LAPORAN UDP SOCKET PROGRAMMING

disusun untuk memenuhi tugas besar mata kuliah II2120- Jaringan Komputer yang dibimbing oleh Bapak Ir. I Gusti Bagus Baskara Nugraha, S.T., M.T., Ph.D.

disusun oleh:

Muhammad Farhan	18223004
Darryl Rizqi Ramadhan	18223084

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2024

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh,

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan ini tepat pada waktunya.

Dokumen ini merupakan laporan yang disusun untuk memenuhi tugas besar mata kuliah II2120-Jaringan Komputer. Selain itu, laporan ini bertujuan menambah wawasan terkait pemanfaatan komputer dan jaringan-jaringannya untuk melakukan komunikasi dengan bantuan library Binascii, Queue, Socket dan Threading dalam bahasa pemrograman *python.*

Kami mengucapkan terima kasih kepada Bapak Ir. I Gusti Bagus Baskara Nugraha, S.T., M.T., Ph.D. selaku dosen pengampu mata kuliah II2120-Jaringan Komputer. Ucapan terima kasih juga kami sampaikan kepada semua pihak yang telah membantu dalam penyelesaian laporan ini.

Kami menyadari laporan ini masih jauh dari kata sempurna. Oleh sebab itu, apabila terdapat kesalahan dalam penulisan atau pun adanya ketidaksesuaian materi yang kami angkat pada laporan ini, kami mohon maaf. Kami mengharapkan saran dan kritik yang membangun demi kesempurnaan laporan ini. Semoga proposal ini dapat bermanfaat bagi kita semua.

Bandung, 31 Oktober 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
SPESIFIKASI PROGRAM

1.1. Spesifikasi wajib

No	Spesifikasi	check
1	Server mampu menerima pesan yang dikirim client dan mencetaknya ke layar.	V
2	Server mampu meneruskan pesan satu client ke client lain.	V
3	Client mampu mengirimkan pesan ke server dengan IP dan port yang ditentukan pengguna.	V
4	Client mampu menerima pesan dari client lain (yang diteruskan oleh server), dan mencetaknya ke layar.	V
5	Client harus memasukkan *password* untuk dapat bergabung ke chatroom.	V
6	Client memiliki username yang unik.	V

Tabel 1.1 Spesifikasi wajib

1.2. Spesifikasi Tambahan

No	Spesifikasi	check
1	Aplikasi mengimplementasikan TCP over UDP.	
2	Seluruh pesan dienkripsi menggunakan algoritma kriptografi klasik simetris, misal cipher Vigenere atau Caesar.	
3	Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern simetris, misal cipher RC4.	
4	Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern asimetris, misal cipher RSA, atau kombinasi algoritma kriptografi modern asimetris dan modern simetris.	V
5	Seluruh pesan dienkripsi menggunakan algoritma Double-Rachet atau MLS.	
6	Aplikasi memiliki GUI.	
7	Aplikasi mampu digunakan untuk mengirimkan dan menerima pesan bertipe file biner.	

8	Aplikasi mampu menunjukkan apabila integritas pesan telah rusak, baik dengan memanfaatkan *checksum* ataupun *digital signature*.	V
9	Aplikasi mampu menyimpan pesan-pesan lampau meskipun telah ditutup; mekanisme dan tempat penyimpanan bebas, baik di *client* maupun di *server*.	V
10	Aplikasi mampu mengotentikasi pengguna.	
11	Aplikasi diprogram menggunakan paradigma *object oriented programming* atau pemrograman berorientasi objek	

Tabel 1.2 Spesifikasi tambahan

BAB II
CARA KERJA PROGRAM

2.1. Server.py

- Impor Library dan Pengaturan Awal
  - Server memulai dengan mengimpor library yang dibutuhkan untuk program,pemrosesan multithreading, dan pengolahan pesan. IP address diatur menggunakan IP Adress local device,port diatur menjadi ‘9999’(karena cuma port ini yang bisa),kata sandi untuk pengetesan diatur menjadi ‘123’ supaya memudahkan.
- Inisialisasi Variabel Global
  - Queue messages disiapkan untuk menyimpan pesan yang diterima dari klien. Clients dan clients_names digunakan untuk menyimpan informasi klien yang terhubung. Kunci RSA (public key, private key, dan n) generate ketika server dijalankan.
- Membuat Socket Server
  - Server membuat socket UDP dan ke alamat IP dan port yang telah ditentukan untuk menunggu koneksi dari klien.

❖ Fungsi untuk menangani history pesan

Function	Kegunaan
get_timestamp()	Menghasilkan cap waktu untuk setiap pesan.
save_message(message)	Menyimpan pesan yang diterima ke dalam file log(namanya chat_history.txt).
load_messages()	Memuat riwayat pesan dari file log untuk ditampilkan kepada klien yang baru bergabung.

Tabel 1.3 Fungsi history pesan

❖ Fungsi receive
  ➢ Berjalan dalam loop terus-menerus, mendengarkan pesan dari klien. Jika klien meminta kunci publik, server mengirimkannya. Pesan lainnya akan didekripsi dan disimpan dalam queue untuk diproses lebih lanjut.
❖ Fungsi broadcast
  ➢ Memproses pesan dalam antrian. Fungsi ini menangani logika untuk klien baru dan klien yang sudah ada, memverifikasi otentikasi, dan mengecek integritas pesan sebelum disiarkan ke klien lain.
❖ Memulai Thread
  ➢ Dua thread dimulai satu untuk menerima pesan dari klien dan satu lagi untuk menyiarkan pesan ke semua klien.

2.2. Client.py

❖ Membuat Socket Klien
  ➢ Klien meminta kunci publik dari server dan kemudian mengikat socket ke port acak antara 8000 dan 9000.
❖ Mengautentikasi Klien
  ➢ Klien meminta kunci publik dari server, kemudian pengguna memasukkan nama pengguna dan kata sandi. Pesan otentikasi (yang berisi nama dan kata sandi) dienkripsi menggunakan kunci publik sebelum dikirim ke server.
❖ Menerima Pesan di Klien

- ➢ Klien memiliki thread terpisah untuk mendengarkan pesan dari server. Jika menerima pesan kesalahan seperti "Incorrect password" atau "Name already taken", klien menutup koneksi.

❖ Mengirim Pesan dari Klien

- ➢ Pengguna dapat memasukkan pesan untuk dikirim. Setiap pesan dienkripsi dengan kunci publik dan dilengkapi dengan checksum untuk memastikan integritas pesan.

2.3. RSA_module.py

❖ Fungsi is_prime(number)

- ➢ Memeriksa apakah bilangan yang diberikan adalah bilangan prima. Fungsi ini mengembalikan `True` jika bilangan tersebut prima, dan `False` jika tidak.

❖ Fungsi generate_prime(min_value, max_value)

- ➢ Menghasilkan bilangan prima acak dalam rentang yang ditentukan dengan menggunakan fungsi `is_prime`.

❖ Fungsi mod_inverse(e, phi)

- ➢ Menghitung invers modular dari `e` terhadap `phi` menggunakan algoritma Extended Euclidean. Jika GCD dari `e` dan `phi` adalah 1, invers modular ditemukan dan dikembalikan.

❖ Fungsi generate_keys()

- ➢ Menghasilkan pasangan kunci publik dan privat untuk algoritma RSA. Dua bilangan prima (`p` dan `q`) dihasilkan, lalu `n` dan `phi_n` dihitung. Sebuah bilangan `e` dipilih secara acak, dan invers modular `d` dihitung untuk mendapatkan kunci privat.

❖ Fungsi encrypt(message, public_key, n)

- ➢ Mengenkripsi pesan menggunakan kunci publik agar hanya penerima yang memiliki `private key` yang dapat mendekripsi dan membaca pesan tersebut.

❖ Proses

- ➢ Pertama, fungsi ini memeriksa apakah tipe data `message` adalah `bytes`. Jika iya, pesan tersebut di-decode menjadi string agar dapat diproses karakter per karakter.
- ➢ Fungsi ini kemudian membuat daftar kosong untuk menyimpan nilai-nilai terenkripsi.
- ➢ Selanjutnya, untuk setiap karakter dalam pesan, fungsi menghitung nilai terenkripsi dengan menggunakan rumus RSA, yang dinyatakan sebagai `c equal m^e mod n`, di mana:

        - ‘m’ adalah nilai ASCII dari karakter yang sedang diproses.
        - `e` adalah eksponen publik (kunci publik) yang diperoleh sebelumnya.
        - `n` adalah hasil kali dua bilangan prima yang digunakan untuk menghasilkan kunci RSA.
    - ➢ Hasil dari perhitungan ini adalah nilai terenkripsi untuk setiap karakter, yang kemudian disimpan dalam daftar.
    - ➢ Setelah semua karakter diproses, fungsi mengembalikan daftar tersebut yang berisi semua nilai terenkripsi, yang akan digunakan untuk mengirim pesan dengan aman melalui jaringan.
- ❖ decrypt(ciphertext, private_key, n)
    - ➢ Mendekripsi ciphertext yang telah dienkripsi menggunakan `public key`, sehingga dapat mengembalikan pesan asli yang dapat dibaca oleh penerima.
- ❖ Proses
    - ➢ Fungsi ini menerima daftar nilai terenkripsi (`ciphertext`) yang dikirimkan dari pengirim dan menginisialisasi daftar kosong untuk menyimpan karakter yang telah didekripsi.
    - ➢ Fungsi kemudian mengiterasi setiap nilai dalam `ciphertext` dan menghitung karakter yang didekripsi menggunakan rumus RSA, yang dinyatakan sebagai `m equal c^d mod n` di mana:
        - `c` adalah nilai terenkripsi.
        - `d` adalah kunci privat yang sesuai dengan kunci publik yang digunakan untuk mengenkripsi.
        - `n’ adalah hasil kali dua bilangan prima yang sama dengan yang digunakan saat mengenkripsi.
    - ➢ Setelah menghitung nilai yang didekripsi, fungsi memeriksa apakah nilai tersebut berada dalam rentang karakter yang valid (0 hingga 0x10FFFF). Jika nilai tersebut valid, karakter yang sesuai diubah menjadi karakter Unicode dan ditambahkan ke daftar karakter yang didekripsi.
    - ➢ Jika nilai tidak valid, fungsi akan mengembalikan pesan kesalahan "Invalid message."
    - ➢ Setelah semua nilai dalam `ciphertext` diproses, fungsi menggabungkan semua karakter yang telah didekripsi menjadi satu string dan mengembalikannya sebagai hasil dekripsi. Pesan asli ini dapat dibaca oleh penerima yang memiliki kunci privat.

BAB III
TATA CARA & LINGKUNGAN TESTING

1. Pengujian dilakukan dengan 2 device yang berbeda
2. kedua device terhubung wifi yang sama
3. salah satu laptop menjalankan server dan client
4. laptop lainnya menjalankan client dan terhubung ke server

BAB IV
DOKUMENTASI & LAMPIRAN

4.1. Dokumentasi pengujian & antarmuka program

- Tampilan server

```
PS C:\Users\Darryl\Documents\Coding\python\socket\18_Tugas-Socket-Programming> py server.py
Server started at 192.168.0.162:9999
Received from ('192.168.0.162', 8591): Darryl:123
Saving message: [2024-11-01 10:36:54] Darryl has joined the chat!
Received from ('192.168.0.162', 8103): Farhan:123
Saving message: [2024-11-01 10:37:06] Farhan has joined the chat!
Received from ('192.168.0.162', 8103): Farhan:haloooo:2784623526
Saving message: [2024-11-01 10:37:10] Farhan: haloooo
Received from ('192.168.0.162', 8591): Darryl:helllooo:2437258575
Saving message: [2024-11-01 10:37:14] Darryl: helllooo
Received from ('192.168.0.162', 8103): Farhan:tesss:1562471065
Saving message: [2024-11-01 10:37:16] Farhan: tesss
Received from ('192.168.0.162', 8591): Darryl:tosss:849119997
Saving message: [2024-11-01 10:37:20] Darryl: tosss
```

Gambar 4.1.1 Tampilan server

Server mampu menerima pesan yang dikirim client dan mencetaknya ke layar. Server terus menerus mendengarkan pesan dari client dan akan langsung menampilkan pesan yang diterima ke layar sekaligus IP address dan port dari client yang mengirim pesan.

- Input IP Adress dan Port

```
PS C:\Users\Darryl\Documents\Coding\python\socket\18_Tugas-Socke
t-Programming> py client.py
Enter IP Address: 192.168.0.162
Enter port number: 9999
Nickname: Rizqi
Enter the password: 123
Rizqi joined!
```

Gambar 4.1.2 IP Adress dan Port berhasil

```
PS C:\Users\Darryl\Documents\Coding\python\socket\18_Tugas-Socke
t-Programming> py client.py
Enter IP Address: 192.168.0.160
Enter port number: 9999
```

Gambar 4.1.3 IP Adress dan Port gagal

Client mampu mengirimkan pesan ke server dengan IP dan port yang ditentukan pengguna.

Ketika program pertama dijalankan client akan diminta untuk memasukkan IP Address dan port tujuan.Jika port dan IP salah/tidak ditemukan,maka program akan stuck dan not responding.Hal ini dikarenakan tidak ada suatu cara untuk mengetahui jika alamat yang dimasukkan salah dan client tidak menerima feedback apapun.

- Autentikasi Password

```
PS C:\Users\Darryl\Documents\Coding\python\socket\18_Tugas-Socke
t-Programming> py client.py
Enter IP Address: 192.168.0.162
Enter port number: 9999
Nickname: Darryl
Enter the password: 123
Darryl joined!
Password accepted. You are now connected.
```

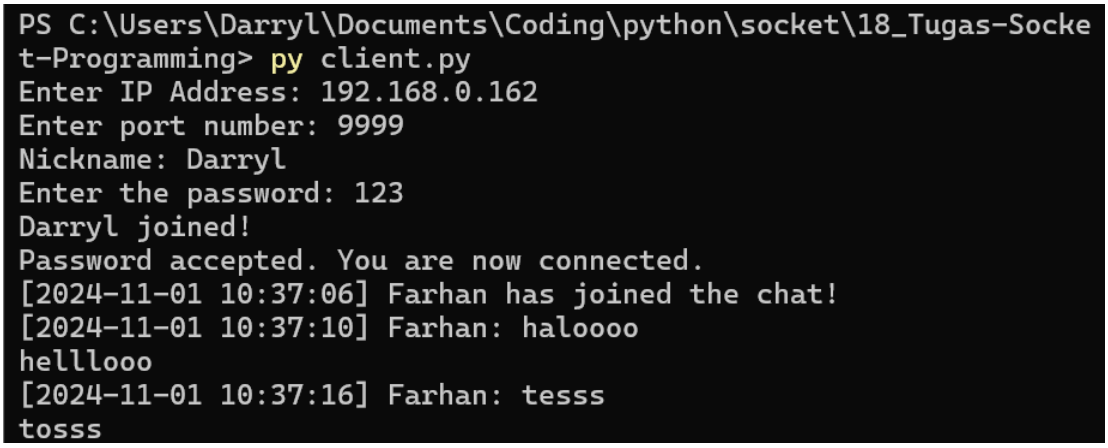
Gambar 4.1.4 Password diterima

```
PS C:\Users\Darryl\Documents\Coding\python\socket\18_Tugas-Socke
t-Programming> py client.py
Enter IP Address: 192.168.0.162
Enter port number: 9999
Nickname: Darryl
Enter the password: 234
Incorrect password.
Exiting...
```

Gambar 4.1.5 Password gagal

Client harus memasukkan *password* untuk dapat bergabung ke chatroom & ketika program pertama dijalankan ,setelah memasukkan IP, port, dan nickname client akan diminta memasukkan password untuk bisa bergabung ke classroom.jika password salah maka program akan langsung berhenti.

- Interaksi antar-Client pada antarmuka program

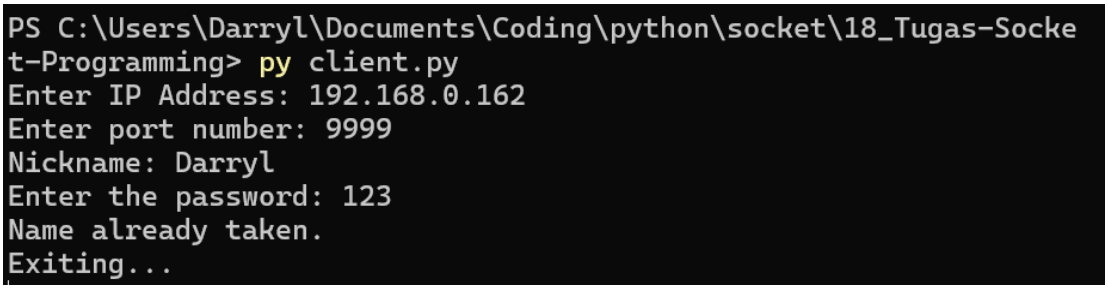


Gambar 4.1.6 tampilan client

Selain itu, Client mampu menerima pesan dari client lain (yang diteruskan oleh server), dan mencetaknya ke layar.

Sama seperti server client akan terus menerus mendengarkan pesan dan akan langsung menampilkan pesan ke layar dan juga waktu diterima pesan.

- Client memiliki username yang unik.



Gambar 4.1.7 username unik

Setelah memasukkan IP Adress dan Port client akan diminta memasukkan nickname dan password.Nickname akan dicek jika sudah ada client lain yang menggunakan nickname sama,maka program akan terhenti dan menuliskan “Name already taken”

- Penggunaan mekanisme enkripsi RSA

Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern asimetris, misal cipher RSA, atau kombinasi algoritma kriptografi modern asimetris dan modern simetris.

Dalam aplikasi ini, implementasi RSA digunakan untuk mengenkripsi dan mendekripsi pesan antara klien dan server. Sebelum pesan dikirim, pesan

tersebut dienkripsi terlebih dahulu menggunakan metode RSA. Setelah proses enkripsi, pesan akan diubah menjadi format yang dapat dikirim (encoded) dan kemudian dikirim melalui jaringan. Ketika server menerima pesan, pesan tersebut akan didekode (decoded) terlebih dahulu sebelum dilakukan proses dekripsi untuk mengembalikannya ke bentuk semula.

```python
def generate_keys():
    p = generate_prime(1000, 50000)
    q = generate_prime(1000, 50000)
    while p == q:
        q = generate_prime(1000, 50000)

    n = p * q
    phi_n = (p - 1) * (q - 1)

    e = random.randint(3, phi_n - 1)
    while math.gcd(e, phi_n) != 1:
        e = random.randint(3, phi_n - 1)

    d = mod_inverse(e, phi_n)
    return e, d, n

def encrypt(message, public_key, n):
    if isinstance(message, bytes):
        message = message.decode()
    return [pow(ord(ch), public_key, n) for ch in message]

def decrypt(ciphertext, private_key, n):
    decrypted_chars = []
    for ch in ciphertext:
        decrypted_value = pow(ch, private_key, n)
        if 0 <= decrypted_value <= 0x10FFFF:
            decrypted_chars.append(chr(decrypted_value))
        else:
            return "Invalid message."
```

Gambar 4.1.8 RSA module(1)

```python
auth_message = f"{name}:{password}"
encrypted_auth_message = encrypt(auth_message, public_key, n)
client.sendto(','.join(map(str, encrypted_auth_message)).encode(), ADDRESS)
```

Gambar 4.1.9 RSA module(2)

```
decrypted_message = decrypt(eval(message.decode()), private_key, n)
messages.put((decrypted_message.encode(), addr))
```

Gambar 4.1.10 RSA module(3)

- Aplikasi mampu menunjukkan apabila integritas pesan telah rusak, baik dengan memanfaatkan *checksum*.

```
        name, message_text, received_checksum = decoded_message.rsplit(":", 2)
        received_checksum = int(received_checksum)
        calculated_checksum = binascii.crc32(message_text.encode())

        if received_checksum == calculated_checksum:
            broadcast_message = f"{name}: {message_text}"
            save_message(broadcast_message)
            for client in clients:
                if client != addr:
                    server.sendto(f"[{get_timestamp()}] {broadcast_message}".encode(), client)
        else:
            server.sendto("Message integrity compromised!".encode(), addr)

    except ValueError:
        server.sendto("Invalid message format.".encode(), addr)
```

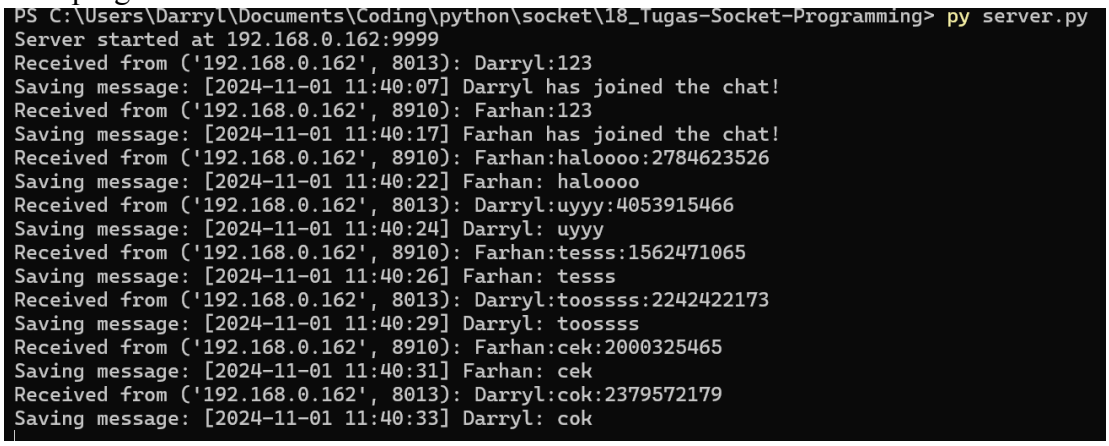
Gambar 4.1.11 Checksum

Aplikasi ini menjaga integritas pesan dengan menggunakan checksum berbasis CRC32. Saat pengguna mengirim pesan, aplikasi menghitung checksum dari pesan tersebut dan menyertakannya dalam format pengiriman sebagai bagian dari pesan. Di sisi server, setelah menerima pesan, server menghitung kembali checksum dari isi pesan dan membandingkannya dengan checksum yang diterima. Jika kedua checksum cocok, pesan dianggap utuh dan diproses; jika tidak, server mengirimkan pesan kesalahan yang menyatakan bahwa "Message integrity compromised!". Metode ini efektif dalam mendeteksi kerusakan data selama pengiriman, meningkatkan keandalan komunikasi antara klien dan server.(Tidak tahu bagaimana caranya tes ketika integritas pesan rusak).

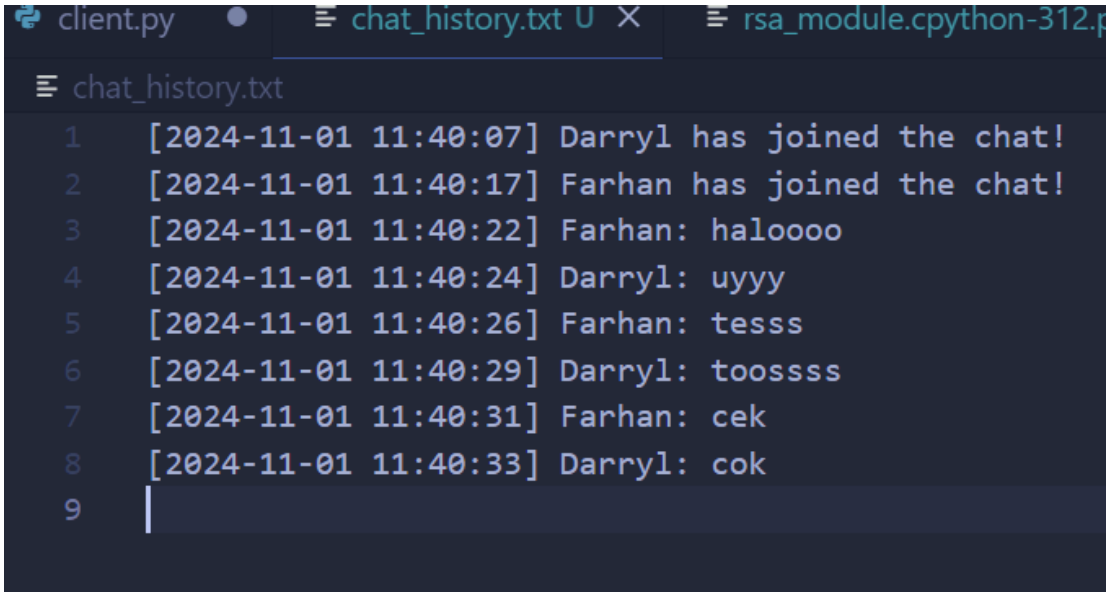
- Aplikasi menyimpan pesan-pesan lampau meskipun telah ditutup (mekanisme

Txt) di *server*.

Setiap pesan yang akan dibrodacast oleh server akan disimpan dalam file terpisah(format .txt),jadi ketika server dimatikan history chat akan tetap ada dan akan langsung ditampilkan ketika server dan client berikutnya memulai program.



Gambar 4.1.12 history chat(1)



Gambar 4.1.13 history chat(2)

4.2. Lampiran source code client dan server

Server

server.py > broadcast

```python
import socket
import threading
import queue
import binascii
from datetime import datetime
from rsa_module import generate_keys, encrypt, decrypt

PORT = 9999
SERVER = socket.gethostbyname(socket.gethostname())
ADDRESS = (SERVER, PORT)
PASSWORD = "123"
print(f"Server started at {SERVER}:{PORT}")

messages = queue.Queue()
clients = []
clients_names = []

public_key, private_key, n = generate_keys()

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind((SERVER, PORT))

LOG_FILE = "chat_history.txt"

def get_timestamp():
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")

def save_message(message):
    timestamp = get_timestamp()
    full_message = f"[{timestamp}] {message}"
    print(f"Saving message: {full_message}")

    with open(LOG_FILE, "a") as file:
        file.write(f"{full_message}\n")

def load_messages():
    try:
        with open(LOG_FILE, "r") as file:
            lines = file.readlines()
            return [line.strip() for line in lines]
    except FileNotFoundError:
        return []

def receive():
    while True:
        message, addr = server.recvfrom(1024)
        if message == b'GET_PUBLIC_KEY':
            server.sendto(f"{public_key},{n}".encode(), addr)
```

Gambar 4.2.1 server(1)

```
def receive():
            continue

        decrypted_message = decrypt(eval(message.decode()), private_key, n)
        messages.put((decrypted_message.encode(), addr))

def broadcast():
    while True:
        while not messages.empty():
            message, addr = messages.get()
            decoded_message = message.decode()
            print(f"Received from {addr}: {decoded_message}")

            if addr not in clients:
                try:
                    name, password = decoded_message.split(":")
                    name = name.strip()
                    password = password.strip()

                    if password != PASSWORD:
                        server.sendto("Incorrect password.".encode(), addr)
                    elif name in clients_names:
                        server.sendto("Name already taken.".encode(), addr)
                    else:
                        clients_names.append(name)
                        clients.append(addr)
                        server.sendto(f"{name} joined!".encode(), addr)
                        server.sendto("Password accepted. You are now connected.".encode(), addr)

                        chat_history = load_messages()
                        if chat_history:
                            for line in chat_history:
                                server.sendto(line.encode(), addr)

                        broadcast_message = f"{name} has joined the chat!"
                        save_message(broadcast_message)
                        for client in clients:
                            if client != addr:
                                server.sendto(f"[{get_timestamp()}] {broadcast_message}".encode(), client)
                except ValueError:
                    server.sendto("Invalid message format.".encode(), addr)
            else:
                try:
                    name, message_text, received_checksum = decoded_message.rsplit(":", 2)
                    received_checksum = int(received_checksum)
                    calculated_checksum = binascii.crc32(message_text.encode())

                    if received_checksum == calculated_checksum:
                        broadcast_message = f"{name}: {message_text}"
```

Gambar 4.2.2 server(2)

```
                if received_checksum == calculated_checksum:
                    broadcast_message = f"{name}: {message_text}"
                    save_message(broadcast_message)
                    for client in clients:
                        if client != addr:
                            server.sendto(f"[{get_timestamp()}] {broadcast_message}".encode(), client)
                else:
                    server.sendto("Message integrity compromised!".encode(), addr)

            except ValueError:
                server.sendto("Invalid message format.".encode(), addr)

t1 = threading.Thread(target=receive)
t2 = threading.Thread(target=broadcast)

t1.start()
t2.start()

```

Gambar 4.2.3 server(3)

Client

```
import socket
import threading
import random
import binascii
from rsa_module import encrypt

SERVER_IP = input("Enter IP Address: ")
PORT = int(input("Enter port number: "))
ADDRESS = (SERVER_IP, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.bind(('', random.randint(8000, 9000)))

client.sendto(b'GET_PUBLIC_KEY', ADDRESS)
public_key_data, _ = client.recvfrom(1024)
public_key, n = map(int, public_key_data.decode().split(','))

name = input("Nickname: ")
password = input("Enter the password: ")

auth_message = f"{name}:{password}"
encrypted_auth_message = encrypt(auth_message, public_key, n)
client.sendto(','.join(map(str, encrypted_auth_message)).encode(), ADDRESS)

stop_receiving = False

def receive():
    while not stop_receiving:
        try:
            message, _ = client.recvfrom(1024)
            decoded_message = message.decode()
            print(decoded_message)
            if decoded_message in ["Incorrect password.", "Name already taken."]:
                print("Exiting...")
                client.close()
                break
        except Exception as e:
            if not stop_receiving:
                print(f"Error receiving message: {e}")
            break

t = threading.Thread(target=receive)
t.daemon = True
t.start()

while True:
    message = input()
    if message == "!q":
```

Gambar 4.2.4 client(1)

```
while True:
    message = input()
    if message == "!q":
        print("Exiting...")
        stop_receiving = True
        break
    else:
        checksum = binascii.crc32(message.encode())
        message_with_checksum = f"{name}:{message}:{checksum}"
        encrypted_message = encrypt(message_with_checksum, public_key, n)
        client.sendto(','.join(map(str, encrypted_message)).encode(), ADDRESS)

client.close()
t.join()
```

Gambar 4.2.5 client(2)

4.3. Pranala video dan github repository

- Lampiran video
  https://youtu.be/OQBHssacWP0?si=K6UKhfLvIHJUEAnx
- github :
  https://github.com/RylRizqi16/18_Tugas-Socket-Programming

4.4. Pembagian Tugas Kelompok

No	Nama	NIM	Pekerjaan
1	Muhammad Farhan	18223004	• Pembuatan spesifikasi tambahan Checksum • Pembuatan spesifikasi tambahan penyimpanan pesan lampau • Pembuatan laporan & video
2	Darryl Rizqi Ramadhan	18223084	• Pembuatan spesifikasi wajib • Pembuatan spesifikasi tambahan enkripsi RSA • Membuat laporan & video

Tabel 1.4 Pembagian tugas

DAFTAR PUSTAKA

- Python Software Foundation. (2023). *socket — Low-level networking interface — Python 3.13.0 documentation*. Diakses pada 1 November 2024, dari https://docs.python.org/3/library/socket.html
- GeeksforGeeks. (2023). *Socket Programming in Python*. Diakses pada 1 November 2024, dari https://www.geeksforgeeks.org/socket-programming-python/
- History Tools. (2023). *How RSA Encryption Works: A Complete Step-by-Step Explanation*. Diakses pada 1 November 2024, dari https://www.historytools.org/rsa-encryption-explanation
- Cloudflare. (2023). *What is RSA encryption and how does it work?*. Diakses pada 1 November 2024, dari https://www.cloudflare.com/learning/cryptography/what-is-rsa-encryption/