

# Generazione sicura di chiavi OpenPGP

Aldo Latino

1 aprile 2018 – 14 luglio 2019

## Sommario

Una guida dettagliata, per quanto possibile, sulla generazione, conservazione e protezione delle chiavi OpenPGP. La guida è stata pubblicata la prima volta su [GitHub](#).

## Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>Generazione delle chiavi</b>	<b>4</b>
2.1	Prerequisiti . . . . .	4
2.2	Generare le chiavi . . . . .	5
2.3	Il certificato di revoca . . . . .	9
2.4	Impostare le preferenze della chiave . . . . .	9
2.5	Impostare il file <code>gpg.conf</code> . . . . .	10
2.6	Impostare il keyserver . . . . .	12
2.7	Copiare la directory <code>.gnupg</code> nel proprio computer di lavoro . . . . .	12
2.8	Esportare la chiave pubblica sul keyserver . . . . .	12
2.9	Segnarsi la data di scadenza delle proprie chiavi nel calendario . . . . .	12
<b>3</b>	<b>Fare il backup</b>	<b>12</b>
3.1	Backup della directory di GnuPG . . . . .	13
3.2	Backup delle chiavi . . . . .	13
3.3	Backup del certificato di revoca . . . . .	14
3.4	Spostare la directory di backup in un luogo sicuro . . . . .	14
<b>4</b>	<b>Trasferire le chiavi OpenPGP su YubiKey</b>	<b>15</b>
<b>5</b>	<b>Eliminare la chiave principale</b>	<b>18</b>
5.1	GnuPG versione 2.1 o successiva . . . . .	18
5.2	GnuPG precedente alla versione 2.1 . . . . .	19
<b>6</b>	<b>Gli agenti</b>	<b>20</b>
<b>7</b>	<b>Importare la chiave principale per operazioni speciali</b>	<b>21</b>
<b>8</b>	<b>Estendere la validità delle nostre chiavi</b>	<b>22</b>
<b>9</b>	<b>Bibliografia</b>	<b>23</b>

# 1 Introduzione

Come sappiamo, tutto il sistema della [crittografia a chiave pubblica](#) si basa sull'uso di una chiave che ha una parte pubblica, da divulgare il più possibile, e una parte privata, che va tenuta rigorosamente in un luogo sicuro. Qualora avessimo il dubbio che le chiavi private possano essere in mani altrui, saremmo obbligati a generare un nuovo mazzo di chiavi, con tutto quello che ne comporta (revoca delle chiavi, perdita del lavoro fatto per il [Web of Trust](#), perdita delle firme apposte sulla nostra chiave).

Con questa guida vedremo come:

1. **generare** in un luogo abbastanza sicuro il nostro mazzo di chiavi personale;
2. **proteggere** le chiavi private in modo da usarle ogni giorno in un modo abbastanza sicuro.

Generalmente un mazzo di chiavi personale è composto in modo predefinito da:

1. una **chiave master (o principale)** per la firma e la certificazione;
2. una **sottochiave** per la cifratura.

Noi, invece, per elevare la sicurezza del nostro portachiavi spingeremo al massimo il concetto delle sottochiavi. La struttura del nostro portachiavi personale, infatti, sarà il seguente:

1. **Chiave master (o principale) per sola certificazione di chiavi altrui** con dimensione 4096 bit e scadenza dopo 3 anni;
  - (a) **Sottochiave per firma** con dimensione 2048 bit e scadenza dopo 1 anno;
  - (b) **Sottochiave per cifratura** con dimensione 2048 bit e scadenza dopo 1 anno;
  - (c) **Sottochiave per autenticazione** con dimensione 2048 bit e scadenza dopo 1 anno.

Avremo quindi una chiave distinta per ogni operazione.

La chiave più importante è la **chiave master**. Questa chiave è talmente importante che, qualora fosse stata compromessa, ci costringerebbe a revocare l'intero mazzo di chiavi e a generarne di nuove. Per questo motivo la terremo in cassaforte, separata dal resto del mazzo di chiavi. Visto che la terremo separata dal resto, la abiliteremo solo per scopi che non sono quotidiani, ma solo periodiche (come ad esempio il rinnovo della validità delle chiavi) o per occasioni speciali (come ad esempio un [Key signing party](#)). Lo scopo della chiave master sarà quello di certificare chiavi altrui ed, essendo la chiave principale, di effettuare operazioni di manutenzione del nostro mazzo di chiavi (estensione validità, aggiunta di nomi utente, ecc.).

Le **sottochiavi**, invece, saranno utilizzate quotidianamente. Le sottochiavi hanno la caratteristica che possono esistere nel portachiavi di GnuPG anche senza la presenza della chiave principale. Per questo motivo, dopo aver fatto un sicuro backup, elimineremo dal portachiavi la chiave principale, lasciando solo le sottochiavi. Le sottochiavi, per loro natura, potrebbero anche essere compromesse (cosa che non dobbiamo permettere mai e, a tal scopo, vedremo come proteggerle), ma questo non invaliderebbe l'intero mazzo di chiavi: qualora una sottochiave fosse compromessa, potremmo revocarla e crearne un'altra per lo stesso scopo. Le sottochiavi potrebbero anche scadere e,

qualora non volessimo rinnovarle, potremmo generarne di nuove. Oppure potremmo generare una sottochiave per un evento particolare (un convegno o altro), che ha quindi una durata limitata (un mese, un anno) e lasciarla scadere dopo la fine dell'evento.

Per proteggere le nostre sottochiavi, le sposteremo su **un token USB come la YubiKey**. Si tratta di token praticamente indistruttibili, che rendono l'uso della firma digitale, la cifratura e l'autenticazione su sistemi (come SSH) molto comodi (basterà usare un PIN numerico al posto della passphrase). Io uso la **YubiKey 4**. I token come YubiKey hanno il pregio che non permettono l'esportazione delle chiavi lì conservate; questo significa che posso usare le mie chiavi anche in un ambiente poco sicuro come un Internet Point, un PC condiviso, ecc., senza il timore che le chiavi private possano essere copiate da qualche parte a mia insaputa. Quand'anche fosse stato intercettato il PIN da un keylogger o da un malware, la passphrase è ancora al sicuro, ed eventualmente posso cambiare il PIN. Il fatto che le sottochiavi, una volta trasferite, non possano essere esportate dalla YubiKey, ci forzerà a effettuare il backup del nostro mazzo di chiavi prima del trasferimento, ma questo aspetto lo vedremo al momento opportuno.

Alla fine della guida avremo quindi la seguente situazione:

1. la chiave principale sarà rimossa dal portachiavi sul disco e trasferita in un luogo sicuro;
2. le sottochiavi risiederanno sulla YubiKey.

Nel nostro disco fisso non avremo quindi nessuna chiave privata. Potremmo anche perdere il nostro PC o potrebbero anche rubarcelo, ma il nostro mazzo di chiavi sarà sempre al sicuro. Potremmo anche perdere la nostra YubiKey, ma essa contiene solo le sottochiavi che, come abbiamo visto, possiamo revocare con la chiave principale (che sta in cassaforte) e ricrearne di nuovi. Quando sarà necessario effettuare un'operazione speciale, come ad esempio apporre la nostra firma su chiavi altrui o aggiungere/eliminare una sottochiave o una UserID, importeremo la chiave principale nel nostro portachiavi, effettueremo l'operazione necessaria e infine la rimuoveremo di nuovo.

Un ultimo punto prima di passare alla pratica. Il motivo di avere una scadenza sulle chiavi è un'ulteriore sicurezza per noi. Immaginiamo il caso in cui dovessimo perdere il controllo del nostro mazzo di chiavi, compresa la chiave principale, e non avessimo più a disposizione nemmeno il certificato di revoca. In questo caso la scadenza sarà una garanzia per noi che a un certo punto le chiavi scadranno in modo naturale. Per i tempi di scadenza ognuno può scegliere quello che più ritenga opportuno, ma 3 anni per la principale e 1 anno per le sottochiavi dovrebbe essere un buon compromesso tra la seccatura di dover mantenere il portachiavi e il tempo di scadenza naturale dopo eventuale compromissione.

Questa guida può essere seguita in modo modulare:

- si può scegliere di mantenere tutte le chiavi sul proprio disco fisso, seguendo solo la parte sulla generazione sicura delle chiavi;
- si può scegliere di spostare in un luogo sicuro la chiave principale dal resto del proprio mazzo di chiavi, mantenendo nel disco fisso solo le sottochiavi;
- si può scegliere di spostare le sottochiavi in un token USB come la YubiKey, mantenendo la chiave principale nel disco fisso;

- si può scegliere di spostare le sottochiavi in un token USB come la YubiKey e di spostare la chiave principale in un luogo sicuro, non lasciando nessuna chiave privata nel disco fisso.

Una nota prima di cominciare. Lungo la guida i vari comandi da terminale useranno ID di chiave, keygrip, percorsi nel disco fisso e quant'altro che fanno riferimento a quanto eseguito per scrivere la guida. Ciò significa che dovrete sostituire gli ID di chiave, i keygrip, i percorsi e quant'altro con quelli vostri. Ad esempio, in un comando come questo:

```
gpg --with-keygrip --list-key 0x9F676B5A4B6E6777
```

dovrete sostituire 0x9F676B5A4B6E6777 con l'ID corretto della vostra chiave.

Inoltre, il percorso /home/user va modificato inserendo il nome utente corretto al posto di user.

## 2 Generazione delle chiavi

Con questa guida genereremo la nostra chiave privata che avrà questo schema:

### 1. Chiave master (o principale) [C - Certification/Certificazione]

- (a) **Sottochiave per firma** [S - Signing/Firma]
- (b) **Sottochiave per cifratura** [E - Encryption/Cifratura]
- (c) **Sottochiave per autenticazione** [A - Authentication/Autenticazione]

Per la lunghezza delle chiavi avremo la chiave primaria a 4096 bit mentre le sottochiavi a 2048 bit. Per ulteriori informazioni su questo dibattito della lunghezza delle chiavi si può vedere:

- [Why does GnuPG default to 2048 bit RSA-2048?](#) sul sito di GnuPG;
- [The Big Debate, 2048 vs. 4096, Yubico's Position](#) sul blog di Yubico.

### 2.1 Prerequisiti

1. È necessario lavorare su un sistema avviato con una distribuzione Linux in modalità *live*. È preferibile una [Tails](#), ma va bene una qualunque. La distribuzione va avviata isolandola da qualsiasi rete.
2. È necessario usare GnuPG versione 2.1 o successiva.
3. È necessario un token USB come, ad esempio, una YubiKey, nel caso si decida di spostare le sottochiavi su questo supporto.

Qualora il comando gpg lanci GnuPG versione 1, accertarsi che la versione 2 sia installata e aggiungere questa riga a /home/user/.bash\_aliases:

```
alias gpg='gpg2'
```

Questa scorciatoia permetterà l'avvio di GnuPG versione 2 ogni volta che digitiamo gpg.

## 2.2 Generare le chiavi

Sul sistema di generazione delle chiavi accertarsi di avere GnuPG versione 2.1 o successiva:

```
gpg --version
```

Iniziare quindi la procedura con:

```
gpg --expert --full-gen-key
```

che darà:

```
Please select what kind of key you want:
```

- (1) RSA and RSA (default)
- (2) DSA and Elgamal
- (3) DSA (sign only)
- (4) RSA (sign only)
- (7) DSA (set your own capabilities)
- (8) RSA (set your own capabilities)
- (9) ECC and ECC
- (10) ECC (sign only)
- (11) ECC (set your own capabilities)

```
Your selection?
```

Scegliamo 8:

```
Possible actions for a RSA key: Sign Certify Encrypt Authenticate
```

```
Current allowed actions: Sign Certify Encrypt
```

- (S) Toggle the sign capability
- (E) Toggle the encrypt capability
- (A) Toggle the authenticate capability
- (Q) Finished

```
Your selection?
```

Come si vede le attuali capacità della chiave che stiamo creando sono: Sign Certify Encrypt. Rimuoviamo le capacità di firma premendo S:

```
Possible actions for a RSA key: Sign Certify Encrypt Authenticate
```

```
Current allowed actions: Certify Encrypt
```

- (S) Toggle the sign capability
- (E) Toggle the encrypt capability
- (A) Toggle the authenticate capability
- (Q) Finished

```
Your selection?
```

Rimuoviamo le capacità di cifratura premendo E:

```
Possible actions for a RSA key: Sign Certify Encrypt Authenticate
```

```
Current allowed actions: Certify
```

- (S) Toggle the sign capability
- (E) Toggle the encrypt capability
- (A) Toggle the authenticate capability
- (Q) Finished

Your selection?

Ora la chiave primaria avrà solo la capacità di certificazione (Certify) su altre chiavi.

Premiamo Q:

RSA keys may be between 1024 and 4096 bits long.  
What keysize do you want? (2048)

Inserire 4096:

Requested keysize is 4096 bits  
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)

Inseriamo 3y (3 anni):

Key expires at mer 31 mar 2021 19:46:05 CEST  
Is this correct? (y/N)

Premiamo Y:

GnuPG needs to construct a user ID to identify your key.

Real name:  
Email address:  
Comment:

Inseriamo il nostro nome e cognome e quindi l'indirizzo email. Come commento lasciamo il campo vuoto.

You selected this USER-ID:  
"Mario Rossi <mario.rossi@example.com>"

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

Accettare con O:

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

A seguire GnuPG chiederà di impostare la passphrase.  
Impostiamo la passphrase.

gpg: key 4B6E6777 marked as ultimately trusted  
gpg: directory '/home/mariorossi/.gnupg/openpgp-revocs.d' created  
gpg: revocation certificate stored as '/home/mariorossi/.gnupg/openpgp-revocs.d/491532759708014725CFE3A79F676B5A4B6E6777.rev'  
public and secret key created and signed.

gpg: checking the trustdb  
gpg: marginals needed: 3 completes needed: 1 trust model: PGP

```
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2021-03-31
pub  rsa4096/4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
     Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
uid      [ultimate] Mario Rossi <mario.rossi@example.com>
```

GnuPG mostra la chiave appena generata ed esce.

Come si vede, esiste solo la chiave principale con la sola capacità di certificazione [C]. Bisogna ora generare le sottochiavi con:

```
gpg --expert --edit-key 0x4B6E6777
```

```
gpg (GnuPG) 2.1.11; Copyright (C) 2016 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Secret key is available.

```
sec  rsa4096/4B6E6777
     created: 2018-04-01 expires: 2021-03-31 usage: C
     trust: ultimate   validity: ultimate
[ultimate] (1). Mario Rossi <mario.rossi@example.com>
```

gpg>

Dare il comando addkey:

Please select what kind of key you want:

- (3) DSA (sign only)
- (4) RSA (sign only)
- (5) Elgamal (encrypt only)
- (6) RSA (encrypt only)
- (7) DSA (set your own capabilities)
- (8) RSA (set your own capabilities)
- (10) ECC (sign only)
- (11) ECC (set your own capabilities)
- (12) ECC (encrypt only)
- (13) Existing key

Your selection?

Scegliere 8:

Possible actions for a RSA key: Sign Encrypt Authenticate

Current allowed actions: Sign Encrypt

- (S) Toggle the sign capability
- (E) Toggle the encrypt capability
- (A) Toggle the authenticate capability
- (Q) Finished

Your selection?

Togliere la capacità di cifratura premendo E:

Possible actions for a RSA key: Sign Encrypt Authenticate

Current allowed actions: Sign

- (S) Toggle the sign capability
- (E) Toggle the encrypt capability
- (A) Toggle the authenticate capability
- (Q) Finished

Your selection?

Scegliere Q:

RSA keys may be between 1024 and 4096 bits long.  
What keysize do you want? (2048)

Premere Invio per accettare 2048:

Requested keysize is 2048 bits  
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)

Inserire 1y per farla scadere dopo 1 anno:

Key expires at lun 01 apr 2019 19:57:16 CEST  
Is this correct? (y/N)

Accettare inserendo Y:

Really create? (y/N)

Accettare inserendo Y.

Inserire quindi la passphrase e la sottochiave sarà aggiunta al portachiavi:

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

Attendiamo che finisca, quindi apparirà:

```
sec  rsa4096/4B6E6777
    created: 2018-04-01 expires: 2021-03-31 usage: C
    trust: ultimate    validity: ultimate
ssb  rsa2048/9114F367
    created: 2018-04-01 expires: 2019-04-01 usage: S
[ultimate] (1). Mario Rossi <mario.rossi@example.com>
```

gpg>

Aggiungiamo ora con la stessa procedura (partendo da addkey) la chiave di cifratura e poi quella di autenticazione. Alla fine, dopo aver aggiunto la sottochiave di autenticazione, dare il comando:

save

per salvare e uscire.

Alla fine avremo questo portachiavi:

gpg --list-secret-keys



```
/home/mariorossi/.gnupg/pubring.kbx
-----
sec  rsa4096/4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
uid          [ultimate] Mario Rossi <mario.rossi@example.com>
ssb  rsa2048/9114F367 2018-04-01 [S] [expires: 2019-04-01]
ssb  rsa2048/2F3AC08A 2018-04-01 [E] [expires: 2019-04-01]
ssb  rsa2048/AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
```

La situazione è, quindi, la seguente:

- **chiave principale**, adibita alla sola **certificazione e gestione dei portachiavi**:  
sec rsa4096/4B6E6777 2018-04-01 [C];
- **sottochiave di firma**:  
ssb rsa2048/9114F367 2018-04-01 [S];
- **sottochiave di cifratura**:  
ssb rsa2048/2F3AC08A 2018-04-01 [E] [expires: 2019-04-01];
- **sottochiave di autenticazione**:  
ssb rsa2048/AFBE0F10 2018-04-01 [A] [expires: 2019-04-01].

Possiamo aggiungere anche altri UserID o una foto con:

```
gpg --expert --edit-key 0x4B6E6777
e poi con adduid.
```

## 2.3 Il certificato di revoca

Con GnuPG versione 2 la generazione del certificato di revoca è stata fatta automaticamente quando le chiavi sono state generate e si trova nella directory /home/user/.gnupg/openpgp-revocs.d/. Il file ha lo stesso nome dell'impronta della chiave, nel nostro caso:

```
491532759708014725CFE3A79F676B5A4B6E6777.rev
```

Questo file andrà tolto da questa directory e conservato in un luogo sicuro. Lo vedremo nel [paragrafo dedicato al backup](#).

## 2.4 Impostare le preferenze della chiave

Modificare la chiave con:

```
gpg --expert --edit-key 0x4B6E6777
```

Dal prompt di GnuPG dare:

```
setpref S10 S9 S8 H10 H9 H8 Z2 Z3 Z1 Z0
```

Per il significato dei codici, vedi la Tabella .

Alla domanda di GnuPG seguente:

```
Set preference list to:
Cipher: TWOFISH, AES256, AES192, 3DES
Digest: SHA512, SHA384, SHA256, SHA1
Compression: ZLIB, BZIP2, ZIP, Uncompressed
Features: MDC, Keyserver no-modify
Really update the preferences? (y/N)
```

CODICE	DESCRIZIONE
S10	Cifrario Twofish
S9	Cifrario AES256
S8	Cifrario AES192
H10	Hash SHA512
H9	Hash SHA384
H8	Hash SHA256
Z2	Compressione ZLIB
Z3	Compressione BZIP2
Z1	Compressione ZIP
Z0	Non compresso

Tabella 1: I codici GnuPG per cifrari, hash e compressione.

premere Y e quindi:

save

## 2.5 Impostare il file gpg.conf

Aprire il file gpg.conf dentro la directory /home/user/.gnupg, eliminare tutto il contenuto e incollarvi questo testo, ricordandosi di **aggiornare le due ID di chiave** nelle righe default-key e encrypt-to con l'ID della vostra chiave:

```
# Per lo skeleton di questo file vedi:
# /usr/share/gnupg2/gpg-conf.skel

#-----
# default key
#-----

# The default key to sign with. If this option is not used, the default
  key is
# the first key found in the secret keyring

default-key 0x9F676B5A4B6E6777
encrypt-to 0x9F676B5A4B6E6777

#-----
# behavior
#-----

# Disable inclusion of the version string in ASCII armored output
no-emit-version

# Disable comment string in clear text signatures and ASCII armored
  messages
no-comments

# Display long key IDs
keyid-format 0xlong
```

```

# List all keys (or the specified ones) along with their fingerprints
with-fingerprint

# List all keys (or the specified ones) along with their keygrips
with-keygrip

# Display the calculated validity of user IDs during key listings
list-options show-uid-validity
verify-options show-uid-validity

# Try to use the GnuPG-Agent. With this option, GnuPG first tries to
    connect to
# the agent before it asks for a passphrase.
use-agent

#-----
# keyserver
#-----

# When using --refresh-keys, if the key in question has a preferred
    keyserver
# URL, then disable use of that preferred keyserver to refresh the key
    from
keyserver-options no-honor-keyserver-url

# When searching for a key with --search-keys, include keys that are
    marked on
# the keyserver as revoked
keyserver-options include-revoked

#-----
# algorithm and ciphers
#-----

# List of personal digest preferences. When multiple digests are
    supported by
# all recipients, choose the strongest one
personal-cipher-preferences TWOFISH AES256 AES192 3DES

# List of personal digest preferences. When multiple ciphers are
    supported by
# all recipients, choose the strongest one
personal-digest-preferences SHA512 SHA384 SHA256 SHA224

# Message digest algorithm used when signing a key
cert-digest-algo SHA512

# List of personal compression preferences.
personal-compress-preferences ZLIB BZIP2 ZIP Uncompressed

# This preference list is used for new keys and becomes the default for
# "setpref" in the edit menu
default-preference-list SHA512 SHA384 SHA256 SHA224 TWOFISH AES256 AES192
    3DES ZLIB BZIP2 ZIP Uncompressed

```

## 2.6 Impostare il keyserver

Modificare (o creare) il file `/home/user/.gnupg/dirmngr.conf`:

```
# Per lo skeleton di questo file vedi:
# /usr/share/gnupg2/dirmngr-conf.skel

# This is the server that --recv-keys, --send-keys, and --search-keys
# will
# communicate with to receive keys from, send keys to, and search for
# keys on
keyserver hkps://hkps.pool.sks-keyservers.net
```

## 2.7 Copiare la directory `.gnupg` nel proprio computer di lavoro

A questo punto, se avete usato una distribuzione *live* per la generazione delle chiavi, copiate tutta la directory `.gnupg` su una chiavetta USB e quindi trasferitela sul disco fisso del computer di lavoro.

Dopo il trasferimento non dimenticate di distruggere la directory `.gnupg` sulla chiavetta. Si possono usare diversi strumenti come ad esempio `shred`.

## 2.8 Esportare la chiave pubblica sul keyserver

```
gpg --send-keys 0x9F676B5A4B6E6777
```

## 2.9 Segnarsi la data di scadenza delle proprie chiavi nel calendario

Segnatevi nel vostro calendario quando scadranno le chiavi e fatevi mandare un avviso almeno un mese prima. Una volta rinnovate le chiavi, inviatele al keyserver. Ricordate che le chiavi scadute possono essere ancora rinnovate, ma non è buona norma farle scadere.

# 3 Fare il backup

Dopo aver generato il nostro mazzo di chiavi **è fondamentale farne il backup**. Davvero, non si prenda alla leggera questo passaggio e **non proseguite oltre se non fate il backup**.

Faremo il backup di questi tre elementi:

- directory di GnuPG;
- chiavi;
- certificato di revoca.

Creiamo una directory backup e relative sottodirectory sulla scrivania del PC dove metteremo tutti i file del backup:

```
mkdir -p /home/user/Scrivania/backup/directory_gnupg
mkdir /home/user/Scrivania/backup/chiavi
mkdir /home/user/Scrivania/backup/certificato_di_revoca
```

Cambiate `Scrivania` con `Desktop` se è il vostro caso.

### 3.1 Backup della directory di GnuPG

Anzitutto facciamo il backup di tutta la directory di GnuPG così come si trova in questo momento. Qualora qualcosa dovesse andare storto, potremmo sempre ripristinarla e cominciare daccapo.

```
cp -r /home/user/.gnupg/ /home/user/Scrivania/backup/directory_gnupg/
```

Verificate che nella directory di backup ci sia la sotto-directory `private-keys-v1.d` con all'interno i file privati delle chiavi. Ce ne devono essere uno per ogni chiave privata, vale a dire uno per la chiave master e uno per ogni sottochiave.

### 3.2 Backup delle chiavi

Vediamo anzitutto la situazione delle nostre chiavi private:

```
gpg --list-secret-keys
```

che restituirà qualcosa del tipo:

```
sec  rsa4096/0x9F676B5A4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
     Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
     Keygrip = C6D1CD1D12CBBC8FA14D30004EAF381803A72597
uid      [ultimate] Mario Rossi <mario.rossi@example.com>
ssb  rsa2048/0xE7E0CAF69114F367 2018-04-01 [S] [expires: 2019-04-01]
     Keygrip = 135159EDEFB9FCB6062F9DE0E21FD90CB07DA041
ssb  rsa2048/0x3C91B3682F3AC08A 2018-04-01 [E] [expires: 2019-04-01]
     Keygrip = C74D47329D5D224B0716879DCF3A4EC2D8951C53
ssb  rsa2048/0x465ED456AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
     Keygrip = 30DEE8A060F3562CD6F1384E0F883D65477E596A
```

Procediamo ad effettuare un backup di tutta la chiave (principale e sottochiavi) e anche delle singole chiavi. Questo è quello che otterremo:

1. backup chiave completa;
2. backup chiave principale;
3. backup di tutte le sottochiavi;
4. backup sottochiave di firma;
5. backup sottochiave di cifratura;
6. backup sottochiave di autenticazione;
7. backup chiave pubblica;
8. backup chiave SSH.

Qui di sotto gli otto comandi di cui sopra, eseguiti uno dopo l'altro. Notate come in alcuni casi l'opzione è `-export-secret-keys` oppure `export-secret-key` senza `s` e con un punto esclamativo `!` dopo l'ID della chiave.

```

gpg --armor --export-secret-keys 0x9F676B5A4B6E6777 > /home/user/
  Scrivania/backup/chiavi/1
  _principale_con_sottochiavi_0x9F676B5A4B6E6777.asc
gpg --armor --export-secret-key 0x9F676B5A4B6E6777! > /home/user/
  Scrivania/backup/chiavi/2_principale_soltanto_0x9F676B5A4B6E6777.asc
gpg --armor --export-secret-subkeys 0x9F676B5A4B6E6777 > /home/user/
  Scrivania/backup/chiavi/3_sottochiavi_0x9F676B5A4B6E6777.asc
gpg --armor --export-secret-subkey 0xE7E0CAF69114F367! > /home/user/
  Scrivania/backup/chiavi/4_sottochiave_firma_0xE7E0CAF69114F367.asc
gpg --armor --export-secret-subkey 0x3C91B3682F3AC08A! > /home/user/
  Scrivania/backup/chiavi/5_sottochiave_cifatura_0x3C91B3682F3AC08A.
  asc
gpg --armor --export-secret-subkey 0x465ED456AFBE0F10! > /home/user/
  Scrivania/backup/chiavi/6
  _sottochiave_autenticazione_0x465ED456AFBE0F10.asc
gpg --armor --export 0x9F676B5A4B6E6777 > /home/user/Scrivania/backup/
  chiavi/7_chiave_pubblica_0x9F676B5A4B6E6777.asc
gpg --armor --export-ssh-key 0x9F676B5A4B6E6777 > /home/user/Scrivania/
  backup/chiavi/8_chiave_pubblica_ssh_0x9F676B5A4B6E6777.asc

```

### 3.3 Backup del certificato di revoca

Come abbiamo detto al paragrafo *Il certificato di revoca*, questo è stato già creato da GnuPG versione 2 nella directory `/home/user/.gnupg/openpgp-revocs.d/` ed il file ha lo stesso nome dell'impronta della chiave, nel nostro caso `491532759708014725CFE3A79F676B5A4B6E6777.rev`.

Spostiamolo dalla directory `/home/user/.gnupg` alla directory di backup:

```

mv /home/user/.gnupg/openpgp-revocs.d/491532759708014725
  CFE3A79F676B5A4B6E6777.rev /home/user/Scrivania/backup/
  certificato_di_revoca/

```

Eventualmente è anche possibile stampare su carta il certificato e conservarlo in cassaforte.

### 3.4 Spostare la directory di backup in un luogo sicuro

Finito il backup, spostare la directory `/home/user/Scrivania/backup` in almeno due supporti da conservare in due luoghi distinti e sicuri.

A seconda di quanta importanza hanno queste chiavi per voi, posso consigliare una situazione del genere:

- copia della cartella su una chiavetta USB da usare **solo** per questo scopo, conservata in un luogo sicuro e nascosto, ad esempio la vostra cassaforte in casa;
- copia di sicurezza su una seconda chiavetta USB da usare **solo** per questo scopo, conservata in un altro luogo sicuro e nascosto, ad esempio la vostra cassetta di sicurezza in banca.

Chiaramente non complicatevi la vita, ma prendete le dovute precauzioni per proteggere le chiavi. Sappiate però che, se perdetes la YubiKey (se la usate per le vostre sottochiavi) e se perdetes i vostri backup, non potrete più accedere ai vostri file cifrati.

## 4 Trasferire le chiavi OpenPGP su YubiKey

Se avete un token USB come la YubiKey, potete trasferire le sottochiavi in essa. L'operazione è molto semplice.

Prima di andare avanti, quando dico "trasferire" intendo proprio spostare una cosa da un posto a un altro. Nel nostro caso, intendo spostare le sottochiavi dal disco fisso alla YubiKey.

Ricordo che trasferire le sottochiavi nel token USB è una **operazione a senso unico e non è possibile tornare indietro**, cioè non si possono trasferire le sottochiavi dal token al nostro disco fisso. Solo se abbiamo un backup delle chiavi possiamo ripristinare la situazione allo stato prima del trasferimento. Per cui **non procedete se non avete fatto il backup**.

Diamo uno sguardo alla situazione attuale:

```
gpg --list-secret-keys
```

```
sec  rsa4096/0x9F676B5A4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
      Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
      Keygrip = C6D1CD1D12CBBC8FA14D30004EAF381803A72597
uid   [ultimate] Mario Rossi <mario.rossi@example.com>
ssb   rsa2048/0xE7E0CAF69114F367 2018-04-01 [S] [expires: 2019-04-01]
      Keygrip = 135159EDEFB9FCB6062F9DE0E21FD90CB07DA041
ssb   rsa2048/0x3C91B3682F3AC08A 2018-04-01 [E] [expires: 2019-04-01]
      Keygrip = C74D47329D5D224B0716879DCF3A4EC2D8951C53
ssb   rsa2048/0x465ED456AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
      Keygrip = 30DEE8A060F3562CD6F1384E0F883D65477E596A
```

Inseriamo il token nella porta USB e controlliamo che venga vista da GnuPG:

```
gpg --card-status
```

Se restituisce:

```
gpg: error getting version from 'scdaemon': No SmartCard daemon
gpg: OpenPGP card not available: No SmartCard daemon
```

installiamo scdaemon:

```
sudo apt-get install pcscd pcsc-tools scdaemon
```

e quindi di nuovo:

```
gpg --card-status
```

che restituirà qualcosa del tipo:

```
Reader .....: Yubico Yubikey 4 OTP U2F CCID 00 00
Application ID ...: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Version .....: 2.1
Manufacturer ....: Yubico
Serial number ....: XXXXXXXX
Name of cardholder: [not set]
Language prefs ...: [not set]
Sex .....: unspecified
URL of public key : [not set]
Login data .....: [not set]
Signature PIN ....: not forced
Key attributes ...: rsa2048 rsa2048 rsa2048
```

```
Max. PIN lengths .: 127 127 127
PIN retry counter : 3 0 3
Signature counter : 0
Signature key ....: [none]
Encryption key....: [none]
Authentication key: [none]
General key info...: [none]
```

Notate come ci siano i tre campi dedicati alla conservazione delle tre sottochiavi che abbiamo creato:

```
Signature key ....: [none]
Encryption key....: [none]
Authentication key: [none]
```

Qui sposteremo le nostre tre sottochiavi. Iniziamo la procedura.  
Entriamo in modalità modifica:

```
gpg --card-edit
```

Cambiamo anzitutto i PIN. I PIN predefiniti di fabbrica sono:

- Admin PIN (8 cifre): 12345678
- PIN (6 cifre): 123456

Entriamo in modalità amministratore dando:

```
admin
```

```
passwd
```

Dal menu scegliere l'opzione da seguire, vale a dire `Change PIN` e, una volta finito, `Change Admin PIN`. Quindi alla fine salvare con `save` e uscire con `quit`.

Una volta aggiornati i due PIN, passare al trasferimento delle chiavi. Dal prompt del terminale dare:

```
gpg --edit-key 0x9F676B5A4B6E6777
```

che restituirà:

```
gpg (GnuPG) 2.1.11; Copyright (C) 2016 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

```
Secret key is available.
```

```
sec rsa4096/0x9F676B5A4B6E6777
   created: 2018-04-01 expires: 2021-03-31 usage: C
   trust: ultimate   validity: ultimate
ssb rsa2048/0xE7E0CAF69114F367
   created: 2018-04-01 expires: 2019-04-01 usage: S
ssb rsa2048/0x3C91B3682F3AC08A
   created: 2018-04-01 expires: 2019-04-01 usage: E
ssb rsa2048/0x465ED456AFBE0F10
   created: 2018-04-01 expires: 2019-04-01 usage: A
[ultimate] (1). Mario Rossi <mario.rossi@example.com>
```

Selezioniamo la prima sottochiave con:



key 1

che mostrerà la chiave selezionata con un asterisco \*:

```
sec  rsa4096/0x9F676B5A4B6E6777
    created: 2018-04-01 expires: 2021-03-31 usage: C
    trust: ultimate    validity: ultimate
ssb*  rsa2048/0xE7E0CAF69114F367
    created: 2018-04-01 expires: 2019-04-01 usage: S
ssb  rsa2048/0x3C91B3682F3AC08A
    created: 2018-04-01 expires: 2019-04-01 usage: E
ssb  rsa2048/0x465ED456AFBE0F10
    created: 2018-04-01 expires: 2019-04-01 usage: A
[ultimate] (1). Mario Rossi <mario.rossi@example.com>
```

Trasferiamo la chiave con:

keytocard

Deselezioniamo quindi la chiave 1:

key 1

L'asterisco verrà tolto. Quindi procedere con lo stesso metodo per le restanti due chiavi. Qui indico di seguito i comandi in sequenza:

```
key 2
keytocard
key 2
key 3
keytocard
key 3
```

Alla fine salviamo e usciamo:

save

Una volta usciti al prompt dei comandi, diamo uno sguardo alla situazione:

```
gpg --edit-key 0x9F676B5A4B6E6777
```

che restituirà:

```
gpg (GnuPG) 2.1.11; Copyright (C) 2016 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Secret key is available.

```
sec  rsa4096/0x9F676B5A4B6E6777
    created: 2018-04-01 expires: 2021-03-31 usage: C
    trust: ultimate    validity: ultimate
ssb>  rsa2048/0xE7E0CAF69114F367
    created: 2018-04-01 expires: 2019-04-01 usage: S
ssb>  rsa2048/0x3C91B3682F3AC08A
    created: 2018-04-01 expires: 2019-04-01 usage: E
ssb>  rsa2048/0x465ED456AFBE0F10
    created: 2018-04-01 expires: 2019-04-01 usage: A
[ultimate] (1). Mario Rossi <mario.rossi@example.com>
```

Come si vede abbiamo il segno > dopo ogni ssb, segno che le sottochiavi sono nel token. La chiave principale invece è ancora nel nostro disco. Date anche uno sguardo a com'è cambiata la YubiKey dando:

```
gpg --card-status
```

Potremmo finire qua se abbiamo scelto di non cancellare la chiave principale dal disco.

## 5 Eliminare la chiave principale

Se abbiamo scelto di eliminare la chiave principale dal disco, dobbiamo anzitutto verificare che versione di GnuPG stiamo usando, perché a seconda della versione bisognerà procedere in due modi diversi:

- **se abbiamo GnuPG versione 2.1 o successiva:** è sufficiente cancellare il file corrispondente alla chiave master dalla directory `/home/user/.gnupg/private-keys-v1.d;`
- **se abbiamo GnuPG precedente alla versione 2.1:** dopo aver fatto il backup delle sottochiavi, bisogna dapprima cancellare tutta la chiave (master e sottochiavi) e poi reimportare solo le sottochiavi.

Nonostante all'inizio della guida ho specificato di usare solo la versione di GnuPG 2.1 o successiva, vediamo comunque i due procedimenti nel dettaglio.

### 5.1 GnuPG versione 2.1 o successiva

GnuPG, dalla versione 2.1 in su, deposita le chiavi private nella directory `/home/user/.gnupg/private-keys-v1.d/` e non più nel file `/home/user/.gnupg/secring.gpg`. Perciò, se avete una chiave master e 3 sottochiavi, dovreste avere 4 file con estensione `.key` in quella directory:

```
C6D1CD1D12CBBC8FA14D30004EAF381803A72597.key  
135159EDEFB9FCB6062F9DE0E21FD90CB07DA041.key  
C74D47329D5D224B0716879DCF3A4EC2D8951C53.key  
30DEE8A060F3562CD6F1384E0F883D65477E596A.key
```

Per identificare quale di questi file è la chiave master, dare nel terminale questo comando:

```
gpg --with-keygrip --list-key 0x9F676B5A4B6E6777
```

Siccome è stato fatto il [backup della directory di GnuPG](#), potete eliminare il file `.key` corrispondente alla chiave master. Per ogni evenienza, prima di cancellare il file, verificate che nel backup ci sia il file che state per cancellare. Il comando per eliminare il file è:

```
cp rm /home/user/.gnupg/private-keys-v1.d/  
C6D1CD1D12CBBC8FA14D30004EAF381803A72597.key
```

oppure usate un programma come `shred` per eliminarlo in modo sicuro:

```
shred -vuzn25 /home/user/.gnupg/private-keys-v1.d/  
C6D1CD1D12CBBC8FA14D30004EAF381803A72597.key
```

OPZIONE	DESCRIZIONE
v	Show progress
u	Truncate and remove file after overwriting
z	Add a final overwrite with zeros to hide shredding
n	Overwrite N times instead of the default (3)

Tabella 2: Le opzioni di shred usate in questo *how-to*.

Per le opzioni di shred vedi la Tabella .

Se avete fatto da poco la migrazione a GnuPG versione 2.1 o superiore, il vecchio file `secring.gpg` che conteneva le chiavi private potrebbe ancora averla al suo interno. Per cui, se la dimensione del file è zero, il file non contiene più nulla; altrimenti, se la dimensione del file è maggiore di zero, cancellatelo del tutto.

Quando in futuro vi servirà usare la chiave master (ad esempio, per firmare una chiave altrui), potrete semplicemente prendere dal backup il file `.key` della chiave master e copiarlo nella directory `private-keys-v1.d` di lavoro. Una volta terminata l'esigenza, potrete eliminare nuovamente il file.

## 5.2 GnuPG precedente alla versione 2.1

Se avete questa versione di GnuPG, bisognerà procedere con questi passi:

1. Essere sicuri di avere il [backup funzionante delle chiavi](#);
2. Rimuovere la chiave privata completa (cioè master e sottochiavi);
3. Reimportare dal backup solo le sottochiavi.

La chiave completa si elimina con:

```
gpg --delete-secret-keys 0x9F676B5A4B6E6777
```

Le sottochiavi si importano con:

```
gpg --import 0xE7E0CAF69114F367
gpg --import 0x3C91B3682F3AC08A
gpg --import 0x465ED456AFBE0F10
```

Vediamo quindi la situazione:

```
gpg --list-secret-keys
```

Ora, se avete spostato le sottochiavi nella **YubiKey**, la risposta sarà:

```
sec# rsa4096/0x9F676B5A4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
      Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
      Keygrip = C6D1CD1D12CBBC8FA14D30004EAF381803A72597
uid          [ultimate] Mario Rossi <mario.rossi@example.com>
ssb> rsa2048/0xE7E0CAF69114F367 2018-04-01 [S] [expires: 2019-04-01]
      Keygrip = 135159EDEFB9FCB6062F9DE0E21FD90CB07DA041
ssb> rsa2048/0x3C91B3682F3AC08A 2018-04-01 [E] [expires: 2019-04-01]
      Keygrip = C74D47329D5D224B0716879DCF3A4EC2D8951C53
ssb> rsa2048/0x465ED456AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
      Keygrip = 30DEE8A060F3562CD6F1384E0F883D65477E596A
```

mentre, se avete deciso di tenere le sottochiavi nel disco, la risposta sarà:

```

sec# rsa4096/0x9F676B5A4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
    Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
    Keygrip = C6D1CD1D12CBBC8FA14D30004EAF381803A72597
uid      [ultimate] Mario Rossi <mario.rossi@example.com>
ssb  rsa2048/0xE7E0CAF69114F367 2018-04-01 [S] [expires: 2019-04-01]
    Keygrip = 135159EDEFB9FCB6062F9DE0E21FD90CB07DA041
ssb  rsa2048/0x3C91B3682F3AC08A 2018-04-01 [E] [expires: 2019-04-01]
    Keygrip = C74D47329D5D224B0716879DCF3A4EC2D8951C53
ssb  rsa2048/0x465ED456AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
    Keygrip = 30DEE8A060F3562CD6F1384E0F883D65477E596A

```

## 6 Gli agenti

Fermiamo ssh-agent:

```
killall ssh-agent
```

Disabilitiamo ssh-agent permanentemente:

```
sudo nano /etc/X11/Xsession.options
```

e commentiamo la riga aggiungendo un cancelletto # all'inizio:

```
# use-ssh-agent
```

Abilitiamo il supporto SSH in GnuPG aggiungendo la seguente riga in /home/user/.gnupg/gpg-agent.conf:

```
enable-ssh-support
```

Già che ci siamo aggiungiamo pure la scelta di quale pinentry usare:

```
pinentry-program /usr/bin/pinentry-gtk-2
```

Alternative sono, ad esempio:

```

pinentry-program /usr/bin/pinentry-curses
pinentry-program /usr/bin/pinentry-tty
pinentry-program /usr/bin/pinentry-qt
pinentry-program /usr/bin/pinentry-gnome3

```

Riavviamo l'agente di GnuPG:

```

gpg-connect-agent killagent /bye
gpg-connect-agent /bye

```

Inseriamo la riga seguente in /home/user/.bashrc:

```

# You should always add the following lines to your .bashrc or whatever
# initialization file is used for all shell invocations.
# @link https://gnupg.org/documentation/manuals/gnupg/Invoking-
#       GPG_002dAGENT.html#Invoking-GPG_002dAGENT
GPG_TTY=$(tty)
export GPG_TTY

# Launch gpg-agent
gpg-connect-agent /bye

# When using SSH support, use the current TTY for passphrase prompts

```

```

gpg-connect-agent updatestartuptty /bye > /dev/null

# Point the SSH_AUTH_SOCK to the one handled by gpg-agent
if [ -S $(gpgconf --list-dirs agent-ssh-socket) ]; then
    export SSH_AUTH_SOCK=$(gpgconf --list-dirs agent-ssh-socket)
else
    echo "$(gpgconf --list-dirs agent-ssh-socket) doesn't exist. Is gpg-
        agent running ?"
fi

```

Facciamo rileggere questo file:

```
source .bashrc
```

Infine, se non l'abbiamo già fatto, assicuriamoci di usare gpg2 invece della versione 1:

```
echo "alias gpg=gpg2" >> /home/user/.bash_aliases
```

## 7 Importare la chiave principale per operazioni speciali

Quando periodicamente dovete fare operazioni occasionali sul vostro portachiavi o se dovete firmare (certificare) una chiave altrui, vi servirà la chiave principale. Per fare queste operazioni dovete importare la chiave principale nel vostro portachiavi, fare l'operazione e quindi cancellarla nuovamente.

C'è un altro metodo per fare uso della chiave principale, come ad esempio dire momentaneamente a GnuPG che la sua *home* non è `/home/user/.gnupg` ma un'altra directory. Io non ho mai usato questo sistema, per cui scrivo solo quello che uso io.

Accertatevi di avere accesso dal terminale alla directory dove c'è il vostro backup. Lo scopo è quello di copiare il file `.key` della chiave master dal backup alla directory attuale di lavoro di GnuPG.

```

cp directory-backup-gnupg/private-keys-v1.d/
    C6D1CD1D12CBBC8FA14D30004EAF381803A72597.key /home/user/.gnupg/
    private-keys-v1.d/C6D1CD1D12CBBC8FA14D30004EAF381803A72597.key

```

Effettuare quindi le operazioni che dovevamo compiere e, alla fine, eliminarla nuovamente:

```

cp rm /home/user/.gnupg/private-keys-v1.d/
    C6D1CD1D12CBBC8FA14D30004EAF381803A72597.key

```

Accertatevi ora che la chiave master non ci sia più con:

```
gpg --list-secret-keys
```

che dovrebbe restituire qualcosa di simile a questo:

```

sec# rsa4096/0x9F676B5A4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
    Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
    Keygrip = C6D1CD1D12CBBC8FA14D30004EAF381803A72597
uid          [ultimate] Mario Rossi <mario.rossi@example.com>
ssb> rsa2048/0xE7E0CAF69114F367 2018-04-01 [S] [expires: 2019-04-01]
    Keygrip = 135159EDEFB9FCB6062F9DE0E21FD90CB07DA041
ssb> rsa2048/0x3C91B3682F3AC08A 2018-04-01 [E] [expires: 2019-04-01]

```

```

Keygrip = C74D47329D5D224B0716879DCF3A4EC2D8951C53
ssb> rsa2048/0x465ED456AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
Keygrip = 30DEE8A060F3562CD6F1384E0F883D65477E596A

```

o a questo, se avete deciso di tenere le sottochiavi nel disco:

```

sec# rsa4096/0x9F676B5A4B6E6777 2018-04-01 [C] [expires: 2021-03-31]
Key fingerprint = 4915 3275 9708 0147 25CF E3A7 9F67 6B5A 4B6E 6777
Keygrip = C6D1CD1D12CBBC8FA14D30004EAF381803A72597
uid [ultimate] Mario Rossi <mario.rossi@example.com>
ssb rsa2048/0xE7E0CAF69114F367 2018-04-01 [S] [expires: 2019-04-01]
Keygrip = 135159EDEFB9FCB6062F9DE0E21FD90CB07DA041
ssb rsa2048/0x3C91B3682F3AC08A 2018-04-01 [E] [expires: 2019-04-01]
Keygrip = C74D47329D5D224B0716879DCF3A4EC2D8951C53
ssb rsa2048/0x465ED456AFBE0F10 2018-04-01 [A] [expires: 2019-04-01]
Keygrip = 30DEE8A060F3562CD6F1384E0F883D65477E596A

```

C'è il simbolo # dopo la parola sec, per cui la chiave master non c'è più.

## 8 Estendere la validità delle nostre chiavi

Se abbiamo scelto una validità a tempo per le nostre chiavi, periodicamente potremo estenderne la validità se ancora sono valide per noi. Teniamo presente che possiamo rinnovare la validità di una chiave già scaduta.

Se abbiamo scelto di togliere la chiave principale dal nostro disco, dovremo importarla nuovamente, come descritto al paragrafo [Importare la chiave principale per operazioni speciali](#).

Se abbiamo scelto, poi, di tenere le sottochiavi nel token USB, dovremo inserirlo in una porta USB.

Iniziamo quindi la procedura di rinnovo:

```

gpg --edit-key 0x9F676B5A4B6E6777

sec rsa4096/0x9F676B5A4B6E6777
  created: 2018-04-01 expires: 2021-03-31 usage: C
  trust: ultimate validity: ultimate
ssb rsa2048/0xE7E0CAF69114F367
  created: 2018-04-01 expires: 2019-04-01 usage: S
ssb rsa2048/0x3C91B3682F3AC08A
  created: 2018-04-01 expires: 2019-04-01 usage: E
ssb rsa2048/0x465ED456AFBE0F10
  created: 2018-04-01 expires: 2019-04-01 usage: A
[ultimate] (1). Mario Rossi <mario.rossi@example.com>

```

I passi di rinnovo per ogni chiave sono i seguenti:

1. selezione della sottochiave da rinnovare;
2. invio del comando EXPIRE;
3. scelta della durata della validità;
4. deselezione della sottochiave.

Nel nostro caso, essendo tre le chiavi, questi i passaggi tutti di seguito:

```
key 1 (per la prima sottochiave; key 2 per seconda; key 3 per terza)
  expire (e seguire le istruzioni)
    1y (per un anno di validita)
key 1 (deseleziona)
key 2 (seleziona la seconda sottochiave)
  expire
    1y
key 2 (deseleziona)
key 3 (seleziona la terza sottochiave)
  expire
    1y
check (verifichiamo che tutto sia stato eseguito)
save
```

## 9 Bibliografia

- DEBIAN WIKI, [Subkeys](#).
- RISEUP, [OpenPGP Best Practices](#).
- TJL73, [Creazione di chiavi sicure \(ossia, come generare con GnuPG una coppia di chiavi per poterla conservare in modo "abbastanza" sicuro](#).
- ERIC SEVERANCE, [PGP and SSH keys on a Yubikey NEO](#).
- ALEX CABAL, [Creating the perfect GPG keypair](#).
- SUVA.SH, [Using GPG and SSH keys \(GnuPG 2.1\) with a Smartcard \(Yubikey 4\)](#).
- DRDUH, [Guide to using YubiKey as a SmartCard for GPG and SSH](#).