PYTHON – PROJECT REPORT

Track:

ALGORITHMS

Topic:

RSA Encryption Algorithm

Group members:

1. Abhigna Banda
2. Nikitha A.N
3. Arpitha Srivathsa

OVERVIEW:

1.INTRODUCTION TO CRYPTOSYSTEMS :

Cryptography used to ensure confidentiality and integrity of data in systems by transforming original text into an encrypted format. It involves use of different algorithms to do the encryption and decryption.

Encryption is broadly classified into 3 categories:
i)symmetric (sender and receiver use the same key)
ii)asymmetric (sender and reciever use 2 different keys)
iii)hash function

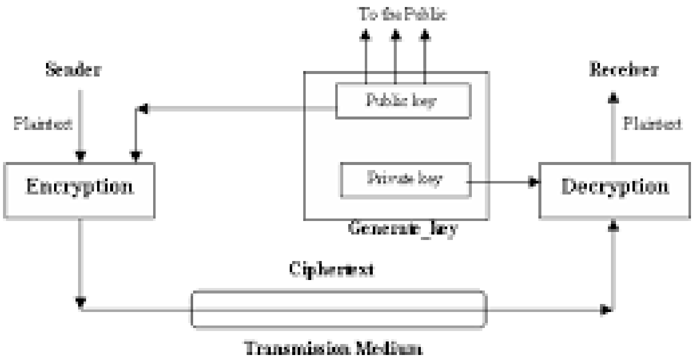
2.INTRODUCTION TO RSA:

RSA (Rivest-Shamir-Adleman) is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Thus it involves the use of 2 seperate keys for encryption and decryption.

This is also called public key cryptography because one of the keys can be given to anyone. The other key must be kept private.

The algorithm is robust as it is based on the fact that finding the factors of a large composite number is difficult: especially when the factors are random prime numbers. Hence even if the public key is known it is difficult to find the private key.

It is also a key pair (public and private key) generator.



3.IMPLEMENTATION :

This project is an implementation of RSA encryption algorithm.
We have two types of implementation:

1)Using File handling:
The data to be encrypted has to be written into the file “toBeEncrypted.txt” .
Two files will be created namely “encrypted.txt” and “Decrypted.txt” containing encrypted and decrypted data respectively.

NOTE: If you are running the program more than once make sure the previoiusly created “encrypted.txt” and “Decrypted.txt” are deleted from the directory.

2)Using GUI tkinter:
A message box will appear right after running the program.
The buttons Encrypt and Decrypt will perform the respective operations.

4.ALGORITHM IN STEPS

1.choose two large prime numbers p and q
2.calculate n=p*q which is the modulus for private and public keys
3.claculate phi=(p-1)*(q-1)
4.choose a random integer e such that

    i) 1<e<phi
    ii) e is not a factor of n
    iii) e and phi are co-prime

e is released as part of public key
5.choose an integer d which satisfies the relation (d*e)mod n=1
i.e d=(K*phi +1)/e where k is any random integer
d is kept secret as part of the private key

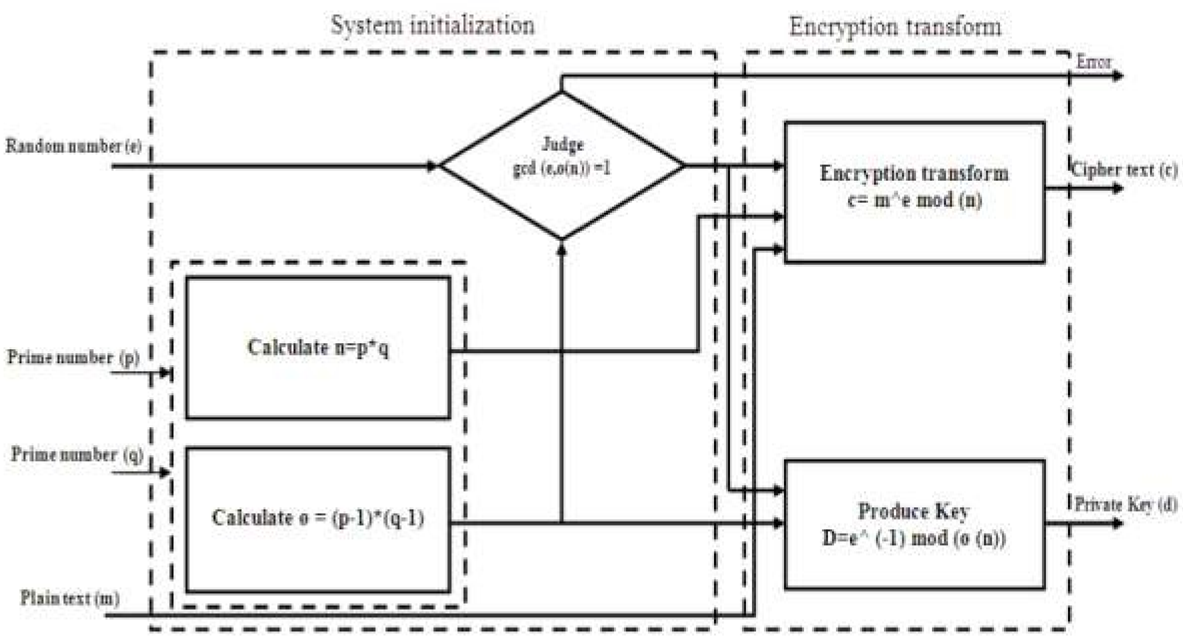
public key: (n,e)
private key: (n,d)

if m is the messsage and c is the ciphertext
To encrypt the formula used is:

$$c=m^{e}\text{mod } n$$

To decrypt the formula used is:

$$m=c^{d}\text{mod } n$$

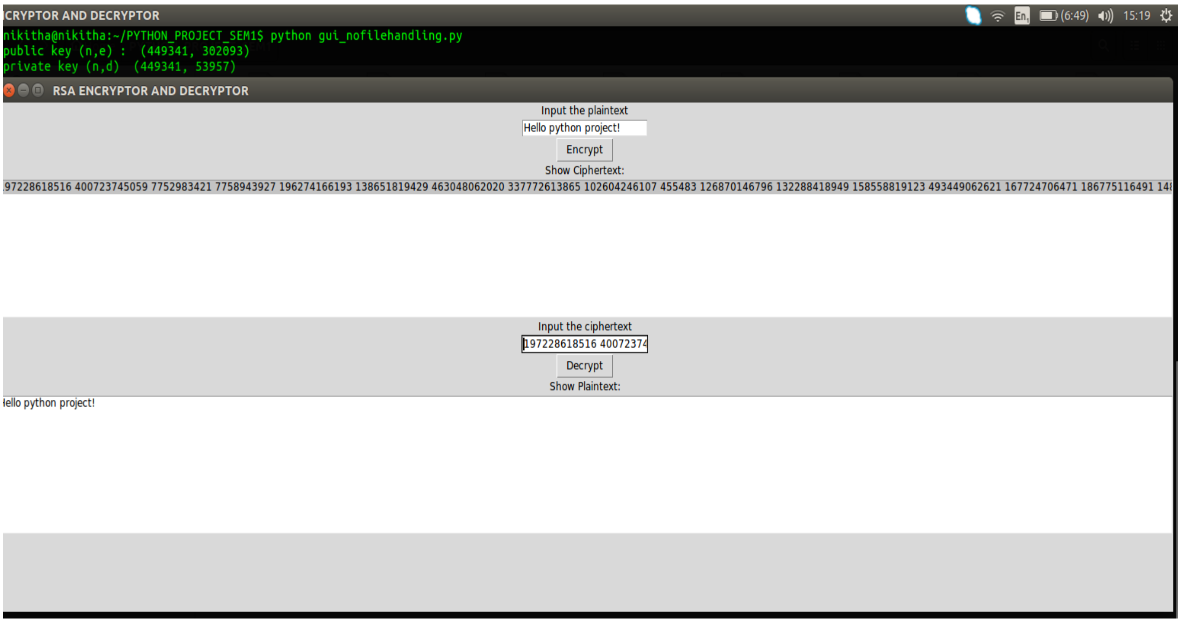


5.SOURCE CODE:

The two files containing the source code are attached with this report.

6.SCREENSHOTS OF OUTPUT:

1.USING GUI:



```
nikitha@nikitha:~/PYTHON_PROJECT_SEM1$ python gui_nofilehandling.py
public key (n,e) :  (449341, 302093)
private key (n,d)  (449341, 53957)
```

Input the plaintext: Hello python project!

Show Ciphertext:

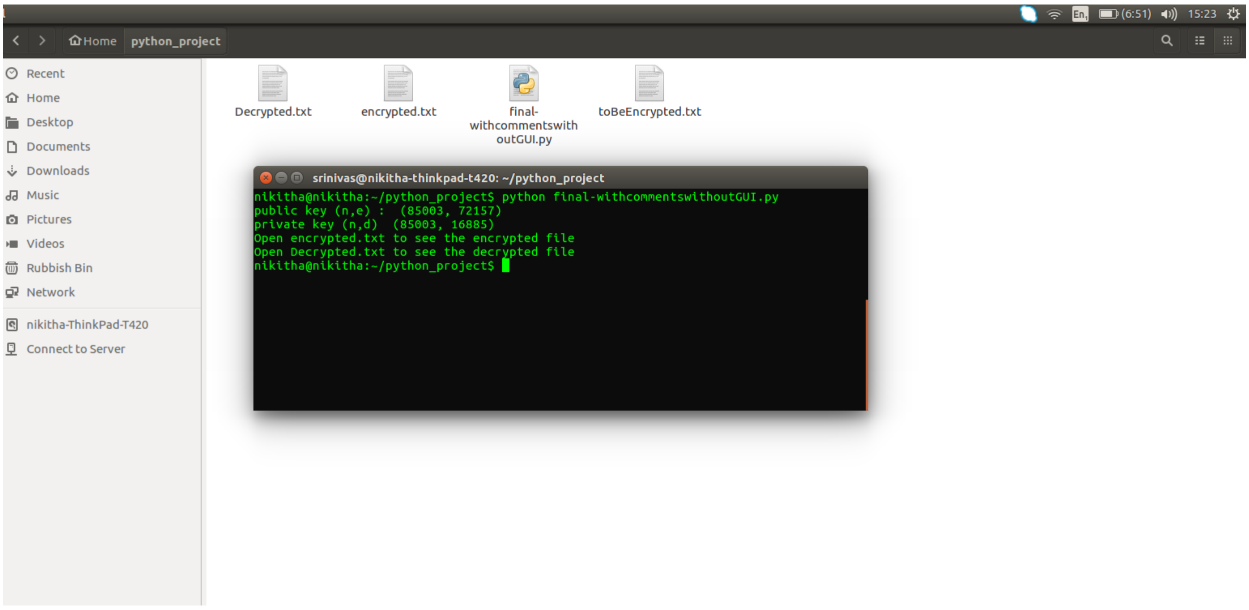
```
97228618516 400723745059 7752983421 7758943927 196274166193 138651819429 463048062020 337772613865 102604246107 455483 126870146796 132288418949 158558819123 493449062621 167724706471 186775116491 14
```

Input the ciphertext: 197228618516 40072374

Show Plaintext:

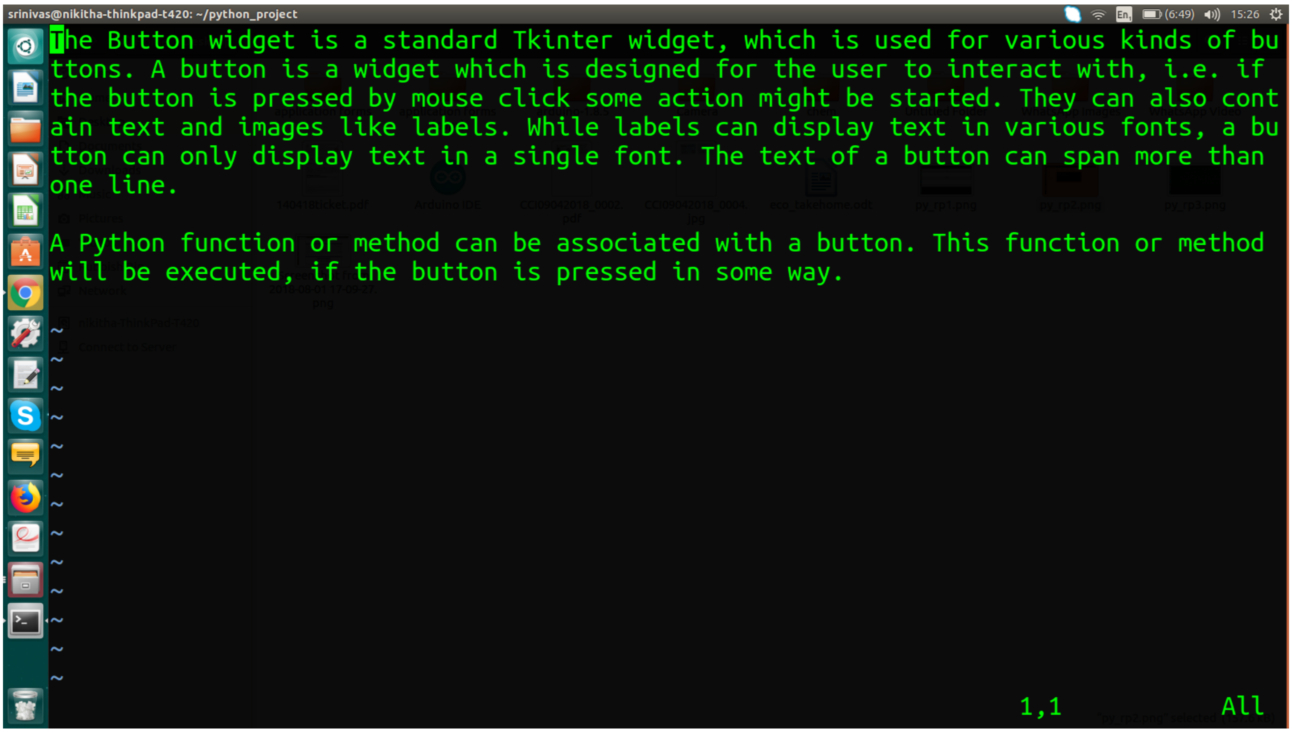
```
Hello python project!
```

2.USING FILE HANDLING:



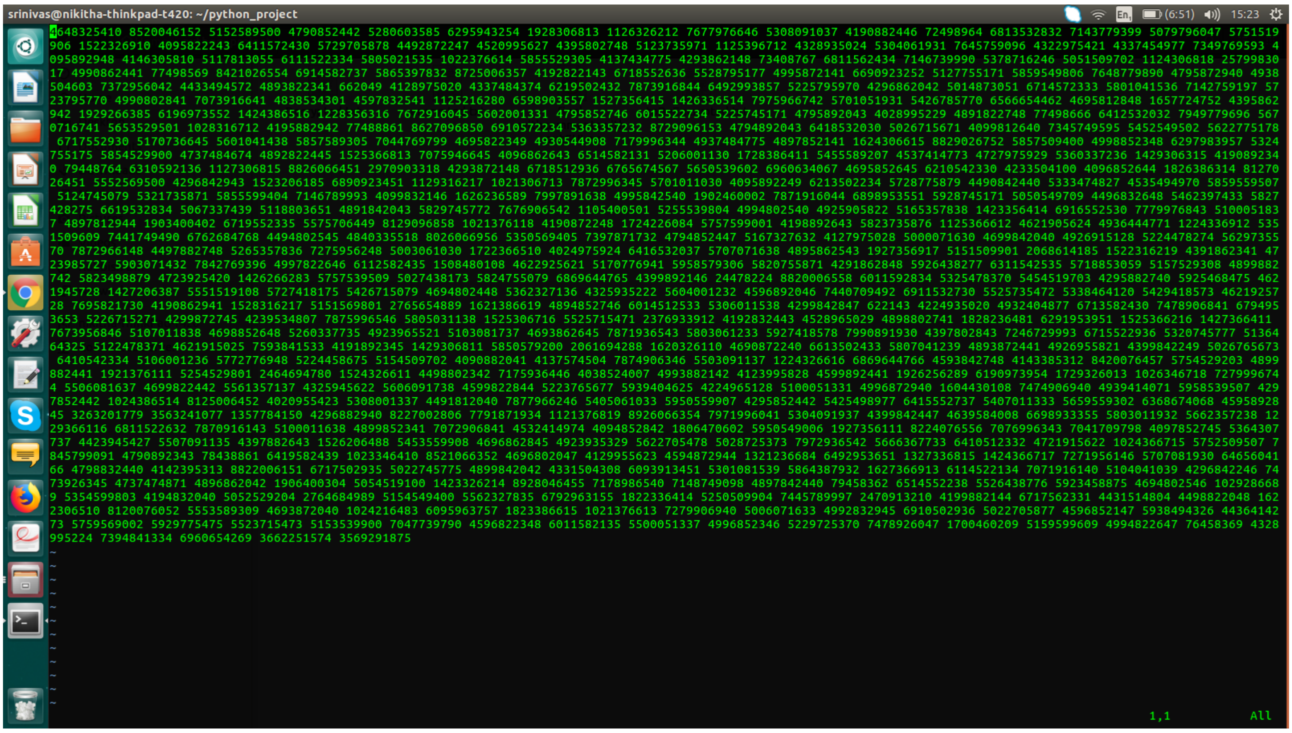
```
nikitha@nikitha:~/python_project$ python final-withcommentswithoutGUI.py
public key (n,e) :  (85003, 72157)
private key (n,d)  (85003, 16885)
Open encrypted.txt to see the encrypted file
Open Decrypted.txt to see the decrypted file
nikitha@nikitha:~/python_project$ 
```

THE DATA THAT HAS TO BE ENCRYPTED



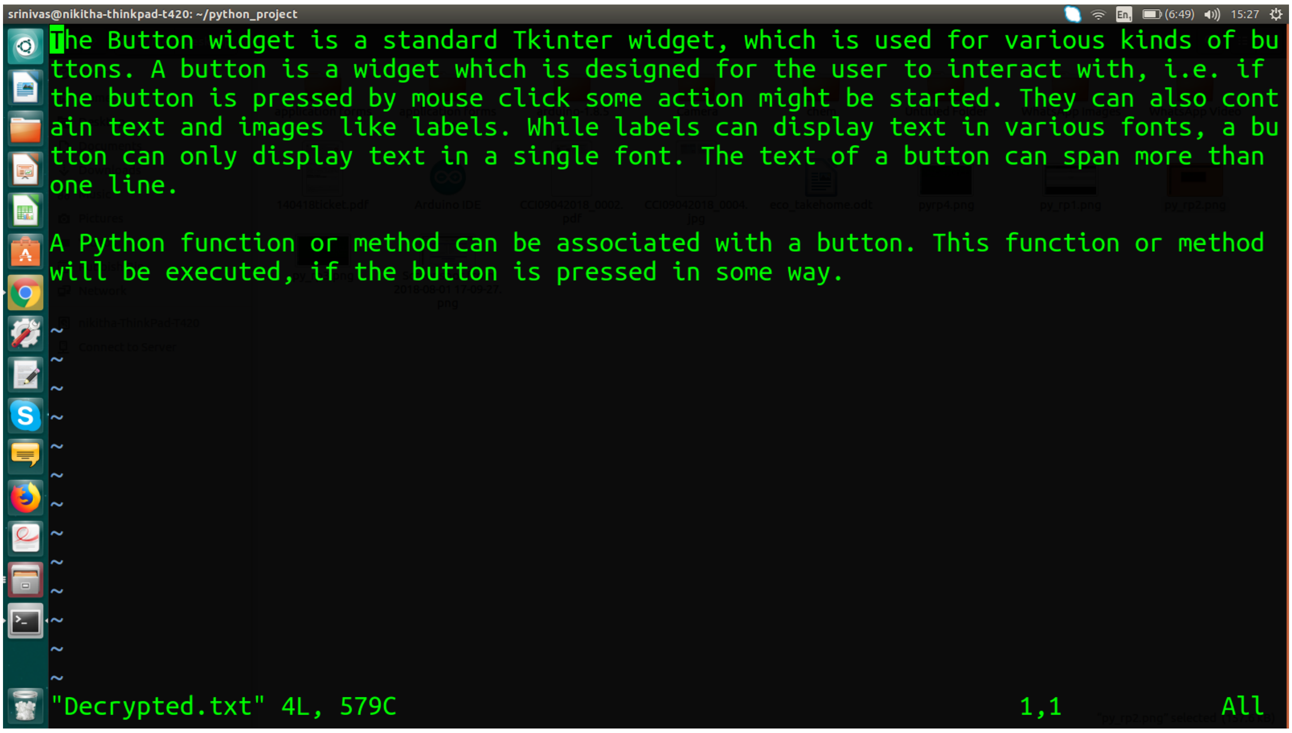
```
The Button widget is a standard Tkinter widget, which is used for various kinds of bu
ttons. A button is a widget which is designed for the user to interact with, i.e. if
the button is pressed by mouse click some action might be started. They can also cont
ain text and images like labels. While labels can display text in various fonts, a bu
tton can only display text in a single font. The text of a button can span more than
one line.

A Python function or method can be associated with a button. This function or method
will be executed, if the button is pressed in some way.
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                1,1           All
```

THE ENCRYPTED DATA



```
4648325410 8520046152 5152589500 4790852442 5280603585 6295943254 1928306813 1126326212 7677976646 5308091037 4190882446 72498964 6813532832 7143779399 5079796047 5751519
906 1522326910 4095822243 6411572430 5729705878 4492872247 4520995627 4395802748 5123735971 1125396712 4328935024 5304061931 7645759096 4322975421 4337454977 7349769593 4
095892948 4146305810 5117813055 6111522334 5805021535 1022376614 5855529305 4137434775 4293862148 73408767 6811562434 7146739990 5378716246 5051509702 1124306818 25799830
17 4990862441 77498569 8421026554 6914582737 5865397832 8725006357 4192822143 6718552636 5528795177 4995872141 6690923252 5127755171 5859549806 7648779890 4795872940 4938
504603 7372956042 4433494572 4893822341 662049 4128975020 4337484374 6219502432 7873916844 6492993857 5225795970 4296862042 5014873051 6714572333 5801041536 7142759197 57
23795770 4990802841 7073916641 4838534301 4597832541 1125216280 6598903557 1527356415 1426336514 7975966742 5701051931 5426785770 6566654462 4695812848 1657724752 4395862
942 1929266385 6196973552 1424386516 1228356316 7672916045 5602001331 4795852746 6015522734 5225745171 4795892043 4028995229 4891822748 77498666 6412532032 7949779696 567
0716741 5653529501 1028316712 4195882942 77488861 8627096850 6910572234 5363357232 8729096153 4794892043 6418532030 5026715671 4099812640 7345749595 5452549502 5622775178
 6717552930 5170736645 5601041438 5857589305 7044769799 4695822349 4930544908 7179996344 4937484775 4897852141 1624306615 8829026752 5857509400 4998852348 6297983957 5324
755175 5854529900 4737484674 4892822445 1525366813 7075946645 4096862643 6514582131 5206001130 1728386411 5455589207 4537414773 4727975929 5360337236 1429306315 419089234
0 79448764 6310592136 1127306815 8826066451 2970903318 4293872148 6718512936 6765674567 5650539602 6960634067 4695852645 6210542330 4233504100 4096852644 1826386314 81270
26451 5552569500 4296842943 1523206185 6890923451 1129316217 1021306713 7872996345 5701011030 4095892249 6213502234 5728775879 4490842440 5333474827 4535494970 5859559507
 5124745079 5321735871 5855599404 7146789993 4099832146 1626236589 7997891638 4995842540 1902460002 7871916044 6898953551 5928745171 5050549709 4496832648 5462397433 5827
428275 6619532834 5067337439 5118803651 4891842043 5829745772 7676906542 1105400501 5255539804 4994802540 4925905822 5165357838 1423356414 6916552530 7779976843 510005183
7 4897812944 1903400402 6719552335 5575706449 8129096858 1021376118 4190872248 1724226084 5757599001 4198892643 5823735876 1125366612 4621905624 4936444771 1224336912 535
1509609 7441749490 6762684768 4494802545 4840335518 8026066956 5350569405 7397871732 4794852447 5167327632 4127975028 5000071630 4699842040 4926915128 5224478274 56297355
70 7872966148 4497882748 5265357836 7275956248 5003061030 1722366510 4024975924 6416532037 5707071638 4895862543 1927356917 5151509901 2068614185 1522316219 4391862341 47
23985727 5903071432 7842769396 4997822646 6112582435 1508480108 4622925621 5170776941 5958579306 5820755871 4291862848 5926438277 6311542535 5718853059 5157529308 4899882
742 5823498879 4723925420 1426266283 5757539509 5027438173 5824755079 6869644765 4399892146 24478224 8820006558 6011592834 5325478370 5454519703 4295882740 5925468475 462
1945728 1427206387 5551519108 5727418175 5426715079 4694802448 5362327136 4325935222 5604001232 4596892046 7440709492 6911532730 5525735472 5338464120 5429418573 46219257
28 7695821730 4190862941 1528316217 5151569801 2765654889 1621386619 4894852746 6014512533 5306011538 4299842847 622143 4224935020 4932404877 6713582430 7478906841 679495
3653 5226715271 4299872745 4239534807 7875996546 5805031138 1525306716 5525715471 2376933912 4192832443 4528965029 4898802741 1828236481 6291953951 1525366216 1427366411
7673956846 5107011838 4698852648 5260337735 4923965521 5103081737 4693862645 7871936543 5803061233 5927418578 7990891930 4397802843 7246729993 6715522936 5320745777 51364
64325 5122478371 4621915025 7593841533 4191892345 1429306811 5850579200 2061694288 1620326110 4690872240 6613502433 5807041239 4893872441 4926955821 4399842249 5026765673
 6410542334 5106001236 5772776948 5224458675 5154509702 4090882041 4137574504 7874906346 5503091137 1224326616 6869644766 4593842748 4143385312 8420076457 5754529203 4899
882441 1921376111 5254529801 2464694780 1524326611 4498802342 7175936446 4038524007 4993882142 4123995828 4599892441 1926256289 6190973954 1729326013 1026346718 727999674
4 5506081637 4699822442 5561357137 4325945622 5606091738 4599822844 5223765677 5939404625 4224965128 5100051331 4996872940 1604430108 7474906940 4939414071 5958539507 429
7852442 1024386514 8125006452 4020955423 5308001337 4491812040 7877966246 5405061033 5950559907 4295852442 5425498977 6415552737 5407011333 5659559302 6368674068 45958928
45 3263201779 3563241077 1357784150 4296882940 8227002806 7791871934 1121376819 8926066354 7971996041 5304091937 4399842447 4639584008 6698933355 5803011932 5662357238 12
29366116 6811522632 7870916143 5100011638 4899852341 7072906841 4532414974 4094852842 1806470602 5950549006 1927356111 8224076556 7076996343 7041709798 4097852745 5364307
737 4423945427 5507091135 4397882643 1526206488 5453559908 4696862845 4923935329 5622705478 5028725373 7972936542 5666367733 6410512332 4721915622 1024366715 5752509507 7
845799091 4790892343 78438861 6419582439 1023346410 8521066352 4696802047 4129955623 4594872944 1321236684 6492953651 1327336815 1424366717 7271956146 5707081930 64656041
66 4798832440 4142395313 8822006151 6717502935 5022745775 4899842042 4331504308 6093913451 5301081539 5864387932 1627366913 6114522134 7071916140 5104041039 4296842246 74
73926345 4737474871 4896862042 1906400304 5054519100 1423326214 8928046455 7178986540 7148749098 4897842440 79458362 6514552238 5526438776 5923458875 4694802546 102928668
9 5354599803 4194832040 5052529204 2764684989 5154549400 5562327835 6792963155 1822336414 5250509904 7445789997 2470913210 4199882144 6717562331 4431514804 4498822048 162
2306510 8120076052 5553589309 4693872040 1024216483 6095963757 1823386615 1021376613 7279906940 5006071633 4992832945 6910502936 5022705877 4596852147 5938494326 44364142
73 5759569002 5929775475 5523715473 5153539900 7047739790 4596822348 6011582135 5500051337 4996852346 5229725370 7478926047 1700460209 5159599609 4994822647 76458369 4328
995224 7394841334 6960654269 3662251574 3569291875
~
~
~
~
~
~
~
~
~
~
~
~
                                                                                                                                   1,1           All
```

THE DECRYPTED DATA THAT IS SAME AS THE TO BE ENCRYPTED DATA



```
The Button widget is a standard Tkinter widget, which is used for various kinds of bu
ttons. A button is a widget which is designed for the user to interact with, i.e. if
the button is pressed by mouse click some action might be started. They can also cont
ain text and images like labels. While labels can display text in various fonts, a bu
tton can only display text in a single font. The text of a button can span more than
one line.

A Python function or method can be associated with a button. This function or method
will be executed, if the button is pressed in some way.
~
~
~
~
~
~
~
~
~
~
~
~
~
"Decrypted.txt" 4L, 579C                                          1,1           All
```

STEPS TAKEN:

1.GENERATION OF KEYS:

Keys were generated using the above algorithm. Choosing random numbers based on conditions was done by making a list of numbers that satisfy the condition and then apply the random function.

2.ENCRYPTION:

The text to be encrypted is taken character by character and its ASCII value is taken as the message m. To obtain the ASCII values the "letters" dictionay which maps characters to their ASCII values.
The encrypted text is then obtained as:

$c = m^e \bmod n$

Where:
c=Encrypted text/cipher text
e=public key exponent
n=modulus for both oublic and private key

3.DECRYPTION

The number to be decrypted (obtained after removing the padding) is taken as the ciphertext c.
The original message is then obtained as:

$m = c^d \bmod n$

Where:
m=original message (Gives the ASCII value of the character)
d=private key exponent
n=modulus for both oublic and private key

To obtain the letters from the ASCII values the "numbers" dictionay is used which maps ASCII values to corresponding characters.

4.PADDING:

As ASCII values are used in encryption hence every alphabet gets encrypted to the same number at every occcurence. To prevent it we use padding.

For encryption:
Now the encrypted number is sent to another method in which a random numberis added in every alternate position and the new encrypted number is returned.

For decryption:
In the number read from the encrypted file, digits appearing in alternate positions are deleted. The resultant number is sent to the decrypt method.

5.FILE HANDLING

The text to be encrypted is in toBeEncrypted.txt
The encrypted text is stored in encrypted.txt
The decrypted text is stored in Decrypted.txt

6.FUNCTIONS used to perform mathematical operations and what it does:

1.GCD
2.PRIME NUMBER
3.MODULAR EXPONENTIATION

For large value of m and e finding m^e directly is difficult. Thus this function finds modular exponetiation through a recursive function.
It uses the concept that (a.b)mod c=(a mod c * b mod c) mod c
To calculate a^bmod n

base case:
if a is 0 returns 0
if b is 0 returns 1

if b is even:
keep breaking it down as $a^{b/2}$* $a^{b/2}$ till (a mod n) is obtained

if b is odd:
break as a^{b-1}*a so that (b-1) becomes even

CONCLUSION

1.THINGS THAT CAN BE IMPROVED:

The concept of signing messages can be added.
Also the private key can be further encrypted and stored to enchance robustness.

2.WHAT WE ACHIEVED AND LEARNT:

Though it seemed very difficult in the beginning, we ended up learning a lot about cryptosystems and finally implemented the RSA algorithm.

We were able to encrypt only numbers initially, but now we can encrypt any character in the ASCII table.

We learnt a lot of modular mathematics, file handling in python ,generating random numbers based on conditions,
and also a bit about GUI tkinter.

3.REFERENCES:

https://simple.wikipedia.org/wiki/RSA_algorithm.

https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

github:RSA implementation through GUI Tkinter