

The following project was done in Pycharm using the coding language Python. The project utilizes RSA encryption and a Caesar cipher to encrypt a message the user enters.

The code prints out the public and private keys once they are created. This was to ensure that it was working and was displayed. For the Caesar cipher, I had it pick a random number that it would be shifted by. -- This is also printed out to see what number it picked.

- After the RSA keys are created and the Caesar cipher number is picked it asks for the user's message.
- Once the user enters the message, it will be encrypted first by the Caesar cipher and then by the public key.
- To decrypt the message the private key is used then the Caesar shift is reversed to get the original message.

Ecption = 1st Caesar cipher, 2nd public key

Decryption = 1st private key, 2nd reversed Caesar cipher

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.asymmetric import padding
import random

# RSA key pairs
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048, )
public_key = private_key.public_key()

# Serialize and print public key
public_key_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)
print(f"Public Key:\n{public_key_pem.decode() }")
```

```

# Serialize and print private key
private_key_pem = private_key.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.PKCS8,
    encryption_algorithm=serialization.NoEncryption()
)
print(f"Private Key:\n{private_key_pem.decode()}")


# Caesar cipher
def caesar_cipher(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            ascii_offset = 65 if char.isupper() else 97
            shifted_char = chr((ord(char) - ascii_offset + shift) % 26 +
ascii_offset)
            result += shifted_char
        else:
            result += char
    return result


# Encrypt a message
def encrypt_message(message, public_key, caesar_shift):
    caesar_encrypted_message = caesar_cipher(message, caesar_shift)
    cipher_text = public_key.encrypt(
        caesar_encrypted_message.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return cipher_text, caesar_shift


# Decrypt the message
def decrypt_message(cipher_text, private_key, caesar_shift):
    plain_text = private_key.decrypt(
        cipher_text,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    decrypted_message = caesar_cipher(plain_text.decode(), -caesar_shift)
    return decrypted_message

```

```

# User input / message
message = input("Enter the message to encrypt: ")

# Encrypt the message
cipher_text, caesar_shift = encrypt_message(message, public_key,
random.randint(1, 25))
print(f"Caesar cipher shift: {caesar_shift}")
print(f"Encrypted message: {cipher_text}")

# Decrypt the message
decrypted_message = decrypt_message(cipher_text, private_key, caesar_shift)
print(f"Decrypted message: {decrypted_message}")

```

```

C:\Users\chery\PycharmProjects\pythonProject8\venv\scripts\python.exe t:\Users\chery\PycharmProjects\pythonProject8\main.py
Public Key:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAACBkCwggsIAgEAAiBAAQcAjf4yz/+0fx
6gqLozZhnvy/KF0mFPNqHks8LKF5Clwf00jrvDL7bv0QIAo3Xu0QjkKQK
F2HLe1dtt9yXL0gfSLN0/o/wy9n0fRLMK21SEYy3LV53tA8JaqULUN/kh8y
FEj+Ig+vvn0xhTUDbYBC6LahVAPFRNzParr5Rtb0Qw/XVmQZb7vhPjvxkP
N/bMrnv22C80/x0Nu1zgP04nStU1c1n1zyxEKpFg+NcIV2z7P8VENWF+2Rx1a
Avf8dJYxnTDLSRv10m0s8h/t1zrQz2BPyvDitoitzz6ixVmJj5oHns+c68R
FFc>SA0fAgAEEggEAu2eABeX05fPKJNx3KzqfUrWuK1n7xyXKVYd7zL
qx+2apwzzMyLo5d0u7A16jqcq72zrLcyuR77ePmruduU.hkggNSQQNu0
Sqj1TzGTP0dxtVekabx7d0SS5qQVFn0j5ae0rn3oghOpseWS051H1P0aq
pr1vpjQ9Am#71010bJb/+xEz5r+FoxE1Uewv111MkmmbuJglnvhv0e014oURRpz
JR+u148CAR9DR775rSAXJL2fKEShqg5wcwFZapRvAUUVu2uCKWwy7A51Ux
tw51p01nyftVA1Vnd1BcB51j6wPucIo2MEU19q0vKbRq0DQlQz25L17Dxs4/SwB
jFc/xWWNAppjK/w0uMUAB.j1oywsvqfEfKxz2z47082SD2K3yel0M2LSf+zFaqTA
Y81Ln0QuvEokqQsk3I+730fVjAoMFTl0VQ1C2+4j1011q+Ate4591Ccgl0SSgH6
Kfp1Yf58141XY56HDv0k1UWKGqgDCPz75Ef2eayf51Yz5e/x10/Anx1ILxW]
Q1fT+*nqxnHLeed1k9/tgc5PLSkzebyL/hyn3jsym7qEWVKLRNpjCj9P
7awMjg9+qed0012xjFKUBmxQqeVaplanoCHNL1J3HsDF1Ns04wcgjxZh051z3
varv9IAccQkBgAJDjZfwmz6tuubJFqn16/RehZQneFuw5dPll9+TLVRD9U10
08171laysf4w7QDn9bqvTm2cNLqkpuhnxqYEAq9gsasJSP6P/J1b09n7/3V4z
pJOHx0Dy7shhKKRZzV0Z4dbvRnCa4BzV42BFktxVxV0A5jbtyt/x0a4T46W
+xx86dJz37aLnxEhr3qLTzbysu0+g5bn1cI3SY2LwB1rvtoJ+FMu06UKY6eb
U12erfVnwLn0xvq17e1mtf76l026kpsg9w5JXHnvvtjViMAK7veE18v0Peif/74Kq
CZHMSyPrzygxlaKPPw51Lb97Az171g3V3cz2ckgYArVwBmcRn1M+41YrxF
477nv0K6zND1kRslgfU5in3sNF4rXz1j6XnPqE/jch33d1fqlVL0990g1unxb
wy1z6GB/xHztzI6aCtf639qaI059HYQ#WUSzKTV1VSKgyJTKn5BeqvxxkjR67YY

```

```

-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwgSjAgEAAoIBAQCaIjf4yz/+00fX
6gqluSZhHnyv/KFHOMFPHqIhkS8JKF5C1wF00jrvDL7bpVoOK1A0Z3XUKOJkkaQK
F2H1e1dtY9vXL0gRfSLNLo/hwv9r0FRLMK21SEYv3LvV53tA8JajqwULUN/Khv8y
FEj+lg+vvnoxhTUNDByBC6LaHvAPFRNzhParr3Rtb0QwL/XVmQZbr7vHfPjkXfP
N/bMRru22CBO/x6NU1ZgPO4nStUU1cn1ZyjXeKPdFq+NolV2a7P8VENWP+ZRxf1a
AvfsdBJYxnTDLsRVi0Dm8s8h/tlrzQZaE2bPyvDHtoiitzGiIvmJjo5HNsc+G8R
FfCrSADfAgMBAECggEAu2eAkBeXOO5rPKJ3hx3kzqFu7wuBKIn7xYXKVYdfzL
qx+Q2apWzzXMyLOy56bUH7zAl6jcqq7ZzrkICYubRf7ePmRud1UjhkggwNSQQMuT
SqjITzGTPDqbXTvEKaBax7d+DSSsq5VFm6jSaJezorm3bgh0pLseWSb5I1H1P0Q
pr1ypJqW4m+7IOiGbJb/+xEz3r+FozEUewhVhiMknmBuJgNvvheGb1A4ouRRpz
JR+fi4H8cAR9DR7T5rsRAXbJLzFkEshgo5zwcuFzaPRVgAUUUV2uCWKwyu7A5iUx
tWsIGpiYhftVAV1MdIBcB5lj6HwPuclozMWEUh9qvQKBgQDLqU2SLI7Dxks4/Sw
jFcVxQWNAOpjK/wduFMUA8Jg1oySvQdEQkXzz4Zq82sDzK3y6eUdMzLSF+zFagIA
Y81LnOQUuEokQ5k3+73OfVjAoMFTploVQIC2+4j0liQ+At4591CcgFloS5gHG
KfpIYfS841ax3GaHDVaukTUWwKBgQDCPzf3EH2eayF5iYzSeJxI0/An4xiLxWj
QiFT+3nQxhhLeedFlk9/tgpc5PLbSkZeBYsL/hyrh3jsYmGq7Ew5VKiRKNpjC9P
7aWFjg9+QeodOb12xfKUBwmXOqeVApLaNoChNLIJ3HDsFiNso4WcgjXwZh05Lz3

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

Enter the message to encrypt: brandon pearson
caesar cipher shift: 7
Encrypted message: b'`x840\xc1\xe7v\x0f:\x17T\x88\xab\xf6\x92u\xf9\x93n\x6\xc7\xca\x7f\xacK\x0b\xefz\xed\xc0\x01\xac\xd8_j\xc1\xea\xfeM\x05*\x08\xaa\xf3\x93\xbc\xfd\xca\x81\xc1\xe4\xaa\x86\xe2b\xfd\x08\xd5=`
Decrypted message: brandon pearson

Process finished with exit code 0

```

## FROM IMAGE ABOVE

### Public Key:

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

```

MIIBljANBgkqhkiG9w0BAQEFAOCQ8AMIIBCgKCAQEAmoiX+Ms//jtH1+oKpbkm
YR58r/yhRzphTx6iB5EvCSheQtcBdNI67wy+26VaDitQNGd11CjiZJGkChdh9XtX
bWPb1y9IEX0izS6P4cL/a9BUSzCttUhGL9y71ed7QPCWo6sFC1Dfyob/MhRI/iIP
r756MYU1DXQcgQui2h7wDxUTc4T2q690bW9EMC/11ZkGW6+7x3z475F3zzf2zEa7
ttggTv8ejVNWDzuJ0rVFNXJ9Wco13ij3RavjaCFdmuz/FRDVj/mUcX9WgL37HQs
WMZ0wy7EVYtA5vLPlf7Zc60GWhNmz8rxw7alorcxolyL5iY6ORzbHPhvERXwq0gA
3wIDAQAB
-----END PUBLIC KEY-----

```

### Private Key:

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

```

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwgSjAgEAAoIBAQCaIjf4yz/+00fX
6gqluSZhHnyv/KFHOMFPHqIhkS8JKF5C1wF00jrvDL7bpVoOK1A0Z3XUKOJkkaQK
F2H1e1dtY9vXL0gRfSLNLo/hwv9r0FRLMK21SEYv3LvV53tA8JajqwULUN/Khv8y
FEj+lg+vvnoxhTUNDByBC6LaHvAPFRNzhParr3Rtb0QwL/XVmQZbr7vHfPjkXfP
N/bMRru22CBO/x6NU1ZgPO4nStUU1cn1ZyjXeKPdFq+NolV2a7P8VENWP+ZRxf1a
AvfsdBJYxnTDLsRVi0Dm8s8h/tlrzQZaE2bPyvDHtoiitzGiIvmJjo5HNsc+G8R
FfCrSADfAgMBAECggEAu2eAkBeXOO5rPKJ3hx3kzqFu7wuBKIn7xYXKVYdfzL
qx+Q2apWzzXMyLOy56bUH7zAl6jcqq7ZzrkICYubRf7ePmRud1UjhkggwNSQQMuT
SqjITzGTPDqbXTvEKaBax7d+DSSsq5VFm6jSaJezorm3bgh0pLseWSb5I1H1P0Q
pr1ypJqW4m+7IOiGbJb/+xEz3r+FozEUewhVhiMknmBuJgNvvheGb1A4ouRRpz
JR+fi4H8cAR9DR7T5rsRAXbJLzFkEshgo5zwcuFzaPRVgAUUUV2uCWKwyu7A5iUx
tWsIGpiYhftVAV1MdIBcB5lj6HwPuclozMWEUh9qvQKBgQDLqU2SLI7Dxks4/Sw
jFcVxQWNAOpjK/wduFMUA8Jg1oySvQdEQkXzz4Zq82sDzK3y6eUdMzLSF+zFagIA
Y81LnOQUuEokQ5k3+73OfVjAoMFTploVQIC2+4j0liQ+At4591CcgFloS5gHG
KfpIYfS841ax3GaHDVaukTUWwKBgQDCPzf3EH2eayF5iYzSeJxI0/An4xiLxWj
QiFT+3nQxhhLeedFlk9/tgpc5PLbSkZeBYsL/hyrh3jsYmGq7Ew5VKiRKNpjC9P
7aWFjg9+QeodOb12xfKUBwmXOqeVApLaNoChNLIJ3HDsFiNso4WcgjXwZh05Lz3

```

varl9AczQKBgAJDJzFwgmz6TuubJFqn1G/ReHZQhEoFuw85dPLL9+TLfVRD9Ui0  
08IZIAysF4w7QdNo9bqVTwM2cNLgkpUehqXoYE A6q9gsaJSGPrJ/ibO9kn7/3V4z  
pJOkxODv7sEhhKKRZ2vOZ4DbvrRnCa4B2V428FklzXVxDVoA5jbTyt/xAoGATJ4W  
+xK8GdJz37aLnzEHr3qLTZBbysuXo0+gSbn1cl3SY2LwABirvtoU+FMuH6UKY Geb  
Ut2mfVnwLn0XvQ17e1mTK76LdZGkpSg9k5JXNh vVtjViMAk7VEel8vDPcif/74Kq  
CzhM3ypRyzgxlaKKPPw51LB97A2VI7riQ3VCzzkCgYEArVwrBMcVRr1M+4IYwfxF  
4f7nv0kGzND1kRsLgf uE3ina3SNF4rXzf1jGXNPqE/jck33diFqIVLO990g1unxb  
vyiZSGB/XHZtzl6aClf639QaIO59HYQmWU5zKTYiVSKGyJTKn5BcqvxkxMjR6TYY  
5+FqG7wAANHhtzOOPAVMGGc=  
----END PRIVATE KEY----

**Enter the message to encrypt: brandon pearson**

**Caesar cipher shift: 7**

**Encrypted message:**

b'\x84r\xc1\xe7v\x0f:\x17T\x88\xab\xf6\x92ou\xf9\x93n\trG\xc7\xca]s\x7f\xacK\x0b\xefz\xed\xc6\x01\xac\xd8\_jE\xc1\xea\xfeM\x05\*\x08\xaeel\x8f\xc1\xf3\x93\xbc\xfd\xca\x81\xc1\xe4>\xaal\x86\xe2b\xfdQG0\xd5~\xdal\x01\x8b\x90\x91\xf4\xe6\xfb\x14\xf9\xc0\xaf\xa3%\ixa4b\x11\xa0iY\x8e\xe5\xe8\x0f\x02\xff2\x03\xde O+\xa9\xec\x04\xfa\xfe\xcc\x1cKt\xb8B\xc8Z\x0fg[W\x86\x12\xe7R\x83K=CF\xb6)\xc7\xf9\xe0\x1e\xeeX9}\xf4\x85\xa2\x9du\x90\x87\xfe\xad\xab\x03{|\x87]]\_3a\x1a\x0c\x02\xd\x8d==J\x16\n\xeb\xfe\xe3\x9c(\x9f\xcds\xd7\x8f\x9a+\.\xc2\x1b\xf8\xfb@\\xa5\x18ax^T\xf5\xb8\x07\xfa\xce\xaa\xbe\xc0\x1a\x9b\x84V\xef\x99\x0e\x8f\xf8\xcb\xd9s\xff\xb7\xe2\xb7\xce\xbf\xd0\x13\xd9\x821e\x01\xfd\x892\x83\x1e\xb3\xc4\xe5\xe7\x91\xc9z\xb9\xdb\tF\x08\r\xc4\x19\x19\x0b'

**Decrypted message: brandon pearson**