It's going to be a lame party isn't it. . .

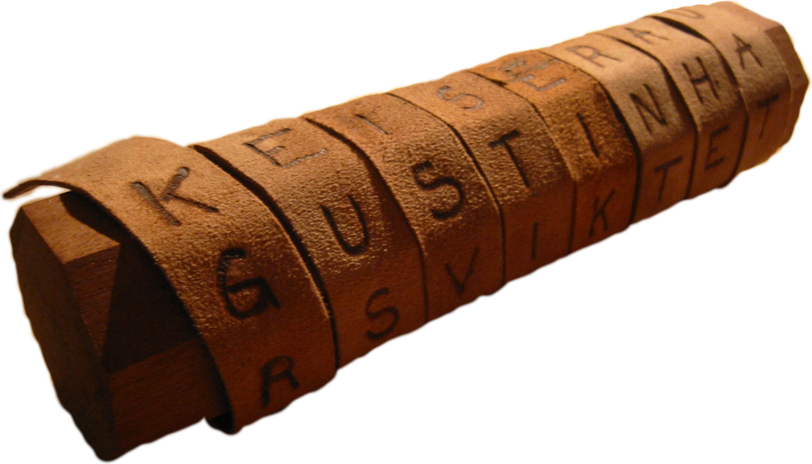


Figure 1: scytale

We dont actually use scytales anymore for encryption. . .

But we still use symmetric encryption

- *Same key* for **encrypt** and **decrypt**

But today we are going to talk about asymmetric encryption

- *Different keys* for **encrypt** and **decrypt**

Asymmetric encryption

Two keys (at least) that are mathematically related.

- **Public key**
  - known to the public (everyone)
  - can **encrypt** messages that only the *private key* can **decrypt**
  - can **decrypt** messages the *private key* **encrypts**
- **Private key**
  - known only to you (or the owner)
  - can **encrypt** messages only the *public key* can **decrypt**
  - can **decrypt** messages the *public key* **encrypts**

Building our own Web of Trust

Today we are going to build a (small) web of trust by having each of you do the following:

1. Create public and private keys with: `gpg --full-generate-key`
2. Check (and write down) your fingerprint: `gpg --fingerprint <email>`
3. `gpg -o mykey.gpg --export <email>` OR `gpg -o mykey.txt --armor --export <email>` (this last one gets you a plain text file)
4. Exchange emails and fingerprints (and trust)
5. Exchange public keys via email (or other means)
6. `gpg --import <their-key-file>`
7. After this we have options:

- Sign each others keys
- encrypt a message and send it
- sign a message with your key

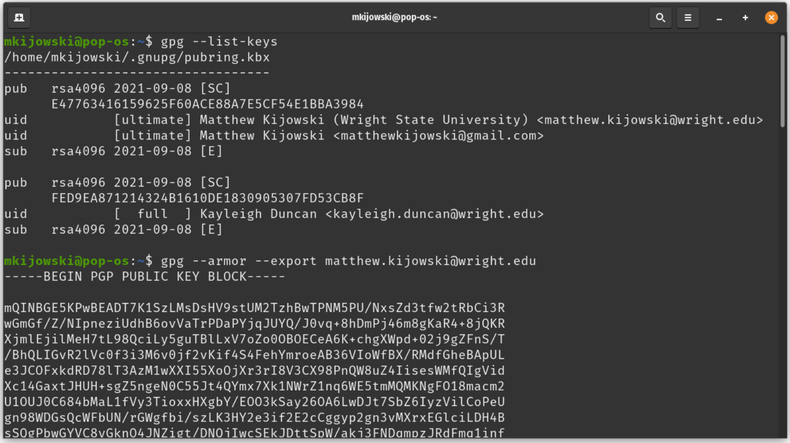
Getting started

GPG Downloads page

```
gpg --full-generate-key
```

- Select the default keys to make (option 1)
- Use your real name and email!
- Use all the bits (4096)!
- Feel free to set an expiration of 1 (day) to 0 (no expiration)
- Use a password that you will not forget!

I have a key, now what?



```
mkijowski@pop-os:~$ gpg --list-keys
/home/mkijowski/.gnupg/pubring.kbx
---------------------------------
pub   rsa4096 2021-09-08 [SC]
      E47763416159625F60ACE88A7E5CF54E1BBA3984
uid           [ultimate] Matthew Kijowski (Wright State University) <matthew.kijowski@wright.edu>
uid           [ultimate] Matthew Kijowski <matthewkijowski@gmail.com>
sub   rsa4096 2021-09-08 [E]

pub   rsa4096 2021-09-08 [SC]
      FED9EA871214324B1610DE1830905307FD53CB8F
uid           [  full  ] Kayleigh Duncan <kayleigh.duncan@wright.edu>
sub   rsa4096 2021-09-08 [E]

mkijowski@pop-os:~$ gpg --armor --export matthew.kijowski@wright.edu
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGE5KPwBEADT7K1SzLMsDsHV9stUM2TzhBwTPNM5PU/NxsZd3tfw2tRbCi3R
wGmGf/Z/NIpneziUdhB6ovVaTrPDaPYjqJUYQ/J0vq+8hDmPj46m8gKaR4+8jQKR
XjmlEjilMeH7tL98QciLy5guTBlLxV7oZo0OBOECeA6K+chgXWpd+02j9gZFnS/T
/BhQLIGvR2lVc0f3i3M6v0jf2vKif4S4FehYmroeAB36VIoWfBX/RMdfGheBApUL
e3JCOFxkdRD78lT3AzM1wXXI55XoOjXr3rI8V3CX98PnQW8uZ4IisesWMfQIgVid
Xc14GaxtJHUH+sgZ5ngeN0C55Jt4QYmx7Xk1NWrZ1nq6WE5tmMQMKNgFO18macm2
U1OUJ0C684bMaL1fVy3TioxxHXgbY/EOO3kSay26OA6LwDJt7SbZ6IyzVilCoPeU
gn98WDGsQcWFbUN/rGWgfbi/szLK3HY2e3if2E2cCggyp2gn3vMXrxEGlciLDH4B
sSOgPbwGYVC8vGknO4JNZigt/DNOjIwcSEkJDttSpW/akj3FNDqmpzJRdFmq1inf
```

- Get fingerprint with: `gpg --fingerprint matthew.kijowski@wright.edu`
- Convince your table mates that you are the person with the given email and share your fingerprint!
- Exchange emails and key fingerprints

Importing public keys

- `gpg --import kijowski.gpg`

Check to make sure the fingerprint matches:

```
E477 6341 6159 625F 60AC  E88A 7E5C F54E 1BBA 3984
                               7E5C F54E 1BBA 3984
```

Signing someone's key

Signing a key simply puts your digital signature on their key, implying that you have some trust that the person that the key belongs to is who they say they are. People who trust you will recognize your signature on other people's keys, increasing trust without having to have actually met in person.

- `gpg --edit-key matthew.kijowski@wright.edu`
- `sign`
- `save`
- `gpg --armor --export matthew.kijowski@wright.edu`

What to do with your public key

Signed sealed delivered

Lets sign a message!

- Create a `sample.txt` file with a public message (Hello World or some such thing).
- `gpg --sign sample.txt`
- share the output sample.gpg with someone you exchanged keys with
- `cat` the output file, can you read the contents?
- `gpg --verify sample.gpg`

Now for some fun

Lets send a secret message!

- Create a text file `secret-message.txt`
- Choose someone you have exchanged keys with
- Encrypt the file: `gpg --output secret-message.gpg --encrypt --recipient their.email@wright.edu`
- Send them `secret-message.txt` via email or discord
- The recipient can decrypt with:
  - `gpg --output secret.txt --decrypt secret-message.gpg`

Back up your gpg keys!!!

```
tar -cpzf gnupg.tar.gz ~/.gnupg/
```

if you are using a Wright State laptop

```
cp gnupg.tar.gz /mnt/c/Users/student/Desktop/
```

Save this file!!!

You can also backup your private key and any public keys with:

- `gpg --armor --export-secret-key your.email@wright.edu`
- `gpg --armor --export your.email@wright.edu`
- `gpg --armor --export friends@wright.edu`