

PROJECT-6 SECURE CHAT APPLICATION

Objective

To develop a basic secure messaging application using RSA encryption and Python socket programming that ensures end-to-end encryption between two users.

Tech Stack & Tools Used:

Component	Technology
Language	Python 3
Networking	socket module
Encryption	PyCryptodome (RSA)
Threading	threading module
Optional GUI	Tkinter <i>(future scope)</i>

Install required library:

```
pip install pycryptodome
```

```
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Nikhilnick>cd OneDrive\Desktop  
  
C:\Users\Nikhilnick\OneDrive\Desktop>pip install pycryptodome  
Collecting pycryptodome  
  Downloading pycryptodome-3.23.0-cp37-abi3-win_amd64.whl.metadata (3.5 kB)  
  Downloading pycryptodome-3.23.0-cp37-abi3-win_amd64.whl (1.8 MB)  
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 24.4 MB/s eta 0:00:00  
Installing collected packages: pycryptodome  
Successfully installed pycryptodome-3.23.0  
  
C:\Users\Nikhilnick\OneDrive\Desktop>
```

Create a new folder securechatapp and a new file generatekey.py.

Add this below code in file.

Python code:

```
from Crypto.PublicKey import RSA
```

```
key = RSA.generate(2048)
```

```
private_key = key.export_key()
```

```
public_key = key.publickey().export_key()
```

```
with open("private.pem", "wb") as f:
```







```
    f.write(private_key)
```

```
with open("public.pem", "wb") as f:
```

```
    f.write(public_key)
```

this code will automatically generate public and private key.

```
C:\Users\Nikhilnick\OneDrive\Desktop>cd securechatapp
C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp>py generatekey.py
C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp>
```

	Name	Status	Date modified	Type	Size
	 generatekey.py		6/20/2025 7:28 PM	Python	1 KB
Personal	 private.pem		6/20/2025 7:29 PM	PEM File	2 KB
	 public.pem		6/20/2025 7:29 PM	PEM File	1 KB

Start the Server:

```
python server.py
```

Start Clients in separate terminals:

```
python client1.py
```

```
python client2.py
```

```
C:\Users\Nikhilnick\OneDrive\Desktop>cd securechatapp  
C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp>py generatekey.py  
C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp>py server.py  
Server is listening...  
(127.0.0.1, 6739) connected.  
(127.0.0.1, 6755) connected.  
|
```

```
PS C:\Users\Nikhilnick> cd OneDrive\Desktop\securechatapp  
PS C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp> py client.py  
You: hi, how are you  
You:  
Friend: hi i am fine  
  
Friend: and u  
am good what are you doing now?  
You:  
Friend: currently i am doing an internship from tamizhanskills!!  
wow its a great news that you got an internship i am happy for you  
You:  
Friend: thankyou and what are you doing now-a-days  
iam currently searching for a job could you please help me??  
You:  
Friend: yes sure without experince no body will take you so first search for internship  
|
```

```
PS C:\Users\Nikhilnick> cd OneDrive\Desktop\securechatapp
PS C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp> py client2.py
You: hi i am fine
You: and u
You:
Friend: am good what are you doing now?
currently i am doing an internship from tamizhanskills!!
You:
Friend: wow its a great news that you got an internship i am happy for you
thankyou and what are you doing now-a-days
You:
Friend: iam currently searching for a job could you please help me??
yes sure without experince no body will take you so first search for internship
You: |
```

```
y': [Errno 2] No such file or directory
PS C:\Users\Nikhilnick> cd OneDrive\Desktop\securechatapp
PS C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp> py client.py
You: hi how are you?
You:
Friend: hi i am good and u

Friend: am good havve you completed your project?
yes
You: |
```

```
PS C:\Users\Nikhilnick> cd OneDrive\Desktop\securechatapp
PS C:\Users\Nikhilnick\OneDrive\Desktop\securechatapp> py client2.py
You: hi i am good and u
You: am good havve you completed your project?
You:
Friend: yes
|
```

How It Works

RSA Key Pair:

Each user has a public and private RSA key.

`public.pem`: Used to encrypt messages.

`private.pem`: Used to decrypt incoming messages.

Client Flow:

Accept user input from the terminal.

Encrypt message using **public key**.

Send it over socket to the server.

Server Flow:

Accept connections from clients.

Relay messages between clients (does not decrypt).

Receiver:

Receives encrypted message.

Decrypts using **private key**.

Displays plaintext in

the terminal.

Encryption Logic

Asymmetric RSA Encryption (2048-bit key)

Messages encrypted with public key → only decrypted by the intended user with the private key.

Uses `PKCS1_OAEP` padding for security.

`generate_keys.py`

Generates `private.pem` and `public.pem` files with RSA 2048-bit key pair.

`server.py`

Handles multiple clients using threading, relays encrypted messages without decrypting.

`client1.py/client2.py`

Encrypts messages before sending and decrypts received messages using RSA.

Security Consideration

Public keys must be exchanged securely in real implementations.

Each client should have a unique key pair (currently simplified).

No logging of decrypted messages.

Future scope includes AES for performance or hybrid encryption.

Conclusion

This project demonstrates the core idea of **end-to-end encrypted**

messaging using **Python** and **RSA**. It's a simple yet powerful foundation for secure communication and can be extended to support GUI, file sharing, and hybrid encryption.