Elements of Security

Pretty Good Privacy

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin

Last updated: April 15, 2015

Some Poetry

Mary had a little key
(It's all she could export)
And all the email that she sent
Was opened at the Fort.
–Ron Rivest

What do you think that poem means?

Need for Strong Encryption

The success of our information economy depends, in large part, on the ability to protect information as it flows. This relies on the power of cryptography.

RSA and other modern algorithms provides extremely strong encryption, but are not particularly easy to use.

Pretty Good Privacy

Phil Zimmermann had the goal of providing strong encryption for the masses, in the form of an encryption system for email that is:

- extremely strong, using state of the art cryptographic algorithms;
- easy to use and accessible to all.

PGP is "the closest you're likely to get to military-grade encryption." –Bruce Schneier, *Applied Cryptography*

Zimmermann's Motivation

Zimmermann had a strong distrust of the government, and believed strongly that everyone had an absolute right to privacy.

The government generally believes that the right to privacy is limited by the need of the government to read messages under certain circumstances. Thus, the government restricts access of the public and commercial enterprises to strong encryption.

PGP is a "end-run" around government restrictions, and almost landed Zimmermann in jail.

Did Zimmermann Succeed?

From Wikipedia page on PGP:

> *In 2003, an incident involving seized Psion PDAs belonging to members of the Red Brigade indicated that neither the Italian police nor the FBI were able to decode PGP-encrypted files stored on them.*
>
> *A more recent incident in December 2006 (see United States v. Boucher) involving US customs agents and a seized laptop PC which allegedly contained child pornography indicates that US Government agencies find it "nearly impossible" to access PGP-encrypted files. Additionally, a judge ruling on the same case in November 2007 has stated that forcing the suspect to reveal his PGP pass-phrase would violate his Fifth Amendment rights i.e. a suspect's constitutional right not to incriminate himself.*

PGP

Zimmermann developed PGP (Pretty Good Privacy) in the late 1980's and early 1990's. Some characteristics include:

1. Uses the best available cryptographic algorithms as building blocks.
2. Integrates these into a general-purpose algorithm that is processor-independent and easy to use.
3. Package and documentation, including source code, are freely available on-line.
4. Now provided by Viacrypt in a compatible, low-cost commercial version.

Why would anyone buy this software from Viacrypt when it's available free?

Growth of PGP

PGP has grown explosively and is widely used.

1. Available free worldwide for Windows, UNIX, Macintosh, and others. The commercial version satisfies businesses needing vendor support.
2. Based on algorithms with extensive public review.
   Public key encryption: RSA, DSS, Diffie-Hellman.
   Symmetric encryption: CAST-128, IDEA, and 3DES.
   Hash coding: SHA-1.
3. Wide applicability: standardized scheme for encryption, supports secure communication over Internet and other networks.
4. Not developed by or controlled by any government.
5. Now on track to become an Internet standard (RFC 3156).

PGP Services

The actual operation of PGP, as opposed to key management, consists of five services:

1. Authentication
2. Confidentiality
3. Compression
4. E-mail compatibility
5. Segmentation

PGP Authentication

This is a digital signature function.

1. Sender creates a message.
2. SHA-1 (or DSS/SHA-1) is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender's private key and the result is prepended to the message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with the decrypted hash code.

What does this look like in our protocol notation? Detached signatures are also supported.

PGP Confidentiality

PGP provides encryption for messages sent or stored as files.

1. The sender generates a message and a random 128-bit session key (used for this message only).
2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
3. The session key is encrypted with RSA, using the recipient's public key, and prepended to the message.
4. The receiver uses RSA with his private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.

Put this into our protocol notation. What are the benefits of this scheme?

Confidentiality and Authentication

Both authentication and confidentiality may be combined for a given message.

1. Sender generates a signature for the plaintext message and prepends it to the message.
2. The plaintext message plus signature is encrypted.
3. The session key is encrypted using RSA and prepended to the message.

Why is it preferable to generate a signature for the plaintext message, rather than for the encrypted message?

Compression

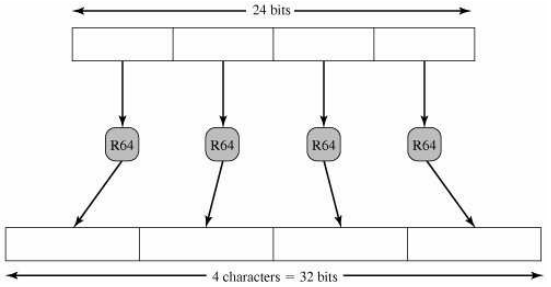
As a default, PGP compresses the message, using the compression algorithm ZIP, after applying the signature and before encryption. This is done because:

- It is preferable to sign an uncompressed message so that the signature does not depend on the compression algorithm.
- Versions of the compression algorithm behave slightly differently, though all version are interoperable.
- Encryption after compression strengthens the encryption, since compression reduces redundancy in the message.

E-mail Compatibility

PGP always involves encryption. Encrypted text contains arbitrary 8-bit octets. However, many email systems permit only ASCII text.

PGP uses radix-64 conversion to map groups of three octets into four ASCII characters. Also appends a CRC for data error checking. By default, even ASCII is converted.



E-mail Compatibility

Use of radix-64 expands the message by 33%. This is usually more than offset by the compression. One website says:

> *HELD96 reports an average compression ratio of about 2.0 using ZIP. If we ignore the relatively small signature and key components, the typical overall effect of compression and expansion of a file of length X would be* $1.33 * 0.5 * x = 0.665 * x$. *Thus, there is still an overall compression of about one-third.*

Segmentation and Reassembly

E-mail facilities are often restricted to a maximum message length. A longer message must be broken into segments, which are mailed separately.

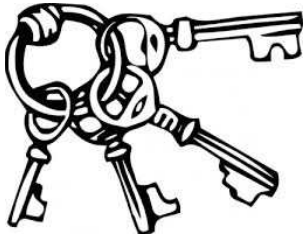
PGP automatically segments messages that are too large. This is done after all of the other steps, including radix-64 conversion. Thus signature and session key appear only once.

At the receiving end, PGP strips off mail headers and reassembles the message from its component pieces.

Key Management

PGP makes use of four types of keys: one-time session symmetric keys, public keys, private keys, passphrase-based symmetric keys.

1. Unpredictable session keys must be generated.
2. PGP allows users to have multiple public/private key pairs. There is not a one-one correspondence between users and public keys.
3. Each entity must maintain a file of its own public/private key pairs as well as the public keys of others.



Session Key Generation

A *session key* is generated on the fly for symmetric encryption of a single message.

Each session key is associated with a single message and used only once. Encryption is done with CAST-128 (128-bit keys), IDEA (128-bit keys), or 3DES (168-bit keys). (Assume CAST-128.)

CAST-128 is used to generate the key from a previous session key and two 64-bit blocks generated based on user keystrokes, including keystroke timing. The two 64-bit blocks are encrypted using CAST-128 and the previous key, and concatenated to form the new key.

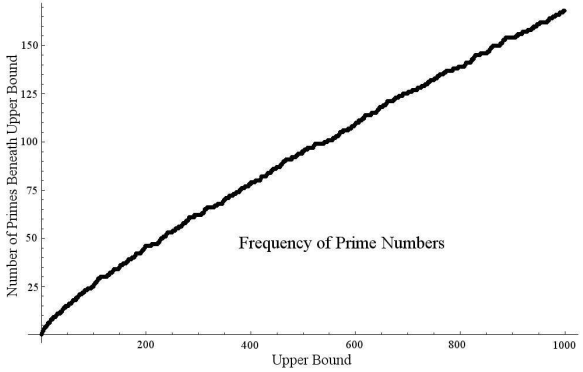
Public/Private Key Generation

Assume the public key algorithm is RSA. Generating keys requires finding a pair of large primes.

An odd number n of sufficient size (usually > 200 bits) is generated and tested for primality. If it is not prime, then repeat with another randomly generated number, until a prime is found. The generation of the initial candidate may be influenced by jiggling the mouse or other user input.

Public/Private Key Generation

A result from number theory says that primes appear in the neighborhood of n about every $ln(n) = lg_e(n)$ numbers. To find a prime of around 200 bits, it takes about $ln(2^{200})/2 = 70$ tries.



This is an expensive operation, but performed infrequently.

Managing Key Pairs

Given the desire to allow one user to have multiple public/private key pairs, how do we know which public key was used to encrypt a message.

- Send the public key along with the message. *Inefficient, since the key might be hundreds of bits.*
- Associate a unique ID with each key pair and send that with the message. *Would require that all senders know that mapping of keys to ID's for all recipients.*
- Generate an ID *likely* to be unique for a given user. *This is PGP's solution. Use the least significant 64-bits of the key as the ID.*

This is used by the receiver to verify that he has such a key on his "key ring." The associated private key is used for the decryption.

Key Rings: Private Key Ring

Each user maintains two key ring data structures: a **private-key ring** for his own public/private key pairs, and a **public-key ring** for the public keys of correspondents.

The private key ring is a table of rows containing:

Timestamp: when the key pair was generated.

Key ID: 64 least significant digits of the public key.

Public key: the public portion of the key.

Private key: the private portion, encrypted using a passphrase.

User ID: usually the user's email address. May be different for different key pairs.

Encrypting the Private Key

The private key is encrypted with a user-supplied passphrase.

1. The user selects a passphrase for encrypting private keys.
2. When a new public/private key pair is generated, the system asks for the passphrase. Using SHA-1, a 160-bit hash code is generated from the passphrase, which is discarded.
3. The private key is encrypted using CAST-128 with 128 bits of the hash code as key. The key is then discarded.

Whenever the user wants to access the private key, he must supply the passphrase.

Public Key Ring

Public keys of other users are stored on a user's public-key ring. This is a table of rows containing (among other fields):

Timestamp: when the entry was generated.

Key ID: 64 least significant digits of this entry.

Public key: the public key for the entry.

User ID: Identifier for the owner of this key. Multiple IDs may be associated with a single public key.

The public key can be indexed by either User ID or Key ID.

Signing a Message

We can now see the mechanics of the operations we described earlier:

Signing a Message

1. PGP retrieves the sender's private key from the private-key ring using the User ID as an index. If no User ID was supplied, retrieve the first private key.
2. PGP prompts the user for a passphrase to recover the encrypted private key.
3. The signature component of the message is constructed.

Encrypting a Message

1. PGP generates a session key and encrypts the message.
2. PGP retrieves the recipient's public key from the public-key ring using her User ID as an index.
3. The session key component of the message is constructed.

Decrypting a Message

1. PGP retrieves receiver's private key from the private-key ring, using the Key ID field in the session key component of the message as an index.
2. PGP prompts the user for the passphrase to recover the unencrypted private key.
3. PGP recovers the session key and decrypts the message.

Authenticating a Message

1. PGP retrieves sender's public key from the public-key ring, using the Key ID field in the signature key component of the message as an index.
2. PGP recovers the transmitted message digest.
3. PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

Public Key Management

In a public key system, the management of keys is a serious practical problem. How does B know that the public key purported to belong to A actually does, and not to some other party E. There are several approaches:

1. B can physically obtain the key from A.
2. B can verify the key over the phone, assuming B knows A's voice.
3. B can obtain A's key from a mutually trusted third party D, which certifies the key using the certification methods we discussed earlier.
4. B can obtain A's key from a trusted certifying authority, which supplies the certificate.

PGP's solution to this is flexible, but allows for *degrees of trust* in the system.

Key Trustworthiness in PGP

Associated with each public key in the user's key ring is a **key legitimacy field** that indicates the extent to which PGP trusts that this is a valid public key for this user.

This involves storing certificates for each key and chains of certificates, and the user's assessment of the trust to be assigned to the key, and various heuristics for computing trust.

Key Trustworthiness

Zimmerman envisioned establishing a "web of trust" that has been partially supplanted by current PKIs.

> *As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.* –*Phil Zimmermann, PGP 2.0 manual (1992)*

Revoking Public Keys

A user may wish to revoke a public key because:

- compromise is suspected, or
- to limit the period of use of the key.

The owner issues a signed key revocation certificate. Recipients are expected to update their public-key rings.

PGP (Cont.)

Zimmermann faced two major issues:

- Inclusion of RSA technology violated the patent protection of the algorithm.
- Exportation of strong cryptographic technology was prohibited by federal law.

Both issues were eventually resolved. PGP was granted a license for RSA, and the government realized it had lost the battle over distribution.