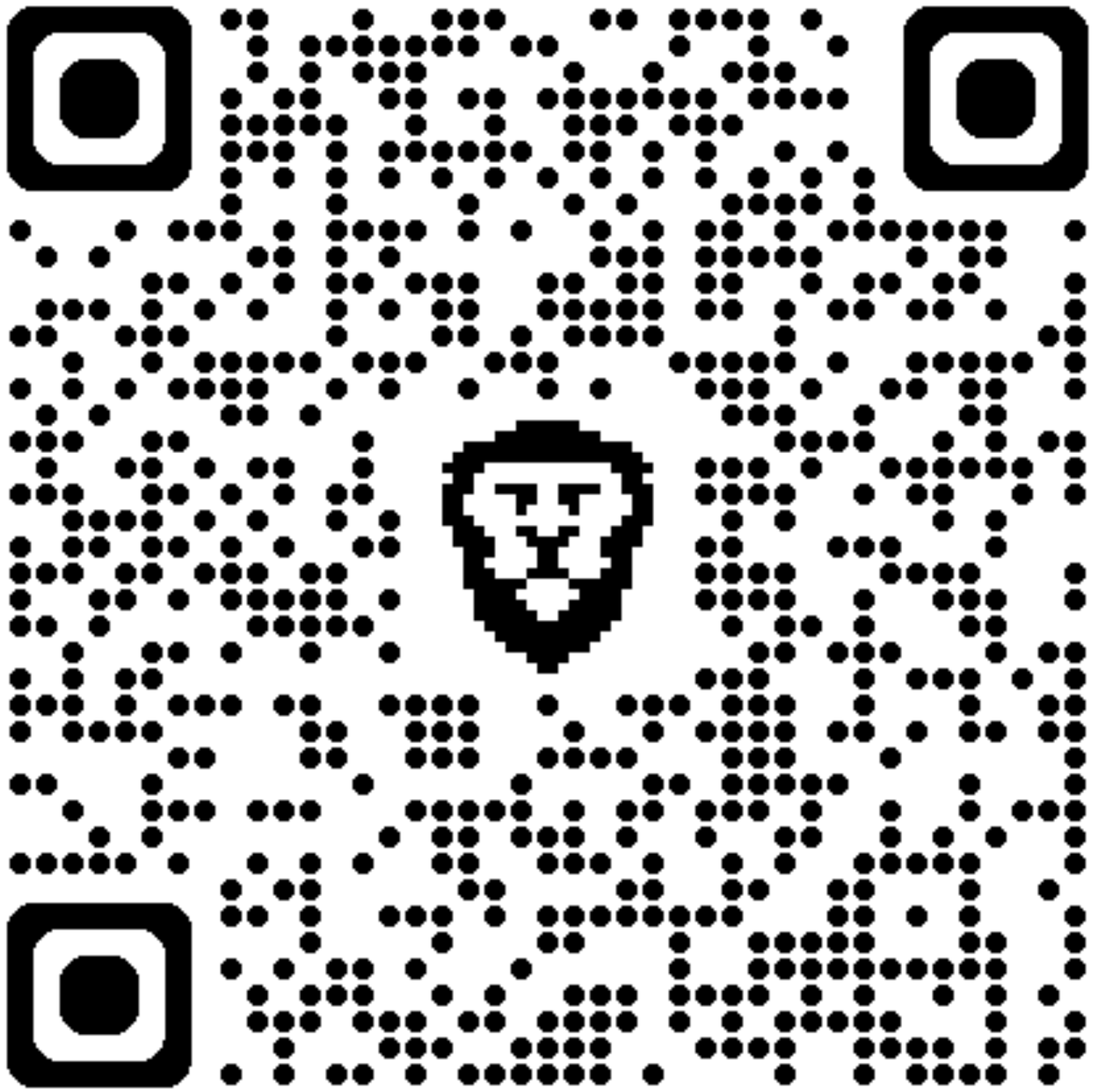


Get the slides!

Implementing HTTPS In Python

Thierry Sans

5 servers - **Only one can be trusted**

*"There are **5 servers but only one can be trusted**. Your mission, if you choose to accept it, is to **implement a proper TLS client to find the legitimate server** and uncover the 4 imposters"*

Your HTTPS client must:

- Exchange a symmetric key
- Verify the server's identify (signature + certificate)
- Use the symmetric key to decrypt the data
  if and only if the server is trusted

Step by Step Process

Step 0: Setting-up the work environment

Step 1: Key Exchange and Data Decryption

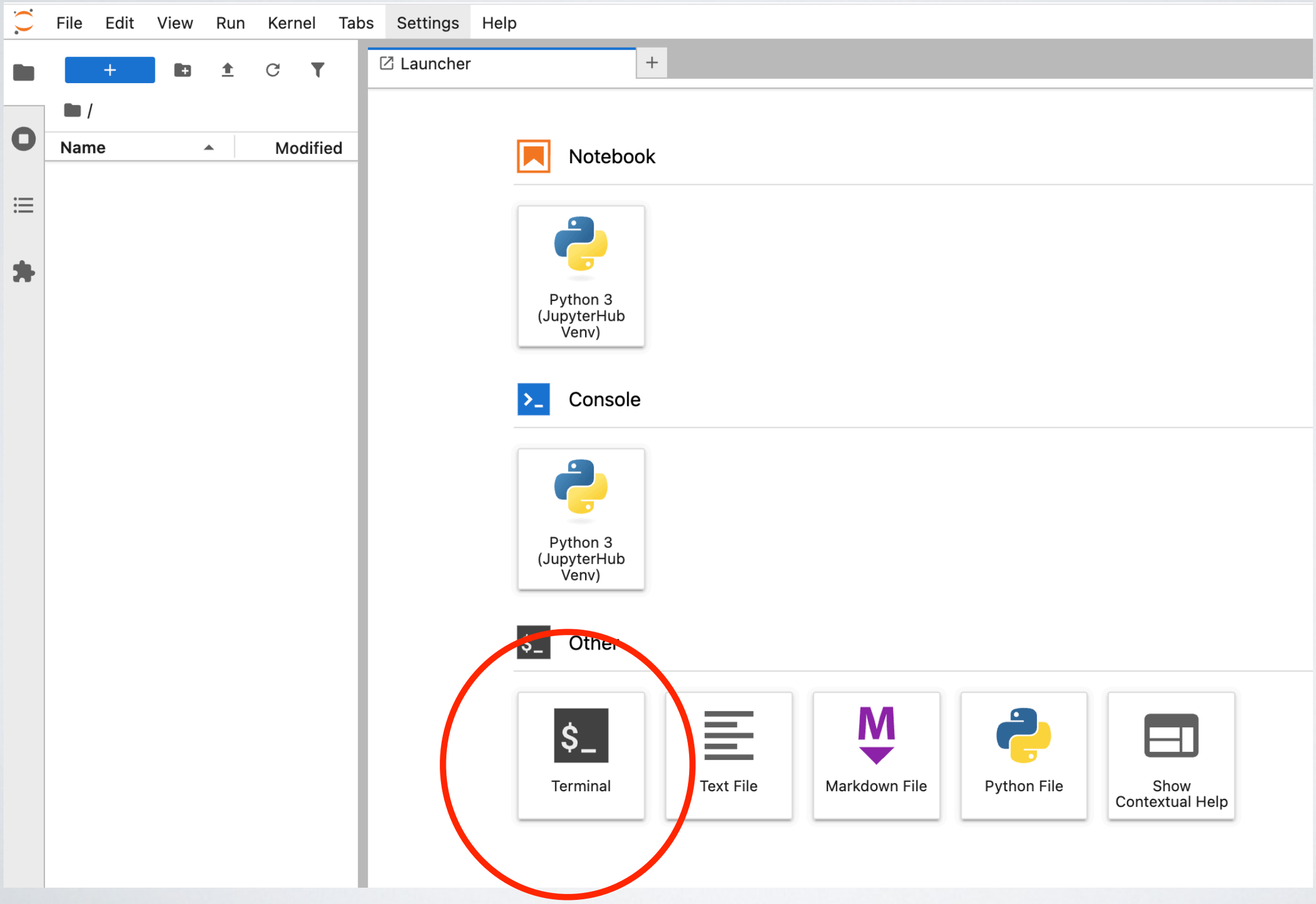
Step 2: Certificate and Server's Signature Verifications

Final Step: Find the trusted server among the imposters

Step 0

Setup the Work Environment

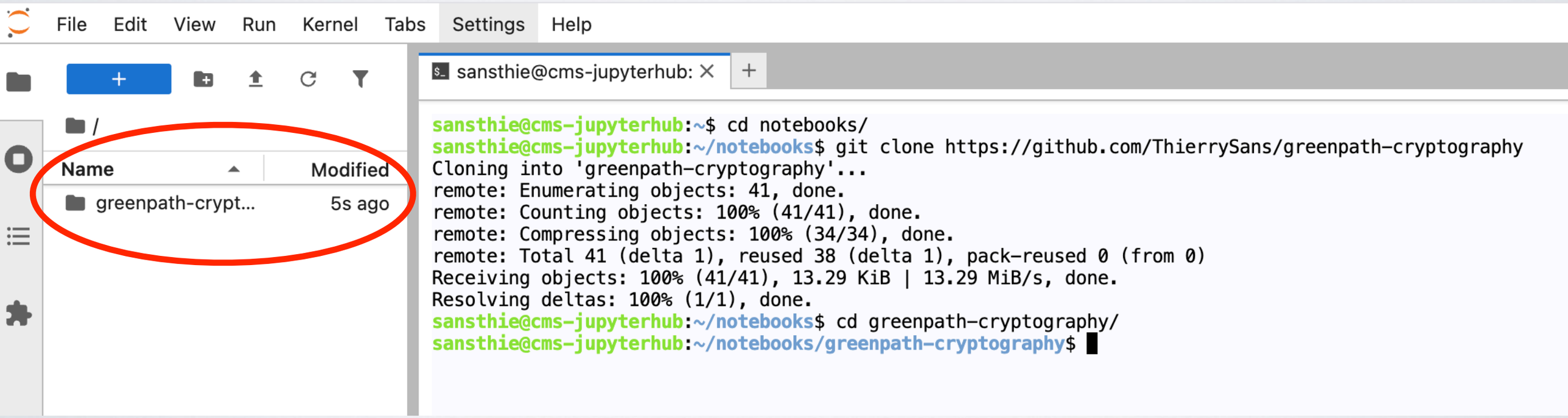
Starting the Console



Fetching the Code

In the console, type these three commands:

1. `cd notebooks`
2. `git clone https://github.com/ThierrySans/greenpath-cryptography`
3. `cd greenpath-cryptography`



Working with the Code

File Edit View Run Kernel Tabs Settings Help

/ greenpath-cryptography /

Name	Modified
config	3m ago
cryptobox	3m ago
client.py	3m ago
README.md	3m ago
requirements.txt	3m ago
run.py	3m ago
server.py	3m ago

client.py

```python
from cryptobox import *

class Client():
    def __init__(self, url: bytes, trusted_certs: list[bytes]):
        self.url = url
        self.trusted_certs = trusted_certs

    def send_client_hello(self) -> bytes:
        print(self.url)
        print(self.trusted_certs)

    def send_client_ready(self, server_hello: bytes) -> bytes:
        None

    def receive_data(self, data: bytes) -> bytes:
        None

```

sansthie@cms-jupyterhub:

```
sansthie@cms-jupyterhub:~$ cd notebooks/
sansthie@cms-jupyterhub:~/notebooks$ git clone https://github.com/ThierrySans/greenpath-cryptography
Cloning into 'greenpath-cryptography'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 41 (delta 1), reused 38 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (41/41), 13.29 KiB | 13.29 MiB/s, done.
Resolving deltas: 100% (1/1), done.
sansthie@cms-jupyterhub:~/notebooks$ cd greenpath-cryptography/
sansthie@cms-jupyterhub:~/notebooks/greenpath-cryptography$
```

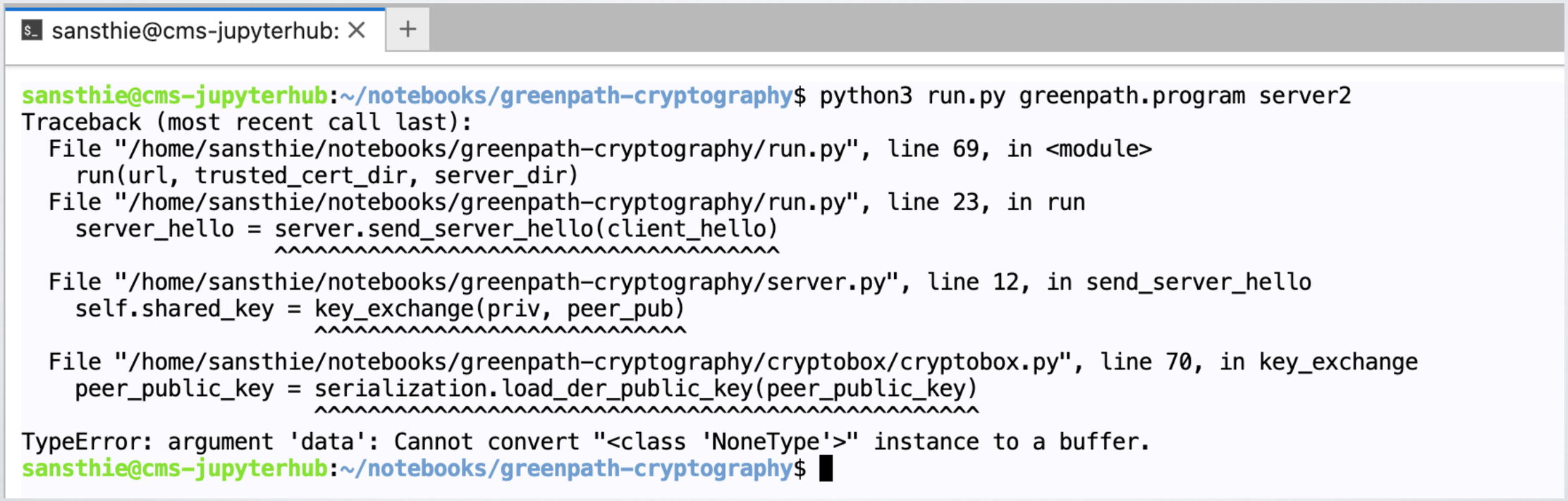
Running the Code

In the console, run the command:

```
python3 run.py greenpath.program server2
```

The server id from **`server1`** to **`server5`**

The domain name from of the url you would type in the browser

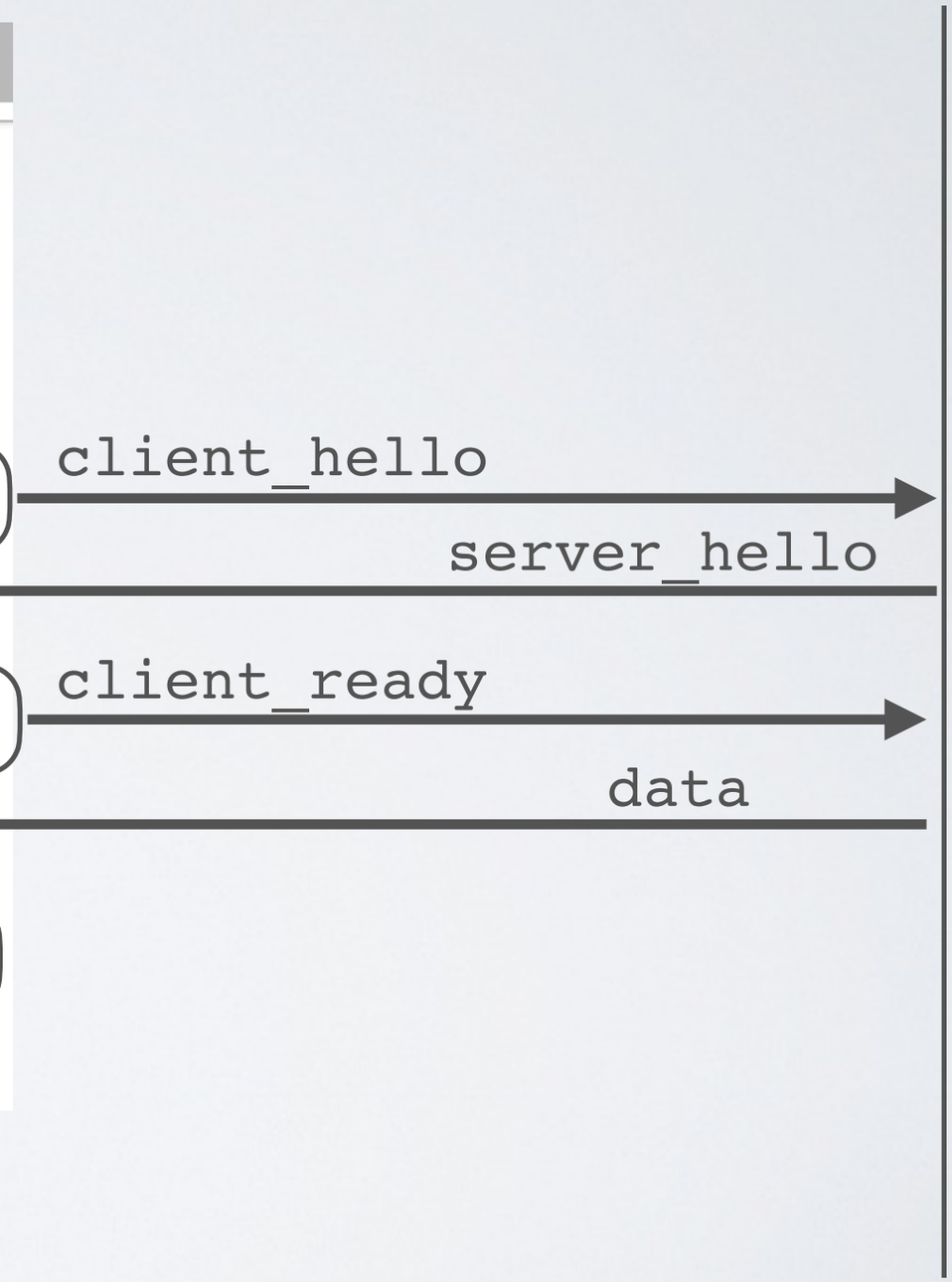


Understanding `client.py`

client.py

```python
from cryptobox import *

class Client():
    def __init__(self, url: bytes, trusted_certs: list[bytes]):
        self.url = url
        self.trusted_certs = trusted_certs

    def send_client_hello(self) -> bytes:
        None

    def send_client_ready(self, server_hello: bytes) -> bytes:
        None

    def receive_data(self, data: bytes) -> bytes:
        None
```

Should return the decrypted data



Step 1

Key Exchange and Data Decryption

send_client_hello:

1. Generate Ks_1, Kp_1

client_hello: Kp_1

send_server_hello:

1. Generate Ks_2, Kp_2
2. Derive $k = kex(Ks_2, Kp_1)$

server_hello: Kp_2

send_client_ready:

1. Derive $k = kex(Ks_1, Kp_2)$
2. Encrypt $readyc = enc(k, b"ready")$

client_ready: readyc

send_data:

1. Decrypt $b"ready"$ using k
2. Encrypt $mc = enc(k, m)$

data: mc

receive_data:

1. Decrypt m using k

Useful Functions from `cryptobox`

Key Exchange

```python
from cryptobox import generate_assymetric_key, key_exchange

priv1, pub1 = generate_assymetric_key()
priv2, pub2 = generate_assymetric_key()

shared1 = key_exchange(priv1, pub2)
shared2 = key_exchange(priv2, pub1)

assert shared1 == shared2
```

Symmetric Encryption

```python
from cryptobox import generate_symmetric_key, encrypt, decrypt

key = generate_symmetric_key()
message = b"Hello World!"
ciphertext = encrypt(key, message)
plaintext = decrypt(key, ciphertext)

assert plaintext == message
```

What we have so far

Let's test all 5 servers :

```
python3 run.py greenpath.program server1
python3 run.py greenpath.program server2
python3 run.py greenpath.program server3
python3 run.py greenpath.program server4
python3 run.py greenpath.program server5
```

Each of these servers give you some content **but which one to trust?**

Step 2

Certificate Verification and Server's Signature Verifications

send_client_hello:

1. Generate Ks_1, Kp_1

client_hello: Kp_1

send_server_hello:

1. Generate Ks_2, Kp_2
2. Derive $k = kex(Ks_2, Kp_1)$
3. **Sign** $sig_B = sign(Ks_B, Kp_1 + Kp_2 + cert)$
4. **Encrypt** $sigc = enc(k, sign_B + cert)$

server_hello: Kp_2 + sigc

send_client_ready:

1. Derive $k = kex(Ks_1, Kp_2)$
2. **Decrypt** $sign_B + cert$ **using** k
3. **Check** $cert$ **using trusted certs**
4. **Check** $verify(Kp_1 + Kp_2 + cert, sign_B, Kp_B)$
5. Encrypt $readyc = enc(k, b"ready")$

client_ready: readyc

send_data:

1. Decrypt $b"ready"$ using k
2. Encrypt $mc = enc(k, m)$

data: mc

receive_data:

1. Decrypt m using k

Useful Functions from `cryptobox`

Digital Signature

```python
from cryptobox import generate_assymetric_key, sign, verify

priv, pub = generate_assymetric_key()
msg = b"Hello World!"
signature = sign(priv, msg)

assert verify(msg, signature, pub)
```

Certificate

```python
from cryptobox import generate_assymetric_key, create_certificate_authority, certificate_signing_request, generate_c

ca_priv, ca_pub = generate_assymetric_key()
ca_cert = create_certificate_authority(ca_priv)

priv, pub = generate_assymetric_key()
csr = certificate_signing_request(priv, "example.com")

cert = generate_certificate(ca_cert, ca_priv, csr)

print(certificate_info(cert))
assert verify_certificate(cert, [ca_cert])
```

Final Step

Find the Trusted Server among the Imposters

Only one can be trusted

Only one command does not fail, which one?

```
python3 run.py greenpath.program server1
python3 run.py greenpath.program server2
python3 run.py greenpath.program server3
python3 run.py greenpath.program server4
python3 run.py greenpath.program server5
```

[Bonus] can you explain what is wrong with the four other servers? What makes them imposters?