# OpenPGP

- Create Smartcard-based PGP Key

# Create Smartcard-based PGP Key

## Requirements

- OpenPGP-compatible Hardware Security Key (Example: YubiKey 5(C))
- GnuPG CLI installed
- Smartcard-related Drivers installed

## Key Setup

### Prepare Card

#### Ensure Card is recognized

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

The Output should look something like this

```
Reader ...........: Yubico YubiKey OTP FIDO CCID 0
Application ID ...: D2760001240100000006379790790000
Application type .: OpenPGP
Version ..........: 3.4
Manufacturer .....: Yubico
```

### Configure Card

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

> **If Command is preceeded by `gpg/card>`, you have to be in card edit Mode using `gpg --card-edit`**

**Configure Pin**

>

> **For YubiKeys, the default Pin is `123456` and default Puk is `4`**

```
gpg/card> admin
gpg/card> passwd
```

- Press `1` to edit User-Pin
- Press `3` to edit Admin Pin
- Press `q` to quit

**Set Algorithm to ECC**

```
gpg/card> key-attr
```

Select Curve 25519 Slot

```
(1) RSA
(2) ECC
```

`(1) Curve 25519 <em>standard</em>(4) NIST P-384(6) Brainpool P-256`

- *ED25519* for Auth/Sign
- *CV25519* for Encryption

# Generate Key

```
gpg/card> generate
```

> **If asked to replace existing keys, confirm with `y` only if you intend to overwrite what's currently on the card**

Follow the prompts:

- **Make off-card backup of encryption key?** → `Y` (this is the only key material that can be backed up — Signature and Authentication keys never leave the card)
- **Key validity period** → e.g. `2y` (avoid `0`/unlimited, to enforce periodic rotation)
- **Real name** → your name
- **Email address** → your primary email (can be extended later, see Add additional User-IDs)
- **Comment** → leave empty

GnuPG will generate all three subkeys (Signature, Encryption, Authentication) directly on the card.

>

> **Key generation requires entropy — moving the mouse or typing in another window speeds this up**

## Verify

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

Signature key, Encryption key and Authentication key should now show fingerprints instead of `[none]`.

```
gpg --list-secret-keys --keyid-format=long
```

Secret keys should be marked `sec>` / `ssb>` — the `>` indicates the private key material resides on the card, not on disk.

## Backup

### Export Public Key

```
gpg --armor --export <KEY-ID> > public-key.asc
```

### Revocation Certificate

Automatically created during key generation, located at:

```
%APPDATA%\gnupg\openpgp-revocs.d\<KEY-ID>.rev
```

> **Store this file separately from the YubiKey (e.g. encrypted USB drive). Without it, a lost/broken card cannot be revoked.**

### Encryption Key Backup

Located at:

```
%APPDATA%\gnupg\private-keys-v1.d\
```

> **Unlike the other two keys, this file contains actual private key material (the Encryption subkey). Store it encrypted and separately — do not leave it unprotected on disk.**

# Add Additional User-IDs

Additional identities (e.g. further email addresses) can be attached to the existing key without regenerating anything on the card.

```
gpg --edit-key <KEY-ID>
```

```
gpg> adduid
```

- **Real name** → e.g. same as before
- **Email address** → additional address
- **Comment** → leave empty

Confirm with `O` (Okay), then:

```
gpg> save
```

Re-export the updated Public Key and re-upload it wherever the old one was registered (e.g. Forgejo):

```
gpg --armor --export <KEY-ID> > public-key-v2.asc
```

# Git Integration

```
git config --global user.signingkey <KEY-ID>
git config --global commit.gpgsign true
git config --global tag.gpgsign true
```

> **Git for Windows ships its own bundled `gpg.exe` (MSYS2), which uses a separate `GNUPGHOME` and will not see the card. Point Git explicitly to the system GnuPG installation:**

```
where.exe gpg
git config --global gpg.program "C:\Program Files (x86)\GnuPG\bin\gpg.exe"
```

**Verify**

```
git commit --allow-empty -m "Test signed commit"
git log --show-signature -1
```

Expected output includes `gpg: Good signature`.