ViaCrypt PGP For Windows™

Introduction

ViaCrypt PGP (Pretty Good Privacy) for Windows is a high security cryptographic program for PCs running MS-Windows 3.1. ViaCrypt PGP allows people to exchange files or messages with privacy, authentication, and convenience.

ViaCrypt PGP combines the convenience of the Rivest-Shamir-Adleman (RSA) public key cryptosystem with the speed of conventional cryptography, message digests for digital signatures, data compression before encryption, good ergonomic design, and sophisticated key management. And ViaCrypt PGP performs the public-key functions faster than most other software implementations.

Why Do You Need ViaCrypt PGP?

It's personal. It's private. And it's no one's business but yours. You may be planning a political campaign, preparing your taxes, or negotiating a contract Whatever it is, you don't want your private electronic mail (e-mail) or confidential files read by anyone else. There's nothing wrong with asserting your privacy. Privacy is as apple-pie as the Constitution.

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 by putting your mail inside envelopes?

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 some brave soul 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.

Today, if the Government wants to violate the privacy of ordinary citizens, it has to expend a certain amount of expense and labor to intercept and steam open and read paper mail, and listen to and possibly transcribe spoken telephone conversation. This kind of labor-intensive monitoring is not practical on a large scale. This is only done in important cases when it seems worthwhile.

More and more of our private communications are being routed through electronic channels. Electronic mail is gradually replacing conventional paper mail. E-mail messages are just too easy to intercept and scan for interesting keywords. This can be done easily, routinely, automatically, and undetectably on a grand scale. International cablegrams are already scanned this way on a large scale by the NSA.

We are moving toward a future when the nation will be crisscrossed with high capacity fiber optic data networks linking together all our increasingly ubiquitous personal computers. E-mail will be the norm for everyone, not the novelty it is today. The Government will protect our e-mail with Government-designed encryption protocols. Probably most people will trust that. But perhaps some people will prefer their own protective measures.

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

Phil Zimmermann

About The Program

The user interface or 'front-end' of ViaCrypt PGP for Windows is designed to be easy-to-use, yet powerful for both novices and 'power users'. Novices will appreciate the intuitive design of the interface that 'leads you by the hand'. By clicking the "Encrypt" button, for example, ViaCrypt PGP for Windows will lead you through a short series of simple, familiar dialog boxes, allowing you to easily encrypt any file on your disk.

The 'back end' of ViaCrypt PGP for Windows is a combination of a DLL for certain key management functions and a 'QuickWIn cryptographic kernel.

Getting Started

Starting ViaCrypt PGP For Windows

Double-click on the ViaCrypt PGP for Windows icon in the ViaCrypt PGP Program Manager Group. The program should begin to run, displaying its Main Window.

Generating Your First Key

One of the first things you will want to do in order to use ViaCrypt PGP is to generate your own public/secret key pair. Choose the "Key Management" button on the Main Window. The Key Management Window should appear:

Choose the "Generate your own key pair" button. ViaCrypt PGP for Windows will open a window and prompt you for the information it needs to generate a new key pair.

At this point you are ready to begin using ViaCrypt PGP to digitally sign your files and messages, and to send your public key to others with whom you want to communicate securely. Later, after they have also sent you their public keys, you can send them encrypted e-mail messages.

Exchanging Keys With Colleagues

To communicate securely with a colleague, you'll both need to exchange keys with each other. In order for your colleague to send you an encrypted file or message, you must provide your colleague with a copy of your public key. And the reverse is also true: To send an encrypted message to a colleague, you will need his or her public key.

The Key Management Window contains two buttons that are used when exchanging keys. The "Copy (Extract) key from keyring to file" is used to create a file containing a copy of your public key. You can then send that file to your colleague.

When you receive the file containing your colleague's public key, the "Add key from file to your keyring" button is used to add his or her key to your keyring.

Procedures (How To)

Encrypt a File

Choose this button to encrypt any file on your disk.

1. Select the file you want to encrypt in the "Select FILE to Encrypt" dialog box.

2. Next, select the recipient(s) using the "Select Recipients" dialog box. The recipient's public key will be used to encrypt the file. After the file is encrypted, only the recipient will be able to decrypt it.

3. Using the "Save Encrypted File As" dialog box, specify a name for the encrypted file, or accept the default.

4. Next you should see a "Working" box. You may also see briefly an icon representing the cryptographic kernel near the bottom of your screen.

When the "Working" box disappears, your file has been encrypted.

See Also:

Making an Encrypted File "For Your Eyes Only"

Disabling ASCII Encoding

Suppressing Public Key Encryption

Leaving No Traces of Plaintext on the Disk

Sending ASCII Text Across Different Machine Environments

Encrypting a File Using Drag and Drop

Making an Encrypted File "For Your Eyes Only"

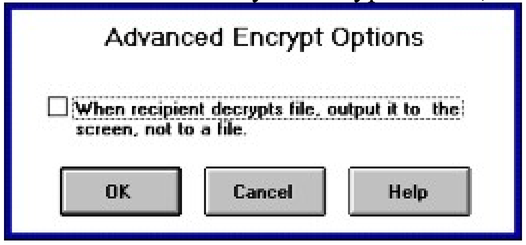
(When recipient decrypts file, output it to the screen, not to a file.)

Occasionally, you may be encrypting a file that you feel is sufficiently sensitive that upon decryption, must be displayed directly on the recipient's screen and not written to disk.

To set the "For Your Eyes Only" flag in the encrypted file, choose the "Settings" button in the Main Window. Then select the check box labeled:

Enable Advanced Options for Encrypt/Sign/Decrypt

Thereafter when you encrypt the file, an additional dialog box will be displayed:



Advanced Encrypt Options

☐ When recipient decrypts file, output it to the screen, not to a file.

Select the check box and choose OK. When the recipient decrypts the file, the following prompt will appear:

```
This message is marked "For your eyes only".  Display now (Y/n)?
```

If the recipient enters 'Y', the decrypted file will be displayed and not written to a file. This feature is the safest way for you to prevent your sensitive message from being inadvertently left on the recipient's disk.

Disabling ASCII Encoding

By default, ViaCrypt PGP for Windows will encode the encrypted and signed file in an ASCII armor so that it consists entirely of ASCII characters. This makes it easy to send encrypted files by e-mail.

You can disable this ASCII armor function, forcing ViaCrypt PGP to save the file as binary data. Choose the "Settings" button, and de-select the check box labeled:

ASCII Armor PGP Output

Suppressing Public Key Encryption

Sometimes you just need to encrypt a file the old-fashioned way, with conventional single-key encryption. ViaCrypt PGP's default mode is to use both conventional and public key encryption together when encrypting a file. It is a simple matter to 'turn off' the public key cryptography and use just the conventional cryptography in ViaCrypt PGP.

To do so, choose the "Settings" button in the Main Window. Select the check box labeled:

Encrypt using IDEA only, not IDEA and RSA

When you encrypt a file while this option is selected, ViaCrypt PGP will prompt you for a pass phrase to use as a conventional key to encipher the file. This pass phrase need not be (and, indeed, should not be) the same pass phrase that you use to protect your own secret key. Note that ViaCrypt PGP compresses the plaintext before encrypting it.

ViaCrypt PGP will not encrypt the file the same way twice, even if you use the same pass phrase every time.

Leaving No Traces of Plaintext on the Disk

After ViaCrypt PGP makes a ciphertext file for you, you can have ViaCrypt PGP automatically overwrite the plaintext file and delete it, leaving no trace of plaintext on the disk so that no one can recover it later using a disk block scanning utility. This is useful if the plaintext file contains sensitive information that you don't want to keep.

To wipe out the plaintext file after producing the ciphertext file, choose the "Settings" button on the Main Window, and select the check box labeled:

Delete (Wipe) input file after encryption

Obviously, you should be careful with this option. Also note that this will not wipe out any fragments of plaintext that your word processor might have created on the disk while you were editing the message before running ViaCrypt PGP. Most word processors create backup files, scratch files, or both. Also, it overwrites the file only once, which is enough to thwart conventional disk recovery efforts, but not enough to withstand a determined and sophisticated effort to recover the faint magnetic traces of the data using special disk recovery hardware.

Sending ASCII Text Across Different Machine Environments

You may use ViaCrypt PGP to encrypt any kind of file: binary 8-bit data or ASCII text. Probably the most common usage of ViaCrypt PGP will be for e-mail, when the plaintext is ASCII text.

ASCII text is sometimes represented differently on different machines. For example, on an MS-DOS system, all lines of ASCII text are terminated with a carriage return followed by a linefeed. On a Unix system, all lines end with just a linefeed. On a Macintosh, all lines end with just a carriage return. This is a sad fact of life.

Normal unencrypted ASCII text messages are often automatically translated to some common "canonical" form when they are transmitted from one machine to another. Canonical text has a carriage return and a linefeed at the end of each line of text. For example, the popular KERMIT communication protocol can convert text to canonical form when transmitting it to another system. This gets converted back to local text line terminators by the receiving KERMIT. This makes it easy to share text files across different systems.

By default, ViaCrypt PGP converts ASCII text to canonical form before it gets encrypted. At the receiving end, the decrypted plaintext is automatically converted back to whatever text form is appropriate for the local environment.

To disable this feature, choose the "Settings" button on the Main Window, and de-select the check box labeled:

Use Canonical format for newlines on textfile

This mode is automatically turned off if ViaCrypt PGP detects that the plaintext file contains what it thinks is non-text binary data.

For ViaCrypt PGP users that use non-English 8-bit character sets, when ViaCrypt PGP converts text to canonical form, it may convert data from the local character set into the LATIN1 (ISO 8859-1 Latin Alphabet 1) character set, depending on the setting of the CHARSET parameter in the ViaCrypt PGP configuration file. LATIN1 is a superset of ASCII, with extra characters added for many European languages.

Sign a File

Choose this button to digitally sign any file on your disk.

1. Select the file you want to sign in the "Select File to Sign" dialog box.

2. If you have more than one secret key on your secret keyring, ViaCrypt PGP will ask you to select the key to use.

3. Using the "Save Signed File As" dialog box, specify a name for the signed file.

4. A dialog box will appear on your screen, asking you for your pass phrase. Type your pass phrase and press ENTER.

5. ViaCrypt PGP will then digitally sign your file:

See Also:

Format of Digitally Signed ASCII Text Files

Format of Digitally Signed Binary Files

Separating Signatures from Messages

Signing Files Using Drag and Drop

Format of Digitally Signed ASCII Text Files

When digitally signing a file containing only ASCII text, by default ViaCrypt PGP's output will contain the ASCII text in its readable form, followed by the digital signature. The result will look something like this:

```
-----BEGIN PGP SIGNED MESSAGE-----

Now is the time for all
good men to come to the
aid of their country.
-----BEGIN PGP SIGNATURE-----
Version: 2.7

iQBVAgUBLlkKnndnGF5n7PE9AQEDhQH+PPPJzXK4HvdgdlfMWVW2nWmQvpBBAIsZ
Z8TqP3/d6TrOfD1z5KCaUKjEfiY9NueWhGATr+eBL5qkyb+a/TKE0Q==
=wM76
-----END PGP SIGNATURE-----
```

You can force ViaCrypt PGP to compress the ASCII text and encode it and the digital signature in an ASCII armor like so:

```
-----BEGIN PGP MESSAGE-----
Version: 2.7

owHrZAhlYmXUi+R5VZ4uEZf+5qMtI+O5x4z/LN5dum3AzMRzbkp2W+uLmTfeXTHb
wqfH8NaQI6CeZ4vM2r+yBQy/9+y4/9Bk47xlsV7XM5hCG93edT0w57/5luWtTMOa
2BK+ZCv91IrE3IKcVL2SihIGIPDLL1fILFYoyUhVKMnMTVVIyy9SSMzJUeDlSs/P
T1HITc1TKMlXSM4HSgFpkDJersTMFIX8NBAnswgoVZpXUlSpBwA=
=GuEy
-----END PGP MESSAGE-----
```

To do so, choose the "Settings" button on the Main Window, and:

1. De-select the check box labeled "Use Clearsig option for digital signatures"
2. Verify the "ASCII Armor PGP Output" check box is selected:

You can also force ViaCrypt PGP to save the signed file as a binary file. Choose the "Settings" button on the Main Window, and deselect the check box labeled "ASCII Armor PGP Output"

Format of Digitally Signed Binary Files

By default, ViaCrypt PGP for Windows will encode the signed file in an ASCII armor so that it consists entirely of ASCII characters. This makes it easy to send signed binary files, such as spreadsheets and word-processing documents, by e-mail.

You can disable this ASCII armor function, forcing ViaCrypt PGP to save the signed file as binary data. Choose the "Settings" button from the Main Window, and de-select the check box labeled:

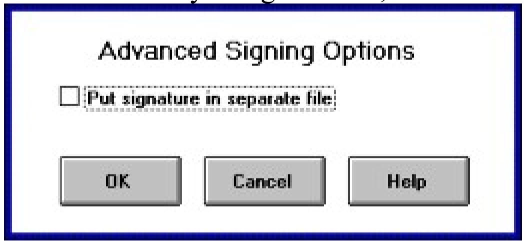
ASCII Armor PGP Output

Separating Signatures from Messages

Normally, the file created by ViaCrypt PGP contains both the signed material and the digital signature. This makes it convenient in simple cases to check signatures. It is desirable in some circumstances to have signature certificates stored separately from the messages they sign (for example, when you want to digitally sign a word processing document, but still be able to view it with your word processor). To generate signature certificates that are detached from the signed material, choose the "Settings" button on the Main Window and check the box labeled:

Enable Advanced Options for Encrypt/Sign/Decrypt

Then when you sign the file, an additional dialog box will be displayed:


Advanced Signing Options

☐ Put signature in separate file

Select the check box and choose OK. The title bar of the dialog box that follows will be "Save Signature File As" instead of "Save Signed File As". Specify the name for the file containing just the digital signature.

When signing files in this manner and sending them to someone, you need to send both files: the file containing the signed material and the file containing the signature. The recipient must have both files to check the file's integrity. When the recipient attempts to process the signature file, ViaCrypt PGP notices that there is no text in the same file with the signature and prompts the user for the filename of the text. Only then can ViaCrypt PGP properly check the signature integrity.

Encrypt and Sign a File

Choose this button to simultaneously encrypt and sign any file on your disk.

1. Select the file in the "Select File to Encrypt and Sign" dialog box.

2. Select the recipient using the "Select Recipients" dialog box. The recipient's public key will be used to encrypt the file. After the file is encrypted, only the recipient will be able to decrypt it.

3. If you have more than one secret key on your secret keyring, ViaCrypt PGP will ask you to select the key to use to sign the file.

4. Next specify a name for the encrypted and signed file using the "Save Encrypted and Signed File As" dialog box.

5. A dialog box will appear on your screen, asking you for your pass phrase. Type your pass phrase and press ENTER.

6. ViaCrypt PGP will then encrypt and sign the file.

See Also:

Making an Encrypted File "For Your Eyes Only"

Disabling ASCII Encoding

Suppressing Public Key Encryption

Leaving No Traces of Plaintext on the Disk

Sending ASCII Files Across Different Machine Environments

Encrypting and/or Signing Files Using Drag and Drop

Decrypt a File

Choose this button to decrypt a file. If the file is also digitally signed, the signature will be checked.

1. Select the file you want to decrypt using the "Select File to Decrypt" dialog box.
2. Using the "Save Decrypted File As" dialog box, specify the name for the decrypted file, or accept the default.
3. A dialog box will appear on your screen, asking you to enter your pass phrase.   Type the correct pass phrase and press ENTER.
4. ViaCrypt PGP will decrypt the file.
5. If the file is also digitally signed, the signature will be checked.

See Also:

Output Decrypted Text To Your Screen, not to a File

Decrypting a Message and Leaving the Signature On It

Preserving the Original Plaintext Filename

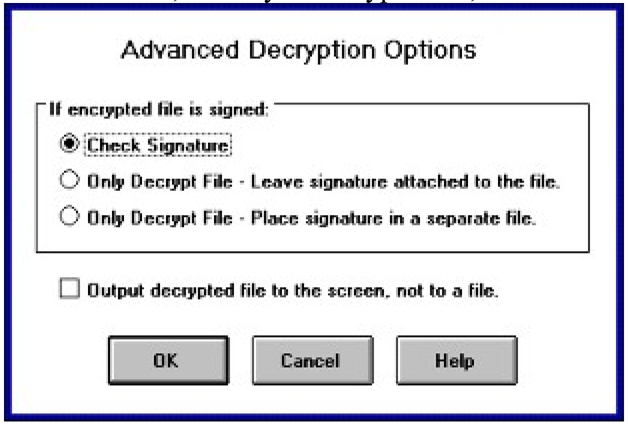
Decrypting Files Using Drag and Drop

Output Decrypted Text To Your Screen, not to a File

Occasionally, you may be decrypting an ASCII text file that you feel is sufficiently sensitive that it must be displayed on your screen instead of written to your disk. In this case, choose the "Settings" button in the Main Window, and select the check box labeled:

Enable Advanced Options for Encrypt/Sign/Decrypt

Thereafter, When you decrypt a file, an additional dialog box will be displayed:


Advanced Decryption Options

If encrypted file is signed:

- (●) Check Signature
- ( ) Only Decrypt File - Leave signature attached to the file.
- ( ) Only Decrypt File - Place signature in a separate file.

☐ Output decrypted file to the screen, not to a file.

Select the check box labeled:

Output decrypted file to the screen, not to a file.

If the decrypted file is ASCII text, it will be displayed on your screen one page at a time.

Decrypting a Message and Leaving the Signature On It

If the file you are decrypting is also digitally signed, you usually want ViaCrypt PGP to verify the signature as well as decrypt the file. But sometimes you want to decrypt an encrypted file, leaving the signature either

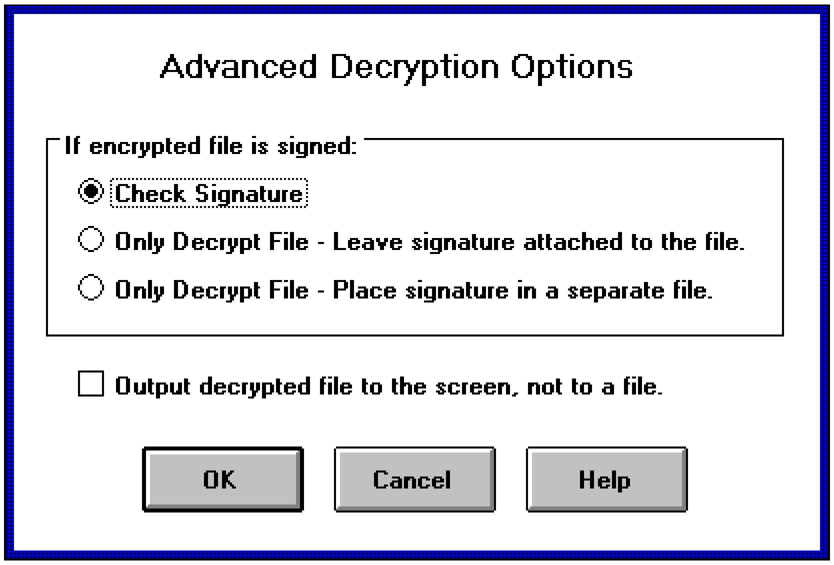
still attached, or:

in a separate file.

This may be useful if you want to send a copy of a signed document to a third party, perhaps re-encrypting it.

To specify encrypted and signed files are to be decrypted only, choose the "Settings" button on the Main Window, and select the check box labeled: "Enable Advanced Options for Encrypt/Sign/Decrypt"   Thereafter when you decrypt a file, an additional dialog box will be displayed:

Click on the parts of the window where the appears for descriptions of each item in the window:



Advanced Decryption Options

If encrypted file is signed:

- [✓ Check Signature]
- Only Decrypt File - Leave signature attached to the file.
- Only Decrypt File - Place signature in a separate file.

☐ Output decrypted file to the screen, not to a file.

Simply select the appropriate option from the "If encrypted file is signed" group of buttons, and press OK. The file will be decrypted, and if it also contains a digital signature, the signature is either left intact or placed in a separate file, depending upon your choice

Preserving the Original Plaintext Filename

Normally, you specify the name for the decrypted file, but you can tell ViaCrypt PGP you want to preserve the original plaintext filename and use it as the name of the decrypted output file. This is useful if ViaCrypt PGP is used on files whose names are important to preserve. To recover the original names of files when decrypting, choose the "Settings" button on the Main Window, and select the check box labeled:

Preserve original plaintext filename when decrypting

Verify a Digital Signature

Choose this button to verify the digital signature on a digitally signed file.

For example, if you receive a file:

```
-----BEGIN PGP SIGNED MESSAGE-----

Now is the time for all
good men to come to the
aid of their country.
-----BEGIN PGP SIGNATURE-----
Version: 2.7

iQBVAgUBLlkKnndnGF5n7PE9AQEDhQH+PPPJzXK4HvdgdlfMWVW2nWmQvpBBAIsZ
Z8TqP3/d6TrOfD1z5KCaUKjEfiY9NueWhGATr+eBL5qkyb+a/TKE0Q==
=wM76
-----END PGP SIGNATURE-----
```

You can read the file, but you need an easy way to verify the digital signature attached to the bottom.

1. ViaCrypt PGP will display a dialog box for you to specify the name of the signed file.

2. Next you should see a "Working" box. You may also see briefly an icon representing the cryptographic kernel near the bottom of your screen.

3. When the "Working" box disappears, the results of the signature verification will appear in the "Results" dialog box:

4. If the display If the "Results" box contains the phrase "WARNING: Bad signature...", it indicates that the file may be corrupted (accidentally or intentionally), or the signature may be a forgery.

See Also:

Detaching Signatures from Signed Files

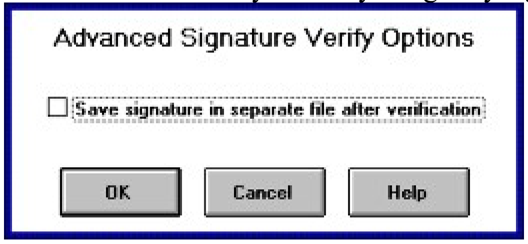
Verifying Signatures Using Drag and Drop

Detaching Signatures from Signed Files

You may want to verify a digital signature, and at the same time separate the signature and the signed data into two distinct files. To do so, choose the "Settings" button from the Main Window, and select the check box labeled:

Enable Advanced Options for Encrypt/Sign/Decrypt.

Thereafter when you verify a digitally signed file, an additional dialog box will appear:


Advanced Signature Verify Options

☐ Save signature in separate file after verification

Select the check box. ViaCrypt PGP will first verify the signature. Then it will place the signature in a file called:

```
filename.sig
```

where the filename portion is the same as that of the input file.

Key Management

Since the time of Julius Caesar, key management has always been the hardest part of cryptography. One of the principal distinguishing features of ViaCrypt PGP is its sophisticated key management. When you choose the "Key Management" button on the Main Window, the Key Management Window appears. The Key Management Window provides all of the functions associated with managing cryptographic key.

Running Windows Notepad

Use the "Run Notepad" button to run a copy of Windows Notepad. This is useful when you want to create and encrypt a short message, or to view a text file you have just decrypted

You can also specify that some other editor is to be used in place of Notepad.   Add a line to the [Settings] section of VPGPW.INI:

```
Editor=Your_Editor's_Path_and_Name
```

For example, if your editor is named "MYED.EXE" and is located in "C:\MYAPPS", then specify:

```
Editor=c:\myapps\myed.exe
```

Note that ViaCrypt PGP doesn't do anything fancy. It just executes the string after the '=' when you press the button on the Main Window.

Changing Configuration Parameters (Settings)

Two types of configuration parameters are available:

The Settings Window

Choose the "Settings" button on the Main Window to display the Settings Window. The Settings Window provides access to the most common options. For example, you can specify that decrypted files retain the same filename as was used before encryption. You can also change the foreground and background colors of ViaCrypt PGP's windows.

CONFIG.TXT File

The configuration parameters in the CONFIG.TXT file are those that are not as commonly used. The CONFIG.TXT file should be in your VPGP directory (or in the directory where you installed ViaCrypt PGP for Windows). Configuration parameters in CONFIG.TXT should be changed using NOTEPAD or some other editor.

Exiting The Program

Press the "Quit" button on the Main WINDOW to Exit ViaCrypt PGP.

Drag and Drop

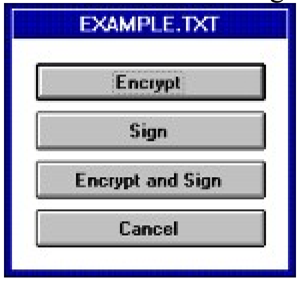
Decrypting Files Using Drag and Drop

Dropping an encrypted file onto ViaCrypt PGP causes the same action as pressing the "Decrypt" button in the Main Window, except that you are not asked to specify the file to decrypt. If the encrypted file has also been signed, the signature will be checked unless you have specified otherwise via the "Advanced Encrypt/Sign/Decrypt Options".

Encrypting and/or Signing Files Using Drag and Drop

Dragging a file from Windows File Manager and dropping it on ViaCrypt PGP's Main Window or its icon is a convenient way to encrypt and/or digitally sign the file.

When you drop a file on the Main Window or icon, ViaCrypt PGP analyzes it. If the file does not appear to have been created by ViaCrypt PGP, then it assumes you want to encrypt it, sign it, or both. The following window is displayed:



Note that the file's name is displayed in the window's title bar. You can then select the operation to perform.

Verifying Signatures Using Drag And Drop

Dropping a digitally signed file onto ViaCrypt PGP causes the same action as pressing the "Verify Signature" button in the Main Window

Configuration Parameters

Two types of configuration parameters are available:

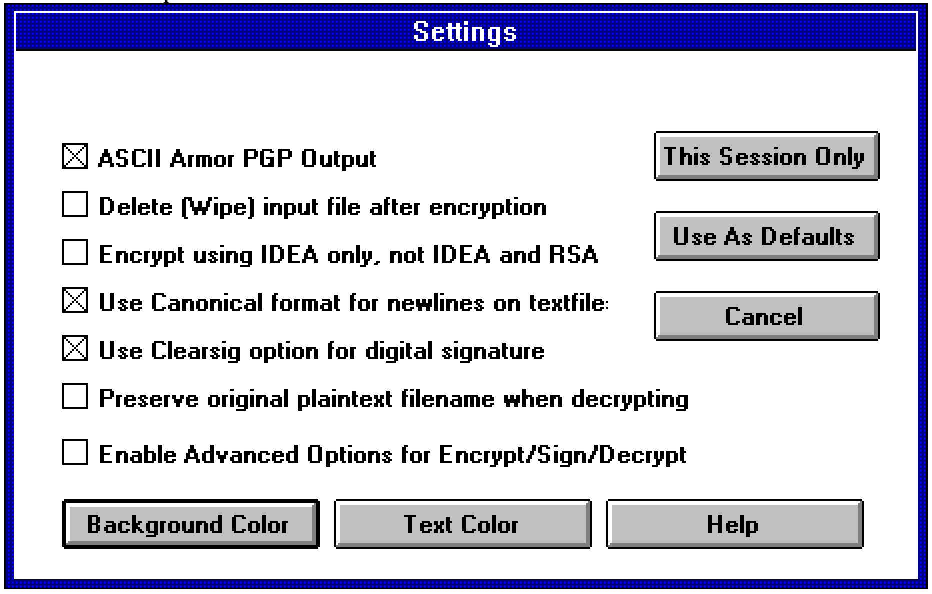
The Settings Window

CONFIG.TXT File

The Settings Window

The Settings Window appears when you choose the Settings button in the Main Window. The Settings Window provides access to the most often needed configuration parameters. Less often used configuration parameters appear in the CONFIG.TXT file.

The Settings Window is Shown below. Click on the parts of the window where the appears for descriptions of each item in the window:



Settings

☒ ASCII Armor PGP Output

☐ Delete (Wipe) input file after encryption

☐ Encrypt using IDEA only, not IDEA and RSA

☒ Use Canonical format for newlines on textfile:

☒ Use Clearsig option for digital signature

☐ Preserve original plaintext filename when decrypting

☐ Enable Advanced Options for Encrypt/Sign/Decrypt

CONFIG.TXT File

ViaCrypt PGP has a number of parameters that can be specified by the user in a special configuration text file called "CONFIG.TXT". This file is normally in the VPGP directory. If you installed ViaCrypt PGP for Windows in a different directory, then CONFIG.TXT will be in the directory where ViaCrypt PGP for Windows resides.   The configuration parameters in CONFIG.TXT should be changed using NOTEPAD or some other editor.

This file contains parameters used by the cryptographic kernal that is executed by the Windows shell. Some of the parameters in CONFIG.TXT must be defined in a specific manner for ViaCrypt PGP for Windows to work properly:

PARAMETER	REQUIRED SETTING
ARMOR	OFF (or not defined)
TEXTMODE	OFF (or not defined)

Configuration parameters may be assigned integer values, character string values, or on/off values, depending on what kind of configuration parameter it is. Text in configuration parameters is case-insensitive. This means that "ENCRYPTTOSELF=YES" is the same as "EncryptToSelf=Yes". A sample configuration file is provided with ViaCrypt PGP, so you can see some examples.

In the configuration file, blank lines are ignored, as is anything following the '#' comment character. Keywords are not case-sensitive. Here is a short sample fragment of a typical configuration file:

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

If some configuration parameters are not defined in the configuration file, or if there is no configuration file, or if ViaCrypt PGP can't find the configuration file, the values for the configuration parameters default to some reasonable value.

Configuration Parameters

TMP	Directory Pathname for Temporary Files
LANGUAGE	Foreign Language Selector
MYNAME	Default User ID for Making Signatures
ENCRYPTTOSELF	Always Include Yourself a Recipient for Encrypted Files
TEXTMODE	Assuming Plaintext is a File
CHARSET	Specifies Local Character Set for Text Files
ARMOR	Enable ASCII Armor Output
ARMORLINES	Size of ASCII Armor Multipart Files
KEEPBINARY	Keep Binary Ciphertext Files After Decrypting

COMPRESS	Enable Compression
COMPLETES_NEEDED	Number of Completely Trusted Introducer Needed
MARGINALS_NEEDED	Number of Marginally Trusted Introducers Needed
CERT_DEPTH	How Deep May Introducers Be Nested
PAGER	Selects Command to Display Plaintext Output
SHOWPASS	Echo Pass Phrase to User
TZFIX	Time Zone Adjustment
CLEARSIG	Enable Signed Messages to be Encapsulated as Clear Text
VERBOSE	Quiet, Normal, or Verbose Messages
INTERACTIVE	Ask for Confirmation for Key Adds
PKCS_COMPAT	Backwards Compatibility with Version 2.4
PUBRING	Location for Your Public Keyring
SECRING	Location for Your Secret Keyring
RANDSEED	Location for Your Random Number Seed File
COMMENT	Adding a Comment Line to the Output

Cryptography & Privacy Basics

Introduction to Cryptography

First, some elementary terminology. Suppose I want to send you a message, but I don't want anyone but you to be able to read it. I can "encrypt", or "encipher" the message, which means I scramble it up in a hopelessly complicated way, rendering it unreadable to anyone except you, the intended recipient of the message. I supply a cryptographic "key" to encrypt the message, and you have to use the same key to decipher or "decrypt" it. At least that's how it works in conventional "single-key" cryptosystems.

In conventional cryptosystems, such as the U.S. Federal Data Encryption Standard (DES), a single key is used for both encryption and decryption. This means that a 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?

In public key cryptosystems, everyone has two related complementary keys, a publicly revealed key and a secret key. Each key unlocks the code that the other key makes. Knowing the public key does not help you deduce the corresponding secret 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 a conventional cryptosystem requires.

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

Message authentication is also provided. The sender's own secret key can be used to encrypt a message, thereby "signing" it. This creates a digital signature of a message, which the recipient (or anyone else) can check by using the sender's public key to decrypt it. 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 secret key that made that signature. Forgery of a signed message is infeasible, and the sender cannot later disavow his signature.

These two processes can be combined to provide both privacy and authentication by first signing a message with your own secret key, then encrypting the signed message with the recipient's public key. The recipient reverses these steps by first decrypting the message with her own secret key, then checking the enclosed signature with your public key. These steps are done automatically by the recipient's software.

Because the public key encryption algorithm is much slower than conventional single-key encryption, encryption is better accomplished by using a high-quality fast conventional single-key encryption algorithm 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 recipient uses her own secret key to recover this temporary session key, and then uses that key to run the fast conventional single-key algorithm to decipher the large ciphertext message.

Public keys are kept in individual "key certificates" that include the key owner's user ID (which is that person's name), a timestamp of when the key pair was generated, and the actual key material. Public key certificates contain the public key material, while secret key certificates contain the secret key material. Each secret key is also encrypted with its own password, in case it gets stolen. A key file, or "key ring" contains one or more of these key certificates. Public keyrings contain public key certificates, and secret key rings contain secret key certificates.

The keys are also internally referenced by a "key ID", which is an "abbreviation" of the public key (the least significant 64 bits of the large public key). When this key ID is displayed, only the lower 24 bits are shown for further brevity. While many keys may share the same user ID, for all practical purposes no two keys share the same key ID.

ViaCrypt PGP uses "message digests" to form signatures. A message digest is a 128-bit cryptographically strong one-way hash function of the message. 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 computationally infeasible for an attacker to devise a substitute message that would produce an identical message digest. The message digest gets encrypted by the secret key to form a signature.

Documents are signed by prefixing them with signature certificates, which contain the key ID of the key that was used to sign it, a secret-key-signed message digest of the document, and a timestamp of when the signature was made. The key ID is used by the receiver to look up the sender's public key to check the signature. The receiver's software automatically looks up the sender's public key and user ID in the receiver's public key ring.

Encrypted files are prefixed by the key ID of the public key used to encrypt them. The receiver uses this key ID message prefix to look up the secret key needed to decrypt the message. The receiver's software automatically looks up the necessary secret decryption key in the receiver's secret key ring.

These two types of keyrings are the principal method of storing and managing public and secret keys. Rather than keep individual keys in separate key files, they are collected in keyrings to facilitate the automatic lookup of keys either by key ID or by user ID. Each user keeps his own pair of keyrings. An individual public key is temporarily kept in a separate file long enough to send to your friend who will then add it to her key ring.

Beware of Snake Oil

Philip Zimmermann has this to say on the subject of encryption products and trust:

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. So cock-sure.

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 good conventional algorithm recommended by the Government for commercial use (but not for classified information, oddly enough - hmmm). There are several "modes of operation" the 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. 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 the 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 products do not come from companies that specialize in cryptographic technology.

There is a company in Utah that sells a package that cracks the built-in encryption schemes used by many popular word processing and spreadsheet programs. 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 the author of this program, 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. He also told me that the password encryption feature of a popular and widely-used file compression program can often be easily broken, and that his law enforcement customers already have that service regularly provided to them from another vendor.

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.

I remember a conversation with 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 U.S. 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.

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. NSA has been pushing a conventional encryption algorithm that they designed, and they won't tell anybody how it works because that's classified. They want 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? Are they deliberately selling snake oil?

I'm not as certain about the security of ViaCrypt PGP as I once was about my brilliant encryption software from college. If I were, that would be a bad sign. But I'm pretty sure that ViaCrypt PGP does not contain any glaring weaknesses. The crypto algorithms were developed by people at high levels of civilian cryptographic academia, and have been individually subject to extensive peer review. It's reasonably well researched, and has been years in the making. And I don't work for the NSA. I hope it doesn't require too large a "leap of faith" to trust the security of ViaCrypt PGP.

A Peek Under the Hood

Random Numbers

ViaCrypt PGP uses a cryptographically strong pseudorandom number generator for creating temporary conventional session keys. The seed file for this is called "randseed.bin". It too can be kept in whatever directory is indicated by the PGPPATH environmental variable. If this random seed file does not exist, it is automatically created and seeded with truly random numbers derived from timing your keystroke latencies.

This generator reseeds the disk file each time it is used by mixing in new key material partially derived with 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 to key the conventional encryption engine for the random generator.

This random seed file should be at least slightly 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 at least try to keep it from falling into the wrong hands.

If you feel uneasy about trusting any algorithmically derived random number source however strong, keep in mind that you already trust the strength of the same conventional cipher to protect your messages. If it's strong enough for that, then it should be strong enough to use as a source of random numbers for temporary session keys. Note that ViaCrypt PGP still uses truly random numbers from physical sources (mainly keyboard timings) to generate long-term public/secret key pairs.

ViaCrypt PGP's Conventional Encryption Algorithm

As described earlier, ViaCrypt PGP "bootstraps" into a conventional single-key encryption algorithm by using a public key algorithm to encipher the conventional session key and then switching to fast conventional cryptography. So let's talk about this conventional encryption algorithm. It isn't the DES.

Time, technology, and recent advances in cryptanalysis have taken their toll on the security of DES. Eli Biham and Adi Shamir published a paper in the CRYPTO '92 proceedings describing an attack upon DES using differential cryptanalysis. At CRYPTO '93, Michael Wiener of Bell Northern Research in Ottawa presented a paper on a chip he designed that guesses DES keys at high speed until it finds the right one. Using this chip, a processor could be built for about $1 million that would provide a solution in an average of 3.5 hours. This implementation could be scaled to reduce the time to about 2 minutes.

ViaCrypt PGP does not use the DES as its conventional single-key algorithm to encrypt messages. Instead, ViaCrypt PGP uses a different conventional single-key block encryption algorithm, called IDEA. A future version of ViaCrypt PGP may support the DES as an option, if enough users ask for it. But I suspect IDEA is much better than DES.

For the cryptographically curious, the IDEA cipher has a 64-bit block size for the plaintext and the ciphertext. It uses a key size of 128 bits. It is based on the design concept of "mixing operations from different algebraic groups". It runs much faster in software than the DES. Like the DES, it can be used in cipher feedback (CFB) and cipher block chaining (CBC) modes. ViaCrypt PGP uses it in 64-bit CFB mode.

The IDEA block cipher was developed at ETH in Zurich by James L. Massey and Xuejia Lai, and published in 1990. This is not a "home-grown" algorithm. Its designers have a distinguished reputation in the cryptologic community. Early published papers on the algorithm called it IPES (Improved Proposed Encryption Standard), but they later changed the name to IDEA (International Data Encryption Algorithm). So far, IDEA has resisted attack much better than other ciphers such as FEAL, REDOC-II, LOKI, Snefru and Khafre. And preliminary evidence suggests that IDEA may be more resistant than the DES to Biham & Shamir's highly successful differential cryptanalysis attack. Biham and Shamir have been examining the IDEA cipher for weaknesses, without success. Academic cryptanalyst groups in Belgium, England, and Germany are also attempting to attack it, as well as the military services from several European countries. As this new 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.

A famous hockey player once said, "I try to skate to where I think the puck will be." A lot of people are starting to feel that the days are numbered for the DES. I'm skating toward IDEA.

It is not ergonomically practical to use pure RSA with large keys to encrypt and decrypt long messages. Absolutely no one does it that way in the real world. But perhaps you are concerned that the whole package is weakened if we use a hybrid public key and conventional scheme just to speed things up. After all, a chain is only as strong as its weakest link. Many people less experienced in cryptography mistakenly believe that RSA is intrinsically stronger than any conventional cipher. It's not. RSA can be made weak by using weak keys, and conventional ciphers can be made strong by choosing good algorithms. It's usually difficult to tell exactly how strong a good conventional cipher is, without actually cracking it. A really good conventional cipher might possibly be harder to crack than even a "military grade" RSA key. The attraction of public key cryptography is not because it is intrinsically stronger than a conventional cipher - its appeal is because it helps you manage keys more conveniently.

Data Compression

ViaCrypt PGP normally compresses the plaintext before encrypting it. It's too late to compress it 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 seems worth it, at least in my cautious opinion.

Files that are too short to compress or just don't compress well are not compressed by ViaCrypt PGP.

If you prefer, you can use PKZIP to compress the plaintext before encrypting it. PKZIP is a widely-available and effective MS-DOS shareware compression utility from PKWare, Inc. Or you can use ZIP, a PKZIP-compatible freeware compression utility on Unix and other systems, available from Jean-Loup Gailly. There is some advantage in using PKZIP or ZIP in certain cases, because unlike ViaCrypt PGP's built-in compression algorithm, PKZIP and ZIP have the nice feature of compressing multiple files into a single compressed file, which is reconstituted again into separate files when decompressed. ViaCrypt PGP will not try to compress a plaintext file that has already been compressed. After decrypting, the recipient can decompress the plaintext with PKUNZIP. If the decrypted plaintext is a PKZIP compressed file, ViaCrypt PGP automatically recognizes this and advises the recipient that the decrypted plaintext appears to be a PKZIP file.

For the technically curious readers, the current version of ViaCrypt PGP uses the freeware ZIP compression routines written by Jean-loup Gailly, Mark Adler, and Richard B. Wales. This ZIP software uses functionally-equivalent compression algorithms as those used by PKWare's new PKZIP 2.0. This ZIP compression software was selected for ViaCrypt PGP mainly because it has a really good compression ratio and because it's fast.

Message Digests and Digital Signatures

To create a digital signature, ViaCrypt PGP encrypts with your secret key. But ViaCrypt PGP doesn't actually encrypt your entire message with your secret key - that would take too long. Instead, ViaCrypt PGP encrypts a "message digest".

The message digest is a compact (128 bit) "distillate" of your message, similar in concept to a checksum. You can also think of it as a "fingerprint" of the message. 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 would 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.

A message digest alone is not enough to authenticate a message. The message digest algorithm is publicly known, and does not require knowledge of any secret keys to calculate. If all we did was attach a message digest to a message, then a forger could alter a message and simply attach a new message digest calculated from the new altered message. To provide real authentication, the sender has to encrypt (sign) the message digest with his secret key.

A message digest is calculated from the message by the sender. The sender's secret key is used to encrypt the message digest and an electronic timestamp, forming a digital signature, or signature certificate. The sender sends the digital signature along with the message. The receiver receives the message and the digital signature, and recovers the original message digest from the digital signature by decrypting it with the sender's public key. The receiver computes a new message digest from the message, and checks to see if it matches the one recovered from the digital signature. If it matches, then that proves the message was not altered, and it came from the sender who owns the public key used to check the signature.

A potential forger would have to either produce an altered message that produces an identical message digest (which is infeasible), or he would have to create a new digital signature from a different message digest (also infeasible, without knowing the true sender's secret key).

Digital signatures prove who sent the message, and that the message was not altered either by error or design. It also provides non-repudiation, which means the sender cannot easily disavow his signature on the message.

Using message digests to form digital signatures has other advantages besides being faster than directly signing the entire actual message with the secret key. Using message digests allows signatures to be of a standard small fixed size, regardless of the size of the actual message. It also allows the software to check the message integrity automatically, in a manner similar to using checksums. And it allows signatures to be stored separately from messages, perhaps even in a public archive, without revealing sensitive information about the actual messages, because no one can derive any message content from a message digest.

The message digest algorithm used here is the MD5 Message Digest Algorithm, placed in the public domain by RSA Data Security, Inc. MD5's designer, Ronald Rivest, writes this about MD5:

"It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 264 operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2^64 operations. The MD5 algorithm has been carefully scrutinized for weaknesses. It is, however, a relatively new algorithm and further security analysis is of course justified, as is the case with any new proposal of this sort. The level of security provided by MD5 should be sufficient for implementing very high security hybrid digital signature schemes based on MD5 and the RSA public key cryptosystem."

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 is 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. Let's first look at a potential disaster, then at how to safely avoid it with ViaCrypt PGP.

Suppose you wanted 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 secret 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 secret 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 mutual 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 secret key.

This would create a signed public key certificate, and would show that Alice's key had not been tampered with. This requires 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 "key server", or as a "Certifying Authority". Any public key certificates bearing the key server'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 key server's public key, so that the key server's signatures could be verified.

A trusted centralized key server or 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 a trusted introducers for their friends would probably work better than a centralized key server.

One of the attractive features of ViaCrypt PGP is that it can operate equally as well in either a centralized environment with a Certifying Authority or the "organic", more decentralized approach. ViaCrypt PGP tends to emphasize the latter. It better reflects the natural way humans interact on a personal social level, and allows people to better choose who they can trust for key management.

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 actually belongs to the person it claims to. 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 - and you will be tempted - never, 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 firsthand 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 file to her to get her to confirm that the key you have really is 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 were completely confident that the key really belonged 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.

Trust is not necessarily transferable; I have a friend who I trust not to lie. He's a gullible person who trusts politicians not to lie. That doesn't mean I trust politicians not to lie. This is just common sense. If I trust Alice's signature on a key, and Alice trusts Charlie's signature on a key, that does not imply that I have to trust Charlie's signature on a key.

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 your own public key's validity. 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.

ViaCrypt PGP keeps track of which keys on your public key ring are properly certified with signatures from introducers that you trust. All you have to do is tell ViaCrypt PGP which people you trust as introducers, and certify their keys yourself with your own ultimately trusted key. ViaCrypt 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. More on this later.

Make sure no one else can tamper with your own public key ring. 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 key ring. Maintain physical control of your public key ring, preferably on your own personal computer rather than on a remote timesharing system, just as you would do for your secret key. This is to protect it from tampering, not from disclosure. Keep a trusted backup copy of your public key ring and your secret key ring 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 key ring, it is the most important key to protect from tampering. You may wish to keep a backup copy on a write-protected floppy disk.

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

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

How Does ViaCrypt PGP Keep Track of Which Keys are Valid?

Before you read this section, be sure you have read the section on "How to Protect Public Keys from Tampering".

ViaCrypt PGP keeps track of which keys on your public key ring are properly certified with signatures from introducers that you trust. All you have to do is tell ViaCrypt PGP which people you trust as introducers, and certify their keys yourself with your own ultimately trusted key. ViaCrypt 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 ViaCrypt 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?

ViaCrypt PGP can calculate the answer to the first question. To answer the second question, ViaCrypt PGP must be explicitly told by you, the user. When you supply the answer to question 2, ViaCrypt 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 ViaCrypt PGP. The keys belonging to trusted introducers must themselves be certified either by you or by other trusted introducers.

ViaCrypt 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 to ViaCrypt PGP as unknown, untrusted, marginally trusted, or completely trusted to certify other public keys. This trust information is stored on your key ring with their key, but when you tell ViaCrypt PGP to copy a key off your key ring, ViaCrypt PGP will not copy the trust information along with the key, because your private opinions on trust are regarded as confidential.

When ViaCrypt 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. ViaCrypt PGP's skepticism is adjustable - for example, you may tune ViaCrypt 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 ViaCrypt PGP, needing no introducer's signature to prove its validity. ViaCrypt PGP knows which public keys are yours, by looking for the corresponding secret keys on the secret key ring. ViaCrypt PGP also assumes you ultimately trust yourself to certify other keys.

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each 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 Government standard public key management schemes, 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. ViaCrypt PGP's decentralized probabilistic method for determining public key legitimacy is the centerpiece of its key management architecture. ViaCrypt PGP lets you alone choose who you trust, putting you at the top of your own private certification pyramid. ViaCrypt 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 ViaCrypt 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 everyone else's keys.

How to Protect Secret Keys from Disclosure

Protect your own secret key and your pass phrase carefully. Really, ***REALLY*** carefully. If your secret key is ever compromised, you'd better get the word out quickly to all interested parties (good luck) 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 secret key could expose all messages sent to you.

To protect your secret key, you can start by always keeping physical control of your secret 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 secret key rings 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 secret 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 pass phrase, and then obtain your actual secret key from the remote system. You should only use your secret key on a machine that you have physical control over.

Don't store your pass phrase anywhere on the computer that has your secret key file. Storing both the secret key and the pass phrase 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 pass phrase and the secret key file. It would be most secure if you just memorize your pass phrase and don't store it anywhere but your brain. If you feel you must write down your pass phrase, keep it well protected, perhaps even more well protected than the secret key file.

And keep backup copies of your secret key ring - remember, you have the only copy of your secret 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 ViaCrypt 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 secret key compromise. You just have to spread the word and hope everyone hears about it.

If the worst case happens - your secret key and pass phrase 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 ViaCrypt PGP to create such a certificate by using the "Revoke your own key" button on the Key Management Window. 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 ViaCrypt 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 secret/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.

Protecting Against Bogus Timestamps

A somewhat obscure vulnerability of ViaCrypt 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/secret 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.

A remedy for this could involve some trustworthy Certifying Authority or notary that would create notarized signatures with a trustworthy timestamp. This might 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. A public key certificate could be signed by the notary, and the trusted timestamp in the notary's signature would have some legal significance. The notary could enter the signed certificate into a special certificate log controlled by the notary. Anyone can read this log.

The notary could also sign other people's signatures, creating a signature certificate of a signature certificate. This would serve as a witness to the signature the same way real notaries do now with paper. Again, the notary could enter the detached signature certificate (without the actual whole document that was signed) into a log controlled by the notary. The notary's signature would have a trusted timestamp, which might have greater credibility than the timestamp in the original signature. A signature becomes "legal" if it is signed and logged by the notary.

This problem of certifying signatures with notaries and trusted timestamps warrants further discussion. This can of worms will not be fully covered here now. There is a good treatment of this topic in Denning's 1983 article in IEEE Computer (see references). There is much more detail to be worked out in these various certifying schemes. This will develop further as ViaCrypt PGP usage increases and other public key products develop their own certifying schemes.

What if you Lose Your Secret Key?

Normally, if you want to revoke your own secret key, you can use the "Revoke your own key" button to issue a revocation certificate, signed with your own secret key

But what can you do if you lose your secret key, or if your secret key is destroyed?   You can't revoke it yourself, because you must use your own secret key to revoke it, and you don't have it anymore. A future version of ViaCrypt PGP will offer a more secure means of revoking keys in these circumstances, allowing trusted introducers to certify that a public key has been revoked. But for now, you will have to get the word out through whatever informal means you can, asking users to "disable" your public key on their own individual public keyrings.

Vulnerabilities

No data security system is impenetrable. ViaCrypt 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.

Compromised Pass Phrase and Secret Key

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

Don't use obvious passwords that can be easily guessed, such as the names of your kids or spouse. If you make your pass phrase 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 pass phrase is so much better than a password. A more sophisticated attacker may have his computer scan a book of famous quotations to find your pass phrase. An easy to remember but hard to guess pass phrase can be easily constructed by some creatively nonsensical sayings or very obscure literary quotes.

For further details, see the topic "How to Protect Secret Keys from Disclosure".

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 topic "How to Protect Public Keys from Tampering".

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 key ring. Maintain physical control of both your public key ring and your secret key ring, preferably on your own personal computer rather than on a remote timesharing system. Keep a backup copy of both keyrings.

"Not Quite Deleted" Files

A 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 overwrite the original plaintext file after encryption by using the "Delete (Wipe) input file after encryption" option on the Settings Window. 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.

Even if you overwrite the plaintext data on the disk, it may still be possible for a resourceful and determined attacker to recover the data. Faint magnetic traces of the original data remain on the disk after it has been overwritten. Special sophisticated disk recovery hardware can sometimes be used to recover the data.

Viruses and Trojan Horses

An attack could involve a specially-tailored hostile computer virus or worm that might infect ViaCrypt PGP or your operating system. This hypothetical virus could be designed to capture your pass phrase or secret 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 ViaCrypt 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. ViaCrypt 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 ViaCrypt PGP that behaves like ViaCrypt 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 ViaCrypt PGP directly from ViaCrypt.

There are other ways to check ViaCrypt PGP for tampering, using digital signatures. You could use another trusted version of ViaCrypt PGP to check the signature on a suspect version of ViaCrypt PGP. But this will not help at all if your operating system is infected, nor will it detect if your original copy of ViaCrypt 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 ViaCrypt PGP executable.

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 ViaCrypt 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.

Exposure on Multi-User Systems

ViaCrypt PGP was originally designed for a single-user PC under your direct physical control. If you run ViaCrypt 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 pass phrase (or your pass phrase is simple enough to guess).

ViaCrypt 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 secret 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 ViaCrypt 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. ViaCrypt 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 ViaCrypt 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 the DES in withstanding differential cryptanalysis.

Besides, even if this algorithm has some subtle unknown weaknesses, ViaCrypt 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 ViaCrypt 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 ViaCrypt PGP. ViaCrypt PGP gives you Pretty Good Privacy.

Recommended Reading Material

1. Bruce Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley & Sons, 1994

2. Dorothy Denning, "Cryptography and Data Security", Addison-Wesley, Reading, MA 1982

3. Dorothy Denning, "Protecting Public Keys and Signature Keys", IEEE Computer, Feb. 1983

4. Martin E. Hellman, "The Mathematics of Public key Cryptography," Scientific American, Aug. 1979

5. Steven Levy, "Crypto Rebels", WIRED, May/Jun. 1993, page 54.

Other Readings:

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

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

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

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

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

ViaCrypt PGP Files

This topic describes the files copied to your hard disk when ViaCrypt PGP was installed. The changes made to your AUTOEXEC.BAT are also explained.

The SETUP program defaults to using a directory named VPGP on your hard drive as the location for ViaCrypt PGP files. In addition, if SETUP finds an existing copy of PGP.EXE, its directory will be as the default. You can specify a different directory if you wish. For the remainder of this section, this directory will be referred to as the VPGP directory.

If SETUP finds certain files, it will rename them:

If SETUP Finds:	**It Will Be Renamed:**
PGP.EXE	PGPEXE.OLD
CONFIG.TXT	CONFTXT.OLD
PGP.HLP	PGPHLP.OLD

SETUP will then copy the following files into the VPGP directory:

VPGPW.EXE
VPGPW1.DLL
VPGPW.HLP
PGP.EXE
PGP.HLP
CONFIG.TXT
KEYS.ASC
PGPSIG.ASC

The following file will be copied to your Windows Directory:

VPGPW1.PIF

If Visual Basic 3.0 is not installed on your system, the following file will be copied to your Windows Directory:

VBRUN300.DLL

If Visual Basic 3.0 is not installed on your system, the following file will be installed in your Windows/System Directory:

CMDIALOG.VBX

AUTOEXEC.BAT Changes

The VPGP directory will be added to your PATH statement.

A "SET PGPPATH=" statement will be added, pointing to the VPGP directory.

A "SET TZ=" statement will be added containing timezone information.

ViaCrypt PGP Naming Conventions

This topic explains the naming conventions used in ViaCrypt PGP.

ViaCrypt PGP follows the rules in the following table when creating files:

Input Filename	Operation Performed	Output Filename
filename.any	Encrypt/Sign	filename.pgp or filename.asc (see Note 1)
filename.any (see Note 2)	Decrypt/Verify Signature	filename
n/a	Copy (extract) key	filename.asc or filename.pgp (see Note 1)

NOTE 1: The choice of suffix will be determined by the "ASCII Armor PGP Output" setting. If this setting is ON, then the suffix will be ".ASC". If this setting is OFF, the suffix will be ".PGP".

NOTE 2: When "ASCII Armor PGP Output" is ON, and the size of the output file exceeds the length specified by ARMORLINES, ViaCrypt PGP will create 2 or more files, each containing a segment of the output . In this case the suffixes for the output files will be ".AS1", ".AS2", ".AS3", etc. When decrypting segments, it is important that the correct suffixes be retained. If you are going to e-mail the segments, you should either include the name and suffix of the segment in the 'subject' line of the e-mail message. Without the correct suffixes, it will be extremely difficult for the recipient to decrypt your file.

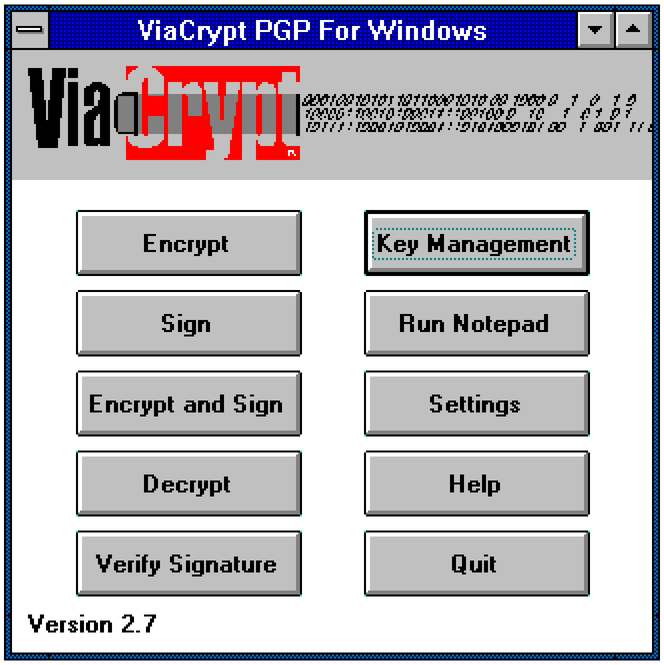
Adding Public Keys to your Keyring Using Drag and Drop

Dropping a file containing someone's public key onto ViaCrypt PGP is an easy way to add their key to your keyring. ViaCrypt PGP will perform the same actions as if you pressed the "Add key from file to your keyring" button on the Key Management Window.

The Main Window

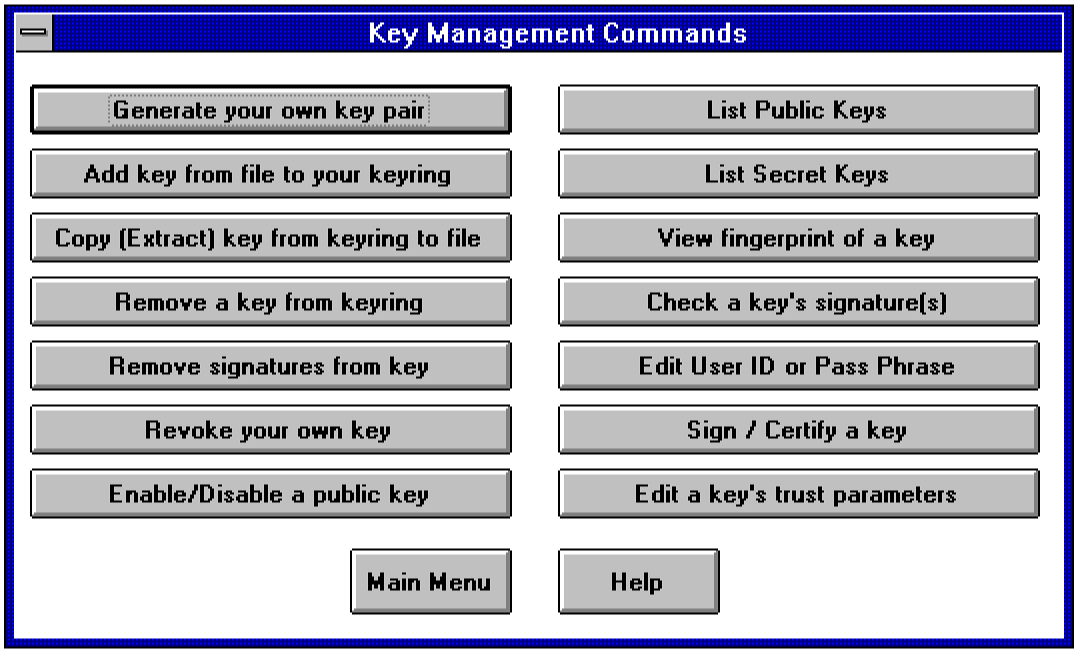
ViaCrypt PGP for Windows' Main Window is shown below.

Click on the parts of the window below where appears for descriptions of the function each button performs.



Key Management Window

The Key Management Window is shown below. Click on the parts of the window below where ☝ appears for descriptions of the function each button performs.


Key Management Commands
Generate your own key pair
List Public Keys
Add key from file to your keyring
List Secret Keys
Copy (Extract) key from keyring to file
View fingerprint of a key
Remove a key from keyring
Check a key's signature(s)
Remove signatures from key
Edit User ID or Pass Phrase
Revoke your own key
Sign / Certify a key
Enable/Disable a public key
Edit a key's trust parameters
Main Menu
Help

Setting: ASCII Armor PGP Output

Default: ON

Many electronic mail systems only allow messages made of ASCII text, not the 8-bit raw binary data that ciphertext is made of. To get around this problem, ViaCrypt PGP supports ASCII radix-64 format for encrypted files. This special format represents binary data by using only printable ASCII characters, so it is useful for transmitting binary encrypted data through 7-bit channels or for sending binary encrypted data as normal e-mail text. This format acts as a form of "transport armor", protecting it against corruption as it travels through intersystem gateways such as on Internet. It also appends a CRC (Cyclic Redundancy Check) to detect transmission errors.

Radix-64 format converts the plaintext by expanding groups of 3 binary 8-bit bytes into 4 printable ASCII characters, so the file grows by about 33%. But this expansion is acceptable when you consider that the file was compressed more than that by ViaCrypt PGP before it was encrypted.

Most Internet e-mail facilities prohibit sending messages that are more than 50,000 bytes long. Longer messages must be broken into smaller chunks that can be mailed separately. If your encrypted message is very large, and you requested radix-64 format, ViaCrypt PGP automatically breaks it up into chunks that are each small enough to send via e-mail. The chunks are put into files named with extensions ".as1", ".as2", ".as3", etc. You can specify the maximum size of these chunks using the "Armorlines" parameter in the CONFIG.TXT file.

If you want to digitally sign an ASCII text file so that the text remains readable and the digital signature appears at the bottom, the following two configuration parameters should be on:

ASCII Armor PGP Output

Use Clearsig option for digital signature

Setting: Delete (Wipe) input file after encryption

Default: OFF

When ViaCrypt PGP makes a ciphertext file for you, you can have it automatically overwrite the plaintext file and delete it. This leaves no trace of plaintext on the disk, so no one can recover it later using a disk block scanning utility. This is useful if the plaintext file contains sensitive information that you don't want to keep. The file is overwritten once with pseudo-random data, then deleted.

Obviously, you should be careful with this option. Also note that this will not wipe out any fragments of plaintext that your word processor might have created on the disk while you were editing the message before running ViaCrypt PGP. Most word processors create backup files, scratch files, or both. Also, it overwrites the file only once, which is enough to thwart conventional disk recovery efforts, but not enough to withstand a determined and sophisticated effort to recover the faint magnetic traces of the data using special disk recovery hardware.

Setting: Encrypt using IDEA only, not IDEA and RSA

Default: OFF

Sometimes you just need to encrypt a file the old-fashioned way, with conventional single-key encryption. This approach is useful for protecting archive files that will be stored but will not be sent to anyone else. Since the same person that encrypted the file will also decrypt the file, public key encryption is not really necessary.

When you encrypt a file while this option is selected, ViaCrypt PGP will prompt you for a pass phrase to use as a conventional key to encipher the file. This pass phrase need not be (and, indeed, should not be) the same pass phrase that you use to protect your own secret key. Note that ViaCrypt PGP compresses the plaintext before encrypting it.

ViaCrypt PGP will not encrypt the file the same way twice, even if you use the same pass phrase every time. For a more detailed explanation, read "Introduction to Cryptography"

Setting: Use Canonical format for Newlines on textfile

Default: ON

You may use ViaCrypt PGP to encrypt any kind of plaintext file, binary 8-bit data or ASCII text. Probably the most common usage of ViaCrypt PGP will be for e-mail, when the plaintext is ASCII text.

ASCII text is sometimes represented differently on different machines. For example, on an MS-DOS system, all lines of ASCII text are terminated with a carriage return followed by a linefeed. On a Unix system, all lines end with just a linefeed. On a Macintosh, all lines end with just a carriage return. This is a sad fact of life.

Normal unencrypted ASCII text messages are often automatically translated to some common "canonical" form when they are transmitted from one machine to another. Canonical text has a carriage return and a linefeed at the end of each line of text. For example, the popular KERMIT communication protocol can convert text to canonical form when transmitting it to another system. This gets converted back to local text line terminators by the receiving KERMIT. This makes it easy to share text files across different systems.

But encrypted text cannot be automatically converted by a communication protocol, because the plaintext is hidden by encipherment. To remedy this inconvenience, ViaCrypt PGP lets you specify that the plaintext should be treated as ASCII text (not binary data) and should be converted to canonical text form before it gets encrypted. At the receiving end, ViaCrypt PGP automatically converts the decrypted plaintext back to whatever text form is appropriate for the local environment.

This mode is automatically turned off if ViaCrypt PGP detects that the plaintext file contains what it thinks is non-text binary data.

For ViaCrypt PGP users that use non-English 8-bit character sets, when ViaCrypt PGP converts text to canonical form, it may convert data from the local character set into the LATIN1 (ISO 8859-1 Latin Alphabet 1) character set, depending on the setting of the CHARSET parameter in the ViaCrypt PGP configuration file. LATIN1 is a superset of ASCII, with extra characters added for many European languages.

Setting: Use Clearsig option for digital signature

Default: ON

When digitally signing a file containing only ASCII text, it is usually desirable for the ASCII text to appear in readable form, followed by the digital signature, like this:

```
-----BEGIN PGP SIGNED MESSAGE-----

Now is the time for all
good men to come to the
aid of their country.
-----BEGIN PGP SIGNATURE-----
Version: 2.7

iQBVAgUBLlkKnndnGF5n7PE9AQEDhQH+PPPJzXK4HvdgdlfMWVW2nWmQvpBBAIsZ
Z8TqP3/d6TrOfD1z5KCaUKjEfiY9NueWhGATr+eBL5qkyb+a/TKE0Q==
=wM76
-----END PGP SIGNATURE-----
```

To achieve this set "ASCII Armor PGP Output" and "Use Clearsig option for digital signature" ON.

If the Clearsig option is OFF, ViaCrypt PGP will compress and ASCII armor the text also, like this:

```
-----BEGIN PGP MESSAGE-----
Version: 2.7

owHrZAhlYmXUi+R5VZ4uEZf+5qMtI+O5x4z/LN5dum3AzMRzbkp2W+uLmTfeXTHb
wqfH8NaQI6CeZ4vM2r+yBQy/9+y4/9Bk47xlsV7XM5hCG93edT0w57/5luWtTMOa
2BK+ZCv91IrE3IKcVL2SihIGIPDLL1fILFYoyUhVKMnMTVVIyy9SSMzJUeDlSs/P
T1HITc1TKMlXSM4HSgFpkDJersTMFIX8NBAnswgoVZpXUlSpBwA=
=GuEy
-----END PGP MESSAGE-----
```

Preserve original plaintext filename when decrypting

Default: OFF

When decrypting files, you normally specify an output plaintext filename using the appropriate dialog box. For most files, this is a reasonable way to name the decrypted file, because you decide its name when you decrypt it.

But when ViaCrypt PGP encrypts a plaintext file, it always saves the original filename and attaches it to the plaintext before it compresses and encrypts the plaintext. Normally, this hidden original filename is discarded by ViaCrypt PGP when it decrypts, but by turning this option ON, you can tell ViaCrypt PGP you want to preserve the original plaintext filename and use it as the name of the decrypted plaintext output file. This is useful if ViaCrypt PGP is used on files whose names are important to preserve.

Setting: Enable advanced options for Encrypt/Sign/Decrypt

Default: OFF

When you turn this option ON, a special "Advanced Options" window will appear each time you encrypt, sign, or decrypt a file. The advanced options window will contain only the specific advanced options that are applicable to the operation you are performing:

Encrypt Advanced Options:

When recipient decrypts file, output it to the screen, not to a file

Signing Advanced Options:

Put signature in a separate file

Decrypt Advanced Options:

If Encrypted File Is Signed:

Output decrypted file to the screen, not to a file.

Signature Verification Advanced Options:

Save signature in a separate file.

This advanced option is available when you decrypt a file using the "Decrypt" button. If the encrypted file also has been digitally signed, the signature will automatically be checked. This is the default action.

This advanced option is available when you decrypt a file using the "Decrypt" button. If the encrypted file also has been digitally signed, the file will be decrypted only. The result will be a digitally signed, compressed file.

This advanced option is available when you decrypt a file using the "Decrypt" button. If the encrypted file also has been digitally signed, the file will be decrypted, and the digital signature will be saved in a separate file.

Setting: Background Color

The "Background Color" button can be used to customize the background color used in each window. The colors of the various 'File Open' and 'File Save As' dialog boxes cannot be customized.

NOTE: Windows requires that the background color behind text be a solid color, not a dithered color. If you select a dithered color as the background color, you may notice that the background color behind text is displayed slightly differently.

Setting: Text Color

The "Text Color" button can be used to customize the foreground color used in each window. The colors of the various 'File Open' and 'File Save As' dialog boxes cannot be customized.

NOTE: Windows requires that the background color behind text be a solid color, not a dithered color. If you select a dithered color as the background color, you may notice that the background color behind text is displayed slightly differently.

Setting: Use for This Session Only

Choose the "This Session Only" button to save your newly-selected setting for this session only. When you exit ViaCrypt PGP, the settings will return to their previous values.

Setting: Use as Defaults

Choose the "Use As Defaults" button to save your new settings as the defaults.

Setting: Cancel Button

Choose the "Cancel" button to revert to the settings in use before opening the Settings Window.

Adding a key from a file to your keyring

Choose this button on the Key Management Window to add someone's public key to your keyring.

1. Select the file containing the key in the "Select File Containing Key To Add" dialog box:

2. A window will appear on your screen, asking you if you want to certify any of the keys in the file. If you want to certify one or more of the displayed keys, press 'y' and ENTER.

If the file contains more than one key, you will go through the remaining steps once for each key:

3. ViaCrypt PGP will display a description of the key, and ask you if you want to certify it:

   If you received this key via mail or e-mail, now would be a good time to telephone the key's owner and ask them to read their key fingerprint to you over the telephone. The fingerprint read to you should match the one in displayed after "Key fingerprint = ".

4. ViaCrypt PGP will then ask if you have verified that the key actually belongs to the person specified by its User ID. If it does, then press 'y' and ENTER. ViaCrypt PGP will then sign or certify Bill's key with yours.

5. Enter your pass phrase in the dialog box so that ViaCrypt PGP can use your secret key to digitally sign Bill's key.

6. Now that you have certified Bill's key, ViaCrypt PGP will ask you if you trust Bill to certify other people's keys, becoming an introducer:

If the file contains more than one public key, then the above steps are repeated for each one added to your public keyring. If the key is already on your key ring, ViaCrypt PGP will not add it again. All of the keys in the keyfile are added to the keyring, except for duplicates. If the key being added has attached signatures certifying it, the signatures are added with the key. If the key is already on your key ring, ViaCrypt PGP just merges in any new certifying signatures for that key that you don't already have on your key ring.

Copy (Extract) a key from your keyring to a file

Choose this button on the Key Management Window to create a file containing a copy of your public key. You can also use this button to make a copy of any key on your public keyring.

When you create a file containing a public key, it typically looks like:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.7

mQBNAi5ZEk8AAAECAMaLvK2KWL9iYWSsfnXbZZ3YIW2CW1nJIS1VUgFYyTNYhYSX
MQYDvYjLtdZIVHdjmngWSfaQUcvf0H2Fz6ags5kABRG0DkpvZSBTY2htdWNrbGV5
=8iMp
-----END PGP PUBLIC KEY BLOCK-----
```

You can then use any medium of your choice (e-mail, snail-mail, sneaker-net, etc.) to send this file to the person who needs it.

1. Specify the key you want to copy in the "Select Key to Copy (Extract)" dialog box.
2. In the Save Key As dialog box, specify the name of the file that will contain your key, or accept the default
3. Next you should see a "Working" box. You may also see briefly an icon representing the cryptographic kernel near the bottom of your screen

When the "Working" box disappears, the specified file containing the selected key has been created. If the key has any certifying signatures attached to it on your key ring, they are copied along with the key. Any trust parameters attached to the key are not copied, because your private opinions on trust are regarded as confidential.

Removing a key from your keyring

Choose this button to remove a public key from your public keyring. If your secret keyring also contains a key with a matching user, you can optionally remove the key from your secret keyring also.

You should also choose this button if your key contains multiple user IDs and you want to remove one of them.

1. ViaCrypt PGP will display the "Select Key To Remove" dialog box.
2. A window will appear on your screen, asking you if you want the key removed.
3. If a secret key with a matching user ID is on your secret keyring, then ViaCrypt PGP asks you if you want it removed too.
4. ViaCrypt PGP will ask you one more time, to make really sure you want the key removed from your secret keyring (once it has been removed, there is no way to 'un-remove' it):

See Also:

Removing One User ID from your key

Removing just one User ID from your key

If a key has more than one user ID, you can choose the "Remove a key from keyring" button to remove just selected user IDs from the key. For example, suppose your key contains two user IDs reflecting two e-mail addresses:

```
Joe Schmuckley <joe@schmuckley.com>
Joe Schmuckley <12345.678@compuserve.com>
```

1. Choose the "Remove a key from keyring" button.
2. Select the key in the "Select Key To Remove" dialog box (Only the first, or primary user ID for each key is displayed in the dialog box).
3. A window will appear on your screen, asking you if you want to remove the entire key.
4. Press 'n' (or 'N') and ENTER. (**WARNING: If you press 'y', the entire key will be removed**.)
5. The key's user IDs will by displayed one at a time so that you can select the correct user ID to remove
6. Because your secret key also contains your user ID, ViaCrypt PGP repeats the process for your secret key.
7. ViaCrypt PGP will, ask you if you want to remove the entire secret key.
8. Press 'n' (or 'N') and ENTER. **(WARNING: If you press 'y', the entire key will be removed from your secret keyring.**)
9. The key's user IDs will by displayed one at a time so that you can select the correct user ID to remove

Removing signatures from a key

Choose this button to remove selected signatures from public keys. If, for example, your keyring may contain a key that you have previously signed, but you no longer want in your 'chain of introducers'. You can use this function to remove your signature from that person's key without removing the key itself.

1. ViaCrypt PGP will display the "Select Key to Remove Signatures From" dialog box.
2. A window will appear on your screen, displaying the key.   Each of the key's signatures will be displayed one-at-a-time, so you can select the specific signatures to remove.

Revoking your own key

Suppose your secret key and your pass phrase have somehow both been compromised. You have to get the word out to the rest of the world, so that they will all stop using your public key. To do this, you will have to issue a Key Compromise Certificate to revoke your public key.

Choose this button to revoke your key, and to optionally generate a Key Compromise Certificate.

1. If you have more than one secret key, you will be asked to select the key to revoke.
2. A window will appear on your screen, asking you to confirm that you want to permanently revoke your key. **After your key has been revoked, you cannot 'un-revoke' it.**
3. You'll be asked to enter your pass phrase (you are the only one who can revoke your key):
4. You should then see a message "Key compromise certificate created".

   NOTE: While the message indicates that the Key COMPROMISE Certificate has been created, in actuality your key has simply been marked on your keyring as 'revoked'. Actual generation of the key compromise certificate occurs in the next step.
5. Next, you'll see a dialog box asking if you want to optionally generate the Key Compromise Certificate. If you choose NO, your key is still marked on your keyring as revoked, but no Key Compromise Certificate is created. If you choose YES, then ViaCrypt PGP will create a file containing the Key Compromise Certificate.
6. Specify the name of the file containing the Key Compromise Certificate, or accept the default.

Visually, the Key Compromise Certificate looks identical to a key. It contains special information, though, flagging it as a Key Compromise Certificate. This certificate bears your signature, made with the same key you are revoking. You should widely disseminate this key revocation certificate as soon as possible. Other people who receive it can add it to their public keyrings, and their ViaCrypt PGP software then automatically prevents them from accidentally using your old public key ever again. You can then generate a new secret/public key pair and publish the new public key.

You may choose to revoke your key for some other reason than the compromise of a secret key. If so, you may still use the same mechanism to revoke it.

Enabling or disabling a public key

While you cannot revoke other people's keys, you can disable them. Disabling a key does not prevent you from using it, it simply warns you the key has been disabled whenever you try to use it.

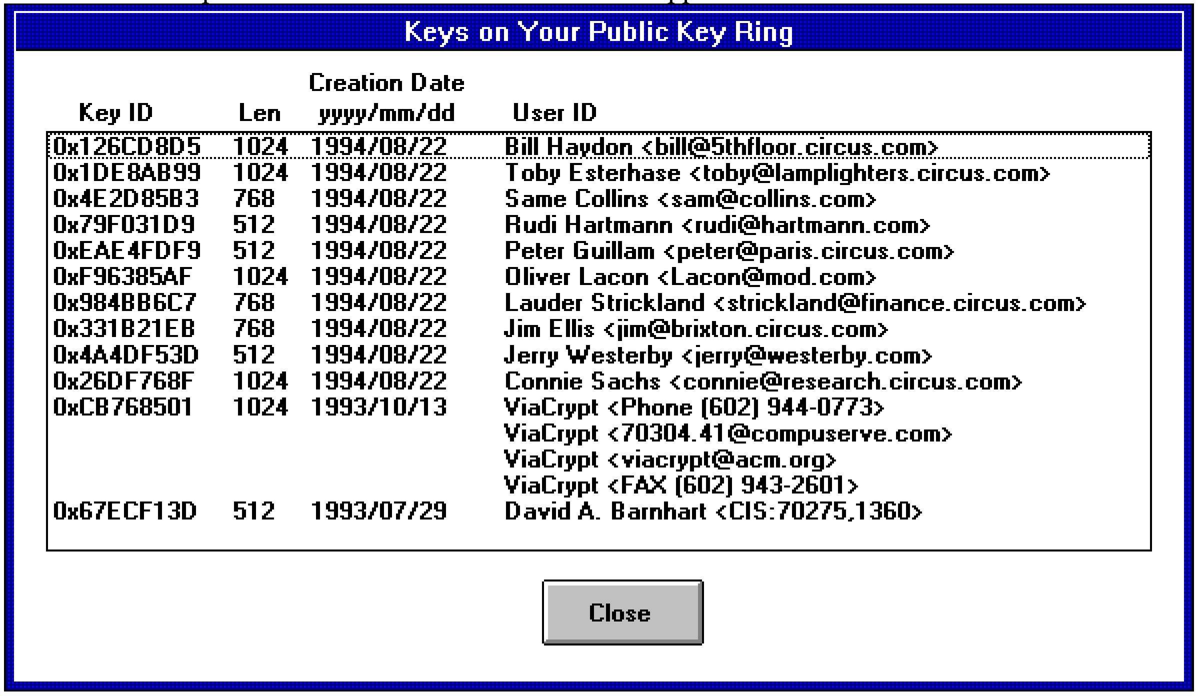
If the specified key has already been disabled, choosing this button will re-enable it.

1. ViaCrypt PGP will display the "Select Key to Disable/Enable" dialog box
2. A window will appear on your screen, asking you to verify the selection of the key to disable/enable.

Viewing keys on your public keyring

Choose this button to display the keys on your public keyring. ViaCrypt PGP will display the following dialog box containing the keys.

Click on the parts of the window below where ☝ appears for more information.

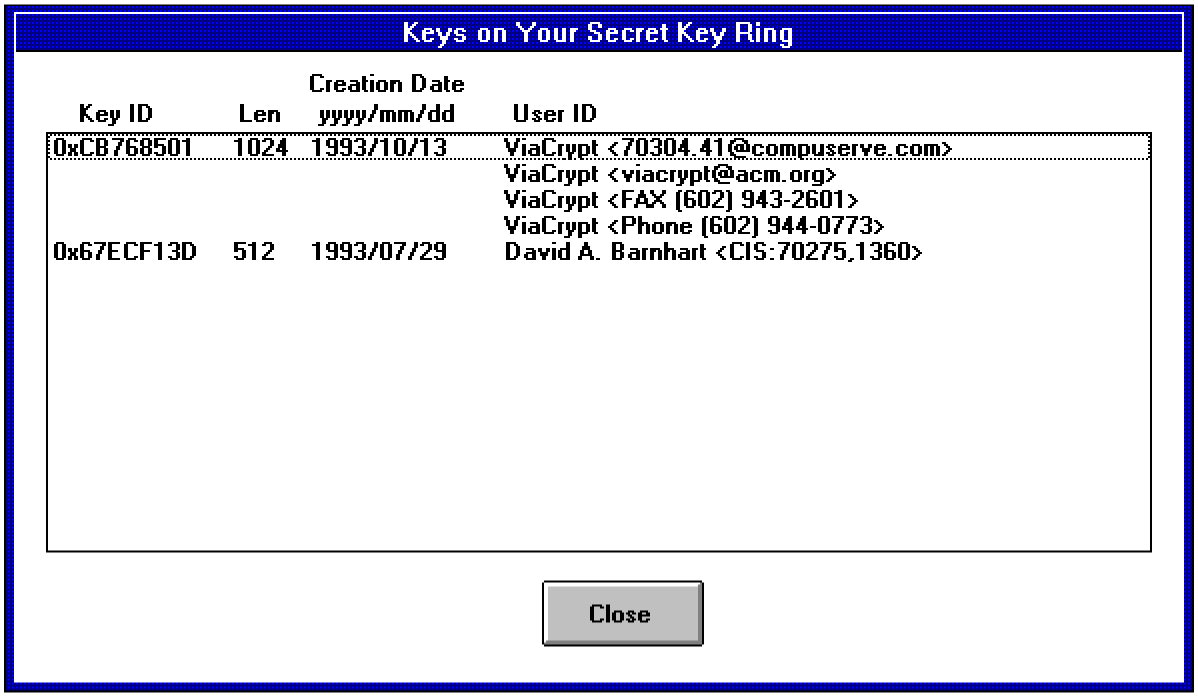


```
                    Creation Date
 Key ID       Len    yyyy/mm/dd    User ID
0x126CD8D5   1024   1994/08/22    Bill Haydon <bill@5thfloor.circus.com>
0x1DE8AB99   1024   1994/08/22    Toby Esterhase <toby@lamplighters.circus.com>
0x4E2D85B3   768    1994/08/22    Same Collins <sam@collins.com>
0x79F031D9   512    1994/08/22    Rudi Hartmann <rudi@hartmann.com>
0xEAE4FDF9   512    1994/08/22    Peter Guillam <peter@paris.circus.com>
0xF96385AF   1024   1994/08/22    Oliver Lacon <Lacon@mod.com>
0x984BB6C7   768    1994/08/22    Lauder Strickland <strickland@finance.circus.com>
0x331B21EB   768    1994/08/22    Jim Ellis <jim@brixton.circus.com>
0x4A4DF53D   512    1994/08/22    Jerry Westerby <jerry@westerby.com>
0x26DF768F   1024   1994/08/22    Connie Sachs <connie@research.circus.com>
0xCB768501   1024   1993/10/13    ViaCrypt <Phone (602) 944-0773>
                                  ViaCrypt <70304.41@compuserve.com>
                                  ViaCrypt <viacrypt@acm.org>
                                  ViaCrypt <FAX (602) 943-2601>
0x67ECF13D   512    1993/07/29    David A. Barnhart <CIS:70275,1360>
```

Viewing keys on your secret keyring

Choose this button to display the keys on your secret keyring. ViaCrypt PGP will display the following dialog box containing the keys.

Click on the parts of the window below where appears for more information.



Keys on Your Secret Key Ring

```
                        Creation Date
  Key ID        Len      yyyy/mm/dd     User ID
0xCB768501     1024     1993/10/13     ViaCrypt <70304.41@compuserve.com>
                                       ViaCrypt <viacrypt@acm.org>
                                       ViaCrypt <FAX (602) 943-2601>
                                       ViaCrypt <Phone (602) 944-0773>
0x67ECF13D     512      1993/07/29     David A. Barnhart <CIS:70275,1360>
```

Viewing the Fingerprint of a key

Chose this button to view the fingerprint of any key

1. Select the key from the "Select Key" dialog box.
2. Next you should see a "Working" box. You may also see briefly an icon representing the cryptographic kernel near the bottom of your screen
3. When the "Working" box disappears, it will be replaced with a box displaying the Key's user ID, key ID, and fingerprint.

See Also:

Verifying a public key over the phone

Verifying a public key over the phone

If you get a public key from someone that is not certified by anyone you trust, how can you tell if it's really their key?   The best way to verify an uncertified key is to verify it over some independent channel other than the one you received the key through. One convenient way to tell, if you know this person and would recognize them on the phone, is to call them and verify their key over the telephone. Rather than reading their whole tiresome (ASCII-armored) key to them over the phone, you can just read their key's "fingerprint" to them.

The key's fingerprint is a 16-byte digest of the public key components. Read this 16-byte fingerprint to the key's owner on the phone, while they check it against their own. You can both verify each other's keys this way, and then you can sign each other's keys with confidence. This is a safe and convenient way to get the key trust network started for your circle of colleagues.

Checking a key's signatures

Choose this button to display a list of a key's digital signatures. This is a quick and easy way to see who has signed/certified a key.

1. Select the key from the "Select Key to Check" dialog box:
2. Next you should see a "Working" box. You may also see briefly an icon representing the cryptographic kernel near the bottom of your screen
3. When the "Working" box disappears, it will be replaced with a box displaying the key and its signatures.
4. Choose the Close button to dismiss the dialog box.

Editing your User ID or Pass Phrase

Sometimes you may need to change your pass phrase, perhaps because someone looked over your shoulder while you typed it in. Perhaps you may need to change your user ID, because you got married and changed your name, or maybe you changed your e-mail address. Maybe you want to add a second or third user ID to your key, because you may be known by more than one name, e-mail address, job title, or other unique identifier.

Choose the "Edit User ID or Pass Phrase " button to:

Change your pass phrase

Change your user ID

Add additional user IDs to your key

See Also:

Removing One User ID from your key

Changing your pass phrase

1. If you have more than one secret key, ViaCrypt PGP will ask you to select the key to edit
2. A dialog box will appear on your screen, asking you to enter your pass phrase. Type your pass phrase and press ENTER.
3. Next, you will see the prompt asking you if you want to **add a user ID**. Type 'n' and press ENTER.
4. ViaCrypt PGP will ask you if you want to change your pass phrase. Type 'y' and press ENTER.
5. A dialog box will appear, prompting for your new pass phrase.

Changing your User ID

1. If you have more than one secret key, ViaCrypt PGP will ask you to select the key to edit
2. A dialog box will appear on your screen, asking you to enter your pass phrase. Type your pass phrase and press ENTER.
3. Next, you will see the prompt asking you if you want to **add a user ID**. Type 'y' and press ENTER.
4. ViaCrypt PGP will prompt you for the new user ID. Enter your user ID the way you now want it to appear, and press ENTER.
5. ViaCrypt PGP will ask if you want this new user ID to be the primary user ID. Type 'y' and press ENTER.
6. Since you don't want to change your pass phrase, enter 'n' at the prompt.

Adding an additional User ID to your key

1. If you have more than one secret key, ViaCrypt PGP will ask you to select the key to edit
2. A dialog box will appear on your screen, asking you to enter your pass phrase. Type your pass phrase and press ENTER.
3. Next, you will see the prompt asking you if you want to **add a user ID**. Type 'y' and press ENTER.
4. ViaCrypt PGP will prompt you for the new user ID. Enter your user ID the way you now want it to appear, and press ENTER.
5. ViaCrypt PGP will ask if you want this new user ID to be the primary user ID. Type 'n' and press ENTER.
6. Since you don't want to change your pass phrase, enter 'n' at the prompt.

The new user ID will be added to your key below the existing ones.

Signing / Cerifying someone's key

When you digitally sign, or certify, someone's key you are performing an important service. You are publicly stating that you have verified the correspondence between the key and its owner.

If you are using ViaCrypt PGP for personal use, you would certify the keys of people whom you trust as introducers, establishing your own 'pyramid of trust' with you at the top of the pyramid. If you are using ViaCrypt PGP in a commercial setting, then you probably want to appoint someone, such as the security officer, to be the corporate certifying authority. In this setting you would sign the certifying authority's key. The certifying authority would, in turn, certify everyone's public key.

1. ViaCrypt PGP will display the "Select Key to Sign" dialog box. From it, select the public key you want to certify:
2. When you certify someone's public key, you are actually digitally signing it with your secret key. If your secret keyring contains more than one key, then you will be asked to select the secret key to use:
3. Awindow will appear on your screen, and you asked to verify that the key actually belongs to the user specified by the user ID. Press 'y' and ENTER.
4. Enter your pass phrase in the dialog box.
5. Next, you are asked to make your own personal assessment of your confidence in Bill's ability to faithfully follow the rules when certifying other people's keys:

Your trust for a key's owner to act as an introducer does not just reflect your estimation of their personal integrity. Your response ("I don't know", "No", "Usually", or "Yes") reflects how competent you think they are at understanding key management and using good judgment in signing keys.

You can edit this value later.

Editing a key's trust parameters

Choose this button to edit a key's trust parameters.

1. ViaCrypt PGP will display the "Select Key to Edit" dialog box. From it, select the public key whose trust parameters you want to edit:
2. A window will appear on your screen, asking you to specify your level of trust un the key's owner as an introducer.

Your trust for a key's owner to act as an introducer does not just reflect your estimation of their personal integrity. Your response ("I don't know", "No", "Usually", or "Yes") reflects how competent you think they are at understanding key management and using good judgment in signing keys.

Generating your own key pair

Choose this button to generate your own public/secret key pair. Before anyone can send you an encrypted message, you'll need to generate your own public/secret key pair, and provide the sender with a copy of your public key   Your secret key is also needed when you digitally sign a file.

1. A window will open on your screen, and in it ViaCrypt PGP will prompt you for a key size, suggesting some good defaults:

2. Next, you are asked for a user ID.

3. Next, ViaCrypt PGP will ask for a pass phrase used to protect your secret key. No one can use your secret key without this pass phrase. The pass phrase is like a password, except that it can be a whole phrase or sentence with many words, spaces, punctuation, or anything else you can type:

   Do not lose this pass phrase. There is no way to recover it if you lose it. If you lose it, you will no longer be able to use this key to decrypt your messages, or digitally sign messages you send. **If you forget it, forget it!**

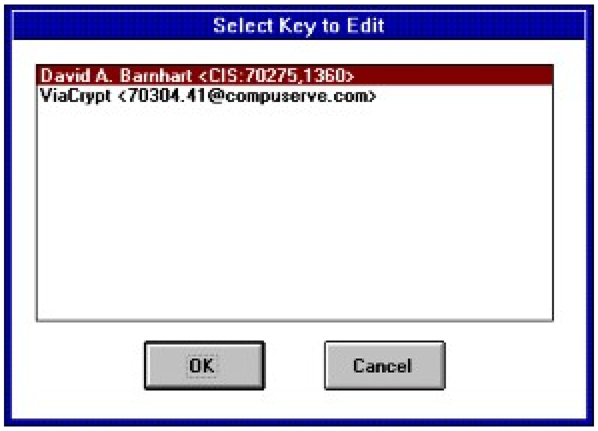
4. ViaCrypt PGP will next ask you to enter some random keystrokes to help it accumulate some random bits for your keys:

5. Finally, ViaCrypt PGP will begin generating your key pair. This a lengthy process, and may take several seconds. If your PC is a slower one, it may even take a few minutes. During this process, you should see something like this on your screen. Don't worry if the display is not exactly like the on shown:

When ViaCrypt PGP generates your first key pair, it will create a pair of files, called keyrings, to hold your keys. Secret keys are on your secret keyring, a file typically named secring.pgp. Public keys are on your public keyring, a file typically named pubring.pgp.

Never give your secret key or pass phrase to anyone else. For the same reason, don't make key pairs for your friends. Everyone should make their own key pair. Always keep physical control of your secret key, and don't risk exposing it by storing it on a remote timesharing computer. Keep it on your own personal computer.

You should also digitally sign, or certify your own public key. By using your secret key to digitally signing your public key, you protect against someone undetectably tampering with your public key.



David A. Barnhart <CIS:70275,1360>
ViaCrypt <70304.41@compuserve.com>

COMMENT

If you specify that ViaCrypt PGP's output is to be ASCII armored, you can add a comment line to the output. For example, if your CONFIG.TXT contained the line:

```
COMMENT=This is a comment
```

Then ViaCrypt PGP's output would appear as:

```
-----BEGIN PGP MESSAGE-----
Version: 2.7
Comment: This is a comment

hEwCd2cYXmfs8T0BAf9QGGniB6rbyAX8Pczh2WkAwn95WUI6ZJ58zQV7pfRQkh/T
bolt/wl9egTrCpDhszBs6J1xRf8cdC08cA3JSzXTpgAAAF8RrysX7XuA9hpbBXVC
O3AqcWnSsH4FT24rEZ/1XWwFPCsbLxVmLth/uSX9CLOHYR7r4VdWxQjUdvmRh45a
zG5oPJotHqMfvVfsExkOr4mkNpssVvM9wasw0GBYADgoOQ==
=YfBN
-----END PGP MESSAGE-----
```

If you specify "+comment=comment" on the command line and your comment includes spaces, enclose the entire comment in double quotes.

KEEPBINARY

Default setting: KEEPBINARY = off

When ViaCrypt PGP reads a ".asc" file, it recognizes that the file is in radix-64 format and will convert it back to binary before processing as it normally does, producing as a by-product a "ViaCrypt PGP" ciphertext file in binary form. After further processing to decrypt the ".pgp" file, the final output file will be in normal plaintext form.

You may want to delete the binary ".pgp" intermediate file, or you may want ViaCrypt PGP to delete it for you automatically. You can still rerun ViaCrypt PGP on the original ".asc" file.

The configuration parameter KEEPBINARY enables or disables keeping the intermediate ".pgp" file during decryption.

CERT_DEPTH

Default setting: CERT_DEPTH = 4

The configuration parameter CERT_DEPTH specifies how many levels deep you may nest introducers to certify other introducers to certify public keys on your public key ring. For example, If CERT_DEPTH is set to 1, there may only be one layer of introducers below your own ultimately-trusted key. If that were the case, you would be required to directly certify the public keys of all trusted introducers on your key ring. If you set CERT_DEPTH to 0, you could have no introducers at all, and you would have to directly certify each and every key on your public key ring in order to use it. The minimum CERT_DEPTH is 0, the maximum is 8.

For further details, see "How Does ViaCrypt PGP Keep Track of Which Keys are Valid?"

LANGUAGE

Default setting: LANGUAGE = "en"

ViaCrypt PGP displays various prompts, warning messages, and advisories to the user on the screen. For example, messages such as "File not found.", or "Please enter your pass phrase:". These messages are normally in English. But in future releases it will be possible to get ViaCrypt PGP to display its messages to the user in other languages, without having to modify the ViaCrypt PGP executable program.

CLEARSIG

Default setting: CLEARSIG = on

Normally, unencrypted ViaCrypt PGP signed messages have a signature certificate prepended in binary form. To send this through a 7-bit e-mail channel, radix-64 ASCII armor is applied , rendering the message unreadable to casual human eyes, even though the message is not actually encrypted. The recipient must use ViaCrypt PGP to strip the armor off before reading the message.

If the original plaintext message is in text (not binary) form, there is a way to send it through an e-mail channel in such a way that the ASCII armor is applied only to the binary signature certificate, but not to the plaintext message. This makes it possible to read the signed message with human eyes, without the aid of ViaCrypt PGP. Of course, you still need ViaCrypt PGP to actually check the signature.

This option is provided for in the Settings Window. It should not be set or altered here.

This message representation is analogous to the MIC-CLEAR message type used in Internet Privacy Enhanced Mail (PEM). It is important to note that since this method only applies ASCII armor to the binary signature certificate, and not to the message text itself, there is some risk that the unarmored message may suffer some accidental molestation while en route. This can happen if it passes through some e-mail gateway that performs character set conversions, or in some cases extra spaces may be added to or stripped from the ends of lines. If this occurs, the signature will fail to verify, which may give a false indication of intentional tampering. But since PEM lives under a similar vulnerability, it seems worthwhile having this feature despite the risks.

Trailing blanks are ignored on each line in calculating the signature for text in CLEARSIG mode.

INTERACTIVE

Default Setting: INTERACTIVE = off

Enabling this mode will mean that if you add a key file containing multiple keys to your key ring, ViaCrypt PGP will ask for confirmation for each key before adding it to your key ring.

RANDSEED

If you want to specify a non-standard location for your random number seed file (randseed.bin, add a line to your CONFIG.TXT file specifying its full path and filename, like this:

```
randseed=/usr/dab/pgp/myrandseed.bin
```

This is useful when you need to keep randseed.bin on a floppy disk instead of on a hard disk.

VERBOSE

Default setting: VERBOSE = 1

VERBOSE may be set to 0, 1, or 2, depending on how much detail you want to see from ViaCrypt PGP diagnostic messages. The settings are:

0	Display messages only if there is a problem.
1	Normal default setting. Displays a reasonable amount of detail in diagnostic or advisory messages.
2	Displays maximum information, usually to help diagnose problems in ViaCrypt PGP. Not recommended for normal use. Besides, ViaCrypt PGP doesn't have any problems, right?

COMPLETES_NEEDED

COMPLETES_NEEDED - Number of Completely Trusted Introducers Needed

Default setting: COMPLETES_NEEDED = 1

The configuration parameter COMPLETES_NEEDED specifies the minimum number of completely trusted introducers required to fully certify a public key on your public key ring. This gives you a way of tuning ViaCrypt PGP's skepticism.

For further details, see "How Does ViaCrypt PGP Keep Track of   Which Keys are Valid?"

PAGER

Default setting: PAGER = ""

ViaCrypt PGP lets you view the decrypted plaintext output on your screen (like the Unix-style "more" command), without writing it to a file using the "Output decrypted file to the screen, not to a file" advanced option.

If you prefer to use a fancier page display utility, rather than ViaCrypt PGP's built-in one, you can specify the name of a shell command that ViaCrypt PGP will invoke to display your plaintext output file. The configuration parameter PAGER specifies the shell command to invoke to display the file. For example,   you might want to use the popular MS-DOS shareware program "list.com" to display your plaintext message. Assuming you have a copy of "list.com", you may set PAGER accordingly:

```
PAGER = "list"
```

However, if the sender specified that this file is for your eyes only, and may not be written to disk, ViaCrypt PGP always uses its own built-in display function.

COMPRESS

Default setting: COMPRESS = on

The configuration parameter COMPRESS enables or disables data compression before encryption. Normally, ViaCrypt PGP attempts to compress the plaintext before it encrypts it. You may, however, choose to use some external compression routine with ViaCrypt PGP.

ViaCrypt PGP will recognize the compression used by most of the commonly available compression programs, and skip its own compression step. In some cases, however, the 'signature' of your external compression program may not be recognizable by ViaCrypt PGP. In that case, set "COMPRESS=OFF" to disable ViaCrypt PGP's compression.

SECRING

If you want to specify a non-standard location for your secret, add a line to your CONFIG.TXT file specifying its full path and filename, like this:

```
secring=/usr/dab/pgp/mysecring.pgp
```

This is useful when you want to keep your secret keyring on a floppy disk instead of on a hard disk, for example.

ARMORLINES

Default setting: ARMORLINES = 720

When ViaCrypt PGP creates a very large ".asc" radix-64 file for sending ciphertext or keys through the e-mail, it breaks the file up into separate chunks small enough to send through Internet mail utilities. Normally, Internet mailers prohibit files larger than about 50,000 bytes, which means that if we restrict the number of lines to about 720, we'll be well within the limit. The file chunks are named with suffixes ".as1", ".as2", ".as3", ...

The configuration parameter ARMORLINES specifies the maximum number of lines to make each chunk in a multipart ".asc" file sequence. If you set it to zero, ViaCrypt PGP will not break up the file into chunks.

Fidonet e-mail files usually have an upper limit of about 32,767 bytes, so 450 lines would be appropriate for Fidonet environments.

ENCRYPTTOSELF

Default setting: NO

This is a convenient way to keep copies of files and messages you send. By setting "ENCRYPTTOSELF=YES", the file will automatically be encrypted under your public key in addition to the public key of the recipient you select. If your secret keyring contains more than one key, you must use the MYNAME parameter to specify the key to use.

MYNAME

Default setting: MYNAME = ""

The configuration parameter MYNAME specifies the default user ID to use to select the secret key for making signatures. If MYNAME is not defined, the most recent secret key you installed on your secret key ring will be used.

Since ViaCrypt PGP for Windows always prompts for the secret key if there is more than one, the MYNAME parameter is seldom used. However, if you want to use ENCRYPTTOSELF and have more than one secret key, you must define MYNAME.

TZFIX

Default setting: TZFIX = 0

ViaCrypt PGP provides timestamps for keys and signature certificates in Greenwich Mean Time (GMT), or Coordinated Universal Time (UTC), which means the same thing for our purposes. When ViaCrypt PGP asks the system for the time of day, the system is supposed to provide it in GMT.

But sometimes, because of improperly configured MS-DOS systems, the system time is returned in US Pacific Standard Time plus 8 hours. Sounds weird, doesn't it?   Perhaps because of some sort of US west-coast jingoism, MS-DOS presumes local time is US Pacific time, and pre-corrects Pacific time to GMT. This adversely affects the behavior of the internal MS-DOS GMT time function that ViaCrypt PGP calls. However, if your MS-DOS environmental variable TZ is already properly defined for your timezone, this corrects the misconception MS-DOS has that the whole world lives on the US west coast.

The configuration parameter TZFIX specifies the number of hours to add to the system time function to get GMT, for GMT timestamps on keys and signatures. If the MS-DOS environmental variable TZ is defined properly, you can leave TZFIX=0. Unix systems usually shouldn't need to worry about setting TZFIX at all. If you are using some other obscure operating system that doesn't know about GMT, you may have to use TZFIX to adjust the system time to GMT.

On MS-DOS systems that do not have TZ defined in the environment, you should make TZFIX=0 for California, -1 for Colorado, -2 for Chicago, -3 for New York, -8 for London, -9 for Amsterdam. In the summer, TZFIX should be manually decremented from these values to account for Daylight Savings Time.

It would be much cleaner to set your MS-DOS environmental variable TZ in your AUTOEXEC.BAT file, and not use the TZFIX correction. Then MS-DOS gives you good GMT timestamps, and will handle daylight savings time adjustments for you. Here are some sample lines to insert into AUTOEXEC.BAT, depending on your time zone:

For Los Angeles:	SET TZ=PST8PDT	
For Denver:	SET TZ=MST7MDT	
For Arizona:	SET TZ=MST7	(Arizona never uses daylight savings time)
For Chicago:	SET TZ=CST6CDT	
For New York:	SET TZ=EST5EDT	

PUBRING

If you want to specify a non-standard location for your public keyring, add a line to your CONFIG.TXT file specifying its full path and filename, like this:

```
pubring=/usr/dab/pgp/mypubring.pgp
```

This is useful when you want to keep a department-wide or company-wide keyring on a LAN server.

TMP

Default setting: TMP = ""

The configuration parameter TMP specifies what directory to use for ViaCrypt PGP's temporary scratch files. The best place to put them is on a RAM disk, if you have one. That speeds things up quite a bit, and increases security somewhat. If TMP is undefined, the temporary files go in the current directory. If the shell environmental variable TMP is defined, ViaCrypt PGP instead uses that to specify where the temporary files should go.

MARGINALS_NEEDED

Default setting: MARGINALS_NEEDED = 2

The configuration parameter MARGINALS_NEEDED specifies the minimum number of marginally trusted introducers required to fully certify a public key on your public key ring. This gives you a way of tuning ViaCrypt PGP's skepticism.

For further details, see "How Does ViaCrypt PGP Keep Track of Which Keys are Valid?"

ARMOR

Default setting: ARMOR = off

This configuration parameter causes ViaCrypt PGP to emit ciphertext or keys in ASCII Radix-64 format suitable for transporting through e-mail channels. Output files are named with the ".asc" extension.

The ARMOR configuration parameter is provided for in the Settings Window, and should not be set here.

TEXTMODE

Default setting: TEXTMODE = off

If enabled, this parameter causes ViaCrypt PGP to assume the plaintext is a text file, not a binary file, and converts it to "canonical text" before encrypting it. Canonical text has a carriage return and a linefeed at the end of each line of text.

This feature is normally provided for in the Settings Window, and should not be altered in the CONFIG.TXT file.

This mode will be automatically turned off if ViaCrypt PGP detects that the plaintext file contains what it thinks is non-text binary data.

For further details, see   "Sending ASCII Text Files Across Different Machine Environments".

PKCS_COMPAT

Default Setting: PKCS_COMPAT=1

In May of 1994, the Massachusetts Institute of Technology (MIT) began distribution of a PGP version 2.6. MIT-PGP 2.6 is licensed for personal, non-commercial use along with other restrictions. After September 1, 1994, messages created by MIT-PGP 2.6 are unreadable by earlier versions of PGP, including ViaCrypt's previous version 2.4. This version (2.7) is designed to maintain compatibility with ViaCrypt's previous version (2.4), as well as MIT-PGP 2.6.

ViaCrypt PGP Version 2.7 is able to correctly read and understand messages, signatures, and keys generated by ViaCrypt PGP Version 2.4 and 2.7, as well as MIT-PGP 2.6.

By using the PKCS_COMPAT parameter either on the command line or in the CONFIG.TXT file, ViaCrypt PGP Version 2.7 can generate output that is understandable by ViaCrypt PGP 2.4, MIT-PGP 2.6, or both.

PKCS_COMPAT=1

Until September 1, 1994, ViaCrypt PGP version 2.7 generated output that is understandable by both ViaCrypt PGP 2.4, ViaCrypt PGP 2.7, and by MIT-PGP 2.6.

After September 1, 1994, ViaCrypt PGP version 2.7 generates output that is readable by ViaCrypt PGP 2.7 and MIT-PGP 2.6 only.

ViaCrypt PGP version 2.7 is able to understand output generated by ViaCrypt PGP 2.4 and 2.7, as well as MIT-PGP 2.6 with no date restrictions.

PKCS_COMPAT=2

ViaCrypt PGP version 2.7 will generate output that is understandable by ViaCrypt PGP 2.4 and 2.7, as well as MIT-PGP 2.6 with no date restrictions.

ViaCrypt PGP version 2.7 is able to understand output generated by ViaCrypt PGP 2.4 and 2.7, as well as MIT-PGP 2.6 with no date restrictions.

CHARSET

Default setting: CHARSET = NOCONV

Because ViaCrypt PGP must process messages in many non-English languages with non-ASCII character sets, you may have a need to tell ViaCrypt PGP what local character set your machine uses. This determines what character conversions are performed when converting plaintext files to and from canonical text format. This is only a concern if you are in a non-English non-ASCII environment.

The configuration parameter CHARSET selects the local character set. The choices are NOCONV (no conversion), LATIN1 (ISO 8859-1 Latin Alphabet 1), KOI8 (used by most Russian Unix systems), ALT_CODES (used by Russian MS-DOS systems), ASCII, and CP850 (used by most western European languages on standard MS-DOS PCs).

LATIN1 is the internal representation used by ViaCrypt PGP for canonical text, so if you select LATIN1, no conversion is done. Note also that ViaCrypt PGP treats KOI8 as LATIN1, even though it is a completely different character set (Russian), because trying to convert KOI8 to either LATIN1 or CP850 would be futile anyway. This means that setting CHARSET to NOCONV, LATIN1, or KOI8 are all equivalent to ViaCrypt PGP.

If you use MS-DOS and expect to send or receive traffic in western European languages, set CHARSET = "CP850". This will make ViaCrypt PGP convert incoming canonical text messages from LATIN1 to CP850 after decryption.

For further details, see "Sending ASCII Text Files Across Different Machine Environments".

SHOWPASS

Default setting: SHOWPASS = off

Normally, ViaCrypt PGP does not let you see your pass phrase as you type it in. This makes it harder for someone to look over your shoulder while you type and learn your pass phrase. But some typing-impaired people have problems typing their pass phrase without seeing what they are typing, and they may be typing in the privacy of their own homes.

The configuration parameter SHOWPASS enables ViaCrypt PGP to echo your typing during pass phrase entry.

Privacy means that only those intended to receive a message can read it.

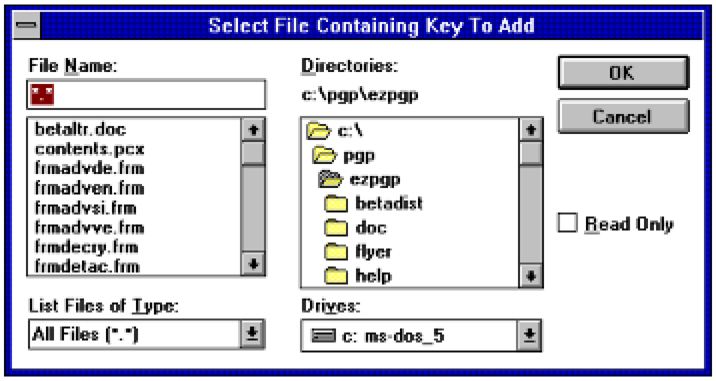
Authentication means that messages that appear to be from a particular person can only have originated from that person.

Convenience means that privacy and authentication are provided without the hassles of managing keys associated with conventional cryptographic software. No secure channels are needed to exchange keys between users, which makes ViaCrypt PGP easy to use. This is because ViaCrypt PGP is based on a powerful new technology called ***public key cryptography***.

It is not necessary to understand anything about cryptographic keys in order to use them, but a short introduction to the concepts will make other things clearer.

In public key cryptosystems, like that used in ViaCrypt PGP, everyone has two different but related, complimentary keys. The two keys are the public key and the secret key. Each key unlocks the code that the other key makes. You would normally distribute your public key to everyone you want to communicate with securely. You should never, however reveal your secret key.

When someone wants to send you an encrypted message, their copy of ViaCrypt PGP will encrypt the message using your public key. When you receive the message, your copy of ViaCrypt PGP will decrypt it using your secret key. Since you are the only one who has your secret key, no one can decrypt it except you.



If the file containing the key is on a different drive, select the drive you want from the Drives box. In the directories box, choose the directory you want to view (Double-click the directory, or press the UP ARROW or DOWN ARROW key to select the directory, then press ENTER). From the list of files, select the file containing the key you want to add to your keyring.

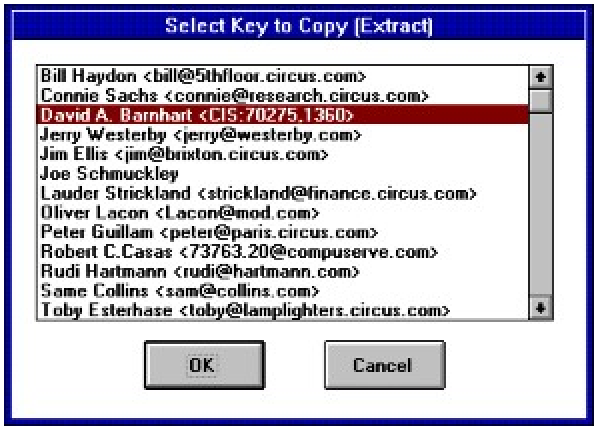
Over the years a convention has developed for the syntax for CompuServe e-mail addresses in ViaCrypt PGP user IDs. Replace the comma (',') with a period ('.') and append @compuserve.com to it. For example, a person named "James R. Prideaux" with a CompuServe e-mail address of 12345,678 would specify his user ID as:

```
James R. Prideaux <12345.678@compuserve.com>
```

Why make a copy of your public key? . The concept is similar to e-mail: If someone wants to send you an e-mail message, they will need your public key to do so. Similarly, if someone wants to send you an encrypted file, they will need your public key to do so. In addition, anyone who wants to verify your digital signature will also need your public key.

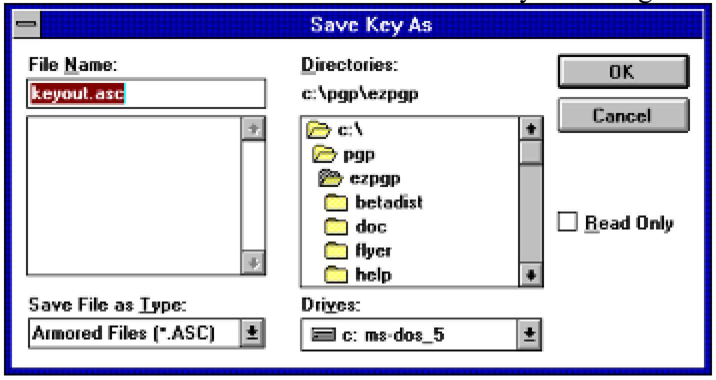
Case-sensitive means that meaning that "JACK AND THE BEANSTALK" is not the same as "Jack and the Beanstalk".

The "Working" message box appears while the cryptographic kernel is running.



Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275.1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Joe Schmuckley
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Robert C.Casas <73763.20@compuserve.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>

The Save Key As dialog box appears below:



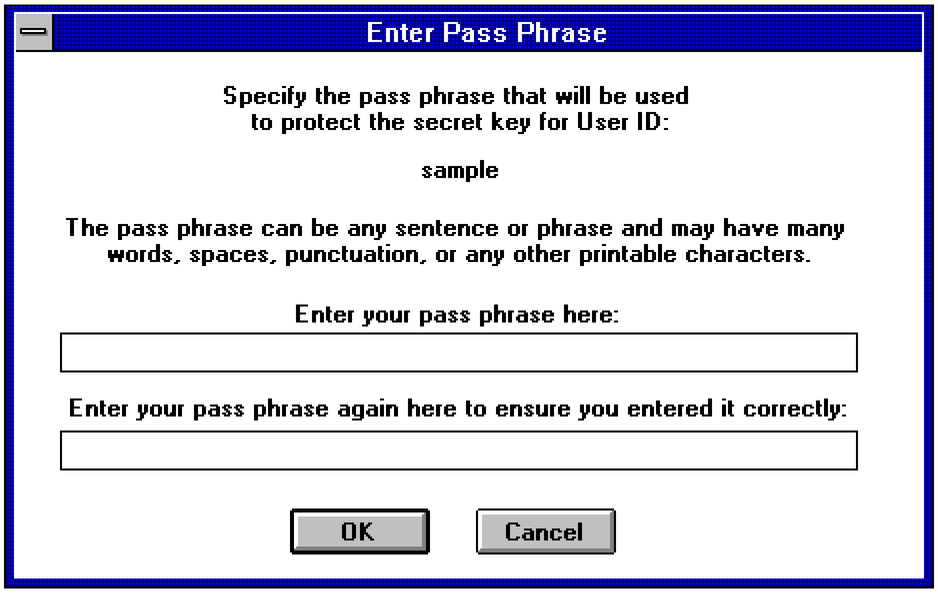
If you want to save the file on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory in which you want to save the encrypted file. In the File Name box, type a name for the file. While the default extension for the file's name will be .ASC or .PGP, the filename you specify can have any extension you want. Choose the OK button.

```
Pick your RSA key size:
    1)   512 bits- Low commercial grade, fast but less secure
    2)   768 bits- High commercial grade, medium speed, good security
    3)  1024 bits- "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits (up to 2048):
```

We suggest you choose option 3, a 1024-bit key. Even though it says "slow", today's higher performance PCs mean the difference is not very noticeable, and the added security is worth it.

```
Generating an RSA key with a 1024-bit modulus...
You need a user ID for your public key. The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example:  John Q. Smith <12345.6789@compuserve.com>

Enter a user ID for your public key:
```

It's a good idea to use your full name as your user ID, because there is a greater assurance that it is unique. Spaces and punctuation are allowed in your user ID. You should also put your e-mail address, CompuServe ID, or some other unique identifier in <angle brackets> after your name.


Enter Pass Phrase

Specify the pass phrase that will be used
to protect the secret key for User ID:

sample

The pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

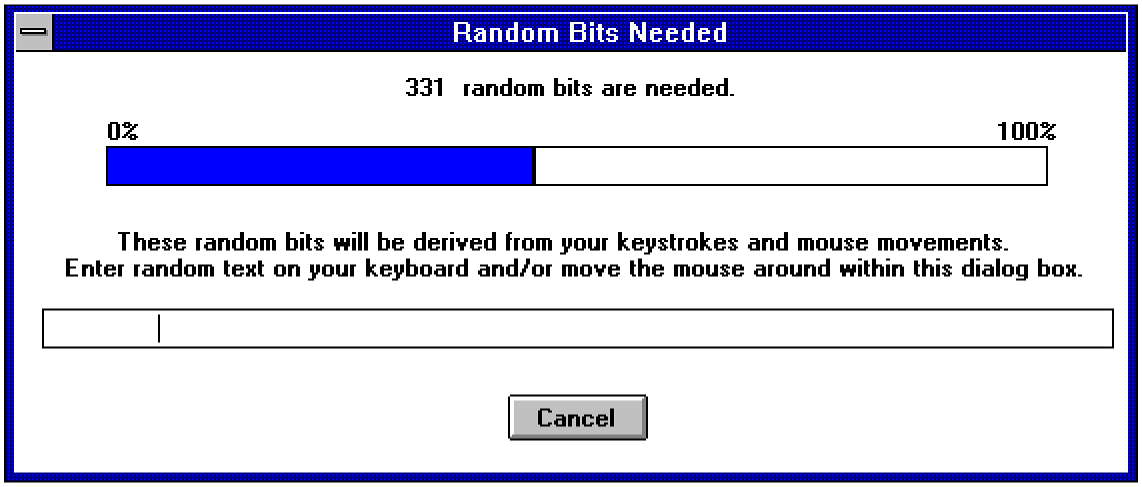
Enter your pass phrase here:

Enter your pass phrase again here to ensure you entered it correctly:

This pass phrase is case-sensitive, This pass phrase will be needed each time you want to decrypt a received message, or digitally sign a message you are sending. Your pass phrase should be something that is easy for you to remember, but impossible for someone else to guess. To continue the above example, James Prideaux might choose the pass phrase:

```
Percy eats unpeeled bananas
```

(Of course, we strongly recommend that you do not use this same pass phrase. Anyone trying to guess your pass phrase will try this one as one of their first guesses.)


Random Bits Needed

331 random bits are needed.

These random bits will be derived from your keystrokes and mouse movements.
Enter random text on your keyboard and/or move the mouse around within this dialog box.

Your key pair is derived from large truly random numbers that are in turn derived mainly from your mouse movements and measuring the intervals between your keystrokes. You should enter some keystrokes that are reasonably random in their timing, and move the mouse around. Don't simply hold down a single key and let the repeat function do the work for you.

While your key is being generated, ViaCrypt PGP will display a string of characters indicating its progress:

```
........................................................................
................................................++++!...................
.............++++!EK
Key generation completed.
```

Each character has meaning:

.	Number tested is not a prime number.
+	Number tested may be a prime number.
!	Number tested is a prime number.
E	Trying an exponent.
K	Generating the components of your key.

Why add someone's public key to your keyring? The concept is similar to e-mail: If you want to send someone an e-mail message, you will need their e-mail address. Similarly, in order to send someone an encrypted file or message, you will need their public key.

When you receive a file containing someone's public key, it typically looks like:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.7

mQBNAi5ZEk8AAAECAMaLvK2KWL9iYWSsfnXbZZ3YIW2CW1nJIS1VUgFYyTNYhYSX
MQYDvYjLtdZIVHdjmngWSfaQUcvf0H2Fz6ags5kABRG0DkpvZSBTY2htdWNrbGV5
=8iMp
-----END PGP PUBLIC KEY BLOCK-----
```

You must add the key in this file to your public keyring in order to use it.

The key does not need to be isolated into a file by itself. The file can contain other ASCII text before and after the public key block, and ViaCrypt PGP will find the key block.

```
Looking for new keys...
pub  1024/126CD8D5 1994/08/22  Bill Haydon <bill@5thfloor.circus.com>
Checking signatures...


Keyfile contains:
   1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)?
```

The question "Do you want to certify any of these keys yourself?" is an important one. When you certify a key, you are certifying the correspondence between the key and its owner.

```
Key for user ID: Bill Haydon <bill@5thfloor.circus.com>
1024-bit key, Key ID 126CD8D5, created 1994/08/22
Key fingerprint =  A9 71 27 D7 C7 15 1A 57  4E CD 87 72 CA 0B 5D DE
This key/userID association is not certified.

Do you want to certify this key yourself (y/N)?
```

When you certify a key, you are certifying the relationship between the key and its owner. Using the above example, you would be certifying that this key really belongs to Bill Haydon.

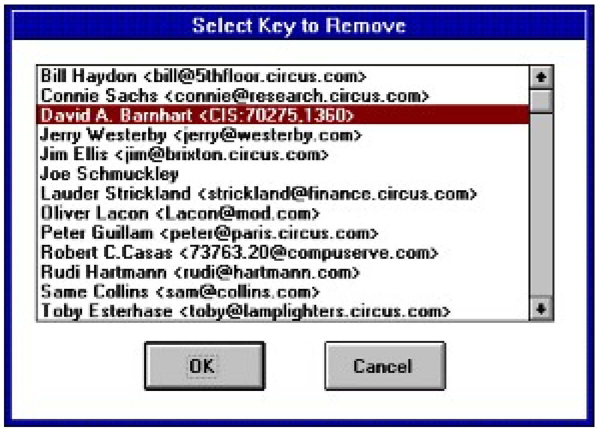
```
Looking for key for user 'Bill Haydon <bill@5thfloor.circus.com>':

Key for user ID: Bill Haydon <bill@5thfloor.circus.com>
1024-bit key, Key ID 126CD8D5, created 1994/08/22


READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)?
```

In this example, you are being asked to verify that this key really belongs to "Bill Haydon <bill@5thfloor.circus.com>".

```
You need a pass phrase to unlock your RSA secret key.
Key for user ID "ViaCrypt <70304.41@compuserve.com>"

Enter pass phrase:
```

```
Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on
available evidence.  If you think it does, then based on your
estimate of that person's integrity and competence in key management,
answer the following question:

Would you trust "Bill Haydon <bill@5thfloor.circus.com>"
to act as an introducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ?
```

Your trust for a key's owner to act as an introducer does not just reflect your estimation of their personal integrity. Your response ("I don't know", "No", "Usually", or "Yes") reflects how competent you think they are at understanding key management and using good judgment in signing keys.



Select Key to Remove

Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275.1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Joe Schmuckley
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Robert C.Casas <73763.20@compuserve.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>

```
Removing from key ring: 'c:\pgp\pubring.pgp', userid "0x6BF25893".

Key for user ID: Joe Schmuckley <joe@schmuckley.com>
512-bit key, Key ID 6BF25893, created 1994/08/24

Are you sure you want this key removed (y/N)?
```

```
Key or user ID is also present in secret keyring.
Do you also want to remove it from the secret keyring (y/N)?
```

```
Removing from key ring: 'C:\PGP\secring.pgp', userid "0x6BF25893".

Key for user ID: Joe Schmuckley <joe@schmuckley.com>
512-bit key, Key ID 6BF25893, created 1994/08/24

Are you sure you want this key removed (y/N)?
```

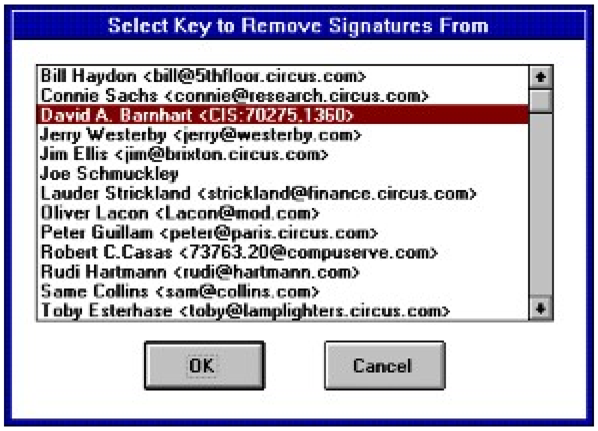
```
Removing from key ring: 'c:\pgp\pubring.pgp', userid "0xB9A00941".

Key for user ID: Joe Schmuckley <joe@schmuckley.com>
512-bit key, Key ID B9A00941, created 1994/08/24
Also known as: Joe Schmuckley <12345.678@compuserve.com>

Key has more than one user ID.
Do you want to remove the whole key (y/N)?
```

The key's user IDs are displayed one at a time so that you can select the correct user ID to remove:

```
Remove "Joe Schmuckley <joe@schmuckley.com>" (y/N)? n

Remove "Joe Schmuckley <12345.678@compuserve.com>" (y/N)Y?
User ID removed from key ring.
```

```
Key or user ID is also present in secret keyring.
Do you also want to remove it from the secret keyring (y/N)? y
```

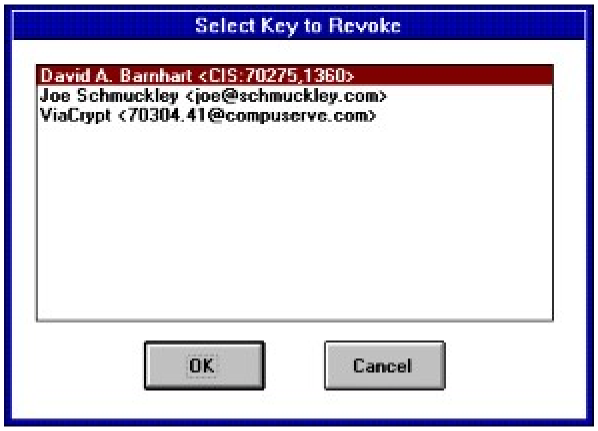
Press 'y' and ENTER.



Select Key to Remove Signatures From

Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275.1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Joe Schmuckley
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Robert C.Casas <73763.20@compuserve.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>

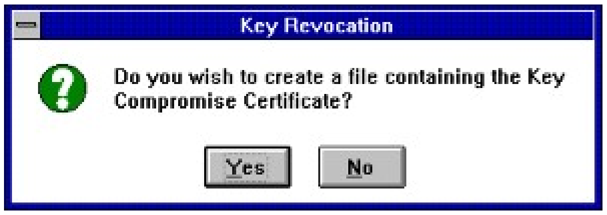
```
Removing signatures from userid '0x126CD8D5' in key ring 'c:
\pgp\pubring.pgp'

Key for user ID: Bill Haydon <bill@5thfloor.circus.com>
1024-bit key, Key ID 126CD8D5, created 1994/08/22

Key has 1 signature(s):
sig      CB768501               ViaCrypt <Phone (602) 944-0773>
Remove this signature (y/N)?
```



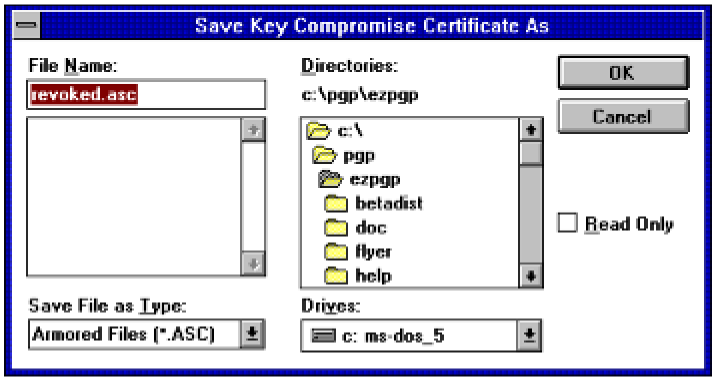
David A. Barnhart <CIS:70275,1360>
Joe Schmuckley <joe@schmuckley.com>
ViaCrypt <70304.41@compuserve.com>

```
Key for user ID: Joe Schmuckley <joe@schmuckley.com>
512-bit key, Key ID DD28EF3B, created 1994/08/24

Do you want to permanently revoke your public key
by issuing a secret key compromise certificate
for "Joe Schmuckley <joe@schmuckley.com>" (y/N)?
```

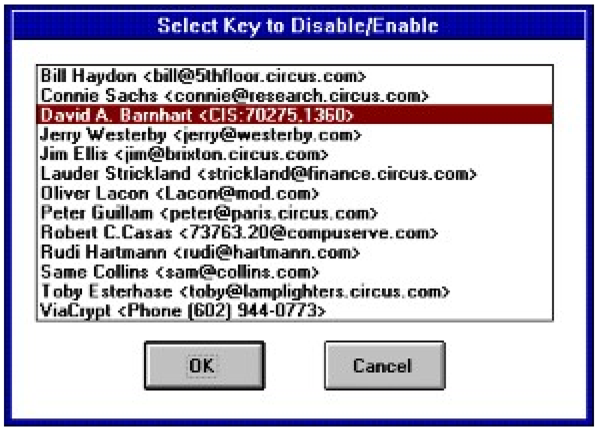
(NOTE: While the prompt says you will be issuing a key compromise certificate, in actuality, your key will simply be marked on your keyring as 'revoked'. Generating the Key Compromise Certificate occurs in a later step.)



Do you wish to create a file containing the Key Compromise Certificate?



If you want to create the Key Compromise Certificate file on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory in which you want to save the encrypted file. In the File Name box, type a name for the file. While the default extension for the encrypted file's name will be .ASC or .PGP, the filename you specify can have any extension you want. Choose the OK button.



Select Key to Disable/Enable

```
Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275.1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Robert C.Casas <73763.20@compuserve.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>
ViaCrypt <Phone (602) 944-0773>
```

```
Key for user ID: Bill Haydon <bill@5thfloor.circus.com>
1024-bit key, Key ID 126CD8D5, created 1994/08/22

Disable this key (y/N)?
```

If the specified public key is already disabled, then you will instaed be asked if you want the key reenabled.

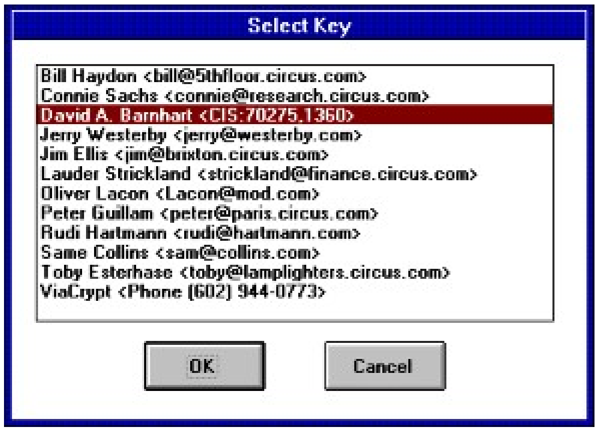
The "Key ID" column contains the Key's ID.   The key ID is the least significant a 32 bits of the key   hexadecimal number.   It it shown as a four-byte hexadecimal number.   While many keys may share the same user ID, for all practical purposes no two keys have the same key ID.

The "Len" column contains the keys length in bits.

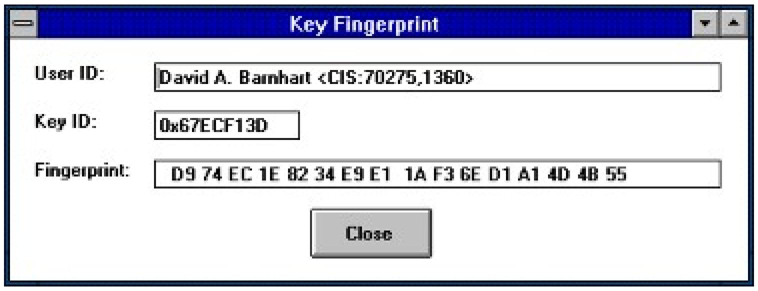
The "Creation date" column contains the date the key was generated.

The User ID column contains the user ID(s) for the key. Note that ViaCrypt's key contains four user IDs.

Choose the Close button to dismiss the dialog box.



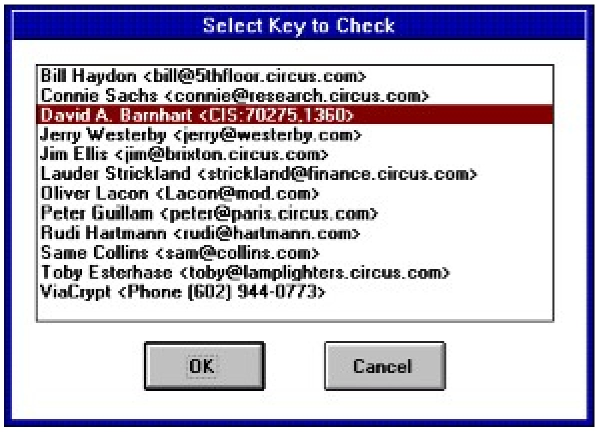
Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275.1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>
ViaCrypt <Phone (602) 944-0773>



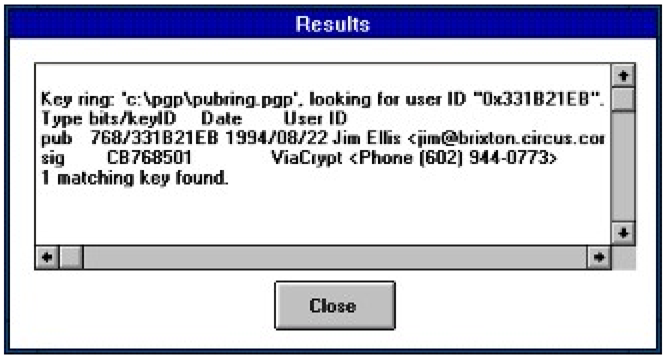
User ID: David A. Barnhart <CIS:70275,1360>

Key ID: 0x67ECF13D

Fingerprint: D9 74 EC 1E 82 34 E9 E1 1A F3 6E D1 A1 4D 48 55



Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275,1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>
ViaCrypt <Phone (602) 944-0773>



```
Key ring: 'c:\pgp\pubring.pgp', looking for user ID "0x331B21EB".
Type bits/keyID    Date       User ID
pub   768/331B21EB 1994/08/22 Jim Ellis <jim@brixton.circus.cor
sig       CB768501            ViaCrypt <Phone (602) 944-0773>
1 matching key found.
```

The line that begins with "pub" contains the key length, key ID, creation date, and user ID of the key.

The line(s) that begin with "sig" contains the key ID and User ID of the keys used to sign/certify this key.

```
Editing userid "0xB9A00941" in key ring: 'C:\PGP\pubring.pgp'.

Key for user ID: Joe Schmuckley <joe@schmuckley.com>
512-bit key, Key ID B9A00941, created 1994/08/24

You need a pass phrase to unlock your RSA secret key.
Key for user ID "Joe Schmuckley <12345.678@compuserve.com>"

Enter pass phrase:
```

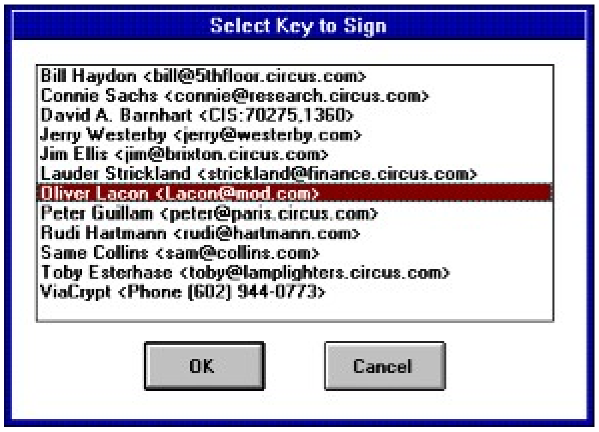
```
Current user ID: Joe Schmuckley <joe@schmuckley.com>
Do you want to add a new user ID (y/N)?
Do you want to change your pass phrase (y/N)? y
```

```
Enter pass phrase:
Enter same pass phrase again:
Secret key ring updated...
(No need to update public key ring)
```

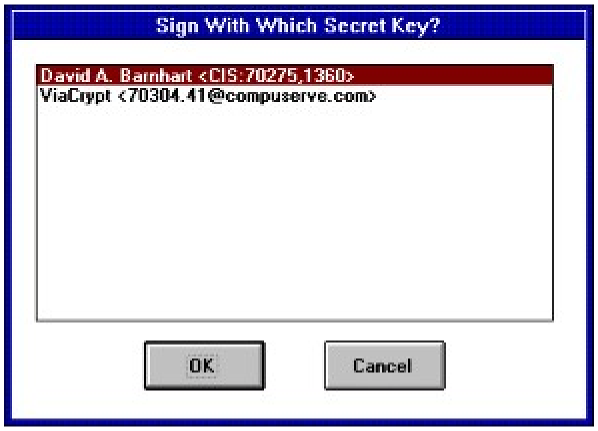
```
Enter the new user ID: Joe Schmuckley <12345.678@compuserve.com>
```

```
Make this user ID the primary user ID for this key (y/N)? y
```

```
Do you want to change your pass phrase (y/N)? n
Secret key ring updated...
Public key ring updated.
```



Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275,1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Lauder Strickland <strickland@finance.circus.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>
ViaCrypt <Phone (602) 944-0773>



David A. Barnhart <CIS:70275,1360>
ViaCrypt <70304.41@compuserve.com>

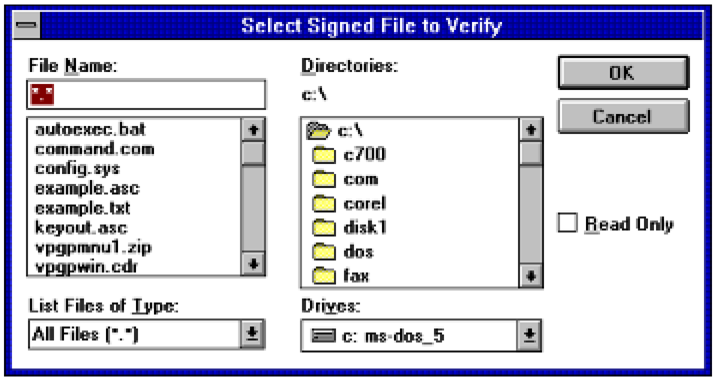
```
Looking for key for user '0x126CD8D5':

Key for user ID: Bill Haydon <bill@5thfloor.circus.com>
1024-bit key, Key ID 126CD8D5, created 1994/08/22


READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify
that the above public key actually belongs to the user specified by
the above user ID (y/N)?
```

Using this example, you would be certifying that this key really belongs to Bill Haydon.

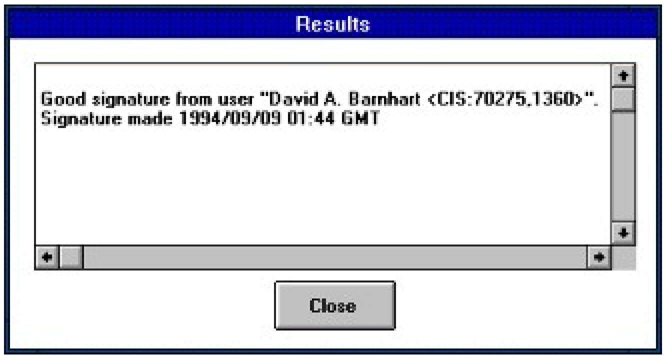
```
Would you trust "Bill Haydon <bill@5thfloor.circus.com>"
to act as an introducer and certify other people's public keys
to you?(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ?
```

```
Key for user ID: Bill Haydon <bill@5thfloor.circus.com>
1024-bit key, Key ID 126CD8D5, created 1994/08/22
This user is untrusted to certify other keys.
This key/userID association is fully certified.
  Axiomatically trusted certification from:
  David A. Barnhart <CIS:70275,1360>
Current trust for this key's owner is: untrusted

Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on
available evidence.  If you think it does, then based on your
estimate of that person's integrity and competence in key
management, answer the following question:

Would you trust "Bill Haydon <bill@5thfloor.circus.com>"
to act as an introducer and certify other people's public keys to
you? (1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ?
```

This advanced option is available when you decrypt a file using the "Decrypt" button. Occasionally, you may be decrypting an ASCII text file that you feel is sufficiently sensitive that it must be displayed on your screen instead of written to your disk. In this case, select this option. If the decrypted file is ASCII text, it will be displayed on your screen, one page at a time.



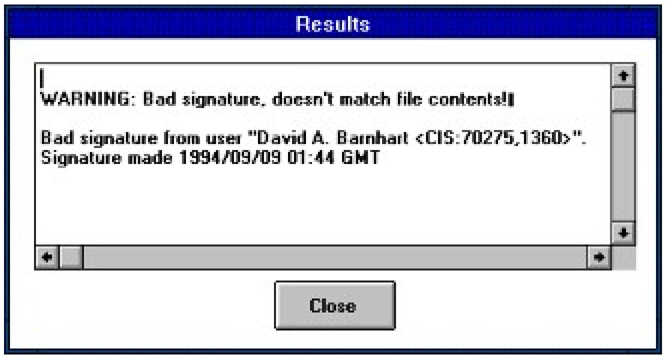
If the signed file is on a different drive, select the drive you want from the Drives box. In the directories box, choose the directory you want to view (Double-click the directory, or press the UP ARROW or DOWN ARROW key to select the directory, then press ENTER). From the list of files, select the signed file to verify.



```
Good signature from user "David A. Barnhart <CIS:70275,1360>".
Signature made 1994/09/09 01:44 GMT
```

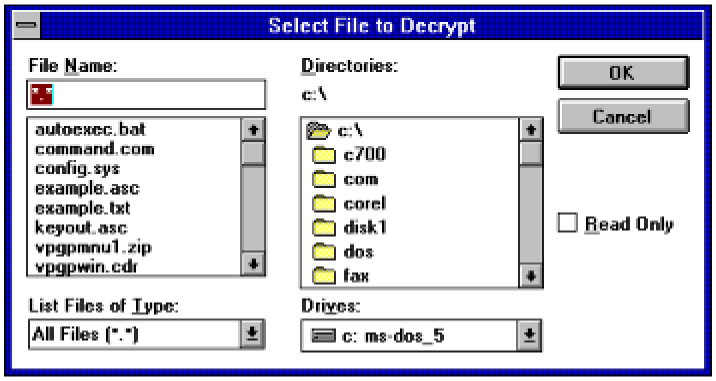
This indicates when ViaCrypt PGP checked the file's digital signature and it was correct. A correctly certified digital signature assures you that:

1. The files contents have not been altered since it was signed.
2. The file actually came from (or was signed by) the signer, provided you 'trust' the signer's key.

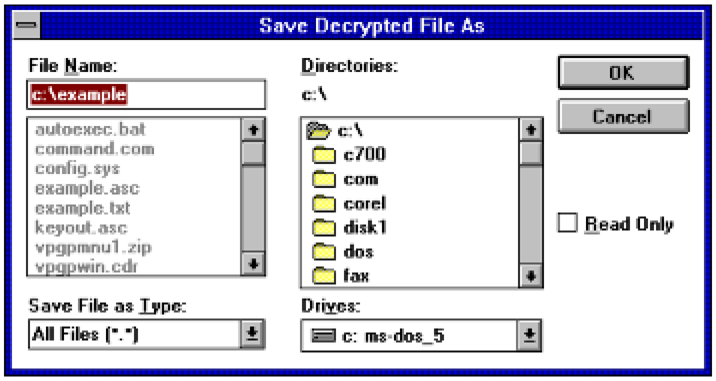


```
WARNING: Bad signature, doesn't match file contents!

Bad signature from user "David A. Barnhart <CIS:70275,1360>".
Signature made 1994/09/09 01:44 GMT
```

This indicates that the file may be corrupted (accidentally or intentionally), or the signature may be a forgery.



If the file you want to decrypt is on a different drive, select the drive you want from the Drives box. In the directories box, choose the directory you want to view (Double-click the directory, or press the UP ARROW or DOWN ARROW key to select the directory, then press ENTER). From the list of files, select the file you want to decrypt.Next, you may specify a name for the decrypted file, or accept the default:



Save Decrypted File As

File Name: c:\example

autoexec.bat
command.com
config.sys
example.asc
example.txt
keyout.asc
vpgpmnu1.zip
vpgpwin.cdr

Directories: c:\

c:\
c700
com
corel
disk1
dos
fax

Save File as Type: All Files (*.*)

Drives: c: ms-dos_5

If you want to save the decrypted file on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory in which you want to save the decrypted file. In the File Name box, type a name for the file. While the default extension for the decrypted file's name will be .ASC or .PGP, the filename you specify can have any extension you want. Choose the OK button.

```
File is encrypted.  Secret key is required to read it.
Key for user ID: David A. Barnhart <CIS:70275,1360>
512-bit key, Key ID 67ECF13D, created 1993/07/29

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase:
```

You should see a display similar to this one while ViaCrypt PGP decrypts your file:

```
Pass phrase is good.  Just a moment......
Plaintext filename: C:\EXAMPLE
```

If the encrypted file is also digitally signed, you should see.

```
File has signature.  Public key is required to check signature. .
Good signature from user "David A. Barnhart <CIS:70275,1360>".
Signature made 1994/08/22 22:49 GMT
```

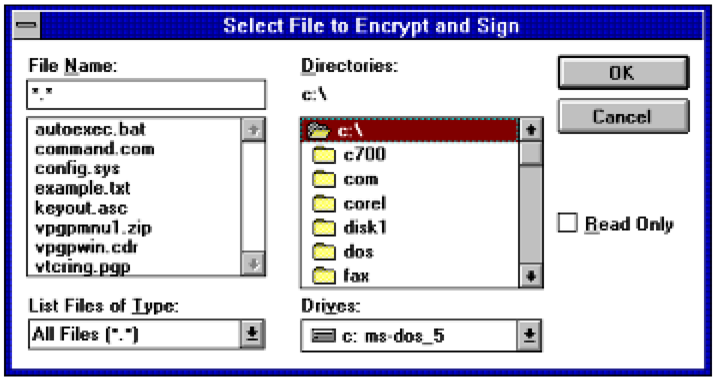
This indicates ViaCrypt PGP checked the file's digital signature and it was correct. A correctly certified digital signature assures you that:

> The files contents have not been altered since it was signed.
>
> The file actually came from (or was signed by) the signer. (Provided that you 'trust' the signer's key.

If the display is:

```
WARNING: Bad signature, doesn't match file contents!

Bad signature from user . . .
```

This indicates that the file may be corrupted (accidentally or intentionally), or the signature may be a forgery.



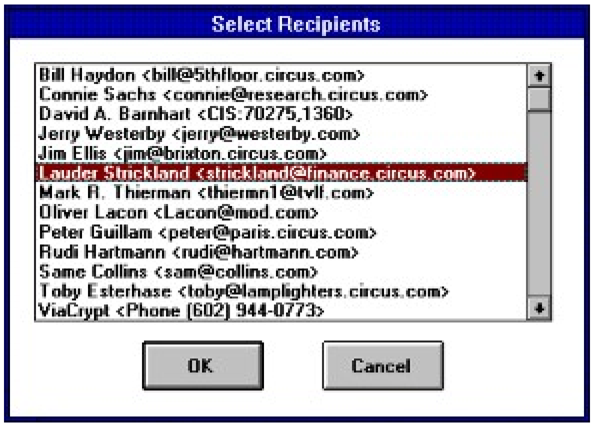
File Name: *.*

Directories: c:\

List Files of Type: All Files (*.*)

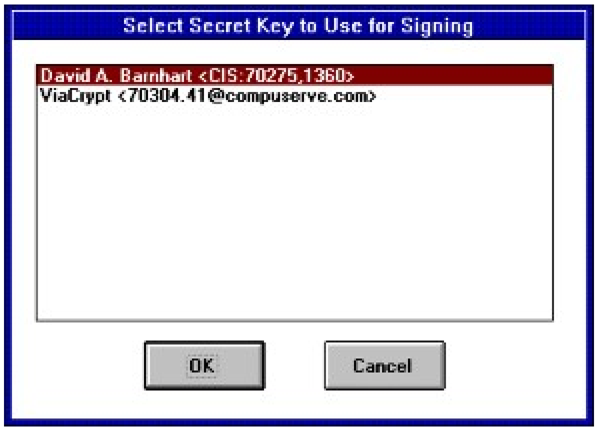
Drives: c: ms-dos_5

If the file you want to encrypt and sign is on a different drive, select the drive you want from the Drives box. In the directories box, choose the directory you want to view (Double-click the directory, or press the UP ARROW or DOWN ARROW key to select the directory, then press ENTER). From the list of files, select the file you want to encrypt and sign.

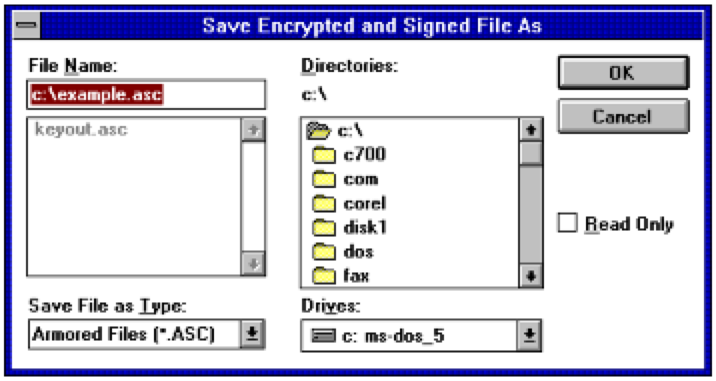


Bill Haydon <bill@5thfloor.circus.com>
Connie Sachs <connie@research.circus.com>
David A. Barnhart <CIS:70275,1360>
Jerry Westerby <jerry@westerby.com>
Jim Ellis <jim@brixton.circus.com>
Lauder Strickland <strickland@finance.circus.com>
Mark R. Thierman <thiermn1@tvlf.com>
Oliver Lacon <Lacon@mod.com>
Peter Guillam <peter@paris.circus.com>
Rudi Hartmann <rudi@hartmann.com>
Same Collins <sam@collins.com>
Toby Esterhase <toby@lamplighters.circus.com>
ViaCrypt <Phone (602) 944-0773>

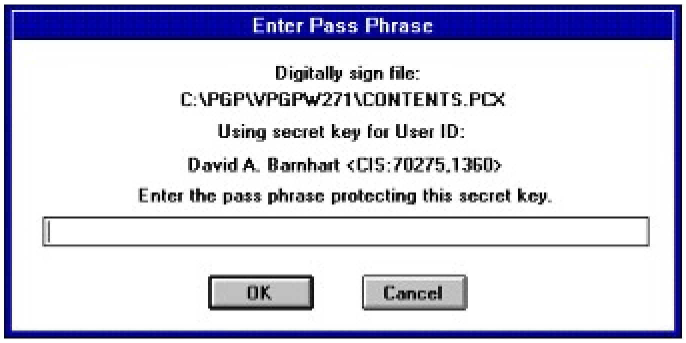
One or more recipients may be selected. To select multiple recipients, hold down the 'Ctrl' key while clicking on your selections. Release the 'Ctrl' key, then choose the OK button.



David A. Barnhart <CIS:70275,1360>
ViaCrypt <70304.41@compuserve.com>



If you want to save the file on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory in which you want to save the file. In the File Name box, type a name for the file. While the default extension for the encrypted and signed file's name will be .ASC or .PGP, the filename you specify can have any extension you want. Choose the OK button.



Enter Pass Phrase

Digitally sign file:

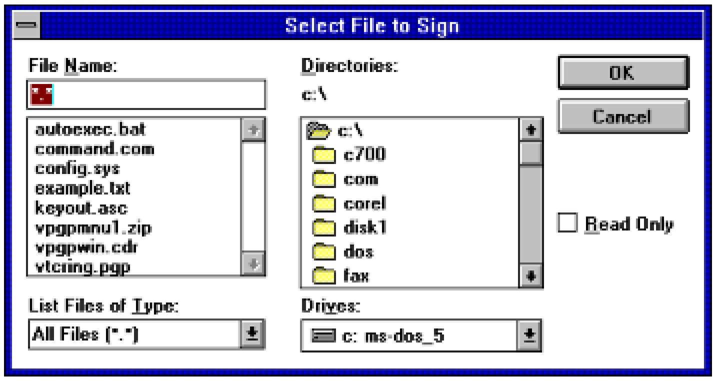
C:\PGP\VPGPw271\CONTENTS.PCX

Using secret key for User ID:

David A. Barnhart <CIS:70275,1360>

Enter the pass phrase protecting this secret key.

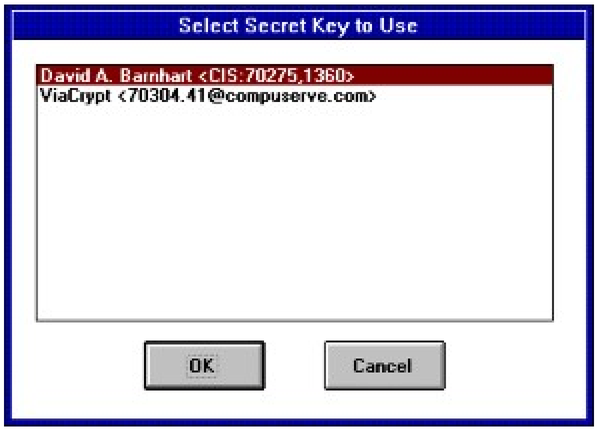
```
Pass phrase is good.
Key for user ID: David A. Barnhart <CIS:70275,1360>
512-bit key, Key ID 67ECF13D, created 1993/07/29
Just a moment....

Recipients' public key(s) will be used to encrypt.
Key for user ID: David A. Barnhart <CIS:70275,1360>
512-bit key, Key ID 67ECF13D, created 1993/07/29
.
Transport armor file: C:\EXAMPLE.ASC
Press any key to continue
```



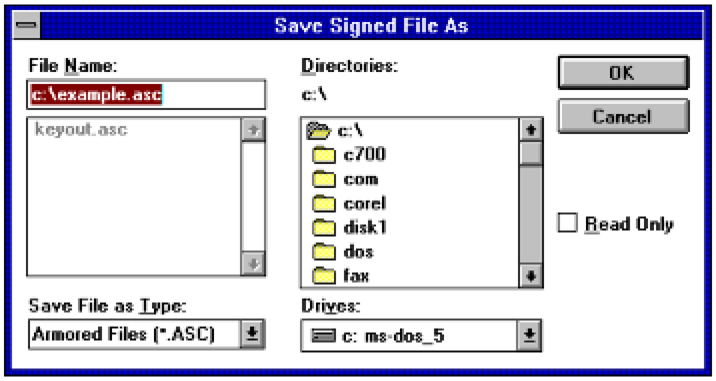
autoexec.bat
command.com
config.sys
example.txt
keyout.asc
vpgpmnu1.zip
vpgpwin.cdr
vtcring.pgp

c:\
c700
com
corel
disk1
dos
fax

If the file you want to sign is on a different drive, select the drive you want from the Drives box. In the directories box, choose the directory you want to view (Double-click the directory, or press the UP ARROW or DOWN ARROW key to select the directory, then press ENTER). From the list of files, select the file you want to sign.

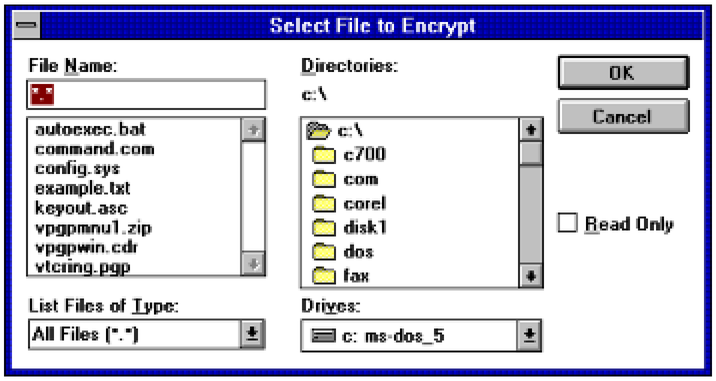


David A. Barnhart <CIS:70275,1360>
ViaCrypt <70304.41@compuserve.com>

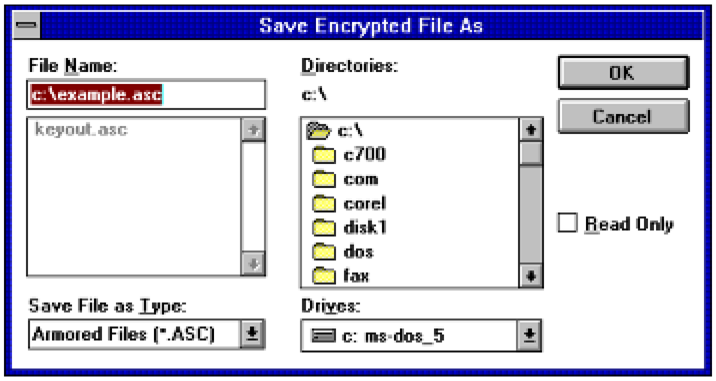


If you want to save the signed file on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory in which you want to save the signed file. In the File Name box, type a name for the file. While the default extension for the signed file's name will be .ASC or .PGP Choose the OK button.

```
Pass phrase is good.
Key for user ID: David A. Barnhart <CIS:70275,1360>
512-bit key, Key ID 67ECF13D, created 1993/07/29
Just a moment....
Clear signature file: C:\PGP\CONFIG.ASC
Press any key to continue . . .
```



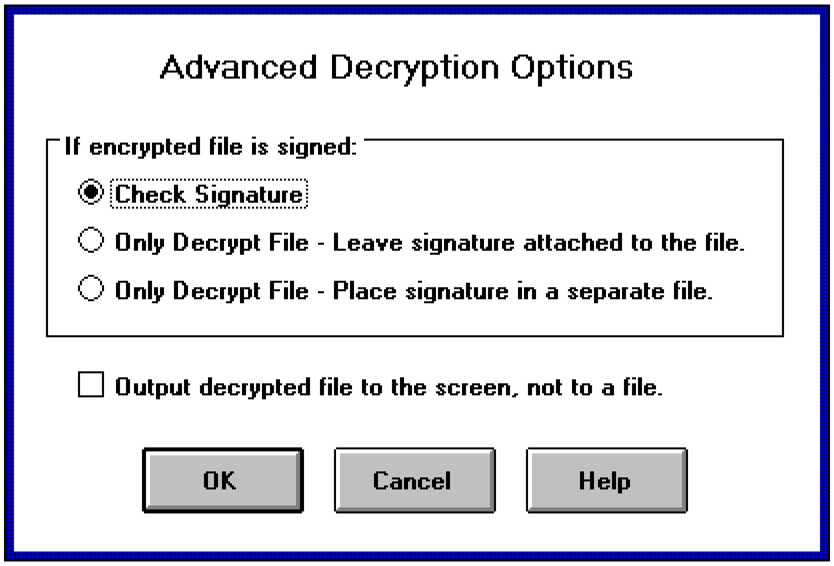
If the file you want to encrypt is on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory you want to view (Double-click the directory, or press the UP ARROW or DOWN ARROW key to select the directory, then press ENTER). From the list of files, select the file you want to encrypt.



If you want to save the encrypted file on a different drive, select the drive you want from the Drives box. In the Directories box, choose the directory in which you want to save the encrypted file. In the File Name box, type a name for the file. While the default extension for the encrypted file's name will be .ASC or .PGP Then choose the OK button.

Advanced Decrypt Options

Click on the parts of the window where the appears for descriptions of each item in the window:



Advanced Decryption Options

If encrypted file is signed:

- [✓ Check Signature]
- Only Decrypt File - Leave signature attached to the file.
- Only Decrypt File - Place signature in a separate file.

☐ Output decrypted file to the screen, not to a file.

About ViaCrypt PGP for Windows

ViaCrypt PGP for Windows

Version 2.7.1

April 13, 1995