# GNU Privacy Guard (GPG)

*guida base all'utilizzo di GPG per proteggere la propria privacy*


---

Mircha Emanuel `ryuujin` D'Angelo

ryuujin(at)olografix(dot)org

www.olografix.org/ryuujin

# Sommario

- Concetti di crittografia
- Primi passi
- Gestione delle chiavi
- Gestire la propria rete della fiducia

# Concetti di crittografia

principio di Kerckhoffs: "La sicurezza di un sistema crittografico è basata esclusivamente sulla conoscenza della chiave, in pratica si presuppone noto a priori l'algoritmo di cifratura e decifraura."

| *Algoritmo* | *Tipo* | *Esempi* |
|---|---|---|
| Simmetrici (private key) | Stream | RC4, SEAL, WAKE, PKZIP |
| | Block | DES, 3 DES, RC2, RC5, RC6, Blowfish, ... |
| Asimmetrici (public key) | Block | RSA, Diffie-Hellman, Elliptic curve |
| Hash | - - - - - | MD5, SHA-1 |

# Concetti di crittografia

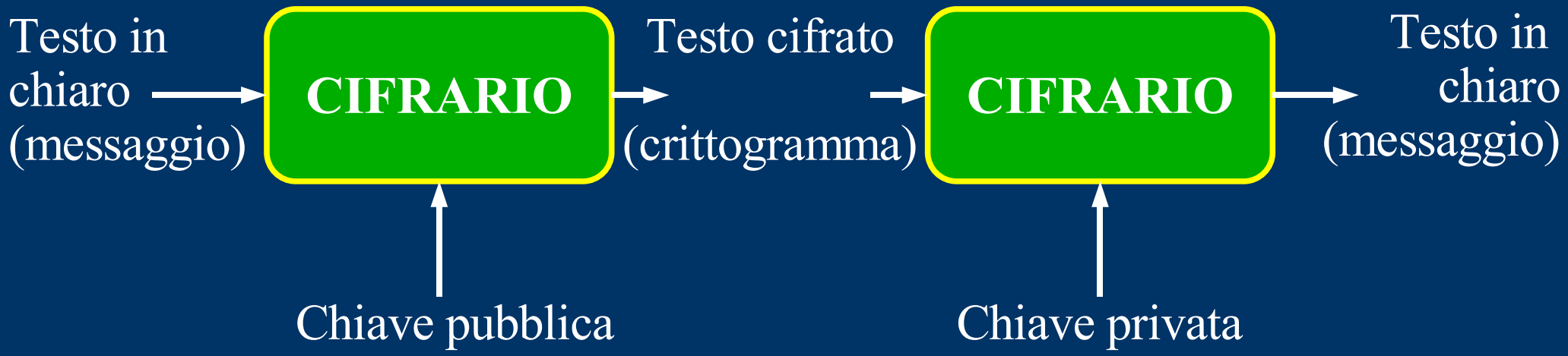
## algoritmi simmetrici


Nel cifrario simmetrico viene usata una sola chiave per cifrare e decifrare il messaggio. Introduciamo un parametro chiamato k (key= chiave) all'interno delle funzioni di cifratura C(m,k) e decifrazione D(c,k).

# Concetti di crittografia

## algoritmi asimmetrici a chiave pubblica

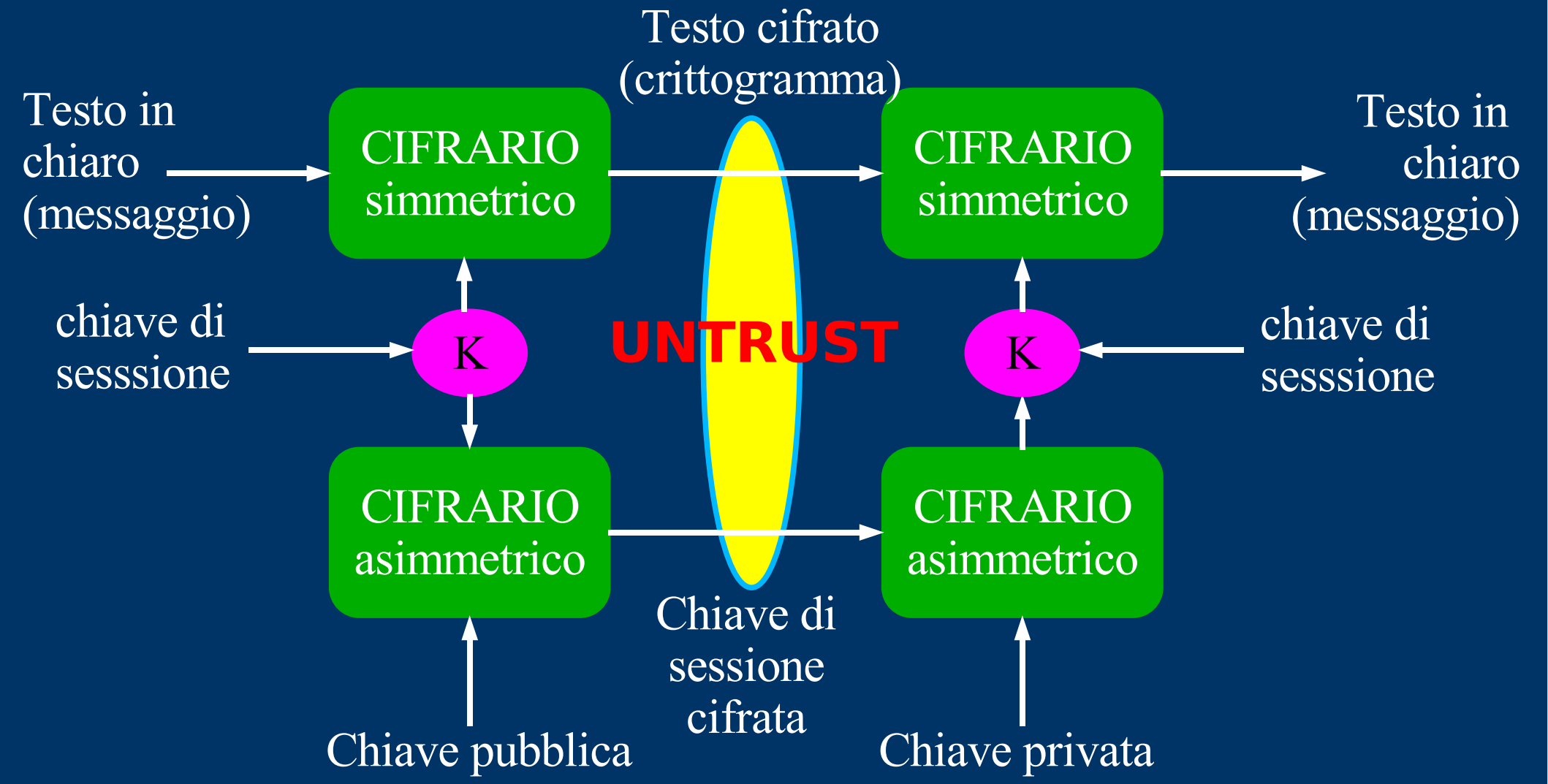
Pensato nel 1976 da Diffie ed Hellmann e poi applicato nel 1977 da Ron Rivest e con l'aiuto di altri due matematici del MIT l'israeliano Adi Shamir, e Leonard Adleman definendo l'RSA (acr. Rivest, Shamir, Adleman)




Un algoritmo a chiave pubblica utilizza una coppia di chiavi per spedire il messaggio: una chiave pubblica che può essere data a chiunque e una chiave privata tenuta strettamente segreta dal suo possessore. Il mittente cifra il messaggio con la chiave pubblica e una volta ricevuto può essere decifrato solo con la chiave privata.

# Concetti di crittografia

## algoritmi ibridi




Metodo usato da GPG, openswan....

# Primi passi con gpg

## generare una coppia di chiavi

```
ryuujin:~$ gpg --gen-key
gpg (GnuPG) 1.2.5; Copyright (C) 2004 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Per favore scegli che tipo di chiave vuoi:
   (1) DSA e ElGamal (default)
   (2) DSA (firma solo)
   (4) RSA (firma solo)
Cosa scegli?
```

# Primi passi con gpg

## generare una coppia di chiavi

...
Per favore scegli che tipo di chiave vuoi:
**(1) DSA e ElGamal (default)**
(2) DSA (firma solo)
(4) RSA (firma solo)
Cosa scegli?

GnuPG è in grado di creare diversi tipi di coppie di chiavi, ma una chiave primaria deve essere capace di fare firme. L'opzione (1) crea in realtà due coppie di chiavi: una coppia DSA che rappresenta le primarie ed è utilizzabile solo per firmare; una coppia subordinata di tipo ElGamal (trapdoor su log discreti), usata per criptare. L'opzione (2) crea solo una coppia di chiavi DSA utili solo per la firma.
In tutti i casi è possibile creare sotto-chiavi in un secondo momento.

# Primi passi con gpg

## generare una coppia di chiavi

...
La coppia DSA avrà 1024 bit.
Sto per generare una nuova coppia di chiavi ELG-E.
la dimensione minima è 768 bit
**la dimensione predefinita è 1024 bit**
la dimensione massima consigliata è 2048 bit
Di che dimensioni vuoi la chiave? (1024)

La resistenza a attacchi a forza bruta è direttamente proporzionale alla lunghezza della chiave; allo stesso tempo il tempo di elaborazione è direttamente proporzionale alla lunghezza della chiave. Una lunghezza di 1024 bit è più che sufficiente.

# Primi passi con gpg

## generare una coppia di chiavi

...
La dimensione richiesta della chiave è 1024 bit
Per favore specifica per quanto tempo la chiave sarà valida.
0 = la chiave non scadrà
<n> = la chiave scadrà dopo n giorni
<n>w = la chiave scadrà dopo n settimane
<n>m = la chiave scadrà dopo n mesi
<n>y = la chiave scadrà dopo n anni
Chiave valida per? (0)

Una chiave scaduta non può essere più utilizzata per firmare o cifrare documenti, mentre è ancora possibile utilizzarla per decifrare.

La scadenza va scelta con cura, anche se è possibile modificarla in seguito potrebbe essere difficoltoso comunicarla agli utenti che possiedono la relativa chiave pubblica.

# Primi passi con gpg

## generare una coppia di chiavi

...
Ti serve un **User ID** per identificare la tua chiave;
...
Hai selezionato questo User Id:
"Mircha Emanuel D'Angelo (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"
...

La chiave è associata ad una persona reale tramite lo User ID. Al momento della generazione della chiave viene creato un solo UID; in seguito è possibile aggiungerne altri (Es. UID per il lavoro, personale, associazione...).
NB: Gli UID creati non possono venir né cancellati né modificati, ma solo revocati.

# Primi passi con gpg

## generare una coppia di chiavi

...
Ti serve un **User ID** per identificare la tua chiave; il software costruisce l'user id a partire da Nome e Cognome, Commento e Indirizzo di Email indicati in questa forma:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

**Nome e Cognome:** Mircha Emanuel D'Angelo
**Indirizzo di Email:** ryuujin@olografix.org
**Commento:** http://www.olografix.org/ryuujin
Hai selezionato questo User Id:
    "Mircha Emanuel D'Angelo (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"

Modifica (N)ome, (C)ommento, (E)mail oppure (O)kay/(Q)uit?

# Primi passi con gpg

## generare una coppia di chiavi

Ti serve una **passphrase** per proteggere la tua chiave segreta.

Inserisci la passphrase:

GnuPG protegge le chiavi per mezzo di una passphrase. Questo è il punto più debole di GnuPG ed in effetti è l'unico dato sicuro che dipende dall'utente finale: la passphrase è l'unica protezione che si possiede nel caso in cui un'altra persona entri in possesso della propria chiave privata.
Per regole su come scegliere una buona password vedi:
http://www.wowarea.com/italiano/aiuto/pwdit.htm

# Primi passi con gpg

## generare una coppia di chiavi

Dobbiamo generare un mucchio di byte casuali. È una buona idea eseguire qualche altra azione (scrivere sulla tastiera, muovere il mouse, usare i dischi) durante la generazione dei numeri primi; questo da al generatore di numeri casuali migliori possibilità di raccogliere abbastanza entropia.

```
+++++++++++.+++++++++++.++++++++++++++++++++++++++++++++
++++++.++++++.++++++++++++++++++++++++++++.++++++++++++++++
..+++++++++++++++++++++..++++++.++++++>++++++++++.....................
.>++++++..............++++++
....
```

chiavi pubbliche e segrete create e firmate.
chiavi marcate definitivamente affidabili.

pub  1024D/E1BE4970 2005-03-24 Nome e Cognome (commento) <email>
    Impronta digitale = D00B 95D8 A9DA 4D8E CFDF  0E35 45EB ED70 E1BE 4970
sub  1024g/6C616F66 2005-03-24[scadenza: 2006-03-24]

# Primi passi con gpg

## generare una coppia di chiavi

```
pub  1024D/E1BE4970 2005-03-24 Nome e Cognome (commento) <email>
     Impronta digitale = D00B 95D8 A9DA 4D8E CFDF  0E35 45EB ED70 E1BE 4970
sub  1024g/6C616F66 2005-03-24[scadenza: 2006-03-24]
```

La creazione della chiave è completa.
La prima riga mostra la chiave pubblica DSA: in ordine sono mostrati lunghezza (1024), tipo (D=DSA), data creazione e UID.
La seconda riga è l'impronta digitale (FPR) della chiave pubblica. Permette di verificare se la copia della chiave è uguale all'originale: se la FPR è identica si può avere (quasi) assoluta certezza che le copie sono identiche.
La terza riga è relativa alla chiave secondaria utilizzata per la cifratura (g = ElGamal).

# Primi passi con gpg

## generare una coppia di chiavi

```
ryuujin:~$ gpg --list-key
/home/ryuujin/.gnupg/pubring.gpg
------------------------------
pub  1024D/08467AFC 2004-04-06 ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>
uid                  shrud <shrud@users.sourceforge.net>
sub  1024g/5028665B 2004-04-06

ryuujin:~$ gpg --fingerprint ryuujin
pub  1024D/08467AFC 2004-04-06 ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>
     Impronta digitale = 77C9 8977 3EDB FEB5 8333  39B9 D022 35D3 0846 7AFC
uid                  shrud <shrud@users.sourceforge.net>
sub  1024g/5028665B 2004-04-06
```

# Primi passi con gpg

## generare un certificato direvoca

Se la chiave viene compromessa o per qualche motivo non è più utilizzabile (es. passphrase dimenticata) un certificato di revoca può essere pubblicato per segnalare ad altri che la chiave pubblica non deve essere più utilizzata. In ogni modo una chiave revocata può essere usata per verificare firme fatte in passato o decifrare documenti spediti in passato (se si possiede ancora l'accesso alla chiave privata).

```
ryuujin:~$ gpg --output revoke_ryuujin.asc --gen-revoke ryuujin
```

--gen-revoke deve essere seguito da una parte qualsiasi dell'UID che specifichi in modo univoco tale chiave. Se l'opzione --output è omessa, il certificato viene stampato nello stdout; l'opzione --output va seguita dal nome del file.

**NB: un certificato di revoca dev'essere custodito con attenzione: se cadesse in mani sbagliate potrebbe venir pubblicato per rendere la chiave pubblica completamente inutilizzabile.**

# Primi passi con gpg

## generare un certificato direvoca

```
ryuujin:~$ gpg --output revoke_ryuujin.asc --gen-revoke ryuujin

sec  1024D/08467AFC 2004-04-06   ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>

Creare un certificato di revoca per questa chiave? si
Per favore scegli il motivo della revoca:
  0 = Nessuna ragione specificata
  1 = Questa chiave è stata compromessa
  2 = Questa chiave è stata sostituita
  3 = La chiave non è più usata
  Q = Cancella
(Probabilmente volevi scegliere 1)
Cosa hai deciso? 1
Inserisci una descrizione opzionale; terminala con una riga vuota:
> La chiave è stata compromessa e non deve essere più utilizzata per cifrare
documenti.
>
```

# Primi passi con gpg

## generare un certificato direvoca

```
ryuujin:~$ gpg --output revoke_ryuujin.asc --gen-revoke ryuujin

sec  1024D/08467AFC 2004-04-06   ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>

Creare un certificato di revoca per questa chiave? si
Per favore scegli il motivo della revoca:
  0 = Nessuna ragione specificata
  1 = Questa chiave è stata compromessa
  2 = Questa chiave è stata sostituita
  3 = La chiave non è più usata
  Q = Cancella
(Probabilmente volevi scegliere 1)
Cosa hai deciso? 1
Inserisci una descrizione opzionale; terminala con una riga vuota:
> La chiave è stata compromessa e non deve essere più utilizzata per cifrare
documenti.
>
```

# Primi passi con gpg

## generare un certificato direvoca

```
ryuujin:~$ gpg --output revoke_ryuujin.asc --gen-revoke ryuujin
...
Motivo della revoca: Questa chiave è stata compromessa
La chiave è stata compromessa e non deve essere più utilizzata per cifrare documenti.
Va bene così? si

Ti serve una passphrase per sbloccare la chiave segreta
dell'utente: "ryuujin (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"
chiave DSA di 1024 bit, ID 08467AFC, creata il 2004-04-06

Forzato l'output con armatura ASCII.
Creato un certificato di revoca.
....
```

Nella creazione di un certificato di revoca non è necessaria l'opzione --armor dato che l'output con armatura ASCII *(ASCII-armored – un formato protetto da un'armatura ASCII simile ai documenti uuencode)* viene forzato di default.

# Primi passi con gpg

## generare un certificato direvoca

```
ryuujin:~$ gpg --output revoke_ryuujin.asc --gen-revoke ryuujin
...
Per favore spostalo su un media che puoi nascondere; se l'uomo nel mezzo
riuscirà ad accedere a questo certificato potrà usarlo per rendere
inutilizzabile la tua chiave. È una buona idea stamparlo ed archiviarlo,
nel caso il media diventasse illeggibile. Ma fai attenzione: il sistema di
stampa della tua macchina potrebbe immagazzinare i dati e renderli disponibili
ad altri!
....
```

Viene ribadita l'importanza di custodire il certificato di revoca. Il certificato di revoca è breve e si può pensare di stamparne una copia e tenerla al sicuro.

# Primi passi con gpg

## Cifrare e decifrare documenti

Una chiave pubblica può essere vista come una cassaforte aperta. Quando un corrispondente cripta un documento utilizzando una chiave pubblica, quel documento viene messo nella cassaforte, la cassaforte viene chiusa ed il lucchetto a combinazione fatto girare diverse volte. La chiave privata corrispodente è la combinazione che può riaprire la cassaforte e recuperare il documento.

Questo implica che chiunque può cifrare un documento utilizzando la mia chiave pubblica, ma solo io che posseggo la chiave privata posso decifrare il documento.

# Primi passi con gpg

## Cifrare e decifrare documenti

Se voglio criptare un documento per isazi, lo cifro utilizzando la sua chiave pubblica. In seguito solo lui, possedendo la chiave privata, potrà recuperare il documento.

```
ryuujin:~$ gpg --output doc.gpg --encrypt --recipient isazi@olografix.org doc
```

L'opzione **--recipient** viene utilizzato una volta sola per ogni destinatario e come argomento richiede un dato che consenta di identificare la chiave pubblica, presente nel nostro mazzo, da utilizzare per la cifratura. Tale documento può essere dectiptato solo da qualcuno in possesso di una chiave privata che complementi una delle chiavi pubbliche dei destinatari. In particolare non è possibile decifrare un documento criptato da voi stessi, a meno che non abbiate incluso la vostra pubkey tra i destinatari.
Il documento cifrato, doc.gpg, è un file binario. Se vogliamo un output ASCII-armored dobbiamo premettere l'opzione **--armor**

```
ryuujin:~$ gpg --armor --output doc.asc --encrypt --recipient isazi@olografix.org doc
```

# Primi passi con gpg

## Cifrare e decifrare documenti

Per decriptare un documento utilizziamo l'opzione **--decrypt**. È necessario possedere la chiave privata con la quale era stato cifrato il messaggio.

```
ryuujin:~$  gpg --output doc --decrypt doc.gpg

You need a passphrase to unlock the secret key for
user: "ryuujin (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"
1024-bit ELG-E key, ID 5028665B, created 2004-04-06 (main key ID 08467AFC)

Inserisci la passphrase:

gpg: encrypted with 1024-bit ELG-E key, ID 5028665B, created 2004-04-06
      "ryuujin (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"
```

# Primi passi con gpg

## Fare e verificare firme

Una firma digitale certifica e appone la data ad un documento. Se il documento viene successivamente modificato in qualsiasi modo, una verifica della firma fallirà.

La creazione e la verifica di firme utilizzano la coppia di chiavi pubblica/privata in modo differente dalle operazione di cifratura e decifratura. Una firma è fatta utilizzando la chiave privata di colui che firma. La firma viene verificata utilizzando la corrispondente chiave pubblica.

Una conseguenza dell'utilizzo di firme digitali consiste nel fatto che è difficile negare di aver apposto una firma, in quanto ciò implicherebbe che la propria chiave privata era stata compromessa.

# Primi passi con gpg

## Fare e verificare firme

Per firmare digitalmente un documento utilizziamo l'opzione **--sign**. Il documento da firmare è l'ingresso, quello firmato è l'uscita:

```
ryuujin:~$  gpg --output doc.sig --sign doc

You need a passphrase to unlock the secret key for
user: "ryuujin (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"
1024-bit DSA key, ID 08467AFC, created 2004-04-06

Inserisci la passphrase:
```

Il documento viene compresso prima di essere firmato e l'output è in formato binario.
Come per la cifratura, possiamo forzare l'output in ASCII-armored utilizzando l'opzione **--armor**.

# Primi passi con gpg

## Fare e verificare firme e documenti firmati in chiaro

Un uso comune di firme digitali consiste nel firmare messaggi di posta. In questi casi è desiderabile lasciare il documento in chiaro.
Con l'opzione **--clearsign** avvolgiamo il documento in una firma ASCII-armored senza modificarlo in alcun modo:

```
ryuujin:~$ gpg --clearsign doc

You need a passphrase to unlock the secret key for
user: "ryuujin (http://www.olografix.org/ryuujin) <ryuujin@olografix.org>"
1024-bit DSA key, ID 08467AFC, created 2004-04-06

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
[...]
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.1 (GNU/Linux)
iEYEARCAAY[...]
=y6kj
-----END PGP SIGNATURE-----
```

# Primi passi con gpg

## fare e verificare firm e –verifica

Con **--verify** controlliamo la firma del documento, per estrarre il documento e controllare la firma utilizziamo l'opzione **--decrypt**:

```
ryuujin:~$ gpg --verify doc.sig
gpg: Signature made gio 01 set 2005 15:25:55 CEST using DSA key ID 08467AFC
gpg: Good signature from "ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>"
gpg:                 aka "shrud <shrud@users.sourceforge.net>"
gpg:                 aka "ryuujin (http://www.dc3946.org [DEFCON: italian defcon users
group]) <ryuujin@dc3946.org>"

ryuujin:~$ gpg [--output doc] --decrypt doc.sig
[...]
gpg: Signature made gio 01 set 2005 15:25:55 CEST using DSA key ID 08467AFC
gpg: Good signature from "ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>"
gpg:                 aka "shrud <shrud@users.sourceforge.net>"
gpg:                 aka "ryuujin (http://www.dc3946.org [DEFCON: italian defcon users
group]) <ryuujin@dc3946.org>"
```

# Primi passi con gpg

## Fare firme - firme distaccate

Con l'opzione **--detach-sig** noi creiamo una firma distaccata in un file separato:

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

You need a passphrase to unlock the secret key for
user: "Mircha Emanuel D'Angelo (Linux Uber Alles) <m.dangelo@tenovisnewtel.it>"
1024-bit DSA key, ID B4242D00, created 2004-03-09

Inserisci la passphrase:
```

Sia il documento che la firma distaccata sono necessarie per verificare la firma stessa. L'opzione **--verify** può essere utilizzata per controllare la firma.

```
ryuujin:~$ gpg --verify doc.sig doc
gpg: Signature made gio 01 set 2005 15:25:55 CEST using DSA key ID 08467AFC
gpg: Good signature from "ryuujin (http://www.olografix.org/ryuujin)
<ryuujin@olografix.org>"
gpg:                 aka "shrud <shrud@users.sourceforge.net>"
gpg:                 aka "ryuujin (http://www.dc3946.org [DEFCON: italian defcon users
group]) <ryuujin@dc3946.org>"
```

# Primi passi con gpg

## Scambiarsi le chiavi – esportare una chiave pubblica

```
ryuujin:~$ gpg --export ryuujin > pubkey_ryuujin.gpg
ryuujin:~$ gpg --output pubkey_ryuujin.gpg --export ryuujin
```

I due comandi sono equivalenti. Con l'opzione --export UID esportiamo la chiave pubblica corrispondente all'UID indicato. In questo modo esportiamo la chiave in binario, il che può essere sconveniente per pubblicarla o scambiarla.

```
ryuujin:~$ gpg --armor --output pubkey_ryuujin.asc --export ryuujin
```

Aggiungendo l'opzione --armor forziamo l'output in ASCII-armored. Una chiave esportata in ASCII-armored è facilmente distribuibile e pubblicabile sul web. In generale l'opzione --armor si può applicare anche alla cifratura e alla firma dei documenti.
Da notare l'estensione asc: i file armored hanno usualmente questa estensione, ma nulla vieta di omettere o modificare l'estensione.

# Primi passi con gpg

## Scambiarsi chiavi – esportare una chiave pubblica

```
ryuujin~$ gpg --armor --export ryuujin
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.5 (GNU/Linux)

mQGiBEByZj0RBAD6Iz2HhtVvlnpdJ28J/NbzBr33RMMYyfpqPXMPnQsSldvF5+8V
z0yS39FmXY3Q2seONZ9ez3XVVcex2WU0H2JQoBa9UJS4bGl9TaoHbvCTRP9N6q+Q
FGWjGyYvioDcj2gy56dCp531jEG1xXqnbqEn6rWpAaz4fpyTslFTmV2kWwCg2iDG
4avzwqT586XJg0xcLyCNQr8EAOCGqI56UzrAsLU2HF7YDbnh2BBKOj9MePObIu+J
[...]
aNZr7p86NHsylhs6c8ZAN7wzOEv5LRxyrz76z99tJ4QBSMkr+hO3b3pa4i1iaphR
0Rr8jXDnNukEF6VsDPXnBtt4UlP/R9VxLBuwcx0RGCw4FpE9BRUuMOvU0fXaYmvZ
tNZaP0zBtcn09QhoIhxCBbOtRxdHmdZ+iEYEGBECAAYFAkByZkIACgkQ0CI10whG
evzbugCgm727oqQIGxcSwK2yStSquj8g1jcAoItAFMeGj6F4c/2xD/YZtdSGW2nf
=kNxw
-----END PGP PUBLIC KEY BLOCK-----
```

Forma “armorizzata” di una chiave pubblica. Da notare che i delimitatori vanno esportati insieme alla chiave.

# Primi passi con gpg

## Scambiarsi le chiavi – importare una chiave pubblica

```
ryuujin:~$ gpg --import musashi.asc
gpg: chiave EACB1768: importata la chiave pubblica "Serafino Sorrenti
<musashi@linux.it>"
gpg: Numero totale esaminato: 1
gpg:               importate: 1
```

L'importazione di una chiave pubblica si effettua tramite l'opzione --import seguita dal file contenente la chiave (binario o armored).
Una chiave importata deve essere convalidata: bisogna accertarsi che appartenga alla persona designata e che non sia stata compromessa nel trasferimento. GnuPG richiede all'utente di convalidare ogni chiave che viene importata. Per convalidare una chiave si verifica la FPR della chiave importata e succesivamente si firma certificandola come valida.

# Primi passi con gpg

## Scambiarsi chiavi – convalidare una chiave importata

Per convalidare una chiave è necessario verificare l'FPR:

```
ryuujin:~$  gpg --fingerprint isazi
pub  1024D/2AA2A2E2 2003-05-19 isazi (http://www.ografix.org/isazi)
<isazi@ografix.org>
     Impronta digitale = 33DB EC17 4368 787B CE65  CDFB 1A70 5532
2AA2 A2E2
sub  2048g/2FDDEC36 2003-05-19
```


La FPR così calcolata va confrontata con quella del possessore della chiave. Ciò andrebbe fatto utilizzando un mezzo che ci garantisce che stiamo comunicando con il vero possessore della chiave: di persona, per telefono...

Per leggere la FPR di una chiave importata utilizziamo l'opzione --fingerprint o in fase di editor il comando fpr.

# Primi passi con gpg

## Scambiarsi le chiavi – convalidare una chiave importata

Per visualizzare la FPR in fase di editazione della chiave:

```
ryuujin:~$ gpg -u ryuujin --edit-key isazi
[...]
pub  1024D/2AA2A2E2  creata: 2003-05-19 scade: mai       fiducia: -/f
sub  2048g/2FDDEC36  creata: 2003-05-19 scade: mai
(1). isazi (http://www.olografix.org/isazi) <isazi@olografix.org>
(2)  [revocata]isazi (http://www.pescarafree.org)
<isazi@pescarafree.org>

Comando> fpr
pub  1024D/2AA2A2E2 2003-05-19 isazi
(http://www.olografix.org/isazi) <isazi@olografix.org>
 Impronta digitale della chiave primaria: 33DB EC17 4368 787B CE65
CDFB 1A70 5532 2AA2 A2E2
[...]
```

# Primi passi con gpg

## Scambiarsi le chiavi – convalidare una chiave importata

Per certificare una chiave è necessario editarla e firmarla in modo da convalidarla. Utilizziamo l'opzione **--edit-key** per editare una chiave seguida dall'UID. L'opzione **-u** (equivalente a **--local-user**) permette, nel caso avessimo più "mazzi di chiave" personali, di scegliere quale usare; se non viene specificato viene utilizzato il mazzo di default.

```
ryuujin:~$ gpg -u ryuujin --edit-key isazi
[...]
Comando>
```

Dopo aver controllato la FPR ed essersi accertati dell'autenticità della chiave possiamo procedere alla sua convalida con il comando **sign**.

```
Comando> sign
```

GnuPG ci chiederà quale grado di attenzione abbiamo mostrato nella verifica della FPR e se vogliamo firmare tutti gli uid della chiave.

# Primi passi con gpg

## Scambiarsi le chiavi – convalidare una chiave importata

Comando> **sign**
Firmo davvero tutti gli user ID? si
[...]
    isazi (http://www.olografix.org/isazi) <isazi@olografix.org>

Con quanta attenzione hai verificato che la chiave che stai per firmare appartiene veramente alla persona indicata sopra?
Se non sai cosa rispondere digita "0".

(0) Preferisco non rispondere. (predefinito)
(1) Non l'ho controllata per niente.
(2) L'ho controllata superficialmente.
(3) L'ho controllata molto attentamente.

Cosa scegli? (inserisci '?' per ulteriori informazioni):

# Primi passi con gpg

## Scambiarsi le chiavi – convalidare una chiave importata

...
Cosa scegli? (inserisci '?' per ulteriori informazioni): 3
Sei davvero sicuro di volere firmare questa chiave
con la tua chiave: "XXX" (KeyID)

Ho controllato questa chiave molto attentamente.

Firmo davvero? **si**

Ti serve una passphrase per sbloccare la chiave segreta
dell'utente: "XXX"
chiave DSA di 1024 bit, ID KeyID, creata il <data>

Inserisci la **passphrase**:
Comando>

# Primi passi con gpg

## Scambiarsi le chiavi – convalidare una chiave importata

```
Comando> check
uid  isazi (http://www.olografix.org/isazi) <isazi@olografix.org>
[...]
sig!3       2AA2A2E2 2003-05-19   [autofirma]
sig!3       1521E409 2003-05-25   neuro <neuro@olografix.org>
[...]
Comando>
```

Con il comando **check** possiamo listare le firme di convalida della chiave importata (ci tornerà molto utile quando parleremo della *"rete della fiducia"*.
Anticipiamo che una firma di convalida può essere revocata in seguito.
NB: la chiave non sarà effettivamente firmata fino a quando non salveremo i cambiamenti.

# Primi passi con gpg

## Fiducia nelpossessore della chiave

Per evitare di dover convalidare tutte le chiavi in nostro possesso, GnuPG ha un meccanismo denominato **“rete della fiducia”**. Nel modello della “rete della fiducia” la responsabilità di convalidare le chiavi pubbliche è delegata a persone di cui ci si fida.
Se io sono convinto che isazi sia capace di convalidare propriamente le chiavi che firma, posso dedurre che le chiavi firmate da isazi siano valide senza dover eseguire alcun controllo.
La fiducia è soggettiva e il modello della “rete della fiducia” tiene in considerazione questo carattere soggettivo associando ad ogni chiave pubblica presente nel proprio mazzo un'indicazione di quanto ci si fidi del possessore di quella chiave.
I livelli di fiducia sono 5: *sconosciuto, nessuna, marginale, completamente e definitivamente.*
Il livello di fiducia si assegna da soli e viene considerato un'informazione privata: non viene esportata insieme alla chiave.

# Primi passi con gpg

## Fiducia nelpossessore della chiave

Per assegnare il livello di fiducia ad una chiave utilizziamo il comando **trust** in fase di editing della chiave stessa.
La rete della fiducia permette di utilizzare un algoritmo più elaborato per convalidare una chiave; una chiave K è considerata valida se soddisfa due condizioni:

- è firmata da un numero sufficiente di chiavi valide, cioé
  - è stata firmata di persona, oppure
  - è stata firmata da una chiave di cui ci si fida pienamente, oppure
  - è stata firmata da tre chiavi con fiducia marginale
- il percorso delle chiavi firmate che risale dalla chiave K alla propria chiave è al massimo di 5 passi

# Primi passi con gpg

## Rete della fiducia