Keyoxide: *Bob, sei tu?*

Antefatto

Keyoxide è una piattaforma opensource per la verifica delle identità digitali. No, non è la CIE, non è SPID, EIDAS o quel che vi pare: non è niente di tutto ciò. Non serve a dimsotrare che una persona è quella che è riportata sul documento di identità tipo la patente, ma certifica che chi si attribuisce un qualcosa (un repository Github, una identità su Lemmy, etc) sia effettivamente il detentore della chiave privata con cui è stata creata la propria firma digitale.

Detta in termini più classici:

Keyoxide è una piattaforma open source per la validazione dell'identità digitale decentralizzata. A differenza dei sistemi di identità federata (come SPID o CIE) o dei protocolli di autenticazione centralizzati (OAuth), Keyoxide non agisce come autorità di certificazione anagrafica, bensì come aggregatore di prove crittografiche.

Il core del software si basa sulla verifica della proprietà (ownership) piuttosto che sull'identità civile. Utilizzando standard crittografici come OpenPGP o il protocollo ASPE, Keyoxide permette a un utente di collegare univocamente diverse proprietà digitali (repository di codice, account social, domini web) a una singola radice crittografica.

"... *Ma poi alla fine.. un mescolio! Chi guardava.. il rospo guardava quel.. il tacchino guardava la cagna.. dico: “Sta’ fermo, sennò cosa nasce, la.. la tac-cagna?! Sta’ bono..!”. Chi si mischiava di qua.. poi, bastava distrarsi, trombavano tutto! Trombavano tutti! Chi trombava di qua, chi di là..! La talpa poverina era cieca, non ci vedeva, l’hanno trombata tutti! Ma tutti proprio! Ma devi vedè che gli hanno fatto! Lei ogni tanto se n’accorgeva che qualcosa non andava, era troppo! Allora lo domandava: “Scusa ma sei il talpo te?” – “Sì sì.. e sono il talpo!”.. e se.. ma proprio dei maiali!* ...."

Questo è tratto dallo spettacolo di "Tutto Benigni 95/96", ed è un chiaro esempio che probabilmente la sig.ra Talpa avrebbe potuto

beneficiare di Keyoxide per essere sicura che chi c'era dietro, fosse effettivamente il talpo.

Caso d'uso

Create un'identità su un social come Mastodon, e poi su Lemmy, poi su Github, poi vi rendete conto che la vostra vita sociale forse sarebbe meglio fosse più orientata alla vita fuori, ma tant'è. In ogni caso, adesso nella depressione più totale vi viene anche in mente che probabilmente la gente non sa che le tre identità sono legate tra di loro e sono sempre riconducibili a voi. Come fate per rassicurarli, come se la loro unica preoccupazione fosse sapere se l'utente *Ciottolino94* di Github è lo stesso *Ciottolino94* di cui ha visto un post su Mastodon? Il nome è abbastanza vergognoso, quindi credo ci siano poche persone in grado di avere fegato sufficiente da scegliere proprio questo nome, ma se volete esserne proprio sicuri che cosa fate?

- Create una chiave di crittografica con gpg (si può usare anche il protocollo ASPE, ma ora copriremo solo la versione pgp che è INCOMPATIBILE con aspe)
- Ci associate una mail
- Sui vari servizi che andate a utilizzare inserirete una Ariadne Proof, che serve a far capire a KeyOxide che chi ha creato quella pagina è la stessa persona che detiene la firma
- Chi vuole controllare, controlla su Keyoxide.

E vai di guida: passo 1, GPG

Installate GPG, anche se sono speranzoso sul fatto che lo abbiate già. Dopodiché create una chiave di firma possibilmente ellittica.

Negli esempi sottostanti io sto creando una chiave per *admin@example.org*.

```
gpg --expert --full-generate-key
gpg (GnuPG) 2.4.9; Copyright (C) 2025 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Per favore scegli che tipo di chiave vuoi:
```

```
   (1) RSA e RSA
   (2) DSA ed Elgamal
   (3) DSA (solo segno)
   (4) RSA (solo segno)
   (7) DSA (impostare le proprie funzionalità)
   (8) RSA (impostare le proprie funzionalità)
   (9) ECC (firmare e crittografare) *predefinito*
  (10) DSA (firma solo)
  (11) ECC (impostare le proprie funzionalità)
  (13) Chiave esistente
  (14) Chiave esistente dalla carta
Cosa scegli? 9
Selezionare la curva ellittica desiderata:
   (1) Curve 25519 *predefinito*
   (2) Curve 448
   (3) NIST P-256
   (4) NIST P-384
   (5) NIST P-521
   (6) Brainpool P-256
   (7) Brainpool P-384
   (8) Brainpool P-512
   (9) secp256k1
Cosa scegli? 1
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) 0
La chiave non scade affatto
È corretto? (y/N) y

GnuPG deve costruire un ID utente per identificare la chiave.

Nome e Cognome: Permission Denied
Indirizzo di Email: admin@example.org
Commento:
Hai selezionato questo User Id:
    "Permission Denied <admin@example.org>"

Modifica (N)ome, (C)ommento, (E)mail oppure (O)kay/(Q)uit? O
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.
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.
gpg: certificato di revoca archiviato come '/home/errno0x0d/.gnupg/openpgp-
revocs.d/3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9.rev'
chiavi pubbliche e segrete create e firmate.

pub   ed25519 2026-04-06 [SC]
      3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9
uid                      Permission Denied <admin@example.org>
sub   cv25519 2026-04-06 [E]

```

Passo 2: modifica della chiave

Guardate bene i 40 caratteri del fingerprint `3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9`: vi serve più tardi

```
pub   ed25519 2026-04-06 [SC]
      3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9
uid                      Permission Denied <admin@example.org>
sub   cv25519 2026-04-06 [E]
```

Adesso: che cosa volete certificare? I claim disponibili sono qui (https://docs.keyoxide.org/service-providers/).

Prendiamo Lemmy (https://docs.keyoxide.org/service-providers/lemmy/).

Modifichiamo la chiave, scegliamo lo *uid 1* (solitamente c'è solo quello), aggiungiamo la *notation* per la Ariadne proof, e salviamo:

```
gpg --edit-key 3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9
gpg (GnuPG) 2.4.9; Copyright (C) 2025 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

È disponibile una chiave segreta.

gpg: controllo il trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: profondità: 0  valido:   9  firmato:   2  trust: 0-, 0q, 0n, 0m, 0f, 9u
gpg: profondità: 1  valido:   1  firmato:   0  trust: 1-, 0q, 0n, 0m, 0f, 0u
gpg: il prossimo controllo del trustdb sarà fatto il 2029-01-21
```

```
sec  ed25519/E035163A2EC53669
     creato: 2026-04-06  scadenza: mai         utilizzo: SC
     attendibilità: definitivo    validità: definitivo
ssb  cv25519/7249B111D145D261
     creato: 2026-04-06  scadenza: mai         utilizzo: E
[ultimo] (1). Permission Denied <admin@example.org>

gpg> uid 1

sec  ed25519/E035163A2EC53669
     creato: 2026-04-06  scadenza: mai         utilizzo: SC
     attendibilità: definitivo    validità: definitivo
ssb  cv25519/7249B111D145D261
     creato: 2026-04-06  scadenza: mai         utilizzo: E
[ultimo] (1)* Permission Denied <admin@example.org>

gpg> notation
Immettere la notazione: proof@ariadne.id=https://lemmy.example.org/u/errno0x0d
No notations on user ID "Permission Denied <admin@example.org>"
Adding notation: proof@ariadne.id=https://lemmy.example.org/u/errno0x0d

sec  ed25519/E035163A2EC53669
     creato: 2026-04-06  scadenza: mai         utilizzo: SC
     attendibilità: definitivo    validità: definitivo
ssb  cv25519/7249B111D145D261
     creato: 2026-04-06  scadenza: mai         utilizzo: E
[ultimo] (1)* Permission Denied <admin@example.org>

gpg> save
```

Passo 3: esportazione chiave e pubblicazione

Dobbiamo caricare la chiave pubblica su xxx per farla vedere a Keyoxide:

```
gpg --armor --export 3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9 > key.asc
```

Dopodiché andiamo su `https://keys.openpgp.org/upload/` e carichiamo la chiave. Ci verrà anche detto che per renderla visibile bisogna verificare la mail che ci verrà inviata (per dimostrare che la casella di posta esiste ed è nostra). Facciamolo.

Passo 4: aggiunta della prova Ariadne

Nel caso di Lemmy, è sufficiente andare sul profilo e inserire la prova nella bio la stringa (notare i 40 caratteri del fingerprint):

```
openpgp4fpr:3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9
```

Potete vederlo anche visitando il mio profilo qui (https://lemmy.example.org/u/errno0x0d).

Passo 5: la verifica

Andando su `https://keyoxide.org/3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9` (come si vede, è stato inserito il fingerprint della chiave), si possono vedere quali identità digitali sono associate all'utenza.

Passo 6: next times

Se in futuro vogliamo inviare aggiornamenti della chiave direttamente dal terminale senza passare dal browser, possiamo configurare il nostro `~/.gnupg/gpg.conf` aggiungendo `keyserver hkps://keys.openpgp.org` e usare: `gpg --send-keys 3AC3F8D6A11B6CED3330DB8BAE1C2D88ADD99AF9`.

Per ogni prova che aggiungeremo, i passi 2-6 devono essere ripetuti.

[Che il talpone sia con voi]

[EOF]

Fluxer Icon

Canale di discussione e supporto

Canale di discussione e supporto. Per confrontarti su configurazioni o altro entra su fluxer.dreadful.work (https://fluxer.dreadful.work/invite/dreadful).