



Internet Privacy, Data Encryption, and PGPTM

(Pretty Good PrivacyTM)



Dan Nikkel

djnikkel@home.com

Some Privacy Concerns



- **E-mail**
 - private communication
 - sensitive data (personal info, credit card #, proprietary info)

- **WWW**
 - personal/sensitive information entered onto forms

- **Electronic Commerce**

- **Files on your system**
 - preventing unauthorized or incidental access to information



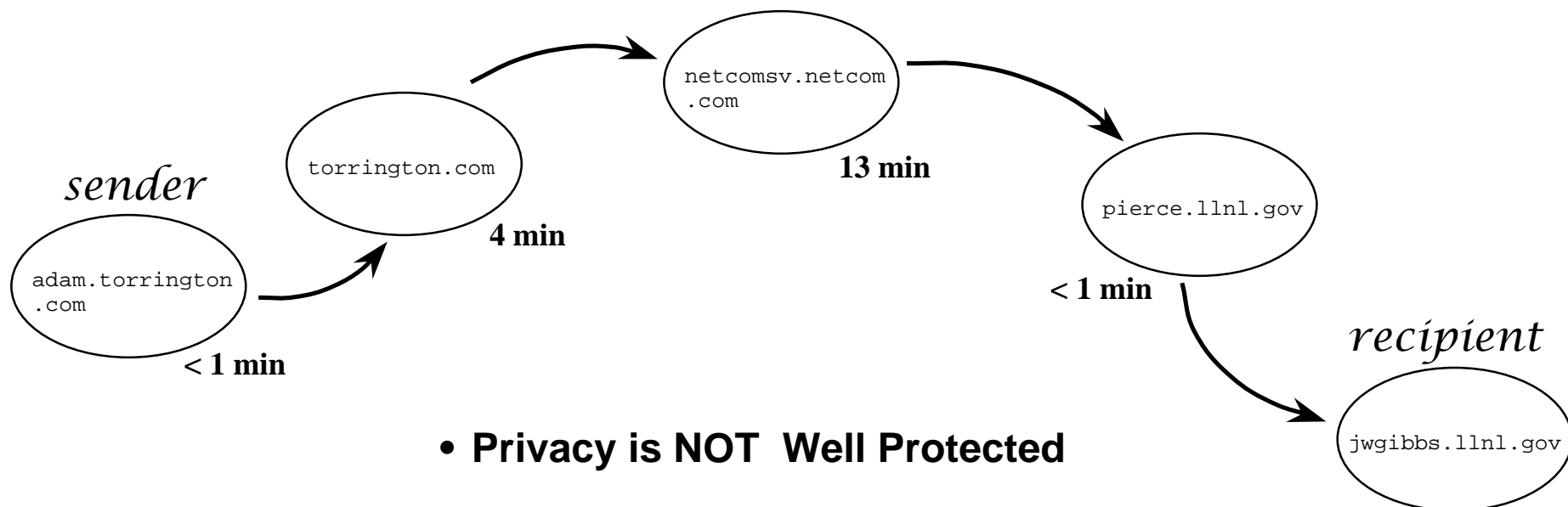
E-Mail — It's Like a Postcard

E-mail Goes Through Many Computers from Sender to Recipient

From barney!adam.torrington.com!MCKENZIE@netcom.com

Received: from pierce.llnl.gov by jwgibbs.llnl.gov	09:25:44 PDT
Received: from netcomsv.netcom.com by pierce.llnl.gov	09:24:42
Received: from barney.UUCP by netcomsv.netcom.com with UUCP	09:11:41
Received: from adam.torrington.com by torrington.com	11:52:42 EDT

From: MCKENZIE@adam.torrington.com <barney!adam.torrington.com!MCKENZIE@netcom.com>
To: djnikkel%llnl.gov@barney.torrington.com



- Privacy is NOT Well Protected
- Normal e-mail is plain-text

Cryptography Can Do Several Things



- **Keeping information hidden (confidentiality of shared information)**
- **Authenticating the source (digital signatures)**
- **Verifying information integrity**
- **Nonrepudiation of messages**

Secure Cryptography is a System



- **There are various protocols depending on what you're trying to do.**
- **Security requires more than just a good encryption algorithm:**
 - **key generation**
 - **key management**
 - **removal of unencrypted information**
 - **multi-user system issues**
- **Improper use of a cryptography system is as bad as (and probably worse than) using no cryptography at all.**



An Example of Encryption with a Key

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
	↓	↓																								↓	
a:	(rot-1)	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
b:	(rot-2)	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
c:	(rot-3)	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	⋮																										

Simple Substitution Cipher

Using rot-3: **ELEPHANT** → **HOHSKDQW**

both E's encode to the same letter

Poly-Alphabetic Substitution Cipher (requires a key)

For each character, use a different substitution cipher corresponding to successive characters from a key.

Key = ade {⇒ rot-1, rot-4, rot-5, rot-1, rot-4, ...}

∴ **ELEPHANT** → **FPJQLFOT**

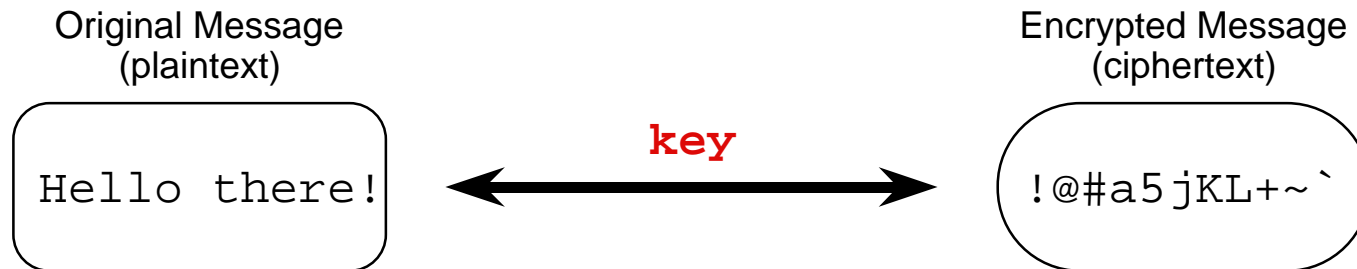
the two E's encode to the different letters

{rot-1(E)=F, rot-4(L)=P, rot-5(E)=J, rot-1(P)=Q, etc.}



Two Types of Key-Based Encryption Algorithms

Single-Key Cryptography (secret-key, conventional, symmetric)

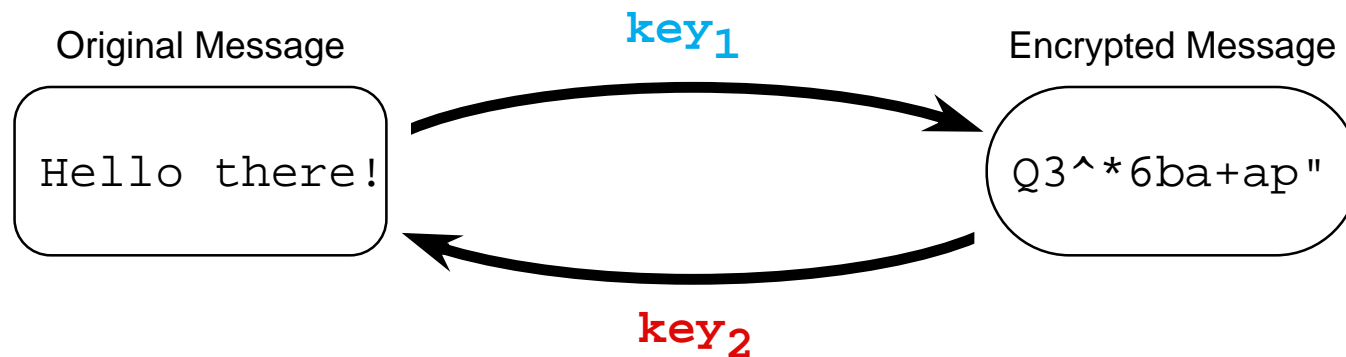


The encryption and decryption keys may actually be different, but are readily derived from each other.

Problems:

- securely getting the key to others.
- multiple recipients \Rightarrow lots of keys.

Public-Key Cryptography (asymmetric)



Some Cryptographic Algorithms



Single-Key Algorithms

- **Data Encryption Standard (DES)**
{ANSI: DEA, ISO: DEA-1} — **56-bit**
- **Tripple-DES** — **2 × 56-bit**
- **Skipjack** — **80-bit**
- **IDEA** — **128-bit**

DES key usually displayed as 64-bit
but one bit/Byte is used for parity check

Public-Key Algorithms

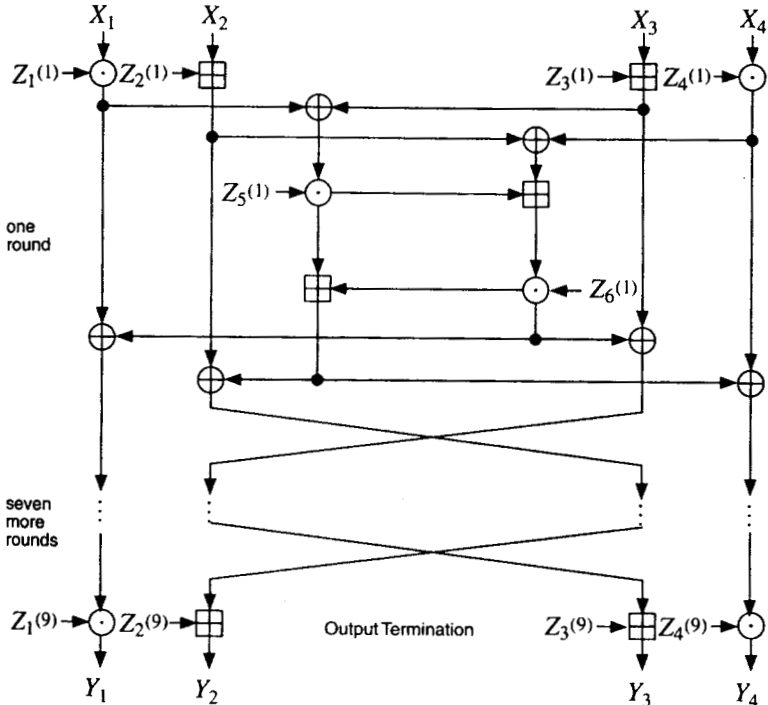
- **Diffie-Hellman** (key exchange)
- **DSA** (digital signatures)
- **RSA**

Typically 100 times slower
than single-key algorithms

HW: RSA 1000 x slower than DES
SW: RSA 100 x slower than DES



How the IDEA (single-key) Block Cipher Works



X_i : 16-bit plaintext sub-block
 Y_i : 16-bit ciphertext sub-block
 $Z_i^{(r)}$: 16-bit key sub-block
 \oplus : bit-by-bit exclusive-or (XOR) of 16-bit sub-blocks
 \boxtimes : addition modulo 2^{16} of 16-bit integers
 \odot : multiplication modulo $2^{16} + 1$ of 16-bit integers
with the zero sub-block corresponding to 2^{16}



How RSA Public-Key Encryption Works

Key Generation

- generate two very large random prime numbers p , q
- calculate the *modulus* $n = p \times q$
- generate random *encryption key* e such that e and $(p-1) \times (q-1)$ are relatively prime.
- calculate *decryption key* d from $d = e^{-1} \bmod ((p-1) \times (q-1))$

relatively prime: no common factors

Public Key: e , n Private Key: d

Cracking:

- factor n to get p and q
- weakness in random number gen.

Message Encryption / Decryption

Encryption: **Ciphertext** = (**Message**) $^e \bmod n$

Decryption: **Message** = (**Ciphertext**) $^d \bmod n$



PGP Encrypted Message

Single Recipient

$$\text{Ciphertext} = \text{RSA}_{\text{Pub-Key (1024*)}} \{ \text{128-bit session-key} \} \\ + \\ \text{IDEA}_{\text{Session-Key (128)}} \{ \text{Original Message} \}$$

* public key length can vary from user to user: 512, 768, 1024

Multiple Recipients

$$\text{Ciphertext} = \text{RSA}_{\text{PK1}} \{ \text{session-key} \} \\ + \\ \text{RSA}_{\text{PK2}} \{ \text{session-key} \} \\ \vdots \\ + \\ \text{IDEA}_{\text{Session-Key (128)}} \{ \text{Original Message} \}$$

Digital Signatures with PGP



A “digital signature” can be used to authenticate messages.

- verify the sender of the message
- verify that the message hasn’t been modified

Signature = $\text{RSA}_{\text{Secret-Key}} \{ \text{MD5} \{ \text{Original Message} \} \}$

MD5 is a 128-bit one-way hash function (message digest)

Key Authentication



- **Forged keys are the bane of public-key cryptography.**
- **PGP keys have a 128-bit “fingerprint” to aid in certification.**
- **Keys can be signed to certify validity by the signer.**
- **Key certificates have two parameters: validity, trust**
- **PGP uses the concept of a “web of trust” to certify keys.**



How Secure is PGP?

RSA (public-key algorithm)

- Cracking RSA requires factoring very large numbers.
- RSA-129 (\approx 426-bit key) cracked with 5000 MIPS-years.

Estimating: 512-bit key \Rightarrow 30,000 MIPS-years

1024-bit key $\Rightarrow 3 \times 10^{11}$ MIPS-years

- Mathematical methods of factoring steadily improve.
- Poor random number generation reduces security.

Netscape cracked because of weak random numbers.

MIT student brute force cracked SSL in 8 days using 112 wkstns (40-bit keys).

IDEA (single-key algorithm)

- brute-force attack:

128-bit key $\Rightarrow 2^{128}$ keys = 3×10^{38} keys

\$ 100 K (1995) $\Rightarrow 10^{19}$ years

64-bit:

\$100K \rightarrow 1 year

\$1M \rightarrow 37 days

56-bit:

\$100K \rightarrow 35 years

\$1M \rightarrow 3.5 hours

Platforms Supported



- **UNIX**
- **VMS**
- **Macintosh**
 - **scripts to link with Eudora**
- **DOS**
 - **front ends for Windows, WinCIM / CSNav**
- **OS/2**
 - **front ends for Presentation Manager**
- **Atari**
- **Amiga**

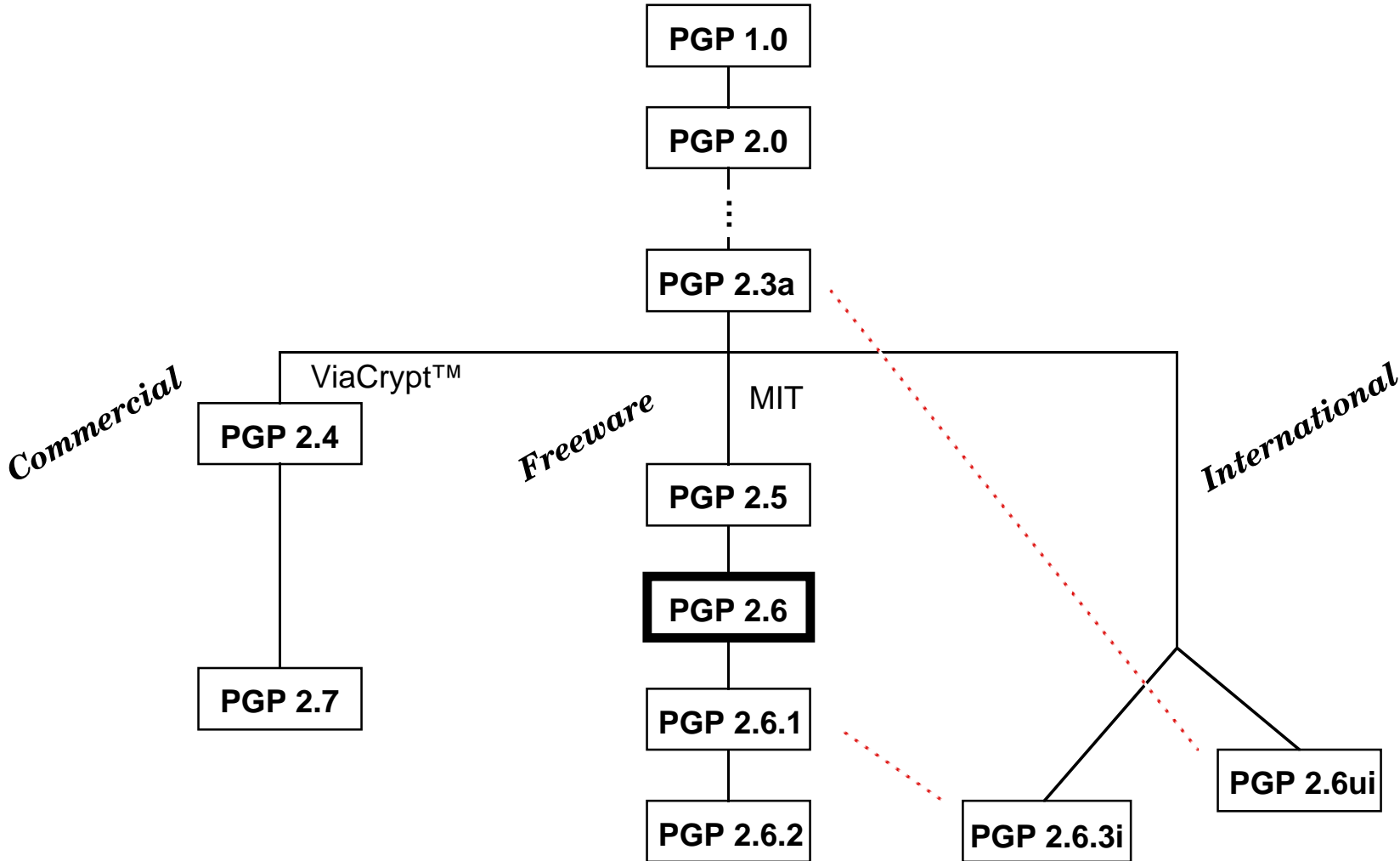


Legal Issues

- **“Strong” cryptography (key > 40-bits) is subject to export control. It’s legally the equivalent of a munition.**
 - **PGP cannot be exported.**
 - **Versions of PGP exist outside the U.S.**
- **Various cryptographic algorithms are covered by patents (e.g. RSA) and software which utilizes them may require appropriate licensing agreements.**
 - **PGP version 2.6 or later is properly licensed from RSADSI, Inc.; earlier version’s may infringe on RSADSI patents.**
- **PGP (freeware from MIT) is for noncommercial use only. For commercial use, use ViaCrypt™ PGP (\$100).**



Versions of PGP



How to get PGP



1. anonymous **ftp** to `net-dist.mit.edu`
 - a. `cd` to `pub/PGP`
 - b. get the three files `README`, `rsalicen.txt`, `mitlicen.txt`
(`README` explains the rest of the process, the other two files are the software licenses with which you must abide)
2. **telnet** (not `ftp`) back to `net-dist.mit.edu` using username `getpgp`
 - a. answer the required questions
 - b. record the directory it gives you
3. anonymous **ftp** back to `net-dist.mit.edu`
 - a. `cd` to the directory your were given in the telnet session
`pub/PGP/dist/U.S.-only-xxxx`
 - b. get the desired PGP files

[The commercial version is available from ViaCrypt: (800) 536-2664]



References

Usenet:

`alt.security.pgp` , `sci.crypt`

Web:

`ftp://ftp.prairienet.org/pub/providers/pgp/pgpfaq.txt`

`ftp://ftp.csn.net/mpj/getpgp.asc`

`http://www.pgpi.org`

`http://http.cs.berkeley.edu/~daw/crypto.html`

`http://www.enter.net/~chronos/cryptolog.html`

`ftp://rtfm.mit.edu/pub/usenet-by-group/sci.crypt`

`http://www.rsa.com`

Books:

***PGP: Pretty Good Privacy*, Simson Garfinkel, O'Reilly, 1995**

***Applied Cryptography*, Bruce Schneier, Wiley, 1996**



First Things to Do With PGP

1. Generate your own key pair.

```
pgp -kg
```

⇒ { pubring.pgp
secring.pgp

2. Extract an ASCII version of your public-key.

```
pgp -kxa your_userid output_file
```

you might also want
to get the fingerprint
`pgp -kvc`

3. Make your public-key available to anyone you wish
to be able to send you encrypted messages.

you should probably
also sign your own
key before distributing it

4. Add the public-keys of others to which you wish to
send encrypted messages to your public key ring.

```
pgp -ka keyfile
```

you'll be prompted if
you want to certify it

A 1024-bit PGP Public-Key (ASCII representation)



```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
Version: 2.6.2
```

```
mQCNAi4JHdwAAAEEMq5buCKZVU6TpKFbWDarl8OxoXaup8q2KOPU7HWKW0uMUS1  
Y0sfj2NwcS9kFHg1PG75AiRcwVm1+E4M2sR3dQTe0HrGpuwXWV0UN8lwUliX1GwK  
fmTtkVFFWrRmpbE0tz8jcdOqZTS+ZbX1fnqpAPqJ0Q1n7XzTlkiUS+4+8qqtAAUR  
tChEYW5pZWwgSi4gTmlra2VsLCBKciA8ZGpuaWtrZWxAbGxubC5nb3Y+  
=LfnJ
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

Key “fingerprint” to aid in certification

```
Key fingerprint = BE 6F FF 34 B1 58 B2 A8 19 4D 77 68 CA FD D4 41
```

A Signed Cleartext Message



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

Here is a sample message.
Here are the items I authorize you to buy:

- 1) computer parts \$2000
- 2) network services from Joe Blow
- 3) security services from Jane Smith, 449-1234

-----BEGIN PGP SIGNATURE-----

Version: 2.6.2

iQCVAwUBMPQsuEiUS+4+8qqtAQH+wQQAnCjWgr69Bmjy3eKxbeBrIIzOHjgv/t00
hFkMPa3P/HQhjpOWe4chpFcCsPTZHlcTYDj7kteY2LED8vYedyl2gCeREGDysBQt
T4GOZiMiXBfpmXY4wy50mhpVwkCkO1evyrM7x7rPYgapY6ucVm/Gsj9oBC0EMkd+
w1Ro6QvSuJA=
=10J5

-----END PGP SIGNATURE-----

- 
- unique to sender
 - verifies message hasn't been altered



Encrypted Message (default, binary format)

```
pgp -e message_file recipient's_id
```

```
Ñå HîKÓ>Û™      /: aÛî»V-ûé4"Æ ...<ïE ·R´y<¶£  ç/  
¶ ê~U>+S¯ÿfi{q$•`p™α Fγμ  CíùBõÏÿS =0c£?~]Ô*#ØcÊe~Ä ...3ðë-  
ιόνÖçζóuÆ/ q.É°J£J  uQμÑMμ °?<¬\0¶¶  £Âkg È  »iUí{>ç α +=° Ìv~ι'êfi,·,,  lOfIwÜflæ >N  
° $,îë"SLZ ®ð ." ]ÉßÖüÔ!w ôê° } éKEQ íæVzÊ E D^i>à'(ÒqLÀJ°Säâ®I7ã «  
çμα`´´ e" <  @',Á?N|"CÁ1fióP# bα"  ...Û,μÆÄ JRqÖ BhÛ%
```

- The resultant characters are just whatever character happens to correspond to a generated group of bits. These aren't necessarily all printable, and may in fact correspond to control characters.
- A message encrypted to this format is not meant to be viewed or transmitted in its encrypted form, and it is not appropriate for e-mail.
- Use the ASCII output option for encrypted messages meant for e-mail.

```
pgp -eat ...
```



Encrypted Message (ASCII representation)

```
-----BEGIN PGP MESSAGE-----  
Version: 2.6.2
```

ASCII armor:

- e-mail transmission
- xmit through 7-bit channel
- protects against corruption (CRC)

RADIX-64: 3 8-bit bytes => 4 printable ASCII characters (33% growth)

```
hIwDSJRL7j7yqq0BA/9RuXAlBBhodwBeBLDNf1Tm1oHaA01U3w4NXEDqOhrJeLi1  
VBH0mxWTJuiNOzftM7QLqIBVJwL6QOLgxwi4nn42rLrF3+HyYQZ7oHj9ojiCx14X  
vId5UrmMMHw+vWx4aPqgwTcn7IDhskS+DdOZKcj5EUKGSAkTaJPvbsrgOEDBnbqYA  
AAFwoahlleWsky2nosg2k93tjHRuIaGWBMLDCI6dK62Zvne7oxm7jM2mJkI0H2vC  
s3eyjEZky4Vx3M8hentaCueyeXlnpZyze0DxtUiOhGaqivdFGQ2wZ3gpsZE711f9  
J6nRjnOgU++eas3QmF7Ys6SwHKjQnXAm2DtqQd4UbrzVlT5ytz7iZFp1YucVuPiD  
Ismk4Q2N6NV/oiZyHbjOyw6q69/8E/QisJYDy1S3/jXq2euINemd9004AehHGHow  
xkCcx9AHsIK2AzuW7Mr+Sc0bzGQ5QsEmEGYNATeFlDKWB6q6B3yi9sqxzZicU033  
NRQjCyrq5TXqvjKY2R5ZaewnG666Dm8DSbo7PPu9L165J/fioSBnd10DizYhec7H  
Tu68tjqFsFEvftQE93ekIj3dls0TnUqpxfTL4D3e8UTlVDsEJsWb845hyCozQc4E  
Xm6rzEwetovX  
=eXW5  
-----END PGP MESSAGE-----
```

It's not apparent just by looking at it whether or not it's signed.

Steganography — Obscuring Messages



**Original Image
(40K GIF)**



**Image Containing
4K Text File**

Protecting Communications



- **Code**

BANANA	⇒	Bob
NEWSPAPER	⇒	eat lunch
BLUE	⇒	1:00 pm

- **Cipher / Encryption**

Any plaintext ⇒ *Unreadable ciphertext*

- **Steganography**

Obscure the very existence of a message

Patents, Licenses, Lawyers, and Corporations

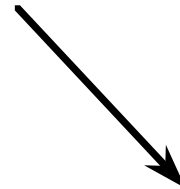


W. Diffie
M. Hellman
R. Merkle
(Stanford)

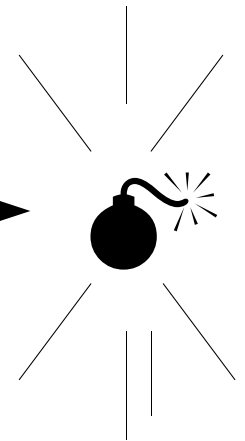


patented the
concept of
public-key
cryptography

Cylink
(Caro-Kann Corp.)



Public Key Partners
(PKP)

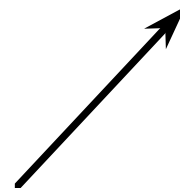


R. Rivest
A. Shamir
L. Adelman
(MIT)



developed the
first workable
public-key
encryption algorithm

RSA Data Security, Inc.
(RSADSI)
Jim Bidzos



PGP™ (Pretty Good Privacy™)



PGP is a program, written by Phil Zimmerman, supported on a wide variety of hardware platforms, which provides:

- **RSA key-pair generation.**
- **Key management.**
- **Data encryption using public-key (RSA) and conventional (IDEA) cryptography.**
- **Digital signatures using MD5 one-way hash function.**

The Birth of “Guerrilla Freeware”



1991: S.266 omnibus anti-crime bill:

communications equipment should be wire-tap ready

⇒ **raised concerns of privacy advocates, and mobilized resistance**

⇒ **Phil Zimmerman released PGP 1.0**

He gave it to a friend who posted it on Usenet → outside U.S.

Results:

- **RSADSI (Bidzos) attacked for patent violation.**
- **U.S. Govt started an investigation regarding criminal export of cryptographic technology.**