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. [J H Ellis: The Possibility of Secure Non-Secret Digital Encryption, CESG Report, January 1970]

- Public key cryptography is an asymmetric scheme that uses a pair of keys for encryption: a public key, which encrypts data, and a corresponding private, or secret key for decryption. 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.

- 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 (Taher Elgamal)
  –RSA (Ron Rivest, Adi Shamir, Leonard Adleman)
  –Diffie-Hellman
  –DSA, the Digital Signature Algorithm (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. Public key encryption is the technological revolution that provides strong cryptography to the adult masses.

RSA Cryptosystem

The RSA cryptosystem is named after Rivest, Shamir, and Adleman.

Public-Key Encryption Scheme:

Alice can set up a mechanism whereby she can receive and decode encoded messages from an arbitrary person.

This message can be transmitted over a public channel because the system ensures that nobody else can decode the message.

Alice advertises an encoding function E, which has the property that anyone can efficiently compute $E(M)$ for a message M, but no one but Alice may efficiently compute M from $E(M)$.

In fact, Alice has a decoding function D such that, for all M, $D(E(M)) = M$.

In the RSA scheme, Alice constructs functions E and D as follows:

- She chooses two distinct (odd) primes p, q, and computes $n = p \cdot q$.

- Alice keeps p, q secret, while n is given to the public.

- Alice also chooses an element $k \in \mathbb{Z}^*_{\phi(n)}$ with $k > 1$, and advertises k along with n. (Note that $\phi(n) = \phi(pq) = (p-1)(q-1)$, so it is easy to compute.)

- The encoding function is $E(M) = M^k \pmod{n}$, assuming that messages correspond to the elements of $\mathbb{Z}_n$ (if messages are bigger, then divide them into portions, each portion in $\mathbb{Z}_n$, and encode each portion separately).

- Alice can easily compute $l = k^{-1}$, that is, the multiplicative inverse of k in the group $\mathbb{Z}^*_{\phi(n)}$, so that her decoding function is $D(C) = C^l \pmod{n}$.

- Note that if $C = E(M)$, then $D(C) = M^{kl} = M \pmod{n}$ since $kl = 1 \pmod{\phi(n)}$.

PGP

`http://www.pgpi.org/`

- PGPi is the international variant of PGP (Pretty Good Privacy), a public key encryption program originally written by Phil Zimmermann in 1991. Later PGP versions have been developed and distributed by MIT, ViaCrypt, PGP Inc., and now Network Associates Inc. (NAI).

- PGP is the de-facto standard for email encryption today, with millions of users worldwide.

- Can PGP be cracked? Yes. Any PGP version can be cracked, provided that the attacker has enough time and resources (= money) for the job. However, a typical 1024-bit PGP message would take about 300,000,000,000 MIPS* years to crack, so the ordinary citizen is relatively safe, at least for the next few decades.

(*) MIPS Technologies, Inc. designs high-performance, low-power, 32- and 64-bit RISC (reduced instruction-set computing) processor architectures and cores for embedded systems.

How PGP 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 don't compress well aren't 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.

- This 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.

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

- The combination of the two encryption methods combines the convenience of public key encryption with the speed of conventional encryption. Conventional encryption is about 1,000 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.

```
~$ pgp -kg
Pretty Good Privacy(tm) Version 6.5.8
(c) 1999 Network Associates Inc.
Uses the RSAREF(tm) Toolkit, which is copyright RSA
Data Security, Inc.
Export of this software may be restricted by the
U.S. government.

Choose the public-key algorithm to use with your new key
1) DSS/DH (a.k.a. DSA/ElGamal) (default)
2) RSA
Choose 1 or 2: 1
Choose the type of key you want to generate
1) Generate a new signing key (default)
2) Generate an encryption key for an existing signing key
Choose 1 or 2: 1
Pick your DSS ‘‘master key’’ size:
1)  1024 bits- Maximum size (Recommended)
Choose 1 or enter desired number of bits: 1
Generating a 1024-bit DSS key.

You need a user ID for your public key.  The desired form
for this user ID is your name, followed by your E-mail
address enclosed in <angle brackets>, if you have an E-mail
address. For example:  John Q. Smith <jqsmith@nai.com>
Enter a user ID for your public key:
```

```
Enter the validity period of your signing key in days
from 0 - 10950 0 is forever (the default is 0):

You need a pass phrase to protect your DSS secret key.
Your pass phrase can be any sentence or phrase and may have
many words, spaces, punctuation, or any other printable
characters.

Enter pass phrase:
Enter same pass phrase again:

PGP will generate a signing key. Do you also require an
encryption key? (Y/n) Y
Pick your DH key size:
1)  1024 bits- High commercial grade, secure for many years
2)  2048 bits- "Military" grade, secure for future
3)  3072 bits- Archival grade, slow, highest security
Choose 1, 2, 3, or enter desired number of bits: 1

Enter the validity period of your encryption key in days
from 0 - 10950 0 is forever (the default is 0):
```

```
Note that key generation is a lengthy process.

PGP needs to generate some random data. This is done by
measuring the time intervals between your keystrokes.

Please enter some random text on your keyboard until the
indicator reaches 100%.
Press ^D to cancel
100% of required data
Enough, thank you.
.******* ......................******* .
Make this the default signing key? (Y/n) Y
.........................................
...............................................********
*******
Key generation completed.
```