

# Introduction to Cryptography

CSH5 Chapter 7  
 “Encryption”  
 Stephen Cobb &  
 Corinne Lefrançois

1

Copyright © 2011 M. E. Kabay. All rights reserved.

## Topics

- Basic Concepts & Terminology
- Types of Algorithm
- Cryptanalysis



CSH5 Chapter 7 pp 7.1-7.16

2

Copyright © 2011 M. E. Kabay. All rights reserved.

## Encryption in INFOSEC

- Foundation technology
- Underlies almost everything in INFOSEC
- Ensures or supports
  - ❑ Confidentiality
  - ❑ Control and possession
  - ❑ Integrity
  - ❑ Authenticity
  - ❑ Non-repudiation



3

Copyright © 2011 M. E. Kabay. All rights reserved.

## Basic Concepts & Terminology (1)

- **Plaintext** (aka *cleartext*): original, readable data
- **Ciphertext**: scrambled form of plaintext
- **Encryption**: reversible conversion of plaintext into ciphertext
- **Decryption**: conversion of ciphertext back into plaintext
- **Crack** (aka *break*) **code**: decrypt ciphertext without knowing key

4

Copyright © 2011 M. E. Kabay. All rights reserved.

## Basic Concepts & Terminology (2)

- **Key**: secret allowing encryption and decryption to be restricted to possessors of key
- **Symmetric encryption**: encryption requiring a shared key for both encryption and decryption
- **Asymmetric encryption**: algorithm using a different key for decryption than for encryption



5

Copyright © 2011 M. E. Kabay. All rights reserved.

## Basic Concepts & Terminology (3)

- **Keylength**: number of bits in key
- **Keyspace**: number of possible keys
- **Keyspace** =  $2^{\text{keylength}}$  Keylength in bits

$$\begin{aligned} \text{➤ } 2^n &\approx 10^{n(\log_{10} 2)} \\ &\approx 10^{0.30103n} \end{aligned}$$

6

Copyright © 2011 M. E. Kabay. All rights reserved.

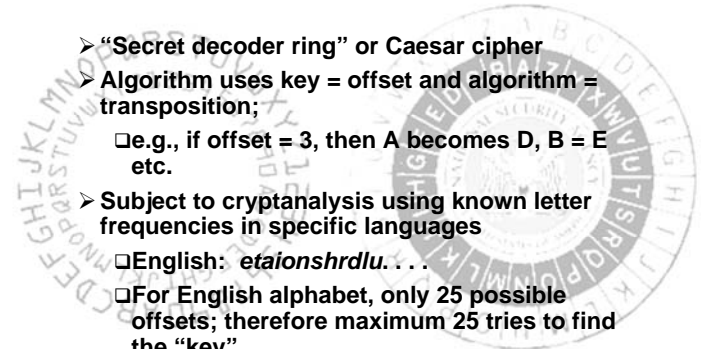
## Topics

- Basic Concepts & Terminology
- Types of Algorithm
- Cryptanalysis



## Monoalphabetic Substitution Ciphers (1)

- “Secret decoder ring” or Caesar cipher
- Algorithm uses key = offset and algorithm = transposition;
  - ❑ e.g., if offset = 3, then A becomes D, B = E etc.
- Subject to cryptanalysis using known letter frequencies in specific languages
  - ❑ English: *etaionshrdlu...*
  - ❑ For English alphabet, only 25 possible offsets; therefore maximum 25 tries to find the “key”



## Monoalphabetic Substitution Cipher: Example

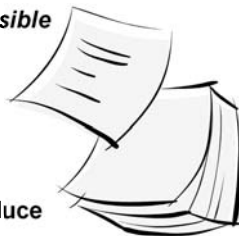
	A	B	C	D	E	F
1	Monoalphabetic Substitution Cipher Demonstration					
2						
3	Enter offset below:					
4	5	" = IF(A6 < > "", MOD(CODE(A6)-CODE("A")+\$A\$4,26)+CODE("A"), "")"				
5	Cleartext	ASCII	Transformed ASCII code	Ciphertext		
6	T	84	89	Y		
7	H	72	77	M		
8	E	69	74	J		
9						
10	Q	81	86	V		
11	U	85	90	Z		
12	I	73	78	N		
13	C	67	72	H		
14	K	75	80	P		

## Polyalphabetic Substitution Ciphers

- Can use different offsets for *each position* in plaintext
  - ❑ E.g., *Vigenère* cipher is like 26 Caesar ciphers
  - ❑ Use key indicating *which offset* to use for *which position* in sequence of 26 letters
- See <http://www.trincoll.edu/depts/cpsc/cryptography/vigenere.html>
- Or [http://en.wikipedia.org/wiki/Vigen%C3%A8re\\_Cipher](http://en.wikipedia.org/wiki/Vigen%C3%A8re_Cipher) for detailed illustrations of how to perform the *Vigenère* cipher with a particular key on a specific plaintext

## One-Time Pad

- Use a fixed and shared secret to determine offsets
- In theory, is only cipher *impossible* to break IFF
  - ❑ Pad kept secret
  - ❑ Key data truly random
  - ❑ Key data never re-used
- In practice, people use natural language (e.g., novels) and reduce strength of algorithm
- Major problem: how to distribute the pad securely?
  - ❑ *Explain the problem.*



IFF means “if and only if”

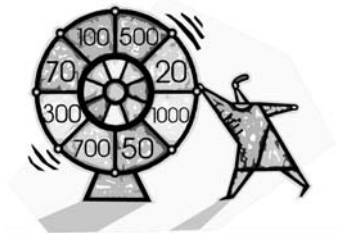
## Secure Key Distribution

- The problem of distributing a key securely is completely general to all *secret key* algorithms
  - ❑ Shared secret essential for both enciphering and deciphering data
- Therefore both sender and receiver must share the secret securely
- But if it were secure to transmit the key, you could transmit the plaintext message too
- So how do you get the secret from one to the other securely?
- Need an alternate communications channel with higher security



## Example of Linear Congruential Pseudo-Random Number Generator

$$n_{i+1} = \text{frac}(\pi^{n_i})$$



Copyright © 2011 M. E. Kabay. All rights reserved.

## Linear Congruential Pseudo-Random Number Generators

- Amateurs assume that RAND() function can be used as basis of one-time pad
  - ❑ But in fact such functions are NOT truly random
- For one thing, use floating-point data with specific number (e.g., 17) of significant figures
  - ❑ Thus must inevitably repeat (WHY?)
- Therefore ciphers with natural-language plaintext are subject to frequency analysis
  - ❑ WHY?
- Nonetheless, can be useful for simulations and demonstrations

Copyright © 2011 M. E. Kabay. All rights reserved.

## Topics

- Basic Concepts & Terminology
- Types of Algorithm
- Cryptanalysis



Copyright © 2011 M. E. Kabay. All rights reserved.

## Cryptanalysis

- Kerckhoffs' Principle
- Cryptanalytical Methods
- Types of Cryptanalytical Attacks



Copyright © 2011 M. E. Kabay. All rights reserved.

## Kerckhoffs' Principle\*

- The strength of an encryption algorithm does not reside in the secrecy of the algorithm



Corollary:

- The strength of an encryption algorithm is not measurable unless the algorithm is known

\* Published in 1883. Not to be confused with Kirchoff's Laws (physics)

Copyright © 2011 M. E. Kabay. All rights reserved.

## Dangers of Proprietary Algorithms

*Therefore beware of secret, proprietary algorithms*

- Many amateurs have failed utterly to defeat cryptanalysis
- Must demonstrate that even with *knowledge of the algorithm* and even *knowledge of a plaintext & ciphertext sample*, still too expensive to decrypt general ciphertext to make cryptanalysis worthwhile



Copyright © 2011 M. E. Kabay. All rights reserved.

## Cryptanalytical Methods

- Frequency-Based Cryptanalysis
- Brute-Force Cracking
- Attacking Weak Algorithms



## Frequency-Based Cryptanalysis

- Possible to use frequency of single letters and *digraphs* (pairs of letters) to analyze ciphertext
  - But this technique works only for plaintext based on natural language
  - Must know (or guess) which language is used\*
  - Need large amounts of data
- Does not help with cryptanalysis of purely numerical data unless there are regularities in the plaintext

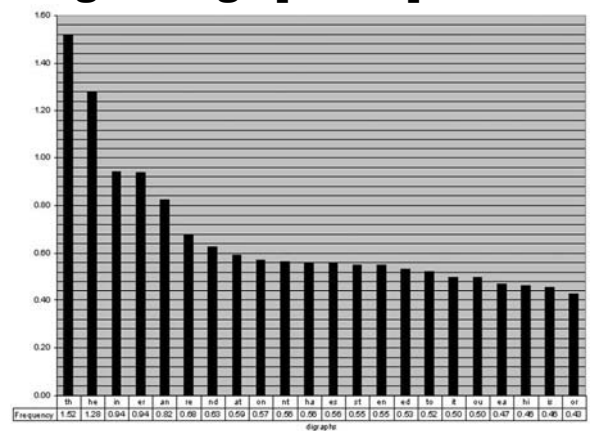
\* e.g., frequency of single letters in plain English follows sequence *ETAOINSHRDLU*

## Frequency-Based Analysis: A Bit More Detail

- Ciphertext only: Study patterns in ciphertext
  - Digraphs: pairs of symbols in sequence
  - Trigraphs: sets of three symbols in a row
- Plot frequencies of digraphs, trigraphs etc.
- Tables exist of known frequencies of transition probabilities for letters in natural language
  - E.g., in English *th* more common than *tx*
  - AKA *Markoff Chain probabilities*
- Use transition probabilities to spot likely transformed ciphertext

See chart on next slide from Cornell University's "Math Explorer's Club"  
<http://www.math.cornell.edu/~mec/>  
 (Used with permission)

## English Digraph Frequencies

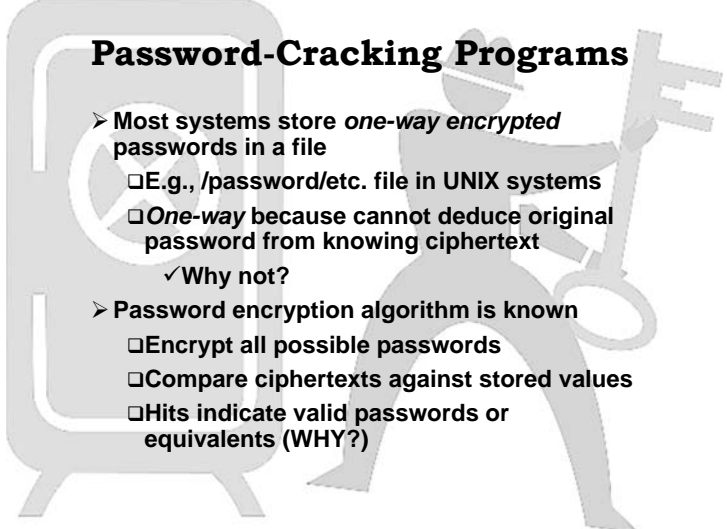


## Brute-Force Cracking

- Try every possible key
  - Facilitated by massively parallel computing
- *Dictionary attacks* narrow the range of keys
  - Helpful when one suspects that the target user has chosen *bad key*
    - ✓ Names of pets, friends, sports teams, hobbies, objects on desk
  - Password-cracking programs* use dictionaries
    - ✓ Try every word and combination
    - ✓ Can also introduce numbers and symbols

## Password-Cracking Programs

- Most systems store *one-way encrypted* passwords in a file
  - E.g., /password/etc. file in UNIX systems
  - One-way* because cannot deduce original password from knowing ciphertext
    - ✓ Why not?
- Password encryption algorithm is known
  - Encrypt all possible passwords
  - Compare ciphertexts against stored values
  - Hits indicate valid passwords or equivalents (WHY?)



## Defending Against Password-Cracking Programs

- How can you choose passwords that are hard to crack?
- Don't use real words
  - ❑ Why?
- Introduce numbers and symbols into the password sequence
  - ❑ Why?
- Change your password periodically
  - ❑ Why?
- Don't use the same password on public Web sites as on important / secure production sites
  - ❑ Why?

## Interfering with Brute-Force Cracking

- Need to know the algorithm used for encryption
  - ❑ Why?
- Must be able to *recognize* successful decryption
  - ❑ Why?
- *Superencryption* of plaintext makes brute-force cracking *more difficult* but not impossible
  - ❑ Suppose adversary uses two algorithms,  $E_1$  and  $E_2$  using keys  $k_1$  and  $k_2$  respectively
  - ❑ Thus must crack  $E_{2k_2}(E_{1k_1}(P))$  which has a *keyspace* that is the *product* of  $k_1$  and  $k_2$
- Using different data encoding schemes can confuse cryptanalyst (e.g., use EBCDIC & ASCII)

## Attacking Weak Algorithm

- Find methods of deducing key due to bad algorithms
  - ❑ But may be able to find key only one message at a time
  - ❑ May be able to demonstrate that algorithm is fundamentally flawed – may not successfully protect ciphertext against analysis (e.g., Knapsack algorithm)
- Fundamental principle: strength of encryption measured by *time* and *cost* of cryptanalysis *for specific application*

## Types of Cryptanalytical Attacks

- *Ciphertext only*:
  - ❑ No idea of plaintext at all
- *Known plaintext*:
  - ❑ Examine plaintext + ciphertext
- *Chosen plaintext*:
  - ❑ Choose plaintext and examine ciphertext
- *Adaptive chosen plaintext (aka differential cryptanalysis)*:
  - ❑ Repeatedly choose plaintext and examine results of encryption

## Stronger Encryption & the PKC

- Stronger Encryption
  - ❑ Transposition Ciphers
  - ❑ Block Ciphers & Chaining
  - ❑ Product Ciphers
    - ❑ DES: Data Encryption Standard
    - ❑ Triple DES (3DES)
    - ❑ AES: Advanced Encryption Standard
- Public Key Cryptosystem (PKC)

CSH5 Chapter 7  
pp 7.16-7.43

## Stronger Encryption

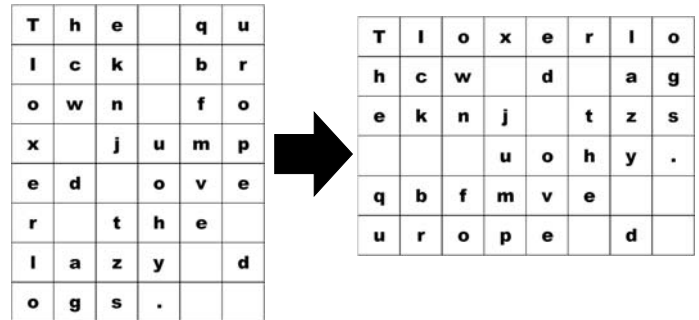
- Substitution ciphers are generally *weak* (i.e., cheap or quick to crack)
- Stronger ciphers include
  - ❑ Transposition ciphers
  - ❑ Block ciphers & chaining
  - ❑ Product ciphers

## Transposition Ciphers

- Change order of plaintext
  - ❑ Use specific algorithm (rule)
- Example: matrix rotation
  - ❑ Matrix dimensions can serve as key; e.g., 6 x 8 then read as 8 x 6
  - ❑ Read text in opposite direction of matrix
  - ❑ See next slide for illustration
- Interferes with expected frequencies of digraphs, trigraphs etc.

## Transposition Ciphers: Example

The quick brown fox jumped over the lazy dogs.  
*Tioxrlohcw d ageeknj tzs uohy.qbfmve urope d*



## Cryptanalytical Attacks on Transposition Ciphers

- Susceptible to combination of brute-force and frequency-based analysis
  - ❑ Try different offsets looking for familiar / frequent digraphs
  - ❑ This helps to determine the original matrix and its rotation
- Nonetheless, transposition is an important part of more complex encryption schemes

## Block Ciphers & Chaining

- Introduce additional complexity:
  - ❑ Break plaintext into blocks
  - ❑ Apply transposition and substitution ciphers to each block
- Chaining
  - ❑ Introduce an element from previous block into next block
  - ❑ Thus specific ciphertext becomes context dependent
  - ❑ Risk: transmission error may render ciphertext unrecoverable

## Product Ciphers

- Use all methods at once
  - ❑ Blocks
  - ❑ Chaining
  - ❑ Transpositions
- Problem: no mathematical way of proving that a product cipher is actually strong
  - ❑ Therefore try to look at degree of randomness
  - ❑ Measure frequencies of symbols
  - ❑ Also 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, ... n<sup>th</sup>-order correlations

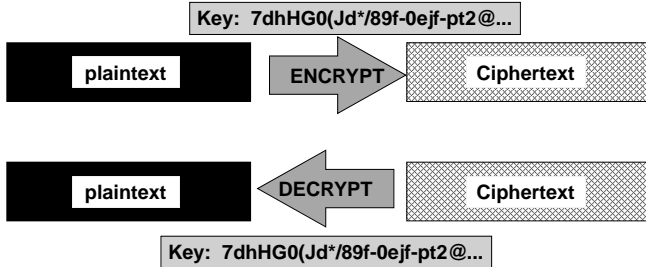


## DES: Data Encryption Standard

- 1971: IBM created "Lucifer"
  - ❑ Used in financial transactions
  - ❑ Single-key symmetric algorithm using 56-bit keys
- NIST selected Lucifer to be the DES
  - ❑ 1977: Federal Information Processing Standard (FIPS) 46
  - ❑ US government standard for unclassified use
- Widely adopted in commercial banking
- Originally defined in hardware
  - ❑ Increased general computer speed led to approval for software implementations

## Encryption: DES

- Data Encryption Standard
  - example of *symmetric* encryption algorithm



37

Copyright © 2011 M. E. Kabay. All rights reserved.

## Triple DES (3DES)

$$C = E_{k1}[D_{k2}[E_{k1}(P)]]$$

Where

- $E_{k1}(P)$  means “encrypt plaintext using key 1”
- C means “ciphertext”
- Keylength 110 bits
- Keyspace  $2^{110} \approx 10^{36}$
- Much used for key management (see next lecture)

38

Copyright © 2011 M. E. Kabay. All rights reserved.

## AES: Advanced Encryption Standard

- 1997: NIST requested new encryption algorithm
  - Protect sensitive unclassified US government information
- Competition among candidate algorithms
  - Winner: Rijndael (“Rhine doll”)
  - Drs Joan Daeman & Vincent Rijmen from Belgium
- Block cipher w/ variable block length & variable key length (easily extendible)
  - Easy to implement in hardware (e.g., smart cards) as well as software

39

Copyright © 2011 M. E. Kabay. All rights reserved.

## Public Key Cryptosystem (PKC) Topics

- History
- Functions
- Illustrations of Digital Signatures

*Michel E. Kabay*

Michel E. Kabay

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Michel E. Kabay

-----BEGIN PGP SIGNATURE-----
Version: PGP Desktop 9.8.3 (Build 4028)
Charset: utf-8

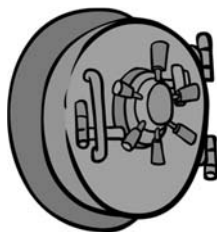
wJ8DBQPKED6xU5bF73uXq1J8RAmKAJ4znixKrcGdsyva20+Pr1M+tRw+4hgCdEPGe
MqWT6v0++1/YqkK86Z418cA=
~FIqS
-----END PGP SIGNATURE-----
```

40

Copyright © 2011 M. E. Kabay. All rights reserved.

## Functions of the PKC

- Protecting confidentiality
- Assuring integrity
- Demonstrating authenticity



41

Copyright © 2011 M. E. Kabay. All rights reserved.

## PKC: Public Key Cryptosystem (1)

- Discoverers / Inventors – 1974-1975
  - Stanford University
    - ✓ Whitfield Diffie
    - ✓ Martin Hellman
  - UC Berkeley
    - ✓ Ralph Merkle
- Radically new concept at the time
  - Create 2 complementary keys
  - Make one of them public
  - Dispense with problems of secure key distribution for decryption



Diffie, Hellman, Merkle

42

Copyright © 2011 M. E. Kabay. All rights reserved.

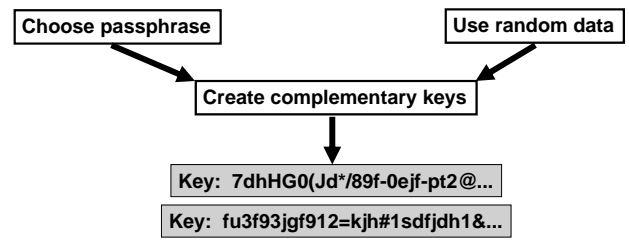
## PKC History (2)

- Idea developed into the Public Key Cryptosystem (PKC) by three scientists
  - Ron Rivest
  - Adi Shamir
  - Len Adleman
- Founded RSA Data Security Inc. (RSADSI)
  - Now one of best-known security firms
  - Sponsor highly-regarded annual conference
  - Web site has much useful information  
<http://www.rsa.com>



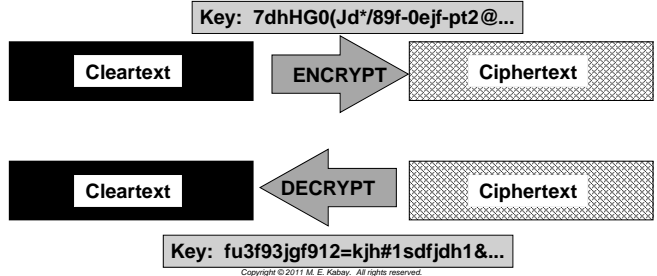
## Encryption Using PKC (1)

- Key generation produces 2 keys
  - Each can decrypt the ciphertext produced by the other
  - One is defined as *public*
  - Other is kept as *private*



## Encryption Using PKC (2)

- Key generation produces 2 keys
  - Each can decrypt the ciphertext produced by the other
  - One is defined as *public*
  - Other is kept as *private*

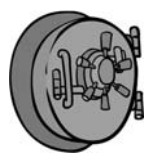


## Using the PKC for Encryption (1)

- How can we send a message so only the desired *recipient* can read it? Select the correct answers in the following description.
- First, encrypt the plaintext using the

Sender's       Public  
 or                      or  
 Recipient's       Private

Key

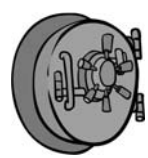


## Using the PKC for Encryption (2)

- Send the ciphertext to the recipient
- Next, decrypt the ciphertext using the

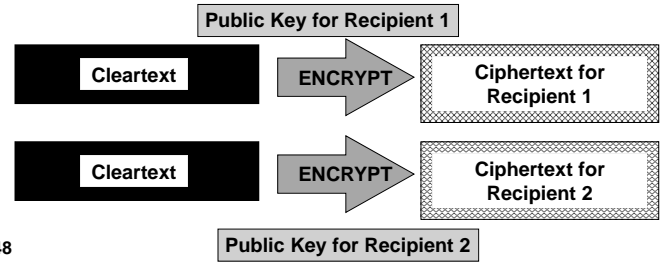
Sender's       Public  
 or                      or  
 Recipient's       Private

Key



## Sending a Ciphertext to Multiple Recipients

- What if you have to send a message securely to many people?
  - Obvious way is to encrypt the message separately for each recipient
  - Thus generate as many ciphertexts as recipients



## Multiple Recipients (2)

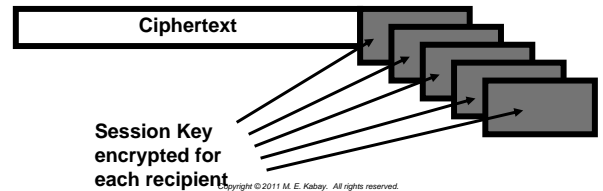
- However, e-mail normally makes it easy to send one message to multiple recipients
  - ❑ Don't want to send a different ciphertext to each recipient
- PKC algorithms are computationally demanding
  - ❑ Can take significant time to encrypt messages
  - ❑ Encrypting same message  $n$  times could take a long time

49

Copyright © 2011 M. E. Kabay. All rights reserved.

## Multiple Recipients (3)

- Use a one-time symmetric key to create ciphertext -- the *session key*
- Prepare as many copies of this symmetric key as necessary to reach all the recipients
- Encrypt a copy of the symmetric key with the public key of a specific recipient
  - ❑ Do this step for each recipient

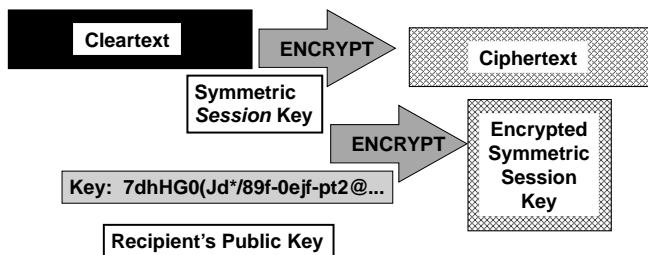


50

Copyright © 2011 M. E. Kabay. All rights reserved.

## Multiple Recipients (4)

- Send both the ciphertext and the encrypted decryption keys to all the recipients

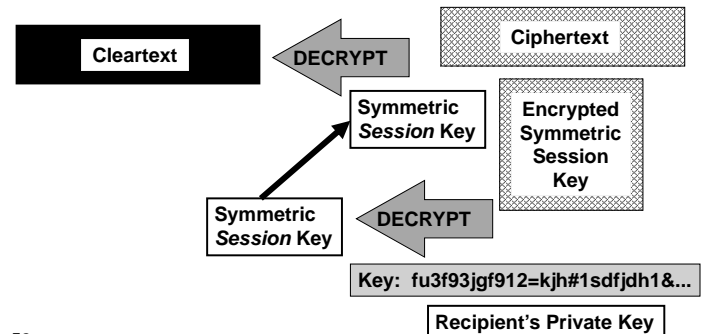


51

Copyright © 2011 M. E. Kabay. All rights reserved.

## Multiple Recipients (5)

- Each recipient decrypts the asymmetric key using their own private key

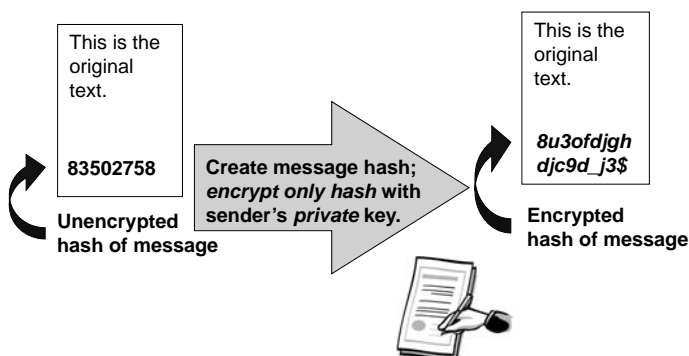


52

Copyright © 2011 M. E. Kabay. All rights reserved.

## Digital Signature Using PKC (1)

- Signing a document using PKC

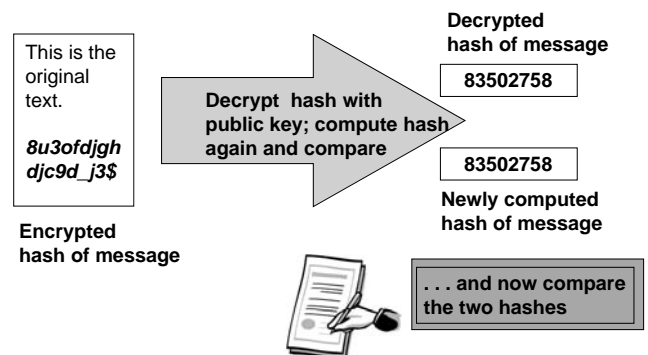


53

Copyright © 2011 M. E. Kabay. All rights reserved.

## Digital Signature Using PKC (2)

- Verifying the signature using PKC



54

Copyright © 2011 M. E. Kabay. All rights reserved.

## Digital Signature Using PKC (3)



- IF (decrypted hash = newly computed hash)
- THEN
  - ❑ The message has not been modified in transit
  - ❑ The message was signed by the owner of the private key corresponding to the public key\*



\* Or by someone who has compromised that key!

55

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP & GPG



- Screenshots illustrating use of popular PKC encryption and digital signature tools
- Creating a new key
- Signing a document
- Effects of corrupting a document
- Encrypting a document
- Decrypting a document
- Establishing the Web of Trust



56

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP



- Tool for applying PKC
- Supports
  - ❑ File encryption / decryption
  - ❑ Message encryption / decryption
  - ❑ Disk encryption / decryption
  - ❑ Authentication through digital signatures
- Provides facilities for establishing *web of trust* (see lecture on PKI) among users
  - ❑ Key distribution
  - ❑ Key signing
  - ❑ Key revocation & expiration



57

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP: Pretty Good Privacy



- Phil Zimmermann
  - ❑ Computer programmer
  - ❑ Civil libertarian
- Released Pretty Good Privacy\*
  - ❑ June 1991 – worldwide distribution
  - ❑ Became most widely-used encryption program in world
- PZ worked with Viacrypt to create PGP 3 (renamed PGP 5) in 1997
- Developed OpenPGP (RFC 4880)
- Free Software Foundation developed GNU Privacy Guard (GPG) in compliance with OpenPGP



\* Reference to Garrison Keillor's *Prairie Home Companion* radio show, where a mythical sponsor was "Ralph's Pretty Good Grocery."

58

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP (cont'd)



- Zimmermann investigated by grand jury for *supposedly* violating ITAR (Intl Traffic in Arms Regulations)
  - ❑ Much protest (cypherpunks & also Prof Kabay writing vehemently in *Network World!*)
  - ❑ PZ responded by publishing entire code in a book – not subject to ITAR
  - ❑ Prosecution abandoned after several years
- Bought by Network Associates (NAI) in 1997
- New PGP company created 2002 & bought code from NAI
- Now PGP Corporation; see FAQ for more info <http://www.pgp.com/company/faqs.htm>

59

Copyright © 2011 M. E. Kabay. All rights reserved.

## Getting PGP



- For professional use see range of products at <http://www.pgp.com/products/index.html>
- PGP 9.0 available for 30 free trial <http://www.pgp.com/products/index.html>
- Freeware for various platforms <http://www.pgp.com/downloads/freeware/index.html>
  - ❑ Amiga, Atari, BeOS, EPOC (Psion), MacOS, MS-DOS, Newton, OS/2, PalmOS, Unix, Windows 2000, Windows 3.x, Windows 95/98/NT, Windows ME, Windows XP
- International users
  - ❑ <http://www.pgpi.org/>

60

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP Personal Privacy v8.1 \*



Despite older images used here, there are no significant changes in the current versions (2009).

\* In Aug 2009 the current version was PGP Desktop v 9.8.3

61

Copyright © 2011 M. E. Kabay. All rights reserved.

## Encryption: PGP Demo

Watch as your instructor demonstrates the actions of PGP (commercial version 8.1)\* and take notes on what you see and learn.

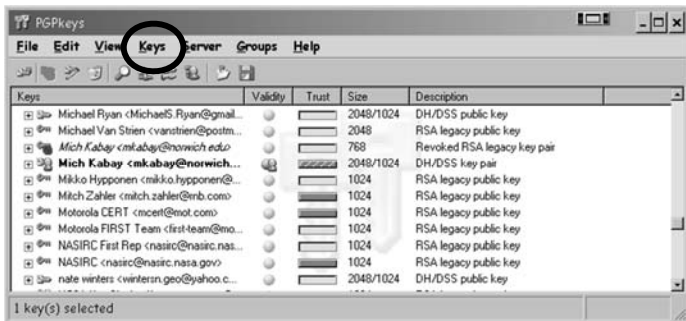
- Creating a private key / public key pair
- Signing a document with a private key
- Validating a signature with a public key
- Effect of a single-byte change on validity of a digital signature
- Encrypting a document using a public key
- Decrypting a document using a private key
- Effect of a single-byte change on decryption
- Signing someone's public key
- GPG

\*Using screen captures

62

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP: Creating a Private Key / Public Key Pair



63

Copyright © 2011 M. E. Kabay. All rights reserved.

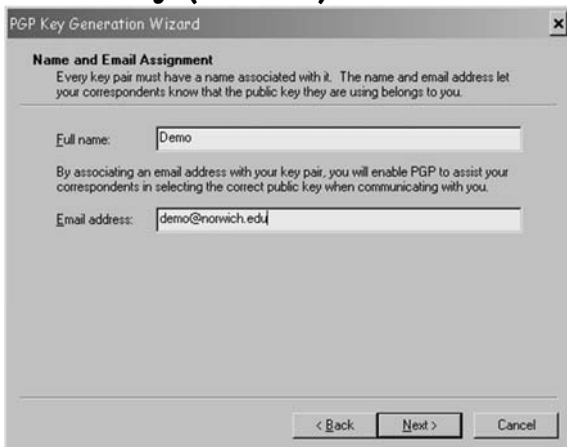
## New Key (cont'd)



64

Copyright © 2011 M. E. Kabay. All rights reserved.

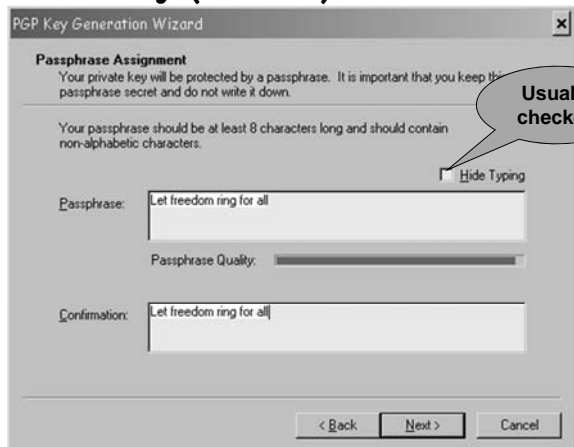
## New Key (cont'd)



65

Copyright © 2011 M. E. Kabay. All rights reserved.

## New Key (cont'd)

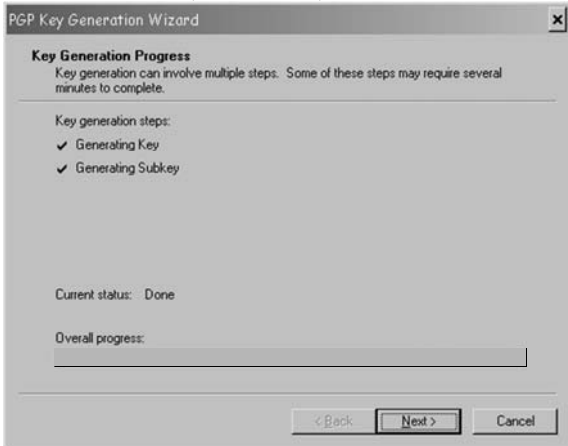


Usually checked

66

Copyright © 2011 M. E. Kabay. All rights reserved.

## New Key (cont'd)



67

Copyright © 2011 M. E. Kabay. All rights reserved.

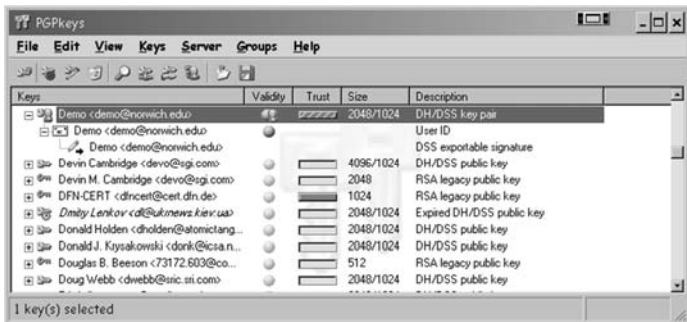
## New Key (cont'd)



68

Copyright © 2011 M. E. Kabay. All rights reserved.

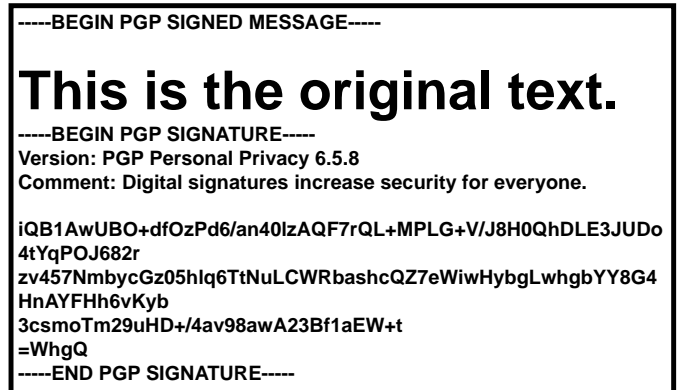
## New Key (cont'd)



69

Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP: Signing a Document With a Private Key



70

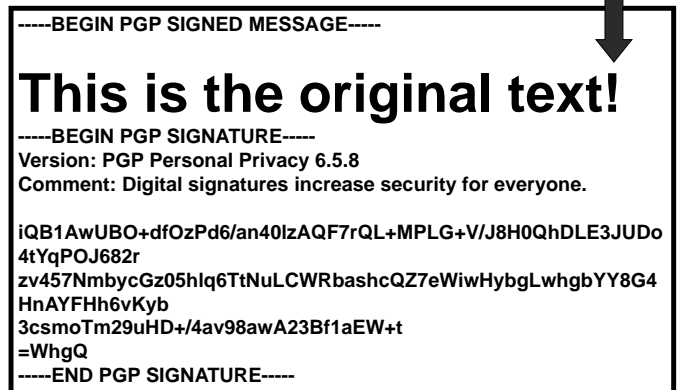
Copyright © 2011 M. E. Kabay. All rights reserved.

## PGP: Validating a Signature With a Public Key



71

## PGP: Single-byte Change Alters Digital Signature



72

Copyright © 2011 M. E. Kabay. All rights reserved.

# Single-byte Change Alters Digital Signature (cont'd)



Everything from here on is different

iQB1AwUBO+dfOzPd6/an40lzAQ F7rQL+MPLG+V/J8H0QhDLE3JU Do4tYqPOJ682r zv457NmbycGz05hlq6TtNuLCW RbashcQZ7eWiwHybgLwhgbYY8 G4HnAYFHh6vKyb 3csmoTm29uHD+/4av98awA23Bf 1aEW+t =WhgQ	iQB1AwUBO+fETTPd6/an40lzAQ FagQL/Thfw3DAJA/KRgoH+kSfC oRL39eJp4s5h v3zeHUESokgQk2zSUF+evbRhw 5cxZJKUA1Qid6cg58tEaP9jl+7J3 wLmJrFPF/K L42qO9yjxalNssnflUaSF7ry7xXV3 bIK =svYa
---	--

This is the original text.

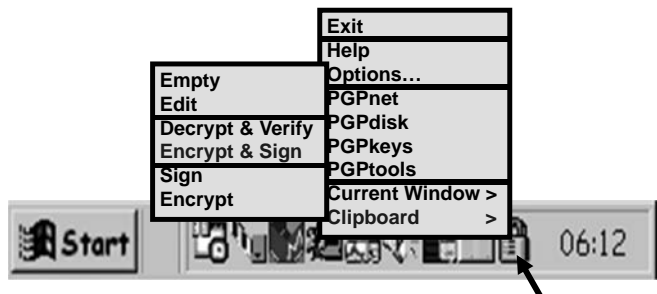
This is the original text!

Remember, exclamation mark is different from original

# Single-byte Change (cont'd)



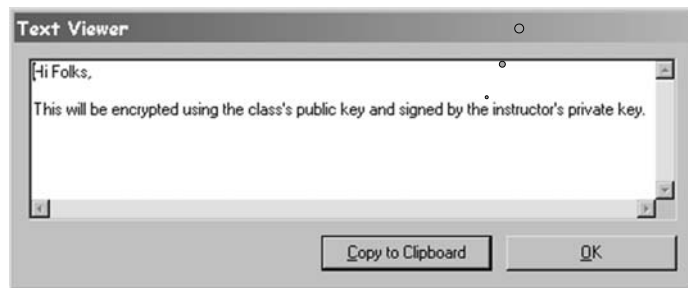
# PGP: Encrypting a Document Using a Public Key



# Encryption (cont'd)

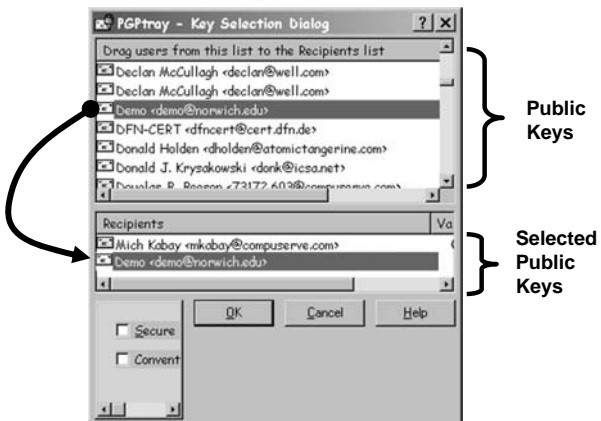


Cleartext



Note that the sender should ALWAYS encrypt using the sender's public key as well as the recipient's public key to allow decryption by the sender.

# Encryption (cont'd)



# Encryption (cont'd)



Ciphertext

```
-----BEGIN PGP MESSAGE-----
Version: PGP Personal Privacy 6.5.8
Comment: Digital signatures increase security for everyone.

hQIOA8BxvYx90+tvEAf/eXW8i9vY45PwN27g4kz0DyoCJiucXxA4eDtzHqBUJm
nWadsATpcspqQNaZwgp8d5FwxRhafOTg2WqSXl2SgsFUIjraYgXdiRBM2qeU/dKh
8+nfnZv2ol5bPaDAg9oiYDRV7KcFAMm3RfIISTRv6f72mE+Y6VPAvQmL6zJL AeOD
i/9n7Sh0UyM+71 YuNeY0V+EZ88rTba7JP4xr/GtAFKwwSHIWEhOfiwe82L MxOnSg
t8vj6amULTYp4daULqDt+qewULR4XWLhQ73zVT9578BSaUmgSniRXvelsHj4fj9n
2+Ry7QZZL7k2XDybljAw8QfLcShDgC3n5Wlt+mWLwWgA8zJSyzTdkYtVh0kuo3CP
9kkGpkgqyqitVFhua+J4bvQrrhLCLBzCWXBi6RSdFvChJ7JNhtgzajV16L5SepDS
tHMcahPvGoE1e+uG5P80xqngbc5Si7BY0jfsUjGBlfVrpt+oC37I4W2ZKZ6mgRU
4yY+H2NTi28gC0SfXjzwrW54qARSicJluUkYyUURqSDfK4slaV8FVs/xg+Ra7U3
HuyD8VmBi0/uO3RssyPldbh2FqA1+raqeL2yuoUXLe8Dl1CToLQ595/4s0wNlZT
Ys2W5ZbOT+P2gjoVfNaRQIWFzntkRXBcs/Kx9R8pu+NMTdpJjsip0oqiGfH2jqk
2lUAAbAMZ3ev2p+NJwECf/8eog7FornXxjzB5/hQ3le0Ww+Vxk2LZVhdT07eqiz
8eN9e2XSt4cr216MFCPOf1Wj8j3suYvX7mnJa7hu0mvJvxalHawEq3U+r4ZczcC
e+q9Yaukl9IXyRHYCrWZwGt6dj1Bnyni6hzglfApq+270u8QwaGud1d/OHGhID7
+9JcSc1AKxoVZsA3ltaOMPP7frlOOZfdyFzr874mhG+Lru9sBFUy1S7h3nQfUwX
ZEe9uGnDfNUth33VrrtMqjlvUjh9gZn8BOxdkKOK0WgKvVJdy6D8bSphSaQR+vvP
V83K4BaD24kiAJ70NLbeQXPx2H5j0HYT+4bDORTt4RQgberLHagwzFkpfVldXC13
P06+MTFlliqcs+p4OJo/Mj67H6x877KWU3G7SKG4pBpgmy6KwKeUW8j9EpcKtGh+
+8tgDZNAzcm8vncQ9HEAAsN6KM9v0QoCiyDDA==
=llr+
-----END PGP MESSAGE-----
```

# PGP: Decrypting a Document Using a Private Key



79

# Decryption (cont'd)



80

Copyright © 2011 M. E. Kabay. All rights reserved.

# PGP: Effect of a Single-byte Change on Decryption



-----BEGIN PGP MESSAGE-----  
Version: PGP Personal Privacy 6.5.8  
Comment: Digital signatures increase security for everyone.

*Changed a C to X in this position*

hQIOA8BxvY90+tvEAf/eXW819vY45PwN27gd4kzg0DyoCJiucXXA4eDtZHQBUJm  
nWadsATpcspqQNaZwgp8d5FwxRhafOtG2WqSX125gsFUljraYgXdlRBm2qeU/dKH  
8+NfNZv2ol5bPaDag9oiYDRV7KcFAMm3RfiiSTXv6f72mE+Y6VPAvQmL6zjLAe0D  
i9n7Sh0UyM+71YuNeY0V+EZ88rTba7JP4xGfAVKwwSHIWEhOfiwe82LMxOnSg  
t8vj6amULTYp4daULQDt+qewULR4XWlWQ73zVT9578BSaUmgSnlRXvelshJ4fj9n  
2+Ry7QZZL7k2XDybljAw8QFLc8mDy3n5Wlt+mWLwWgA8zJSyZtdKYvVh0kuo3CP  
9kkGpkgyqitVFhua+J4bvQrrt(LXLbzCWXB16RSdFvChJ7JNhtzajV16L5SepDS  
tHMcAhPvGoE1e+uG5P80xqngbeS17BY0jrsUjGBlyfVRpt+oC3714W2ZKZ6mgRU  
4yY+H2NTI28gC0SfXjzRwS4qARsIcJluUKYVUuRqSDfK4siaV8FVs/xg+Ra7U3  
HuyD8VmBi0/uO3RSsyPldbh2FqA1+raqeL2yuoUXLe8D1CToLQ595/4s0wINlzT  
Ys2W5ZbOT+P2gjoVfNaRQIWFzntkRXBcs/Kx9R8pu+NMTdpjJspidooqIGfH2jtk  
2IUAbAmz3ev2p+NjcwECf8eoq7FornXxjzB5/HQ3le0Ww+Vxk2LZVHD707eqiz  
8eN9e2XSxt4cr216MFCPOf1Wj83suYvX7mnJa7hu0mvJvxalHawEq3U+r4Zczc  
e+q9YAUkL9iXyRHYCrWZwQ76dj1Bynm6hgzLfApq+270u8QwaGud1d/aOHGIhD7  
+9JcSC1AkXoVZsA3ItaOMPp7frlOOZfdyFzr874mhG+Lru9sBFUy1S7h3gNQfUwx  
ZEe9uGnDFnuh33VrrtMqjilVjh9gZn8BOXdKkOKoWGWkVvJdy6D8bSphSaQR+vvP  
V83K4BaD24kiAJ70NLbeQXPx2H5jHYT+4bD0RT14RQgbeRLhggwZfKpVdldXC13  
P06+MTFlliqcs+P40Jo/Mj67H6x877KWU3G7SKG4pBpgmy6KwKeUW8j9EpcKTgh+  
+8tgDZNAZcm8vncQ9HEAA5N6KM9V0qoCiyDDA==  
=llr+

81

Copyright © 2011 M. E. Kabay. All rights reserved.

# Single-byte Change & Decryption



82

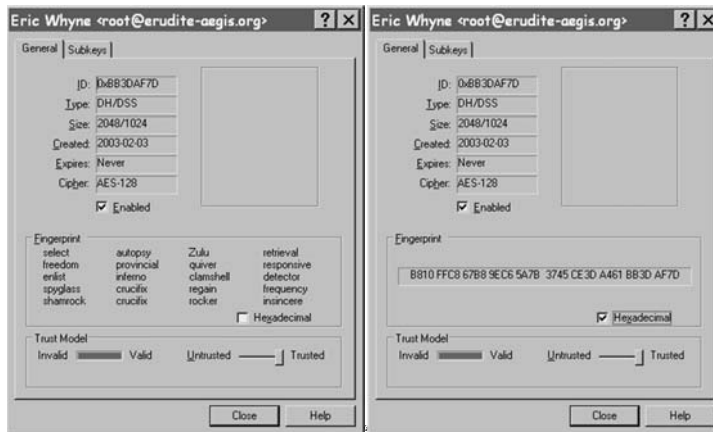
Copyright © 2011 M. E. Kabay. All rights reserved.

# Signing Someone's Public Key



83

# Checking the Public Key Fingerprint



## GPG: GNU Privacy Guard

From <http://www.gnu.org/software/gnupg/gnupg.html> :

- Implements OpenPGP Internet standard (RFC2440)
- Full interoperability with other modern encryption programs
- Meets all requirements of a standard Unix utility
- Uses DSA, ElGamal, 3DES and Twofish as encryption algorithms and more
- Easy implementation of new algorithms using extension modules
- Portuguese, French, German, Italian, Polish, Russian and Spanish language support
- Online help system
- Optional anonymous message receivers
- Integrated support for HKP key servers
- Runs on most Unix platforms and support for other platforms is coming soon

# DISCUSSION