

BÁO CÁO ĐỒ ÁN CUỐI KỲ

ĐỀ TÀI:

**ỨNG DỤNG IDENTITY-BASED  
ENCRYPTION TRONG VIỆC GỬI  
VÀ NHẬN TIN NHẮN**

**NHÓM 7**

Vũ Vinh Hiển

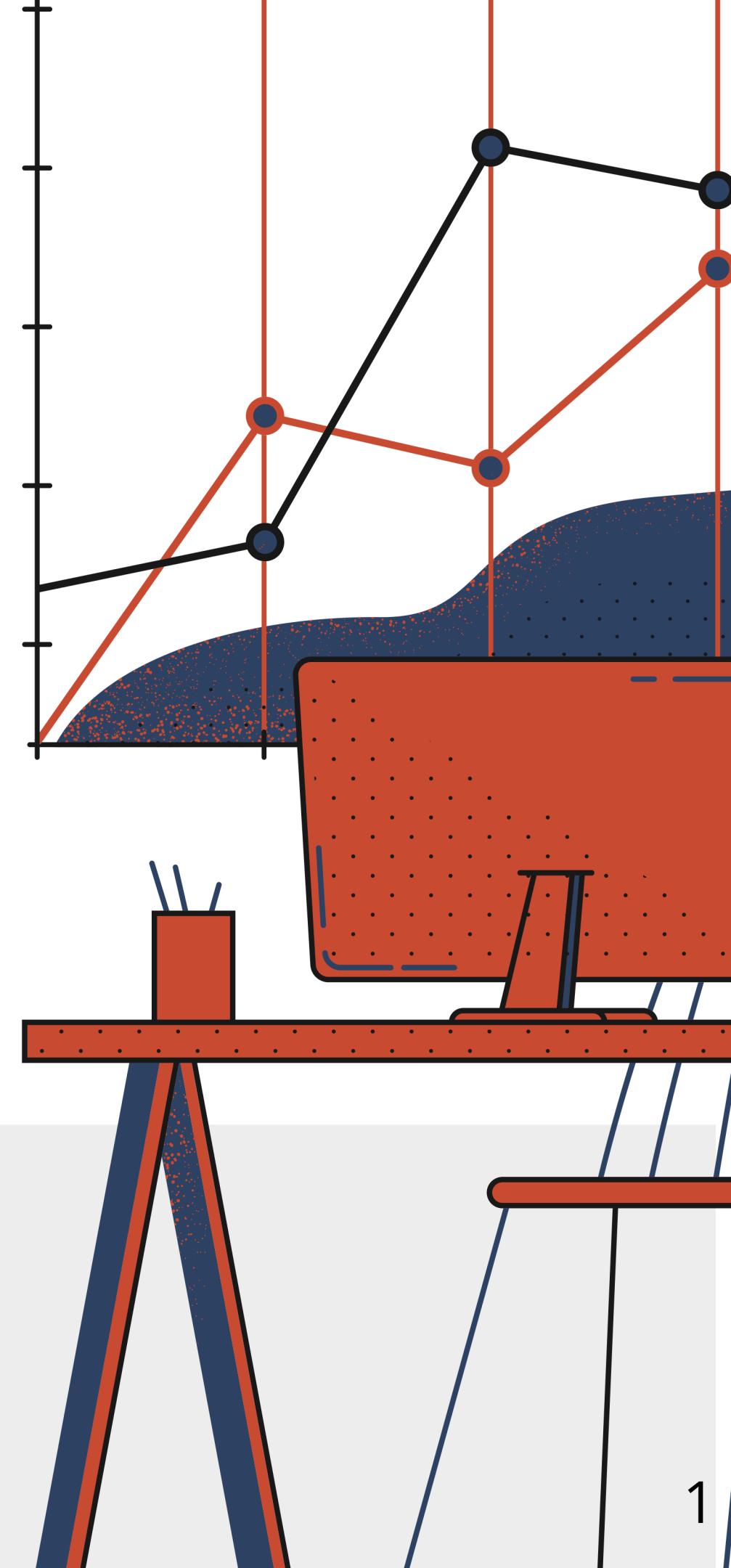
20520498

Nguyễn Trần Đức An

20520373

Nguyễn Hoàng Tuấn

20522118



I. Ngữ cảnh

II. Tài liệu tham khảo

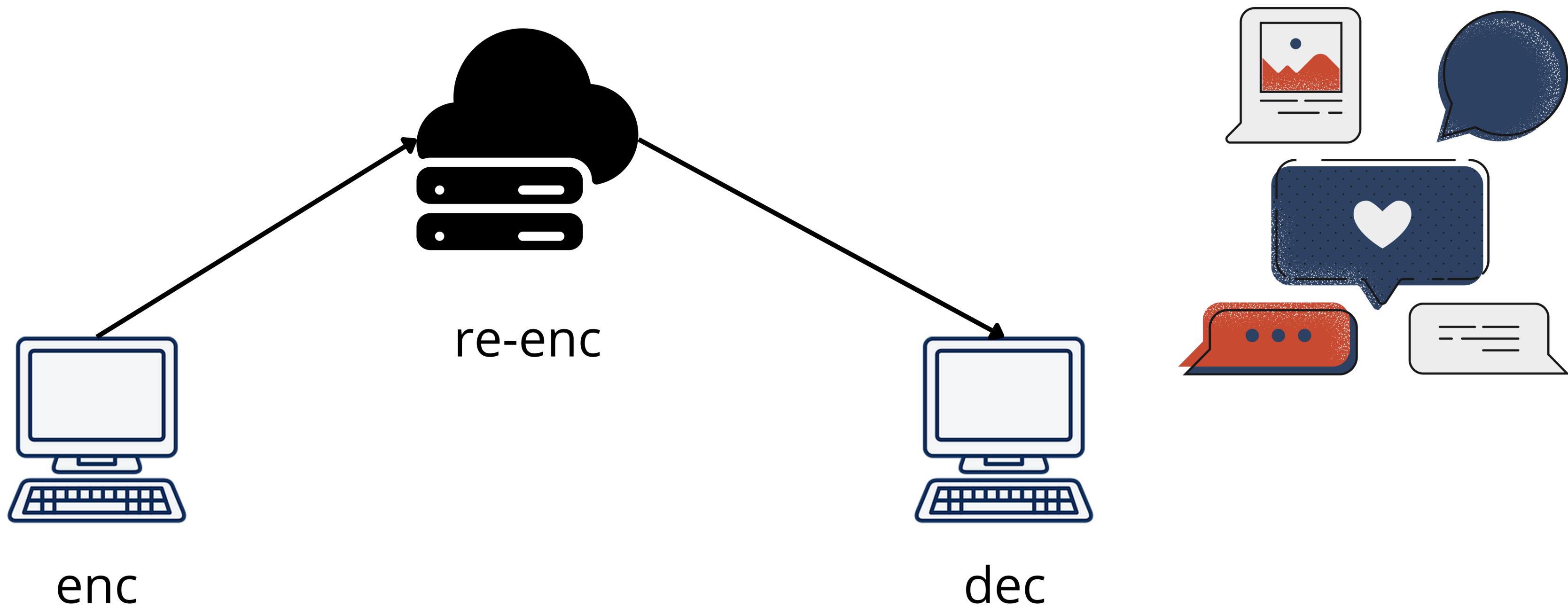
III. Các bên liên quan

IV. Yêu cầu bảo mật

V. Giải pháp

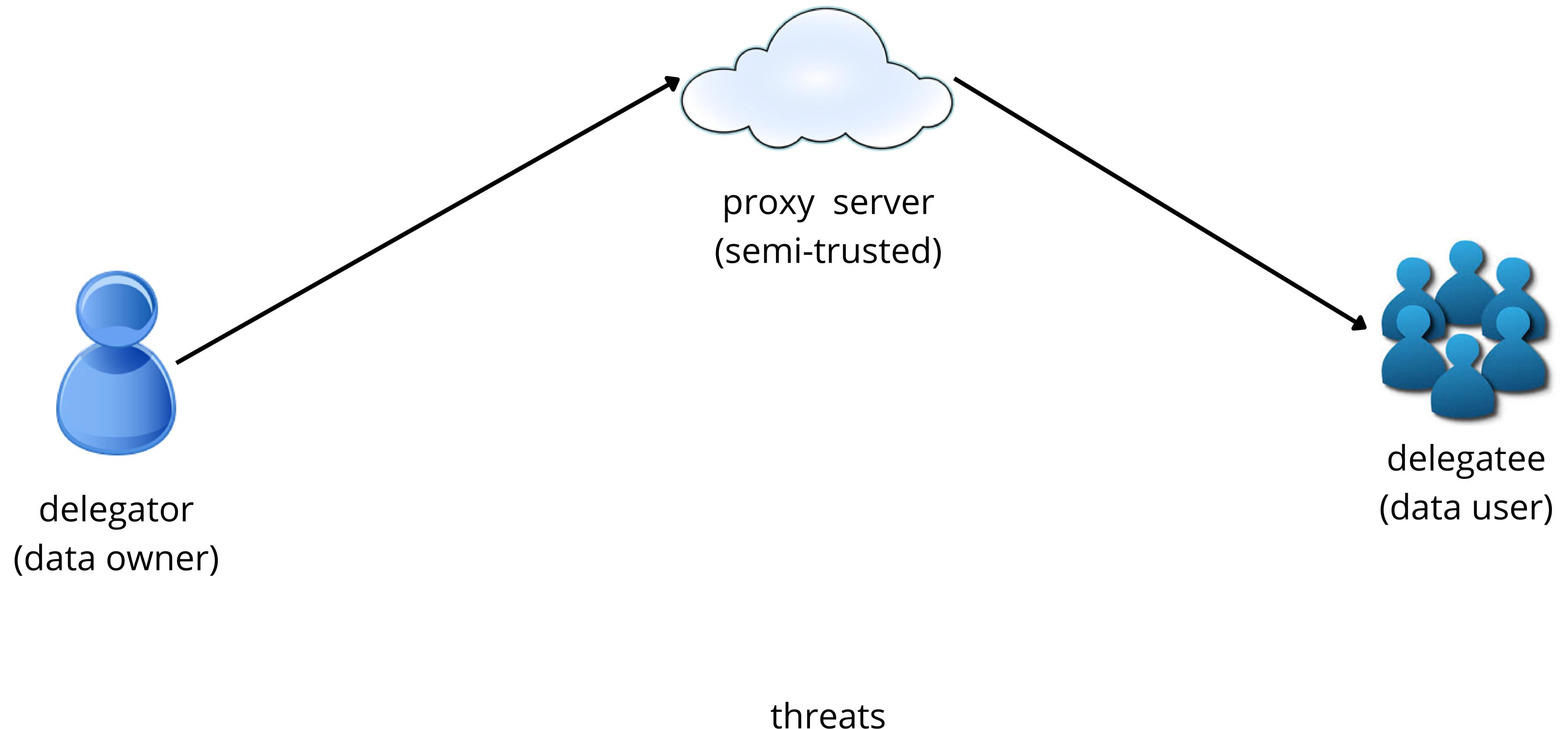
VI. Demo

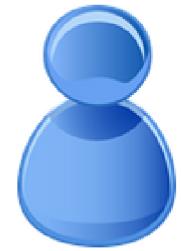
Trong việc gửi tin nhắn khi người gửi muốn gửi cho người nhận mà người nhận không có online, nên cần 1 server để có thể mã hóa và gửi cho người nhận khi người nhận online.



- Ge, C., Liu, Z., Xia, J., & Fang, L. (2019). Revocable identity-based broadcast proxy re-encryption for data sharing in clouds. *IEEE Transactions on Dependable and Secure Computing*, 18(3), 1214-1226.
- Yao, S., Dayot, R. V. J., Kim, H. J., & Ra, I. H. (2021). A Novel Revocable and Identity-Based Conditional Proxy Re-Encryption Scheme With Ciphertext Evolution for Secure Cloud Data Sharing. *IEEE Access*, 9, 42801-42816.







## delegator

- Sở hữu data
- Sở hữu danh sách người nhận
- Có quyền loại bỏ những người nhận



## delegatees

- Được uỷ quyền để nhận data
- Giải mã data để sử dụng



## proxy server

- Semi-trusted
- Mã hóa data dựa trên ID



## threats

- Sử dụng máy tính cổ điển
- Đánh cắp thông tin trên đường truyền
- Xâm nhập vào cloud



## Bên trong

- Bảo mật trên server
- Có thể xóa data user

## Bên ngoài

- Cần có key và nằm trong trusted list để có thể nhận và gửi message
- Mã hóa trước khi gửi



- Mã hóa lần 1 bằng ID người gửi trước khi gửi tin nhắn
- Mã hóa lần 2 bằng ID người nhận ở trên cloud
- Sử dụng **Advanced Encryption Standard (AES-256 CBC)** để mã hóa
- Gửi nhận bằng raw TCP
- Cơ chế revoke: loại người dùng ra khỏi danh sách hợp pháp

## Ngữ cảnh ứng dụng

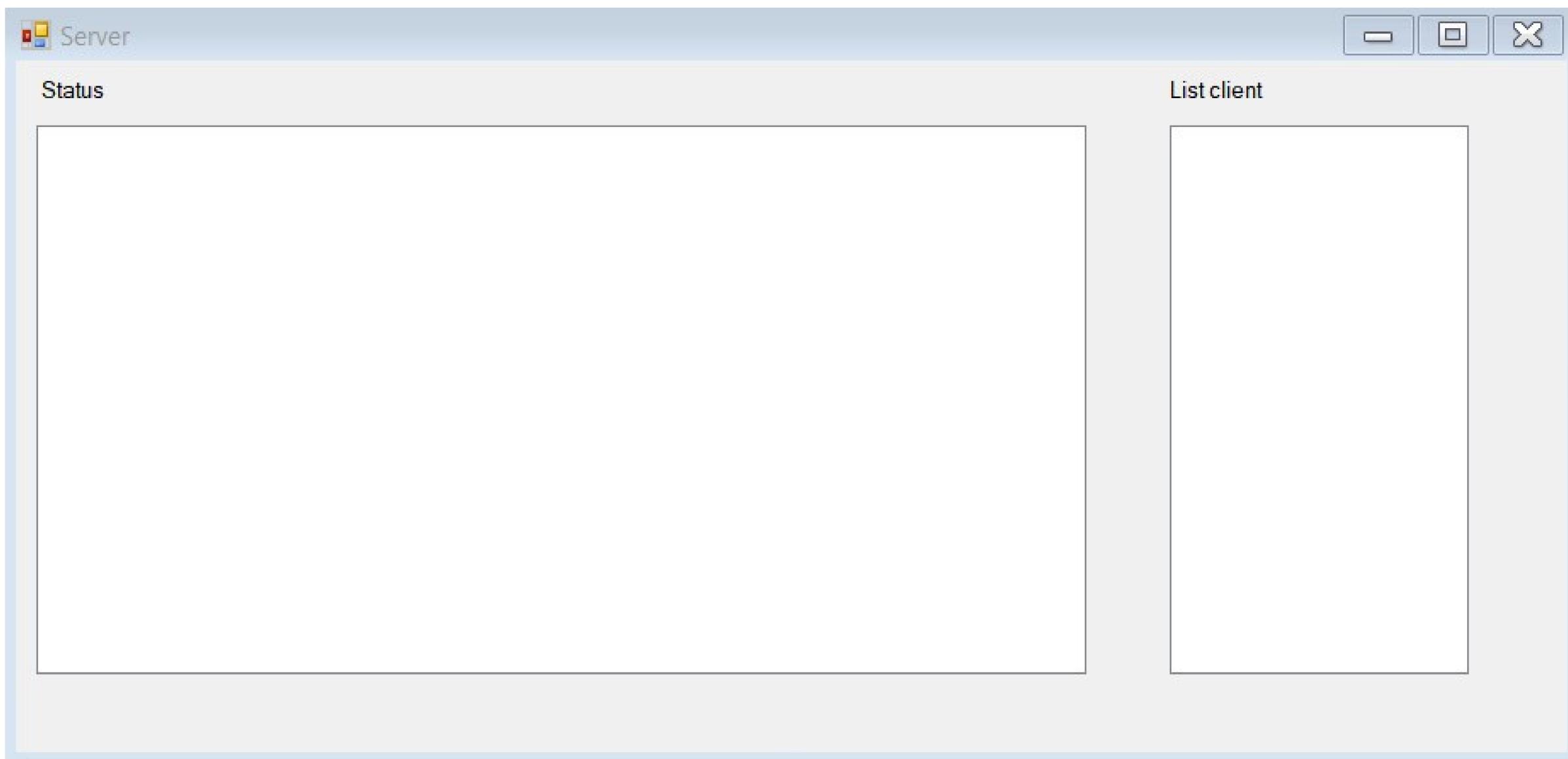
- Ứng dụng nhắn tin gồm 2 bên: server và client
- Đã biết trước danh sách người nhận
- Có thể xóa người nhận ra khỏi danh sách
- Mã hóa tin nhắn bằng ID (email)

## Demo

- Ứng dụng nhắn tin bằng ngôn ngữ C#
- Sử dụng .NET framework
- Data được mã hóa 2 lần
- Truyền bằng giao thức TCP

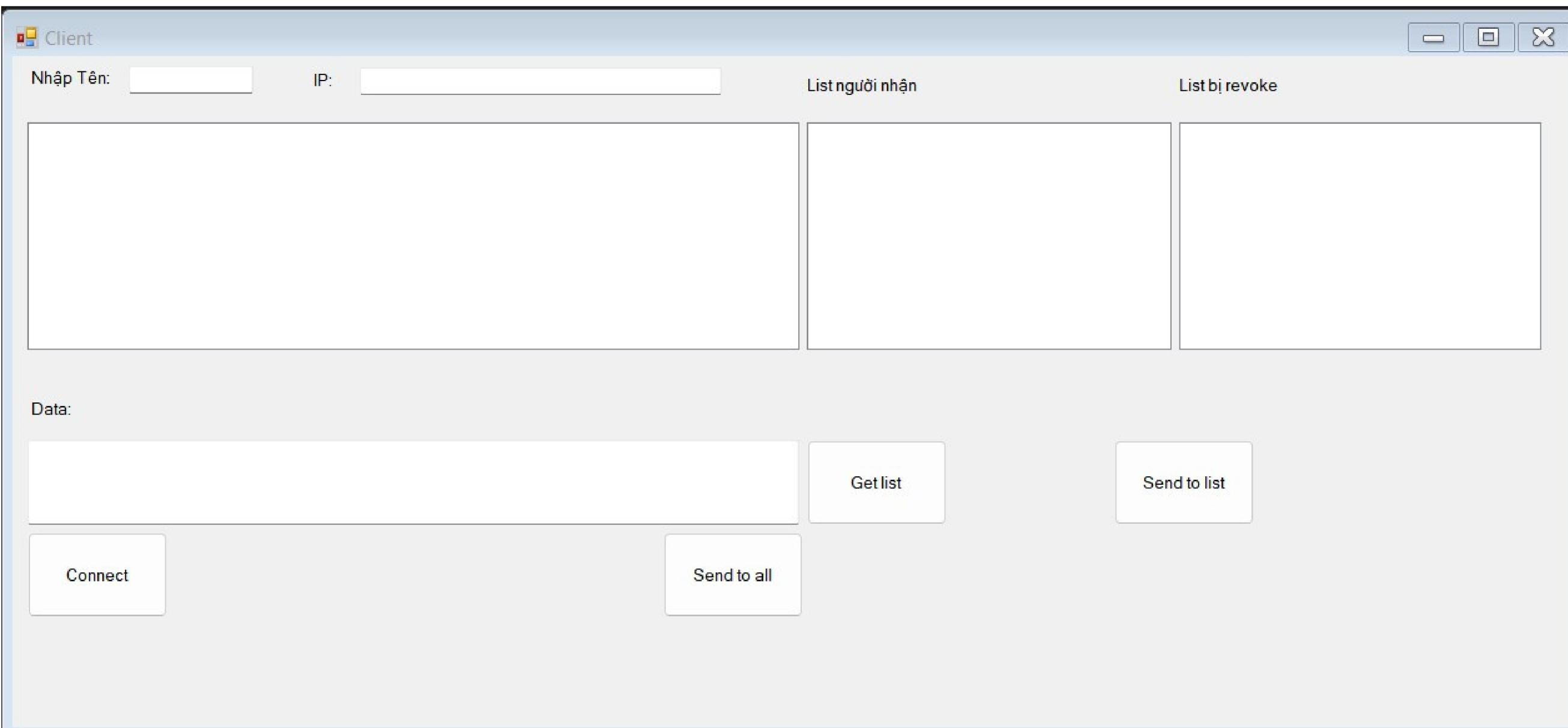


# Giao diện



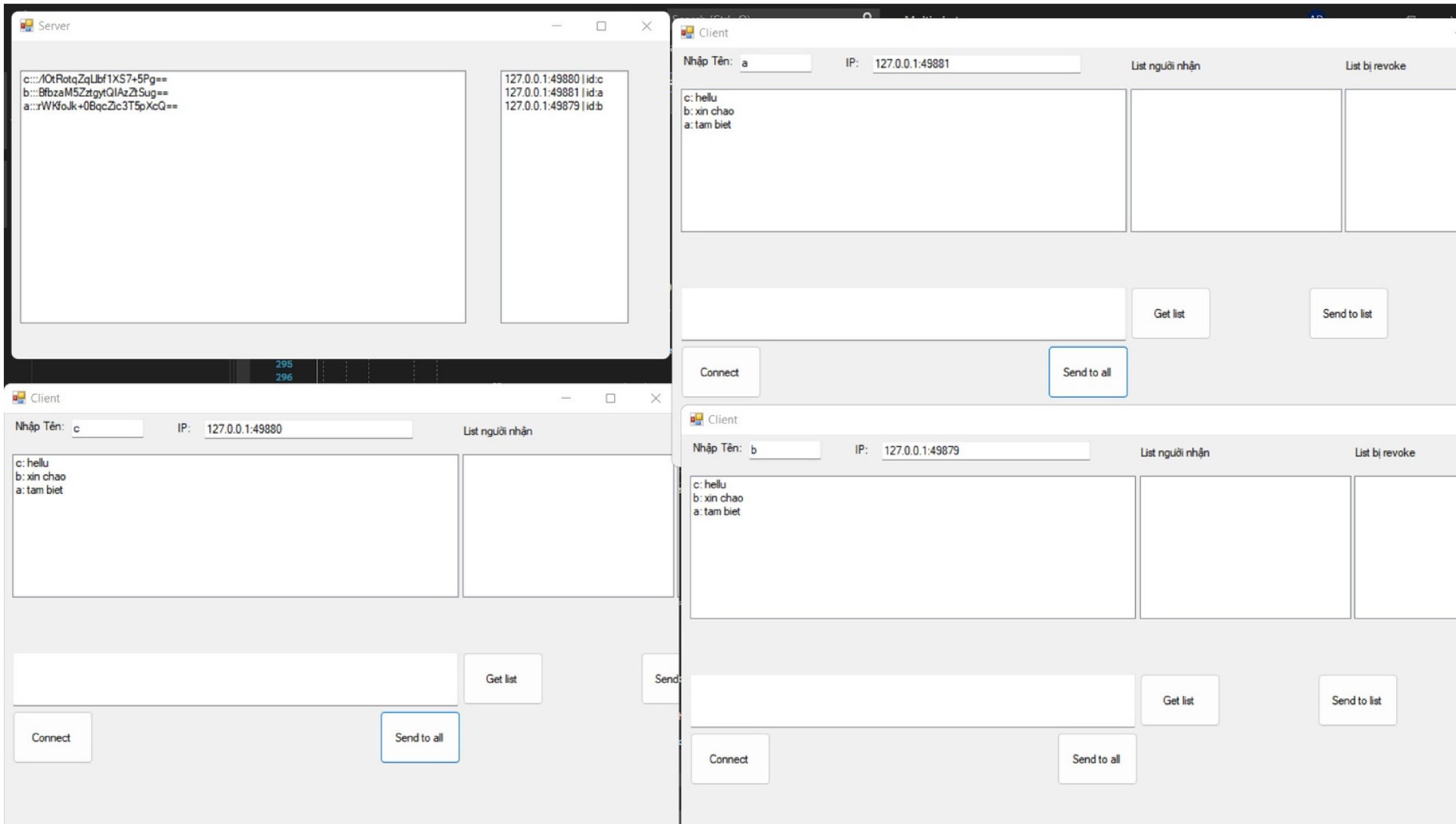
server

# Giao diện



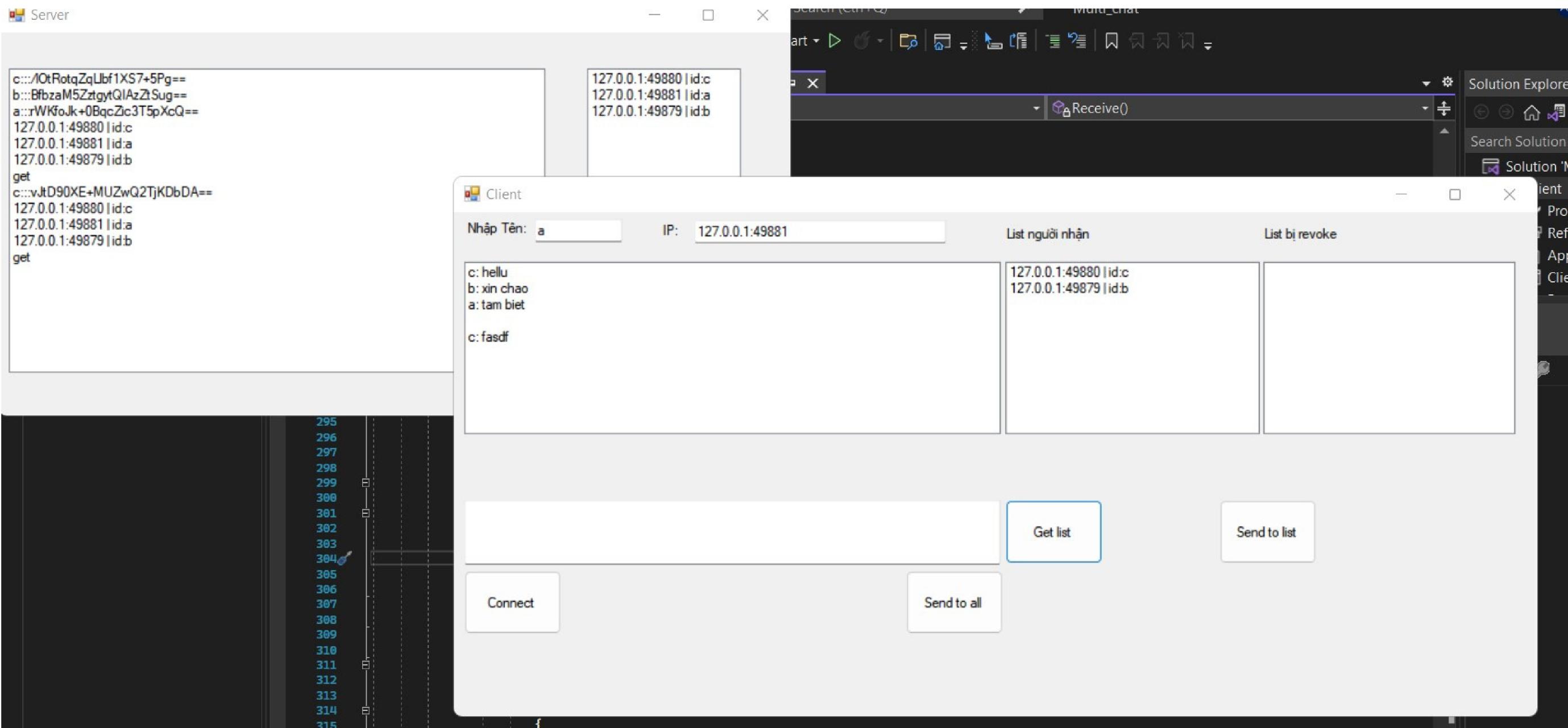
client

# Chức năng



gửi cho tất cả mọi người trong danh sách

# Chức năng



lấy danh sách người nhận đang kết nối tới server

## Code chính

```
3 references
public static class Cryptography1
{
    #region Settings

    private static int _iterations = 2;
    private static int _keySize = 256;

    private static string _hash = "SHA256";
    private static string _salt = "aselrias38490a32"; // Random
    private static string _vector = "8947az34awl34kjq"; // Random

    #endregion
}
```

Cài đặt

# Code chính

```
2 references
public static string Encrypt(string value, string password)
{
    return Encrypt<AesManaged>(value, password);
}

1 reference
public static string Encrypt<T>(string value, string password)
    where T : SymmetricAlgorithm, new()
{
    byte[] vectorBytes = ASCIIEncoding.ASCII.GetBytes(_vector);
    byte[] saltBytes = ASCIIEncoding.ASCII.GetBytes(_salt);
    byte[] valueBytes = ASCIIEncoding.ASCII.GetBytes(value);

    byte[] encrypted;
    using (T cipher = new T())
    {
        PasswordDeriveBytes _passwordBytes =
            new PasswordDeriveBytes(password, saltBytes, _hash, _iterations);
        byte[] keyBytes = _passwordBytes.GetBytes(_keySize / 8);

        cipher.Mode = CipherMode.CBC;

        using (ICryptoTransform encryptor = cipher.CreateEncryptor(keyBytes, vectorBytes))
        {
            using (MemoryStream to = new MemoryStream())
            {
                using (CryptoStream writer = new CryptoStream(to, encryptor, CryptoStreamMode.Write))
                {
                    writer.Write(valueBytes, 0, valueBytes.Length);
                    writer.FlushFinalBlock();
                    encrypted = to.ToArray();
                }
            }
            cipher.Clear();
        }
        return Convert.ToBase64String(encrypted);
    }
}
```

mã hóa

# Code chính

```
1 reference
public static string Decrypt(string value, string password)
{
    return Decrypt<AesManaged>(value, password);
}
1 reference
public static string Decrypt<T>(string value, string password) where T : SymmetricAlgorithm, new()
{
    byte[] vectorBytes = ASCIIEncoding.ASCII.GetBytes(_vector);
    byte[] saltBytes = ASCIIEncoding.ASCII.GetBytes(_salt);
    byte[] valueBytes = Convert.FromBase64String(value);

    byte[] decrypted;
    int decryptedByteCount = 0;

    using (T cipher = new T())
    {
        PasswordDeriveBytes _passwordBytes = new PasswordDeriveBytes(password, saltBytes, _hash, _iterations);
        byte[] keyBytes = _passwordBytes.GetBytes(_keySize / 8);

        cipher.Mode = CipherMode.CBC;

        try
        {
            using (ICryptoTransform decryptor = cipher.CreateDecryptor(keyBytes, vectorBytes))
            {
                using (MemoryStream from = new MemoryStream(valueBytes))
                {
                    using (CryptoStream reader = new CryptoStream(from, decryptor, CryptoStreamMode.Read))
                    {
                        decrypted = new byte[valueBytes.Length];
                        decryptedByteCount = reader.Read(decrypted, 0, decrypted.Length);
                    }
                }
            }
        catch (Exception ex)
        {
            return String.Empty;
        }

        cipher.Clear();
    }
    return Encoding.UTF8.GetString(decrypted, 0, decryptedByteCount);
}
```

giải mã

# Phân chia công việc

- An: code mã hóa, giải mã
- Tuấn: code kết nối server-client
- Hiển: tìm thuật toán, code giao diện app

**Thank you  
for listening**

