Laporan Project UTS Kelas Kriptografi(B) - 2023
Departemen Teknologi Informasi - Institut Teknologi Sepuluh Nopember

Anggota Kelompok 9 :

Muhammad Ilham Yumna	5027211024
Fransiskus Benyamin Sitompul	5027211021
Rifqi Akhmad Maulana	5027211035
Gilbert Immanuel Hasiholan	5027211056

Judul Studi Kasus :

Implementasi RSA dalam peer-to-peer messaging app

***Abstraksi** :*

Implementasi RSA pada P2P messaging

Enkripsi pesan pada P2P messaging app sudah menjadi sebuah hal yang lazim. Banyak dari perusahaan besar yang sudah menggunakannya di dalam aplikasinya, seperti Whatsapp, yang diakuisisi Meta pada tahun 2014, menggunakan enkripsi end to end pada penggunaan messagingnya.

Pada aplikasi kami, dilakukan enkripsi RSA pada saat mengirimkan pesan dan dilakukan dekripsi saat menerima pesan dari klien yang terhubung ke server yang sama.

Project overview ETEREMA (End-to-end RSA Encryption Messaging App) :

Project yang kami lakukan membuahkan sebuah sistem yang dimana, dari tampilan luar terdapat sisi client dan server. Enkripsi dan dekripsi RSA terdapat di antara komunikasi client dan server. Program disusun menggunakan bahasa python

Pertama, client 1 dan 2, dalam posisi memiliki public key dan private key, bertukar public key yang digunakan untuk enkripsi pesan, pertukaran public key melewati server. Setelah mendapatkan public key dari client, lalu kedua client bisa melakukan pertukaran pesan. Pesan terenkripsi terpantau di server induk.

Penambahan lapisan enkripsi RSA pada aplikasi messaging dapat menambah tingkat keamanan dan privasi pengguna saat bertukar pesan dengan user lain ,sehingga pesan asli user tidak dapat diketahui secara langsung

Implementasi program RSA :

Aplikasi python client dan server, dimana server bertugas meneruskan pesan yang sudah terenkripsi dari satu client ke client lainnya. Client disini mengenkripsikan pesan yang akan dikirim dan mendekripsi pesan yang diterimanya. Enkripsi dan Dekripsi menggunakan skema *encryption* Rivest-Shamir-Adleman yang mengimplementasikan sistem public key dan private key pada tiap clientnya. Data yang ditransmisikan adalah text(pesan).

Implementasi Client-Server :

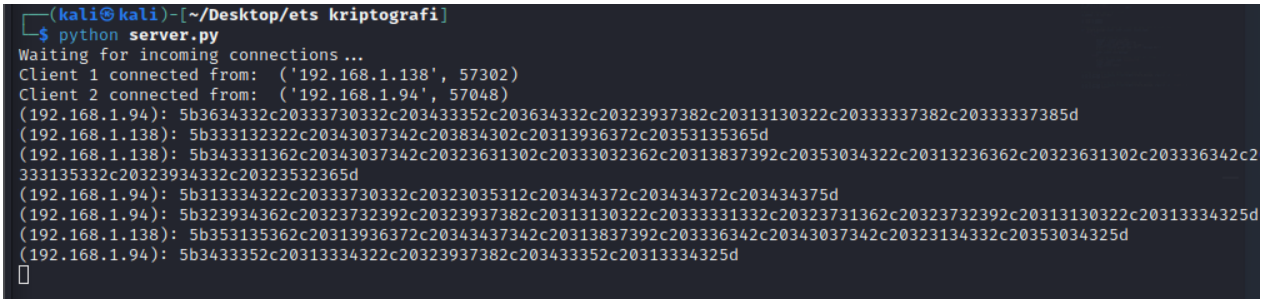
Diimplementasikan 3 point (2 Client dan 1 Server) yang terhubung melalui router dalam satu subnet. Server tidak bisa membaca pesan yang melewatinya karena pesan dalam bentuk terenkripsi. Hanya client yang dapat mengenkripsi dan mendekripsi pesan dari/kepada client lainnya.

Bahasa Pemrograman :

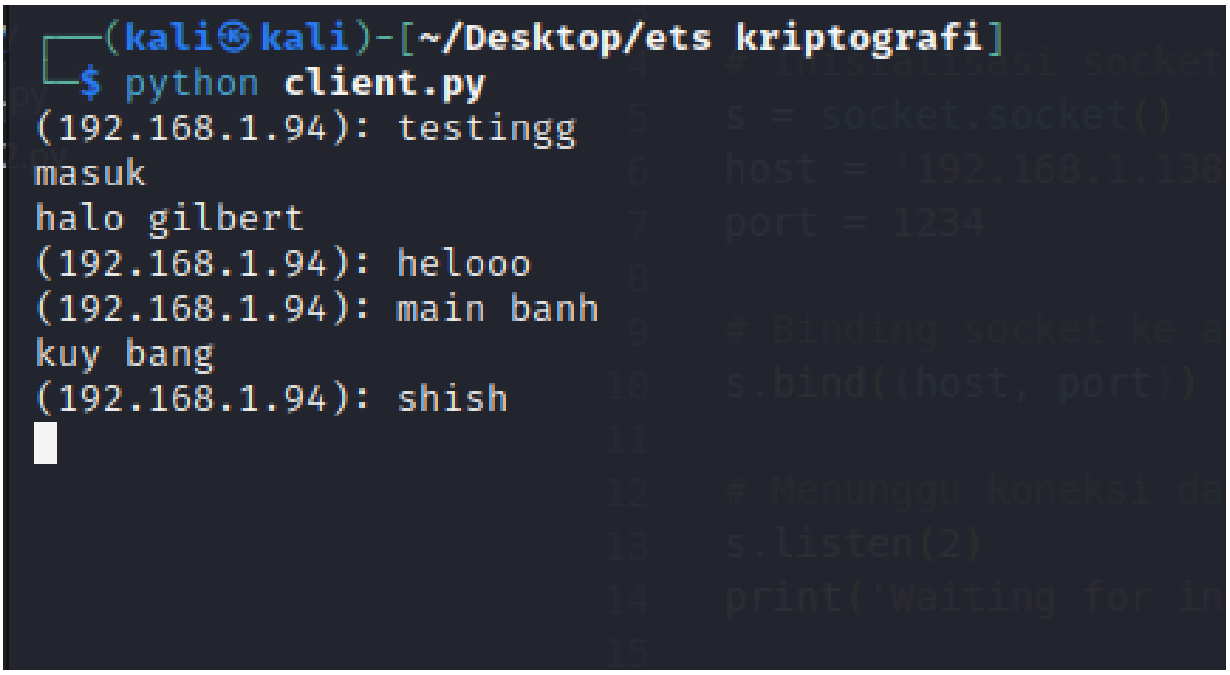
python

Dokumentasi Demo:

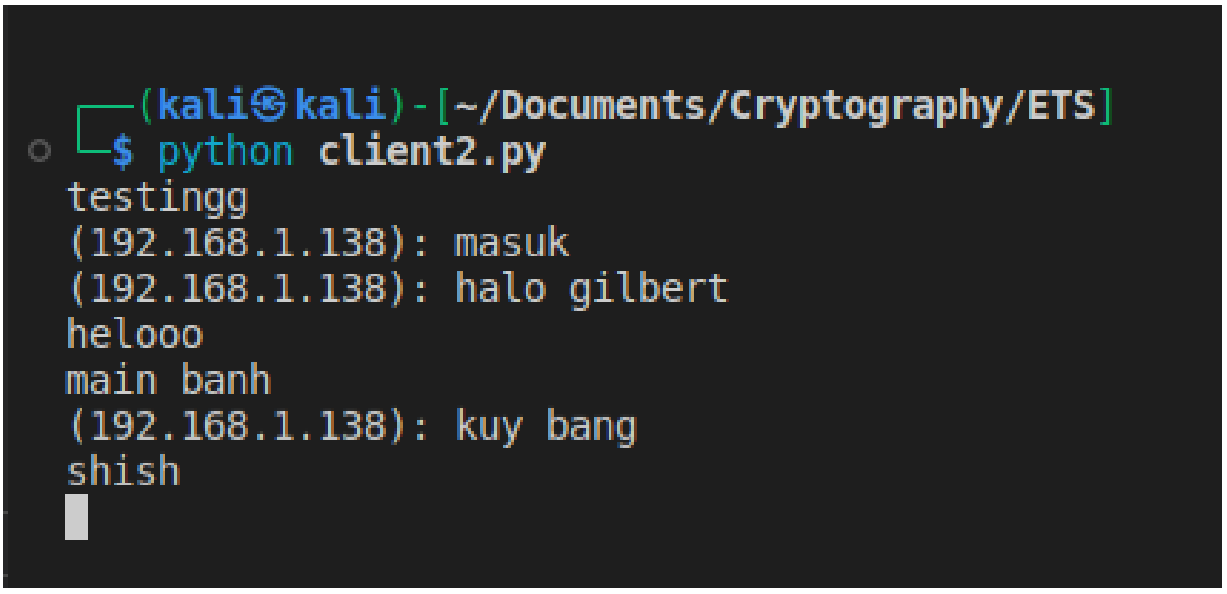
Server



Client 1



Client 2



Program:

client.py

```
import socket
import threading
import random
import math

# Fungsi untuk menghasilkan bilangan prima secara acak
def generate_prime_number():
    while True:
        # Pilih bilangan prima secara acak dari 1000 hingga 10000
        p = random.randint(1000, 10000)
        if is_prime(p):
            return p

# Fungsi untuk memeriksa apakah sebuah bilangan prima
def is_prime(n):
    if n == 2 or n == 3:
        return True
    if n == 1 or n % 2 == 0:
        return False
    for i in range(3, int(math.sqrt(n))+1, 2):
        if n % i == 0:
            return False
    return True

# Fungsi untuk menghasilkan bilangan coprime
def generate_coprime(p):
    while True:
        # Pilih bilangan acak dari 2 hingga p-1
        e = random.randint(2, p-1)
        if math.gcd(e, p-1) == 1:
            return e

# Fungsi untuk menghitung inversi modular menggunakan Algoritma
Extended Euclidean
def modinv(a, m):
    m0, x0, x1 = m, 0, 1
    if m == 1:
        return 0
    while a > 1:
        q = a // m
        a, m = m, a % m
```

```
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
        x1 += m0
    return x1

# Inisialisasi socket client
s = socket.socket()
host = '192.168.1.138'
port = 1234

# Terhubung ke server
s.connect((host, port))

# Menghasilkan bilangan prima secara acak
p = generate_prime_number()

# Menghasilkan bilangan coprime
e = generate_coprime(p)

# Menghitung bilangan d sebagai inversi modular dari e modulo p-1
d = modinv(e, p-1)

# Mengirimkan kunci publik ke server
pubkey = (p, e)
s.send(str(pubkey).encode())

# Menerima kunci publik client lain dari server
client_pubkey = eval(s.recv(1024).decode())

# Fungsi untuk mengenkripsi pesan menggunakan kunci publik
def encrypt(message, pubkey):
    p, e = pubkey
    # Konversi pesan menjadi bilangan menggunakan ASCII
    m = [ord(char) for char in message]
    # Mengenkripsi setiap bilangan menggunakan Algoritma Exponentiation
Modulo
    c = [pow(char, e, p) for char in m]
    return c

# Fungsi untuk mendekripsi pesan menggunakan kunci privat
def decrypt(c, privkey):
    p, d = privkey
```

```
    # Mendekripsi setiap bilangan menggunakan Algoritma Exponentiation
Modulo
    m = [chr(pow(char, d, p)) for char in c]
    # Menggabungkan setiap karakter menjadi pesan
    message = ''.join(m)
    return message

# Fungsi untuk menerima pesan dari server
def receive_message():
    while True:
        try:
            # Terima pesan dari server
            sender = s.recv(1024).decode()
            message = s.recv(1024).decode()
            # Dekripsi pesan
            message = decrypt(eval(message), (p, d))
            print(f"({sender}): {message}")
            # print(message)
        except:
            # Jika ada kesalahan koneksi, keluar dari thread
            break

#Thread untuk menerima pesan
receive_thread = threading.Thread(target=receive_message)
receive_thread.start()

#Loop untuk mengirim pesan ke server
while True:
    message = input()
    # Enkripsi pesan dengan kunci publik client lain
    encrypted_message = encrypt(message, client_pubkey)
    # Kirim pesan ke server
    s.send(str(encrypted_message).encode())
```

server.py

```
import socket
import threading
import random

# Beberapa fungsi enkripsi


def generate_rsa_keypair(p, q):
    n = p * q
    phi_n = (p-1) * (q-1)

    # Mencari nilai e yang relatif prima dengan phi_n
    e = random.randrange(1, phi_n)
    while gcd(e, phi_n) != 1:
        e = random.randrange(1, phi_n)

    # Mencari nilai d yang memenuhi syarat e*d mod phi_n = 1
    d = multiplicative_inverse(e, phi_n)

    # Mengembalikan kunci publik dan kunci privat
    return ((e, n), (d, n))


def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a


def multiplicative_inverse(a, m):
    """
    Mencari inversi multiplikatif dari a modulo m
    Menggunakan Algoritma Extended Euclidean
    """
    gcd, x, y = extended_euclidean(a, m)
    if gcd != 1:
        raise ValueError('Inversi multiplikatif tidak ada')
    return x % m


def extended_euclidean(a, b):
    """
```

```
    Algoritma Extended Euclidean
    Menghitung gcd(a, b) dan koefisien x, y yang memenuhi ax + by =
gcd(a, b)
    """
    if a == 0:
        return (b, 0, 1)
    else:
        gcd, x, y = extended_euclidean(b % a, a)
        return (gcd, y - (b // a) * x, x)

# func rsa_encode

def rsa_encode(pubkey, message):
    e, n = pubkey
    encoded_message = []
    for char in message:
        encoded_char = pow(ord(char), e, n)
        encoded_message.append(encoded_char)
    return encoded_message

# func rsa_decode


def rsa_decode(privkey, encoded_message):
    d, n = privkey
    decoded_message = ''
    for char_code in encoded_message:
        decoded_char_code = pow(char_code, d, n)
        decoded_char = chr(decoded_char_code)
        decoded_message += decoded_char
    return decoded_message


# Inisialisasi socket client
s = socket.socket()
host = '192.168.1.138'
port = 12345

# Terhubung ke server
s.connect((host, port))

# custom p & q values
p = 3
```

```
q = 11

pubkey, privkey = generate_rsa_keypair(p, q)

# Mengirimkan kunci publik ke server
s.send(pubkey.save_pkcs1())

# Menerima kunci publik client lain dari server
client_pubkey = s.recv(1024)

# Fungsi untuk menerima pesan dari server


def receive_message():
    while True:
        try:
            # Terima pesan dari server
            message = s.recv(1024)
            # Dekripsi pesan
            # message = rsa.decrypt(message, privkey).decode()
            message = rsa_decode(privkey, message)
            print(message)
        except:
            # Jika ada kesalahan koneksi, keluar dari thread
            break


# Thread untuk menerima pesan
receive_thread = threading.Thread(target=receive_message)
receive_thread.start()

# Loop untuk mengirim pesan ke server
while True:
    message = input()
    # Enkripsi pesan dengan kunci publik client lain
    # message = rsa.encrypt(message.encode(), client_pubkey)
    message = rsa_encode(pubkey, message)
    s.send(message)
```