

# TARUN KUMAR SURYAVANSHI

## 2019163

Aes.py:

```
File Edit Selection View Go Run Terminal Help aes.py - aes algo - Visual Studio Code
client.py aes.py server.py
aes.py > ...
1 # TARUN KUMAR SURYAVANSHI
2 # 2019163
3
4 # STRING TO BITS-STRING
5 def createbits(msg):
6     print(msg)
7     res = ''.join(format(ord(i), '08b') for i in msg)
8     return res
9
10 # GENERATE SUBNIB WITH S-BOX
11 def subnib(w):
12     w = str(w)
13     if w == "0000": return "1001"
14     if w == "0001": return "0100"
15     if w == "0010": return "1010"
16     if w == "0011": return "1011"
17     if w == "0100": return "1101"
18     if w == "0101": return "0001"
19     if w == "0110": return "1000"
20     if w == "0111": return "0101"
21     if w == "1000": return "0110"
22     if w == "1001": return "0010"
23     if w == "1010": return "0000"
24     if w == "1011": return "0011"
25     if w == "1100": return "1100"
26     if w == "1101": return "1110"
27     if w == "1110": return "1111"
28     if w == "1111": return "0111"
29
30 # GENERATE SUBNIB WITH INVERSE S-BOX
31 def subnib_inv(w):
32     w = str(w)
33     if w == "1001": return "0000"
34     if w == "0100": return "0001"
35     if w == "1010": return "0010"
36     if w == "1011": return "0011"
37     if w == "1101": return "0100"
38     if w == "0001": return "0101"
39     if w == "1000": return "0110"
40     if w == "0010": return "0111"
41     if w == "0000": return "1000"
42     if w == "0011": return "1001"
43     if w == "0110": return "1010"
44     if w == "0111": return "1011"
45     if w == "1110": return "1100"
46     if w == "1111": return "1101"
47     if w == "1100": return "1110"
48     if w == "1101": return "1111"
49
50 # XOR OR THREE BITS STRING
51 def xor(s1,s2,s3):
52     s = ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s1,s2))
53     return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s,s3))
54
55 # MIX COLUMN FUNCTION FOR ENCREPTION
56 def mixcolumn(m):
57     m = m[:4]+m[8:12]+m[4:8]+m[12:16]
58     pt = ''
59     pt = pt + str(int(m[0])^int(m[10])) + str(int(m[1])^int(m[4])^int(m[7])) + str(int(m[2])^int(m[4])^int(m[5])) + str(int(m[3])^int(m[5]))
60     pt = pt + str(int(m[8])^int(m[14])) + str(int(m[9])^int(m[12])^int(m[15])) + str(int(m[10])^int(m[12])^int(m[13])) + str(int(m[11])^int(m[13]))
61     pt = pt + str(int(m[2])^int(m[4])) + str(int(m[0])^int(m[3])^int(m[5])) + str(int(m[0])^int(m[1])^int(m[6])) + str(int(m[1])^int(m[7]))
62     pt = pt + str(int(m[10])^int(m[12])) + str(int(m[8])^int(m[11])^int(m[13])) + str(int(m[8])^int(m[9])^int(m[14])) + str(int(m[9])^int(m[15]))
63     pt = pt[:4]+pt[8:12]+pt[4:8]+pt[12:16]
64     return pt
65
66 # MIX COLUMN FUNCTION FOR DECREPTION
67 def imixcolumn(m):
68     m = m[12:16]+m[4:8]+m[8:12]+m[0:4]
69     pt = ''
70     pt = pt + str(int(m[0])^int(m[10])) + str(int(m[1])^int(m[4])^int(m[7])) + str(int(m[2])^int(m[4])^int(m[5])) + str(int(m[3])^int(m[5]))
71     pt = pt + str(int(m[8])^int(m[14])) + str(int(m[9])^int(m[12])^int(m[15])) + str(int(m[10])^int(m[12])^int(m[13])) + str(int(m[11])^int(m[13]))
72     pt = pt + str(int(m[2])^int(m[4])) + str(int(m[0])^int(m[3])^int(m[5])) + str(int(m[0])^int(m[1])^int(m[6])) + str(int(m[1])^int(m[7]))
73     pt = pt + str(int(m[10])^int(m[12])) + str(int(m[8])^int(m[11])^int(m[13])) + str(int(m[8])^int(m[9])^int(m[14])) + str(int(m[9])^int(m[15]))
74     pt = pt[12:16]+pt[4:8]+pt[8:12]+pt[0:4]
75     return pt
```

```
File Edit Selection View Go Run Terminal Help aes.py - aes algo - Visual Studio Code
client.py aes.py server.py
aes.py > ...
48     if w == "1111": return "1110"
49
50 # XOR OR THREE BITS STRING
51 def xor(s1,s2,s3):
52     s = ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s1,s2))
53     return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s,s3))
54
55 # MIX COLUMN FUNCTION FOR ENCREPTION
56 def mixcolumn(m):
57     m = m[:4]+m[8:12]+m[4:8]+m[12:16]
58     pt = ''
59     pt = pt + str(int(m[0])^int(m[10])) + str(int(m[1])^int(m[4])^int(m[7])) + str(int(m[2])^int(m[4])^int(m[5])) + str(int(m[3])^int(m[5]))
60     pt = pt + str(int(m[8])^int(m[14])) + str(int(m[9])^int(m[12])^int(m[15])) + str(int(m[10])^int(m[12])^int(m[13])) + str(int(m[11])^int(m[13]))
61     pt = pt + str(int(m[2])^int(m[4])) + str(int(m[0])^int(m[3])^int(m[5])) + str(int(m[0])^int(m[1])^int(m[6])) + str(int(m[1])^int(m[7]))
62     pt = pt + str(int(m[10])^int(m[12])) + str(int(m[8])^int(m[11])^int(m[13])) + str(int(m[8])^int(m[9])^int(m[14])) + str(int(m[9])^int(m[15]))
63     pt = pt[:4]+pt[8:12]+pt[4:8]+pt[12:16]
64     return pt
65
66 # MIX COLUMN FUNCTION FOR DECREPTION
67 def imixcolumn(m):
68     m = m[12:16]+m[4:8]+m[8:12]+m[0:4]
69     pt = ''
70     pt = pt + str(int(m[0])^int(m[10])) + str(int(m[1])^int(m[4])^int(m[7])) + str(int(m[2])^int(m[4])^int(m[5])) + str(int(m[3])^int(m[5]))
71     pt = pt + str(int(m[8])^int(m[14])) + str(int(m[9])^int(m[12])^int(m[15])) + str(int(m[10])^int(m[12])^int(m[13])) + str(int(m[11])^int(m[13]))
72     pt = pt + str(int(m[2])^int(m[4])) + str(int(m[0])^int(m[3])^int(m[5])) + str(int(m[0])^int(m[1])^int(m[6])) + str(int(m[1])^int(m[7]))
73     pt = pt + str(int(m[10])^int(m[12])) + str(int(m[8])^int(m[11])^int(m[13])) + str(int(m[8])^int(m[9])^int(m[14])) + str(int(m[9])^int(m[15]))
74     pt = pt[12:16]+pt[4:8]+pt[8:12]+pt[0:4]
75     return pt
```

The screenshot shows the Visual Studio Code editor with the 'aes.py' file open. The file contains a Python script for an AES algorithm. The terminal output shows the execution of the script, which prompts the user to enter values for p, q, e, a public key, and a secret key. It then displays the results of the encryption and decryption process, including the message, its digest, and the client signature.

```
1 # TARUN KUMAR SURYAVANSHI
2 # 2019163
3
4 # STRING TO BITS-STRING
5 def createbits(msg):
6     print(msg)
7     res = ''.join(format(ord(i), '08b') for i in msg)
8     return res
9
10 # GENERATE SUBNIB WITH S-BOX
```

enter the value of p :5  
enter the value of q :11  
enter the value of e (1 < e <= {phin}) :13  
give public key  
['1111101100101010', '11', '9', '7']  
secret key : 11  
11  
key is in bits 0011000100110001  
keys : 0011000100110001 1111101011001011 1111011000111101  
round key 2 add ropund 0000110100010111  
round 2 shift row 000001100011101  
round 2 nibble sub 1010111101010100  
add round 1 key 0101010110011111  
round 1 mix column 0111111011001000  
round 1 shift row 0111100011001110  
round 1 nibble sub 1111011011001101  
message is binary 1100011111111100  
message is decrypted r s  
msg digest 8523967897401273671  
client signature : 51  
PS C:\Users\User\OneDrive\Desktop\aes algo>

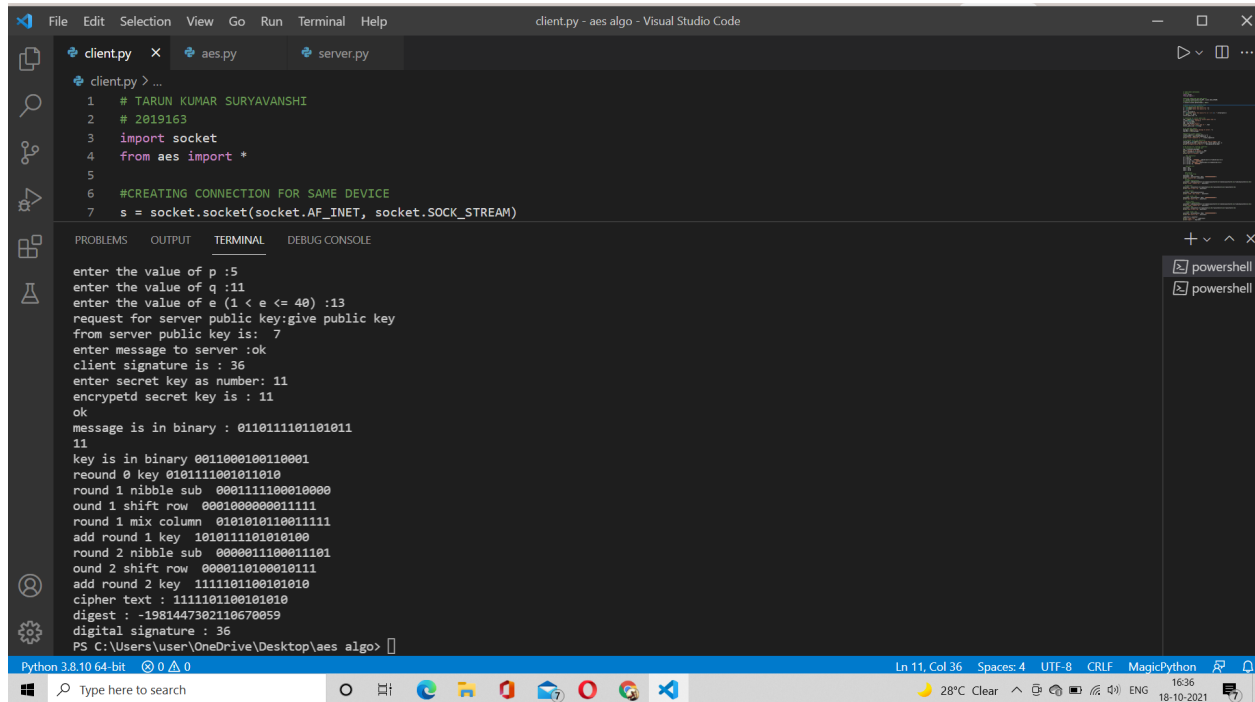
## Server.py :

The screenshot shows the Visual Studio Code editor with the 'server.py' file open. The file contains a Python script for a server that listens for a connection and processes a message. The terminal output shows the execution of the script, which prompts the user to enter values for p, q, e, a public key, and a secret key. It then displays the results of the encryption and decryption process, including the message, its digest, and the client signature.

```
1 # TARUN KUMAR SURYAVANSHI
2 # 2019163
3 import socket
4 from aes import *
5
6 #CREATING CONNECTION FOR SAME DEVICE
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 #CONNECTING WITH PORT NO 1024
9 s.bind((socket.gethostname(), 1024))
10 s.listen(1)
```

enter the value of p :5  
enter the value of q :11  
enter the value of e (1 < e <= {phin}) :13  
give public key  
['1111101100101010', '11', '9', '7']  
secret key : 11  
11  
key is in bits 0011000100110001  
keys : 0011000100110001 1111101011001011 1111011000111101  
round key 2 add ropund 0000110100010111  
round 2 shift row 000001100011101  
round 2 nibble sub 1010111101010100  
add round 1 key 0101010110011111  
round 1 mix column 0111111011001000  
round 1 shift row 0111100011001110  
round 1 nibble sub 1111011011001101  
message is binary 1100011111111100  
message is decrypted r s  
msg digest 8523967897401273671  
client signature : 51  
PS C:\Users\User\OneDrive\Desktop\aes algo>

# Client.py



The image shows a Visual Studio Code editor window with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar reads 'client.py - aes algo - Visual Studio Code'. The editor has three tabs: 'client.py', 'aes.py', and 'server.py'. The 'client.py' tab is active, showing a Python script with the following code:

```
1 # TARUN KUMAR SURYAVANSHI
2 # 2019163
3 import socket
4 from aes import *
5
6 #CREATING CONNECTION FOR SAME DEVICE
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Below the editor, the 'TERMINAL' tab is active, displaying the output of the script. The output shows a series of prompts and responses, including binary data and a final digest and signature. The status bar at the bottom indicates 'Python 3.8.10 64-bit', 'Ln 11, Col 36', 'Spaces: 4', 'UTF-8', 'CRLF', 'MagicPython', and the date '18-10-2021'.

```
enter the value of p :5
enter the value of q :11
enter the value of e (1 < e <= 40) :13
request for server public key:give public key
from server public key is: 7
enter message to server :ok
client signature is : 36
enter secret key as number: 11
encryptd secret key is : 11
ok
message is in binary : 01101111101101011
11
key is in binary 0011000100110001
round 0 key 0101111001011010
round 1 nibble sub 000111100010000
round 1 shift row 000100000011111
round 1 mix column 0101010110011111
add round 1 key 1010111101010100
round 2 nibble sub 000001100011101
round 2 shift row 0000110100010111
add round 2 key 1111101100101010
cipher text : 1111101100101010
digest : -1981447302110670059
digital signature : 36
PS C:\Users\user\OneDrive\Desktop\aes algo>
```