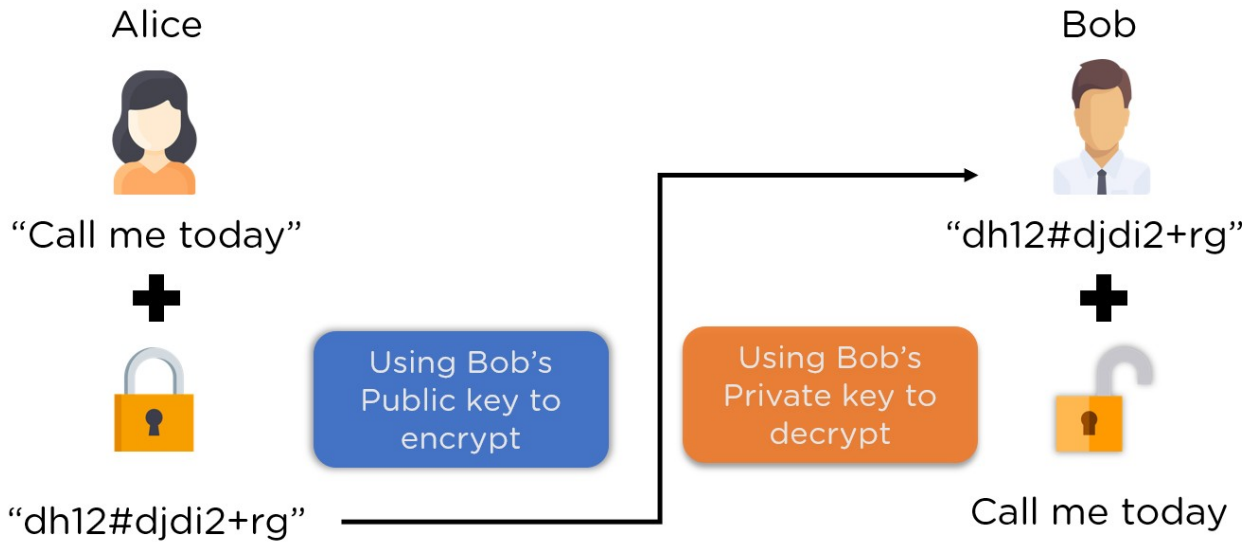
Server and Client Communication with RSA

PowerPoint Presentation in Information and Cybersecurity

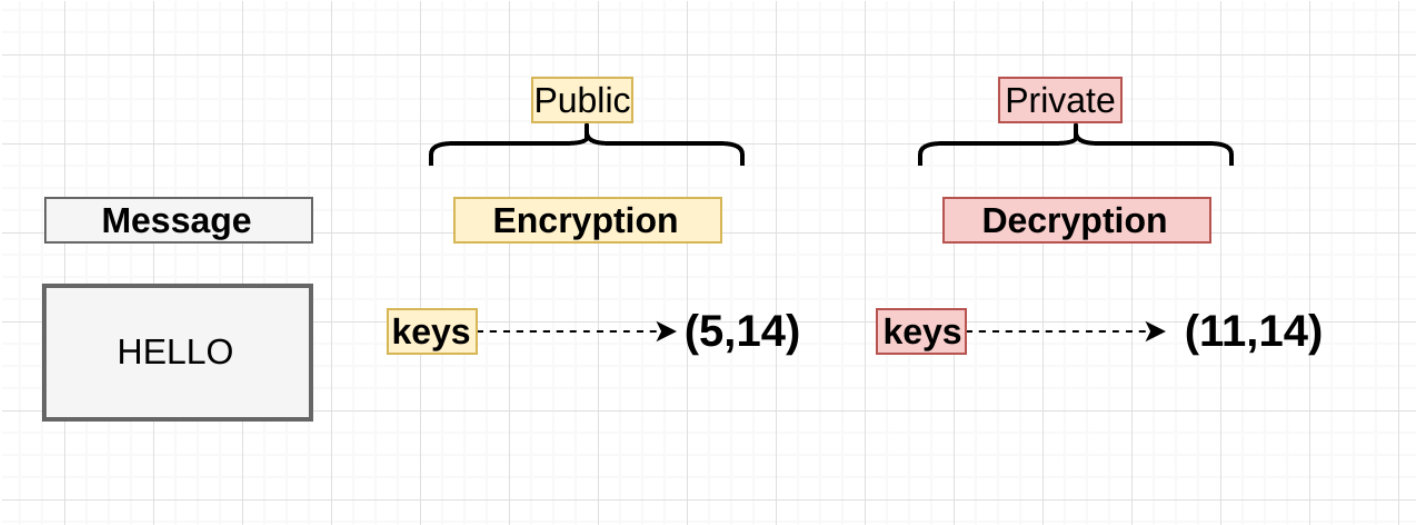
By
Krishnaraj Thadesar, PA20, Batch A1

Contents

Basic Working of Encrypted Chats with Client and Server



Simple Working of RSA



How the Keys looks like

Server Clients usually work with End to End Encryption

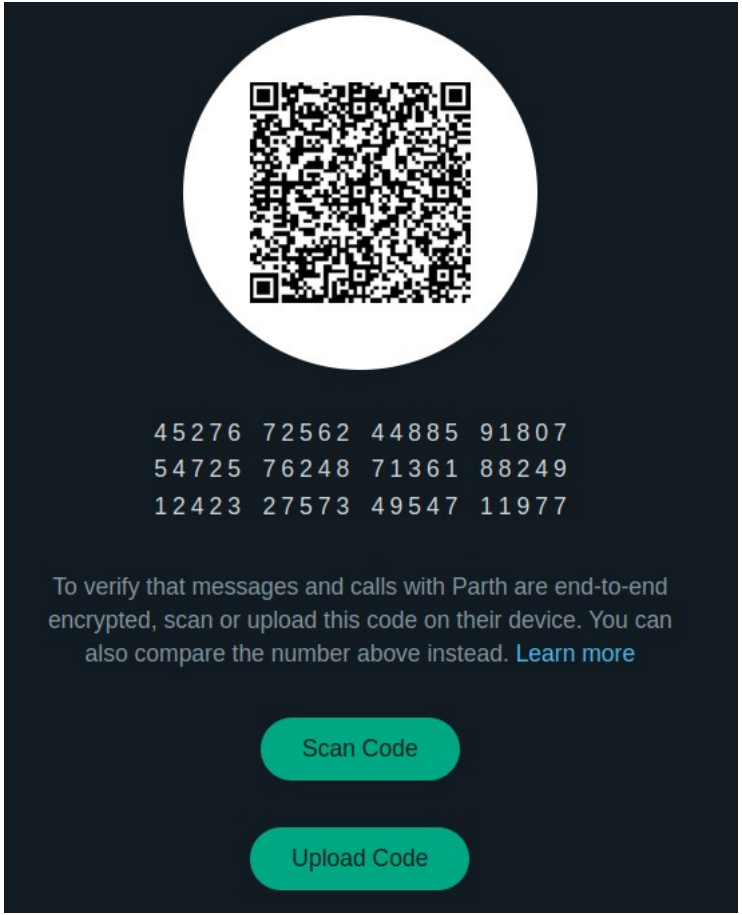
A good example of that is Signal, and Whatsapp, telegram etc.

WhatsApp’s E2E Encryption Protocol

WhatsApp uses open source *Signal Protocol* developed by Open Whisper Systems (They have their own messaging application, Signal). Signal Protocol uses primitives like Double Ratchet Algorithm, prekeys, Triple Diffie Hellman, Curve25519, AES and HMAC_SHA256.

What goes on in whatsapp - A simple case study

1. Session setup by Ankita using x3dh.
2. Ankita calculates master secret and Using DH Ratchet step, derives a root key and a chain key to be used by Hashing Ratchet.
3. Ankita derives a message key and next chain key using the chain key using Hashing Ratchet.
4. She encrypts her message using message key (AES256 in CBC mode).
5. Every time she sends a message her Hashing Ratchet moves forward.
6. Every time she receives a response from Bud, which includes a new public key in header, she advances her DH Ratchet, calculates a new root key and a new chain key.


45276 72562 44885 91807
54725 76248 71361 88249
12423 27573 49547 11977
To verify that messages and calls with Parth are end-to-end encrypted, scan or upload this code on their device. You can also compare the number above instead. Learn more
Scan Code
Upload Code

The QR code contains:

1. A version.
2. The user identifier for both parties.
3. The full 32-byte public Identity Key for all devices of both parties.

From Whatsapp Documentation

When either device scans the other's QR code, the keys are compared to ensure that what is in the QR code matches the Identity Key as retrieved from the server. The 60-digit number is computed by concatenating the two 30-digit numeric fingerprints for each user's device Identity Keys. To calculate a 30-digit numeric fingerprint:

1. Lexicographically sort public Identity Keys for all of the user's devices and concatenate them.
2. Iteratively SHA-512 hash the sorted Identity Keys and user identifier 5200 times.
3. Take the first 30 bytes of the final hash output.
4. Split the 30-byte result into six 5-byte chunks.
5. Convert each 5-byte chunk into 5 digits by interpreting each 5-byte chunk as a big-endian unsigned integer and reducing it modulo 100000.
6. Concatenate the six groups of five digits into thirty digits.

Our Simple Setup

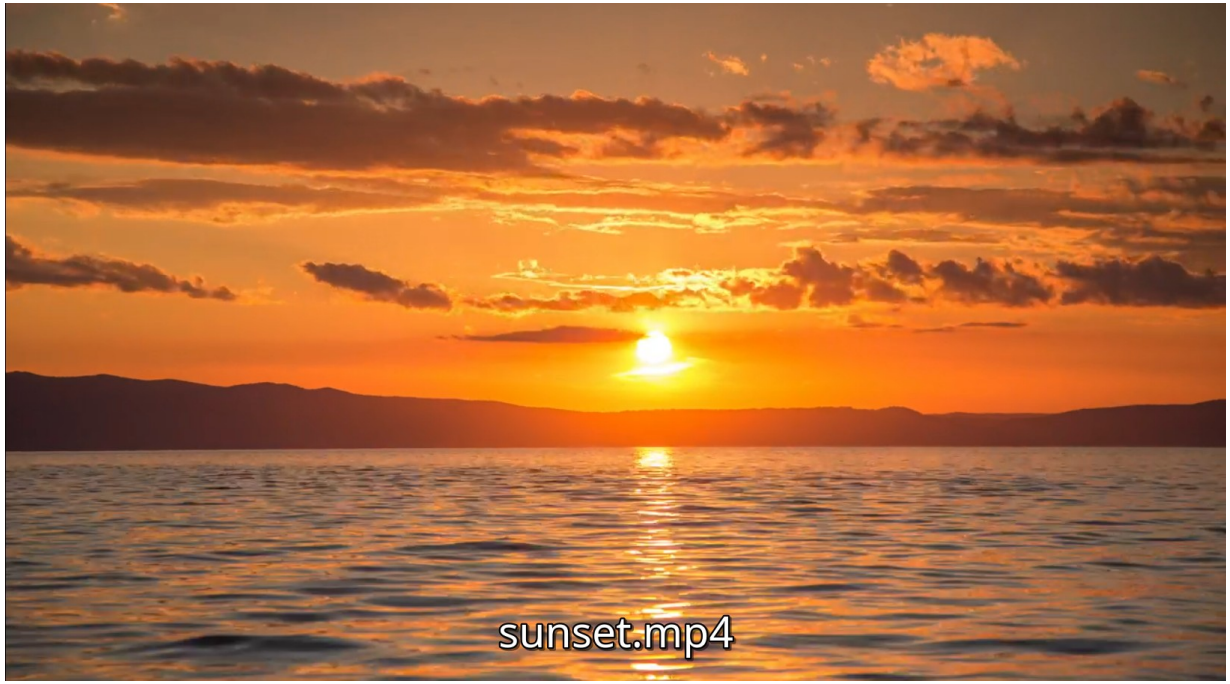
- We have used RSA to encrypt and decrypt messages between the client and the server.
- Their keys are shared using Diffie Hellman learnt and executed in a previous Assignment.
- They communicate with TCP/IP protocol using the socket inbuilt library in python.
- Communication is possible on any network, as long as host IP and port are known.

Our setup limitations

1. Only 1 client can talk to the server at one time. This can be improved by tweaking the code a little bit.
2. There is no means of reconnecting.
3. We can’t know if the server or client are both online or not once they go offline, so things like blue ticks and gray ticks would not work.

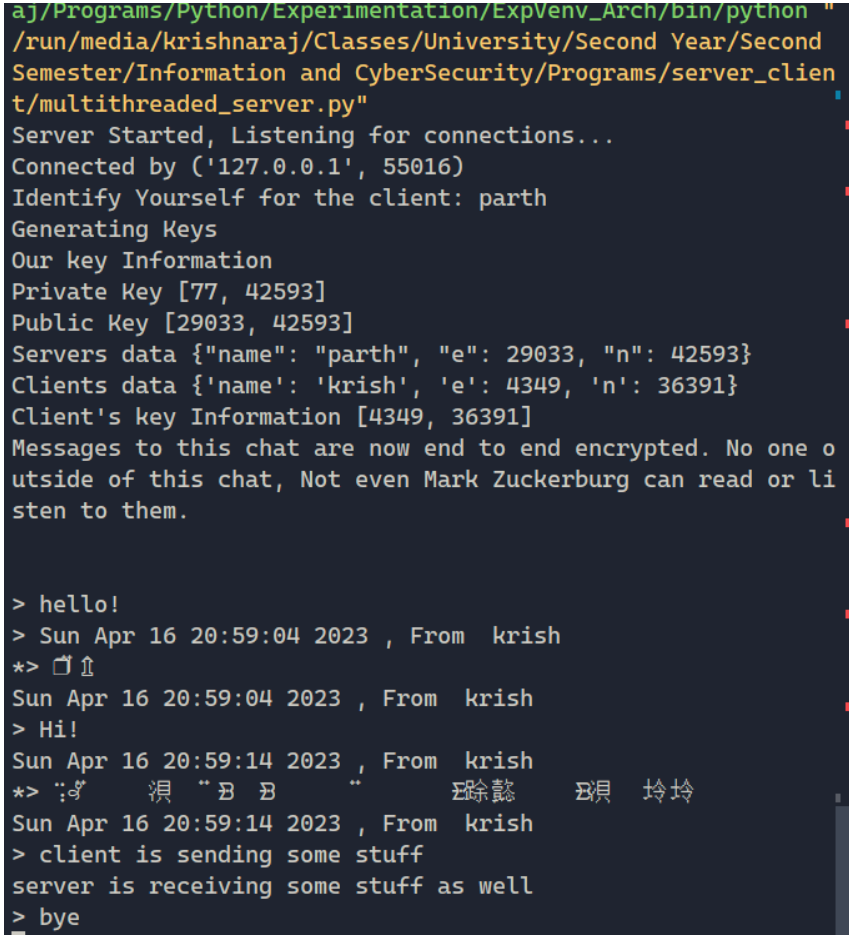
Code and Output

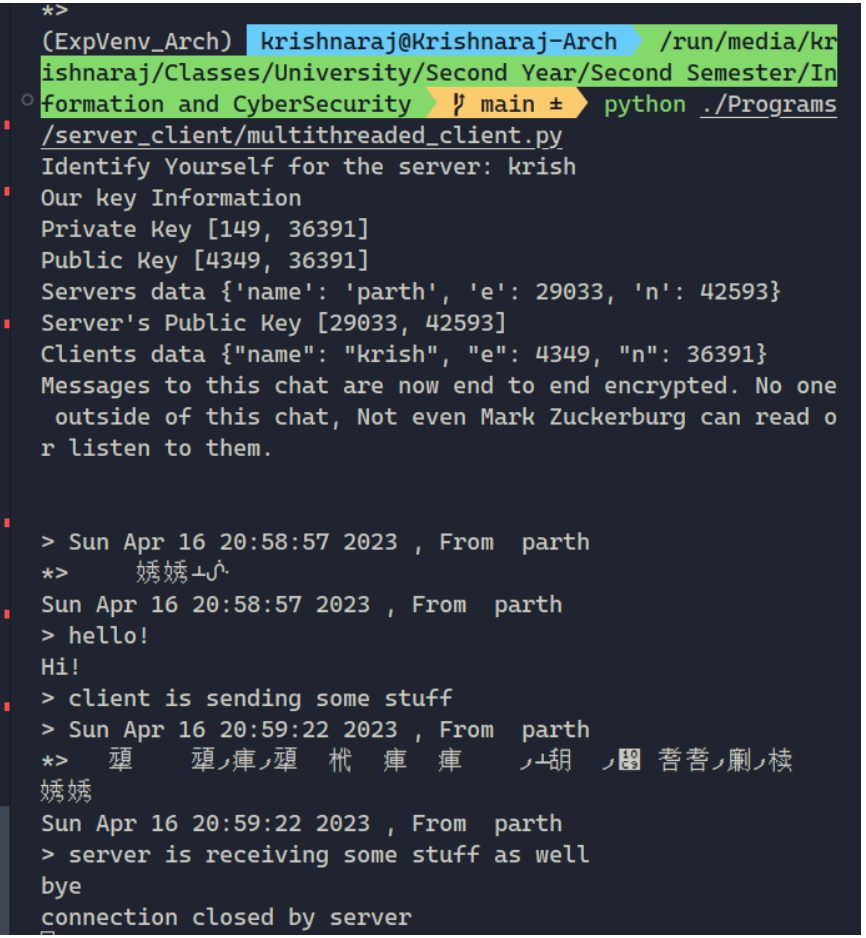
Transferring Video, or any media files is also possible this way.



```python
# code for sending video file to client

# echo-server.py

import socket
import os

HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)

# Define the file to be sent
file_name = '../Assignment_3/sunset.mp4'

# Create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:

    # Bind the socket to a specific address and port number
    s.bind((HOST, PORT))

    # Listen for incoming connections
    s.listen()

    # Wait for a client to connect
    conn, addr = s.accept()

    # Send the file to the client
    with open(file_name, 'rb') as f:
        data = f.read(1024)
        while data:
            conn.send(data)
            data = f.read(1024)

    # Close the connection
    conn.close()
```

```python
# Code for receiving video from server
import socket

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432  # The port used by the server
# Create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:

    # Connect to the server
    s.connect((HOST, PORT))

    # Receive the file from the server
    with open('received_video.mp4', 'wb') as f:
        while True:
            data = s.recv(1024)
            if not data:
                break
            f.write(data)

    # Close the connection
    s.close()
```

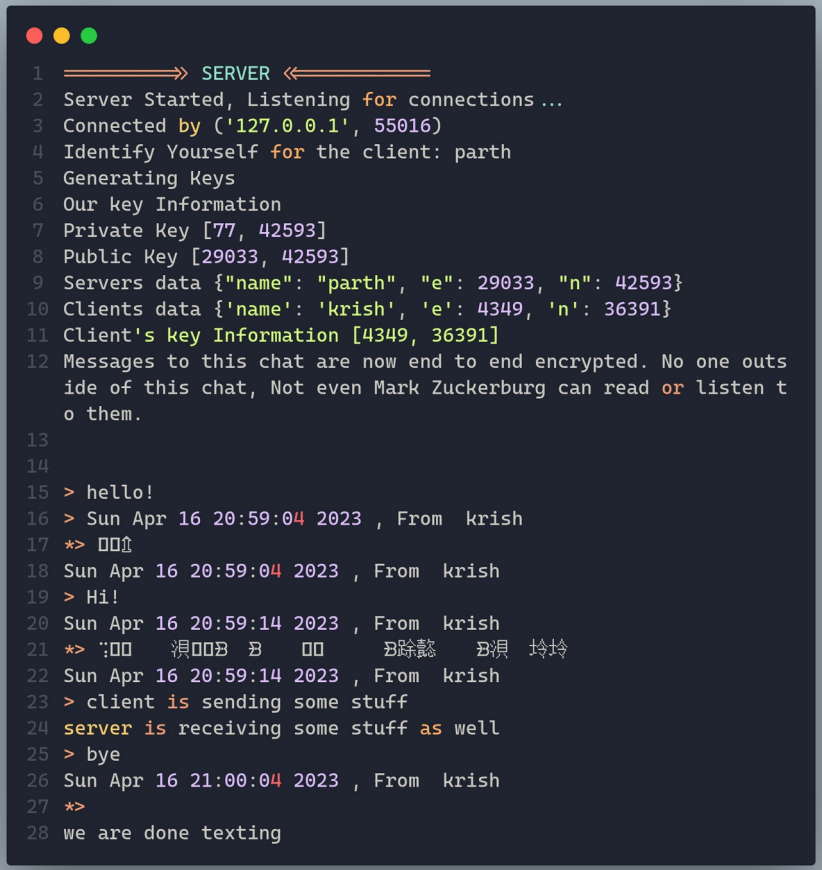

aj/Programs/Python/Experimentation/ExpVenv_Arch/bin/python "
/run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSecurity/Programs/server_client/multithreaded_server.py"
Server Started, Listening for connections...
Connected by ('127.0.0.1', 55016)
Identify Yourself for the client: parth
Generating Keys
Our key Information
Private Key [77, 42593]
Public Key [29033, 42593]
Servers data {"name": "parth", "e": 29033, "n": 42593}
Clients data {'name': 'krish', 'e': 4349, 'n': 36391}
Client's key Information [4349, 36391]
Messages to this chat are now end to end encrypted. No one outside of this chat, Not even Mark Zuckerburg can read or listen to them.
> hello!
> Sun Apr 16 20:59:04 2023 , From krish
*>
Sun Apr 16 20:59:04 2023 , From krish
> Hi!
Sun Apr 16 20:59:14 2023 , From krish
*>
Sun Apr 16 20:59:14 2023 , From krish
> client is sending some stuff
server is receiving some stuff as well
> bye


*>
(ExpVenv_Arch) krishnaraj@Krishnaraj-Arch /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSecurity main ± python ./Programs/server_client/multithreaded_client.py
Identify Yourself for the server: krish
Our key Information
Private Key [149, 36391]
Public Key [4349, 36391]
Servers data {'name': 'parth', 'e': 29033, 'n': 42593}
Server's Public Key [29033, 42593]
Clients data {"name": "krish", "e": 4349, "n": 36391}
Messages to this chat are now end to end encrypted. No one outside of this chat, Not even Mark Zuckerburg can read or listen to them.
> Sun Apr 16 20:58:57 2023 , From parth
*>
Sun Apr 16 20:58:57 2023 , From parth
> hello!
Hi!
> client is sending some stuff
> Sun Apr 16 20:59:22 2023 , From parth
*>
Sun Apr 16 20:59:22 2023 , From parth
> server is receiving some stuff as well
bye
connection closed by server

```python
def encrypt_alg(plain_text):
    # print("encrypting: ", plain_text, type(plain_text))
    cipher_texts = []
    plain_texts = []

    for i in plain_text:
        # print(ord(i))
        cipher_text = rsa_encryption(ord(i), key=server_public_key)
        cipher_texts.append(cipher_text)
        # print(cipher_text)

    cipher_texts = [chr(i) for i in cipher_texts]
    cipher_text = "".join(cipher_texts)

    time.sleep(0.01)
    return cipher_text
```

```python
def decrypt_alg(cipher_text):
    plain_texts = []

    for i in cipher_text:
        # print(ord(i))
        plain_text = rsa_decryption(ord(i), key=private_key)
        plain_texts.append(plain_text)

    plain_texts = [chr(i) for i in plain_texts]
    plain_text = "".join(plain_texts)

    time.sleep(0.01)
    return plain_text
```

```
# BEGIN CONVERSATION


input_thread = threading.Thread(target=send_data, args=(s,))
input_thread.start()

encrypt_thread = threading.Thread(target=encrypt)
encrypt_thread.start()

decrypt_thread = threading.Thread(target=decrypt, args=(server_name,))
decrypt_thread.start()


while texting:
    data = s.recv(1024)
    print(time.ctime(time.time()), ", From ", server_name)
    print(
        "*>",
        data.decode(encoding="utf-32").strip(" "),
    )
    messages_received.insert(0, data.decode(encoding="utf-32").strip(" "))

s.close()

```


SERVER
Server Started, Listening for connections ...
Connected by ('127.0.0.1', 55016)
Identify Yourself for the client: parth
Generating Keys
Our key Information
Private Key [77, 42593]
Public Key [29033, 42593]
Servers data {"name": "parth", "e": 29033, "n": 42593}
Clients data {'name': 'krish', 'e': 4349, 'n': 36391}
Client's key Information [4349, 36391]
Messages to this chat are now end to end encrypted. No one outside of this chat, Not even Mark Zuckerburg can read or listen to them.
> hello!
> Sun Apr 16 20:59:04 2023 , From krish
*> □□
Sun Apr 16 20:59:04 2023 , From krish
> Hi!
Sun Apr 16 20:59:14 2023 , From krish
*> □□ □□ □ □□ □□
Sun Apr 16 20:59:14 2023 , From krish
> client is sending some stuff
server is receiving some stuff as well
> bye
Sun Apr 16 21:00:04 2023 , From krish
*>
we are done texting


CLIENT
Identify Yourself for the server: krish
Our key Information
Private Key [149, 36391]
Public Key [4349, 36391]
Servers data {'name': 'parth', 'e': 29033, 'n': 42593}
Server's Public Key [29033, 42593]
Clients data {"name": "krish", "e": 4349, "n": 36391}
Messages to this chat are now end to end encrypted. No one outside of this chat, Not even Mark Zuckerburg can read or listen to them.
> Sun Apr 16 20:58:57 2023 , From parth
*>
Sun Apr 16 20:58:57 2023 , From parth
> hello!
Hi!
> client is sending some stuff
> Sun Apr 16 20:59:22 2023 , From parth
*>
Sun Apr 16 20:59:22 2023 , From parth
> server is receiving some stuff as well
bye

Thank You!