GPG on YubiKey's and installing PASS (the standard UNIX password manager):

> **Caution -**
> As an additional security measure, this process may be undertaken on an offline (non network-connected) machine or single-use Virtual Machine (VM). After installing the pre-requisite packages and *only* the pre-requisite packages, disconnect it from the network and continue with the steps below.

What you will need: YubiKey 5 Series and Linux OS.

$ sudo apt update

Packages to install on client computer, not a server:
First let's install GPG Tools.
$ sudo apt install gnupg2

Now let's install PASSword manager.
$ sudo apt install pass

Lastly let's instrall YubiKey's tools.
$ sudo apt install scdaemon yubikey-manager libpam-yubico libpam-u2f libu2f-udev

Creating Ed25519 Keys:
$ mkdir /tmp/my_keys
$ cd /tmp/my_keys
$ openssl rand -base64 24 > passwd.txt
Display your secure password to use, later on:
$ cat passwd.txt
Copy this password to the clipboard, and use it when prompted…

$ gpg2 --expert --full-generate-key

gpg: keybox '/tmp/new-gpg/pubring.kbx' created
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)
(13) Existing key

(14) Existing key from card
Your selection? **11**

Now enter 11 – ECC (set your own capabilities).

Possible actions for a ECDSA/EdDSA key: Sign Certify Authenticate
Current allowed actions: Sign Certify

(S) Toggle the sign capability
(A) Toggle the authenticate capability
(Q) Finished

Your selection? **s**

Now enter s – This will remove sign capabilities.

Possible actions for a ECDSA/EdDSA key: Sign Certify Authenticate
Current allowed actions: Certify

(S) Toggle the sign capability
(A) Toggle the authenticate capability
(Q) Finished

Your selection? **q**

Now enter q – Finsh.

Please select which elliptic curve you want:
(1) Curve 25519
(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
Your selection? **1**

Now enter 1 – for Curve 25519.

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) **0**

Now enter 0 – for Never Expire.

Key does not expire at all
Is this correct? (y/N) **y**

Now enter y – for yes it's correct.

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

Real name: Robert S.

Now enter [YOUR name goes in here].
Email address: Robert@host.local

Now enter [YOUR email address or fake address]. This will not get emailed...it's for ID purposes.

Comment: My black NFC YubiKey.

Now enter a description of your Security Key or enter nothing here.

You selected this USER-ID:
"Robert S. <Robert@host.local>"

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

Now enter O – for Okay.

gpg: key E0B518BE7CA2EC83 marked as ultimately trusted
public and secret key created and signed.

pub   ed25519 2022-10-05 [C]
1FB5E22C91B86AC51C4DC3B3E0B518BE7CA2EC83
uid Robert S. <Robert@host.local>

Generating the Authentication Subkey

```
$gpg2 --expert --edit-key key-id
```

```
Replace key-id with the eight-character string output from the key
generation process. This will be found in the line beginning with
pub. In the example above, the ID is E0B518BE7CA2EC83
```

```
Secret key is available.


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
sec  ed25519/E0B518BE7CA2EC83
     created: 2022-10-05  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1). Robert S. <Robert@host.local>

gpg> addkey
```

Now enter addkey – for creating Encrypt sub key.

```
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
  (14) Existing key from card
Your selection? 12
```

Enter 12 – for ECC (Encrypt Only).

```
Please select which elliptic curve you want:
   (1) Curve 25519
   (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

Your selection? **1**

Enter 1 – for Curve 25519.

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) **3y**

Enter 3y – for 3 years to be valid.

Is this correct? **y**

Enter y – for it’s correct.

Really create? **y**

Enter y – to create it.

Enter your secure password, made in the 1st few steps.

gpg> **addkey**

Now enter addkey – for creating Encrypt sub key.

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
(14) Existing key from card
Your selection? **10**

Enter 10 – for ECC (Sign Only).

Please select which elliptic curve you want:
(1) Curve 25519
(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
Your selection? **1**

Enter 1 – for Curve 25519.

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) **3y**

Enter 3y – for 3 years to be valid.

Is this correct? **y**

Enter y – for it’s correct.

Really create? **y**

Enter y – to create it.

Enter your secure password, made in the 1st few steps.

gpg> **addkey**

Enter addkey in the gpg prompt.

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

(14) Existing key from card

Your selection? **11**

Enter 11 – for ECC own capabilities (Auth only).

Possible actions for a ECDSA/EdDSA key: Sign Authenticate

Current allowed actions: Sign

(S) Toggle the sign capability

(A) Toggle the authenticate capability

(Q) Finished

Your selection? **S**

Enter s – for removing Signature.

Possible actions for a ECDSA/EdDSA key: Sign Authenticate

Current allowed actions:

(S) Toggle the sign capability

(A) Toggle the authenticate capability

(Q) Finished

Your selection? **a**

Enter a – for adding Authenticate.

Possible actions for a ECDSA/EdDSA key: Sign Authenticate

Current allowed actions: Authenticate

(S) Toggle the sign capability

(A) Toggle the authenticate capability

(Q) Finished

Your selection? q

Enter q – for Finish.

Please select which elliptic curve you want:

(1) Curve 25519

(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

Your selection? **1**

Enter 1 – for Curve 25519.

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) **3y**

Enter 3y – for 3 years to allow logins.

Is this correct? (y/N) **y**

Enter y – for correct.

Really create? (y/N) **y**

Enter y – to create.

Enter your secure password, made in the 1st few steps.

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.

sec ed25519/E0B518BE7CA2EC83

created: 2022-10-05 expires: never usage: C

trust: ultimate validity: ultimate

ssb cv25519/0D89FFD9061948B6

created: 2022-10-05 expires: 2025-10-04 usage: E

ssb ed25519/44625D254F64802E

created: 2022-10-05 expires: 2025-10-04 usage: A

[ultimate] (1). Robert S. <Robert@host.local>

gpg> **save**

Enter save – for Save and Exit.

In the following, replace **[YOUREMAIL]** with your email used in GPG, here I used Robert@host.local.

$ gpg2 --armor --export-secret-keys **[YOUREMAIL]** > master.key

Enter your secret password you made earlier.

$ gpg2 --gen-revoke **[YOUREMAIL]** > revoke.asc

gpg: WARNING: unsafe permissions on homedir '/tmp/new-gpg'

sec ed25519/E0B518BE7CA2EC83 2022-10-05 Robert S. <Robert@host.local>

Create a revocation certificate for this key? (y/N) y

Please select the reason for the revocation:

0 = No reason specified

1 = Key has been compromised

2 = Key is superseded

3 = Key is no longer used

Q = Cancel

(Probably you want to select 1 here)

Your decision? **0**

Enter 0 – for No reason specifically.

Enter an optional description; end it with an empty line:

>

Reason for revocation: No reason specified

(No description given)

Is this okay? (y/N) **y**

Enter your secret password you made earlier.

Revocation certificate created.

$ gpg2 --armor --export **[YOUREMAIL]** > pub.key

$ gpg2 --armor --export-secret-subkeys **[YOUREMAIL]** > sub.key

Enter your secret password you made earlier.

$ gpg2 --fingerprint --fingerprint **[YOUREMAIL]** > fingerprint.txt

Let’s make an compressed archive:

$ tar -czvf ~/my-gpg-secrets.tar.gz -C /tmp/my_keys *

Now you’ve backed up to: /home/$USER/my-gpg-secrets.tar.gz

You’ll want to keep this ULTRA safe in LUKS USB drive!!

If, something bad happened and you've lost your keys, you may re-import them: gpg2 --allow-secret-key-import --import key-file

Be sure to replace key-file with the location of each of your files.

----------Now let's move your GPG keys into YubiKey----------

Check out YubiKey USB device:

$ gpg --card-status

$ ykman info

$ ykman openpgp info

Note: Some of these commands may ask for a PIN or Admin PIN. The default PIN is usually 123456, and the default Admin PIN is usually 12345678. If these don't work, contact the manufacturer or review online documentation.

(Insert USB Yubikey)...now.

Force YubiKey to be touched before Auth/Logins...

$ ykman openpgp set-touch enc on

$ ykman openpgp set-touch aut on

$ ykman openpgp set-touch sig on

$ gpg2 --card-edit

> admin

> passwd

1. Change the password to your device by selecting `2 - unblock PIN`. This will unblock your PIN, and prompt you to change it. This PIN will be required every time you want to access your GPG key (e.g. every time you authenticate with SSH), and has a limit of eight characters.
2. Change the admin PIN by selecting `3 - change Admin PIN`. This PIN is required to make administrative changes, like in step 2, and has a limit of 6 characters. For optimum security, never store this PIN in a digital location, since it will be unnecessary for daily use of the YubiKey.
3. Exit these menus by selecting `Q` and then typing `quit`.

gpg2 --edit-key [YOUR key-id]

> trust

>> 5 to Trust Ultimately

> quit

gpg2 --list-key

(Should show uid [ultimate] email@domain.com)

(Insert USB Yubikey)...now, if not in already.

$ gpg2 --edit-key [YOUR key-id]

> toggle

> key 1

> keytocard

> key 2

> keytocard

Repeat as needed...for each sub-key used.

> save

gpg2 --card-status

gpg2 --list-secret-keys

ssb> The > means not on the computer, but kept on external USB device! Cool!!!

Now do: $ssh-add -L

Copy this output to your Server’s ~/.ssh/authorized_keys file.

------------------Using GPG---------------------------

(Sign a File:) $ gpg2 --clear-sign doc.txt

(Verify file:) $ gpg2 --verify doc.txt.asc

(Encrypt a File:) $ gpg2 -r [EMAIL] --encrypt doc.txt

(decode:) $ gpg2 --decrypt doc.txt.gpg > document.txt

--------------------Using PASS--------------------------

gpg2 --list-key

Copy the fingerprint below pub [C]

pass init (Paste in that fingerprint HERE.)

To add a password: $ pass insert email/myserver22

To see all: $ pass

To view a password: $ pass email/myserver22

To copy it to the clipboard: $ pass -c email/myserver22

To store multiple lines of text: $ pass -m sites/linode

{{The first line is the password, others maybe username, etc...}}

Data is stored in: ~/.password-store

Manually adding: cd ~/.password-store/btc , then

$ gpg2 -r [EMAIL] --encrypt bitcoin1

[Note: if you forgot to have the Yubikey inserted and get an error...] $ sudo systemctl restart pcscd