

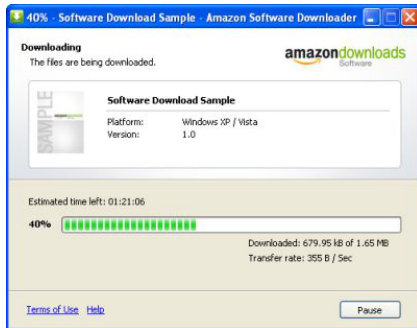
# GNU Privacy Guard

Andrea Lanzi

UNIVERSITÀ DEGLI STUDI DI MILANO  
Facoltà di Scienze Matematiche, Fisiche e Naturali  
Anno Accademico 2014/2015

21 Aprile 2015

## Esempio: download di un applicazione



*Installazione dell'applicazione*



**CORRETTO**

**MD5(software.exe)**

**== 79054025255fb1a26e4bc422aef54eb4**



**SBAGLIATO**

*Nuovo tentativo di  
download dell'applicazione...*

## Esempio: download di un applicazione



*Installazione dell'applicazione*



**CORRETTO**

**MD5(software.exe)**

== 79054025255fb1a26e4bc422aef54eb4



**SBAGLIATO**

*Nuovo tentativo di  
download dell'applicazione...*

## Esempio: download di un applicazione



*Installazione dell'applicazione*

**CORRETTO**

Ma in questo modo  
siamo 100% sicuri  
di quello che stiamo  
installando???

software.exe)

b1a26e4bc422aef54eb4

**BAGLIATO**

*Nuovo tentativo di  
download dell'applicazione...*

## Cifrari simmetrici

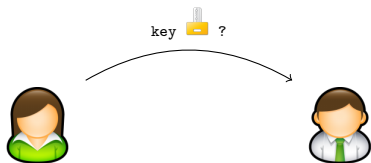
- usano la stessa chiave per effettuare le operazioni di cifratura/decifratura;
- cifrari simmetrici “attuali”: DES, 3DES, Blowfish, IDEA, AES;
- un buon cifrario pone tutta la sicurezza nella chiave e *mai* nell’algoritmo;
  - l’insieme di tutte le chiavi possibili, il *key space*, dev’essere “grande”;
- scambio e gestione delle chiavi problematico ( $n \Rightarrow \frac{n(n-1)}{2}$ )

## Cifrari simmetrici - Funzionamento



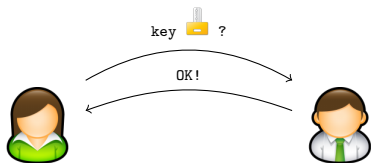
Nel mezzo del cammin di  
nostra vita  
mi ritrovai per una selva  
oscura  
chè la diritta via era  
smarrita.

## Cifrari simmetrici - Funzionamento



Nel mezzo del cammin di  
nostra vita  
mi ritrovai per una selva  
oscura  
chè la diritta via era  
smarrita.

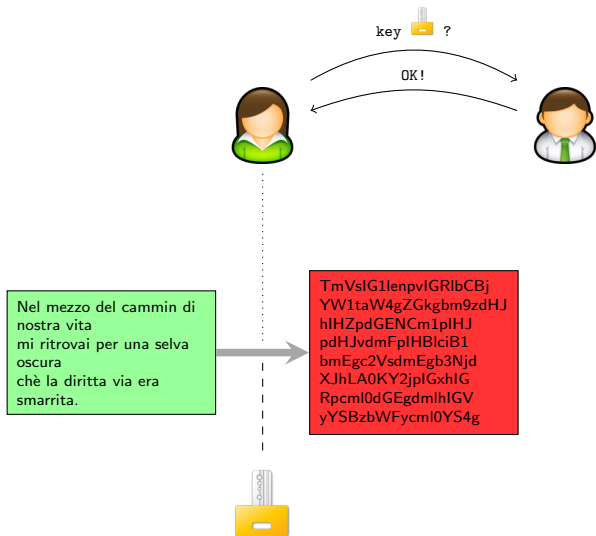
## Cifrari simmetrici - Funzionamento



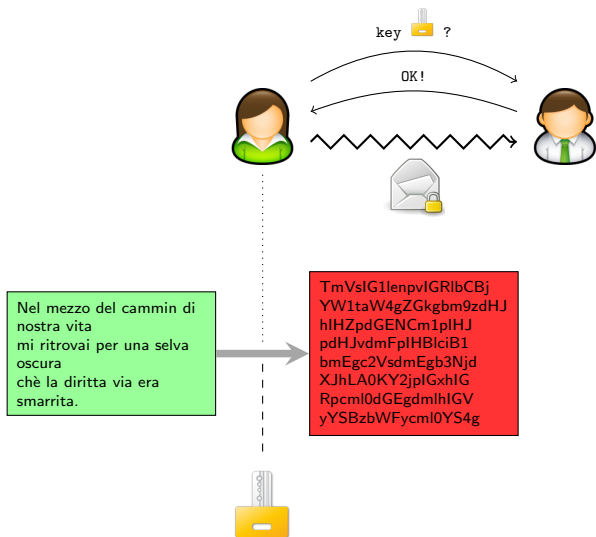
Nel mezzo del cammin di  
nostra vita  
mi ritrovai per una selva  
oscura  
chè la diritta via era  
smarrita.



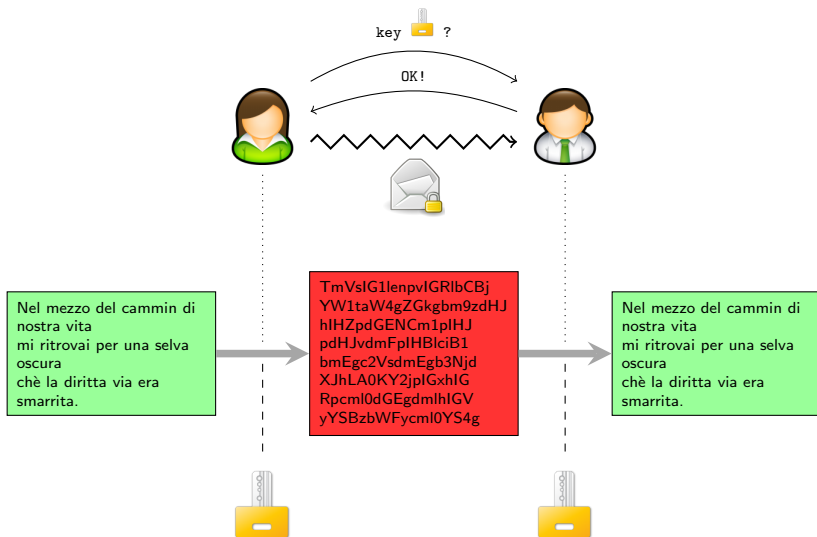
# Cifrari simmetrici - Funzionamento



## Cifrari simmetrici - Funzionamento



# Cifrari simmetrici - Funzionamento



## Cifrari a chiave pubblica

- chiavi generate a coppie:
  - chiave pubblica usata per cifrare/verificare la firma dei documenti;
  - chiave privata usata per decifrare/firmare i documenti;
- si elimina il problema dello scambio delle chiavi e il numero di chiavi necessarie per comunicare con un utente;
- cifrari basati sul concetto di funzione *one-way trapdoor*, e.g. RSA;
  - funzione facile da computare “in un senso”;
  - funzione inversa difficile da computare;
  - *trapdoor*: funzione inversa facile da computare con le dovute ipotesi, e.g. un fattore del modulo di una chiave RSA;

# Cifrari a chiave pubblica - Funzionamento



Nel mezzo del cammin di  
nostra vita  
mi ritrovai per una selva  
oscura  
chè la diritta via era  
smarrita.

## Cifrari a chiave pubblica - Funzionamento



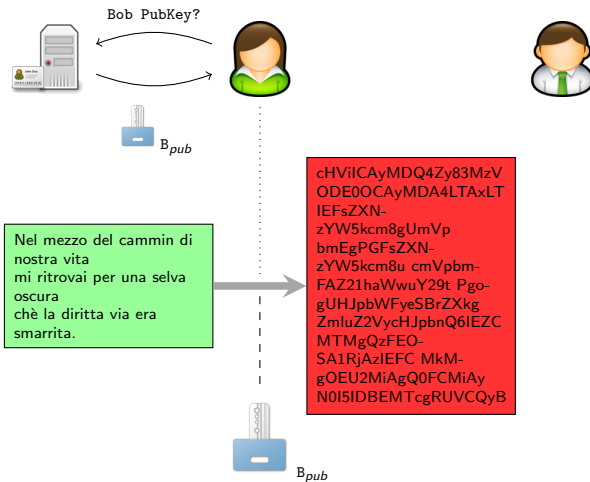
Nel mezzo del cammin di  
nostra vita  
mi ritrovai per una selva  
oscura  
chè la diritta via era  
smarrita.

## Cifrari a chiave pubblica - Funzionamento



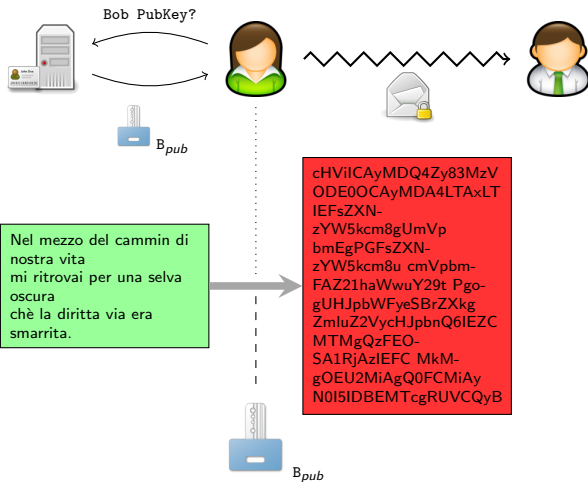
Nel mezzo del cammin di  
nostra vita  
mi ritrovai per una selva  
oscura  
chè la diritta via era  
smarrita.

# Cifrari a chiave pubblica - Funzionamento

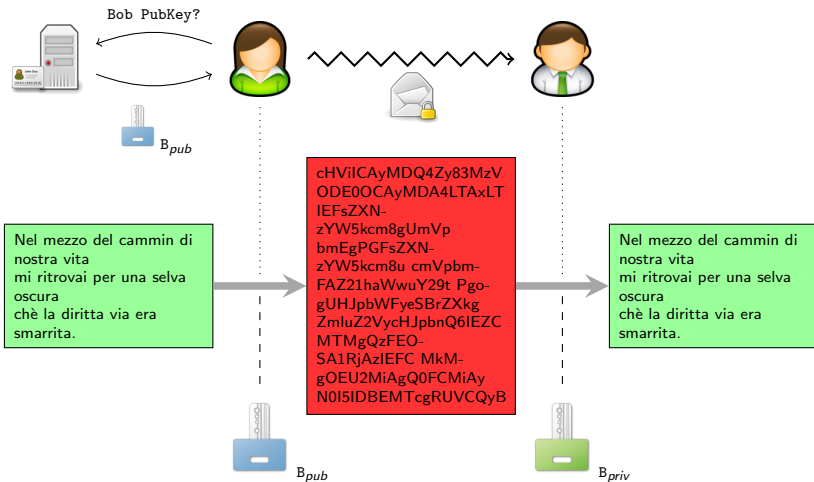




# Cifrari a chiave pubblica - Funzionamento



# Cifrari a chiave pubblica - Funzionamento



## Cifrari ibridi

- ✗ molti cifrari simmetrici sono più “forti” da un punto di vista della sicurezza;
- ✗ cifratura/decifratura effettuata con algoritmi a chiave pubblica sono più dispendiosi in termini di tempo;
- ✓ cifrari asimmetrici sono comunque migliori per quanto riguarda lo scambio e la gestione delle chiavi;
  - cifrari ibridi usano sia cifrari simmetrici che asimmetrici;
    - la chiave pubblica viene usata per cifrare una chiave di sessione, scelta in modo casuale, con la quale si cifra il documento;
    - soltanto il giusto ricevente sarà in grado di decifrare la chiave di sessione, usando la sua chiave privata, per poi decifrare il messaggio ricevuto;
  - sia PGP che GnuPG usando cifrari ibridi.

## Cifrari ibridi

- ✗ molti cifrari simmetrici sono più “forti” da un punto di vista della sicurezza;
- ✗ cifratura/decifratura effettuata con algoritmi a chiave pubblica sono più dispendiosi in termini di tempo;
- ✓ cifrari asimmetrici sono comunque migliori per quanto riguarda lo scambio e la gestione delle chiavi;
  - cifrari ibridi usano sia cifrari simmetrici che asimmetrici;
    - la chiave pubblica viene usata per cifrare una chiave di sessione, scelta in modo casuale, con la quale si cifra il documento;
    - soltanto il giusto ricevente sarà in grado di decifrare la chiave di sessione, usando la sua chiave privata, per poi decifrare il messaggio ricevuto;
  - sia PGP che GnuPG usando cifrari ibridi.

# GNU Privacy Guard

## Introduzione

- GnuPG: strumento per cifrare e firmare dati, usato prevalentemente nella comunicazione tramite email
- Utilizza la crittografia *ibrida*;
  - La chiave privata, considerata segreta, non deve mai essere divulgata; protetta tramite una *passphrase*;
  - La chiave pubblica deve essere comunicata a tutti coloro con i quali l'utente vuole comunicare;
  - Chiave privata usata per decifrare o firmare<sup>a</sup> i dati (messaggi);
  - Chiave pubblica usata per cifrare<sup>b</sup> o verificare la firma di dati (messaggi).

---

<sup>a</sup>in realtà usata insieme a funzioni di hash crittografiche.

<sup>b</sup>in realtà viene cifrata asimmetricamente una chiave di sessione che cifra in modo simmetrico i dati.

# GNU Privacy Guard

## Obiettivi crittografia PGP

- 1 confidenzialità
- 2 integrità
- 3 non-repudiabilità

Se valgono 1 && 2 && 3  $\implies$  *autenticità*.

# GNU Privacy Guard

## Obiettivi crittografia PGP

- 1 confidenzialità
- 2 integrità
- 3 non-repudiabilità

Se valgono 1 && 2 && 3  $\implies$  *autenticità*.

# Chiavi primarie e chiavi subordinate (1)

## Tipo di chiavi e dimensione

Creazione di due chiavi:

- una per la firma digitale (solitamente la primaria);
- una per la cifratura.

```
$ gpg --quiet --gen-key
gpg (GnuPG) 1.2.4; Copyright (C) 2003 Free Software Foundation, Inc.
...
Please select what kind of key you want:
  (1) DSA and ElGamal (default)
  (2) DSA (sign only)
  (3) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
  minimum keysize is 768 bits
  default keysize is 1024 bits
  highest suggested keysize is 2048 bits
What keysize do you want? (1024)
```



## Chiavi primarie e chiavi subordinate (2)

### Durata delle chiavi e User-ID

```
Please specify how long the key should be valid.
  0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct (y/n)? y

You need a User-ID to identify your key; the software constructs ...
Real name: Foo Bar
Email address: foobar@domain.org
Comment:
You selected this USER-ID:
  "Foo Bar <foobar@domain.org>"

Change (N)ame, (C)omment, (E)mail or (0)kay/(Q)uit? 0
```

# Chiavi primarie e chiavi subordinate (3)

## Passphrase, coppia di chiavi e fingerprint

```
You need a Passphrase to protect your secret key.
```

```
Enter passphrase:
```

```
Repeat passphrase:
```

```
...
```

```
public and secret key created and signed.
```

```
key marked as ultimately trusted.
```

```
pub 1024D/8EE5C69A 2004-11-01 Foo Bar <foobar@domain.org>
```

```
Key fingerprint = CA1B 8A00 8EB7 D051 4DCE 376C 1075 81A3 8EE5 C69A
```

```
sub 1024g/C01C49FA 2004-11-01
```

```
$
```

## Scelta di una passphrase \*sicura\*

### La *passphrase*:

- può contenere caratteri di spaziatura
- “no” vincoli sulla lunghezza
- possibilità di utilizzare caratteri, numeri, caratteri speciali [es. “~”, “\$”, “.”, “>”, ...]
- mix di caratteri (ed anche lingue diverse)

### Esempio conseguenze passphrase debole:

**Real name:** Danilo Bruschi

**Email address:** prof.bruschi@docenti.di.unimi.it

**Comment:** profilo e keypair *fittizi* del prof

**PASSWORD: ???**  $\leftarrow [a-z0-9]\{3\}$

## Scelta di una passphrase \*sicura\*

### La *passphrase*:

- può contenere caratteri di spaziatura
- “no” vincoli sulla lunghezza
- possibilità di utilizzare caratteri, numeri, caratteri speciali [es. “~”, “\$”, “.”, “>”, ...]
- mix di caratteri (ed anche lingue diverse)

### Esempio conseguenze passphrase debole:

**Real name:** Danilo Bruschi

**Email address:** prof.bruschi@docenti.di.unimi.it

**Comment:** profilo e keypair *fittizi* del prof

**PASSWORD: ???**  $\leftarrow [a-z0-9]\{3\}$

## Generazione del certificato di revoca (1)

Il *certificato di revoca* è l'unico strumento che può essere usato per invalidare la chiave primaria:

- in caso di perdita della passphrase;
- in caso di compromissione della chiave.

```
$ gpg --output revoke.gpg --gen-revoke 8EE5C69A # <- id della chiave
```

```
sec 1024D/8EE5C69A 2004-11-01 Foo Bar <foobar@domain.org>
```

```
Create a revocation certificate for this key? yes
```

```
Please select the reason for the revocation:
```

- 0 = No reason specified
- 1 = Key has been compromised
- 2 = Key is superseded
- 3 = Key in no longer used
- Q = Cancel

```
(Probably you want to select 1 here)
```

```
Your decision? 0
```

## Generazione del certificato di revoca (2)

```
Enter an optional description; end it with an empty line:
>
Reason for revocation: No reason specified
(No description given)
Is this okay? yes

You need a passphrase to unlock the secret key for
user: "Foor Bar <foobar@domain.org>"
1024-bit DSA key, ID 8EE5C69A, created 2004-11-01

Enter passphrase:

ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; ...
$
```

## Vedere ed esportare chiavi pubbliche

Per poter vedere le chiavi presenti nel proprio *keyring*:

```
$ gpg --list-keys  
/home/foobar/.gnupg/pubring.gpg  
-----  
pub 1024D/8EE5C69A 2004-11-01 Foo Bar <foobar@domain.org>  
sub 1024g/C01C49FA 2004-11-01  
$
```

Per esportare la propria chiave pubblica:

```
$ gpg --armor --export 8EE5C69A  
-----BEGIN PGP PUBLIC KEY BLOCK-----  
Version: GnuPG v1.2.4 (GNU/Linux)  
  
mQGi.....RVra  
.....  
=LT1S  
-----END PGP PUBLIC KEY BLOCK-----  
$
```

# Keyserver

Meccanismo per favorire lo scambio di chiavi

## Keyserver disponibili:

- `hkp://subkeys.pgp.net`
- `hkp://pgp.mit.edu`
- `hkp://pool.sks-keyservers.net`
- `hkp://zimmermann.mayfirst.org`
- `http://keyserver.linux.it/`

## Example

```
gpg --search-keys Aristide Fattori  
gpg --recv-keys 291D712D
```



## Importare chiavi pubbliche

Per poter importare una chiave nel proprio *keyring*:

```
$ gpg --import blake.gpg
gpg: key 9E98BC16: public key imported
gpg: Total number processed: 1
gpg: imported: 1
$ gpg --list-keys
/home/foobar/.gnupg/pubring.gpg
-----
pub 1024D/8EE5C69A 2004-11-01 Foo Bar <foobar@domain.org>
sub 1024g/C01C49FA 2004-11-01

pub 1024D/9E98BC16 1999-06-04 Blake (Executioner) <blake@cyb.org>
sub 1024g/5C8CBD41 1999-06-04
$
```

Una volta importata, la chiave dovrebbe essere *validata*:

- *fingerprint* della chiave
- *firma* della chiave

## Validazione “classica” delle chiavi (1)

Fingerprint di una chiave pubblica:

```
$ gpg --edit-key blake@cyb.org

pub 1024D/9E98BC16 created: 1999-06-04 expires: never trust: -/q
sub 1024g/5C8CBD41 created: 1999-06-04 expires: never
(1) Blake (Executioner) <blake@cyb.org>

Command> fpr
pub 1024D/9E98BC16 1999-06-04 Blake (Executioner) <blake@cyb.org>
  Fingerprint: 268F 448F CCD7 AF34 183E 52D8 9BDE 1A08 9E98 BC16
$
```

- prima di firmare una chiave pubblica, è necessario verificare la correttezza del fingerprint associato ad essa *di persona*, *telefonicamente*, ...;
- firmando la chiave si *attesta* la sua appartenenza all'utente corretto, ma **NON** si dà *fiducia* a questo utente.

## Validazione “classica” delle chiavi (2)

### Firma di una chiave pubblica:

```
Command> sign
pub 1024D/9E98BC16 created: 1999-06-04 expires: never trust: -/q
  Fingerprint: 268F 448F CCD7 AF34 183E 52D8 9BDE 1A08 9E98 BC16

  Blake (Executioner) <blake@cyb.org>

Are you really sure that you want to sign this key
with your key: "Foo Bar <foobar@domain.org>"

Really sign? yes
```

### Verifica della firma effettuata sulla chiave:

```
Command> check
uid Blake (Executioner) <blake@cyb.org>
sig! 9E98BC16 1999-06-04 [self-signature]
sig! 8EE5C69A 2004-11-01 Foo Bar <foobar@domain.org>
```

# Chiave pubblica e chiave privata (1)

## Cifratura

- la cifratura avviene usando la chiave *pubblica* del destinatario del messaggio;
- il destinatario decifra il messaggio ricevuto, usando la sua chiave *privata*, corrispondente alla chiave pubblica usata dal mittente;
- l'opzione `--encrypt` di GnuPG viene usata per *cifrare* un messaggio;
- l'opzione `--decrypt` di GnuPG viene usata per *decifrare* un messaggio.

```
$ gpg --output doc.gpg --encrypt --recipient foobar@domain.org doc
```

## Chiave pubblica e chiave privata (2)

### Decifrazione e cifratura simmetrica

Decifrare un documento:

```
$ gpg --output doc --decrypt doc.gpg
```

Cifrare un documento in modo *simmetrico*:

- “chiave” fornita dall’utente al momento della cifratura;
- rappresenta la *passphrase*, condivisa tra le parti;
- problema dello scambio delle chiavi condivise in modo sicuro, su canali insicuri.

```
$ gpg --output doc.gpg --symmetric doc.gpg  
Enter passphrase:
```

# La firma digitale

## Firmare e comprimere il documento

- una *firma digitale* certifica un documento;
- successive modifiche al documento fanno fallire la verifica della firma
- la firma avviene usando la chiave *privata* dell'utente che firma il documento;
- la verifica della firma viene effettuata usando la chiave *pubblica* corrispondente alla chiave privata usata nella firma;
- l'opzione `--sign` è usata per firmare un documento.

```
$ gpg --output doc.sign --sign doc
```

```
You need a passphrase to unlock the private key for  
user: "Foo Bar <foobar@domain.org>"  
1024-bit DSA key, ID 8EE5C69A, created 2004-11-01  
Enter passphrase:
```

# La firma digitale

Firma in chiaro: clearsigned document

- l'opzione `--sign` firma e *comprime* anche il documento;
- generalmente si vuole mantenere leggibile il documento;
- l'opzione `--clearsign` serve a questo proposito, per firmare "in chiaro", mantenendo leggibile il documento.

```
$ gpg --clearsign doc
```

```
You need a passphrase to unlock the private key for  
user: "Foo Bar <foobar@domain.org>" ...
```

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

```
Hash: SHA1
```

```
[...] <document is here>
```

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

```
Version: GnuPG v0.9.7 (GNU/Linux)
```

```
Comment: For info see http://www.gnupg.org
```

```
iEY\leftECAAYFAjdYQoACgkQJ9S6ULt1dqz6IwCfQ7wP6i/i8HhbcOSKF4ELyQB1 ....
```

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

# La firma digitale

## Firma separata: detached signature

- il documento firmato con `--clearsign` dev'essere modificato per poter recuperare il documento originale;
- per ovviare a questo “inconveniente” esiste un terzo modo per firmare i documenti, *detached signature*.

```
$ gpg [--armor] --output doc.sig --detach-sig doc
```

```
You need a passphrase to unlock the secret key for  
user: "Foo Bar <foobar@domain.org>" ...  
1024-bit DSA key, ID 8EE5C69A, created 2004-11-01
```

```
Enter passphrase:  
$
```



## Web of trust (1)

Grazie al modello “web of trust”, la responsabilità di validare chiavi pubbliche degli utenti viene *delegata* a persone in cui *si ha fiducia*.

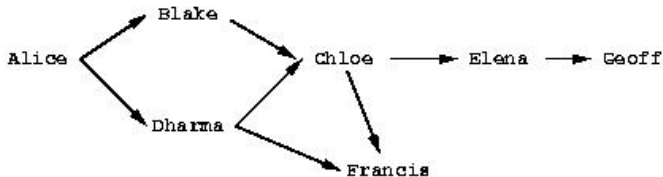
Esistono 4 differenti “trust levels”:

- unknown
- none
- marginal
- full

## Web of trust (2)

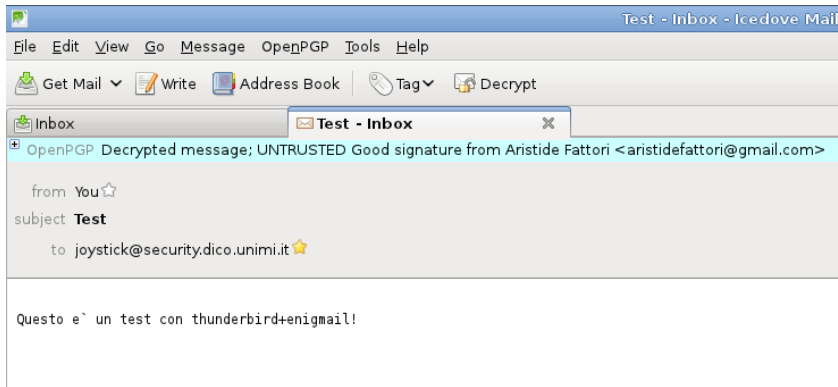
### Algoritmo per stabilire se una chiave $K$ è valida:

- 1 è stata firmata da un numero sufficiente di chiavi valide
  - io ho firmato personalmente la chiave
  - la chiave è stata firmata da un utente in cui io ripongo totale fiducia [full]
  - la chiave è stata firmata da 3 utenti in cui ho fiducia "marginal" &&
- 2 il percorso di chiavi firmate da  $K$  fino alla propria chiave consiste di al più 5 passi



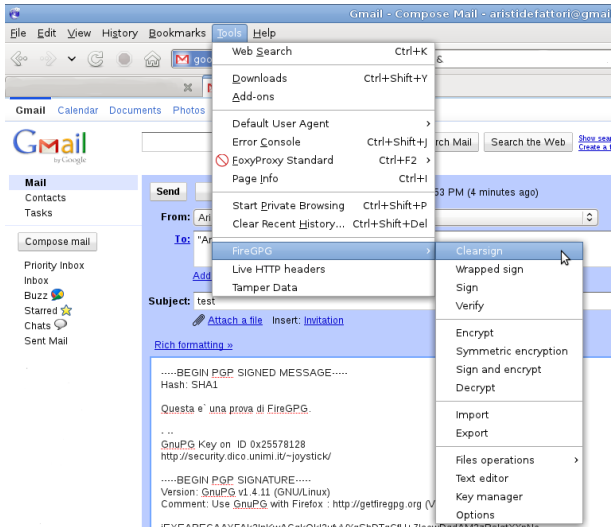
# GPG for Dummies

## Enigmail



# GPG for Dummies

## FireGPG



## 0xA0

Creare il file `clear.txt` [il contenuto è irrilevante]. Usando GPG, cifrare tale file con crittografia **simmetrica** utilizzando gli algoritmi: AES, TRIPLE DES & BLOWFISH. I file così cifrati devono essere salvati come `cipher-[algoname].gpg` (Es: `cipher-3DES.gpg`).

# 0xA1

Decifrare i file cifrati creati al punto precedente.

## 0xB0

Sul sistema sono presenti due utenti `alice` e `bob`. Per entrambi questi utenti generare una coppia di chiavi da 1024 bit.

## 0xB1

Dall'utente `alice`, esportare la chiave pubblica appena creata. Fare lo stesso dall'utente `bob`. Dall'utente `alice`, importare la chiave pubblica di `bob`, controllarne il fingerprint e firmarla con la chiave pubblica di `alice`. Effettuare le stesse operazioni invertendo gli utenti.

### Hint

Usare due shell per evitare i continui login/logout (Ctrl+F2)



## 0xB2

Dall'utente alice creare il file `sign-test.txt` [il contenuto è irrilevante]. Firmare il file con tutti e tre i metodi citati [`sign`, `clearsign`, `detached`] salvando l'output rispettivamente nei file `sign.gpg`, `clear.gpg`, `detached.gpg`. Dall'utente bob verificare le firme di alice sui tre file.

## 0xB3

Dall'utente alice creare il file `encrypt-test.txt` [il contenuto è irrilevante]. Crittare il file con la chiave pubblica di bob salvando l'output nel file `encrypt-test.txt.gpg`. Dall'utente bob decifrare il file.

## 0xB4

Dall'utente alice creare il file `sign-encrypt-test.txt` [il contenuto è irrilevante]. Crittare il file con la chiave pubblica di bob e firmarlo, salvando l'output nel file `sign-encrypt-test.txt.gpg`. Dall'utente bob decifrare il file e verificare la firma.

## 0xC0

Utilizzando l'account "studente", generare la propria coppia di chiavi da 2048 bit ed il rispettivo certificato di revoca.

## 0xC1

Risalire al fingerprint della propria chiave pubblica generata durante l'esercizio precedente.

Inviare via mail la propria chiave al compagno di banco e ricevere la sua. Utilizzare il fingerprint verificare la chiave ricevuta. Importare la chiave pubblica nel proprio keyring e firma di quest'ultima.

## 0xC2

Inviare al proprio compagno di banco la sua chiave pubblica *firmata* in modo che possa importarla nel proprio keyring. Controllare tutte le firme presenti nella propria chiave pubblica.

Ripetere l'esercizio precedente (scambio chiave pubblica e verifica di quest'ultima tramite fingerprint) con un secondo compagno di banco; ulteriore firma della chiave e re-invio al mittente; verifica che ora la propria chiave pubblica abbia almeno 3 firme.

## 0xC3

Utilizzare l'opzione comando `--edit-key` per generare una seconda chiave subordinata.

## 0xC4

Esportare la propria chiave pubblica, in formato ASCII-armored, in un file denominato “my\_new\_key.gpg”.



## 0xD0

Trovare la chiave pubblica (Srdjan Matic) sul keyserver  
<http://zimmermann.mayfirst.org/>.  
Dopo avere verificato che il fingerprint della chiave scaricata  
corrisponda a quello reperibile alla pagina  
<http://security.di.unimi.it/~andrew/srdjan.pub>,  
importare la chiave nel proprio keyring.

## 0xD1

Dall'url <http://security.di.unimi.it/~andrew> scaricare il file "root\_password.zip.gpg".

- estrarre l'archivio utilizzando la cifratura simmetrica e password "**I learned to decrypt!**"
- individuare la password per l'utente root verificando la firma di ciascun messaggio (la firma da verificare è quella associata alla chiave `srdjan@security.di.unimi.it`)

## 0xD1

Dall'url <http://security.di.unimi.it/~andrew> scaricare il file "root\_password.zip.gpg".

- estrarre l'archivio utilizzando la cifratura simmetrica e password "**I learned to decrypt!**"
- individuare la password per l'utente root verificando la firma di ciascun messaggio (la firma da verificare è quella associata alla chiave `srdjan@security.di.unimi.it`)

## 0xD2

Oltre ad essere appassionato di crittografia, Bob adora la musica. Egli ha inviato ad Alice, una mail dal titolo *“Vengo anch’io? No tu no”*; tale mail conteneva un allegato con il nome `song.gpg`. Si sospetta che tale file possa \*nascondere informazioni\* preziose riguardo alla chiave che i due intendono utilizzare per scambiarsi futuri messaggi segreti. Trovare la chiave di 4 caratteri con cui il messaggio è stato crittato.

**ASSUNZIONE:** `ASCII.lower + digits`

Durante un successivo scambio di mail, Bob ha inviato ad Alice un messaggio contenente un file denominato `meeting_at.gpg`; decifrare tale file utilizzando la password trovata al punto precedente.

### Hint

COMANDI: `cat`, `grep`, `for`

FILE-1: `http://security.dico.unimi.it/~andrew/song.gpg`

FILE-2: `http://security.dico.unimi.it/~andrew/meeting_at.gpg`

## 0xD3

TESTO ESERCIZIO:

`http://security.dico.unimi.it/~andrew/0xD3.txt.gpg`