# Data Engineering 2 – Home Assignment 1

**Authors: 2402779 & CEU ID #2**

## Generating keys

First, we need to generate our ceu\_key keypair. We can do this with the following shell script:

ssh-keygen -t rsa -f "$(pwd)/ceu\_key" -N ''

This command generates the key pairs in the current working directory.

## Inspecting the keys and passing the public key to the website visitor

Then, with a Python script, we can write out the keys to the screen:

from pathlib import Path

from Crypto.PublicKey import RSA

#define our key files

pr\_key\_file = "ceu\_key"  #private key

pub\_key\_file = "ceu\_key.pub"  #public key

#checking if the keys really exist

assert Path(pr\_key\_file).exists(), f"Private key file {pr\_key\_file} does not exist!"

assert Path(pub\_key\_file).exists(), f"Public key file {pub\_key\_file} does not exist!"

#loading the private key from the file

with open(pr\_key\_file, "r", encoding="utf8") as key\_file:

    private\_key = RSA.import\_key(key\_file.read())

#printing out the keys

public\_key = private\_key.publickey()

print(f"Public key:\n{public\_key.export\_key().decode('utf-8')}")

print(f"Private key:\n{private\_key.export\_key().decode('utf-8')}")

The two keys that were printed out are the following:

Public key:

-----BEGIN PUBLIC KEY-----

MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAthNpahyGJeCSUOIqHTcU

5ygNGYHK+9uj00PSpHghP7oN2l2NG9uQtJdgnuFD4CjvR86CqKYli2JjEH00YW9W

MywEqJX1YWACXjeFrl/26XKAdkhebydG8s4TdoJXXpv81N4IUZhuXvMlG5qcSGRL

bbvwO7s5B0/zm5WZ/0ZZEXOQWclVxEaM4JRx3YY8ivk4eQ1cRyYlIGO2qKflSQTX

mZBSrMbzMqJaU7QJHso16KqxbJWJumQO0W5VQgtcNiS/BPx8ITHMg9tCt17kfRaA

zWz085UieR+R+0qLfln8t11cNuRbnXCgwUnc5VN4DDk60EG60r5thlh9xIpbr3ZM

Ex8bQtrmG8IMLjFXKbnuZgAANZAfkGjEsLI59HHG9F7jufFurH0wN4vm/r9l6FtK

cPrDP8Th0+8W1xy+wq13zamcnZP8erfmO/IeUe81+6l+Z+1DuGmsOKlGjrHuQe85

7r+GF5/+fJSL1Xq3ia/NHjTiyuqj+XwFz//DLEk/E6nxAgMBAAE=

-----END PUBLIC KEY-----

Private key:

-----BEGIN RSA PRIVATE KEY-----

MIIG5AIBAAKCAYEAthNpahyGJeCSUOIqHTcU5ygNGYHK+9uj00PSpHghP7oN2l2N

G9uQtJdgnuFD4CjvR86CqKYli2JjEH00YW9WMywEqJX1YWACXjeFrl/26XKAdkhe

bydG8s4TdoJXXpv81N4IUZhuXvMlG5qcSGRLbbvwO7s5B0/zm5WZ/0ZZEXOQWclV

xEaM4JRx3YY8ivk4eQ1cRyYlIGO2qKflSQTXmZBSrMbzMqJaU7QJHso16KqxbJWJ

umQO0W5VQgtcNiS/BPx8ITHMg9tCt17kfRaAzWz085UieR+R+0qLfln8t11cNuRb

nXCgwUnc5VN4DDk60EG60r5thlh9xIpbr3ZMEx8bQtrmG8IMLjFXKbnuZgAANZAf

kGjEsLI59HHG9F7jufFurH0wN4vm/r9l6FtKcPrDP8Th0+8W1xy+wq13zamcnZP8

erfmO/IeUe81+6l+Z+1DuGmsOKlGjrHuQe857r+GF5/+fJSL1Xq3ia/NHjTiyuqj

+XwFz//DLEk/E6nxAgMBAAECggGAAI5HT9PrhzABIM2Gk9UVTWjCGutjs0cAHk8d

ewsyMqOH4SAWKa9JTLq0DEB1rt0oEK3SrWsWzBDVG53rsXTQTMrbVi49nr9bvLo3

27KGqvXd4waLKnTkXVrV1b+uNwqyo7GhHopRn23U8seRNidI1o4kz3ZHEoSo/9Ui

mOnX5MAdbT28V9VU5nQcBGnI7c/zEBTL6Cth+Rexppj1kqoyQUvJJg4FKXybiT/S

OkL4ArG/qX4epCglvsyy0cVSu0KQRAwf+g+0i02TUlBncerG8m6iQOx6QIib72QD

huklNkFcNJ/+m1XXwQjfgRvQZzhWVkE7g6tA0NXaFBFQdbGpjOGlVhX+fadK79cM

iwi5kqoacybz2HCJHeBRoehhCXzFnuL/DXM2U2A4FOi5K80TQ2Xzg/B+jMG2eiXp

3xMXSEdQrBu4pO7R1npB0t0KzEDKOIGBmbC6HcSQH3FJzqxvtOukKt9Zc99jlGKR

tWMYklXQRLoKRv8hecFmiAbzdWJhAoHBAMshp+7aC3/ppFvx2b6nsy0UQhISPQyw

PLdeRnsGTA7rRrdjrO/RICbUP37lpiYnu1XubBF1a165OcLeygxqEu6OdXy/zqY5

8LxdRdoWVjzPNLwVgnWpiRjDNXggj2HsvU2+z+C8MzrqHD/vmtL0XbCD+5wn9O6v

Is5vy7JatHuWH4epoX2pCbaQFpqnS/Cht2AWgtCIdr5s38pULzFlyVYZUSkDdxL9

Ik0P+fkeACeUZwRzzYrkt8obmDp03R+d4QKBwQDldtmfcZWi59jsEm5Le/UjIleZ

z7dfiu3Hm8+HPoMihGEMF8jtZkDf0HTHc82mIiyIYkDw4mOTZ8QKk8K+dOzr6dIO

Cs8eOZFlqUDZkHDaZBOHwX+lGMR/Q62MUBVT4bxX/xPTKe68gV1bD8T+HkC0WjD7

bJyhiqXaqWovQhuVjKNJlmtVSkjE0DYN12G3b9LiayDDVAgFomUJvk1in8pBLOBK

PkMupVtqhUv1XECNocRXdxKH1QFvGjOqbeOT7hECgcEAxtXoA2bOcOQsbY/8u6J+

QjcdQYE23y+4DyoqRYxxcP0e2K6p/omvNjL3AGkdTSYBO2lJwYE1m6AmCTl5f9Np

OriaCoXaa1415rxKfuL0gUu2bBGGBVTxjRqwQSlmEM3is7J+25Z9c0Lsai1JWQu4

letrpHx8RhOLN5W5R5mAJ6VYsbv7Bv0rM9gxOCtgq2gxDs6aODQMP/RkzzG+jFT9

UtkvV707lovQQqzL3O1f615ZxMLyRO4DdcOzLC1usd8hAoHBAKg4OhBOp8F2sKtY

U61Y4XxxV8E16xvK8MiN7FUcuewbGj5QTYfkl5i87G+v8MpjcTxGs48kmJVe0/Rh

ILqZY3sLvmd2+yIQWAwsSZN19ZXVGhBDBb3V62/VAKzFpO0KpxXntPPwYMmnGPaC

GAunyA2vtQsNM8KlrzMfUe31S92V7bsr3+H2BGTss0Pwav2cqAA/QxSPTRY8WFAN

SBQOSqr/KCqlfID8zojH0ci9acGrHxJ0A4y61kNJ9ShzSQyQ0QKBwH7faCa5WdDV

K7mQdtvYhtM508sfb3bGfcgflEVIL4J93dINMWIEWUO+F8P3csnJY12xf8WZ6+DZ

HWCexG0RfhpbA8wBHyv/6tSXhxPwIIfK4PyRGFvjcmMxwBSkgNKGAqEkkzhBqGqf

qS5FQdwNKuF6qevkr8VMl9ahcPWqlD7t6EowUMqq/CjtI0puqrkd3K6wLYWAmgaj

/PbN9CunjzZ3D6mMvRMnHQXjoI2qwXPZjTvDrhyy/H4RUfPb20S4+A==

-----END RSA PRIVATE KEY-----

Then, the public key (ceu\_key.pub) was sent over to the website visitor through Teams.

## Encrypting a message with the public key

## Decrypting the message

After receiving the encrypted message (encrypted\_message.bin), we can decrypt it using the private key. This is done through the following Python code:

from pathlib import Path

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP

#define our key files

pr\_key\_file = "ceu\_key"  #private key

pub\_key\_file = "ceu\_key.pub"  #public key

#checking if the keys really exist

assert Path(pr\_key\_file).exists(), f"Private key file {pr\_key\_file} does not exist!"

assert Path(pub\_key\_file).exists(), f"Public key file {pub\_key\_file} does not exist!"

#loading the private key from the file

with open(pr\_key\_file, "r", encoding="utf8") as key\_file:

    private\_key = RSA.import\_key(key\_file.read())

# Decrypting the received message using the private key.

#opening the encrypted message I have received

with open('encrypted\_message.bin', "rb") as f:

    rec\_encrypted\_msg = f.read()

#create a cipher object using the private key for decryption

private\_key\_cipher = PKCS1\_OAEP.new(private\_key)

#decrypt the message using the private key and print out the result

decrypted\_message = private\_key\_cipher.decrypt(rec\_encrypted\_msg)

print(f"Decrypted message: {decrypted\_message.decode('utf-8')}")

#write the decrypted message into a simple txt file

with open('decrypted\_message.txt', "w", encoding = 'utf8') as f:

    f.write(decrypted\_message.decode('utf-8'))

The decryption was successful, and we could read that the message sent was: XXX. The decrypted message is also saved to a file.