

# Lab Report: Implementation of RSA Algorithm with TCP Client/Server Sockets in Python

## Purpose

The purpose of this lab is to:

1. Write a program to implement the RSA algorithm with a key size of 1024 bits.
2. Demonstrate the encryption and decryption process using a Client/Server application over TCP. The server generates the public and private keys and sends the public key to the client via a TCP socket. The client uses the public key to encrypt a message and send the ciphertext back to the server for decryption.

## Methods

The implementation consists of two main classes: Server and Client.

- **Server:**
  - Generates RSA public and private keys.
  - Sends the public key to the client through a TCP socket.
  - Receives ciphertext from the client and decrypts it using the private key.
- **Client:**
  - Receives the public key from the server through a TCP socket.
  - Encrypts a message using the received public key and sends the ciphertext to the server.

## Steps:

1. **Key Generation:**
  - The server generates two large prime numbers  $p$  and  $q$ .
  - Computes  $n = p * q$  and the Euler's totient function  $\phi(n) = (p-1) * (q-1)$ .
  - Chooses a public exponent  $e = 65537$  and calculates the private exponent  $d$  such that  $e * d \equiv 1 \pmod{\phi(n)}$ .
2. **Encryption and Decryption:**
  - To encrypt a message  $M$  with the public key, compute the ciphertext  $C$  using  $C = M^e \bmod n$ .
  - To decrypt the ciphertext  $C$  with the private key, compute the plaintext  $M$  using  $M = C^d \bmod n$ .

## Implementation

The code below demonstrates the Server and Client implementation in Python using sockets and the RSA algorithm.

## Server Code(Alice)

```
python
Copy code
import socket
import signal
```

```

from Crypto.Util.number import getPrime, inverse, bytes_to_long,
long_to_bytes

# RSA Key generation
def generate_rsa_keys(key_size=1024):
    e = 65537
    p = getPrime(key_size // 2)
    q = getPrime(key_size // 2)
    n = p * q
    phi_n = (p - 1) * (q - 1)
    d = inverse(e, phi_n)
    return (e, d, n, p, q, phi_n), (e, n)

# Encryption and decryption functions
def encrypt(m, e, n):
    return pow(bytes_to_long(m), e, n)

def decrypt(c, d, n):
    return long_to_bytes(pow(c, d, n))

HOST = '127.0.0.1'
PORT = 65432

server_private_key, server_public_key = generate_rsa_keys()
(e, d, n, p, q, phi_n) = server_private_key
(e, n) = server_public_key

print(f"Server's Public key: (e={e}, n={n})")

def handle_interrupt(sig, frame):
    print("Server shutting down...")
    server_socket.close()
    exit(0)

signal.signal(signal.SIGINT, handle_interrupt)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
    server_address = (HOST, PORT)
    print('Server listening on %s port %s' % server_address)
    server_socket.bind(server_address)
    server_socket.listen()

    while True:
        print("Waiting for a connection...")
        conn, addr = server_socket.accept()
        with conn:
            print('Connected by', addr)

            # Exchange public keys
            client_public_key = conn.recv(1024).decode()
            conn.sendall(f"{e},{n}".encode())

            client_e, client_n = map(int, client_public_key.split(','))
            print(f"Client's Public key: (e={client_e}, n={client_n})")

            while True:
                # Receive encrypted data from client
                encrypted_data = conn.recv(1024).decode()
                if not encrypted_data:
                    print("Client disconnected.")
                    break

```

```

print(f"Received encrypted message: {encrypted_data}")
decrypted_data = decrypt(int(encrypted_data), d, n)
message = decrypted_data.decode()
print(f"Decrypted message: {message}")

if message.lower() == 'quit':
    print("Client requested to quit. Closing connection.")
    break

# Send a response
response = input("Enter your message (or 'quit' to exit):")

encrypted_response = encrypt(response.encode(), client_e,
client_n)

print(f"Sending encrypted message: {encrypted_response}")
conn.sendall(str(encrypted_response).encode())

if response.lower() == 'quit':
    print("Closing connection.")
    break

print("Connection closed. Waiting for new connections...")

```

## Server Output

The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays the RSA server code, which includes key generation, encryption, and decryption functions. The terminal shows the output of the server, including the public key, the server listening on port 65432, and the receipt of a client connection from 127.0.0.1:58277.

```

1 import socket
2 import signal
3 from Crypto.Util.number import getPrime, inverse, bytes_to_long, long_to_bytes
4
5 # RSA Key generation
6 def generate_rsa_keys(key_size=1024):
7     e = 65537
8     p = getPrime(key_size // 2)
9     q = getPrime(key_size // 2)
10    n = p * q
11    phi_n = (p - 1) * (q - 1)
12    d = inverse(e, phi_n)
13    return (e, d, n, p, q, phi_n), (e, n)
14
15 # Encryption and decryption functions
16 def encrypt(m, e, n):
17     return pow(bytes_to_long(m), e, n)
18
19 def decrypt(c, d, n):
20     return long_to_bytes(pow(c, d, n))
21
22 HOST = '127.0.0.1'
23 PORT = 65432
24
25 server_private_key, server_public_key = generate_rsa_keys()
26 (e, d, n, p, q, phi_n) = server_private_key
27 (e, n) = server_public_key
28
29 print(f"Server's Public key: (e={e}, n={n})")
30
31 def handle_interrupt(sig, frame):
32     print("Server shutting down...")
33
34 signal.signal(signal.SIGINT, handle_interrupt)
35
36 if __name__ == '__main__':
37     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
38     s.bind((HOST, PORT))
39     s.listen(1)
40     print(f"Server listening on {HOST} port {PORT}")
41     while True:
42         conn, addr = s.accept()
43         print(f"Waiting for a connection...")
44         print(f"Connected by {addr[0]}: {addr[1]}")
45         client_e, client_n = generate_rsa_keys()
46         client_public_key = (client_e, client_n)
47         print(f"Client's Public key: (e={client_e}, n={client_n})")
48         message = input("Enter your message (or 'quit' to exit):")
49         encrypted_message = encrypt(message.encode(), client_e, client_n)
50         conn.sendall(str(encrypted_message).encode())
51         if message.lower() == 'quit':
52             print("Closing connection.")
53             break
54         response = input("Enter your message (or 'quit' to exit):")
55         encrypted_response = encrypt(response.encode(), client_e, client_n)
56         conn.sendall(str(encrypted_response).encode())
57         if response.lower() == 'quit':
58             print("Closing connection.")
59             break
60     print("Connection closed. Waiting for new connections...")

```

```

PS C:\Users\coll\Downloads> python -u "C:\Users\coll\Downloads\RSAServer.py"
Server's Public key: (e=65537, n=25683598528299122298464468759089968655748861475358652732734441970259788138622803278914874797747357372985541528747815648626799714935443186857728765437889892287256373191385
122740759611528624278811951853252808655978080236038769457318815648866888995536182677837697997714442544788824955183988835685475613)
Server listening on 127.0.0.1 port 65432
Waiting for a connection...
Connected by ('127.0.0.1', 58277)
Client's Public key: (e=65537, n=13955874882393185832765785289475288465340886478866320636546852568856682875148872883561882385475639418955751641065863968158714248668228329429846631972845688389353854815799
7261836634997346927723618572783782687518118819349188918866687876315688187945625713582474258935574197588819404133798023422597648639)

```



```

    q = getPrime(key_size // 2)
    n = p * q
    phi_n = (p - 1) * (q - 1)
    d = inverse(e, phi_n)
    return (e, d, n, p, q, phi_n), (e, n)

# Encryption and decryption functions
def encrypt(m, e, n):
    return pow(bytes_to_long(m), e, n)

def decrypt(c, d, n):
    return long_to_bytes(pow(c, d, n))

HOST = '127.0.0.1'
PORT = 65432

client_private_key, client_public_key = generate_rsa_keys()
(e, d, n, p, q, phi_n) = client_private_key
(e, n) = client_public_key

print(f"Client's Public key: (e={e}, n={n})")

try:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        server_address = (HOST, PORT)
        print(f'Connecting to {s} port {s}' % server_address)
        s.connect(server_address)

        # Exchange public keys
        s.sendall(f"{e},{n}".encode())
        server_public_key = s.recv(1024).decode()
        server_e, server_n = map(int, server_public_key.split(','))

        print(f"Server's Public key: (e={server_e}, n={server_n})")

        while True:
            # Send message to server
            message = input("Enter your message (or 'quit' to exit): ")
            encrypted_message = encrypt(message.encode(), server_e,
server_n)
            print(f"Sending encrypted message: {encrypted_message}")
            s.sendall(str(encrypted_message).encode())

            if message.lower() == 'quit':
                print("Closing connection.")
                break

            # Receive response from server
            encrypted_response = s.recv(1024).decode()
            if not encrypted_response:
                print("Server disconnected.")
                break

            print(f"Received encrypted message: {encrypted_response}")
            decrypted_response = decrypt(int(encrypted_response), d, n)
            print(f"Decrypted message: {decrypted_response.decode()}")

            if decrypted_response.decode().lower() == 'quit':
                print("Server requested to quit. Closing connection.")
                break

```

```

except ConnectionError:
    print('Failed to connect to the server.')
except KeyboardInterrupt:
    print('Client interrupted.')

print("Connection closed.")

```

## Client Output

```

1 import socket
2 from Crypto.Util.number import getPrime, inverse, bytes_to_long, long_to_bytes
3
4 # RSA Key generation
5 def generate_rsa_keys(key_size=1024):
6     e = 65537
7     p = getPrime(key_size // 2)
8     q = getPrime(key_size // 2)
9     n = p * q
10    phi_n = (p - 1) * (q - 1)
11    d = inverse(e, phi_n)
12    return (e, d, n, p, q, phi_n), (e, n)
13
14 # Encryption and decryption functions
15 def encrypt(m, e, n):
16     return pow(bytes_to_long(m), e, n)
17
18 def decrypt(c, d, n):
19     return long_to_bytes(pow(c, d, n))
20
21 HOST = '127.0.0.1'
22 PORT = 65432
23
24 client_private_key, client_public_key = generate_rsa_keys()
25 (e, d, n, p, q, phi_n) = client_private_key
26 (e, n) = client_public_key
27
28 print(f"Client's Public key: (n={n}, e={e})")
29
30 try:
31     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
32         server_address = (HOST, PORT)

```

PS C:\Users\collin> python -u "C:\Users\collin\Downloads\VSClient.py"

Client's Public key: (n=65537, e=1395587488239118583276578528947752884653408886478006320536540825698566628751488728835618828385456394189557516418058639601587142409692283294298466319728456088389353854815799726183663499734692772306185727837626875181188193491180138066687261316881879456271335247425893537419758813404113759023422597648039)

Connecting to 127.0.0.1 port 65432

Server's Public key: (n=65537, e=56835985202912298464466750849960605748808147535865273273444197825978813862208327891487479774735737280541528747015648862679971493544318685772876543788089287256373191305122754075961352003627081195185375288845597880823683876945731881548860889955961826782769799771444244780824955183989835085475619)

Enter your message (or 'quit' to exit):

PS C:\Users\collin> python -u "C:\Users\collin\Downloads\VSClient.py"

Client's Public key: (n=65537, e=1395587488239118583276578528947752884653408886478006320536540825698566628751488728835618828385456394189557516418058639601587142409692283294298466319728456088389353854815799726183663499734692772306185727837626875181188193491180138066687261316881879456271335247425893537419758813404113759023422597648039)

Connecting to 127.0.0.1 port 65432

Server's Public key: (n=65537, e=56835985202912298464466750849960605748808147535865273273444197825978813862208327891487479774735737280541528747015648862679971493544318685772876543788089287256373191305122754075961352003627081195185375288845597880823683876945731881548860889955961826782769799771444244780824955183989835085475619)

Sending encrypted message: 81772427689648326159489184238835129421386248867661199759181879791735465388488914831222771779181511391777958616742854125846632897968887150364864955150036899279819282887896738068873952796155776135329646459133922413372161598348854699292757296576954883348893835321634967235919334273818125739351273282

```
File Edit Selection View Go Run Terminal Help
Welcome
C:\Users> cd C:\Downloads > RSAClient.py > ...
1 #socket
2 from Crypto.Util.number import getPrime, inverse, bytes_to_long, long_to_bytes
3
4 # RSA Key generation
5 def generate_rsa_keys(key_size=1024):
6     e = 65537
7     p = getPrime(key_size // 2)
8     q = getPrime(key_size // 2)
9     n = p * q
10    phi_n = (p - 1) * (q - 1)
11    d = inverse(e, phi_n)
12    return (e, d, n, p, q, phi_n), (e, n)
13
14 # Encryption and decryption functions
15 def encrypt(m, e, n):
16     return pow(bytes_to_long(m), e, n)
17
18 def decrypt(c, d, n):
19     return long_to_bytes(pow(c, d, n))
20
21 HOST = '127.0.0.1'
22 PORT = 65432
23
24 client_private_key, client_public_key = generate_rsa_keys()
25 (e, d, n, p, q, phi_n) = client_private_key
26 (e, n) = client_public_key
27
28 print(f"Client's Public key: (e={e}, n={n})")
29
30 try:
31     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
32         server_address = (HOST, PORT)
33
34 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
45696292754729867659485294805383521634067225931934275818125739351771282
Received encrypted message: 1105895544223183345146849441796486790391154556209994432753081181489296833828478248804562374575543475709165384654841898665412869053049972799813463903171927786213444237329574203941195351436260738118418786093198078946862965218195482
3748547878867936128848944731498617228096616882159174061902778283597113881
Decrypted message: I'm doing great, thanks for checking.
Enter your message (or 'quit' to exit): goodbye for now mate!
Sending encrypted message: 95587886882872088380812276805968968443832738484676700116636369595959342458794511238135794389132406672257641696878602267158545580576093847952866181425006708952623076155953902520896968063893798847713637718711949452718353478774839958
487813311513236852329597641368842401173899722176883278056914548307
Received encrypted message: 8814475808746845208843026491673355422261471833731104399442169768866190720865857458317189196821763976328135993561996526584215626568742894465977980938780849583835077528986889512744896581540526447138830047832051287286335931466848391721
89782338963672253137145188381106496104691518847390296822215727786875434481
Decrypted message: exit
Enter your message (or 'quit' to exit): exit
Sending encrypted message: 98831195535353426938093818163975283692357686121739172197337786571863802994448161561203140818144778489602189392658258381453688997955980780982897548444211962134646881714741472118899788000850588703169678948717356651468352080644916288459
4176897808652909645377621318776938659118414443378666292383172394
Received encrypted message: 83839783676686458212736902612913791277812855398668818634880753414751186152558888721132487573617659786710888952004781752971489157683557442182968780626527808058980766463628699989511397636839426795768967171945374823145627363522887688942
2842845959877637542888953274638489100636743733322299566341787644762886
Decrypted message: quit
Server requested to quit. Closing connection.
Connection closed.
PS C:\Users\collin> quit
```

## Conclusion

This lab demonstrated the implementation of the RSA algorithm and the establishment of a TCP Client/Server communication for encrypted message exchange. The server generates RSA keys, sends the public key to the client, and decrypts messages received from the client. The client uses the server's public key to encrypt messages and sends the ciphertext to the server. This approach ensures secure communication over the network using RSA encryption.