Crypto Project 2 - Designing an end-to-end cryptography solution to protect a data application from attacks

Max Christ, Huiyu Yang

A Github Application - Introduction

1. Incentive for developing this application
2. What are we trying to accomplish

Incentive

Let users to control who can access the files they posted on websites such as Github or on other platforms. We hope to give back users the control of who can access what files.

Purpose

Improve the functionality of Github’s services by providing users with hybrid encryption solution for the contents they posted to Github.

Allow users to work on private projects, and ensure that the user’s data is encrypted and secure by their own accord

Implementation - hybrid encryption scheme

Step 1. One “super-user” for a given GitHub project, and they will run code to generate a key to be used for symmetric (private key encryption). They will also post a public key to be used for verifying signatures in the asymmetric scheme.

Step 2. Other users who want to participate in the Github project will run code to generate private/public keys as part of the asymmetric scheme.and will send their pk to admin

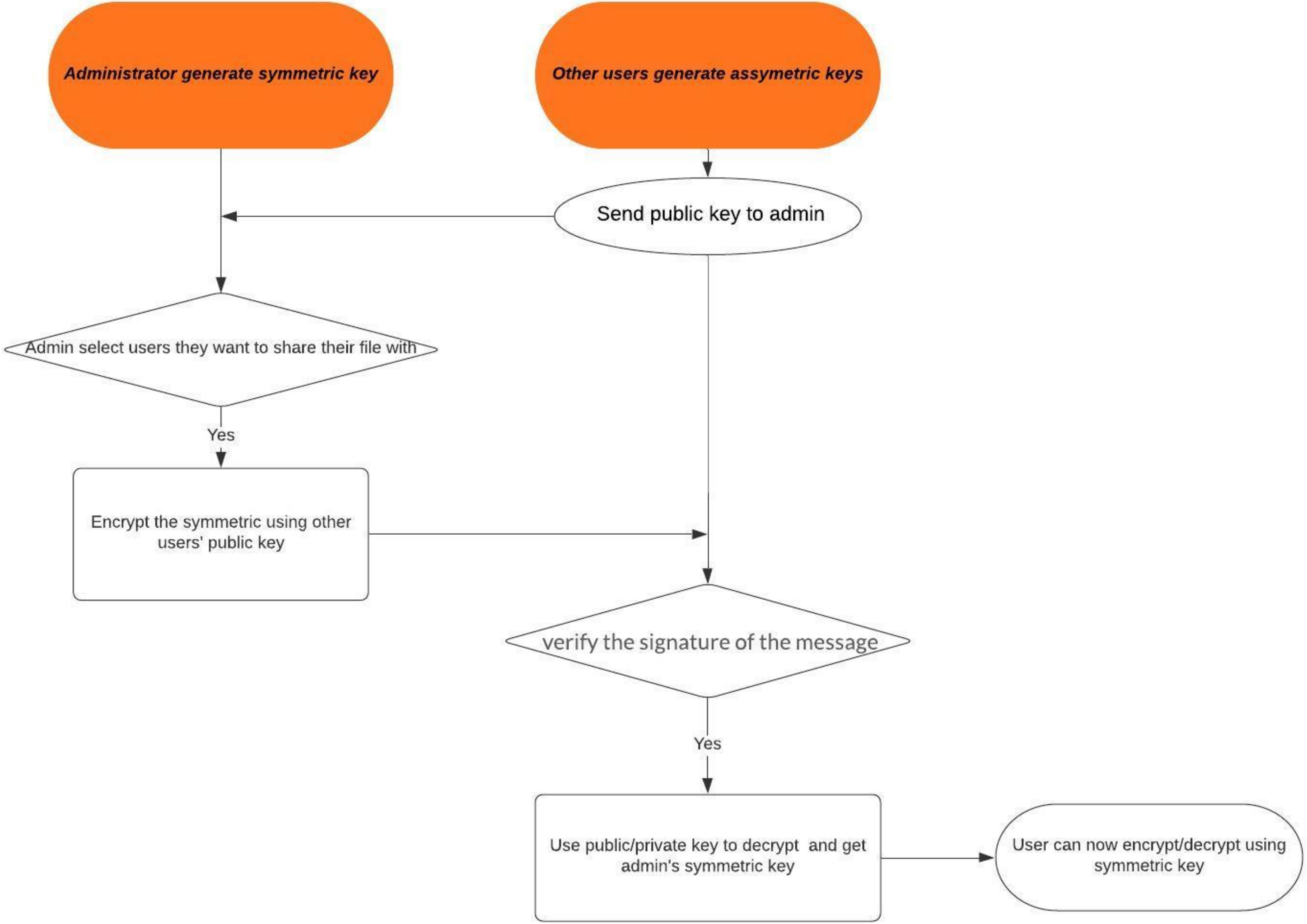
Step 3. The super-user will see the public key files from step 2 for each user. The super-user will then use the public key from a given user to encrypt the symmetric key from step 1, and then write this to a file.

Step 4. At the same time, the super-user will sign the ciphertext from step 3 and write this to a file.

Implementation - hybrid encryption scheme

Step 5. User who originally posted their public key will decrypt the encrypted message from the super-user using their public/private keys. The user will also verify the signature of the message using the public key posted by the super-user

Step 6.User now has access to the common private key for encrypting and decrypting their code files using symmetric encryption.


Administrator generate symmetric key
Other users generate assymetric keys
Send public key to admin
Admin select users they want to share their file with
Yes
Encrypt the symmetric using other users' public key
verify the signature of the message
Yes
Use public/private key to decrypt and get admin's symmetric key
User can now encrypt/decrypt using symmetric key

Attack Analysis

Eavesdropping

- Public repositories on Github
- Transmission to and from Github servers
- Internet service providers could act maliciously

Mitigated by running hybrid encryption, on local machines.

Brute Force Attacks

- Adversary gains access to ciphertexts in a given repository
  - Mitigated using hardness assumptions and large key length

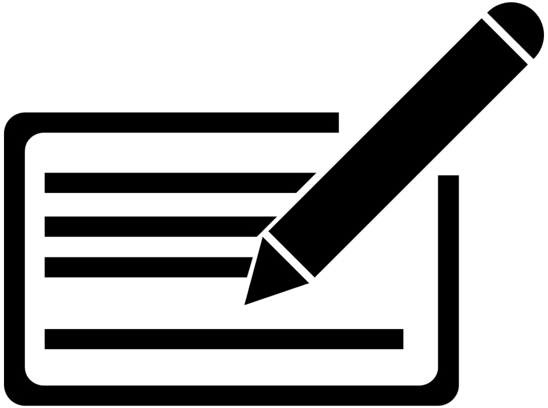
Attack Analysis

“Internal” Attacks

- From Github or a Github employee
- An adversary gains access to Github’s internal systems
- An adversary gains access to the Github repository and modifies/posts files while posing as a vetted user
- An adversary gains access to a user’s private keys (stored on local machines)

Mostly mitigated by running hybrid encryption locally, but not much we can do if keys are exposed.

Attack Analysis



Data Modification/Replay (network level)

- A file can be modified/spoofed when it is being transmitted to/from the Github servers
    - Mitigated by digital signatures and MAC
- Data replay could occur if adversary replays same request as another user to get access to symmetric key
    - Mitigated because adversary would need the private key from the asymmetric scheme

Attack Analysis

Identity Theft

- An attacker could pose as a team member working on the given project and request access to the symmetric key
    - Mitigation: Administrator will need to verify the identity of any team members requesting access to the symmetric encryption key

Encryption Scheme Design

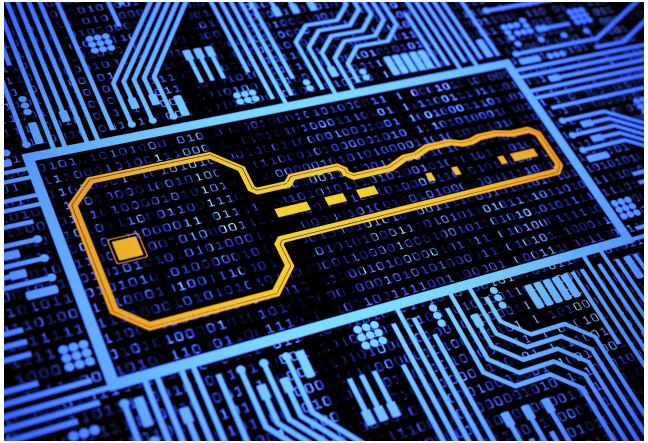
Hybrid Encryption Scheme: To protect against eavesdropping, chosen plaintext, and chosen ciphertext attacks

(IND-CCA) Public key (asymmetric) encryption scheme:

- OAEP/RSA scheme (used in PKCS #1)
- Key Length: 2048 bits
- Hash function: SHA-256

(IND-CCA) Private key (symmetric) encryption scheme:

- AES in CBC mode
- Message authentication: HMAC using SHA-256

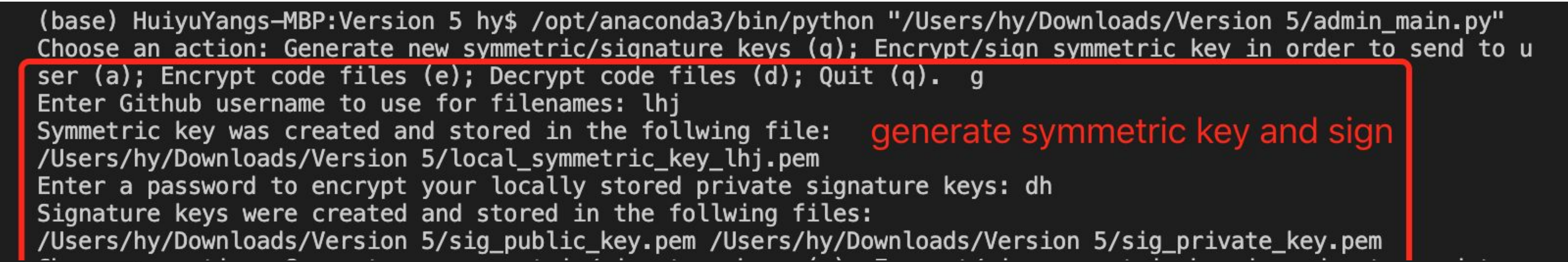


Encryption Scheme Design

To protect against data modification and data originator spoofing of asymmetric encryption scheme:

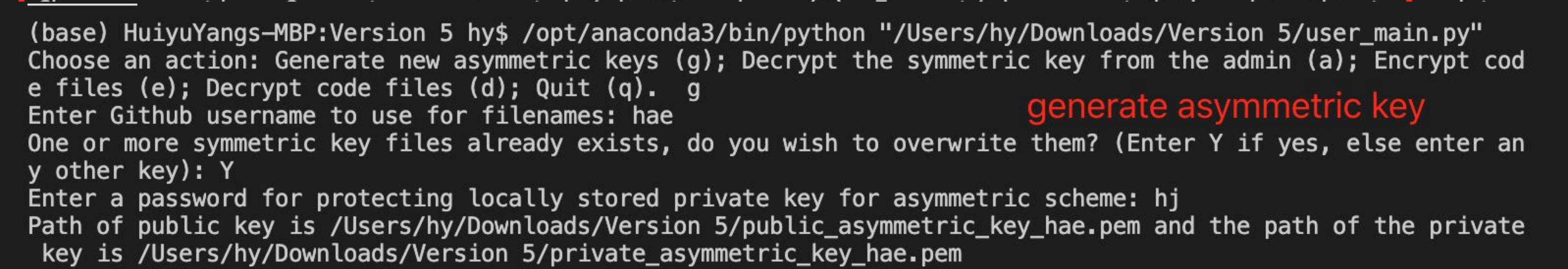
- ECDSA signing algorithm

Graphical Demo



```
(base) HuiyuYangs-MBP:Version 5 hy$ /opt/anaconda3/bin/python "/Users/hy/Downloads/Version 5/admin_main.py"
Choose an action: Generate new symmetric/signature keys (g); Encrypt/sign symmetric key in order to send to u
ser (a); Encrypt code files (e); Decrypt code files (d); Quit (q).  g
Enter Github username to use for filenames: lhj
Symmetric key was created and stored in the follwing file:
/Users/hy/Downloads/Version 5/local_symmetric_key_lhj.pem
Enter a password to encrypt your locally stored private signature keys: dh
Signature keys were created and stored in the follwing files:
/Users/hy/Downloads/Version 5/sig_public_key.pem /Users/hy/Downloads/Version 5/sig_private_key.pem
```

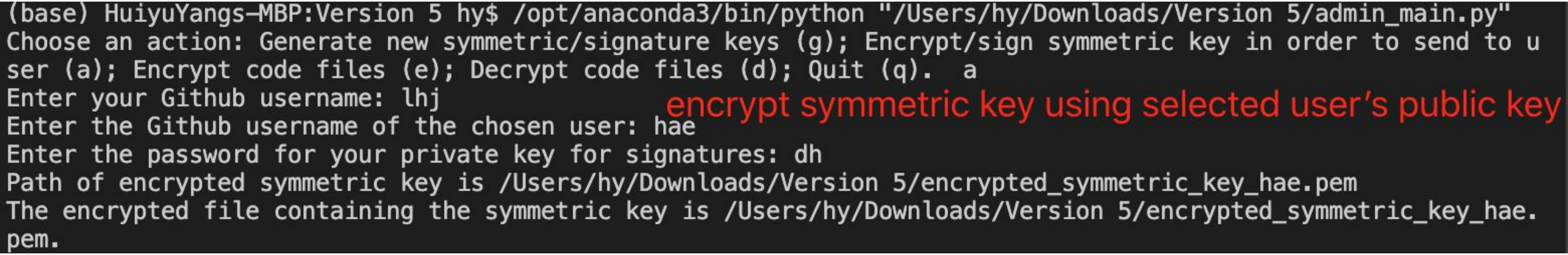
generate symmetric key and sign



```
(base) HuiyuYangs-MBP:Version 5 hy$ /opt/anaconda3/bin/python "/Users/hy/Downloads/Version 5/user_main.py"
Choose an action: Generate new asymmetric keys (g); Decrypt the symmetric key from the admin (a); Encrypt cod
e files (e); Decrypt code files (d); Quit (q).  g
Enter Github username to use for filenames: hae
One or more symmetric key files already exists, do you wish to overwrite them? (Enter Y if yes, else enter an
y other key): Y
Enter a password for protecting locally stored private key for asymmetric scheme: hj
Path of public key is /Users/hy/Downloads/Version 5/public_asymmetric_key_hae.pem and the path of the private
 key is /Users/hy/Downloads/Version 5/private_asymmetric_key_hae.pem
```

generate asymmetric key

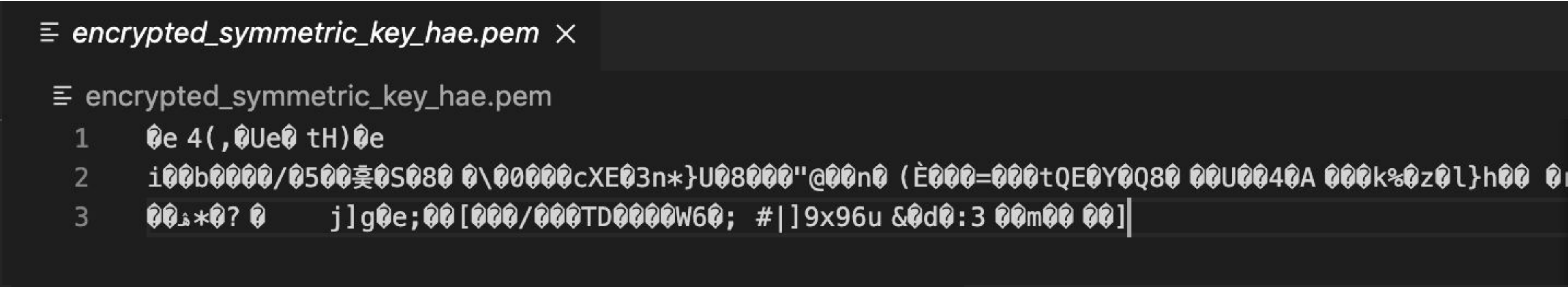
Admin using user’s pk to encrypt the symmetric key



```
(base) HuiyuYangs-MBP:Version 5 hy$ /opt/anaconda3/bin/python "/Users/hy/Downloads/Version 5/admin_main.py"
Choose an action: Generate new symmetric/signature keys (g); Encrypt/sign symmetric key in order to send to u
ser (a); Encrypt code files (e); Decrypt code files (d); Quit (q).  a
Enter your Github username: lhj
Enter the Github username of the chosen user: hae
Enter the password for your private key for signatures: dh
Path of encrypted symmetric key is /Users/hy/Downloads/Version 5/encrypted_symmetric_key_hae.pem
The encrypted file containing the symmetric key is /Users/hy/Downloads/Version 5/encrypted_symmetric_key_hae.
pem.
```

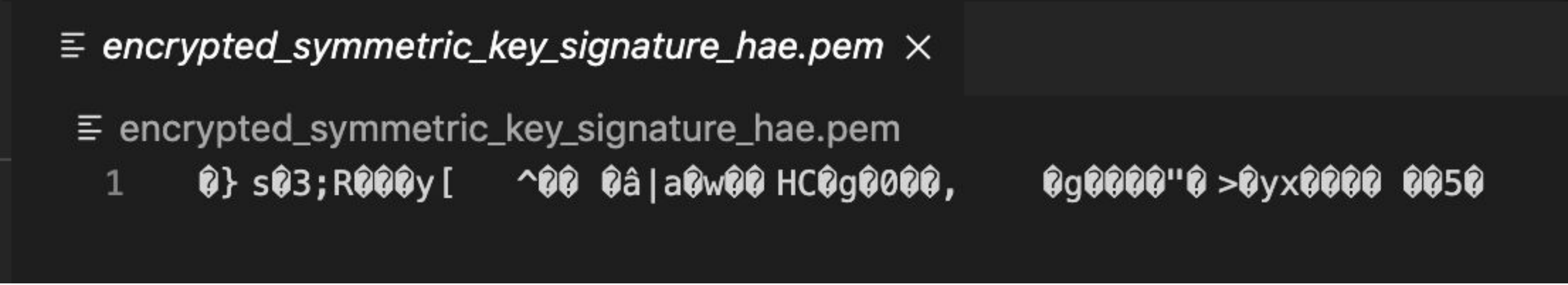
encrypt symmetric key using selected user’s public key

Encrypted key file



encrypted_symmetric_key_hae.pem ×
encrypted_symmetric_key_hae.pem

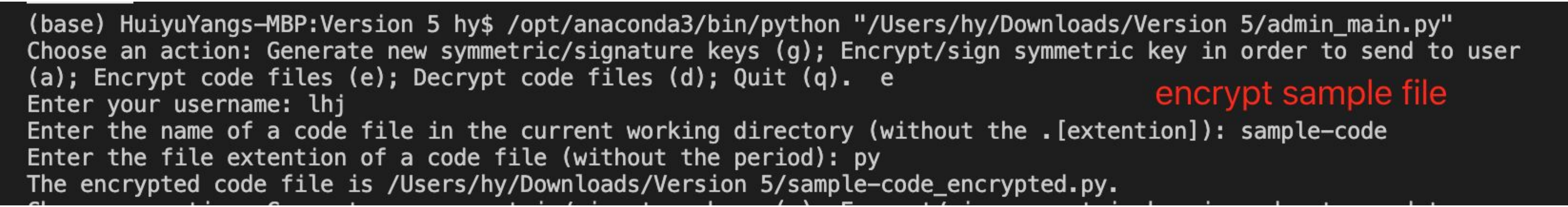
```
�e 4(,�Ue� tH)�e
i��b����/�5��橐�S�8� �\�0���cXE�3n*}U�8���"@��n� (È���=���tQE�Y�Q8� ��U��4�A ���k%�z�l}h�� �
��ۿ*�? �     j]g�e;��[���/���TD����W6�; #|]9x96u &�d�:3 ��m�� ��]
```



encrypted_symmetric_key_signature_hae.pem ×
encrypted_symmetric_key_signature_hae.pem

```
�} s�3;R���y[    ^��  �â|a�w�� HC�g�0��,     �g����"� >�yx����  ��5�
```

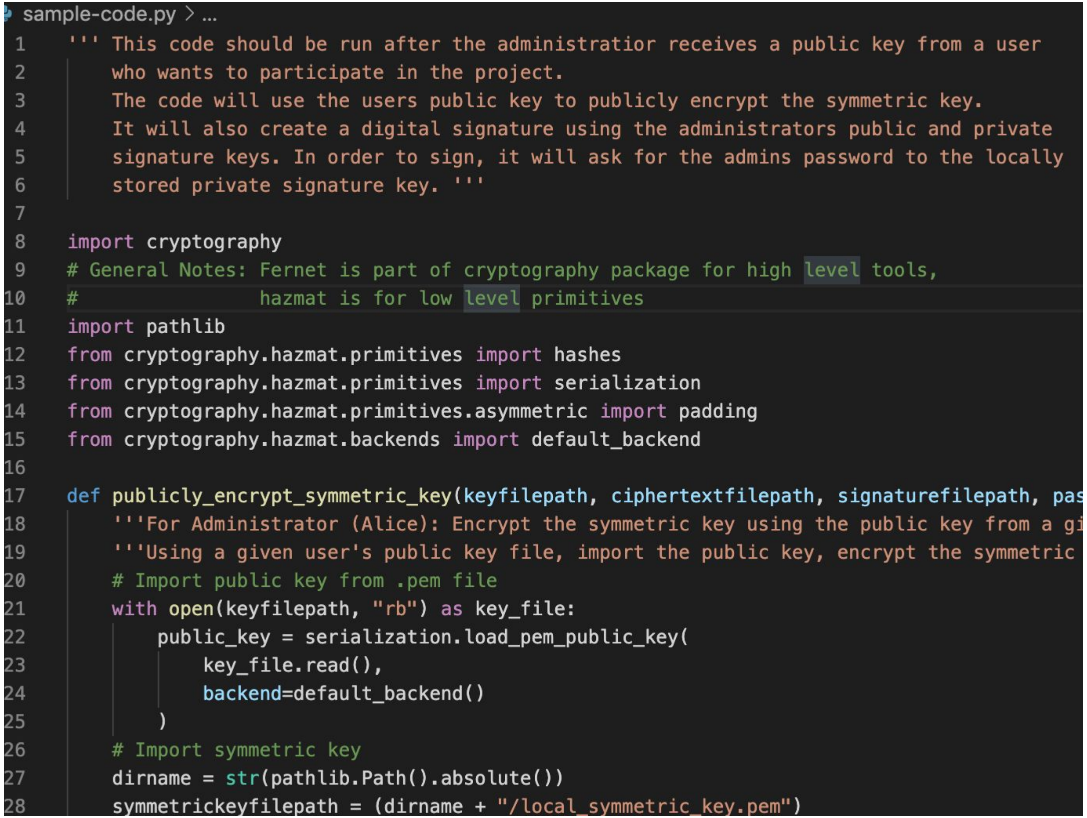
Encrypt sample file



```
(base) HuiyuYangs-MBP:Version 5 hy$ /opt/anaconda3/bin/python "/Users/hy/Downloads/Version 5/admin_main.py"
Choose an action: Generate new symmetric/signature keys (g); Encrypt/sign symmetric key in order to send to user
(a); Encrypt code files (e); Decrypt code files (d); Quit (q).  e
Enter your username: lhj
Enter the name of a code file in the current working directory (without the .[extention]): sample-code
Enter the file extention of a code file (without the period): py
The encrypted code file is /Users/hy/Downloads/Version 5/sample-code_encrypted.py.
```

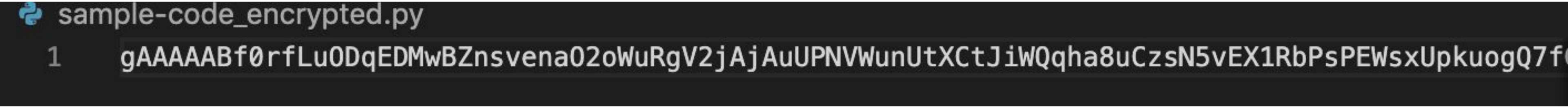
encrypt sample file

Original file (before encryption)



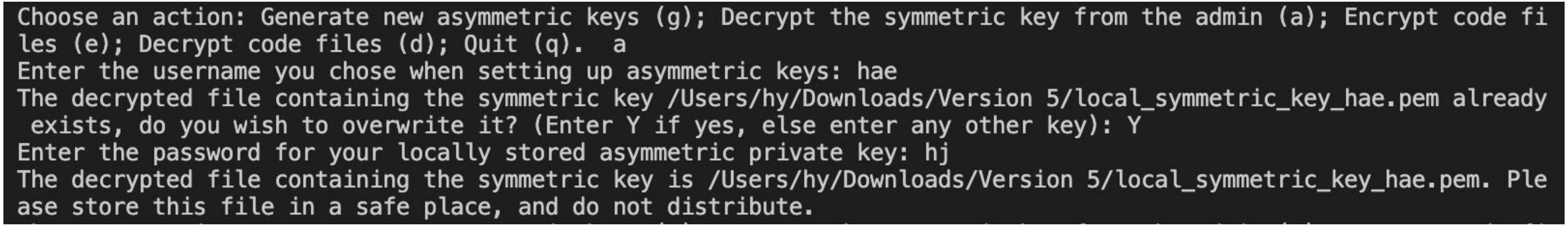
```python
''' This code should be run after the administratior receives a public key from a user
    who wants to participate in the project.
    The code will use the users public key to publicly encrypt the symmetric key.
    It will also create a digital signature using the administrators public and private
    signature keys. In order to sign, it will ask for the admins password to the locally
    stored private signature key. '''

import cryptography
# General Notes: Fernet is part of cryptography package for high level tools,
#                hazmat is for low level primitives
import pathlib
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.backends import default_backend

def publicly_encrypt_symmetric_key(keyfilepath, ciphertextfilepath, signaturefilepath, pas
    '''For Administrator (Alice): Encrypt the symmetric key using the public key from a gi
    '''Using a given user's public key file, import the public key, encrypt the symmetric
    # Import public key from .pem file
    with open(keyfilepath, "rb") as key_file:
        public_key = serialization.load_pem_public_key(
            key_file.read(),
            backend=default_backend()
        )
    # Import symmetric key
    dirname = str(pathlib.Path().absolute())
    symmetrickeyfilepath = (dirname + "/local_symmetric_key.pem")
```

Encrypted file



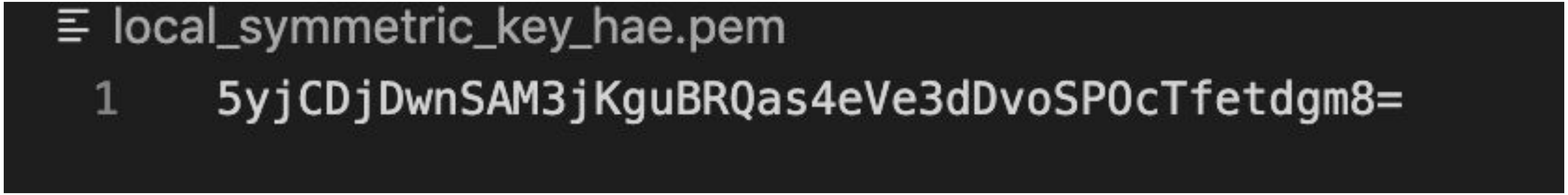
```
gAAAAABf0rfLuODqEDMwBZnsvenaO2oWuRgV2jAjAuUPNVWunUtXCtJiWQqha8uCzsN5vEX1RbPsPEWsxUpkuogQ7f
```

Decrypt for symmetric key



```
Choose an action: Generate new asymmetric keys (g); Decrypt the symmetric key from the admin (a); Encrypt code fi
les (e); Decrypt code files (d); Quit (q).  a
Enter the username you chose when setting up asymmetric keys: hae
The decrypted file containing the symmetric key /Users/hy/Downloads/Version 5/local_symmetric_key_hae.pem already
 exists, do you wish to overwrite it? (Enter Y if yes, else enter any other key): Y
Enter the password for your locally stored asymmetric private key: hj
The decrypted file containing the symmetric key is /Users/hy/Downloads/Version 5/local_symmetric_key_hae.pem. Ple
ase store this file in a safe place, and do not distribute.
```

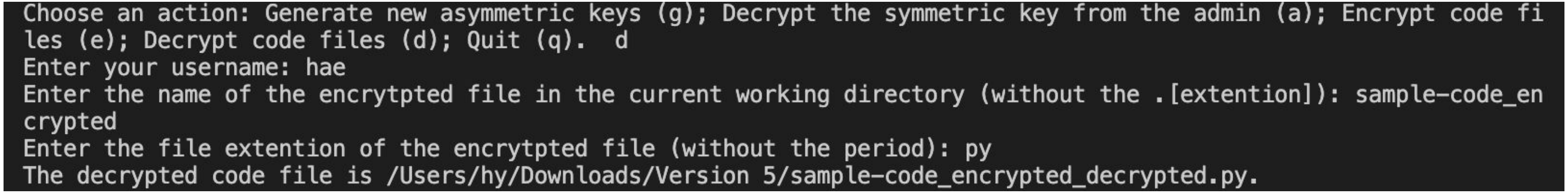
Decrypted key



local_symmetric_key_hae.pem

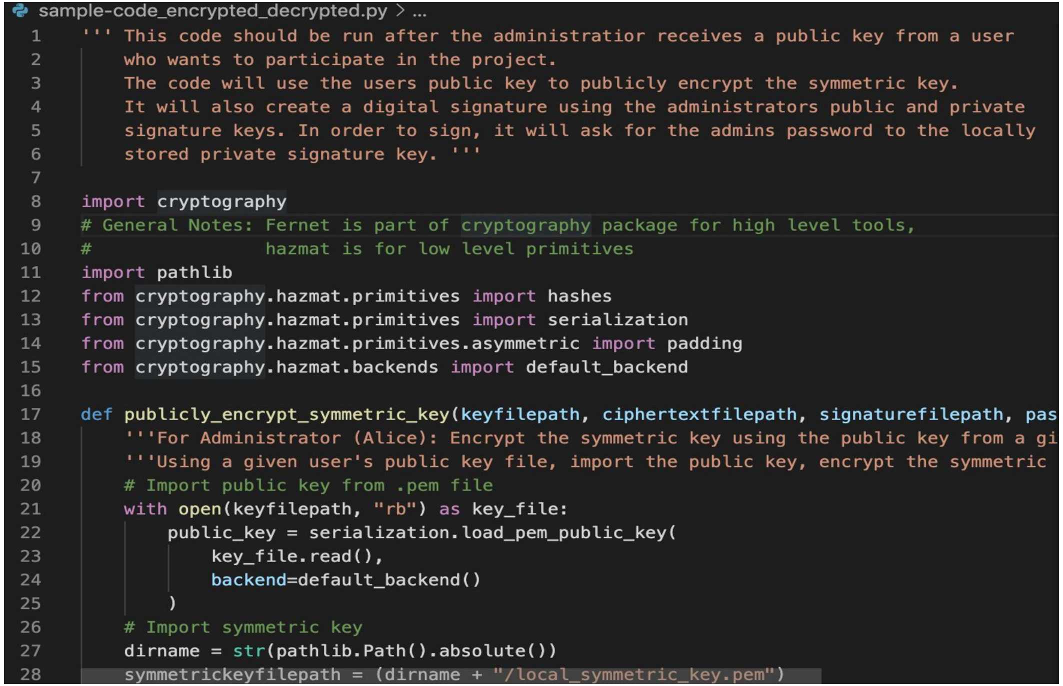
```
5yjCDjDwnSAM3jKguBRQas4eVe3dDvoSPOcTfetdgm8=
```

Decrypt code file



```
Choose an action: Generate new asymmetric keys (g); Decrypt the symmetric key from the admin (a); Encrypt code fi
les (e); Decrypt code files (d); Quit (q).  d
Enter your username: hae
Enter the name of the encrytpted file in the current working directory (without the .[extention]): sample-code_en
crypted
Enter the file extention of the encrytpted file (without the period): py
The decrypted code file is /Users/hy/Downloads/Version 5/sample-code_encrypted_decrypted.py.
```

Decrypted file



sample-code_encrypted_decrypted.py > ...

```python
''' This code should be run after the administratior receives a public key from a user
    who wants to participate in the project.
    The code will use the users public key to publicly encrypt the symmetric key.
    It will also create a digital signature using the administrators public and private
    signature keys. In order to sign, it will ask for the admins password to the locally
    stored private signature key. '''

import cryptography
# General Notes: Fernet is part of cryptography package for high level tools,
#                hazmat is for low level primitives
import pathlib
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.backends import default_backend

def publicly_encrypt_symmetric_key(keyfilepath, ciphertextfilepath, signaturefilepath, pas
    '''For Administrator (Alice): Encrypt the symmetric key using the public key from a gi
    '''Using a given user's public key file, import the public key, encrypt the symmetric
    # Import public key from .pem file
    with open(keyfilepath, "rb") as key_file:
        public_key = serialization.load_pem_public_key(
            key_file.read(),
            backend=default_backend()
        )
    # Import symmetric key
    dirname = str(pathlib.Path().absolute())
    symmetrickeyfilepath = (dirname + "/local_symmetric_key.pem")
```