ykman

```
> brew install ykman
> ykman oath accounts code Amazon -s
```

123456

12345678

Back Setup for macOS

sudo /etc/pam.d/sudo

```
> cat /etc/pam.d/sudo
# sudo: auth account password session
auth       sufficient     pam_tid.so
auth       sufficient     pam_smartcard.so
auth       required       pam_opendirectory.so
account    required       pam_permit.so
password   required       pam_deny.so
session    required       pam_permit.so
```

pam_smartcard.so sudo
pam_tid.so sudo

_____

-
-

_____ git clone make install

OTP

brew install openssh

PATH export PATH=$(brew --prefix)/bin:$PATH

-
- ssh-keygen -t ecdsa-sk -O resident
- id_ecdsa_sk.pub

_____ git pull ssh

```
> git pull
Confirm user presence for key ECDSA-SK SHA256:<FINGERPRINT IS HERE>
User presence confirmed
Already up to date.
```

Confirm user presence for key

```
> ssh-add -K
> ssh-keygen -K
> mv id_ecdsa_sk_rk ~/.ssh/id_ecdsa_sk
```

id_ecdsa_sk

```
> ssh-keygen -t ecdsa-sk -O resident -f id_ecdsa_sk_backup
```

~/.ssh/config

```
Host *
    IdentitiesOnly Yes #Optional
    IdentityFile ~/.ssh/id_ecdsa_sk
    IdentityFile ~/.ssh/id_ecdsa_sk_backup
```

```
> ssh -T git@github.com
Confirm user presence for key ECDSA-SK SHA256:CfVjTqE4nnPnycjFDcymwtK87949jk
sign_and_send_pubkey: signing failed for ECDSA-SK "/Users/fhammerl/.ssh/id_e
Confirm user presence for key ECDSA-SK SHA256:lpNnp6lh+Pf3Y1D0otvvUyDKrefUbQ
User presence confirmed
Hi felixhammerl! You've successfully authenticated, but GitHub does not prov
```

- brew install gnupg gpgme pinentry-mac
- GNUPGHOME .zshrc export GNUPGHOME=~/.gnupg
-
- wget -O $GNUPGHOME/gpg.conf https://raw.githubusercontent.com/drduh/config/master/gpg.conf
- $GNUPGHOME/gpg.conf # throw-keyids # throw-keyids
- wget -O $GNUPGHOME/gpg-agent.conf https://raw.githubusercontent.com/drduh/config/master/gpg-agent.conf
- $GNUPGHOME/gpg-agent.conf # pinentry-program /opt/homebrew/bin/pinentry-mac
- $GNUPGHOME/gpg-agent.conf # pinentry-program /usr/bin/pinentry-curses

```
> gpg --expert --full-generate-key
Please select what kind of key you want:
   (1) RSA and RSA
   (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 (sign and encrypt) *default*
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
  (13) Existing key
  (14) Existing key from card
Your selection? 8
```

```
Possible actions for this 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? S




Possible actions for this 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? E




Possible actions for this 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? Q




RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 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) 0

                                    0


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




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

Real name: Felix Hammerl
Email address: felix.hammerl@gmail.com
Comment:
You selected this USER-ID:
    "Felix Hammerl <felix.hammerl@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 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.
gpg: revocation certificate stored as '/Users/fhammerl/.gnupg/openpgp-revocs
public and secret key created and signed.

pub   rsa4096/0x1E2BD87C697C5DDD 2022-08-30 [C]
      Key fingerprint = 69B3 C01A 5E0F 87BE BC18  1C74 1E2B D87C 697C 5DDD
uid                              Felix Hammerl <felix.hammerl@gmail.com>




> export KEYID=0x1E2BD87C697C5DDD
```

```
> gpg --expert --edit-key $KEYID
Secret key is available.

gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
sec  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1). Felix Hammerl <felix.hammerl@gmail.com>

gpg> adduid



Real name: Felix Hammerl
Email address: felix@example.org
Comment:
You selected this USER-ID:
    "Felix Hammerl <felix@example.org>"

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




sec  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1)  Felix Hammerl <felix.hammerl@gmail.com>
[ unknown] (2). Felix Hammerl <felix@example.org>

gpg> uid 1

                                                                      uid
1                          uid 1



sec  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1)* Felix Hammerl <felix.hammerl@gmail.com>
[ unknown] (2). Felix Hammerl <felix@example.org>

gpg> primary
```

```
sec  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1)* Felix Hammerl <felix.hammerl@gmail.com>
[ unknown] (2)  Felix Hammerl <felix@example.org>

gpg> save
```

uid

```
> gpg --expert --edit-key $KEYID
Secret key is available.

gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
sec  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1). Felix Hammerl <felix@example.org>
[ultimate] (2)  Felix Hammerl <felix.hammerl@gmail.com>



gpg> 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
  (14) Existing key from card
Your selection? 4



RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 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)




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



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  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
ssb  rsa4096/0xB800E83563709867
     created: 2022-08-30  expires: never       usage: S
[ultimate] (1). Felix Hammerl <felix@example.org>
[ultimate] (2)  Felix Hammerl <felix.hammerl@gmail.com>



gpg> 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
```

```
  (14) Existing key from card
Your selection? 6



RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 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)



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



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  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
ssb  rsa4096/0xB800E83563709867
     created: 2022-08-30  expires: never       usage: S
ssb  rsa4096/0x460AAFECD0F316C1
     created: 2022-08-30  expires: never       usage: E
[ultimate] (1). Felix Hammerl <felix@example.org>
[ultimate] (2)  Felix Hammerl <felix.hammerl@gmail.com>



gpg> 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
  (14) Existing key from card
Your selection? 8



Possible actions for this 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? S



Possible actions for this RSA key: Sign Encrypt Authenticate
Current allowed actions: Encrypt

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

Your selection? E



Possible actions for this RSA key: Sign Encrypt Authenticate
Current allowed actions:

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

```
Your selection? A



Possible actions for this RSA key: Sign Encrypt Authenticate
Current allowed actions: Authenticate

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

Your selection? Q



RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 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)



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



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  rsa4096/0x1E2BD87C697C5DDD
     created: 2022-08-30  expires: never       usage: C
     trust: ultimate      validity: ultimate
ssb  rsa4096/0xB800E83563709867
     created: 2022-08-30  expires: never       usage: S
ssb  rsa4096/0x460AAFECD0F316C1
     created: 2022-08-30  expires: never       usage: E
```

```
ssb  rsa4096/0x489C6E09BDDB455B
     created: 2022-08-30  expires: never       usage: A
[ultimate] (1). Felix Hammerl <felix@example.org>
[ultimate] (2)  Felix Hammerl <felix.hammerl@gmail.com>

gpg> save
```

```
> gpg --armor --export-secret-keys $KEYID > master.key
> gpg --armor --export $KEYID > master.pub
```

master.key master.pub

```
> gpg --card-edit
...
Information about your Yubikey
...

gpg/card> admin
```

```
Admin commands are allowed

gpg/card> passwd
```

```
gpg: OpenPGP card no. D2760001240102010006055532110000 detected

1 - change PIN
2 - unblock PIN
3 - change Admin PIN
4 - set the Reset Code
Q - quit

Your selection? 3
```

12345678

```
PIN changed.

1 - change PIN
2 - unblock PIN
3 - change Admin PIN
4 - set the Reset Code
Q - quit

Your selection? 1

                                                    123456


PIN changed.

1 - change PIN
2 - unblock PIN
3 - change Admin PIN
4 - set the Reset Code
Q - quit

Your selection? q




gpg/card> quit
```

```
> gpg --edit-key $KEYID

Secret key is available.

sec  rsa4096/0xFF3E7D88647EBCDB
    created: 2017-10-09  expires: never       usage: C
    trust: ultimate      validity: ultimate
ssb  rsa4096/0xBECFA3C1AE191D15
    created: 2017-10-09  expires: 2018-10-09  usage: S
ssb  rsa4096/0x5912A795E90DD2CF
    created: 2017-10-09  expires: 2018-10-09  usage: E
ssb  rsa4096/0x3F29127E79649A3D
    created: 2017-10-09  expires: 2018-10-09  usage: A
[ultimate] (1). Dr Duh <doc@duh.to>

gpg> key 1
```

```
sec  rsa4096/0xFF3E7D88647EBCDB
    created: 2017-10-09  expires: never       usage: C
    trust: ultimate      validity: ultimate
ssb* rsa4096/0xBECFA3C1AE191D15
    created: 2017-10-09  expires: 2018-10-09  usage: S
ssb  rsa4096/0x5912A795E90DD2CF
    created: 2017-10-09  expires: 2018-10-09  usage: E
ssb  rsa4096/0x3F29127E79649A3D
    created: 2017-10-09  expires: 2018-10-09  usage: A
[ultimate] (1). Dr Duh <doc@duh.to>

gpg> keytocard



Please select where to store the key:
   (1) Signature key
   (3) Authentication key
Your selection? 1




You need a passphrase to unlock the secret key for
user: "Dr Duh <doc@duh.to>"
4096-bit RSA key, ID 0xBECFA3C1AE191D15, created 2016-05-24

gpg> key 1




gpg> key 2




sec  rsa4096/0xFF3E7D88647EBCDB
    created: 2017-10-09  expires: never       usage: C
    trust: ultimate      validity: ultimate
ssb  rsa4096/0xBECFA3C1AE191D15
    created: 2017-10-09  expires: 2018-10-09  usage: S
ssb* rsa4096/0x5912A795E90DD2CF
    created: 2017-10-09  expires: 2018-10-09  usage: E
ssb  rsa4096/0x3F29127E79649A3D
    created: 2017-10-09  expires: 2018-10-09  usage: A
```

```
[ultimate] (1). Dr Duh <doc@duh.to>

gpg> keytocard



Please select where to store the key:
   (2) Encryption key
Your selection? 2




gpg> key 2




gpg> key 3




sec  rsa4096/0xFF3E7D88647EBCDB
    created: 2017-10-09  expires: never       usage: C
    trust: ultimate      validity: ultimate
ssb  rsa4096/0xBECFA3C1AE191D15
    created: 2017-10-09  expires: 2018-10-09  usage: S
ssb  rsa4096/0x5912A795E90DD2CF
    created: 2017-10-09  expires: 2018-10-09  usage: E
ssb* rsa4096/0x3F29127E79649A3D
    created: 2017-10-09  expires: 2018-10-09  usage: A
[ultimate] (1). Dr Duh <doc@duh.to>

gpg> keytocard
Please select where to store the key:
   (3) Authentication key
Your selection? 3



gpg> save
```

```
> gpg --delete-secret-keys $KEYID
gpg (GnuPG) 2.3.7; Copyright (C) 2021 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.


sec  rsa4096/1E2BD87C697C5DDD 2022-08-30 Felix Hammerl <felix@example.org>

Delete this key from the keyring? (y/N) y
This is a secret key! - really delete? (y/N) y



[30/08/22 18:49:53] ~
> gpg --delete-keys $KEYID
gpg (GnuPG) 2.3.7; Copyright (C) 2021 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.


pub  rsa4096/1E2BD87C697C5DDD 2022-08-30 Felix Hammerl <felix@example.org>

Delete this key from the keyring? (y/N) y



> gpg -K




> gpg --import master.key
gpg: key 1E2BD87C697C5DDD: public key "Felix Hammerl <felix@example.org>" im
gpg: key 1E2BD87C697C5DDD: secret key imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg:       secret keys read: 1
gpg:   secret keys imported: 1
[30/08/22 18:52:36] ~
> gpg -K
/Users/fhammerl/.gnupg/pubring.kbx
---------------------------------
sec   rsa4096 2022-08-30 [C]
      69B3C01A5E0F87BEBC181C741E2BD87C697C5DDD
uid           [ unknown] Felix Hammerl <felix@example.org>
uid           [ unknown] Felix Hammerl <felix.hammerl@gmail.com>
ssb   rsa4096 2022-08-30 [S]
ssb   rsa4096 2022-08-30 [E]
ssb   rsa4096 2022-08-30 [A]
```

keytocard

```
$ gpg --send-key $KEYID

$ gpg --keyserver pgp.mit.edu --send-key $KEYID

$ gpg --keyserver keys.gnupg.net --send-key $KEYID

$ gpg --keyserver hkps://keyserver.ubuntu.com:443 --send-key $KEYID
```

```
> gpg --delete-secret-keys $KEYID

> gpg --delete-keys $KEYID

> gpg --keyserver pgp.mit.edu --recv-keys $KEYID

> gpg-connect-agent "scd serialno" "learn --force" /bye
```

```
gpg-connect-agent "scd serialno" "learn --force" /bye
```

throw-keyids gpgme gpg.conf

```
> cat ~/Library/Application\ Support/Mozilla/NativeMessagingHosts/gpgmejson.
{
    "name": "gpgmejson",
    "description": "Integration with GnuPG",
    "path": "/opt/homebrew/bin/gpgme-json",
    "type": "stdio",
    "allowed_extensions": [
        "jid1-AQqSMBYb0a8ADg@jetpack"
    ]
}
```

```
> cat ~/Library/Application\ Support/Google/Chrome/NativeMessagingHosts/gpgm
{
    "name": "gpgmejson",
    "description": "Integration with GnuPG",
    "path": "/opt/homebrew/bin/gpgme-json",
    "type": "stdio",
    "allowed_origins": [
        "chrome-extension://kajibbejlbohfaggdiogboambcijhkke/"
    ]
}
```

gpgmejson.json

```
> sudo launchctl config system path /opt/homebrew/bin:/usr/local/bin:/usr/bi
> sudo launchctl config user path /opt/homebrew/bin:/usr/local/bin:/usr/bin:
```

launchd

- `gpg --armor --export $KEYID | pbcopy`
-

---

gpgconf --kill gpg-agent

```
export GNUPGHOME=~/.gnupg
function reset_gpg() {
  gpg-connect-agent "scd serialno" "learn --force" /bye
}
function kill_gpg() {
  gpgconf --kill gpg-agent
}
```