Implementasi Algoritma AES dan RSA pada Riwayat Obrolan ChatGPT yang Disimpan di Penyimpanan *Cloud*

Christopher Jie – 18220052 (*Author*)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email: christopherjie123@gmail.com

***Abstrak*— Teknologi kecerdasan buatan ada di bibir semua orang dan telah banyak berkembang belakangan ini. Saat ini, teknologi kecerdasan buatan semakin efektif, efisien, dan akurat untuk memenuhi kebutuhan pengguna teknologi. Salah satu produk AI yang banyak digunakan saat ini adalah ChatGPT milik OpenAI yang dapat menerima permintaan pengguna melalui pesan teks dan kemudian AI tersebut merespon dengan memberikan respons pesan teks dengan hasil yang cukup akurat. Uniknya, ChatGPT dapat menerima permintaan untuk menghasilkan kode program. Sekarang semakin banyak pengguna ChatGPT dan semua pertanyaan dan jawaban dari ruang obrolan di ChatGPT mungkin disimpan di penyimpanan *cloud*. Namun, ChatGPT belum menerapkan prinsip kriptografi apa pun untuk menjaga kerahasiaan pesan pengguna dengan respons AI.**

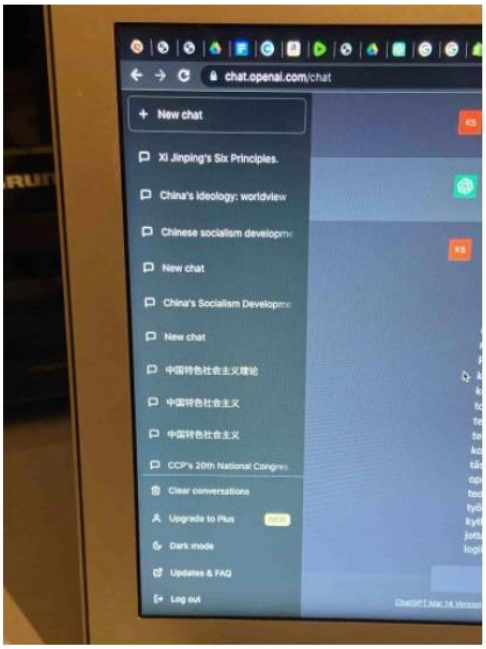
Kata kunci—Kriptografi, RSA, ChatGPT, AI (*Artificial Intelligence*), penyimpanan *cloud*

I. Pendahuluan

ChatGPT sudah digunakan oleh banyak orang dan pertanyaan mereka disimpan dan pengguna dapat melihatnya di *sidebar* sebelah kiri. ChatGPT tidak menerapkan enkripsi saat menyimpan riwayat obrolan dengan penggunanya [1]. Percakapan yang disimpan di penyimpanan *cloud* bisa sewaktu-waktu bocor karena adanya masalah pada kode. Selain informasi yang bocor dapat juga diakses secara tidak sah atau diretas melalui jaringan. Informasi obrolan yang tidak terenkripsi dapat diketahui oleh peretas dengan sangat mudah.

II. Rumusan Masalah

Menyimpan obrolan yang tidak terenkripsi di penyimpanan *cloud* dapat menghasilkan informasi sensitif dari obrolan orang di ChatGPT. Pada 23 Maret 2023, terjadi masalah berupa bocornya riwayat obrolan ChatGPT yang diungkapkan melalui situs media sosial Reddit dan Twitter. Pengguna membagikan gambar dari riwayat obrolan yang mereka klaim bukan milik mereka. Ini mengkhawatirkan banyak pengguna tentang privasi di platform ini. Jutaan orang telah menggunakan ChatGPT untuk menulis pesan, lagu, dan bahkan kode sejak diluncurkan November lalu, dan setiap percakapan dengan obrolan disimpan di bilah riwayat obrolan pengguna untuk referensi di masa mendatang jika diperlukan [2].



Gambar 1 Riwayat ChatGPT Bocor

(Sumber: https://www.hackread.com/chatgpt-bug-conversation-history-titles)

III. Tujuan

1. Memberikan usulan ide untuk meningkatkan keamanan informasi pada ChatGPT.
2. Melakukan percobaan implementasi sederhana dari ide yang diusul.

IV. Metodologi Penelitian

A. Studi Literatur

Bagian pertama yang dilakukan dalam pembuatan dokumen ini adalah studi literatur dari beberapa sumber. Studi literatur yang akan dijabarkan antara lain, sedikit teori tentang ChatGPT, teori kriptografi, teori algoritma RSA dan AES.

B. Gambaran Implementasi pada Sistem Asli

Setelah dilakukan studi literatur tentang teori yang akan digunakan, dilanjutkan dengan membuat gambaran bagaimana jika solusi yang diusulkan diimplementasikan pada sistem nyata. Gambaran ini tidak akan benar-benar dibuat hingga hasil jadi, tetapi hanya berupa diagram sistem atau diagram blok saja.

C. Eksperimen

Setelah dibuat gambaran implementasi pada sistem nyata, akan dilakukan sebuah eksperimen sederhana dengan membuat kode dan *database* yang menggambarkan atau menganalogikan sistem nyata.

D. Lingkup dan Batasan

Lingkup dan batasan dari dokumen ini adalah aspek yang ditinjau hanya sebatas keamanan informasi dengan topik penerapan kriptografi penyimpanan riwayat obrolan pada ChatGPT. Selain itu percobaan yang akan dilakukan tidak akan diimplementasikan pada sistem nyata.

V. Landasan Teori

A. ChatGPT

ChatGPT adalah model saudara dari InstructGPT, dilatih untuk mengikuti instruksi dengan cepat dan memberikan respons yang mendetail. ChatGPT didasarkan pada model RLHF (*Reforcement Learning from Human Feedback*), yang menggunakan metode yang sama dengan InstructGPT, tetapi dengan sedikit perbedaan dalam pengaturan pengumpulan data. Mitigasi keamanan yang diterapkan di ChatGPT versi terbaru adalah pengurangan yang signifikan dari hasil yang berbahaya dan salah yang diperoleh dengan *Reinforcement Learning from Human Feedback* (RLHF) [3].

Berdasarkan rumusan masalah yang sudah dibuat, ChatGPT terbukti menyimpan riwayat pesan di sistem mereka dan bukan di masing-masing akun pengguna. Oleh sebab itu, diperlukan upaya pengamanan riwayat pesan dengan kriptografi agar tidak diketahui oleh pengguna lain, apalagi ketika data tersebut bocor.

Salah satu upaya yang ChatGPT sudah upayakan dalam mencegah kejadian tersebut adalah memberikan fitur untuk menonaktifkan riwayat pesan untuk tersimpan pada tampilan pengguna [1].

Gambar 2 Logo ChatGPT

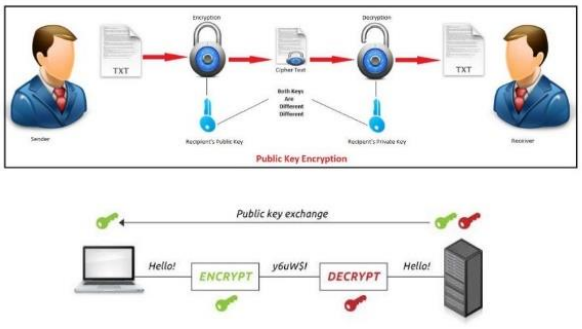
(Sumber: https://mailinvest.blog/index.php/2023/02/06/how-to-use-the-ai-chatbot-for-free/)

B. Kriptografi

Kriptografi merupakan keahlian dan ilmu dari cara-cara untuk komunikasi aman dari bocornya informasi ke pihak ketiga [4]. Berbagai aspek dalam keamanan informasi seperti data rahasia, integritas data, autentikasi, dan non-repudiasi [5]. Jenis kriptografi yang cocok untuk diimplementasikan dalam kasus ChatGPT ini adalah dengan menggunakan metode hybrid cryptography dengan menggabungkan algoritma AES untuk pesan teks dengan algoritma RSA untuk session key dari AES.

C. Algoritma RSA

RSA (Rivest-Shamir-Adleman) adalah kriptografi kunci publik yang biasa digunakan untuk transmisi data yang aman. Sistem ini juga salah satu yang tertua. Singkatan "RSA" berasal dari nama keluarga Ron Rivest, Adi Shamir dan Leonard Adleman, yang secara terbuka menggambarkan algoritma ini pada tahun 1977 [6].



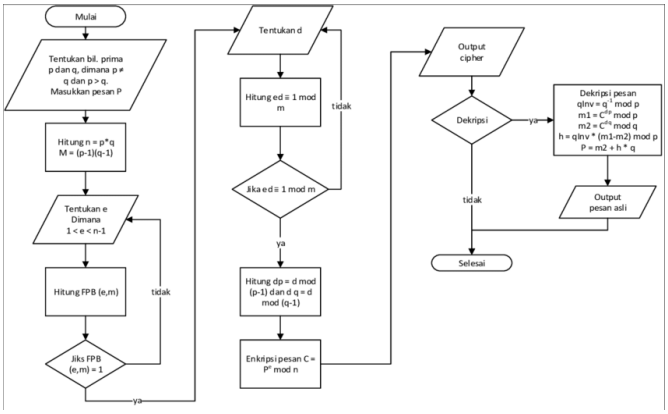
Gambar 3 Kriptografi Kunci Publik

(Sumber: https://image.slidesharecdn.com/rsa-140831052658-phpapp01/95/rsa-algorithm-9-1024.jpg?cb=1409462903)

Dalam kriptografi kunci publik, kunci enkripsi bersifat publik dan terpisah dari kunci dekripsi, yang dirahasiakan. Pengguna RSA membuat dan menerbitkan kunci publik berdasarkan dua bilangan prima yang besar dan nilai pemangkatan. Dua bilangan prima tersebut dirahasiakan. Siapa pun dapat mengenkripsi pesan dengan kunci publik, tetapi hanya mereka yang mengetahui bilangan prima yang dapat mendekripsinya [7].

Kelebihan dari keamanan RSA yaitu pada kesulitan praktis menghitung produk dari dua bilangan prima besar dan melakukan pemfaktoran merupakan tantangan terbesar dalam melakukan kriptanalisis dari algoritma RSA [8].

Pada kasus percakapan ChatGPT ini, kelemahan dari algoritma RSA adalah cenderung lambat apabila diterapkan pada pesan yang panjang karena pada algoritma RSA diterapkan operasi pemangkatan dengan bilangan yang relatif sangat besar. Oleh sebab itu, pada kasus ini penggunaan algoritma RSA adalah untuk mengenkripsi kunci simetri dari AES.



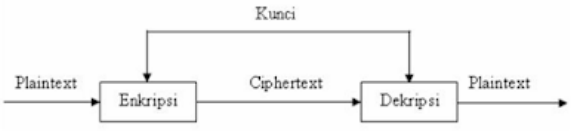
Gambar 4 *Flowchart* Algoritma RSA

(Sumber: https://www.researchgate.net/figure/Gambar-4-Flowchart-Proses-Algoritma-RSA-CRT_fig2_321310951)

D. Algoritma AES

Advanced Encryption Standard (AES) atau dikenal juga dengan nama Rijndael [9] adalah spesifikasi untuk enkripsi data elektronik yang didirikan oleh Institut Standar dan Teknologi Nasional AS (NIST) pada tahun 2001 [10].

AES adalah varian dari sandi blok Rijndael [9] yang dikembangkan oleh dua kriptografer Belgia, Joan Daemen dan Vincent Rijmen, yang mengajukan proposal ke NIST. Rijndael adalah keluarga cipher dengan ukuran kunci dan blok yang berbeda yaitu 128 bit, 192 bit, dan 256 bit kunci. Dalam praktiknya, algoritma AES adalah algoritma kriptografi kunci simetris yang mana kunci untuk enkripsi dan dekripsi pesan adalah sama. Dalam kasus mengenkripsi pesan teks ini, Algoritma AES masih aman dari serangan dan cenderung lebih cepat daripada algoritma RSA, sehingga akan lebih efektif dan efisien dalam melakukan enkripsi dan dekripsi pesan teks yang panjang.

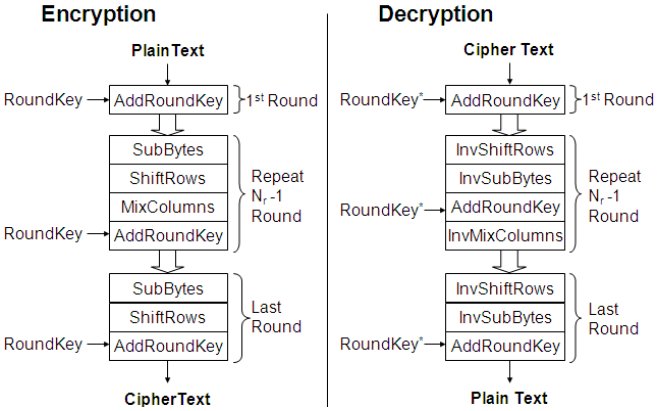


Gambar 5 Kriptografi Kunci Simetris

(Sumber: https://io32.blogspot.com/2020/06/algoritma-kriptografi.html)

Dalam prosesnya algoritma AES memerlukan beberapa komponen dalam melakukan enkripsi dan dekripsi sebagai berikut.

1. *AddRoundKey*: Melakukan operasi XOR antara *plain text* dengan *key*.
2. *Sub Bytes*: Proses menukar tabel yang sudah ada dengan tabel Rijndael *S-Box*.
3. *Shift Rows*: Melakukan pergeseran matriks ke kiri sesuai dengan barisnya. Pada baris pertama tidak melakukan pergeseran, baris kedua melakukan pergeseran ke kiri 1 *byte*, baris ketiga melakukan pergeseran ke kiri 2 *byte*, baris keempat melakukan pergeseran ke kiri 3 *byte*.
4. *Mix Columns*: Melakukan perkalian setiap elemen *block cipher* dengan matriks yang sudah ada [11].



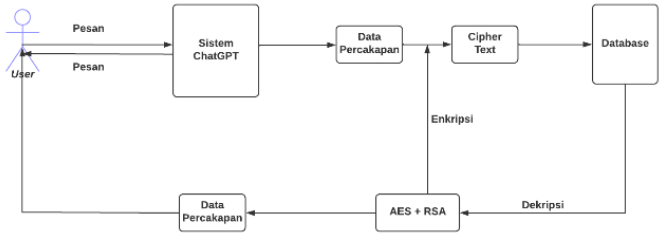
Gambar 6 *Flowchart* Algoritma AES

(Sumber: https://www.researchgate.net/figure/AES-Encryption-Decryption-Flowchart_fig2_221958203)

VI. Pembahasan Implementasi

A. Sistem ChatGPT

Bagian ini membahas hal yang lebih luas berupa gambaran sederhana terkait perubahan yang terjadi pada sistem ChatGPT setelah adanya tambahan elemen kriptografi di dalam sistem ChatGPT. Secara singkat dapat dijelaskan bahwa pengguna tetap seperti biasa bertukar pesan dengan sistem ChatGPT yang kemudian data percakapan akan disimpan oleh sistem ke *database* dalam bentuk *cipher text* setelah dilakukan enkripsi terlebih dahulu dengan menggunakan algoritma AES untuk *plain text* dan algoritma RSA untuk *session key* AES. Sebaliknya, *plain text* yang berada pada tampilan pengguna merupakan hasil dari dekripsi dengan menggunakan algoritma AES untuk *cipher text* dan algoritma RSA untuk *encrypted session key* AES. Dengan adanya elemen kriptografi dalam sistem ini, dapat dipastikan bahwa sistem ChatGPT lebih aman dan sulit sekali untuk mendapatkan informasi percakapan pengguna sekalipun penyerang berhasil mengakses *database*. Satu-satunya cara agar bisa melihat percakapan tersebut hanya ketika penyerang berhasil masuk ke akun pengguna terkait.

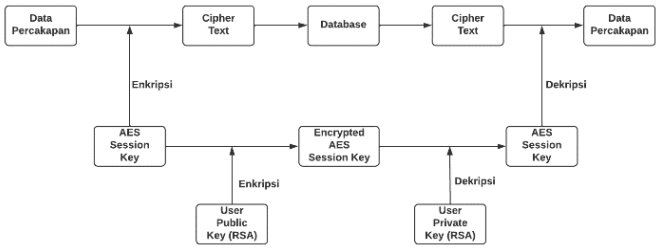


Gambar 7 Sistem ChatGPT dengan Kriptografi

(Sumber: Dokumentasi Penulis)

B. Mekanisme Kriptografi

Bagian ini membahas hal yang lebih teknis dan inti berupa gambaran dari cara kerja kriptografi dengan algoritma AES dan RSA pada sistem ChatGPT ini. Secara singkat dapat dijelaskan bahwa data percakapan pengguna dengan AI ChatGPT dienkripsi dahulu dengan algoritma AES dengan kunci sekali pakai (*session key*), hingga menjadi *cipher text*. Kemudian *session key* AES dienkripsi lagi dengan kunci publik pengguna dengan algoritma RSA hingga menjadi *encrypted session key*. Setelah itu *cipher text* disimpan dalam *database*. Pengguna yang sewaktu-waktu mengakses ChatGPT pastinya memiliki *private key* RSA untuk mendekripsi *encrypted session key* kemudian mendekripsi pesan yang disimpan di *database* agar dapat dilihat kembali oleh pengguna saat ingin melihat riwayat pesan. Dengan adanya *private key* RSA di setiap akun pengguna juga dapat menghindari kejadian berupa kebocoran data percakapan pengguna di akun orang lain lagi karena saat data bocor sekalipun akibat *bug*, data yang ditampilkan di akun orang lain masih dalam bentuk *cipher text*, sehingga apabila ada percakapan yang sensitif atau rahasia tidak akan diketahui oleh pengguna lainnya.



Gambar 8 Mekanisme Kriptografi AES dan RSA

(Sumber: Dokumentasi Penulis)

VII. Eksperimen

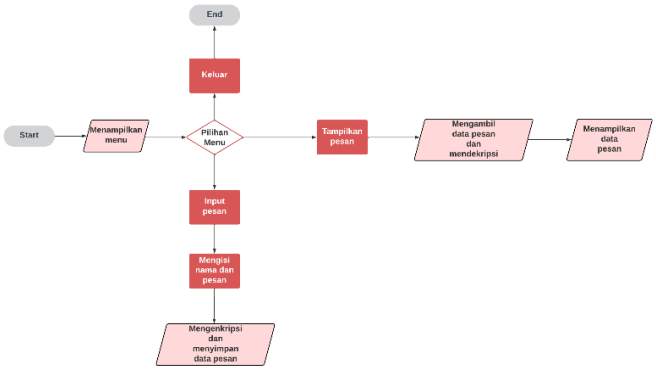
Setelah dibuat gambaran dari implementasi sistem baru dengan kriptografi beserta gambaran mekanisme atau cara kerja dari algoritma kriptografi yang sudah dipilih, maka dilakukan sedikit percobaan terkait dengan ide yang diusulkan. Percobaan ini tidak merepresentasikan lingkungan sistem yang asli dan hanya bersifat pembuktian dari konsep (*proof of concept*) yang sudah dibuat. Batasan dari uji coba ini adalah aspek yang diuji hanya sebatas pada elemen kriptografi dan autentikasi pengguna saja.

Percobaan ini dilakukan dengan bahasa pemrograman Python dengan CMD dan *cloud database* direpresentasikan dengan MongoDB. Asumsikan pula ketika memasukkan nama pengguna pada program sama artinya dengan pengguna yang sudah *register* dan *login* dengan akun.

A. Rancangan Eksperimen

Sebagai tahap awal dari percobaan, ada beberapa hal yang harus dibuat yang terdiri dari *flowchart* untuk menggambarkan logika dan alur dari program yang akan dibuat serta ekspektasi dari *input, output,* dan tampilan pada *database* jika program selesai dibuat.

a. *Flowchart*



Gambar 9 *Flowchart* Program

(Sumber: Dokumentasi Penulis)

b. *Input*

```
Masukkan Nama Pengguna: Rick
Masukkan Pesan: Never gonna give
you up!
Pesan berhasil dienkripsi dan
disimpan ke basis data
```

c. *Output*

- **Jika pengguna sama (teks pesan akan muncul):**

```
Masukkan Nama Pengguna: Rick
Rick: Never gonna give you up!
```

- **Jika pengguna berbeda (teks pesan tidak akan muncul):**

```
Masukkan Nama Pengguna: Rickroll
```

d. Tampilan pada *Database* (MongoDB)


```
{
    "_id": {
    "$oid":
    "6455eb34095c1f48fa394da4"
},
    "name": "Rick",
    "message":
    "DRWBd/DJUi5xqXtpScneJaZEAt9fotom
    dnL5SQniRWzGTwTjomwNSp31VRNceeuYJ
    y8CxaSwFDca1kRGL8G9Y6e28SavmxjUAm
    Hy4yZArYtly0U/DE66HZTWMD8DT9ycp43
    lY/h+as6IEAa7aIrY01wDAHXkx3781qqH
    NbKKQ2vY8N6MyRYdx4SY8vJB98d1WyV+H
    X5GBIMkrUSpi8HqpOvCgFN/XmF6F/vusf
    xdiVDeOIzjE2dBxbS1hdpCFu67DCxAI4P
    3UeWCNcuq97cqloZc3iv0nb5Aadv6YVvy
    tKAe+f8XNgTlSdZloKlCI8rEl5Ti9y8bs
    yjkpZhrDcGHD9c7q7Q/PG/lTm4F5RPfFa
    UHG2BGfC8VutOwjAXhv/FA/EpUG4Vq+Nx
    VqW50XcMnCJjZfHyW8ehMWg8duAt+Zg8/
    j0/+5jdcRfXVtekDTjUXaeW6m4MsYKTRk
    lDfhfT2BHMVQq0Mc5j7WLHrVJUukkBAfw
    pityDjUDgkH2kfvcfjcMfGgZ0z0bxgnIz
    +/f/9uyRUAv7OCasBj/KkdS3qHHvUneEP
    Vp3puD/aq/08TDjQ0O8jGsN5cfHXvRa1f
    iSx"
}
```


B. Hasil Eksperimen

```
--- ChatGPT ---
1. Input chat
2. Tampilkan chat
3. Keluar
Pilih menu:
```

Gambar 10 Tampilan Menu

(Sumber: Dokumentasi Penulis)

Pada tampilan menu terdapat 3 pilihan yaitu opsi 1 *input chat* yang berfungsi untuk meminta nama pengguna, pesan teks, mengenkripsi pesan teks kemudian dikirimkan ke *database*. Opsi 2 Tampilkan *chat* berfungsi untuk mendekripsi *cipher text* dari *database,* menampilkan teks dan memvalidasi nama pengguna. Opsi 3 yaitu keluar untuk keluar dari program.

```
--- ChatGPT ---
1. Input chat
2. Tampilkan chat
3. Keluar
Pilih menu: 1
Masukkan nama pengguna: Alice
Masukkan pesan: halo pak! selamat siang perkenalkan nama saya Alice :)
Pesan berhasil dienkripsi dan disimpan ke basis data
```

Gambar 11 *Input* Nama dan Pesan

(Sumber: Dokumentasi Penulis)

```
--- ChatGPT ---
1. Input chat
2. Tampilkan chat
3. Keluar
Pilih menu: 2
Masukkan nama pengguna: Alice
Alice: halo pak! selamat siang perkenalkan nama saya Alice :)
```

Gambar 12 *Output* Nama dan Pesan

(Sumber: Dokumentasi Penulis)

```
--- ChatGPT ---
1. Input chat
2. Tampilkan chat
3. Keluar
Pilih menu: 2
Masukkan nama pengguna: Alise

--- ChatGPT ---
1. Input chat
2. Tampilkan chat
3. Keluar
Pilih menu:
```

Gambar 13 Autentikasi Pengguna

(Sumber: Dokumentasi Penulis)

Berdasarkan ketiga gambar yang ada di atas, dapat dibuktikan bahwa hanya pengirim pesan saja yang dapat melihat kembali pesan yang sudah dikirim. Apabila pesan tersebut hendak diakses oleh pengguna yang berbeda, maka pesan tersebut tidak akan ditampilkan. Pesan tersebut hanya dapat ditampilkan jika yang meminta adalah pengguna yang mengirim pesan tersebut.

Gambar 14 Tampilan *Database*

(Sumber: Dokumentasi Penulis)

Gambar di atas merupakan tampilan dari *database* dan riwayat percakapan dari Alice yang sudah berbentuk *cipher text*. Hal ini membuktikan bahwa pesan yang dikirim Alice yang disimpan sulit untuk diketahui maknanya apabila sewaktu-waktu diretas oleh penyerang. Walaupun demikian Alice tetap dapat mengerti pesan tersebut apabila dia meminta pada aplikasi untuk menampilkan riwayat pesannya kembali dengan cara didekripsi dari *database*.

VIII. Kesimpulan dan Saran

A. Kesimpulan

Apabila ditinjau dari algoritma kriptografi yang dipilih yaitu AES dan RSA pada ide yang diusul, dapat disimpulkan bahwa jika elemen kriptografi ini diterapkan pada sistem asli, maka tingkat keamanan pada pesan akan tinggi karena algoritma AES sendiri masih sangat sulit untuk diserang, apalagi jika kunci dari AES dienkripsi lagi dengan RSA yang memiliki kompleksitas yang cukup tinggi akan lebih sulit lagi untuk diretas. Selain itu, untuk percobaan

sederhana yang sudah dilakukan dinyatakan berhasil karena sudah dapat mengimplementasikan fungsi-fungsi kriptografi sesuai dengan teori dan ide yang diajukan.

B. Saran

Saran pengembangan pada percobaan yang dilakukan adalah dapat ditambahkan GUI untuk memperindah tampilan, kemudian dapat ditambahkan lagi fungsionalitasnya pada bagian *chat* sehingga dapat berjalan dua arah.

VIDEO LINK AT YOUTUBE

https://youtu.be/J84eFmBsR5I

Referensi

[1] OpenAI. 2022. https://chat.openai.com/, diakses pada Sabtu, 29 April 2023.

[2] Derico, Ben. (2023, Maret 23). https://www.bbc.com/news/technology-65047304, diakses pada Sabtu, 29 April 2023.

[3] OpenAI. (2022, November 30). *Introducing ChatGPT*. https://openai.com/blog/chatgpt, diakses pada Sabtu, 29 April 2023.

[4] Rivest, Ronald L. (1990). "Cryptology". Dalam J. Van Leeuwen. *Handbook of Theoretical Computer Science*. **1**. Elsevier.

[5] Menezes, A. J.; van Oorschot, P. C.; Vanstone, S. A. (2005). *Handbook of Applied Cryptography*.

[6] Smart, Nigel (February 19, 2008). "Dr Clifford Cocks CB".

[7] Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems".

[8] Castelvecchi, Davide (2020-10-30). "Quantum-computing pioneer warns of complacency over Internet security".

[9] Daemen, Joan; Rijmen, Vincent (March 9, 2003). "AES Proposal: Rijndael" (PDF). National Institute of Standards and Technology. P. 1.

[10] "Announcing the ADVANCED ENCRYPTION STANDARD (AES)" (PDF). *Federal Information Processing Standards Publication 197*. United States National Institute of Standards and Technology (NIST).

[11] Purnomo, Imam. (2022, November 12). https://purnomo-imam.medium.com/algoritma-kriptografi-modern-aes-c82f68decaad, diakses pada Sabtu, 6 Mei 2023.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Christopher Jie (18220052)

Lampiran

[1] Kode Program:

```
import pymongo
import base64
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Cipher import AES
from Crypto.Random import
get_random_bytes


# Koneksi ke basis data MongoDB
client =
pymongo.MongoClient("mongodb://localho
st:27017/")
db = client["Chat"]
col = db["Chat"]


# Generate key RSA
key = RSA.generate(2048)


# Fungsi untuk melakukan enkripsi
pesan
def encrypt_message(message):
    public_key = key.publickey()
    cipher_rsa =
PKCS1_OAEP.new(public_key)
    # Generate random key untuk AES
    session_key = get_random_bytes(16)
    # Enkripsi pesan menggunakan AES
    cipher_aes = AES.new(session_key,
AES.MODE_EAX)
    ciphertext, tag =
cipher_aes.encrypt_and_digest(message.
encode())
    # Enkripsi session key menggunakan
RSA
    encrypted_session_key =
cipher_rsa.encrypt(session_key)
    # Gabungkan data enkripsi menjadi
satu pesan yang terenkripsi
    encrypted_message =
encrypted_session_key +
cipher_aes.nonce + tag + ciphertext
    return
base64.b64encode(encrypted_message).de
code()


# Fungsi untuk melakukan dekripsi
pesan
def
decrypt_message(encrypted_message):
    private_key = key
    cipher_rsa =
PKCS1_OAEP.new(private_key)
    # Potong data enkripsi menjadi
bagian-bagian
    encrypted_message =
base64.b64decode(encrypted_message)
```

```
    encrypted_session_key =
encrypted_message[:private_key.size_in
_bytes()]
    nonce =
encrypted_message[private_key.size_in_
bytes():private_key.size_in_bytes()+16
]
    tag =
encrypted_message[private_key.size_in_
bytes()+16:private_key.size_in_bytes()
+32]
    ciphertext =
encrypted_message[private_key.size_in_
bytes()+32:]
    # Dekripsi session key menggunakan
RSA
    session_key =
cipher_rsa.decrypt(encrypted_session_k
ey)
    # Dekripsi pesan menggunakan AES
    cipher_aes = AES.new(session_key,
AES.MODE_EAX, nonce=nonce)
    decrypted_message =
cipher_aes.decrypt_and_verify(cipherte
xt, tag)
    return decrypted_message.decode()


# Halaman untuk menginput chat yang
sudah terenkripsi ke basis data
def input_chat():
    name = input("Masukkan nama
pengguna: ")
    message = input("Masukkan pesan:
")
    encrypted_message =
encrypt_message(message)
    data = {"name": name, "message":
encrypted_message}
    col.insert_one(data)
    print("Pesan berhasil dienkripsi
dan disimpan ke basis data")


# Halaman untuk mengambil hasil chat
yang sudah didekripsi dari basis data
def get_chat():
    name = input("Masukkan nama
pengguna: ")
    results = col.find({"name": name})
    for result in results:
        decrypted_message =
decrypt_message(result["message"])
        print(result["name"] + ": " +
decrypted_message)


# Main program
while True:
    print("\n--- ChatGPT ---")
    print("1. Input chat")
    print("2. Tampilkan chat")
    print("3. Keluar")
    choice = input("Pilih menu: ")
    if choice == "1":
        input_chat()
    elif choice == "2":
        get_chat()
    elif choice == "3":
        break
    else:
        print("Menu tidak tersedia")
```

[2] Link Github: https://github.com/MAINSETS/Makalah