An Introduction to Cryptography

Release Information

An Introduction to Cryptography; released July 2005.

Copyright Information

Trademark Information

Licensing and Patent Information

Acknowledgments

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

Export Information

Limitations

About PGP Corporation

Recognized worldwide as a leader in enterprise encryption technology, PGP Corporation develops, markets, and supports products used by more than 30,000 enterprises, businesses, and governments worldwide, including 90% of the Fortune® 100 and 75% of the Forbes® International 100. PGP products are also used by thousands of individuals and cryptography experts to secure proprietary and confidential information. During the past 14 years, PGP® technology has earned a global reputation for standards-based, trusted security products. It is the only commercial security vendor to publish source code for peer review. The unique PGP encryption product suite includes PGP Universal—an automatic, self-managing, network-based solution for enterprises—as well as desktop, mobile, FTP/batch transfer, and SDK solutions. Contact PGP Corporation at www.pgp.com or +1 650 319 9000.

Table of Contents

Introduction

Cryptography is the stuff of spy novels and action comics. Kids once saved up Ovaltine™ labels and sent away for Captain Midnight's Secret Decoder Ring. Almost everyone has seen a television show or movie involving a nondescript suit-clad gentleman with a briefcase handcuffed to his wrist. The term "espionage" conjures images of James Bond, car chases, and flying bullets.

And here you are, sitting in your office, faced with the rather mundane task of sending a sales report to a coworker in such a way that no one else can read it. You just want to be sure that your colleague is the actual and only recipient of the email and you want him or her to know you were unmistakably the sender. It's not national security at stake, but if your company's competitor got hold of it, it could cost you. How can you accomplish this?

You can use cryptography. You may find it lacks some of the drama of code phrases whispered in dark alleys, but the result is the same: information revealed only to those for whom it was intended.

Who should read this guide

This guide is useful to anyone who is interested in knowing the basics of cryptography; it explains the terminology and technology you will encounter as you use PGP products. You will find it useful to read before you begin working with cryptography.

How to use this guide

This guide includes the following chapters:

- Chapter 1, "The Basics of Cryptography," provides an overview of the terminology and concepts you will encounter as you use PGP products.

- Chapter 2, "The Self-Managing Security Architecture," describes a new kind of system for secure messaging, the self-managing security architecture (SMSA).

- Chapter 3, "Phil Zimmermann on PGP," written by the creator of PGP software, contains discussions of security, privacy, and the vulnerabilities inherent in any security system, even a system protected by PGP technology.

There is also a Glossary and an Index.

Recommended readings

This section identifies Web sites, books, and periodicals about the history, technical aspects, and politics of cryptography.

The History of Cryptography

- *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*, Simon Singh, Doubleday & Company, Inc., 1999, ISBN 0-385-49531-5.

- *The Codebreakers: The Story of Secret Writing*, David Kahn, Simon & Schuster Trade, 1996, ISBN 0-684-83130-9 (updated from the 1967 edition). This book is a history of codes and code breakers from the time of the Egyptians to the end of WWII. Kahn first wrote it in the sixties; this is the revised edition. This book will not teach you anything about how cryptography is done, but it has been the inspiration of the whole modern generation of cryptographers.

- Aegean Park Press, www.aegeanparkpress.com. The Aegean Park Press publishes a number of interesting historic books ranging from histories (such as *The American Black Chamber*, an exposé of U.S. cryptography during and after WWI) to declassified government documents.

Technical Aspects of Cryptography

Websites

- www.iacr.org. International Association for Cryptologic Research (IACR). The IACR holds cryptographic conferences and publishes journals.

- www.ietf.org/rfc/rfc3156.txt. The IETF MIME Security with OpenPGP specification, written by Michael Elkins, Dave del Torto, Raph Levien, and Thomas Roessler.

- www.ietf.org/rfc/rfc2440.txt. The IETF OpenPGP Message Format specification, written by Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer.

- http://en.wikipedia.org/wiki/PGP. The Wikipedia entry for PGP includes a lot of good information about PGP and cryptography.

Books and Periodicals

- *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd edition, Bruce Schneier, John Wiley & Sons, 1996; ISBN 0471117099. If you can only buy one book to get started in cryptography, this is the one to buy.

- *Handbook of Applied Cryptography*, Alfred Menezes, Paul van Oorschot and Scott Vanstone, CRC Press, 1996; ISBN 0-8493-8523-7. This is the technical book you should get after Schneier. There is a lot of heavy-duty math in this book, but it is nonetheless usable for those who do not understand the math.

- *Journal of Cryptology*, International Association for Cryptologic Research (IACR). See www.iacr.org.

- *Advances in Cryptology*, conference proceedings of the IACR CRYPTO conferences, published yearly by Springer-Verlag. See www.iacr.org.

- *The Design of Rijndael: AES - The Advanced Encryption Standard*, by Joan Daemen and Vincent Rijmen, published by Springer-Verlag, 2002; ISBN 3540425802. The new Advanced Encryption Standard (AES) algorithm is based on the block cipher Rijndael. The designers of Rijndael describe how it works.

Politics of Cryptography

Websites

- www.epic.org, Electronic Privacy Information Center.

- www.eff.org, Electronic Frontier Foundation.

- www.privacy.org, privacy.org. Information resource about privacy issues.

- www.cdt.org, Center for Democracy and Technology.

- www.philzimmermann.com, Phil Zimmermann's home page, his Senate testimony, and more.

Books

- *Privacy on the Line: The Politics of Wiretapping and Encryption*, Whitfield Diffie and Susan Landau, The MIT Press, 1998, ISBN 0-262-04167-7. This book is a discussion of the history and policy surrounding cryptography and communications security. It is an excellent read, even for beginners and non-technical people. Includes information that even a lot of experts don't know.

- *Crypto: How the Code Rebels Beat the Government—Saving Privacy in the Digital Age*, Steven Levy, Penguin USA, 2001; ISBN 0140244328.

Network Security

Books

- *Building Internet Firewalls*, Elizabeth D. Zwicky, D. Brent Chapman, Simon Cooper, and Deborah Russell (Editor), O'Reilly & Associates, Inc., 2000; ISBN: 1565928717. This book is a practical guide to designing, building, and maintaining firewalls.

- *Firewalls and Internet Security: Repelling the Wily Hacker*, William R. Cheswick, Steven M. Bellovin, Addison Wesley Longman, Inc., 1994; ISBN: 0201633574. This book is a practical guide to protecting networks from hacker attacks through the Internet.

- *Network Security: Private Communication in a Public World*, Second Edition, Charles Kaufman, Radia Perlman, and Mike Speciner, Pearson Education, 2002; ISBN: 0130460192. This book describes many network protocols, including Kerberos, IPsec, SSL, and others shows how actual systems are constructed.

1   The Basics of Cryptography

When Julius Caesar sent messages to his generals, he didn't trust his messengers. So he replaced every A in his messages with a D, every B with an E, and so on through the alphabet. Only someone who knew the "shift by 3" rule could decipher his messages.

And so we begin.

Encryption and decryption

Data that can be read and understood without any special measures is called plaintext or cleartext. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encrypting plaintext results in unreadable gibberish called ciphertext. You use encryption to make sure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting ciphertext to its original plaintext is called decryption.

The following figure shows this process.

plaintext   **encryption**   ciphertext   **decryption**   plaintext
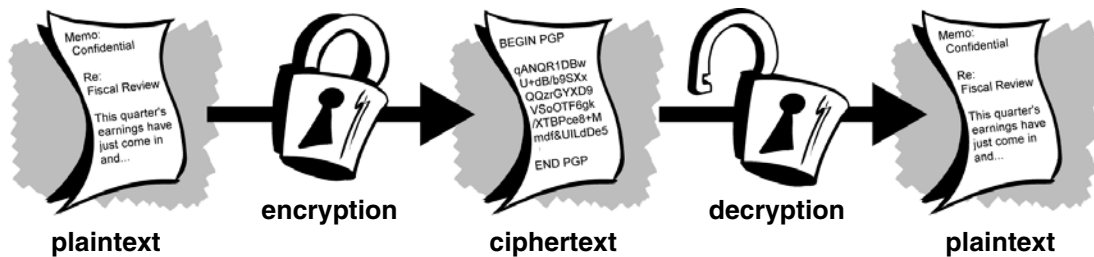
What is cryptography?

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.

While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called attackers.

Cryptology embraces both cryptography and cryptanalysis.

A related discipline is steganography, which is the science of hiding messages rather than making them unreadable. Steganography is not cryptography; it is a form of coding. It relies on the secrecy of the mechanism used to hide the message. If, for example, you encode a secret message by putting each letter as the first letter of the first word of every sentence, it's secret until someone knows to look for it, and then it provides no security at all.

Strong cryptography

"There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files. This book is about the latter."

—Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C

PGP products are also about the latter sort of cryptography.

Cryptography can be strong or weak, as explained above. Cryptographic strength is measured in the time and resources it would require to recover the plaintext. The result of strong cryptography is ciphertext that is very difficult to decipher without possession of the appropriate decoding tool. How difficult? Given all of today's computing power and available time—even a billion computers doing a billion checks a second—it is not possible to decipher the result of strong cryptography before the end of the universe.

One would think, then, that strong cryptography would hold up rather well against even an extremely determined cryptanalyst. Who's really to say? No one has proven that the strongest encryption obtainable today will hold up under tomorrow's computing power. However, the strong cryptography employed by PGP is the best available today. Vigilance and conservatism will protect you better, however, than claims of impenetrability.

How does cryptography work?

A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key—a word, number, or phrase—to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and the secrecy of the key.

A cryptographic algorithm, plus all possible keys and all the protocols that make it work, comprise a cryptosystem. PGP technology is a cryptosystem.

Conventional cryptography

In conventional cryptography, also called secret-key or symmetric-key encryption, one key is used both for encryption and decryption. The Data Encryption Standard (DES) in an example of a conventional cryptosystem that has been widely deployed by the U.S. Government and the banking industry. It is being replaced by the Advanced Encryption Standard (AES). The following figure is an illustration of the conventional encryption process.

plaintext   **encryption**   **ciphertext**   **decryption**   **plaintext**
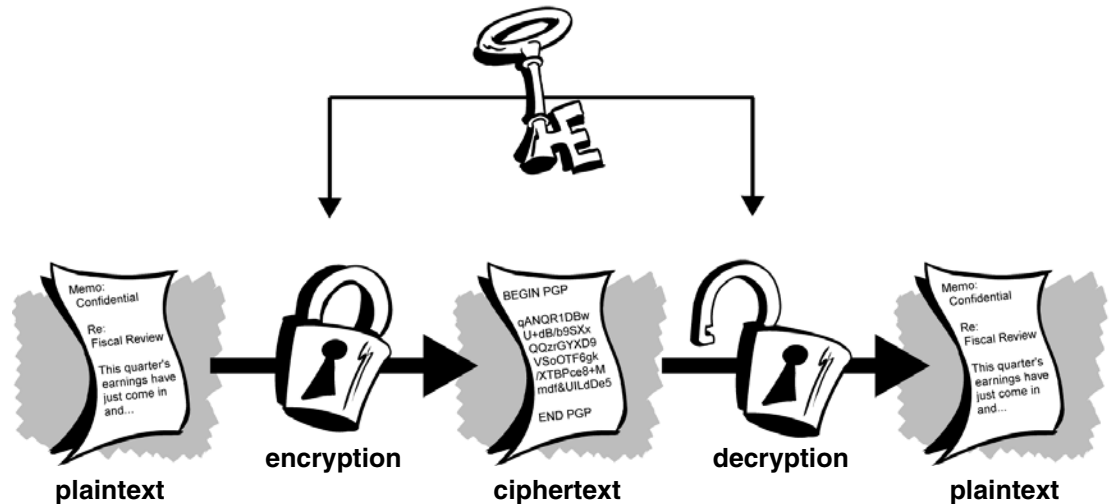
Caesar's cipher

An extremely simple example of conventional cryptography is a substitution cipher. A substitution cipher substitutes one piece of information for another. This is most frequently done by offsetting letters of the alphabet. Two examples are Captain Midnight's Secret Decoder Ring, which you may have owned when you were a kid, and Julius Caesar's cipher. In both cases, the algorithm is to offset the alphabet and the key is the number of characters to offset it.

For example, if we encode the word "SECRET" using Caesar's key value of 3, we offset the alphabet so that the 3rd letter down (D) begins the alphabet.

So starting with

    ABCDEFGHIJKLMNOPQRSTUVWXYZ

and sliding everything up by 3, you get

    DEFGHIJKLMNOPQRSTUVWXYZABC

where D = A, E = B, F = C, and so on.

Using this scheme, the plaintext, "SECRET" encrypts as "VHFUHW." To allow someone else to read the ciphertext, you tell them that the key is 3.

Obviously, this is exceedingly weak cryptography by today's standards, but it worked for Caesar and it illustrates how conventional cryptography works.

Key management and conventional encryption

Conventional encryption has benefits. It is very fast. It is especially useful for encrypting data that is not going anywhere. However, conventional encryption alone as a means for transmitting secure data can be quite expensive simply due to the difficulty of secure key distribution.

Recall a character from your favorite spy movie: the person with a locked briefcase handcuffed to his or her wrist. What is in the briefcase, anyway? It's probably not the secret plan itself. It's the key that will decrypt the secret data.

For a sender and recipient to communicate securely using conventional encryption, they must agree upon a key and keep it secret between themselves. If they are in different physical locations, they must trust a courier, the Bat Phone, or some other secure communications medium to prevent the disclosure of the secret key during transmission.

Anyone who overhears or intercepts the key in transit can later read, modify, and forge all information encrypted or authenticated with that key. From DES to Captain Midnight's Secret Decoder Ring, the persistent problem with conventional encryption is key distribution: how do you get the key to the recipient without someone intercepting it?

Public-key cryptography

The problems of key distribution are solved by public-key cryptography, the concept of which was introduced by Whitfield Diffie and Martin Hellman in 1975. (There is now evidence that the British Secret Service invented it a few years before Diffie and Hellman, but kept it a military secret—and did nothing with it.)[1]

Public-key cryptography uses a pair of keys: a public key, which encrypts data, and a corresponding private key, for decryption. Because it uses two keys, it is sometimes called asymmetric cryptography. You publish your public key to the world while keeping your private key secret. Anyone with a copy of your public key can then encrypt information that only you can read, even people you have never met.

---

[1] J H Ellis, The Possibility of Secure Non-Secret Digital Encryption, CESG Report, January 1970. [CESG is the UK's National Authority for the official use of cryptography.]

It is computationally infeasible to deduce the private key from the public key. Anyone who has a public key can encrypt information but cannot decrypt it. Only the person who has the corresponding private key can decrypt the information.
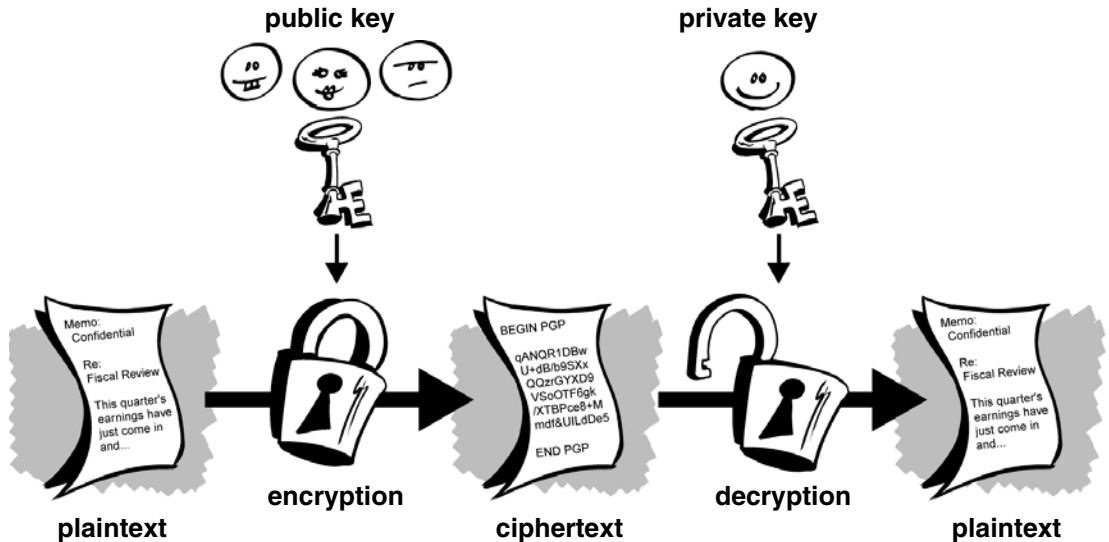
The primary benefit of public-key cryptography is that it allows people who have no preexisting security arrangement to exchange messages securely. The need for sender and receiver to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared. Some examples of public-key cryptosystems are Elgamal (named for its inventor, Taher Elgamal), RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman), Diffie-Hellman (named, you guessed it, for its inventors), and DSA, the Digital Signature Algorithm, (invented by David Kravitz).

Because conventional cryptography was once the only available means for relaying secret information, the expense of secure channels and key distribution relegated its use only to those who could afford it, such as governments and large banks (or small children with secret decoder rings). Public-key encryption is the technological revolution that provides strong cryptography to the adult masses. Remember the courier with the locked briefcase handcuffed to his wrist? Public-key encryption puts him out of business (probably to his relief).

How PGP cryptography works

PGP combines some of the best features of both conventional and public-key cryptography. PGP is a hybrid cryptosystem.

When a user encrypts plaintext with PGP, PGP first compresses the plaintext. Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis. (Files that are too short to compress or which do not compress well are not compressed.)

PGP then creates a session key, which is a one-time-only secret key. This key is a random number generated from the random movements of your mouse and the keystrokes you type. The session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is ciphertext. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the ciphertext to the recipient.

ciphertext

**plaintext is encrypted
with session key**

**session key is encrypted
with public key**

**ciphertext +
encrypted session key**
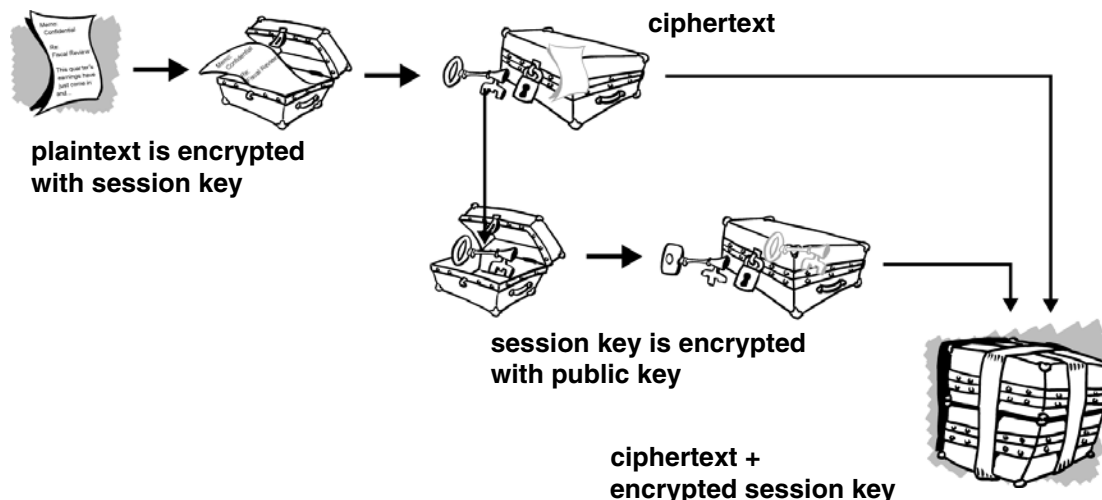
Decryption works in the reverse. The recipient's copy of PGP uses his or her private key to recover the session key, which PGP then uses to decrypt the conventionally encrypted ciphertext.

encrypted message **encrypted session key** **recipient's private key used to decrypt session key**

ciphertext

session key used to decrypt ciphertext **original plaintext**
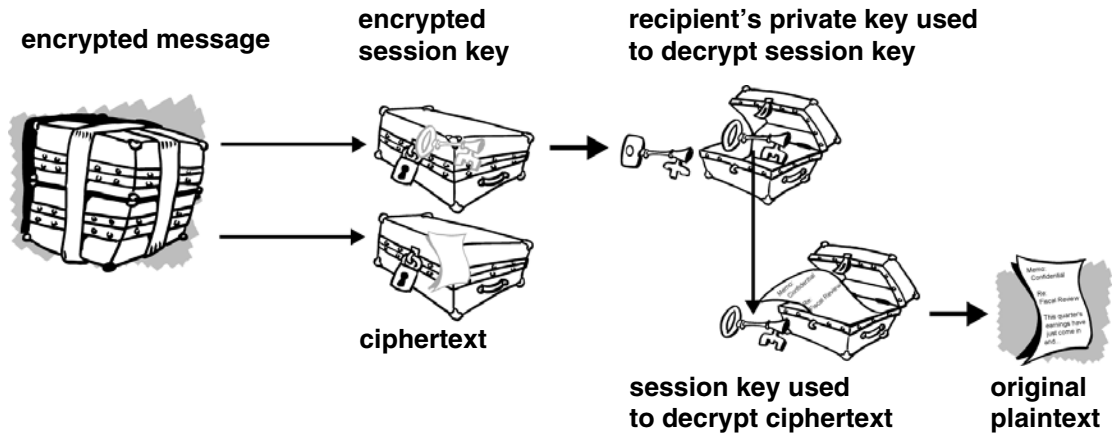
The combination of the two encryption methods combines the convenience of public-key encryption with the speed of conventional encryption. Conventional encryption is about 10,000 times faster than public-key encryption. Public-key encryption in turn provides a solution to key distribution and data transmission issues. Used together, performance and key distribution are improved without any sacrifice in security.

Keys

A key is a value that works with a cryptographic algorithm to produce a specific ciphertext. Keys are basically really, really, really big numbers. Key size is measured in bits; the number representing a 2048-bit key is darn huge. In public-key cryptography, the bigger the key, the more secure the ciphertext.

However, public key size and conventional cryptography's secret key size are totally unrelated. A conventional 80-bit key has the equivalent strength of a 1024-bit public key. A conventional 128-bit key is equivalent to a 3000-bit public key. Again, the bigger the key, the more secure, but the algorithms used for each type of cryptography are very different and thus comparison is like that of apples to oranges.

While the public and private keys are mathematically related, it's very difficult to derive the private key given only the public key; however, deriving the private key is always possible given enough time and computing power. This makes it very important to pick keys of the right size; large enough to be secure, but small enough to be applied fairly quickly. Additionally, you need to consider who might be trying to read your files, how determined they are, how much time they have, and what their resources might be.

Larger keys will be cryptographically secure for a longer period of time. If what you want to encrypt needs to be hidden for many years, you might want to use a very large key. Of course, who knows how long it will take to determine your key using tomorrow's faster, more efficient computers? There was a time when a 56-bit symmetric key was considered extremely safe.

Current thinking is that 128-bit keys will be safe indefinitely, at least until someone invents a usable quantum computer. We also believe that 256-bit keys will be safe indefinitely, even if someone invents a quantum computer. This is why the AES includes options for 128 and 256-bit keys. But history tells is that it's quite possible someone will think this statement amusingly quaint in a few decades.

Keys are stored in encrypted form. PGP software stores the keys in two files on your hard disk; one for public keys and one for private keys. These files are called keyrings. As you use PGP products, you will typically add the public keys of your recipients to your public keyring. Your private keys are stored on your private keyring. If you lose your private keyring you will be unable to decrypt any information encrypted to keys on that ring. Consequently, it's a good idea to keep good backups.

Digital signatures

A major benefit of public key cryptography is that it provides a method for employing digital signatures. Digital signatures let the recipient of information verify the authenticity of the information's origin, and also verify that the information was not altered while in transit. Thus, public key digital signatures provide authentication and data integrity. These features are every bit as fundamental to cryptography as privacy, if not more.

A digital signature serves the same purpose as a seal on a document, or a handwritten signature. However, because of the way it is created, it is superior to a seal or signature in an important way. A digital signature not only attests to the identity of the signer, but it also shows that the contents of the information signed has not been modified. A physical seal or handwritten signature cannot do that. However, like a physical seal that can be created by anyone with possession of the signet, a digital signature can be created by anyone with the private key of that signing keypair.

Some people tend to use signatures more than they use encryption. For example, you may not care if anyone knows that you just deposited $1,000 in your account, but you do want to be darn sure it was the bank teller you were dealing with.

The basic manner in which digital signatures are created is shown in the following figure. The signature algorithm uses your private key to create the signature and the public key to verify it. If the information can be decrypted with your public key, then it must have originated with you.

private key   **public key**

signing

original text   **signed text**   **verified text**

verifying
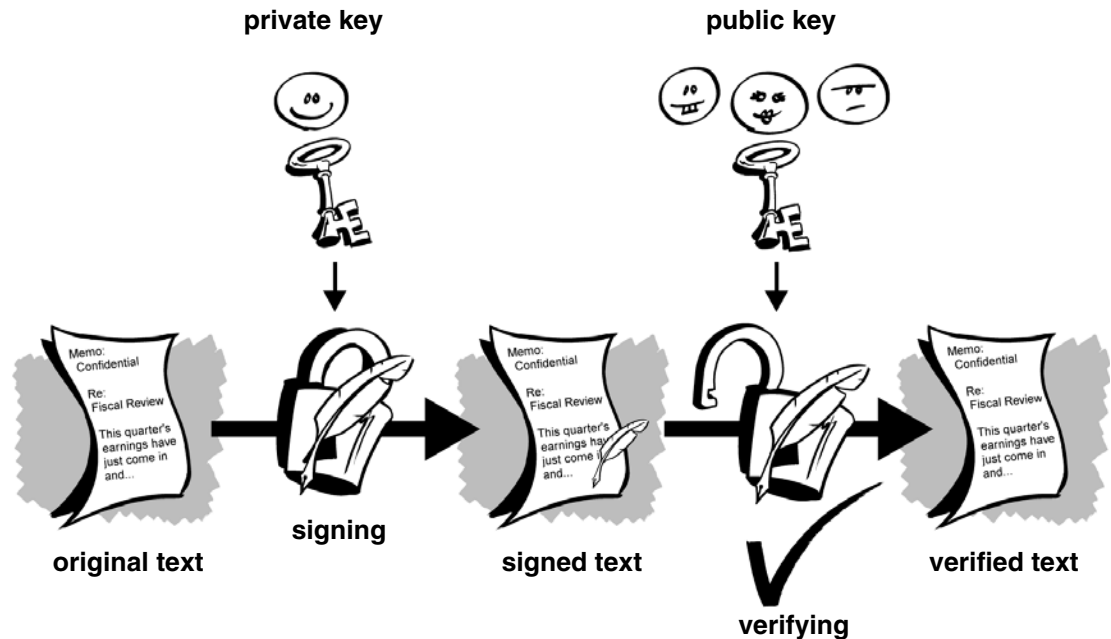
Hash functions

The system described above has some problems. It is slow, and it produces an enormous volume of data—at least double the size of the original information. An improvement on the above scheme is the addition of a one-way hash function in the process. A one-way hash function takes variable-length input—in this case, a message of any length, even thousands or millions of bits—and produces a fixed-length output; say, 160 bits. The hash function ensures that, if the information is changed in any way—even by just one bit— an entirely different output value is produced.

PGP technology uses a cryptographically strong hash function on the plaintext the user is signing. This generates a fixed-length data item known as a message digest. (Again, any change to the information results in a totally different digest.)

Then PGP technology uses the digest and the private key to create the "signature." PGP technology transmits the signature and the plaintext together. Upon receipt of the message, the recipient uses PGP technology to recompute the digest, thus verifying the signature. PGP technology can encrypt the plaintext or not; signing plaintext is useful if some of the recipients are not interested in or capable of verifying the signature.

As long as a secure hash function is used, there is no way to take someone's signature from one document and attach it to another, or to alter a signed message in any way. The slightest change to a signed document will cause the digital signature verification process to fail.

plaintext

hash function

message digest

**digest signed
with private key**

**plaintext
+
signature**

**private key
used for signing**
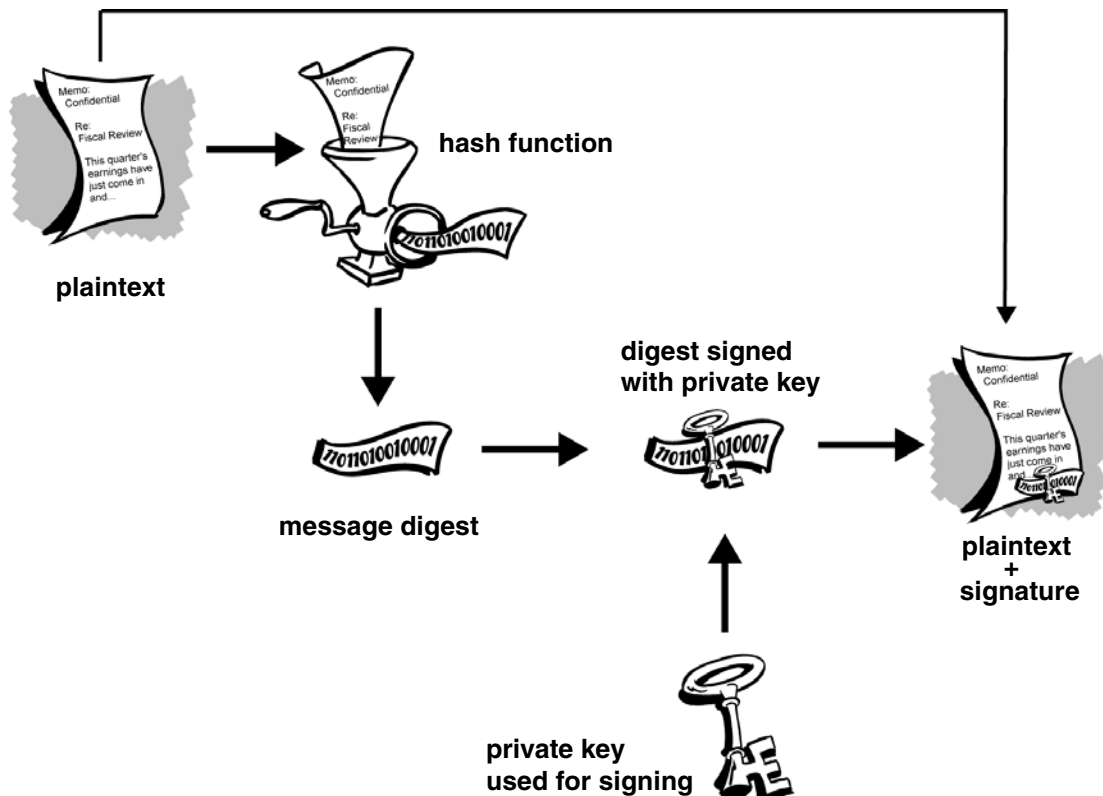
Digital signatures play a major role in authenticating and validating the keys of other PGP software users.

Digital certificates

One issue with public key cryptosystems is that users must be constantly vigilant to make sure they are encrypting to the correct person's key. In an environment where it is safe to freely exchange keys via public servers, man-in-the-middle attacks are a potential threat. In this type of attack, someone posts a phony key with the name and user ID of the user's intended recipient. Data encrypted to—and intercepted by—the true owner of this bogus key is now in the wrong hands.

In a public key environment, it is vital that you know for certain that the public key to which you are encrypting data is in fact the public key of the intended recipient, and not a forgery. You could simply encrypt only to those

keys which have been physically handed to you. But suppose you need to exchange information with people you have never met; how can you be sure you have the correct key?

Digital certificates, or certs, simplify the task of establishing whether a public key truly belongs to the purported owner.

A certificate is a form of credential. Other kinds of credentials include your driver's license, your social security card, and your birth certificate. Each of these has some information on it identifying you and some authorization stating that someone else has confirmed your identity. Some certificates, such as your passport, are important enough confirmation of your identity that you would not want to lose them, lest someone use them to impersonate you.

A digital certificate functions much like a physical certificate. A digital certificate is information included with a person's public key that helps others verify that a key is genuine or valid. Digital certificates are used to thwart attempts to substitute one person's key for another.

A digital certificate consists of three things:

- A public key

- Certificate information (usually "identity" information about the user, such as name, user ID and so on. Certificates may also contain authorization information about the user, such as spending limit, file permissions, and so on.)

- One or more digital signatures

The purpose of the digital signature on a certificate is to state that the certificate information has been attested to by some person or entity. The digital signature does not attest to the authenticity of the certificate as a whole; it vouches only that the signed identity information goes along with, or is bound to, the public key.

Thus, a certificate is basically a public key with one or two forms of ID attached, plus a hearty stamp of approval from some other trusted individual.

key

certification

certificate

Bill Davis

B. Davis

userid

userid

signature

signature

signature
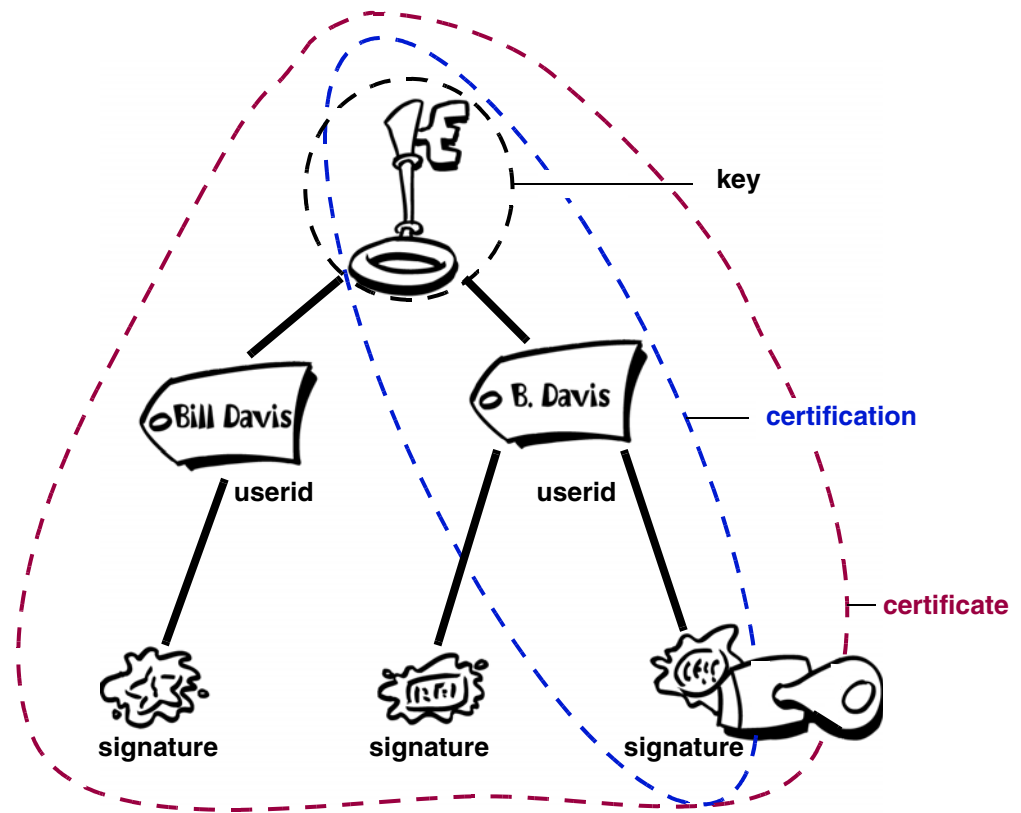
Certificate distribution

Certificates are used when it is necessary to exchange public keys with someone else. For small groups of people who wish to communicate securely, it is easy to manually exchange diskettes or emails containing each owner's public key.

This is manual public key distribution, and it is practical only to a certain point. Beyond that point, it is necessary to put systems into place that can provide the necessary security, storage, and exchange mechanisms so coworkers, business partners, or strangers could communicate if need be.

These can come in the form of storage-only repositories called Certificate Servers, or more structured systems that provide additional key management features and are called Public Key Infrastructures (PKIs).

Certificate servers

A certificate server is a network-available database that allows users to submit and receive digital certificates. Often, certificate servers are also more general-purpose directory servers. A certificate server may also provide some administrative features that help a company maintain its security policies. An

example might be allowing only the storage of keys that meet certain requirements. The PGP Keyserver (formerly known as the PGP Certificate Server) provides these, whereas generic directory servers may not.

Public Key Infrastructures (PKIs)

A PKI includes the certificate storage facilities of a certificate server, but also provides services and protocols for managing public keys. These include the ability to issue, revoke, and trust certificates. The main feature of a PKI is the introduction of what are known as Certification Authority (CA) and Registration Authority (RA) components.

A CA creates certificates and digitally signs them using the CA's private key. Because of its role in creating certificates, the CA is the central component of a PKI. Using the CA's public key, anyone wanting to verify a certificate's authenticity verifies the issuing CA's digital signature, and hence, the integrity of the contents of the certificate (most importantly, the public key and the identity of the certificate holder).

Typically, an RA refers to the people, processes, and tools used to support the registration of users with the PKI (enrollment) and ongoing administration of users. The RA may perform vetting—the process of verifying that a public key belongs to its purported owner.

An RA is a human entity—a person, group, department, company, or other association. A CA on the other hand, is often software that is used to issue the actual certificates to its computer users. There are even fancy hardware CAs that are constructed of gun metal, are tamper proof, and have kill switches on the side that—in the case of some attack—can zero out all the keys.

The role of the RA/CA is analogous to a country's Passport Office, where some people verify that a passport should be issued (the RA function) and others create the actual passport and send it to its holder (the CA function). There need not be an RA with a CA, but it provides for separation of roles, which can be important in some circumstances.

Certificate formats

A digital certificate is basically a collection of identifying information bound together with a public key and signed by a trusted third party to prove its authenticity. A digital certificate can be one of a number of different formats.

PGP software recognizes two different certificate formats:

• PGP certificates (more commonly referred to simply as PGP keys)
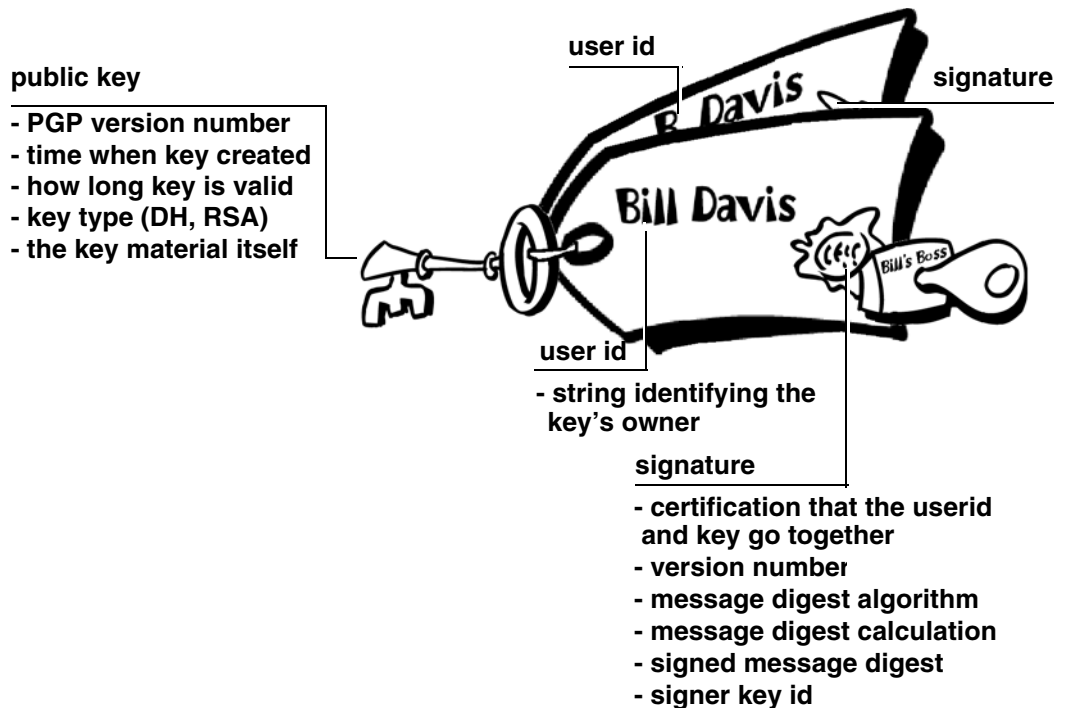
• X.509 certificates

PGP certificate format

A PGP certificate includes (but is not limited to) the following information:

- **The certificate holder's public key**—the public portion of your key pair, together with the algorithm of the key: RSA, RSA Legacy, DH (Diffie-Hellman), or DSA (Digital Signature Algorithm).

- **The certificate holder's information**—this consists of "identity" information about the user, such as his or her name, user ID, email address, ICQ number, photograph, and so on.

- **The digital signature of the certificate owner**—also called a self-signature, this is the signature using the corresponding private key of the public key associated with the certificate.

- **The certificate's validity period**—the certificate's start date/time and expiration date/time; indicates when the certificate will expire. If the key pair contains subkeys, then this includes the expiration of each of the encryption subkeys as well.

- **The preferred symmetric encryption algorithm for the key**—indicates the encryption algorithm to which the certificate owner prefers to have information encrypted. The supported algorithms are CAST, AES, IDEA, Triple-DES, and Twofish.

You might think of a PGP certificate as a public key with one or more labels tied to it. On these "labels" you will find information identifying the owner of the key and a signature of the key's owner, which states that the key and the identification go together. (This particular signature is called a self-signature; every PGP certificate contains a self-signature.)

One unique aspect of the PGP certificate format is that a single certificate can contain multiple signatures. Several or many people may sign the key/identification pair to attest to their own assurance that the public key definitely belongs to the specified owner. If you look on a public certificate server, you may notice that certain certificates, such as that of Phil Zimmermann, who created the first PGP product, contain many signatures.

Some PGP certificates consist of a public key with several labels, each of which contains a different means of identifying the key's owner (for example, the owner's name and corporate email account, the owner's nickname and home email account, a photograph of the owner—all in one certificate). The list of signatures of each of those identities may differ; signatures attest to the authenticity that one of the labels belongs to the public key, not that all the labels on the key are authentic. (Note that "authentic" is in the eye of its beholder—signatures are opinions, and different people devote different levels of due diligence in checking authenticity before signing a key.)

public key

- PGP version number
- time when key created
- how long key is valid
- key type (DH, RSA)
- the key material itself

user id

signature

user id

- string identifying the
  key's owner

signature

- certification that the userid
  and key go together
- version number
- message digest algorithm
- message digest calculation
- signed message digest
- signer key id

X.509 certificate format

X.509 is another very common certificate format. All X.509 certificates comply with the ITU-T X.509 international standard; thus (theoretically) X.509 certificates created for one application can be used by any application complying with X.509. In practice, however, different companies have created their own extensions to X.509 certificates, not all of which work together.

A certificate requires someone to validate that a public key and the name of the key's owner go together. With PGP certificates, anyone can play the role of validator (unless this option is explicitly limited by the company's administrators). With X.509 certificates, the validator is always a Certification Authority or someone designated by a CA. (Bear in mind that PGP certificates also fully support a hierarchical structure using a CA to validate certificates.)

An X.509 certificate is a collection of a standard set of fields containing information about a user or device and their corresponding public key. The X.509 standard defines what information goes into the certificate, and describes how to encode it (the data format).

All X.509 certificates have the following data:

- **The certificate holder's public key**—the public key of the certificate holder, together with an algorithm identifier that specifies which cryptosystem the key belongs to and any associated key parameters.
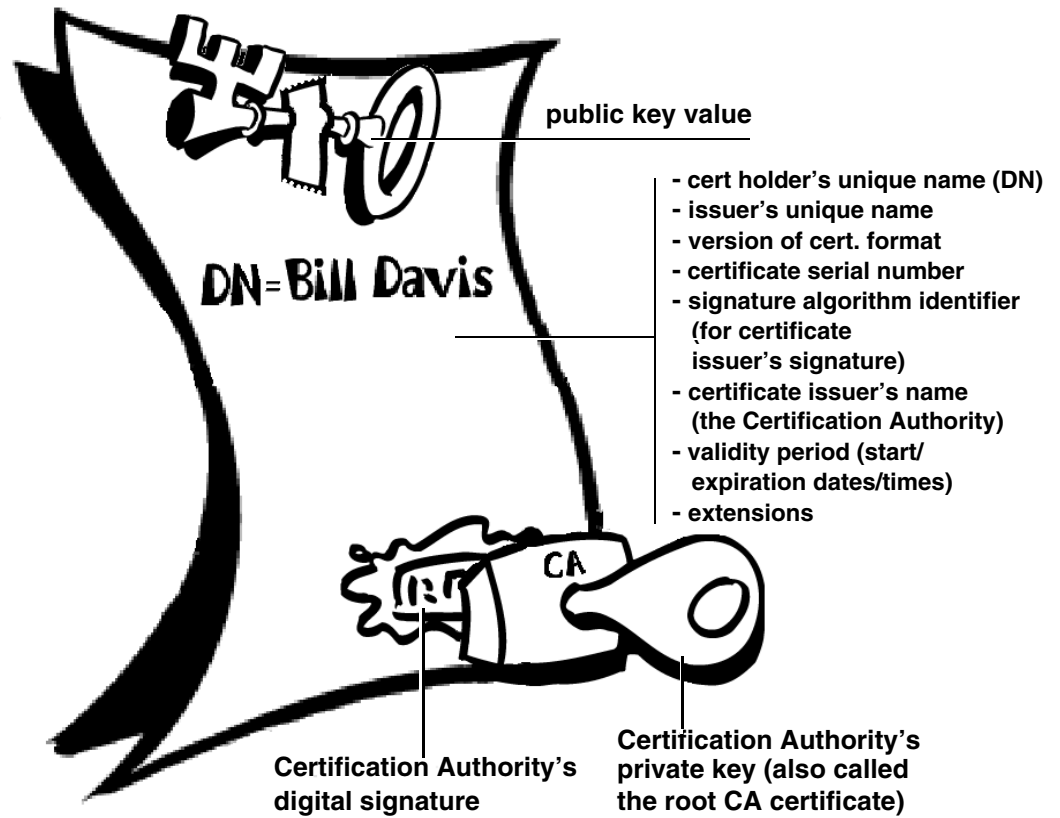
- **The serial number of the certificate**—the entity (application or person) that created the certificate is responsible for assigning it a unique serial number to distinguish it from other certificates it issues. This information is used in numerous ways; for example when a certificate is revoked, its serial number is placed on a Certificate Revocation List (CRL).

- **The certificate holder's unique identifier** (or DN—distinguished name)—this name is intended to be unique across the Internet. A DN consists of multiple subsections and may look something like this:

  CN = Bob Davis, EMAIL = bdavis@pgp.com, OU = PGP Engineering, O = PGP Corporation, C = US

  (These refer to the subject's **C**ommon **N**ame, **O**rganizational **U**nit, **O**rganization, and **C**ountry.)

- The certificate's validity period—the certificate's start date/time and expiration date/time; indicates when the certificate will expire.

- The unique name of the certificate issuer—the unique name of the entity that signed the certificate. This is normally a CA. Using the certificate implies trusting the entity that signed this certificate. (Note that in some cases, such as root or top-level CA certificates, the issuer signs its own certificate.)

- The digital signature of the issuer—the signature using the private key of the entity that issued the certificate.

- The signature algorithm identifier—identifies the algorithm used by the CA to sign the certificate.

There are many differences between an X.509 certificate and a PGP certificate, but the most salient are as follows:

- you can create your own PGP certificate; you must request and be issued an X.509 certificate from a Certification Authority

- X.509 certificates natively support only a single name for the key's owner

- X.509 certificates support only a single digital signature to attest to the key's validity

To obtain an X.509 certificate, you must ask a CA to issue you one. You provide your public key, proof that you possess the corresponding private key, and some specific information about yourself. You then digitally sign the information and send the whole package—the certificate request—to the CA. The CA then performs some due diligence in verifying that the information you provided is correct and, if so, generates the certificate and returns it.

You might think of an X.509 certificate as looking like a standard paper certificate (similar to one you might have received for completing a class in basic First Aid) with a public key taped to it. It has your name and some information about you on it, plus the signature of the person who issued it to you.

public key value

- cert holder's unique name (DN)
- issuer's unique name
- version of cert. format
- certificate serial number
- signature algorithm identifier (for certificate issuer's signature)
- certificate issuer's name (the Certification Authority)
- validity period (start/ expiration dates/times)
- extensions

DN=Bill Davis

Certification Authority's digital signature

Certification Authority's private key (also called the root CA certificate)

Probably the most widely visible use of X.509 certificates today is in Web browsers.

Validity and trust

Every user in a public key system is vulnerable to mistaking a phony key (certificate) for a real one. Validity is confidence that a public key certificate belongs to its purported owner. Validity is essential in a public key environment where you must constantly establish whether or not a particular certificate is authentic.

When you've assured yourself that a PGP key belonging to someone else is valid, you can sign the copy on your keyring to attest to the fact that you've checked it and that it's an authentic one. If you want others to know that you gave the certificate your stamp of approval, you can export the signature to a directory server so that others can see it.

As described in the section "Public Key Infrastructures (PKIs)" some companies designate one or more Certification Authorities (CAs) to indicate certificate validity. In an organization using a PKI with X.509 certificates, it is the job of the RAs to approve certificate requests and the job of the CAs to issue certificates to users—a process which generally entails responding to a user's request for a certificate. In an organization using PGP certificates without a

PKI, it is the job of the CA to check the authenticity of all PGP certificates and then sign the good ones. Basically, the main purpose of a CA is to bind a public key to the identification information contained in the certificate and thus assure third parties that some measure of care was taken to ensure that this binding of the identification information and key is valid.

The CA is the Grand Pooh-bah of validation in an organization; someone whom everyone trusts, and in some organizations, like those using a PKI, no certificate is considered valid unless it has been signed by a trusted CA.

Checking validity

One way to establish validity is to go through some manual process. There are several ways to accomplish this. You could require your intended recipient to physically hand you a copy of his or her public key. But this is often inconvenient and inefficient.

Another way is to manually check the certificate's fingerprint. Just as every human's fingerprints are unique, every PGP certificate's fingerprint is unique. The fingerprint is a hash of the user's certificate and appears as one of the certificate's properties. In PGP software, the fingerprint can appear as a hexadecimal number or a series of so-called biometric words, which are phonetically distinct and are used to make the fingerprint identification process a little easier.

You can check that a certificate is valid by calling the key's owner (so that you originate the transaction) and asking the owner to read his or her key's fingerprint to you and verifying that fingerprint against the one you believe to be the real one. This works if you know the owner's voice, but, how do you manually verify the identity of someone you don't know? Some people put the fingerprint of their key on their business cards for this very reason.

Another way to establish validity of someone's certificate is to trust that a third individual has gone through the process of validating it.

A CA, for example, is responsible for ensuring that prior to issuing to a certificate, he or she carefully checks it to be sure the public key portion really belongs to the purported owner. Anyone who trusts the CA will automatically consider any certificates signed by the CA to be valid.

Another aspect of checking validity is to ensure that the certificate has not been revoked. For more information, see "Certificate Revocation".

Establishing trust

You validate certificates. You trust people. More specifically, you trust people to validate other people' certificates. Typically, unless the owner hands you the certificate, you have to go by someone else's word that it is valid.

Meta and trusted introducers

In most situations, people completely trust the CA to establish certificates' validity. This means that everyone else relies upon the CA to go through the whole manual validation process for them. This is fine up to a certain number of users or number of work sites, and then it is not possible for the CA to maintain the same level of quality validation. In that case, adding other validators to the system is necessary.

A CA can also be a meta-introducer. A meta-introducer bestows not only validity on keys, but bestows the ability to trust keys upon others. Similar to the king who hands his seal to his trusted advisors so they can act on his authority, the meta-introducer enables others to act as trusted introducers. These trusted introducers can validate keys to the same effect as that of the meta-introducer. They cannot, however, create new trusted introducers.

Meta-introducer and trusted introducer are PGP terms. In an X.509 environment, the meta-introducer is called the root Certification Authority (root CA) and trusted introducers subordinate Certification Authorities.

The root CA uses the private key associated with a special certificate type called a root CA certificate to sign certificates. Any certificate signed by the root CA certificate is viewed as valid by any other certificate signed by the root. This validation process works even for certificates signed by other CAs in the system—as long as the root CA certificate signed the subordinate CA's certificate, any certificate signed by the CA is considered valid to others within the hierarchy. This process of checking back up through the system to see who signed whose certificate is called tracing a certification path or certification chain.

Trust models

In relatively closed systems, such as within a small company, it is easy to trace a certification path back to the root CA. However, users must often communicate with people outside of their corporate environment, including some whom they have never met, such as vendors, customers, clients, associates, and so on. Establishing a line of trust to those who have not been explicitly trusted by your CA is difficult.

Companies follow one or another trust model, which dictates how users will go about establishing certificate validity. There are three different models:

- Direct Trust
- Hierarchical Trust
- Web of Trust

Direct Trust

Direct trust is the simplest trust model. In this model, a user trusts that a key is valid because he or she knows where it came from. All cryptosystems use this form of trust in some way. For example, in Web browsers, the root Certi-

fication Authority keys are directly trusted because they were shipped by the manufacturer. If there is any form of hierarchy, it extends from these directly trusted certificates.

In PGP software, a user who validates keys herself and never sets another certificate to be a trusted introducer is using direct trust.

user   **user**
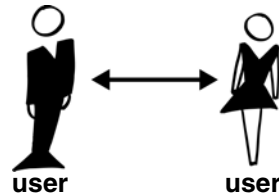
Hierarchical Trust

In a hierarchical system, there are a number of "root" certificates from which trust extends. These certificates may certify certificates themselves, or they may certify certificates that certify still other certificates down some chain. Consider it as a big trust "tree." The "leaf" certificate's validity is verified by tracing backward from its certifier, to other certifiers, until a directly trusted root certificate is found.

meta-introducer (or root CA)

trusted introducers (or CAs)

users
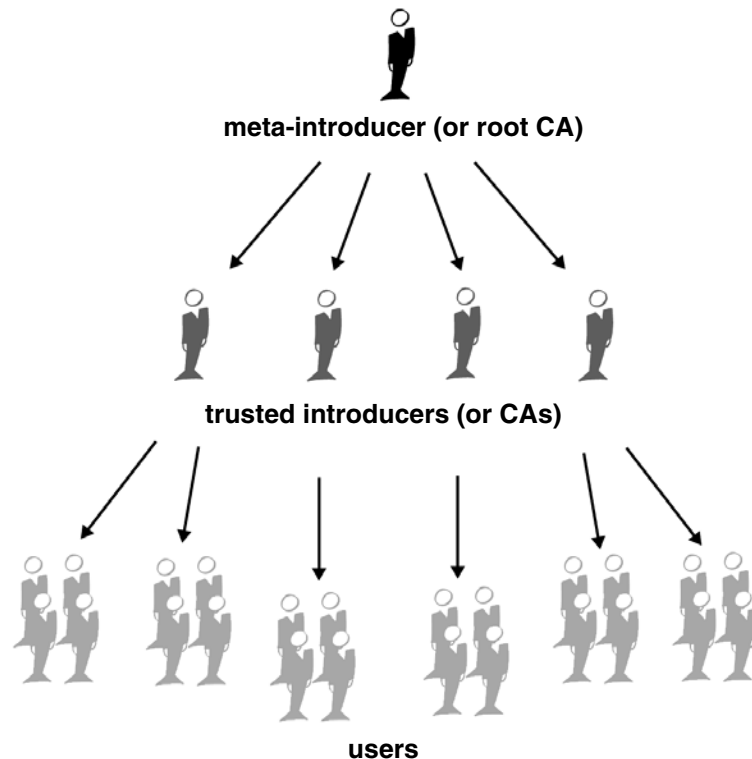
Web of Trust

A Web of trust encompasses both of the other models, but also adds the notion that trust is in the eye of the beholder (which is the real-world view) and the idea that more information is better. It is thus a cumulative trust model. A certificate might be trusted directly, or trusted in some chain going back to a directly trusted root certificate (the meta-introducer), or by some group of introducers.

Perhaps you have heard of the term six degrees of separation, which suggests any person in the world has some link to any other person in the world using six or fewer other people as intermediaries. This is a web of introducers.

It is also the PGP technology view of trust. PGP software uses digital signatures as its form of introduction. When any user signs another's key, he or she becomes an introducer of that key. As this process goes on, it establishes a web of trust.

In a PGP environment, any user can act as a certifying authority. Any PGP user can validate another PGP user's public key certificate. However, such a certificate is only valid to another user if the relying party recognizes the validator as a trusted introducer. (That is, you trust my opinion that others' keys are valid only if you consider me to be a trusted introducer. Otherwise, my opinion on other keys' validity is moot.)

Stored on each user's public keyring are indicators of:

- whether or not the user considers a particular key to be valid
- the level of trust the user places on the key that the key's owner can serve as certifier of others' keys

You indicate, on your copy of my key, whether you think my judgement counts. It's really a reputation system: certain people are reputed to give good signatures, and people trust them to attest to other keys' validity.

Levels of trust

The highest level of trust in a key, implicit trust, is trust in your own key pair. PGP software assumes that if you own the private key, you must trust the actions of its related public key. Any keys signed by your implicitly trusted key are valid.

There are three levels of trust you can assign to someone else's public key:

- Complete trust
- Marginal trust
- No trust (or Untrusted)

To make things confusing, there are also three levels of validity:

- Valid
- Marginally valid
- Invalid

To define another's key as a trusted introducer:

1. Start with a valid key, one that is either:
    - signed by you
    - signed by another trusted introducer
2. Set the level of trust you feel the key's owner is entitled.

For example, suppose your key ring contains Alice's key. You have validated Alice's key and you indicate this by signing it. You know that Alice is a real stickler for validating others' keys. You therefore assign her key with Complete trust. This makes Alice a Certification Authority. If Alice signs another's key, it appears as Valid on your keyring.

PGP requires one completely trusted signature or two marginally trusted signatures to establish a key as valid. PGP's method of considering two marginals equal to one complete is similar to a merchant asking for two forms of ID. You might consider Alice fairly trustworthy and also consider Bob fairly trustworthy. Either one alone runs the risk of accidentally signing a counterfeit key, so you might not place complete trust in either one. However, the odds that both individuals signed the same phony key are probably small.

Certificate Revocation

Certificates are only useful while they are valid. It is unsafe to simply assume that a certificate is valid forever. In most organizations and in all PKIs, certificates have a restricted lifetime. This constrains the period in which a system is vulnerable should a certificate compromise occur.

Certificates are thus created with a scheduled validity period: a start date/ time and an expiration date/time. The certificate is expected to be usable for its entire validity period (its lifetime). When the certificate expires, it will no longer be valid, as the authenticity of its key/identification pair are no longer assured. (The certificate can still be safely used to reconfirm information that was encrypted or signed within the validity period—it should not be trusted for cryptographic tasks moving forward, however.)

There are also situations where it is necessary to invalidate a certificate prior to its expiration date, such as when an the certificate holder terminates employment with the company or suspects that the certificate's corresponding private key has been compromised. This is called revocation. A revoked certificate is much more suspect than an expired certificate. Expired certificates are unusable, but do not carry the same threat of compromise as a revoked certificate.

Anyone who has signed a certificate can revoke their signature on the certificate (provided they use the same private key that created the signature). A revoked signature indicates that the signer no longer believes the public key

and identification information belong together, or that the certificate's public key (or corresponding private key) has been compromised. A revoked signature should carry nearly as much weight as a revoked certificate.

With X.509 certificates, a revoked signature is practically the same as a revoked certificate given that the only signature on the certificate is the one that made it valid in the first place—the signature of the CA. PGP certificates provide the added feature that you can revoke your entire certificate (not just the signatures on it) if you feel the certificate has been compromised.

Only the certificate's owner (the holder of its corresponding private key) or someone whom the certificate's owner has designated as a revoker can revoke a PGP certificate. (Designating a revoker is a useful practice, as it's often the loss of the passphrase for the certificate's corresponding private key that leads a PGP user to revoke his or her certificate—a task that is only possible if one has access to the private key.) Only the certificate's issuer can revoke an X.509 certificate.

Communicating that a certificate has been revoked

When a certificate is revoked, it is important to make potential users of the certificate aware that it is no longer valid. With PGP certificates, the most common way to communicate that a certificate has been revoked is to post it on a certificate server so others who may wish to communicate with you are warned not to use that public key.

In a PKI environment, communication of revoked certificates is most commonly achieved via a data structure called a Certificate Revocation List, or CRL, which is published by the CA. The CRL contains a time-stamped, validated list of all revoked, unexpired certificates in the system. Revoked certificates remain on the list only until they expire, then they are removed from the list—this keeps the list from getting too long.

The CA distributes the CRL to users at some regularly scheduled interval (and potentially off-cycle, whenever a certificate is revoked). Theoretically, this will prevent users from unwittingly using a compromised certificate. It is possible, though, that there may be a time period between CRLs in which a newly compromised certificate is used.

What is a passphrase?

Most people are familiar with restricting access to computer systems via a password, which is a string of characters a user types in as identification.

A passphrase is a longer version of a password, and in theory, a more secure one. Typically composed of multiple words, a passphrase is more secure against standard dictionary attacks, wherein the attacker tries all the words in

the dictionary in an attempt to determine your password. The best pass-phrases are relatively long and complex and contain a combination of upper and lowercase letters, numeric and punctuation characters.

PGP software uses a passphrase to encrypt your private key on your machine. Your private key is encrypted on your disk using a hash of your passphrase as the secret key. You use the passphrase to decrypt and use your private key. A passphrase should be hard for you to forget and difficult for others to guess. It should be something already firmly embedded in your long-term memory, rather than something you make up from scratch. Why? Because if you forget your passphrase, you are out of luck. Your private key is totally and absolutely useless without your passphrase and nothing can be done about it. Remember the quote earlier in this chapter? PGP technology is cryptography that will keep major governments out of your files. It will certainly keep you out of your files, too. Keep that in mind when you decide to change your passphrase to the punchline of that joke you can never quite remember.

Key splitting

They say that a secret is not a secret if it is known to more than one person. Sharing a private keypair poses such a problem. While it is not a recommended practice, sharing a private keypair is necessary at times. Corporate Signing Keys, for example, are private keys used by a company to sign—for example—legal documents, sensitive personnel information, or press releases to authenticate their origin. In such a case, it is worthwhile for multiple members of the company to have access to the private key. However, this means any single individual can act fully on behalf of the company.

In such a case it is wise to split the key among multiple people in such a way that more than one or two people must present a piece of the key in order to reconstitute it to a usable condition. If too few pieces of the key are available, then the key is unusable.

Some examples are to split a key into three pieces and require two of them to reconstitute the key, or split it into two pieces and require both pieces. If a secure network connection is used during the reconstitution process, the key's shareholders need not be physically present in order to rejoin the key.

Technical details

This chapter provided a high-level introduction to cryptographic concepts and terminology. In Chapter 3, "Phil Zimmermann on PGP" Phil Zimmermann, the creator of PGP software, provides a more in-depth discussion of privacy, the technical details of how PGP works, including the various algorithms it uses, as well as various attacks and how to protect yourself against them.

2 The Self-Managing Security Architecture

This chapter describes a system for protecting important information while it is in transit that is easy to deploy and maintain, requires no effort on the part of end users, automatically assembles itself but gives the administrator complete control, and includes options for dealing with users outside the system.

Introduction

Beginning in the 1990s, more and more important information—health records, competitive data, intellectual property, customer and employee information, financial records, and so on—has been created, transmitted, and stored in digital format. This information, and many other kinds of private information, are routinely transmitted across insecure networks like the Internet completely unprotected.

And while companies, governments, and individuals want to keep this sensitive information private, only 5 to 15 percent of computers that are transmitting this information have secure-messaging capabilities. The end result: your sensitive information is extremely vulnerable and could easily fall into the wrong hands.

One potential solution to this problem is a public-key infrastructure (PKI). PKIs, which are described in "Public Key Infrastructures (PKIs)" on page 21, are systems for verifying the identity (using certificates) of other users in the system so that you can send them messages and attachments protected by encryption.

A PKI helps to ensure that the right person gets the message, that the message wasn't modified while being sent, and that, if stolen, the message can't be used by the person who stole it.

And while PKIs generally do what they're supposed to do, implementing a PKI results in a whole different set of problems. It turns out that a PKI:

- is time consuming to implement
- requires significant ongoing maintenance
- works only with other members of the PKI
- is frequently not used because protecting a message requires effort on the part of the end user

A Change in Thinking

"Problems cannot be solved at the same level of awareness that created them."

—Albert Einstein

Clearly the PKI, while doing what it is supposed to do, is not the ideal solution for providing secure messaging. A more comprehensive solution is required. A solution that includes what a PKI does right but also includes what a PKI lacks.

Secure messaging requires a solution that:

• is easy to deploy and maintain

• requires no effort on the part of end users

• automatically assembles itself but gives the administrator complete control

• includes options for users outside the system

That solution is a self-managing security architecture.

Self-Managing Security Architecture (SMSA)

A SMSA does what a PKI does—provides strong security for a messaging system using public-key cryptography—but it also uses a different metaphor and includes other capabilities.

A Different Metaphor

The traditional PKI uses a telephone system metaphor. Alice is enrolled in the system, so she has a phone she can use to call other members in the system, and she has a phone number that other members of the system use to call her. The PKI is the telephone operator; when Alice wants to "talk" to Bob, the telephone operator determines that both are part of the system and connects them. If either of them don't have a phone or a phone number, no communication is possible. It's all or nothing: unless everything is correct, no communication.

The SMSA uses a different metaphor, a postal metaphor. The SMSA is a messenger whose job it is to deliver messages as securely as possible. When it gets a message destined for a particular recipient, the messenger knows it can get the message through as plain text, and it will, if there's no other way. But its job is to add security to that delivery, to make it as secure as possible. The SMSA uses everything it knows about the sender and the recipient to securely deliver that message.

We use the metaphor of the messenger because of the realities of how users use Internet communications. When someone sends an email message, they expect it to be delivered quickly. Security is a feature they desire, but quick delivery is what they really want.

People need to communicate more than they need to communicate securely. We may not like this, but it's our job to get their messages to their destinations as securely as possible rather than block them because they're not secure enough. If, as their telephone operator, we refuse to put their call through, they won't decide they didn't need to make the call, they'll just find another, less secure way, of doing it.

The SMSA metaphor of the messenger is the attitude that the mail must go through. It makes the SMSA an enabling technology rather than a disabling technology.

Self assembly

If a security system requires that it be built from scratch and constantly maintained, it's more trouble than it's worth. This is one of the major problems of the traditional PKI.

The SMSA avoids this problem by building in intelligence such that the SMSA automatically uses the existing network structure and processes to self assemble.

The SMSA gets into the middle of the network processes, monitoring and observing them, assembling itself based on what is actually happening in the network.

Self assembly not only makes the job of the SMSA administrator easier, it also changes their from that of constructing and constantly maintaining the system to that of overseeing it. The administrator establishes policies, fixes errors, and shapes the SMSA to the larger system it serves.

The net effect of self assembly is that the boring, time consuming, repetitive tasks are handled by the computers, which are ideally suited for them, freeing up the human administrators to do what they alone can do, understand the complexity and mold the system to its ideal form.

Ease of deployment

A solution that is difficult, costly, or time consuming to deploy either won't or shouldn't be deployed. This is one factor that weighs down traditional PKIs. To be effective, the SMSA must be easy to add into the existing infrastructure and to maintain.

The SMSA accomplishes this by inserting itself into the existing network infrastructure. Once installed and configured, the SMSA self assembles and requires only oversight from its administrator; mainly establishing and modifying policies and fixing errors that crop up occasionally.

By taking deployment out of the hands of end users and putting it into the network infrastructure, the SMSA ensures wide deployment by virtue of easy deployment.

Transparency to end users

If users won't use a secure messaging system because it's too difficult, they're afraid of it, or they don't understand it, then the system has failed.

And with traditional PKIs, this is too often the case. Studies indicate that five percent or less of users use secure messaging, mostly because they find it difficult to do so.

How do you deal with end users who need secure messaging but who don't want to actually make any effort to get it? You provide it to them without requiring any effort or thought on their part; by making it transparent to them.

The SMSA sits within the existing network and proxies existing protocols, providing security and modifying and deleting users as necessary: all with no thought or effort required by the end user.

In certain situations, it may be beneficial for client software to reside on the computer of the end user. Client software can extend security all the way to the computer of the end user and may offload work from the computer hosting the SMSA. While client software needs to be installed, once set up it also operates transparently.

Policy driven

Every company that uses email has, or should have, policies that dictate how corporate email can be used. Because the SMSA acts on behalf of its users to send and receive email, it too must have policies the administrator can control so that the SMSA's policies are aligned with corporate security policies.

Because these policies must be able to handle every situation the SMSA encounters, policy management must be flexible but comprehensive.

There must be a default policy, which applies if no other policy applies. There must be mechanisms to create policies for specific domains that include a range of options for securing messages. There must be ways to create different policies for different parts of a domain. Finally, there must be a flow to how specific situations are matched to existing policies such that the first policy that applies is implemented, falling back to the default policy if no existing policy matches.

These policy mechanisms are mandatory for the SMSA to be able to provide flexible and comprehensive policy management that enables the administrator to align SMSA actions with corporate security policies.

High-quality logging

As good as the SMSA is at self assembling based on what the network is doing and no matter how good existing policies are in determining how the SMSA handles messages, some things are going to happen that the administrator didn't anticipate and will need to change.

By providing extensive, high-quality logs, the SMSA puts the administrator in the oversight role, detecting and correcting the things that need to be changed.

Log entries must be understandable or they're of no value. Log entries must be human readable, categorized as to their severity, and sorted into logical categories. Finally, they must be exportable to a log file for later analysis, if desired by the administrator.

Extensive, high-quality logging is required so that the administrator can play their part of overseeing the SMSA and modifying it where needed.

Learn mode

Clausewitz said that no battle plan survives contact with the enemy. Similarly, no security policy survives contact with actual users. Consequently, it is imperative to be able to test policies before they're implemented. The SMSA administrator needs to know what's going to happen to the corporate email system when the SMSA is made active.

This vital feature of the SMSA is called Learn Mode. Its goal is to speed deployment, as people will deploy faster what they understand better.

When the SMSA is installed, it starts by default in Learn Mode, which is a special operational mode where the SMSA begins monitoring and proxying traffic normally but does not modify any messages.

Learn Mode gives the SMSA the opportunity to learn the network environment (creating keys for users, for example) without actually affecting email traffic, so that when Learn Mode is turned off it knows the environment and can immediately begin securing messages.

Checking the logs while the SMSA is in Learn Mode shows what it would be doing to email traffic if it were live. Learn Mode gives the SMSA administrator the perfect opportunity to fine tune the SMSA's policies until things are working as required.

Support for users outside the SMSA

No doubt many of the messages protected by the SMSA will be exchanged between existing members. But the world is interconnected; the SMSA must include a mechanism for securing messages to and from email users who aren't currently part of the SMSA, either allowing secure message exchange or making those external users part of the SMSA.

How does the SMSA handle these users? Two policy options come immediately to mind: First, some messages are simply OK to send in the clear; they are sent as plain text. Second, on the other end of the security spectrum, some messages must not be sent at all; these messages are bounced back to the sender.

And while these two options are important, clearly they are inadequate to handle all cases.

One other important option is called the Smart Trailer. The message is sent to the recipient in the clear, but a trailer is added at the end that explains to the recipient that future messages from the sender can be read securely. A URL in the Smart Trailer takes the recipient to a site where she can choose among various methods of securing future messages.

Another option is called PGP Universal Web Messenger. The actual message is stored on a secure server and the recipient receives a separate message explaining how they can access the original message securely. When the recipient accesses their message on the secure server, they are presented with the same options for securing future messages as are available with the Smart Trailer.

Dealing with users outside the SMSA is obviously required in today's interconnected world and, in the case of the Smart Trailer and PGP Universal Web Messenger, actually servers to further grow the SMSA.

PGP Universal

PGP Universal is a product implementation of a SMSA. It takes the objectives of a SMSA, adds some extra features, and makes it a working system for secure messaging.

More information about PGP Universal is available on the PGP Corporation Web site at www.pgp.com.

The main features of PGP Universal include:

- **PGP Universal Server:** A physical device you add to your network that automatically creates and maintains a SMSA by monitoring authenticated users and their email traffic.

- **Administrative Interface:** Each PGP Universal Server is controlled via a Web-based administrative interface. The administrative interface gives the administrator complete control over the server and the SMSA.

- **Self-assembling:** Behind the scenes, the PGP Universal Server creates and manages the SMSA by monitoring authenticated users and their email traffic. The administrator has complete oversight of the system so that if a user already has a key (such as a PGP Desktop key), for example, that key can be manually added to the SMSA.

- **Learn Mode:** When you finish configuring a server, it begins operation in Learn Mode, which is a special mode where the server proxies traffic normally but doesn't encrypt or sign any messages.

  Learn Mode gives the server a chance to build its SMSA (creating keys for authenticated users, for example) so that when the server goes live— that is, when Learn Mode is turned off—the server knows the environment and can immediately begin securing messages.

  The administrator can check the server's logs while it is in Learn Mode to see what it would be doing to email traffic if it were live. The administrator can then make changes to the server's policies while it is in Learn Mode until things are working as expected.

- **Outside Users:** What happens when secure email needs to be sent by someone in your organization to someone ***not*** part of the SMSA?

  Options include: bounce the message back to the sender (so it's not sent unencrypted), send unencrypted, Smart Trailer (which sends the message unencrypted but gives the recipient the option of joining the SMSA), and PGP Universal Web Messenger mail (which lets the recipient securely read the message and gives them the option of joining the SMSA).

- **PGP Universal Satellite:** The PGP Universal Satellite software allows email to be encrypted all the way to and from the desktop, and it is one way for outside users to participate in the SMSA. It also allows users the option of controlling their keys locally (if allowed by the PGP Universal administrator).

- **Mail Processing Modes:** The PGP Universal Server can be located either inside of your mail server or outside, which adds extra security and flexibility in configuration.

  When placed internally, the PGP Universal Server logically sits between the email users and the mail server. The server encrypts and signs outgoing SMTP email and decrypts and verifies incoming POP or IMAP email. Email stored on the mail server is stored encrypted.

  When placed externally, the PGP Universal Server logically sits between the mail server and the Internet. It encrypts and signs outgoing SMTP email and decrypts and verifies incoming SMTP email. Email stored on the mail server is stored unprotected.

- **Cluster:** When you have two or more servers in your network, you configure them to synchronize users, keys, managed domains, and policies with each other; this is called a "cluster."

- **Organization Key:** The Organization Key is used to sign all user keys the server creates and to encrypt server backups.

- **Backup and Restore:** Because full backups of the data stored on the PGP Universal Server are critical in the case of a natural disaster or other unanticipated loss of data or hardware, you can make backups of data on the server and restore from a backup if necessary. All backups are encrypted to the Organization Key and may thus be stored securely off the server.

- **Directory Synchronization:** Existing LDAP directories (Microsoft Active Directory, for example) can be used to create PGP Universal's internal users, including import of X.509 certificates for S/MIME support.

3   Phil Zimmermann on PGP

This chapter contains information about cryptography and PGP as written by PGP's creator, Phil Zimmermann.

Why I wrote PGP

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

—Mahatma Gandhi

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

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

But with the coming of the information age, starting with the invention of the telephone, all that has changed. Now most of our conversations are conducted electronically. This allows our most intimate conversations to be exposed without our knowledge. Cellular phone calls may be monitored by anyone with a radio. Electronic mail, sent across the Internet, is no more secure than cellular phone calls. Email is rapidly replacing postal mail, becoming the norm for everyone, not the novelty it was in the past.

Until recently, if the government wanted to violate the privacy of ordinary citizens, they had to expend a certain amount of expense and labor to intercept and steam open and read paper mail. Or they had to listen to and possibly transcribe spoken telephone conversation, at least before automatic voice recognition technology became available. This kind of labor-intensive monitoring was not practical on a large scale. It was only done in important cases when it seemed worthwhile. This is like catching one fish at a time, with a hook and line. Today, email can be routinely and automatically scanned for interesting keywords, on a vast scale, without detection. This is like driftnet fishing. And exponential growth in computer power is making the same thing possible with voice traffic.

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

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

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

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

A year after the CALEA passed, the FBI disclosed plans to require the phone companies to build into their infrastructure the capacity to simultaneously wiretap one percent of all phone calls in all major U.S. cities. This would represent more than a thousandfold increase over previous levels in the number of phones that could be wiretapped. In previous years, there were only about a thousand court-ordered wiretaps in the United States per year, at the federal, state, and local levels combined. It's hard to see how the government

could even employ enough judges to sign enough wiretap orders to wiretap one percent of all our phone calls, much less hire enough federal agents to sit and listen to all that traffic in real time. The only plausible way of processing that amount of traffic is a massive Orwellian application of automated voice recognition technology to sift through it all, searching for interesting key-words or searching for a particular speaker's voice. If the government doesn't find the target in the first one percent sample, the wiretaps can be shifted over to a different one percent until the target is found, or until everyone's phone line has been checked for subversive traffic. The FBI said they need this capacity to plan for the future. This plan sparked such outrage that it was defeated in Congress. But the mere fact that the FBI even asked for these broad powers is revealing of their agenda.

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

You don't have to distrust the government to want to use cryptography. Your business can be wiretapped by business rivals, organized crime, or foreign governments. Several foreign governments, for example, admit to using their signals intelligence against companies from other countries to give their own corporations a competitive edge. Ironically, the United States government's restrictions on cryptography in the 1990s have weakened U.S. corporate defenses against foreign intelligence and organized crime.

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

The government initially claimed that using Clipper would be voluntary, that no one would be forced to use it instead of other types of cryptography. But the public reaction against the Clipper chip was strong, stronger than the government anticipated. The computer industry monolithically proclaimed its opposition to using Clipper. FBI director Louis Freeh responded to a question in a press conference in 1994 by saying that if Clipper failed to gain public support, and FBI wiretaps were shut out by non-government-controlled cryptography, his office would have no choice but to seek legislative relief. Later,

in the aftermath of the Oklahoma City tragedy, Mr. Freeh testified before the Senate Judiciary Committee that public availability of strong cryptography must be curtailed by the government (although no one had suggested cryptography was used by the bombers).

The government has a track record that does not inspire confidence that they will never abuse our civil liberties. The FBI's COINTELPRO program targeted groups that opposed government policies. They spied on the antiwar movement and the civil rights movement. They wiretapped the phone of Martin Luther King, Jr. Nixon had his enemies list. Then there was the Watergate mess. More recently, Congress has either attempted to or succeeded in passing laws curtailing our civil liberties on the Internet. Some elements of the Clinton White House collected confidential FBI files on Republican civil servants, conceivably for political exploitation. And some overzealous prosecutors have shown a willingness to go to the ends of the Earth in pursuit of exposing sexual indiscretions of political enemies. At no time in the past century has public distrust of the government been so broadly distributed across the political spectrum as it is today.

Throughout the 1990s, I figured that if we want to resist this unsettling trend in the government to outlaw cryptography, one measure we can apply is to use cryptography as much as we can now while it's still legal. When use of strong cryptography becomes popular, it's harder for the government to criminalize it. Therefore, using PGP is good for preserving democracy. If privacy is outlawed, only outlaws will have privacy.

It appears that the deployment of PGP must have worked, along with years of steady public outcry and industry pressure to relax the export controls. In the closing months of 1999, the Clinton administration announced a radical shift in export policy for crypto technology. They essentially threw out the whole export control regime. Now, we are finally able to export strong cryptography, with no upper limits on strength. It has been a long struggle, but we have finally won, at least on the export control front in the U.S. Now we must continue our efforts to deploy strong crypto, to blunt the effects increasing surveillance efforts on the Internet by various governments. And we still need to entrench our right to use it domestically over the objections of the FBI.

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.

The PGP symmetric algorithms

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

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

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

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

As a hedge, PGP includes three-key Triple-DES in its repertoire of available block ciphers. The DES was developed by IBM in the mid-1970s. While it has a good design, its 56-bit key size is too small by today's standards. Triple-DES is very strong, and has been well studied for many years, so it might be a safer bet than the newer ciphers such as CAST and IDEA. Triple-DES is the DES applied three times to the same block of data, using three different keys, except that the second DES operation is run backwards, in decrypt mode. While Triple-DES is much slower than either CAST or IDEA, speed is usually not critical for email applications. Although Triple-DES uses a key size

of 168 bits, it appears to have an effective key strength of at least 112 bits against an attacker with impossibly immense data storage capacity to use in the attack. According to a paper presented by Michael Weiner at Crypto96, any remotely plausible amount of data storage available to the attacker would enable an attack that would require about as much work as breaking a 129-bit key. Triple-DES is not encumbered by any patents.

Starting with PGP Version 7.0, we introduced Bruce Schneier's Twofish algorithm. Twofish was one of the five finalist algorithms in the NIST Advanced Encryption Standard (AES) project. The AES is a new block cipher design, with a 128-bit block size, and key sizes in 128, 192, or 256 bits. Fifteen design teams from around the world submitted candidate algorithms when NIST asked for competitive public submissions in 1996, and NIST selected the best five of them in 1998.

The five finalists were Twofish, Serpent, Rijndael, RC6, and MARS. All of the top five AES finalists have received intense cryptanalytic scrutiny from the best cryptographers in the world, many of whom have AES submissions of their own that compete with the others. NIST selected Rijndael as the winner from these five excellent algorithms. Rijndael is a block cipher designed by Joan Daemen and Vincent Rijmen. Rijndael is included starting in PGP Version 7.1. For further details on the AES, see www.nist.gov/aes.

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

About PGP data compression routines

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

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

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

About the random numbers used as session keys

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

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

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

About the message digest

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

The message digest algorithm now used in PGP (version 5.0 and later) is called SHA-1, which stands for Secure Hash Algorithm, designed by the NSA for the National Institute of Standards and Technology (NIST). SHA-1 is a 160-bit hash algorithm. Some people might regard anything from the NSA

with suspicion, because the NSA is in charge of intercepting communications and breaking codes. But keep in mind that the NSA has no interest in forging signatures, and the government would benefit from a good unforgeable digital signature standard that would preclude anyone from repudiating their signatures. That has distinct benefits for law enforcement and intelligence gathering. Also, SHA-1 has been published in the open literature and has been extensively peer-reviewed by most of the best cryptographers in the world who specialize in hash functions, and the unanimous opinion is that SHA-1 is extremely well designed. It has some design innovations that overcome all the observed weaknesses in message digest algorithms previously published by academic cryptographers. All new versions of PGP use SHA-1 as the message digest algorithm for creating signatures with the new DSS keys that comply with the NIST Digital Signature Standard. For compatibility reasons, new versions of PGP still use MD5 for RSA signatures, because older versions of PGP used MD5 for RSA signatures.

The message digest algorithm used by older versions of PGP is the MD5 Message Digest Algorithm, placed in the public domain by RSA Data Security, Inc. MD5 is a 128-bit hash algorithm. In 1996, MD5 was all but broken by a German cryptographer, Hans Dobbertin. Although MD5 was not completely broken at that time, it was discovered to have such serious weaknesses that no one should keep using it to generate signatures. Further work in this area might completely break it, allowing signatures to be forged. If you don't want to someday find your PGP digital signature on a forged confession, you might be well advised to migrate to the new PGP DSS keys as your preferred method for making digital signatures, because DSS uses SHA-1 as its secure hash algorithm.

How to protect public keys from tampering

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

Suppose you want to send a private message to Alice. You download Alice's public key certificate from a key server. You encrypt your letter to Alice with this public key and send it to her through email.

Unfortunately, unbeknownst to you or Alice, another user named Charlie has infiltrated the key server site 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. Everything looks normal because this bogus key has Alice's user ID. Now Charlie can decipher the message intended for Alice because he has the matching private key. He may even

re-encrypt the deciphered message with Alice's real public key and send it on to her so that no one suspects any wrongdoing. Furthermore, he can even make apparently good signatures from Alice with this private key because everyone will use the bogus public key to check Alice's signatures.

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

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

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

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

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

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

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

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

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

No matter how tempted you are, you should never give in to expediency and trust a public key you downloaded from a key server or Web site, 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 key server or Web site.

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

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

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

It would be a good idea to keep your own public key on hand with a collection of certifying signatures attached from a variety of "introducers," in the hope that most people will trust at least one of the introducers who vouch for the validity of your public key. You could post your key with its attached collec-

tion of certifying signatures on various key servers. If you sign someone else's public key, return it to them with your signature so that they can add it to their own collection of credentials for their own public key.

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

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

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

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

How does PGP keep track of which keys are valid?

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

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

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

• Does the key actually belong to the person to whom it appears to belong? In other words, has it been certified with a trusted signature?

• Does it belong to someone you can trust to certify other keys?

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

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

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

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

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

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

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

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

How to protect private keys from disclosure

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

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

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

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

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

If the worst case happens—your private key and passphrase are both compromised (hopefully you will find this out somehow)—you will have to issue a "key revocation" certificate. This kind of certificate is used to warn other people to stop using your public key. You can use PGP software to create such a certificate by using the Revoke command or by having your Designated Revoker do it for you. Then you must send this to a key server so others can find it. Their own PGP software installs this key revocation certificate on their public keyrings and automatically prevents them from accidentally using your

public key ever again. You can then generate a new private/public key pair and publish the new public key. You could send out one package containing both your new public key and the key revocation certificate for your old key.

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

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

Beware of snake oil

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

When I was in college in the early 1970s, I devised what I believed was a brilliant encryption scheme. A simple pseudorandom number stream was added to the plaintext stream to create ciphertext. This would seemingly thwart any frequency analysis of the ciphertext, and would be uncrackable even to the most resourceful government intelligence agencies. I felt so smug about my achievement.

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

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

This is like selling automotive seat belts that look good and feel good, but snap open in the slowest crash test. Depending on them may be worse than not wearing seat belts at all. No one suspects they are bad until a real crash.

Depending on weak cryptographic software may cause you to unknowingly place sensitive information at risk when you might not otherwise have done so if you had no cryptographic software at all. Perhaps you may never even discover that your data has been compromised.

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

Unfortunately, most of the commercial encryption packages I've looked at (in the early 1990s when I first developed PGP) use ECB mode. When I've talked to the authors of a number of these implementations, they say they've never heard of CBC or CFB modes, and don't know anything about the weaknesses of ECB mode. The very fact that they haven't even learned enough cryptography to know these elementary concepts is not reassuring. And they sometimes manage their DES keys in inappropriate or insecure ways. Also, these same software packages often include a second faster encryption algorithm that can be used instead of the slower DES. The author of the package often thinks his proprietary faster algorithm is as secure as DES, but after questioning him I usually discover that it's just a variation of my own brilliant scheme from college days. Or maybe he won't even reveal how his proprietary encryption scheme works, but assures me it's a brilliant scheme and I should trust it. I'm sure he believes that his algorithm is brilliant, but how can I know that without seeing it?

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

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

There is a company called AccessData (http://www.accessdata.com) that sells a very low-cost package that cracks the built-in encryption schemes used by WordPerfect, Lotus 1-2-3, MS Excel, Symphony, Quattro Pro, Paradox, Microsoft Word, and PKZIP. It doesn't simply guess passwords; it does real cryptanalysis. Some people buy it when they forget their password for their own files. Law enforcement agencies buy it too, so they can read files they seize. I talked to Eric Thompson, the author, and he said his program only takes a split second to crack them, but he put in some delay loops to slow it down so it doesn't look so easy to the customer.

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

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

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

I remember a conversation in 1991 with Brian Snow, a highly placed senior cryptographer with the NSA. He said he would never trust an encryption algorithm designed by someone who had not "earned their bones" by first spending a lot of time cracking codes. That made a lot of sense. I observed that practically no one in the commercial world of cryptography 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 United States sold German Enigma ciphering machines to third-world governments. But they didn't tell them that the Allies cracked the Enigma code during the war, a fact that remained classified for many years. Even today, many UNIX systems worldwide use the Enigma cipher for file encryption, in part because the government has created legal obstacles against using better algorithms. They even tried to prevent the initial publication of the RSA algorithm in 1977. And they have for many years squashed essentially all commercial efforts to develop effective secure telephones for the general public.

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

At the time of PGP's initial publication, there were three main factors that had undermined the quality of commercial cryptographic software in the United States:

- The first is the virtually universal lack of competence of implementors of commercial encryption software (although this started to change after the publication of PGP). Every software engineer fancies himself a cryptographer, which has led to the proliferation of really bad crypto software.

- The second is the NSA deliberately and systematically suppressing all the good commercial encryption technology, by legal intimidation and economic pressure. Part of this pressure is brought to bear by stringent export controls on encryption software which, by the economics of software marketing, had the net effect of suppressing domestic encryption software.

- The third principle method of suppression comes from the granting of all the software patents for all the public key encryption algorithms to a single company, affording a single choke point to suppress the spread of this technology (although this crypto patent cartel broke up in the fall of 1995).

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

I'm not as certain about the security of PGP as I once was about my brilliant encryption software from college. If I were, that would be a bad sign. But I don't think PGP contains any glaring weaknesses (although I'm pretty sure it contains bugs). I have selected the best algorithms from the published literature of civilian cryptologic academia. These algorithms have been individually subject to extensive peer review. I know many of the world's leading cryptographers, and have discussed with some of them many of the cryptographic algorithms and protocols used in PGP. It's well researched, and has been

years in the making. And I don't work for the NSA. But you don't have to trust my word on the cryptographic integrity of PGP, because source code is available to facilitate peer review.

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

Vulnerabilities

"If all the personal computers in the world—260 million—were put to work on a single PGP-encrypted message, it would still take an estimated 12 million times the age of the universe, on average, to break a single message."

—William Crowell, Deputy Director, National Security Agency, in Senate testimony on March 20, 1997

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

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

Compromised passphrase and private key

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

Here are some recommendations for protecting your passphrase:

- Don't use obvious passphrases that can be easily guessed, such as the names of your kids or spouse.

- Don't make your passphrase a single word, because it can be easily guessed by having a computer try all the words in the dictionary until it finds your password. That's why a passphrase is so much better than a password. A more sophisticated attacker may have his computer scan a book of famous quotations to find your passphrase, so don't use famous

quotations. You might also consider including non-alphabetic characters such as numbers or punctuation marks in your passphrase, if they don't make it harder to remember.

- Use an easy to remember but hard to guess passphrase. Avoid the need to write down your passphrase by choosing one that you already remember from your old long-term memories, rather than making one up from scratch. If you make one up from scratch, you are more likely to forget it unless you start using it immediately and often. Perhaps you can remember a silly nonsensical saying that came to you late one night back when you were in college, that has somehow persisted in your memory for all these years.

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.

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

Not quite deleted files

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

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

text on the disk, just because of its internal workings. These temporary copies of your text are deleted by the word processor when it's done, but these sensitive fragments are still on your disk somewhere.

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

More modern operating systems sometimes use what is called a Journaling filesystem. This takes the temp file problem you might find in a word processor to the next level by making a second copy of every single item written to the filesystem, which then gets stored in a private filesystem area. This journal of disk writes serves as a map for everything that has changed on the disk over time and allows the filesystem to recover more easily from damage. The NTFS filesystem on Windows NT/2000/XP is an example of such a filesystem. Mac OS X 10.2.2 also has a Journaling extension for HFS filesystems. Journaling filesystems require the use of PGP's Freespace Wipe to properly overwrite these journals.

Viruses and Trojan horses

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

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

If your computer is tied to a network, an attacker might be able to exploit weaknesses in your computer's operating system to break in and insert some hostile software that could attack the integrity of PGP. This could even happen when you are not around.

A similar attack involves someone creating a clever imitation of PGP that behaves like PGP in most respects, but that doesn't work the way it's supposed to. For example, it might be deliberately crippled to not check signatures properly, allowing bogus key certificates to be accepted. This Trojan

horse version of PGP is not hard for an attacker to create, because PGP source code is widely available, so anyone could modify the source code and produce a lobotomized zombie imitation PGP that looks real but does the bidding of its diabolical master. This Trojan horse version of PGP could then be widely circulated, claiming to be from a legitimate source. How insidious.

You should make an effort to get your copy of PGP directly from PGP Corporation.

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

Swap files or virtual memory

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

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

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

memory for very long. Also, where possible, we try to ask the operating system to lock that data in memory and not allow it to be swapped. But we don't make any guarantees.

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

Physical security breach

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

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

With Server Key Mode, PGP Universal stores unencrypted private keys on the PGP Universal Server. Physical security thus becomes a critical factor in such a scenario. Storing such servers in a locked rack inside a secure, guarded data center might be valid considerations for your environment. From one perspective, it's much easier to protect the crown jewels if they're in one well-guarded place rather than pieces distributed among a thousand citizens.

This kind of attack is cheaper than cryptanalytic attacks on PGP, depending on how much effort you put into physical security.

Tempest attacks

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

Some newer versions of PGP (after Version 6.0) can display decrypted plain-text using a specially designed font that may have reduced levels of radio frequency emissions from your computer's video screen. This may make it harder for the signals to be remotely detected. This special font is available in some versions of PGP that support the "Secure Viewer" feature.

Protecting against bogus timestamps

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

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

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

PGP annotates signed text with lines describing when a message was signed and who it was signed by. While PGP Universal and PGP 8.0.3 and above make every attempt to prevent spoofing of these notations, an attacker could still, for instance, try to create a signature annotation that looked similar to a real annotation or use carefully embedded images in an email to trick you into thinking a message was signed when it was not. One way to help defeat this is to select the text of the message. Selected text generally looks very different from a selected image.

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

A trustworthy Certifying Authority or notary could create notarized signatures with a trustworthy timestamp. This would not necessarily require a centralized authority. Perhaps any trusted introducer or disinterested party could serve this function, the same way real notary publics do now. When a notary signs other people's signatures, it creates a signature certificate of a signa-

ture certificate. This would serve as a witness to the signature in the same way that real notaries now witness handwritten signatures. The notary could enter the detached signature certificate (without the actual whole document that was signed) into a special log controlled by the notary. Anyone could read this log. The notary's signature would have a trusted timestamp, which might have greater credibility or more legal significance than the timestamp in the original signature.

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

Exposure on multi-user systems

PGP was originally designed for a single-user PC under your direct physical control. If you run PGP at home on your own PC, not connected to a network, your encrypted files are generally safe, unless someone breaks into your house, steals your PC, and persuades you to give them your passphrase (or your passphrase is simple enough to guess).

PGP is not designed to protect your data while it is in plaintext form on a compromised system. Nor can it prevent an intruder from using sophisticated measures to read your private key while it is being used. An intruder may be able to get into your computer by accessing it over a network, or he may get access because your computer is a timesharing multiuser system that allows users to log in remotely, such as UNIX. You will just have to recognize these risks on multiuser systems, and adjust your expectations and behavior accordingly. Perhaps your situation is such that you should consider only running PGP on an isolated single-user system under your direct physical control.

Traffic analysis

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

Cryptanalysis

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

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

Still, some optimism seems justified. The public key algorithms, message digest algorithms, and block ciphers used in PGP were designed by some of the best cryptographers in the world. PGP's algorithms have had extensive security analysis and peer review from some of the best cryptanalysts in the unclassified world.

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

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

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

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

Glossary

access control

A method of restricting access to resources, allowing access only to entities with the appropriate privileges.

additional decryption key (ADK)

A special key to which messages are encrypted, in addition to the recipient. Using an ADK is a way to recover a message if the recipient is unable or unwilling to so (the holder of the ADK can decrypt any message that was encrypted to the ADK).

Advanced Encryption Standard (AES)

NIST approved encryption standards, usually used for the next 20 to 30 years. Rijndael, a block cipher designed by Joan Daemen and Vincent Rijmen that has 16-byte blocks and can operate with 128, 192, or 256-bit keys, was chosen as the new AES in October 2000.

algorithm (encryption)

A set of mathematical rules (logic) used in the processes of encryption and decryption.

algorithm (hash)

A set of mathematical rules (logic) used in the processes of message digest creation and key/signature generation.

American National Standards Institute (ANSI)

Develops standards through various Accredited Standards Committees (ASC). The X9 committee focuses on security standards for the financial services industry.

anonymity

Of unknown or undeclared origin or authorship, concealing an entity's identification.

API (Application Programming Interface)

Provides the means to take advantage of software features, allowing dissimilar software products to interact upon one another.

ASN.1 (Abstract Syntax Notation One)

ISO/IEC standard for encoding rules used in ANSI X.509 certificates, two types exist: DER (Distinguished Encoding Rules) and BER (Basic Encoding Rules).

asymmetric keys

A separate but integrated user key-pair, comprised of one public key and one private key. Each key is one way, meaning that a key used to encrypt information can not be used to decrypt the same data.

authentication

The process of demonstrating an entity is what it claims to be.

authorization certificate

An electronic document to prove one's access or privilege rights, also to prove one is who they say they are.

authorization

The process of determining what an entity is allowed to do.

blind signature

Ability to sign documents without knowledge of content, similar to a notary public.

block cipher

A symmetric cipher operating on blocks of plain text and cipher text, usually 64 or 128 bits.

Blowfish

A 64-bit block symmetric cipher consisting of key expansion and data encryption. A fast, simple, and compact algorithm in the public domain written by Bruce Schneier.

CAPI (Crypto API)

Microsoft's crypto API for Windows-based operating systems and applications.

Capstone

An NSA-developed cryptographic chip that implements a U.S. government key escrow capability.

CAST

A block cipher with an 8-byte block and 128-bit key. Developed in Canada by Carlisle Adams and Stafford Tavares.

CBC (Cipher Block Chaining)

The process of having plain text XORed with the previous cipher text block before it is encrypted, thus adding a feedback mechanism to a block cipher.

CERT (Computer Emergency Response Team)

Security clearinghouse that promotes security awareness. CERT provides 24-hour technical assistance for computer and network security incidents. CERT is located at the Software Engineering Institute at Carnegie Mellon University in Pittsburgh, PA.

certificate (digital certificate)

An electronic document consisting of a public key, information about the key's holder, and a digital signature binding the key and information together. It is a document that states the signer's belief that the information and the key belong together. Binding names, email addresses, etc. to a key creates an identity certificate.

certificate authority (CA)

A trusted third party (TTP) who creates certificates that consist of assertions on various attributes and binds them to an entity and/or to their public key.

CFM (Cipher Feedback Mode)

A block cipher that has been implemented as a self-synchronizing stream cipher.

certification

Endorsement of information by a trusted entity.

ciphertext

The output of a cipher. The cipher starts with plaintext (q.v), and uses a key (q.v) to create ciphertext.

clear text

See plaintext.

cluster

Two or more PGP Universal Servers working together in an organization where users, keys, managed domains, and policies are synchronized between Primary and one or more Secondary servers. Clustering provides security, scalability, and reliability for the servers in the cluster.

confidentiality

The act of keeping something private and secret from all but those who are authorized to see it.

cookie

Persistent Client State HTTP Cookie—a file or token of sorts, that is passed from the web server to the web client (your browser) that is used to identify you and could record personal information such as ID and pass-word, mailing address, credit card number, and other information.

corporate signing key (CSK)

A public key that is designated by the security officer of a corporation as the system-wide key that all corporate users trust to sign other keys.

credentials

Something that provides a basis for credit or confidence.

CRL (Certificate Revocation List)

An online, up-to-date list of previously issued certificates that are no longer valid.

cross-certification

Two or more organizations or Certificate Authorities that share some level of trust.

cryptanalysis

The reverse of cryptography, cryptanalysis is the art and science of break-ing ciphers, ciphertexts, or keys.

CRYPTOKI

Also known as PKCS#11, this is a standard API for using cryptographic tokens including smart cards and accelerators.

cryptography

The art and science of creating messages that have some combination of being private, signed, unmodified with non-repudiation.

cryptosystem

A system comprised of cryptographic algorithms, all possible plain text, cipher text, and keys.

data integrity

A method of ensuring information has not been altered by unauthorized or unknown means.

decryption

The process of turning cipher text back into plain text.

DES (Data Encryption Standard)

A 64-bit block cipher, symmetric algorithm also known as Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO. Widely used for over 20 years, adopted in 1976 as FIPS 46.

dictionary attack

A calculated brute force attack to reveal a password by trying obvious and logical combinations of words.

Diffie-Hellman

The first public key algorithm, invented in 1976, using discrete logarithms in a finite field.

direct trust

An establishment of peer-to-peer confidence.

discrete logarithm

The underlying mathematical problem used in/by asymmetric algorithms, like Diffie-Hellman and Elliptic Curve. It is the inverse problem of modular exponentiation, which is a one-way function.

DMS (Defense Messaging System)

Standards designed by the U.S. Department of Defense to provide a secure and reliable enterprise-wide messaging infrastructure for government and military agencies.

DSA (Digital Signature Algorithm)

The signing-only public key algorithm used in the Digital Signature Standard. DSA is a variant of the Elgamal (q.v.) algorithm.

digital signature

An electronic identification of a person or thing created by using a public key algorithm. Intended to verify to a recipient the integrity of data and identity of the sender of the data.

DSS (Digital Signature Standard)

A U.S. Federal Information Processing Standard (FIPS) for digital signatures, using DSA and SHA-1.

ECC (Elliptic Curve Cryptosystem)

Variants of the Diffie-Hellman family of public key algorithms, these operate on other sets than the integers and give smaller keys faster execution.

EDI (Electronic Data Interchange)
The direct, standardized computer-to-computer exchange of business documents (purchase orders, invoices, payments, inventory analyses, and others) between your organization and your suppliers and customers.

Elgamal
A variant of Diffie-Hellman that permits public key encryption similar to RSA encryption. The Diffie-Hellman keys in PGP are Elgamal keys.

Elgamal scheme
Used for both digital signatures and encryption based on discrete logarithms in a finite field; can be used with the DSA function.

encryption
The process of disguising a message in such a way as to hide its substance.

entropy
A mathematical measurement of the amount of uncertainty or randomness.

External Mode
One of the two mail processing modes of a PGP Universal Server (the other is Internal Mode). In External Mode, the server logically sits between your mail server and the Internet. The server encrypts outgoing SMTP email and decrypts incoming SMTP email. Email stored on your mail server is stored unencrypted.

fingerprint
A unique identifier for a key that is obtained by hashing specific portions of the key data.

FIPS (Federal Information Processing Standard)
A U.S. government standard published by NIST.

firewall
A combination of hardware and software that protects the perimeter of the public/private network against certain attacks to ensure some degree of security.

hash function
A one-way hash function—a function that produces a message digest that cannot be reversed to produced the original.

HMAC (Hash-based Message Authentication Code)
A mechanism to use a hash function such as SHA-1 to create a MAC (q.v.) based on a shared key.

hierarchical trust
A graded series of entities that distribute trust in an organized fashion, commonly used in ANSI X.509 issuing certifying authorities. PGP meta-introducers are a mechanism for using hierarchical trust with PGP certificates.

HTTP (HyperText Transfer Protocol)
A common protocol used to transfer documents between servers or from a server to a client.

IDEA (International Data Encryption Standard)
A 64-bit block symmetric cipher using 128-bit keys based on mixing operations from different algebraic groups. Considered one of the strongest algorithms.

IETF (Internet Engineering Task Force)
A large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual. See http://www.ietf.org/.

identity certificate
A signed statement that binds a key to the name of an individual and has the intended meaning of delegating authority from that named individual to the public key.

integrity
Assurance that data is not modified (by unauthorized persons) during storage or transmittal.

Internal Mode
One of the two mail processing modes of a PGP Universal Server (the other is External Mode). In Internal Mode, the server logically sits between your email users and your mail server. The server encrypts outgoing SMTP email and decrypts incoming POP and/or IMAP email. Email stored on your mail server is stored encrypted.

Internet Message Access Protocol (IMAP)
An Internet protocol for retrieving email that is stored on an email server. A newer protocol than POP.

IPsec
A TCP/IP layer encryption scheme under consideration within the IETF.

ISO (International Organization for Standardization)
Responsible for a wide range of standards, like the OSI model and international relationship with ANSI on X.509.

ITU-T (International Telecommunication Union-Telecommunication)
Formally the CCITT (Consultative Committee for International Telegraph and Telephone), a worldwide telecommunications technology standards organization.

Kerberos
A server-based authentication system developed at MIT. Microsoft has built a variant of Kerberos into its domain system.

key
A number used in a cipher to encrypt plaintext into ciphertext.

key escrow

A key recovery (q.v.) mechanism that works by simply keeping copies of keys.

key exchange

A scheme for two or more nodes to transfer a secret session key across an unsecured channel.

key length

The number of bits representing the key size; the longer the key, the stronger it is.

key management

The process and procedure for safely storing and distributing accurate cryptographic keys; the overall process of generating and distributing cryptographic key to authorized recipients in a secure manner.

key recovery

A mechanism for retrieving cryptographic keys with the ultimate intent of decrypting ciphertext with them.

key splitting

A process for dividing portions of a single key between multiple parties, none having the ability to reconstruct the whole key.

Learn Mode

A special mode of the PGP Universal Server where it handles traffic normally (including creating keys for users) but doesn't encrypt or decrypt any messages.

Lightweight Directory Access Protocol (LDAP)

A simple protocol that supports access and search operations on directories containing information such as names, phone numbers, and addresses across otherwise incompatible systems over the Internet.

MAC (Message Authentication Code)

The symmetric-key equivalent of a digital signature. MACs do not hide data, but they let someone who knows the key know whether it has been modified.

MD2 (Message Digest 2)

128-bit, one-way hash function designed by Ron Rivest, dependent on a random permutation of bytes.

MD4 (Message Digest 4)

128-bit, one-way hash function designed by Ron Rivest, using a simple set of bit manipulations on 32-bit operands.

MD5 (Message Digest 5)

Improved, more complex version of MD4, but still a 128-bit, one-way hash function.

message digest

A number that is derived from a message. Change a single character in the message and the message will have a different message digest.

MIC (Message Integrity Check)
Originally defined in PEM for authentication using MD2 or MD5. Micalg (message integrity calculation) is used in secure MIME implementations.

multipurpose Internet mail extensions (MIME)
An open set of specifications that offers a way to interchange text in languages with different character sets, and multimedia email among many different computer systems that use Internet mail standards.

NIST (National Institute for Standards and Technology)
A division of the U.S. Dept. of Commerce that publishes open, interoperability standards called FIPS.

non-repudiation
The belief that signature or MAC verifies, that the key owner cannot deny creating it.

one-time pad
A large non-repeating set of truly random key letters used for encryption, considered the only perfect encryption scheme, invented by Major J. Mauborgne and G. Vernam in 1917.

one-way hash
A function of a variable string to create a fixed length value representing the original pre-image, also called message digest, fingerprint, message integrity check (MIC).

OpenPGP
The IETF standardization of PGP technology. It consists of two RFCs, 2440 and 3156.

Open PGP/MIME
An IETF standard (RFC 3156) that provides privacy and authentication using the Multipurpose Internet Mail Extensions (MIME) security content types described in RFC1847.

Orange Book
The National Computer Security Center book entitled Department of Defense Trusted Computer Systems Evaluation Criteria that defines security requirements.

organization key
A special key used to sign all user keys that the server creates and to encrypt server backups.

passphrase
An easy-to-remember phrase used for better security than a single password; key crunching converts it into a random key.

password
A sequence of characters or a word that a subject submits to a system for purposes of authentication, validation, or verification.

PEM (Privacy Enhanced Mail)

A protocol to provide secure internet mail, (RFC 1421-1424) including services for encryption, authentication, message integrity, and key management. PEM uses ANSI X.509 certificates.

perfect forward secrecy

A property of a cryptosystem in which if one message is compromised, no later messages are compromised.

PGP/MIME

An IETF standard (RFC 2015) that provides privacy and authentication using the Multipurpose Internet Mail Extensions (MIME) security content types described in RFC1847; deployed in PGP 5.0 and later versions.

PGP technology

An application and protocol (RFC 2440) for secure e-mail and file protection developed by Phil R. Zimmermann. Originally published as freeware, the source code has always been available for public scrutiny. PGP software uses a variety of high-quality algorithms for providing encryption, authentication, message integrity, and key management. PGP software has worldwide deployment.

PGP Universal Satellite

The PGP Universal Satellite software is a small program that resides on the computer of the email user. It allows email to be encrypted all the way to and from the desktop, and it is one way for external users to participate in the SMSA. It also allows users the option of controlling their keys locally.

PGP Universal Server

A device you add to your network that provides secure messaging with little user interaction. The server automatically creates and maintains a self-managing security architecture (SMSA) by monitoring authenticated users and their email traffic. You can also send protected messages to addresses that aren't part of the SMSA.

PGP Universal Web Messenger mail

A method used by the PGP Universal system to deal with users who are not currently part of the SMSA. PGP Universal Web Messenger mail gives the recipient of the method both a way to securely read the message and several ways to become a part of the SMSA.

PKCS (Public Key Crypto Standards)

A set of de facto standards for public key cryptography developed in cooperation with an informal consortium (Apple, DEC, Lotus, Microsoft, MIT, RSA, and Sun) that includes algorithm-specific and algorithm-independent implementation standards. Specifications defining message syntax and other protocols controlled by RSA Data Security Inc.

plaintext

The input of a cipher. The cipher starts with plaintext, and uses a key to create ciphertext.

post office protocol (POP)
An Internet protocol for retrieving email that is stored on an email server. An older protocol than IMAP.

pseudo-random number generator
A mathematical process that gives apparently random numbers.

private key
The privately held "secret" component of an integrated asymmetric key pair, often referred to as the decryption key.

public key
The publicly available component of an integrated asymmetric key pair often referred to as the encryption key.

public-key infrastructure (PKI)
A widely available and accessible certificate system for obtaining an entity's public key with some degree of certainty that you have the "right" key and that it has not been revoked.

random number
An important aspect to many cryptosystems, and a necessary element in generating a unique key(s) that are unpredictable to an adversary. True random numbers are usually derived from natural sources.

RC2 (Ron's Cipher 2)
Variable key size, 64-bit block symmetric cipher, a trade secret held by RSA, SDI.

RC4 (Ron's Cipher 4)
Variable key size stream cipher, once a proprietary algorithm of RSA Security, Inc.

RC5 (Ron's Cipher 5)
A block cipher with a variety of arguments, block size, key size, and number of rounds.

revocation
Retraction of certification or authorization.

RFC (Request for Comment)
An IETF document, either FYI (For Your Information) RFC sub-series that are overviews and introductory or STD RFC sub-series that identify specify Internet standards. Each RFC has an RFC number by which it is indexed and by which it can be retrieved (www.ietf.org).

Rijndael
A block cipher designed by Joan Daemen and Vincent Rijmen, chosen as the new Advanced Encryption Standard (AES). It is considered to be both faster and smaller than its competitors. The key size and block size can be 128-bit, 192-bit, or 256-bit in size and either can be increased by increments of 32 bits.

ROT-13 (rotation cipher)
A simple substitution (Caesar) code, rotating each 26 letters 13 places.

secret key

Either the "private key" in public key (asymmetric) algorithms or the "session key" in symmetric algorithms.

secure channel

A means of conveying information from one entity to another such that an adversary does not have the ability to reorder, delete, insert, or read (SSL, IPSec, whispering in someone's ear).

secure multipurpose mail extension (S/MIME)

A proposed standard developed by Deming software and RSA Data Security for encrypting and/or authenticating MIME data. S/MIME defines a format for the MIME data, the algorithms that must be used for interoperability (RSA, RC2, SHA-1), and the additional operational concerns such as ANSI X.509 certificates and transport over the Internet.

secure shell (SSH)

A program that provides strong authentication and secure connections over insecure networks so that a user can: log into another computer over a network, execute commands on a remote machine, or move files from one machine to another.

secure socket layer (SSL)

Developed by Netscape to provide security and privacy over the Internet. Supports server and client authentication and maintains the security and integrity of the transmission channel. Operates at the transport layer and mimics the "sockets library," allowing it to be application independent. Encrypts the entire communication channel and does not support digital signatures at the message level.

self-managing security architecture (SMSA)

A traditional PKI is a certificate system that verifies and authenticates the validity of each party involved in a transaction. PGP Universal, however, automatically creates and maintains a security architecture by monitoring authenticated users and their email traffic. We call this a "self-managing" security architecture (SMSA). PGP Universal uses the SMSA it creates to secure messages between the members of the SMSA.

self-signed key

A public key that has been signed by the corresponding private key for proof of ownership.

session key

The secret (symmetric) key used to encrypt each set of data on a transaction basis. A different session key is used for each communication session.

SHA-1 (Secure Hash Algorithm)

A 160-bit hash function; part of DSS.

simple mail transfer protocol (SMTP)

An Internet protocol for sending email messages. Most Internet email systems use SMTP to send email between email servers. Email clients retrieve email using IMAP or POP.

simple object access protocol (SOAP)
A lightweight, XML-based messaging protocol for encoding the information in a Web service request and response messages before sending them over a network. SOAP messages are independent of any OS or protocol and may be sent using many Internet protocols, including HTTP, MIME, or SMTP.

single sign-on
One log-on provides access to all resources of the network.

Site Security Handbook (SSH)
The Working Group (WG) of the Internet Engineering Task Force has been working since 1994 to produce a pair of documents designed to educate the Internet community in the area of security. The first document is a complete reworking of RFC 1244, and is targeted at system and network administrators, as well as decision makers (middle management).

Skipjack
The 80-bit key encryption algorithm contained in NSA's Clipper chip.

smart trailer
Text added to a message that is sent to the recipient unencrypted. The text tells the recipient that the message could have been encrypted if the recipient were a member of the SMSA. The smart trailer includes a link to a location on the PGP Universal Server where the recipient can download the PGP Universal Satellite software, software the recipient can install on his/her machine making them a part of the SMSA.

STU-III (Secure Telephone Unit)
NSA designed telephone for secure voice and low-speed data communications for use by the U.S. Dept. of Defense and their contractors.

substitution cipher
The characters of the plain text are substituted with other characters to form the cipher text.

symmetric algorithm
Also known as conventional, secret key, and single key algorithms; the encryption and decryption key are either the same or can be calculated from one another. Two sub-categories exist: Block and Stream.

timestamping
Recording the time of creation or existence of information.

transport layer security (TLS) protocol
Provides communications privacy over the Internet. It allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The PGP Universal Server uses the TLS protocol for authentication between the server and Microsoft Exchange servers. Defined in RFC 2246; based on the Secure Sockets Layer version 3.0 protocol.

transposition cipher

The plain text remains the same but the order of the characters is transposed.

Triple DES

An encryption configuration in which the DES algorithm is used three times with three different keys.

trust

A firm belief or confidence in the honesty, integrity, justice, and/or reliability of a person, company, or other entity.

TTP (Trust Third-Party)

A responsible party in which all participants involved agree upon in advance, to provide a service or function, such as certification, by binding a public key to an entity, time-stamping, or key-escrow.

Twofish

A new 256-bit block cipher, symmetric algorithm. Twofish was one of five algorithms that the U.S. National Institute of Standards and Technology (NIST) considered for the Advanced Encryption Standard (AES).

validation

A means to provide timeliness of authorization to use or manipulate information or resources.

verification

To authenticate, confirm, or establish accuracy.

virtual private network (VPN)

Allows private networks to span from the end-user, across a public network (Internet) directly to the home gateway of choice, such as your company's intranet.

Web of Trust

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

World Wide Web Consortium (W3C)

An international industry consortium founded in 1994 to develop common protocols for the evolution of the World Wide Web.

X.509

An ITU-T digital certificate that is an internationally recognized electronic document used to prove identity and public key ownership over a communication network. It contains the issuer's name, the user's identifying information, and the issuer's digital signature, as well as other possible extensions.

X9.17

An ANSI specification that details the methodology for generating random and pseudo-random numbers.

XOR (Exclusive OR)

A mathematical operation on pairs of ones and zeros. The output of XOR is zero if both inputs are the same, either one or zero. It is one if a single input is one and the other is zero.

Zeroconf

An IETF technology (www.zeroconf.org) whose goal is to enable IP networking without any configuration. For example, you could take two laptop computers, connect them with a crossover Ethernet cable, and have them communicate usefully using IP without any further configuration.

Index