```
X-Sender: mji@mail.pgp.com (Unverified)
X-Mailer: QUALCOMM Windows Eudora Pro Version 3.0.2 b4 (32)
Date: Wed, 21 May 1997 10:02:32 -0700
To: mji@pgp.com
From: Mike Iannamico <mji@pgp.com>
Subject: test
```



Decrypt PGP/MIME Message

In this 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 open envelope icon in your application's toolbar. Also, if there are any encrypted file attachments, you will decrypt them from the Windows Explorer.

To decrypt and verify with supported e-mail applications

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

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

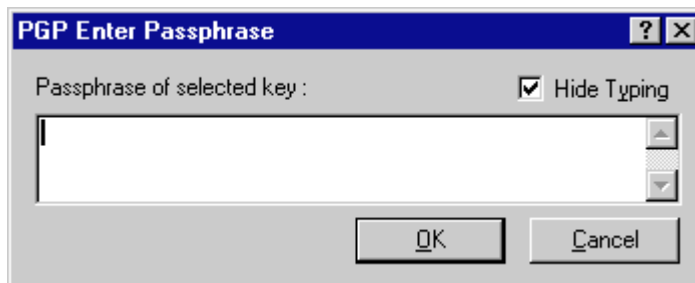
```
-----BEGIN PGP MESSAGE-----
Version: 5.0 alpha
MessageID: 112sx9z2duBj0iEXMCqGGfnxLWHzpn6v

hQEMAxr0Jt4E11lhAQf9G+K9InRx0GXyIocNls0U8dzsEb0h5x6DpmRDK4qsAME+
tth5bU/TydGwCoHR74SX4GaH5VAyxZaryEB/gt/1alahe/nrhi3rU+MvijUPaAKi
FS4Gq/Fh091IYe+RapRGz1cjBnzhKihTbkjrdmoRxeRHnL12VF9Am4GtRHdKctT1
Akq9z5TcxZS9yE/r9qnzGpHNuUVtLVQBx3/WESQvRPh7006Gk8aKKAZGIDaiCXZu
6Wu3a9v8sR58QV162z6X1VyrkLuS3ddrbJS82rneR4hMxjwCeevJdDMuNHh6vNM
lHb9WzU6S1+dmmhPzfTka4j1aFuBYvwX7tEUWoAWFKUBUggVwY36ichJq0eFa8qL8
DbEnVeMV2rXY5Gdzsyg0S/Mc0g0qIMgToyqQ0wZUw+g00GBu5Vj0CGR24mSyyS6f
0UvgyeEA4XfdkyXPOnUJ9B96/9BKw/DABtSR3rQADV4vC0jE10U0D6qL1RFxVORA
Dm8BChulE97gjWfsMFHcEnjqB1UVQRNEiuSRGxEoF0STbHPR3nvPER6A9DYK5Dc
nQmhKb4ujmqFAlc5LJJe1CzEBdk4Z4fU4Jec+uEs5G+LozGWzIbTXluViFXBTI+e
6u0AaIEe48JM6fv60zgFJZ1YZ4I2Zfd1WA6cWx9GUmq0tCMBX/np34oAL1FZWpe2t
PfeX3PXNLOSXUSwTetVAvs/ObQ1ZRPYJ1MPKlkPejppqbbGgALXDx2CvZPCdvk00
LkHeneT2Q3KJuswiNKiG10KxcilTehYIpxvFX1bfm3LAWCyU9F9PAbH86E=
=0AJ3
-----END PGP MESSAGE-----
```

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



The PGP Enter Passphrase dialog box appears requesting that you enter your passphrase.



3. Enter your passphrase and then click **OK**.

The message is decrypted and, if the message is signed, a panel appears indicating whether it 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.

Decrypting and Verifying Via the Clipboard

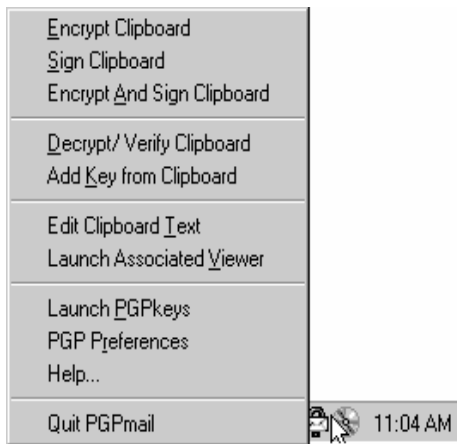
If your e-mail application is not supported by the PGP plug-ins, you must copy the contents of your message to the clipboard in order to decrypt it or to verify any digital signatures. If the e-mail contains file attachments, you decrypt and verify them through the Windows Explorer.

To decrypt and verify using the clipboard

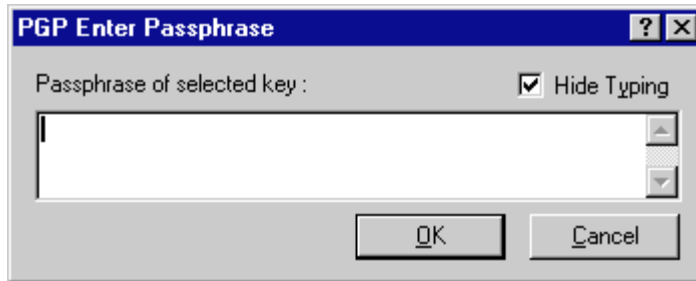
1. In the editor supplied with your e-mail application, select the encrypted text and then copy it to the clipboard.

In most applications, choose **C**opy from the **E**dit menu to copy the text to the Windows clipboard.

2. Click the lock and key icon in the System Tray to open the PGP pop-up menu. Choose **D**ecrypt/**V**erify Clipboard to initiate the decryption and verification process.



The PGP Enter Passphrase dialog box appears requesting that you enter your passphrase.



3. Enter your passphrase and then click **OK**.

The message is decrypted. If there are any signatures, an attempt is made to verify the signature and a message appears indicating whether the signature is valid.

4. To view the contents of the deciphered e-mail message, choose **Edit Clipboard Text** or **Launch Associated Viewer** from the PGP pop-up menu. You can then copy the contents back to your text editor and save it if you like.

Decrypting and Verifying from the Windows Explorer

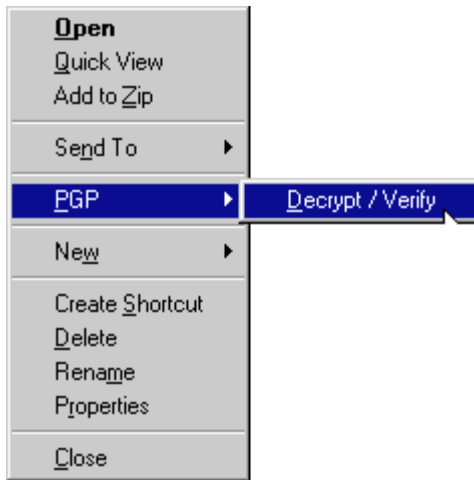
If the e-mail you receive has file attachments, and you are not using a PGP/MIME compliant e-mail application, you must decrypt them from the Windows Explorer.

To decrypt and verify from the Windows Explorer

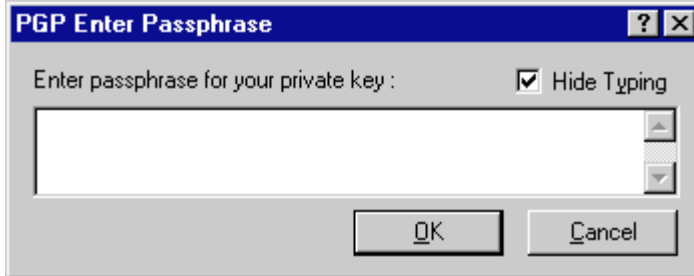
1. Open the Windows Explorer from the **Start** menu.
2. Select the file or files that you want to decrypt and verify.

You can select multiple files, but you must go through the process of decrypting and verifying each individual file.

3. Choose **Decrypt/Verify** from the **PGP** submenu of the **File** menu or press the right mouse button to open the pop-up menu and then choose **Decrypt/Verify**.

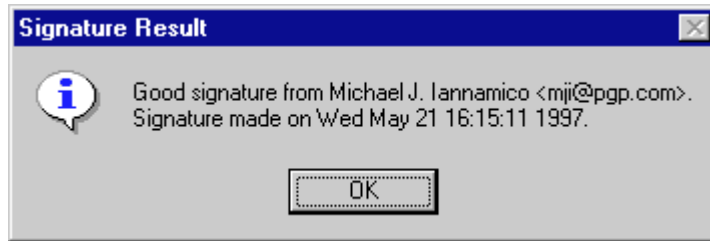


The passphrase dialog box appears requesting that you enter your passphrase.



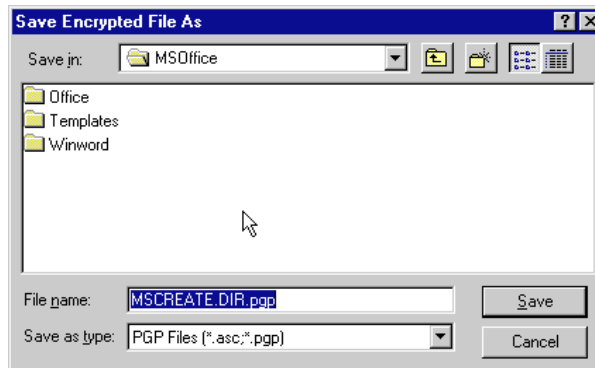
4. Enter your passphrase and then click **OK**.

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



5. Click **OK**.

The "Save Encrypted File As" dialog box appears.



6. Specify the location and enter the name of the file where you want to save the decrypted version of the file.

If you do not explicitly enter a name, the original name is used.

7. Click the **Save** button to save the file.

The decrypted file is saved in the specified location. If there are any signatures, an attempt is made to verify the signature and a message appears indicating whether the signature is valid.

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 `secring.skr` and your public keys are stored in another file named `pubring.pkr`. These files are usually located in the same program directory as the other PGP program files. The following icons are used to represent your private and public keyring files, making them easy to distinguish when you are browsing through your files.



Private Keyring



Public Keyring

NOTE: In the event you have more than one key pair, or if you are not comfortable storing your keys in the usual place, you can choose a different file name or location. See *Setting Your Preferences*.

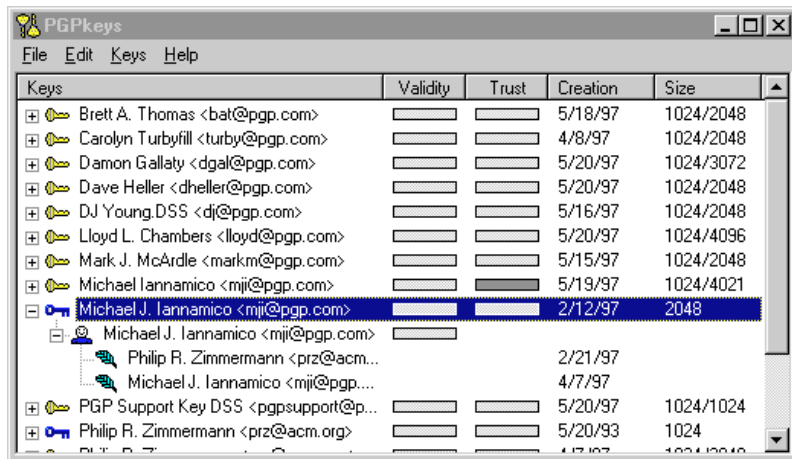
On occasion you may want to examine or change the attributes associated with your keys. For instance, when you obtain someone's public key, you might want to identify its type (either RSA or DSS/Diffie-Hellman), check its fingerprint, or determine its validity based on any digital signatures included 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 perform all of these key management functions from the PGPkeys window.

The PGPkeys Window

To open the PGPkeys window, click the lock and key icon in the System tray then choose **Launch PGPkeys**.



In the PGPkeys window you see 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 and DSS/Diffie-Hellman keys are yellow.

By double-clicking on any of the keys, you can expand the entries to reveal the user ID and e-mail addresses for the owner of the key as represented by the figure icons. By double-clicking a figure icon, you can see the signatures of any users who have certified the key, as represented

by the quill icon. If you don't want to double-click down through the various levels of information for each key, simply select the keys of interest and then choose **Expand Selection** from the **Edit** menu.

PGPkeys Attribute Definitions






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

| | |
|-----------------|---|
| Keys | Shows an iconic representation of the key along with the user name and e-mail address of the owner. |
| 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 will 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. |
| 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 elect to rely on the judgement of other users who have signed the key. When you create a set of keys, they are considered implicitly trustworthy, as represented by the striping in the trust and validity bars. 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 "Properties" dialog box. |

| | |
|-----------------|---|
| 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. |
| 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 DSS/Diffie-Hellman key, there is one number for the DSS portion and another number for the Diffie-Hellman portion. |

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.

| ICONS | WHAT THEY REPRESENT |
|---|---|
|  | A pair of gold keys represents your DSS/Diffie-Hellman key pair. The key pair consists of your private key and your public key. |
|  | A single gold key represents a DSS/Diffie-Hellman public key. |
|  | A pair of blue keys represents your RSA key pair. The key pair consists of your private key and your public key. |
|  | A single blue key represents an RSA public key. |
|  | When a key or key pair is grayed-out, they are temporarily unavailable for decrypting and signing. You can disable a key from the PGPkeys window which prevents seldom used keys from cluttering up the Key Selection dialog box. |

ICONS

WHAT THEY REPRESENT



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 indicates a key that is invalid.



A key with a clock indicates that the key has expired. A key's expiration date is established when the key is created.



A smiley face represents the owner of the key and lists the user names and e-mail addresses associated with the key.



A quill indicates the signatures from those 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.



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.



A striped bar indicates an implicitly valid key and implicitly trusted key. This setting is only available for the private and public key pairs you create.

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 **Key Properties** 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. This is either RSA or DSS/Diffie-Hellman.
- Expires** The date when the key expires. The owner specifies 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 only wants them to be used for a limited period of time.
- 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 appropri-

ate level (Complete, Marginal or Untrusted). The bar is not shown for revoked, expired and implicitly trusted keys.

Fingerprint A unique identification number that is generated when the key is created and 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 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 public key server since it is assumed that the owner periodically checks to make sure that it remains valid.

Enabled Indicates whether the key is currently enabled or not. When a key is disabled, it is dimmed in the PGPkeys window and is not available for performing any PGP functions. However, the key remains on your keyring and you can enable it again if it becomes necessary. To enable or disable a key, select or clear the Enabled check box, or choose **Enable** or **Disable** from the **Keys** menu. This feature is useful for preventing keys that you don't use on a regular basis from cluttering up the recipients dialog when you are sending encrypted e-mail.

Change Passphrase

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

Specifying a Default Key Pair

When you sign a message or someone's public key, your default key set is used. If you have more than one set of keys, you may want to specifically designate one pair as your default set. The current default key set is displayed in bold text 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 As Default Key** from the **Keys** menu.

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

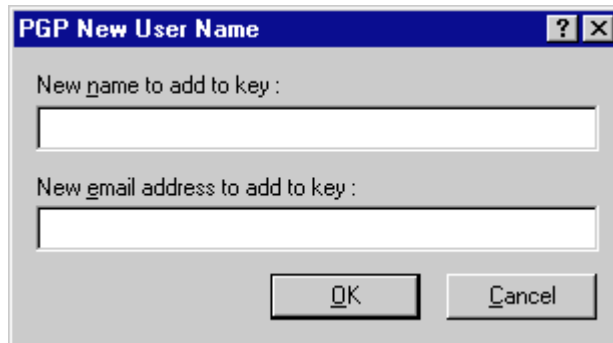
Adding a New User Name or Address

In some cases you may have more than one user name or e-mail address for which you want to use the same set of keys. After initially 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 when you have both the private and public keys.

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

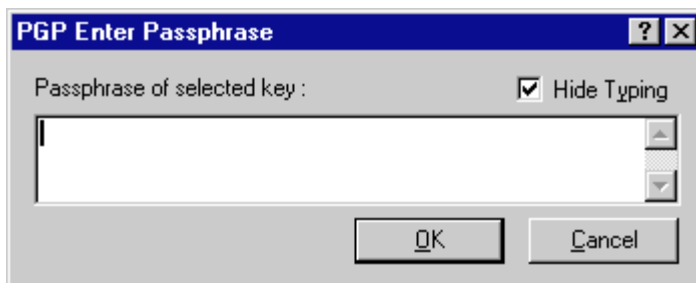
1. Select the key pair for which you want to add another user name or address.
2. Choose **Add Name** from the **Keys** menu.

The PGP New User Name dialog box appears.



3. Enter the new name, then press Tab to move to the next field.
4. Enter the new e-mail address.
5. Click **OK** after you have entered the new name and address.

The PGP Enter Passphrase dialog box appears requesting that you enter your passphrase.



6. 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, then select the name and address and click on the **Set As Primary User ID** command.

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 their key to you on a floppy disk. Since exchanging keys in this manner is not usually practical, especially for users who are located many miles apart, 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 make a call to the person and have them read the fingerprint to you over the phone. It is highly unlikely that someone will be able to intercept this random call and imitate the person 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 **Key Properties** from the **Keys** menu.
3. Note the fingerprint and compare it to the original.

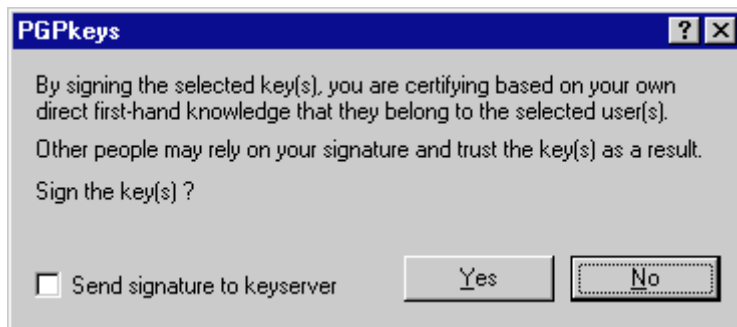
Signing Someone's Public Key

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 their 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.

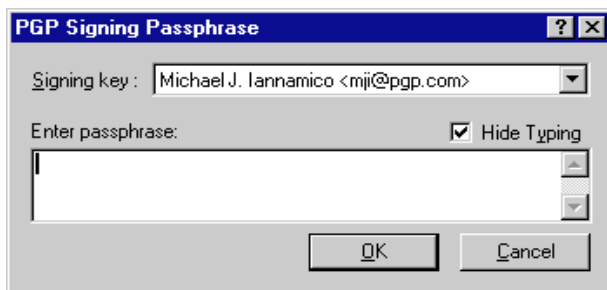
The PGPkeys alert box appears.



3. Click **Yes** to indicate your certainty that the key does indeed belong to the purported owner.

If you would like to send the key with your signature to the key server check the "Send signature to keyserver" checkbox. The public key on the server will subsequently be updated to reflect the inclusion of your signature. Since most users prefer to use their own discretion when allowing others to sign their keys, it is always a good idea to check with the owner before you add your signature to their key on the server.

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



4. 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.
5. Once you have signed someone's public key, a quill icon associated with your user name is shown for that key.

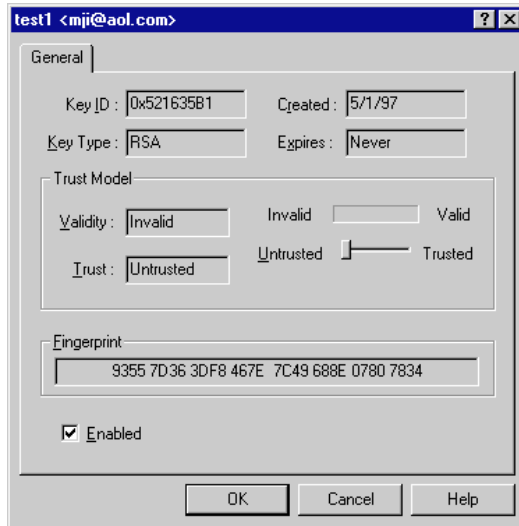
Granting Trust for Key Validations

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 that 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 **Key Properties** from the **Keys** menu.

The Properties dialog box appears.



3. Use the trust level sliding bar to choose the appropriate level of trust for the key. You have a choice of “Untrusted,” “Marginal” or “Complete.”
4. Click **OK** 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 **Disable** from the **Keys** menu.

The key is dimmed and is temporarily unavailable for use.

To enable a key

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

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 **Delete** from the **Edit** menu.

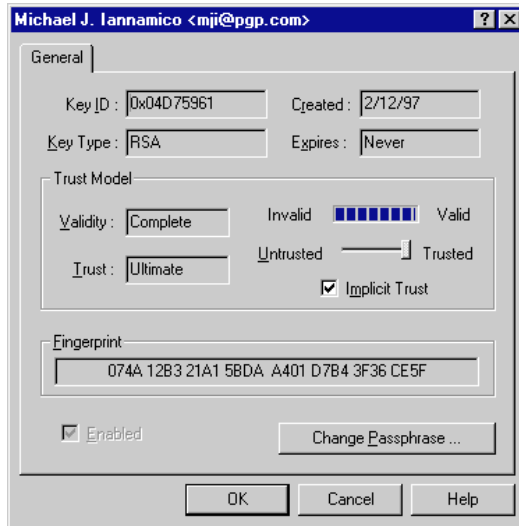
Changing your Passphrase

It is a good idea to periodically change your passphrase. If you want to change your passphrase, you can easily do so.

To change your passphrase

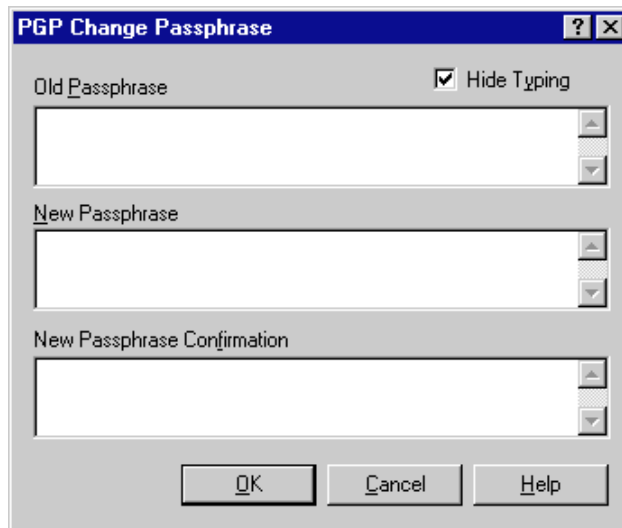
1. Select the key pair for which you want to change the passphrase.
2. Choose **Key Properties** from the **Keys** menu.

The Properties 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 field 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**.

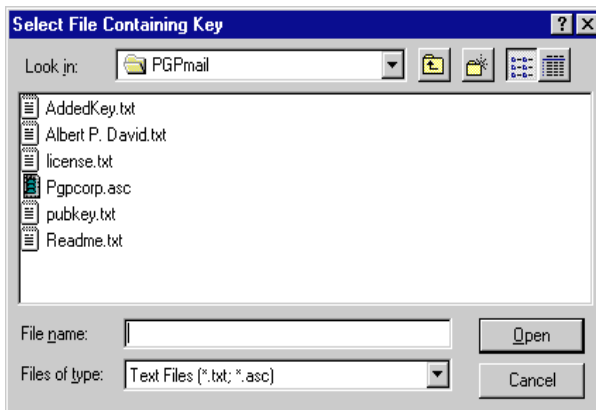
Importing and Exporting Keys

Although you often distribute your public key and obtain the public keys of others by cutting and pasting the raw text from a public 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 from a file.
2. Choose **Import** from the **Keys** menu.

The Select File Containing Key dialog box appears.



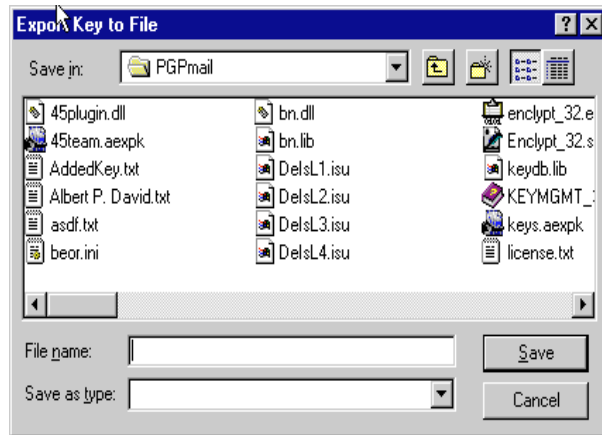
3. Select the file that contains the key you want to import, and then click **Open**.

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

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 Key to File dialog box appears.



3. Enter the name of the file where you want the key to be exported, and then click **Save**.

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

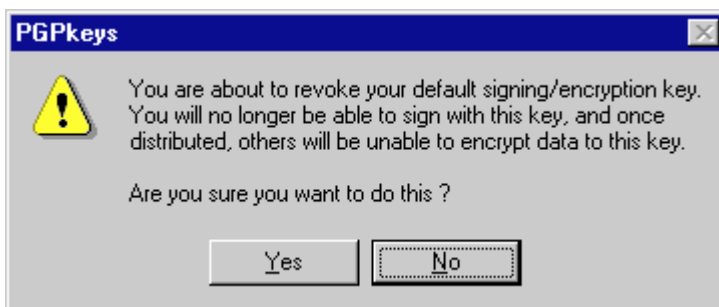
Revoking a Key

If the situation ever arises that you can 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

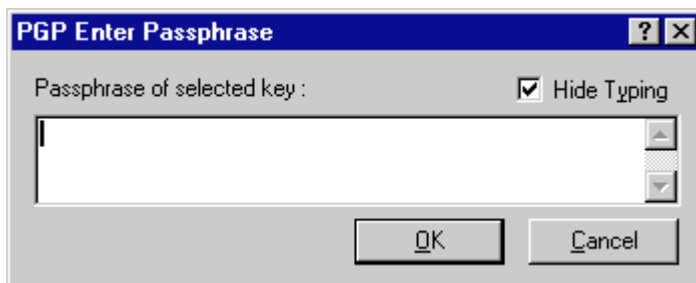
1. Select the key pair to revoke.
2. Choose **Revoke** from the **Keys** menu.

A message appears with some brief information on the implications of revoking a key and asking you to specify whether you really want to revoke the selected key.



3. Click **Yes** to confirm your intent to revoke the selected key.

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



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 it 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 and replace it with one of their own making.

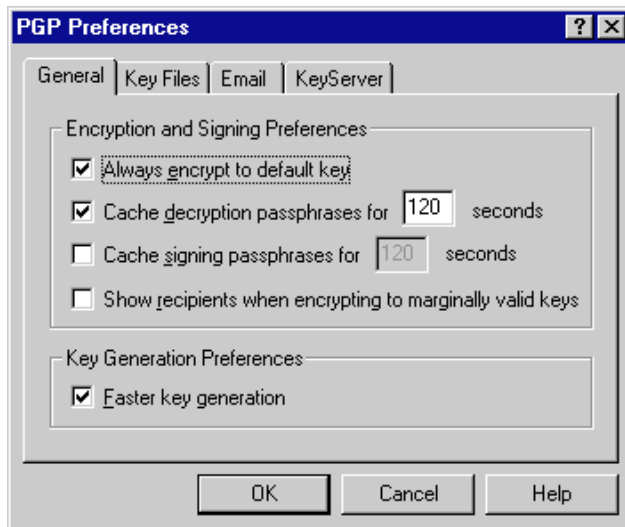
Setting Your Preferences

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 using one of the following methods:

- Click the lock and key icon and choose **Preferences**.
- Choose **Preferences** from the **Edit** menu in the PGPkeys window.

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 or 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 for [] Seconds

This setting specifies the amount of time (in 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 your passphrase over and over again to get through all of your mail. However, you should be aware that the longer your passphrase is stored in your computers memory, the more time a sophisticated snooper has to get hold of this highly compromising bit of information. By default, this setting is set to 120 seconds, 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 someone to determine your passphrase.

Cache Signing Passphrase for [] Seconds

This setting specifies the amount of time (in seconds) that your signature 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 your passphrase over and over again to get through all of your mail.

Show Recipients When Encrypting To Marginally Valid Keys

This setting specifies that you would like to be warned whenever you are encrypting to a recipient for which the validity is only marginally established.

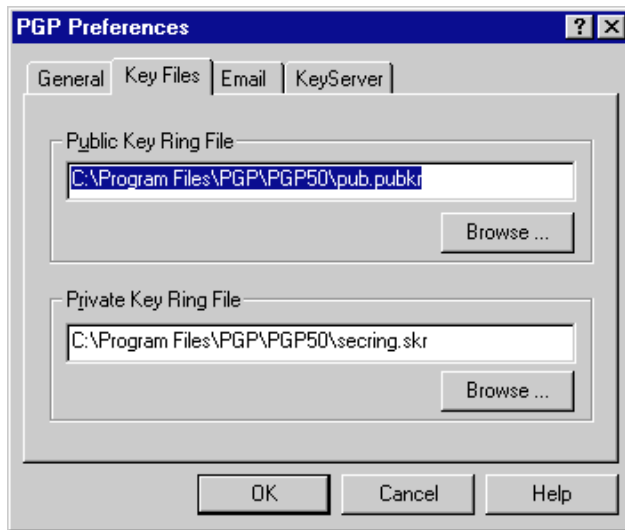
Faster Key Generation

When this setting is selected, it requires less time to generate a new DSS/Diffie-Hellman 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, you should note that fast key generation is only implemented for the fixed key sizes provided as options when you create a key and will not be utilized if you enter some other value. Although it would be just about impossible for anyone to ever 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.

Key Files Preferences

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



Public Key Ring File

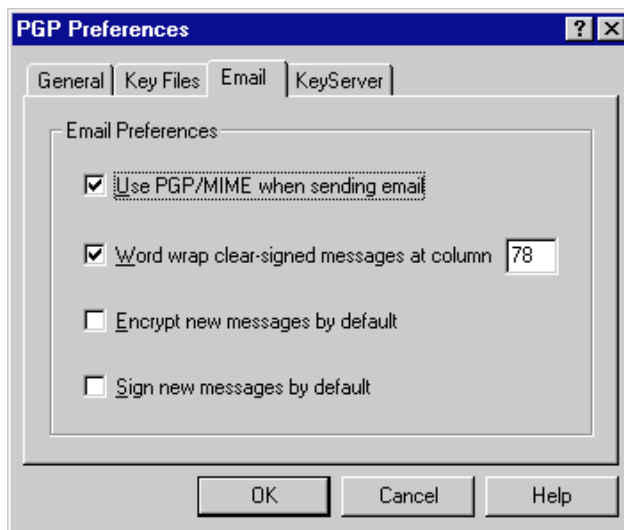
Shows the current location and name of the file where the PGP program expects to find your public keyring file. If you plan to store your public keys in a file with a different name or in some other location, you specify this information here. You can use the Browse button to search through your files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

Private Key Ring File

Shows the current location and name of the file where the PGP program expects to find your private keyring file. If you plan on storing 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.

E-mail Preferences

Click the **e-mail** tab to advance to the pane where you specify preferences that affect the way PGP functions are implemented for those e-mail applications that are supported by the PGP plug-ins. You should note that the PGP/MIME option is not applicable for all e-mail applications.



Use PGP/MIME When Sending e-mail

When this check box is selected, you do not have to go through the trouble of explicitly turning on the PGP/MIME feature every time you send e-mail. For instance, if you are using Eudora, and you turn this setting on, all of your e-mail messages and file attachments are automatically encrypted and signed to the intended recipient. This setting has no effect on other encryptions you

perform from the clipboard or with the Windows Explorer, 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. Although these users can decrypt and verify this type of message, they will not be able to verify clear-signed messages.

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 all applications do not handle word wrapping in the same way, which could cause the lines in you digitally signed messages to be broken up in a way that cannot be read properly. By default, this setting is set to 78 which prevents problems with most applications.

Encrypt New Messages by Default

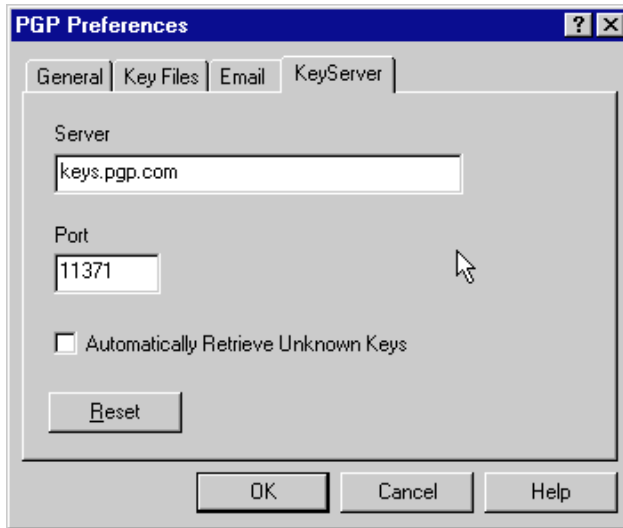
Leaves the Encryption function turned on for your e-mail application. The lock icon will remain indented to indicate that the encryption function is turned on.

Sign New Messages by Default

Leaves the Signature function turned on for your e-mail application. The quill icon will remain indented to indicate that the signatory function is turned on.

Key Server Preferences

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



Server Specifies the address for the public key server that is used by PGP to send and retrieve public keys. You should not change this unless you have a different location.

Port The port address for the public key server.

Automatically Retrieve Unknown Keys

When this setting is selected, any unknown recipients will be retrieved from the public key server when you encrypt or verify your e-mail providing that the keys have been stored on the server.

Reset Reverts to the default server name and port number settings for the key server.

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 US 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. This was only done in important cases when it seemed worthwhile.

Senate Bill 266, a 1991 omnibus anti-crime 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 can 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 rigorous protest from 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 one percent of all phone calls in all major US 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 1000 court-ordered wiretaps in the US 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% 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% sample, the wiretaps can be shifted over to a different 1% 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 US companies to help French corporations get a competitive edge. Ironically, US government restrictions on cryptography have weakened US 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 was under development at 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, like secure phones, secure FAX, etc. AT&T has put Clipper into their 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 anti-war 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 is still legal. When use of strong cryptography becomes popular, it’s harder for the government to criminalize it. Thus, using PGP is good for preserving democracy.

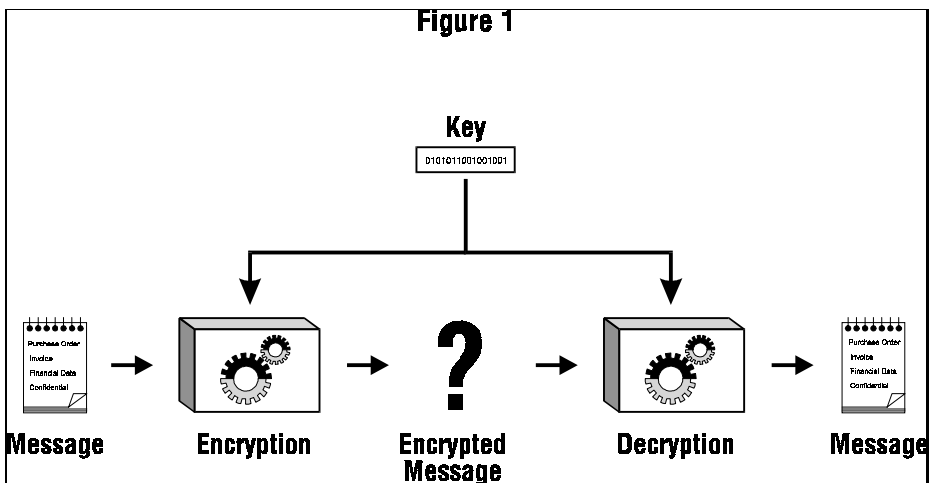
If privacy is outlawed, only outlaws will have privacy. Intelligence agencies have access to good cryptographic technology. So do the big arms and drug traffickers. But ordinary people and grassroots political organizations mostly have not had access to affordable “military grade” public-key cryptographic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

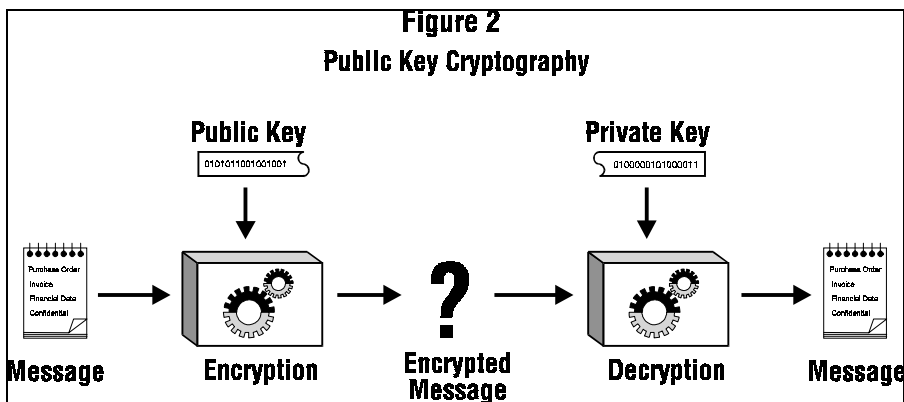


How Public Key Cryptography Works

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.

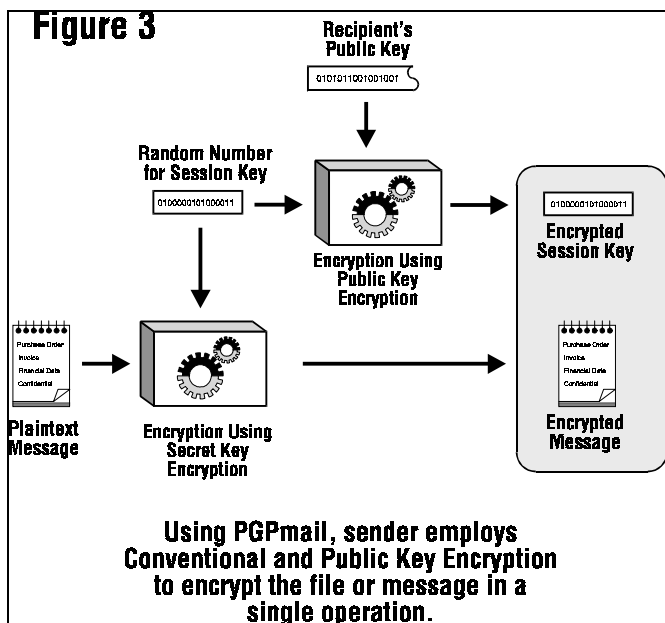
This protocol provides privacy without the need for the same kind of secure channels that conventional secret key encryption requires.

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.



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.



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. While CAST is too new to have developed a long track record, 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. Although triple-DES is much slower than either CAST or IDEA, speed is usually not critical for e-mail applications. While 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 will 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 incompressible. 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

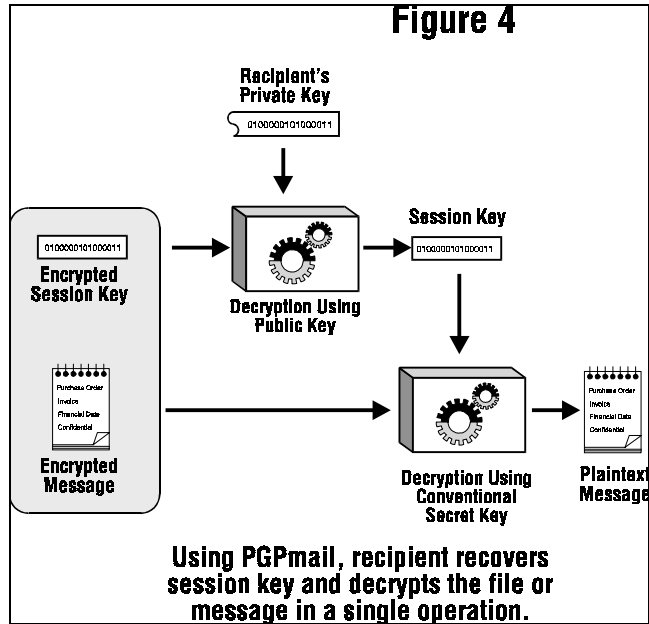
PGP uses a cryptographically strong pseudo-random number generator for creating temporary session keys. If this random seed file does not exist, it is automatically created and seeded with truly random numbers derived from your random events gathered by the PGP program from the timing of your keystroke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

How Decryption Works

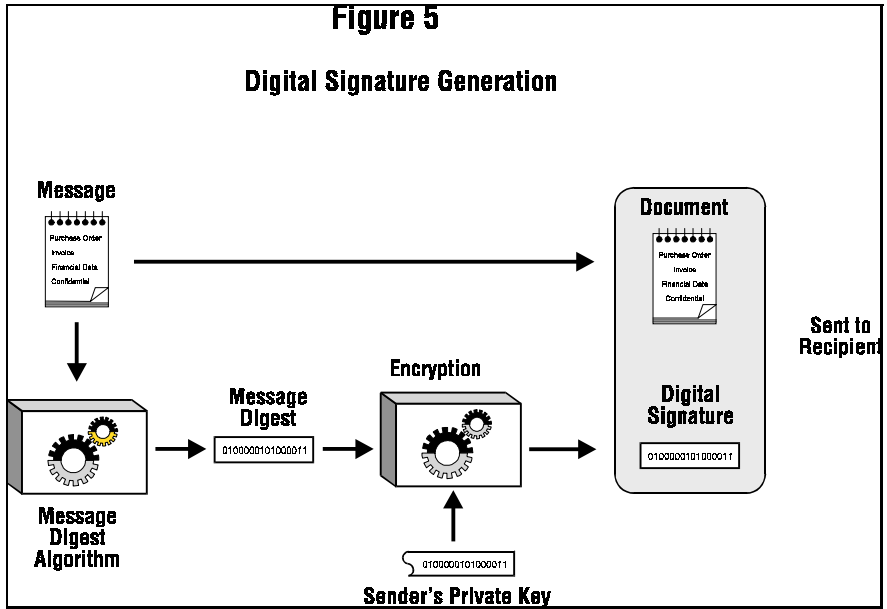
As shown in Figure 4, the decryption process is just the reverse of encryption. The recipient's private key is used to recover the temporary session key, and then that session key is used to run the fast conventional secret-key algorithm to decipher the large ciphertext message.



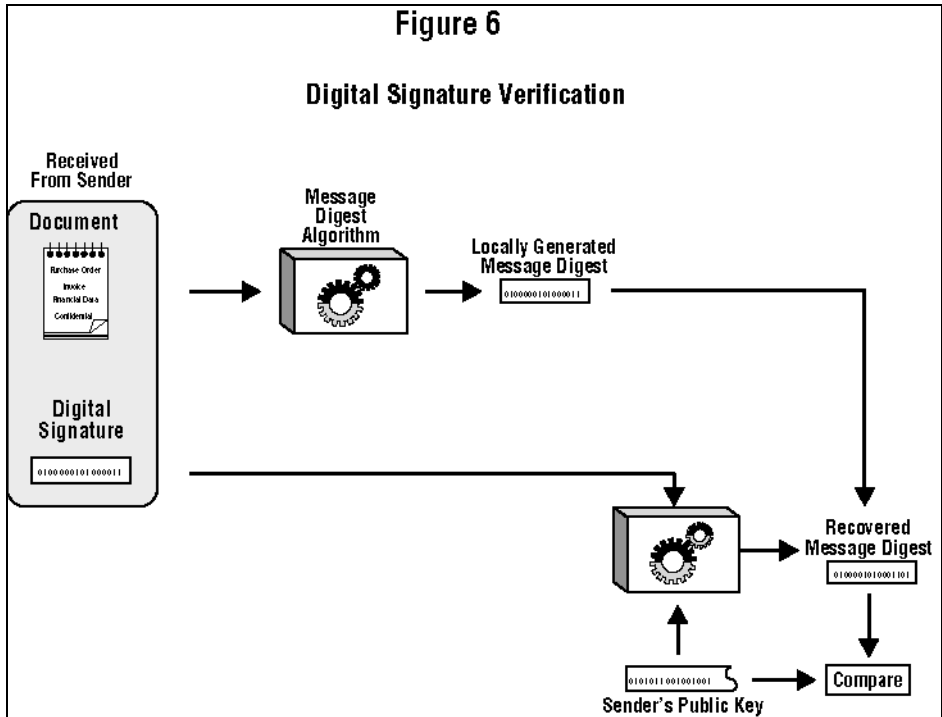
How Digital Signatures Work

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, such 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 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 Hans Dobbertin, a German cryptographer. While 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, thus 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 whom it appears to belong to. 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 re-encrypt 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 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 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 that 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

may 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 firsthand 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 hopes 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.

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.

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 timesharing 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 wish 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 on “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 may directly sign more keys yourself.

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

1. Does the key actually belong 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 you 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 will 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 e.g. two marginally trusted signatures are deemed as credible as one fully trusted signature. The program's skepticism is adjustable—for example, you may 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 introducers 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 you ultimately 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 or 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 as 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, they 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 your private key. 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 5 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 more well 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 non-institutional approach PGP supports for management of public keys has its benefits, but unfortunately this also means we 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 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 will install this key compromise certificate on their public keyrings and will automatically prevent 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 you signed your key to retire his/her certification. Then anyone attempting to use your key based upon 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 even 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. You might not otherwise have done so if you had no cryptographic software at all. Perhaps you may never even discover 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” 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) or 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 didn’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 all 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 may 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 \$2000-\$3000, 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 \$1200 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 (non-digital) 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 okay.

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 did make a lot of sense. I observed that practically no one in the commercial world of cryptography

qualified 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 US 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 US 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 US.

- 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 other principle method of suppression comes from the granting 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 US.

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, they 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.

And 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 US 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 protecting 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 is if you leave your 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 very 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 the section "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 eventually 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 for some reason 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 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 whomever 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 any of the disk utilities available that can overwrite all of the unused blocks on a disk. For example, the Norton Utilities for MS-DOS can do this.

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 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 falls under the category of defending against viral infection generally. There are some moderately capable anti-viral products commercially available, and there are hygienic procedures to follow that can greatly reduce the chances of viral infection. A complete treatment of anti-viral and anti-worm countermeasures is beyond the scope of this document. PGP has no defenses against viruses, and assumes 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.

Another similar attack involves someone creating a clever imitation of PGP that behaves like PGP in most respects, but 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 will not 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 or 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, or 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 may 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 for this, and so does the Mac OS. Turning off virtual memory means you might 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 grassroots 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, etc. 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 is known as “Tempest,” and 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 may write a date next to their handwritten signature on a contract with any date they choose, yet no one seems to be alarmed over 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 the same way 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 can 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 (see the Recommended Introductory Readings section, below). 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 convinces 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 multi-user 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 have been 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.

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., 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

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 `.asc` 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 are assigned the responsibility of certifying the origin of keys and adding them to a common database.

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.

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-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.

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.

Index

A

- address
 - adding new email 64
- attributes
 - changing your keyrings' 58–63
 - viewing your keyrings' 58–63

C

- certifying
 - authority 96
 - public keys 3, 96
- Certifying Authority 96
- Change passphrase property 63
- changing
 - passphrases 69
- checking
 - authenticity of a key 40
 - fingerprints 65
- checksum 94
- Clipboard
 - decrypting, via the 53
 - encrypting via the 46–48
 - using PGP from 13
 - verifying, via the 53
- comparing
 - fingerprints 41
- compatibility
 - PGP/MIME standard 8
 - versions of PGP 7
 - with DSS/Diffie-Hellman technology 8
- Created property 62

- creating
 - key pairs 22
 - private and public key pairs 14
- cryptography
 - overview 2

D

- decrypting
 - email 4
 - from others 51
 - overview 2
 - file attachments 54
 - from the Clipboard 13
 - from Windows Explorer 54, 54–56
 - via the Clipboard 53
- decryption
 - how it works 21
 - setting preferences 75
- defaults
 - specifying 63
- deleting
 - digital signatures 69
 - keys 69
- digital signature
 - and authenticity 41
 - deleting 69
 - overview 2
 - verifying 2
- digital signatures 92
- disabling
 - keys 68
- disclosure 101

- disk
 - system requirements 7
 - distributing
 - your public keys 35
 - DSS/Diffie-Hellman technology
 - keys 8
 - creating 25
- ## E
- email
 - adding a new user name 64
 - checking signature 2
 - copying public keys from 39
 - decrypting 4, 51
 - from Windows Explorer 54–56
 - via the Clipboard 53
 - within Eudora 51–52
 - encrypting 4, 43
 - from Windows Explorer 48–50
 - via the Clipboard 46–48
 - with Eudora 43–46
 - message
 - including your public key in 37
 - private
 - receiving 2, 43
 - sending 2, 43
 - setting preferences 77
 - signing 2, 4, 43
 - from Windows Explorer 48–50
 - via the Clipboard 46–48
 - with Eudora 43–46
 - verifying 4, 51
 - from Windows Explorer 54–56
 - via the Clipboard 53
 - within Eudora 51–52
 - Enabled property 63
 - enabling
 - keys 68
 - encrypting
 - email 4, 43
 - overview 2
 - from the Clipboard 13
 - from Windows Explorer 48–50
 - using Eudora 43–46
 - via the Clipboard 46–48
 - within Eudora 51–52
 - encryption
 - digital signature technology 8
 - DSS/Diffie-Hellman technology 8
 - how it works 21
 - setting preferences 74
 - exchanging
 - public keys 3
 - obtaining others' 38–40
 - expiration
 - setting for key pairs 27
 - Expire property 62
 - exporting
 - keys, to files 72
 - public keys, to files 37
- ## F
- file attachments 54
 - files
 - exporting keys to 72
 - exporting public keys to 37
 - importing keys from 71
 - importing public keys from 40
 - setting location of keyring files 76
 - fingerprint 94
 - Fingerprint property 63
 - fingerprints
 - checking 65
 - comparing 41
- ## G
- generating
 - key pairs 22
 - keys
 - setting preferences 75
 - granting
 - trust for key validations 67
- ## H
- hash function 94

I

- importing
 - keys, from files 71
 - public keys, from files 40
- installing
 - from disk 12
 - from the Web 12
 - PGP 12
- introducer 96, 98, 113

K

- Key ID property 62
- key management window 102
- key pairs
 - creating 3, 22, 23–32
 - creating with PGP Key Wizard 14
 - description of 22
 - examining 14
 - generating 22
 - making 22
 - setting expiration of 27
 - specifying defaults 63
 - viewing your 14
- key server
 - getting someone's public key from 39
 - sending your public key to 31, 35–36
 - setting preferences 79
 - using to circulate revoke keys 72
- key size
 - Diffie-Hellman portion 27
 - DSS portion 27
 - setting 26
 - trade-offs 26
- Key Type property 62
- keyrings
 - changing attributes of 58–63
 - description of 57
 - location of 57
 - overview of 2
 - setting location of 76
 - storing elsewhere 57
 - viewing attributes of 58–63
 - viewing properties of 62

keys

- backing up 33–34
- checking fingerprints 65
- colors of 32
- deleting 69
- disabling 68
- distributing 35
- enabling 68
- examining 14, 62
- exporting to files 72
- generating 22
- granting trust for validations 67
- importing from files 71
- managing 57
- overview of 21
- protecting 33–34
- revoked 73
- revoking 72
- saving 33–34
- setting location of 76
- setting size of 26
- signing 66
- types of
 - DSS/Diffie-Hellman 8, 22
 - RSA 8, 22
- verifying authenticity of 40
- viewing properties of 62

L

- legitimacy
 - determining a key's 40

M

- making
 - key pairs 22
- managing
 - keys 57
- memory
 - system requirements 7
- message digest 94
- MIME standard
 - using to decrypt email 51–52

using to encrypt email 43–46

N

new email address
adding 64

O

obtaining
others' public keys 38–40
opening
PGPkeys window 14
overviews
checking digital signature 2
cryptography 2
decrypting email 2
digital signature 2
encrypting email 2
key concepts 21
keyrings 2
private keys 2
public key cryptography 2
public keys 2
signing email 2
verifying digital signature 2

P

pass phrase 108
passphrase
Change Passphrase property 63
changing 69
forgotten 73
setting 28
setting preferences 75
suggestions for 28
PEM 100
PGP
compatibility 7
history 7
installing 12
overview of 2
running 12

upgrading from a previous version 8
upgrading from PGP, Inc. 8
upgrading from ViaCrypt 8
using from the Clipboard 13
using from the System tray 13
ways to use 12

PGP Key Wizard
creating key pairs 14
using to create key pairs 22
PGP/MIME standard
compatibility 8
using to decrypt email 51–52
using to encrypt email 43–46
PGPkeys window
creating key pairs with 23–32
Creation label 60
description 58
examining keys' properties
Created 62
Enabled 63
Expire 62
Fingerprint 63
Key ID 62
Key Type 62
Trust Model 62
examining keys' properties 62
Change Passphrase 63
Keys label 59
opening 14
Size label 60
Trust label 59
uses 58
Validity label 59
platforms
supported 7
preferences
decryption 75
email 77
encryption 74
general 74
key files 76
key generation 75
key server 79
passphrase cache 75
setting 74

- Privacy Enhanced Mail 100
- private and public key pairs
 - creating 3
 - creating with PGP Key Wizard 14
 - viewing your 14
- private key 108
- private keys
 - creating 3
 - key pairs 3
 - creating with PGP Key Wizard 14
 - location of 57
 - overview 2
 - protecting 33–34
 - setting location of 77
 - storing 33–34
 - viewing your 14
- properties
 - viewing a keyring's 62
- protecting
 - your keys 33–34
- public key cryptography
 - overview 2
- public keys
 - advantages of sending to key server 35
 - certifying 3, 96
 - consequences of sending to key server 31
 - copying from email messages 39
 - creating 3
 - key pairs 3
 - creating with PGP Key Wizard 14
 - distributing your 35
 - exchanging with other users 3
 - exporting to files 37
 - getting from a key server 39
 - giving to other users 3
 - importing from files 40
 - including in an email message 37
 - location of 57
 - obtaining others' 38–40
 - overview 2
 - protecting 33–34
 - sending to key server 31, 35–36
 - setting location of 76
 - signing 66, 96
 - storing 33–34

- trading with other users 3
- validating 3
- viewing your 14
- `pubring.skr` file 57

R

- random numbers 91
- receiving
 - private email 43
- revoking
 - keys 72
- RSA 114, 116
- RSA technology
 - keys 8
 - creating 25
- running
 - PGP 12, 13

S

- saving
 - keys 33–34
 - `secring.skr` file 57
- security breach 112
- sending
 - private email 43
- setting
 - passphrase for a key 28
 - preferences 74
- signing
 - deleting signatures 69
 - email 4, 43
 - checking signature 2
 - from Windows Explorer 48–50
 - overview 2
 - via the Clipboard 46–48
 - keys 66
 - public keys 66, 96
 - using Eudora 43–46
- storing
 - keys 33–34
- system requirements 7
- System tray

using PGP from 13

T

traffic analysis 114

trust

granting for key validations 67

Trust Model property 62

trusted introducer 97, 100

U

upgrading

from a previous version of PGP 8

from ViaCrypt 8

user ID 97

user name

adding 64

using

PGP 12

from the Clipboard 13

from the System tray 13

V

validating

keys

granting trust for 67

public keys 3

validity

checking a key's 40

verifying

authenticity of a key 40

email 4

from others 51

from Windows Explorer 54–56

via the Clipboard 53

within Eudora 51–52

versions

of PGP, compatible 7

upgrading to new 8

ViaCrypt

upgrading from 8

viewing

attributes of keyrings 58–63

key attributes 14

private and public key pairs 14

virus 110

W

Windows

system requirements 7

Windows Explorer

decrypting file attachments 54

decrypting from 54–56

encrypting from 48–50

signing from 48–50

verifying from 54–56

worm 110

