VIT
Vellore Institute of Technology

School of Computer Science and Engineering

J Component report

Programme	: Integrated Mtech in Cse with Specialisation in Business Analytics
Course Title	: Network Security and Cryptography Fundamentals
Course Code	: CSE1029
Slot	: B1

Title: Total Encryption

Team Members: Yash Shah 20MIA1028
J Shree Nikhila 20MIA1023
Atul Patel 20MIA1134

Faculty: Dr Anita X

Sign: [illegible]
Date : 21.11.2022

Total Encryption with AES

Reg No	Name
20MIA1028	Yash Shah
20MIA1134	Atul Patel
20MIA1023	J Shree Nikhila

Abstarct :

Today almost all digital services like internet communication, medical and military imaging systems, multimedia system requires reliable security in storage and transmission of digital images. Due to faster growth in multimedia technology, internet and cell phones, there is a need for security in digital images. Therefore there is a need for image encryption techniques. In this system we use AES (Advanced Encryption Technique) . Such Encryption technique helps to avoid intrusion attacks.

E-mail plays a very crucial role in delivering confidential messages. These messages needs to be delivered to the intended recipient without being prone to any kind of attacks . Hence, a secure computing environment would not be complete without consideration of encryption technology. To obtain this, our system, data encryption using AES, provides a very safe and secure approach to protect all the messages that are been transferred over the net. The main advantage of this system is the use of hybrid encryption ,which makes it even secure .

Keywords : *AES, Encryption , Crypto Cipher , Cryptography,*

Introduction :

The Advanced Encryption Standard (AES), also known by its original name Rijndael is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

AES is a variant of the Rijndael block cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

Motivation :

Encryption is crucial in lots of Wireless Sensor Network (WSN) applications. Several encryption frameworks exist that are mainly based on software program algorithms.

However, almost each trending radio transceiver chip is ready with an included hardware encryption engine.This set of rules has an personal specific shape to encrypt and decrypt sensitive records and is implemented in hardware and software program all around the world.

It is extremely tough for hackers to get the actual records while encrypting with the aid of using AES set of rules. AES has the capacity to cope with 3 different key sizes together with AES 128, 192 and 256 bit and each of this ciphers has 128 bit block size. In this project we will offer an outline of AES set of rules and give an explanation for numerous essential functions of this set of rules in information and demonstrating a few preceding researches which have accomplished on it with evaluating to different algorithms such as DES, 3DES, Blowfish etc.

Challenges :

- It uses too simple algebraic structure.
- Every block is always encrypted in the same way.
- Hard to implement with software.
- AES in counter mode is complex to implement in software taking both performance and security into considerations.
- Cost - computational due to implementation characteristics.
- Complex encryption and decryption
- Slow performance.
- It is a symmetric key algorithm, meaning each recipient must receive the key through a different channel than the message.
- It has been proven to be a weak cipher; therefore, should not be trusted to protect sensitive data.
- Due to the key size, the time it will take to encrypt and decrypt the message hinders efficient communication.

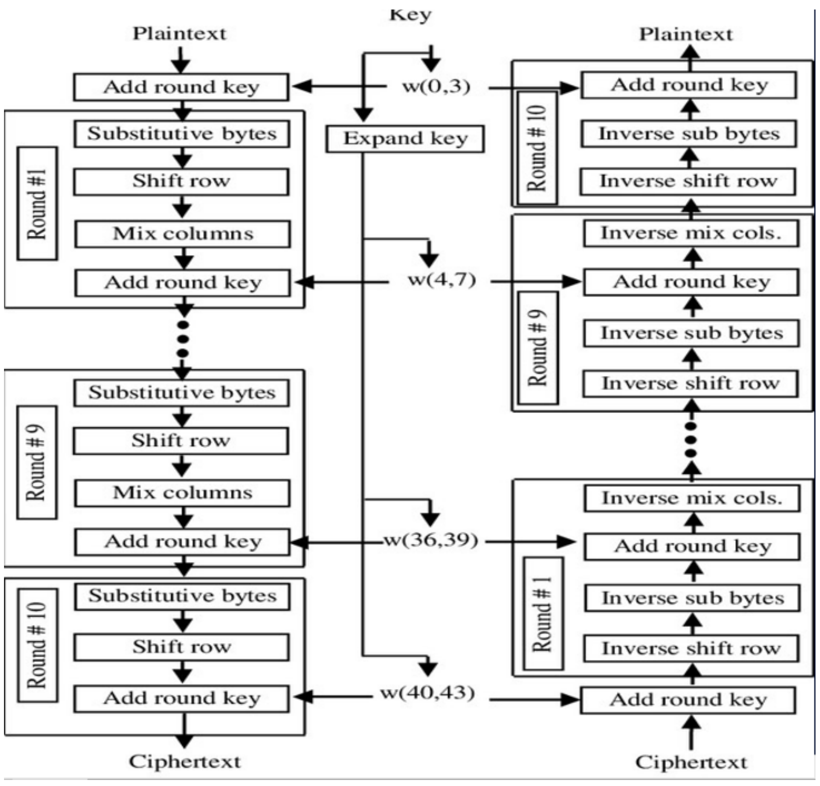
Related Works :

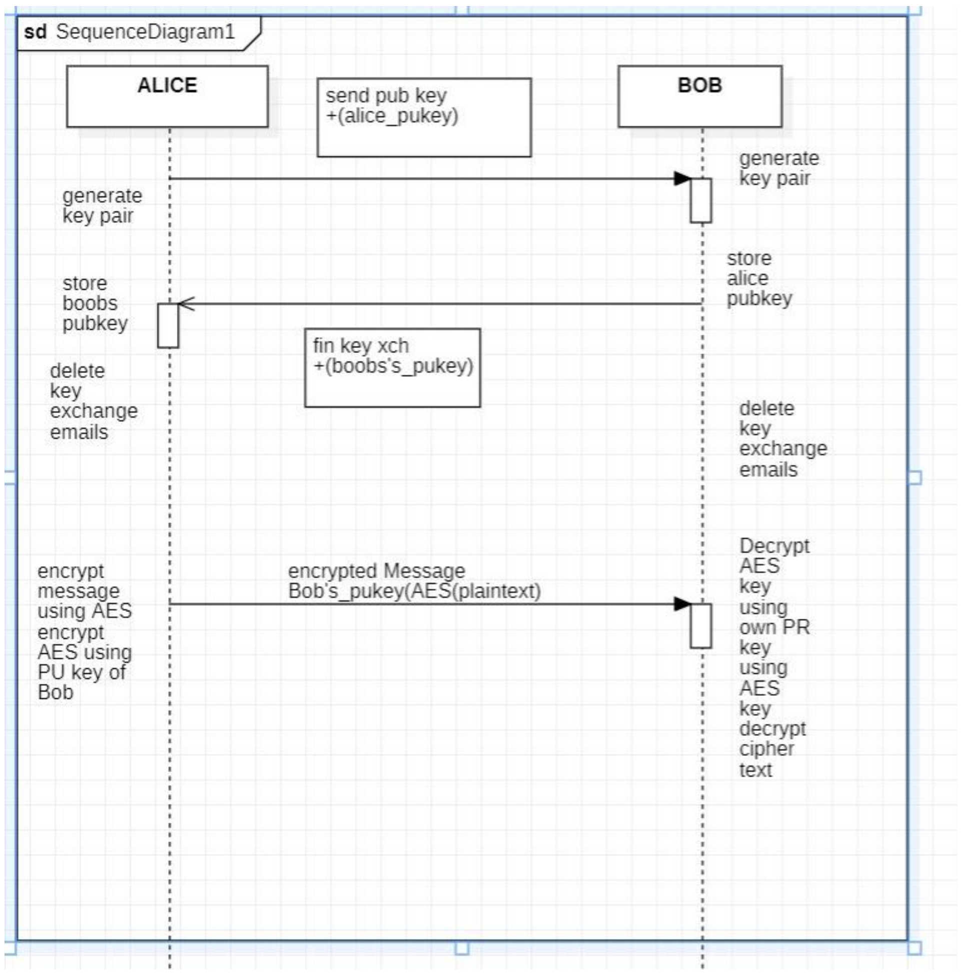
AES data encryption is a more mathematically efficient and elegant cryptographic algorithm, but its main strength rests in the option for various key lengths. AES allows you to choose a 128-bit, 192-bit or 256-bit key, making it exponentially stronger than the 56-bit key of DES.

In terms of structure, DES uses the Feistel network which divides the block into two halves before going through the encryption steps. AES on the other hand, uses permutation-substitution, which involves a series of substitution and permutation steps to create the encrypted block. The original DES designers made a great contribution to data security, but one could say that the aggregate effort of cryptographers for the AES algorithm has been far greater.

Architecture :

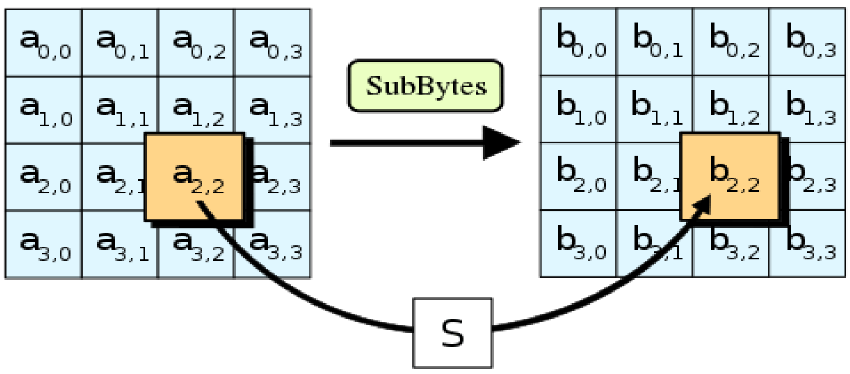
The AES architecture of this design adopts an iterative round- looping structure by mapping sub modules to lower datapaths of 8 and 32 bit. Internally, the AES transformations are performed on a two-dimensional array of bytes called the state. Here the hardwares used for the architecture is 8 bit while the entire data path remains 128 bit. The 128 bit registers are used in between the hardwares in order to obtain speed improvement with very less area utilization. The counter counts the number of rounds executed. The mux is used to skip the mix column operation in last stage. The key is expanded on the fly in order to save the area consumption of the storage of expanded key of 44 words.





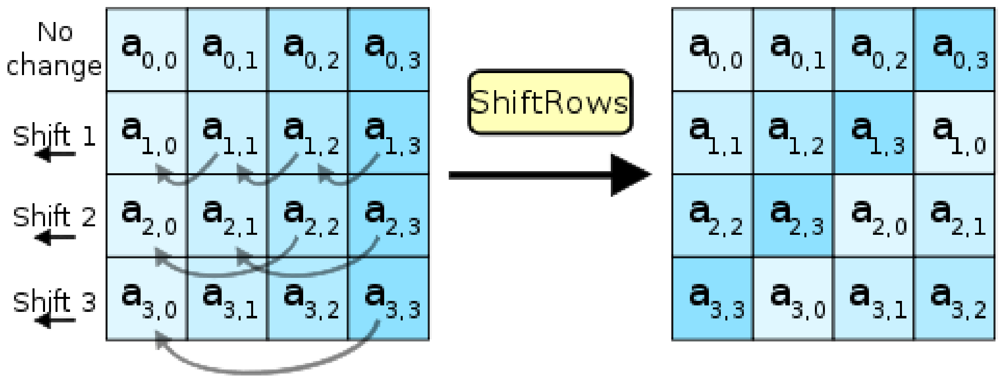
1. Sub Bytes :

   In the SubBytes step, each byte in the state array is replaced with a SubByte using an 8-bit substitution box. Note that before round 0, the state array is simply the plaintext/input. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), i.e., and also any opposite fixed points, i.e. While performing the decryption, the InvSubBytes step (the inverse of SubBytes) is used, which requires first taking the inverse of the affine transformation and then finding the multiplicative inverse.



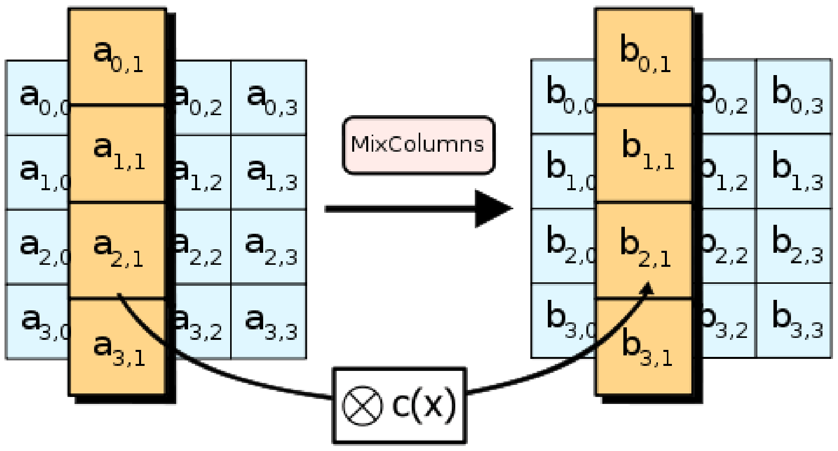
2. Shift Rows

   The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. The importance of this step is to avoid the columns being encrypted independently, in which case AES would degenerate into four independent block ciphers.



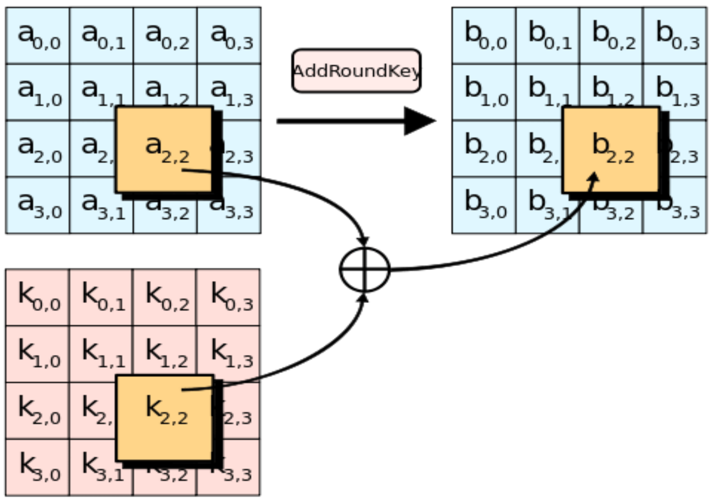
3. The MixColumns Step

   In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher. During this operation, each column is transformed using a fixed matrix.



4. Add Round Key

   In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.



Application of AES :

- Wi-Fi (can be used as part of WPA2)
- Mobile apps (such as WhatsApp and LastPass) Native Processor Support
- Libraries in many software development languages VPN Implementations
- Operating system components such as file systems

Existing Models :

RSA, as we know is a really amazing public key cipher that uses only basic number theory in its description. However, whenever a new cipher appears there will be many people that test its security and whenever possible will try to break it. So far RSA has not been broken but certain bad things can happen with it if you are not careful. Here are a few things that can go wrong.

 Using small primes.

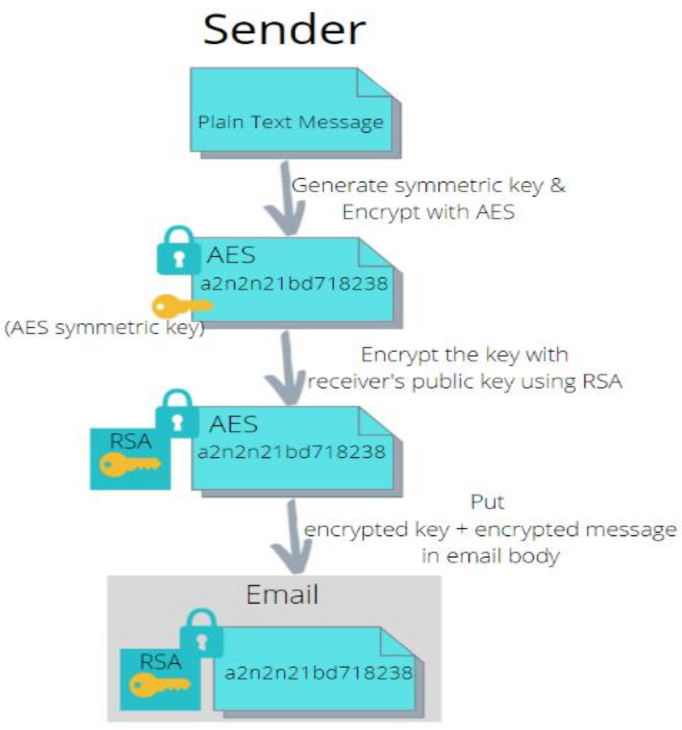
 Using primes that are very close. Message is an observable eth power.

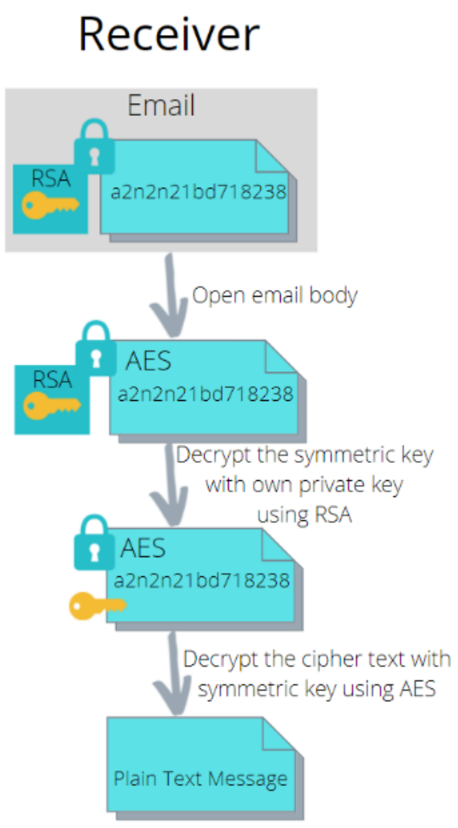
 Two people using the same N, receiving the same message. Sending the same message to one or more people with the same (Hastad’s attack)

E mail Encryption using AES Algorithm :

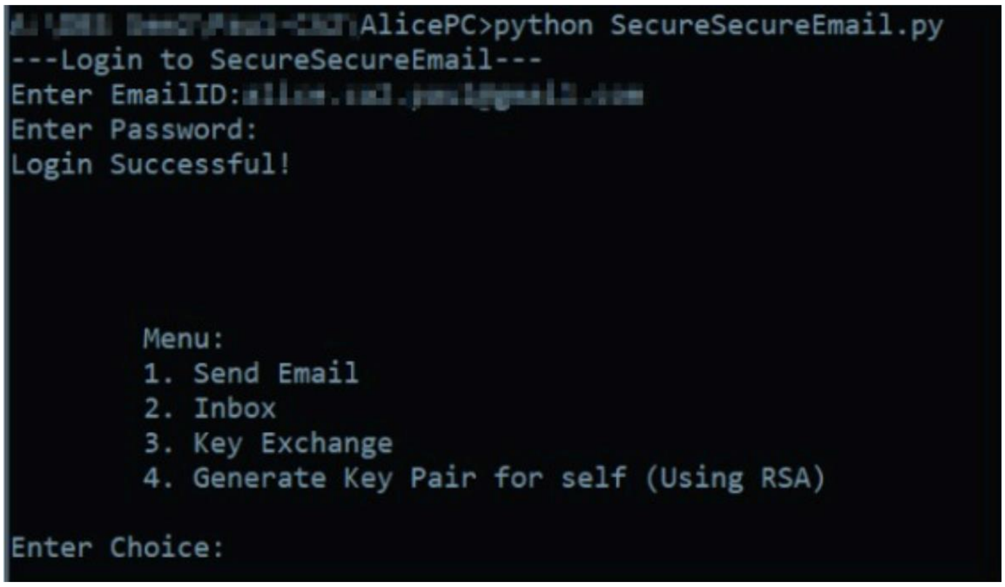
Working : It mainly makes use of two cryptographic algorithms:
(a) AES : Used for the encryption of plain text message.
(b) RSA : Used for transporting the AES symmetric key securely.

AES is a symmetric key encryption method. Meaning, the same key is used for encryption and decryption. That is also the reason, the key is used to encrypt cannot be shared with other party in plain text.
This is where RSA algorithm comes in play. RSA algorithm works with two different keys. For the encryption of message public key is used. And for decryption of message Private key is used. For the implementation of the application these two algorithms are used in such a way that- the data is encrypted with AES key. AES symmetric key is encrypted with public key of receiver. And the whole thing is sent across to the receiver.


Sender
Plain Text Message
Generate symmetric key & Encrypt with AES
AES
a2n2n21bd718238
(AES symmetric key)
Encrypt the key with receiver's public key using RSA
RSA
AES
a2n2n21bd718238
Put encrypted key + encrypted message in email body
Email
RSA
a2n2n21bd718238

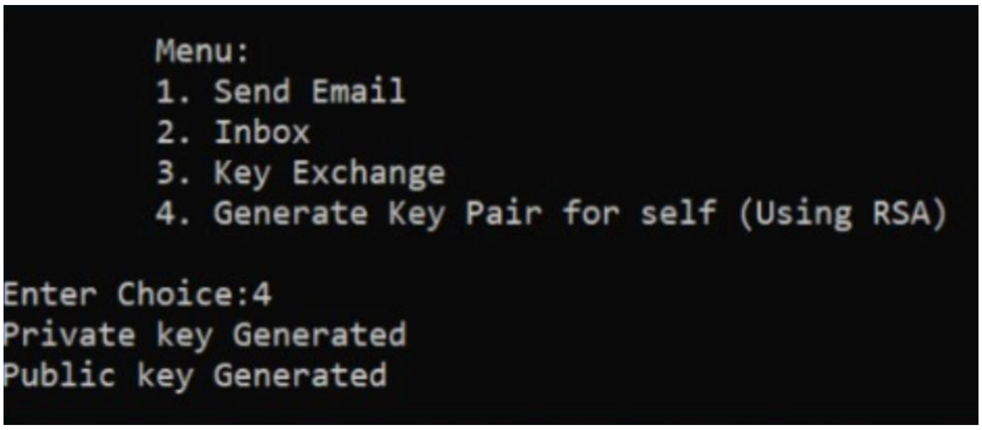

Receiver
Email
RSA
a2n2n21bd718238
Open email body
RSA
AES
a2n2n21bd718238
Decrypt the symmetric key with own private key using RSA
AES
a2n2n21bd718238
Decrypt the cipher text with symmetric key using AES
Plain Text Message

Key Exchange :



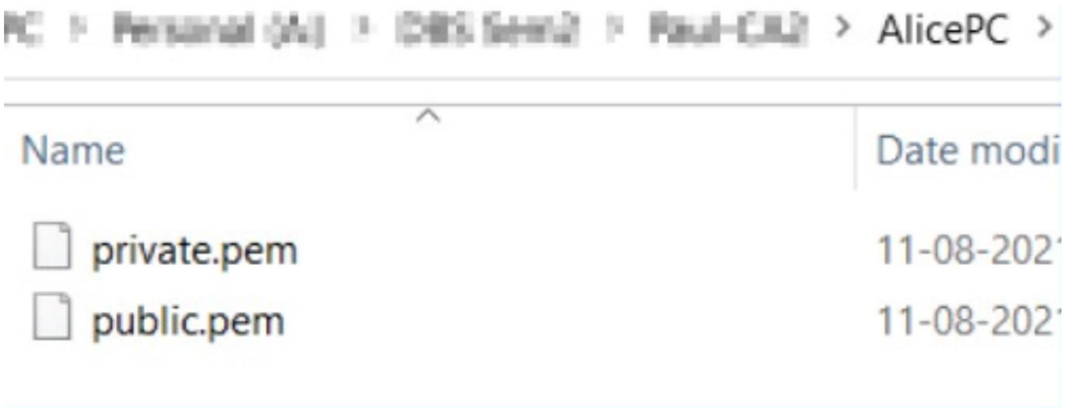
```
[redacted]AlicePC>python SecureSecureEmail.py
---Login to SecureSecureEmail---
Enter EmailID:[redacted]
Enter Password:
Login Successful!



        Menu:
        1. Send Email
        2. Inbox
        3. Key Exchange
        4. Generate Key Pair for self (Using RSA)

Enter Choice:
```

Login Gmail Account



```
        Menu:
        1. Send Email
        2. Inbox
        3. Key Exchange
        4. Generate Key Pair for self (Using RSA)

Enter Choice:4
Private key Generated
Public key Generated
```

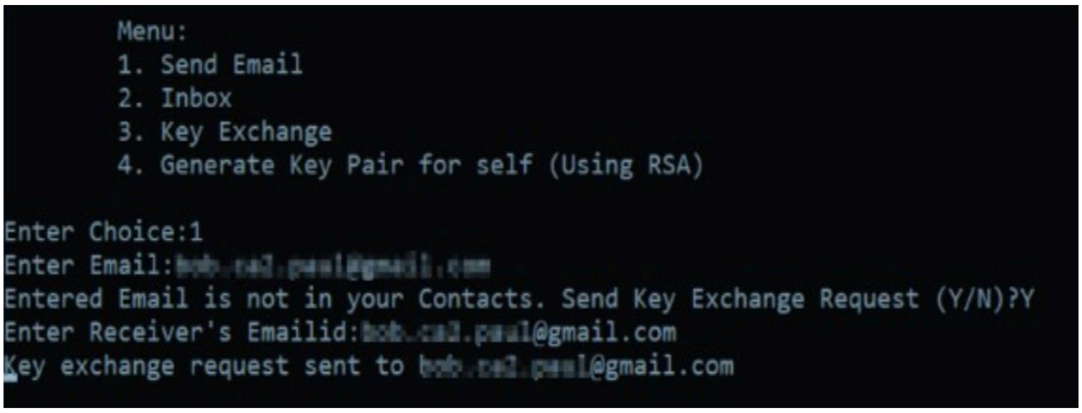
Generate key self



[redacted] > AlicePC >

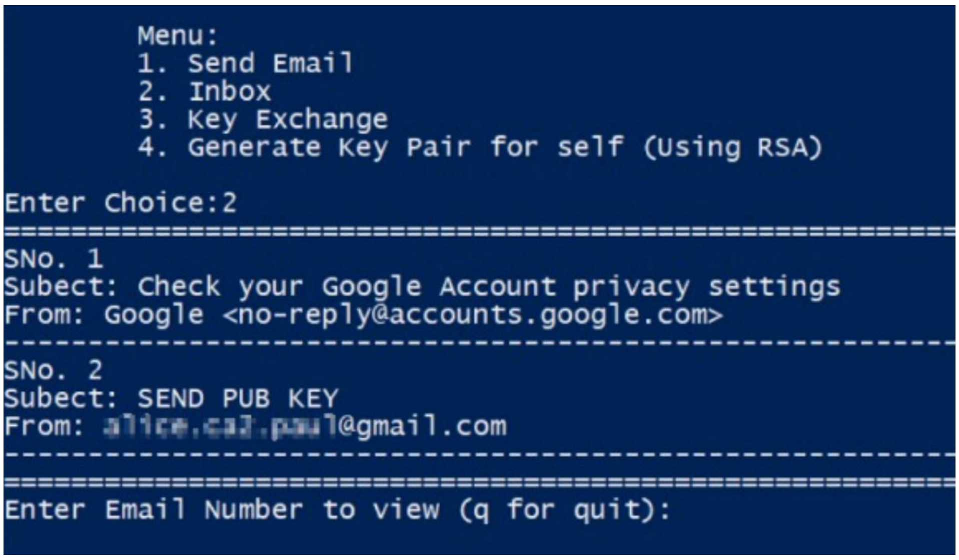
Name	Date modi
private.pem	11-08-202
public.pem	11-08-202

Keys Generated



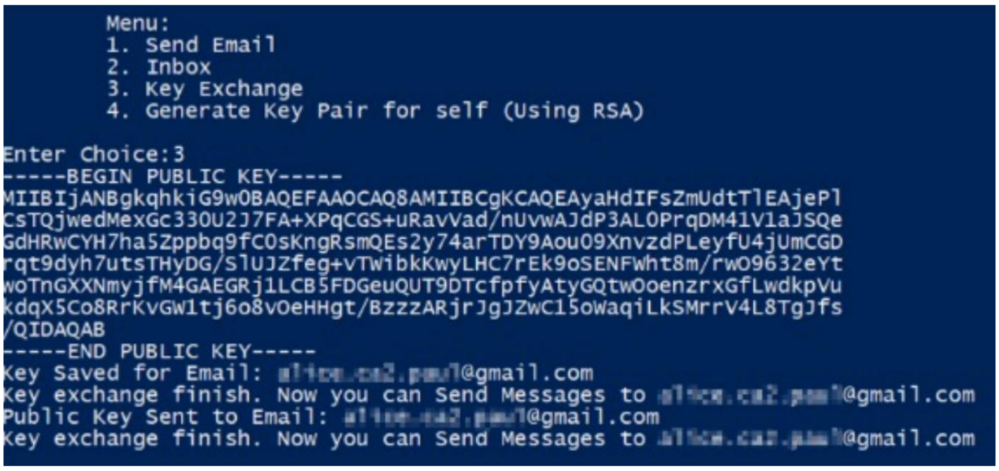
```
        Menu:
        1. Send Email
        2. Inbox
        3. Key Exchange
        4. Generate Key Pair for self (Using RSA)

Enter Choice:1
Enter Email:[redacted]
Entered Email is not in your Contacts. Send Key Exchange Request (Y/N)?Y
Enter Receiver's Emailid:[redacted]@gmail.com
Key exchange request sent to [redacted]@gmail.com
```

Send Key Request



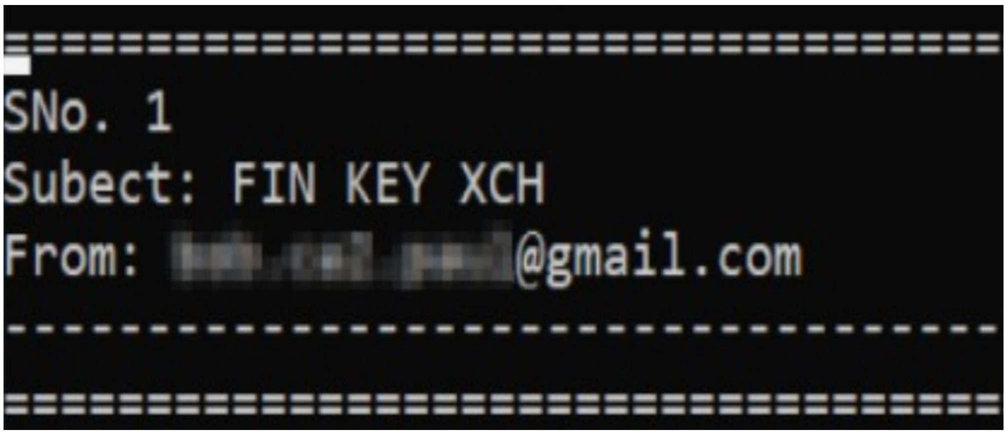
```
        Menu:
        1. Send Email
        2. Inbox
        3. Key Exchange
        4. Generate Key Pair for self (Using RSA)

Enter Choice:2
==========================================================
SNo. 1
Subect: Check your Google Account privacy settings
From: Google <no-reply@accounts.google.com>
----------------------------------------------------------
SNo. 2
Subect: SEND PUB KEY
From: [redacted]@gmail.com
----------------------------------------------------------
==========================================================
Enter Email Number to view (q for quit):
```

Key Exchange Request in Inbox of User B



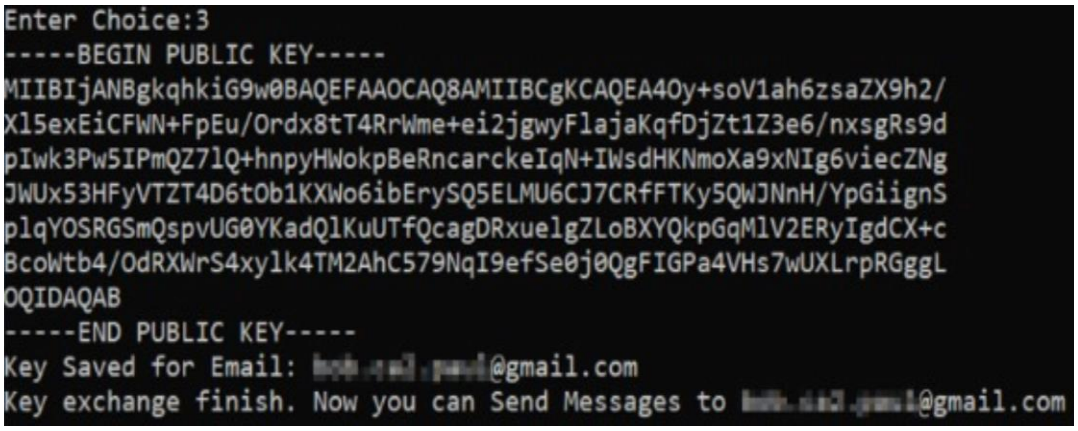
```
        Menu:
        1. Send Email
        2. Inbox
        3. Key Exchange
        4. Generate Key Pair for self (Using RSA)

Enter Choice:3
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyaHdIFsZmUdtTlEAjePl
CsTQjwedMexGc330U2J7FA+XPqCGS+uRavVad/nUvwAJdP3AL0PrqDM41V1aJSQe
GdHRwCYH7ha5Zppbq9fC0sKngRsmQEs2y74arTDY9Aou09XnvzdPLeyfU4jUmCGD
rqt9dyh7utsTHyDG/SlUJZfeg+vTWibkKwyLHC7rEk9oSENFWht8m/rw09632eYt
woTnGXXNmyjfM4GAEGRj1LCB5FDGeuQUT9DTcfpfyAtyGQtwOoenzrxGfLwdkpVu
kdqX5Co8RrKvGW1tj6o8vOeHHgt/BzzzARjrJgJZwC15oWaqiLkSMrrV4L8TgJfs
/QIDAQAB
-----END PUBLIC KEY-----
Key Saved for Email: [redacted]@gmail.com
Key exchange finish. Now you can Send Messages to [redacted]@gmail.com
Public Key Sent to Email: [redacted]@gmail.com
Key exchange finish. Now you can Send Messages to [redacted]@gmail.com
```

Key Saved by B. B's Public key sent to A



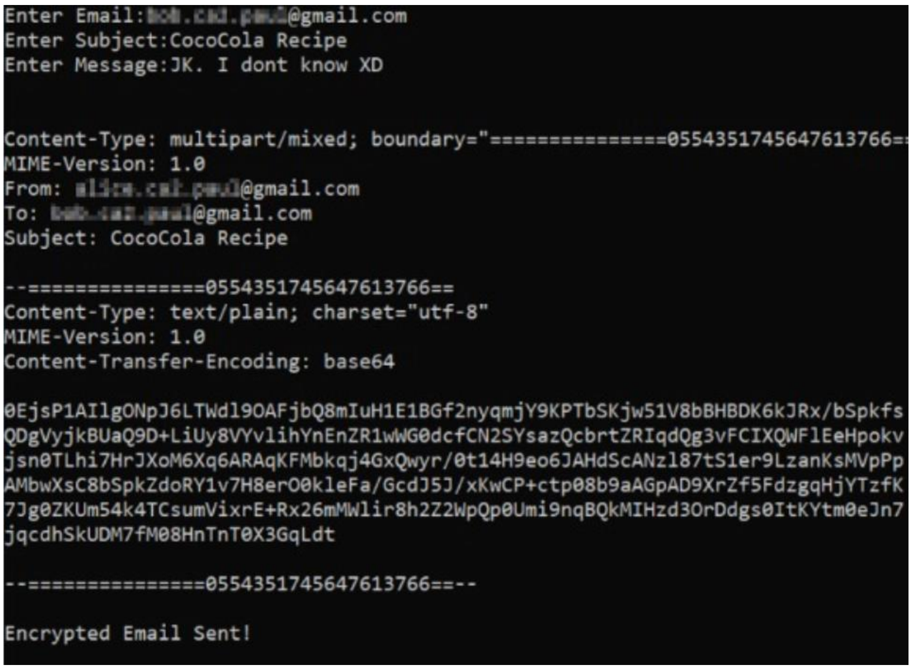
```
==================================
SNo. 1
Subect: FIN KEY XCH
From: [redacted]@gmail.com
----------------------------------
==================================
```

Public key of B in A's inbox



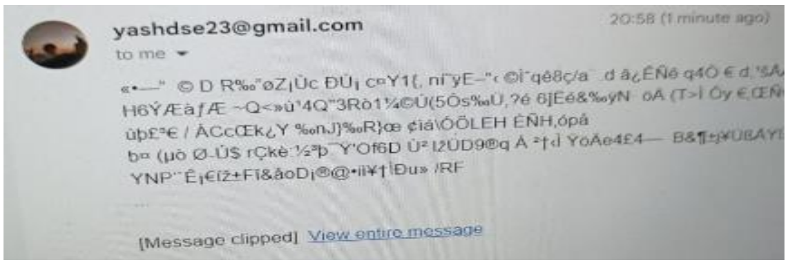
```
Enter Choice:3
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4Oy+soV1ah6zsaZX9h2/
Xl5exEiCFWN+FpEu/Ordx8tT4RrWme+ei2jgwyFlajaKqfDjZt1Z3e6/nxsgRs9d
pIwk3Pw5IPmQZ7lQ+hnpyHWokpBeRncarckeIqN+IWsdHKNmoXa9xNIg6viecZNg
JWUx53HFyVTZT4D6tOb1KXWo6ibErySQ5ELMU6CJ7CRfFTKy5QWJNnH/YpGiignS
plqYOSRGSmQspvUG0YKadQlKuUTfQcagDRxuelgZLoBXYQkpGqMlV2ERyIgdCX+c
BcoWtb4/OdRXWrS4xylk4TM2AhC579NqI9efSe0j0QgFIGPa4VHs7wUXLrpRGggL
OQIDAQAB
-----END PUBLIC KEY-----
Key Saved for Email: [redacted]@gmail.com
Key exchange finish. Now you can Send Messages to [redacted]@gmail.com
```

Key saved at A



```
Enter Email:[redacted]@gmail.com
Enter Subject:CocoCola Recipe
Enter Message:JK. I dont know XD


Content-Type: multipart/mixed; boundary="===============0554351745647613766=
MIME-Version: 1.0
From: [redacted]@gmail.com
To: [redacted]@gmail.com
Subject: CocoCola Recipe

--===============0554351745647613766==
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: base64

0EjsP1AIlgONpJ6LTWdl9OAFjbQ8mIuH1E1BGf2nyqmjY9KPTbSKjw51V8bBHBDK6kJRx/bSpkfs
QDgVyjkBUaQ9D+LiUy8VYvlihYnEnZR1wWG0dcfCN2SYsazQcbrtZRIqdQg3vFCIXQWFlEeHpokv
jsn0TLhi7HrJXoM6Xq6ARAqKFMbkqj4GxQwyr/0t14H9eo6JAHdScANzl87tS1er9LzanKsMVpPp
AMbwXsC8bSpkZdoRY1v7H8erO0kleFa/GcdJ5J/xKwCP+ctp08b9aAGpAD9XrZf5FdzgqHjYTzfK
7Jg0ZKUm54k4TCsumVixrE+Rx26mMWlir8h2Z2WpQp0Umi9nqBQkMIHzd3OrDdgs0ItKYtm0eJn7
jqcdhSkUDM7fM08HnTnT0X3GqLdt

--===============0554351745647613766==--

Encrypted Email Sent!
```

Key saved at A



yashdse23@gmail.com

to me

20:58 (1 minute ago)

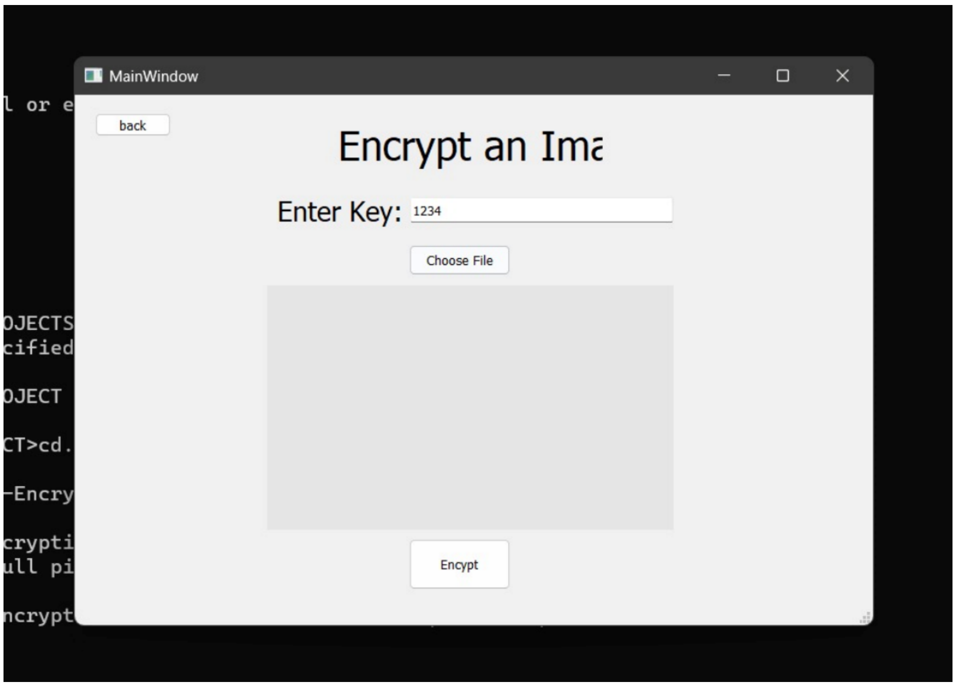
«•—" © D R‰"øZ¡Üc ÐÜ¡ c¤Y1{, nÏ¨ÿE–"‹ ©Ì¨qê8ç/a¨ d â¿ÊÑê q4Ò € d,'šÅ
H6ÝÆàƒÆ ~Q<»ú'4Q"3Rò1¼©Û(5Ôs‰Ü,?é 6]Eé&‰ÿN óA (T>Ì Óy €,ŒÑ
ùþ£ª€ / ÀCcŒk¿Y ‰nJ}‰R}œ ¢ïá\ÖÖLEH ÊÑH,ópá
þ¤ (µò Ø-Û$ rÇkè ½ªþ¯Y'Of6D Û² I2ÜD9®q Á ²†‹Ì YóÄe4£4— B&¶±j¥ÜBAYI
YNP¨Ê¡€Iž±FÏ&åoD¡®@•ïi¥†¦Ðu» /RF

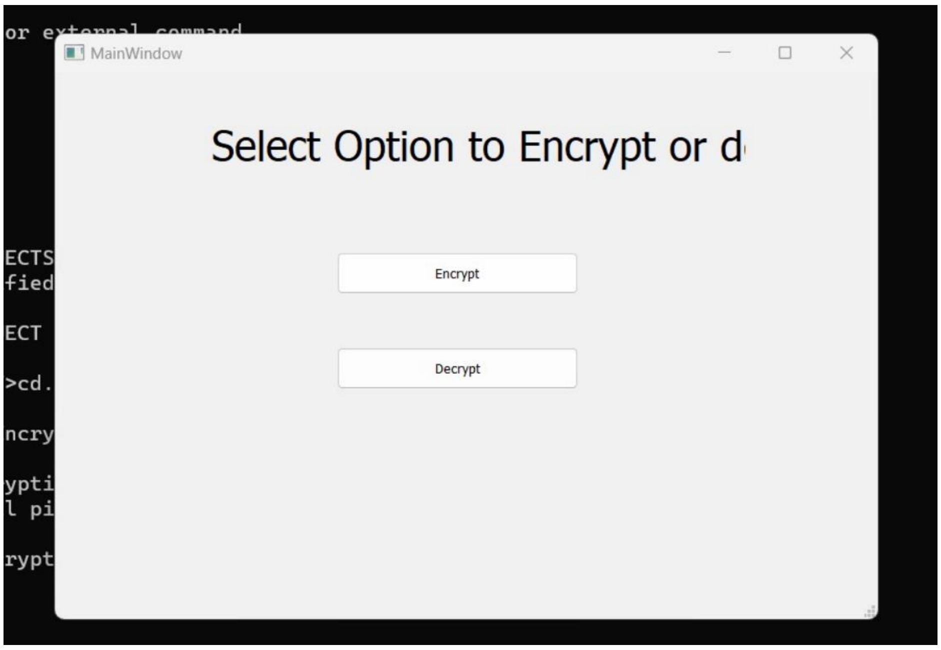
...

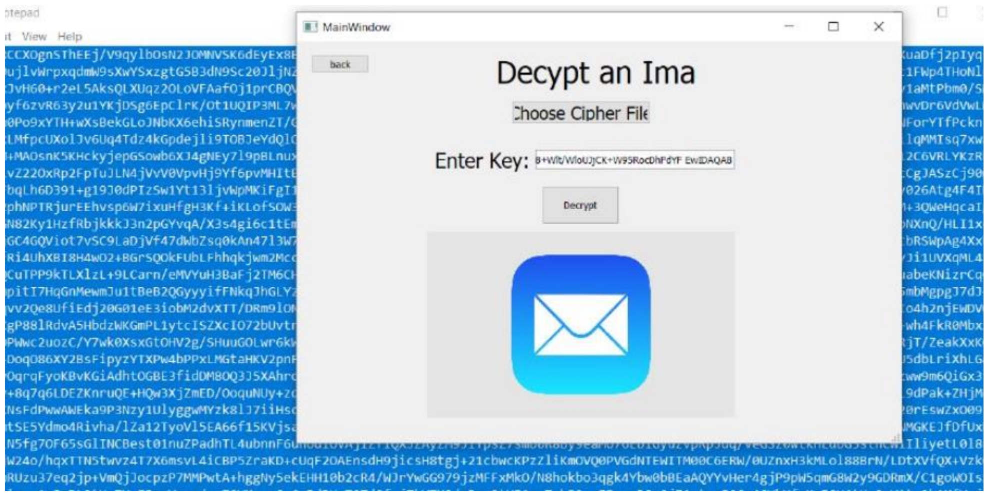
[Message clipped] View entire message

Secret message

Encrypting an image using AES :

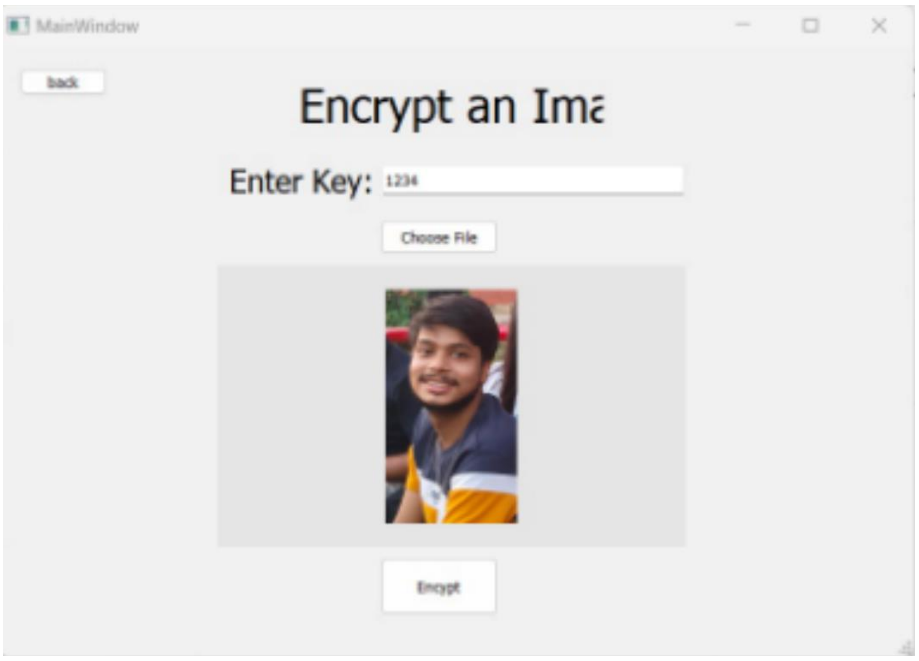




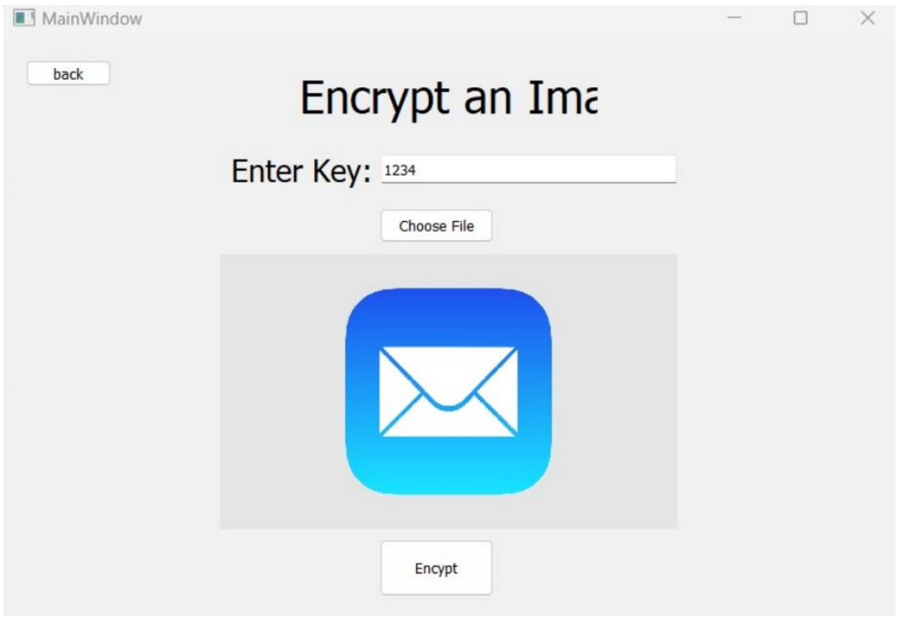


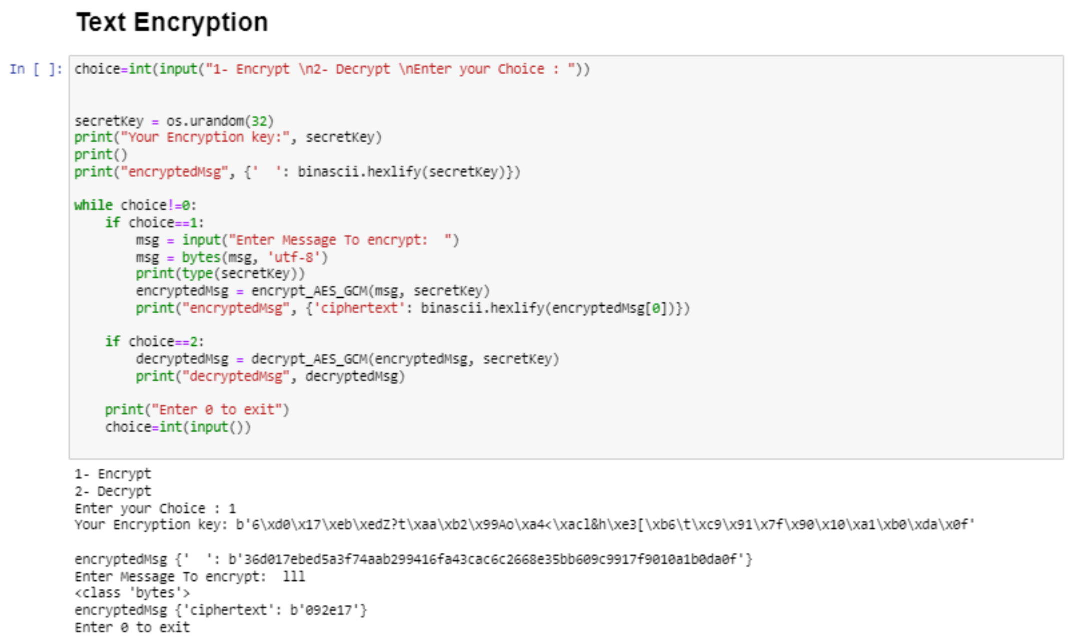
Enter Key: 8+Wlt/WloUJCK+W95RocDhPdYF EwIDAQAB

```
tCCXOgnSThEEj/V9qylbOsN2JOMNVSK6dEyEx8E                                   xuaDfj2pIyq
0ujlvWrpxqdmW9sXwYSxzgtG5B3dN9Sc20J1jNZ                                   t1FWp4THoNl
tJvH60+r2eL5AksQLXUqz2OLoVFAafOj1prCBQV                                   v1aMtPbm0/S
myf6zvR63y2u1YKjDSg6EpClrK/Ot1UQIP3ML7w                                   wvDr6VdVwL
d0Po9xYTH+wXsBekGLoJNbKX6ehiSRynmenZT/G                                   JForYTfPckn
tLMfpcUXolJv6Uq4Tdz4kGpdejli9TOBJeYdQlG                                   lqMMIsq7xw
d+MAOsnK5KHckyjepGSowb6XJ4gNEy7l9pBLnux                                   L2C6VRLYKzR
LvZ22OxRp2FpTuJLN4jVvV0VpvHj9Yf6pvMHItE                                   tCgJASzCj90
fbqLh6D391+g19J0dPIzSw1Yt13ljvWpMKiFgI1                                   v026Atg4F4I
zphNPTRjurEEhvsp6W7ixuHfgH3Kf+iKLofSOW3                                   M+3QWeHqcaI
sN82Ky1HzfRbjkkkJ3n2pGYvqA/X3s4gi6c1tEm                                   bNXnQ/HLI1x
gGC4GQViot7vSC9LaDjVf47dWbZsq0kAn47l3W7                                   tbRSWpAg4Xx
fRi4UhXBI8H4wO2+BGrSQOkFUbLFhhqkjwm2Mcc                                   vJi1UVXqML4
gCuTPP9kTLXlzL+9LCarn/eMVYuH3BaFj2TM6CH                                   uabeKNizrCq
mpitI7HqGnMewmJu1tBeB2QGyyyifFNkqJhGLYz                                   5mbMgpgJ7dJ
qvv2Qe8UfiEdj20G01eE3iobM2dvXTT/DRm9lOM                                   Lo4h2njEWDV
zgP88lRdvA5HbdzWKGmPL1ytcISZXcIO72bUvtr                                   -wh4FkR0Mbx
ePWwc2uozC/Y7wk0XsxGtOHV2g/SHuuGOLwr6kW                                   RjT/ZeakXxK
LDoqO86XY2BsFipyzYTXPw4bPPxLMGtaHKV2pnF                                   U5dbLriXhLG
rOqrqFyoKBvKGiAdhtOGBE3fidDM8OQ3J5XAhrc                                   zww9m6QiGx3
r+8q7q6LDEZKnruQE+HQw3XjZmED/OoquNUy+zc                                   L9dPak+ZHjM
ENsFdPwwAWEka9P3Nzy1UlyggwMYzk8lJ7iiHsc                                   20rEswZxO09
utSE5Ydmo4Rivha/lZa12TyoVl5EA66f15KVjsa                                   UMGKEJfDfUx
lN5fg7OF65sGlINCBest01nuZPadhTL4ubnnF6u                                   lIliyetL0l8
W24o/hqxTTN5twvz4T7X6msvL4iCBP5ZraKD+cUqF2OAEnsdH9jicsH8tgj+21cbwcKPzZliKmOVQ0PVGdNTEWITM00C6ERW/0UZnxH3kMLol88BrN/LDtXVfQX+Vzk
RUzu37eq2jp+VmQjJocpzP7MMPwtA+hggNy5ekEHH10b2cR4/WJrYwGG979jzMFFxMkO/N8hokbo3qgk4Ybw0bBEaAQYYvHer4gjP9pW5qmG8W2y9GDRmX/C1goWOIs
```



Enter Key: 1234




Text Encryption

```
In [ ]: choice=int(input("1- Encrypt \n2- Decrypt \nEnter your Choice : "))


        secretKey = os.urandom(32)
        print("Your Encryption key:", secretKey)
        print()
        print("encryptedMsg", {'  ': binascii.hexlify(secretKey)})

        while choice!=0:
            if choice==1:
                msg = input("Enter Message To encrypt:  ")
                msg = bytes(msg, 'utf-8')
                print(type(secretKey))
                encryptedMsg = encrypt_AES_GCM(msg, secretKey)
                print("encryptedMsg", {'ciphertext': binascii.hexlify(encryptedMsg[0])})

            if choice==2:
                decryptedMsg = decrypt_AES_GCM(encryptedMsg, secretKey)
                print("decryptedMsg", decryptedMsg)

            print("Enter 0 to exit")
            choice=int(input())
```

```
1- Encrypt
2- Decrypt
Enter your Choice : 1
Your Encryption key: b'6\xd0\x17\xeb\xedZ?t\xaa\xb2\x99Ao\xa4<\xacl&h\xe3[\xb6\t\xc9\x91\x7f\x90\x10\xa1\xb0\xda\x0f'

encryptedMsg {'  ': b'36d017ebed5a3f74aab299416fa43cac6c2668e35bb609c9917f9010a1b0da0f'}
Enter Message To encrypt:  lll
<class 'bytes'>
encryptedMsg {'ciphertext': b'092e17'}
Enter 0 to exit
```

Literature Survey :

[A] Data Encryption and Decryption Using RSA Algorithm in a Network Environment

Network Security is premised on the fact that once there is connectivity between computers sharing some resources, the issue of data security becomes critical. This paper presents a design of data encryption and decryption in a network environment using RSA algorithm with a specific message block size. RSA is the most popular public key cryptography (PKC). It uses 2 key cryptosystem, a public key which is known by the sender and the receiver and a private key which is known only by the receiver, so that two parties can engage in a secure communication over a non secure communication channel without having to share key

RSA encryption: The public and the private key-generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, p and q, are generated using the Rabin-Miller primality test algorithm. A modulus n is calculated by multiplying p and q. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length. The public key consists of the modulus n, and a public exponent, e, which is normally set at 65537, as it's a prime number that is not too large. The e figure doesn't have to be a secretly selected prime number as the public key is shared with everyone. The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the totient of n.

[B] A Study of Encryption Algorithms AES, DES and RSA for Security

This paper by Dr. Prerna Mahajan & Abhishek Sachdeva (IITM India)is an effective comparison of the three important cryptography techniques using AES , DES and RSA comparing its performance based on simulation time for encryption and decryption and analysing the experimental result to realise effectiveness of each algorithm. encryption algorithms can be categorized into Symmetric (private) and Asymmetric (public) keys encryption.Public key encryption is based on mathematical functions, computationally intensive and is not very efficient for small mobile devices . Asymmetric encryption techniques are almost 1000 times slower than symmetric techniques due to computational processing power.

Conclusion :

In this project we have used AES to encrypt the text data and then encrypted the AES key using symmetric encryption. AES is strong symmetric algorithm, but it is very fast to implement and thus bulky data can be encrypted using this encryption. E-mail encryption can rely on public-key cryptography, in which users can each publish a public key that others can use to encrypt messages to them, while keeping secret a private key they can use to decrypt such messages or to digitally encrypt and sign messages they send. We have put the screenshots of the implemented model of both image and email encryption using AES algorithm

References :

https://ieeexplore.ieee.org/document/9243651

https://www.iosrjournals.org/iosr-jce/papers/Vol22-issue6/Series-1/I2206013944.pdf

https://www.researchgate.net/publication/357232938_AES_Image_Encryption

https://www.iosrjournals.org/iosr-jvlsi/papers/vol10-issue4/Series-1/B10040819.pdf