1. Setting up the movie server. The server will serve up the AES encrypted file, receive the client's public key and encrypt both the file hash and AES key with the public key.

```
root@caleb-Precision-M4500:/home/caleb/datasecurity/confidentiality# python ./Moviemain1.py -f movie.mp4
Sha-256: eba19cadcfe84ad757ba4d494853abcf8a4abbb169e1e245af927f0b79352e73
Block size: 128-bit
Key size: 128-bit
b64 Key: ntJEvsF9zbu3h4M1dCvlSw==

Encrypted data size: 29.61 MB
The server is ready to receive connections
[*] Receiving projector public RSA key
[*] Projector RSA Public Key =>  <_RSAobj @0xb6fe52ac n(2048),e>
Size of Payload = 29611974 Bytes

[*] SECURE DESTROY IN PROGRESS
[*] COMPLETE!
```

2. The client process will give the server the client's public key, then accept the payload from the server.

```
python ./Projectormain.py -i 127.0.0.1 -o movie.recv.mp4
```

```
[+] Existing RSA secure store file found
[+] Pulling RSA object

<_RSAobj @0xb6f92c2c n(2048),e,d,p,q,u,private>
[*] Sending RSA public key to server =>  <_RSAobj @0xb6f92b4c n(2048),e>
[*] file transfer complete!


++++++ BEGIN ENCRYPTED KEY +++++++

[illegible]

++++++ END ENCRYPTED KEY +++++++

DECRYPTED b64 KEY =>  ntJEvsF9zbu3h4M1dCvlSw==


++++++ BEGIN ENCRYPTED SHA-256 +++++++

[illegible]

++++++ END ENCRYPTED SHA-256 +++++++

DECRYPTED SHA-256 =>  eba19cadcfe84ad757ba4d494853abcf8a4abbb169e1e245af927f0b79352e73


[*] hashes matched, writing data to output file

[*] SECURE DESTROY IN PROGRESS
[*] COMPLETE!
```