Generazione sicura di chiavi OpenPGP

Aldo Latino

1 aprile 2017 – 13 ottobre 2019

Sommario

Una guida dettagliata, per quanto possibile, sulla generazione, conservazione e protezione delle chiavi OpenPGP.

Indice

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 manutenzionare 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.

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 `/.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/4915327597
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 del 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