Enkripsi RSA pada Pesan Suara Menggunakan Pulse Code Modulation

Irgiansyah Mondo - 13521167
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
irgimondo@gmail.com, 13521167@std.stei.itb.ac.id

***Abstract*— Perkembangan pesat teknologi memerlukan keamanan data yang kuat, terutama dalam transmisi audio. Penerapan RSA untuk mengamankan pesan audio menggunakan Pulse Code Modulation (PCM) adalah salah satu cara untuk memastikan kerahasiaan dan integritas dengan menyediakan metode transmisi yang aman dan praktis.**

***Keywords*—Enkripsi audio, RSA, Kriptografi, Pulse Code Modulation.**

I. PENDAHULUAN

Seiring berkembangnya zaman manusia yang semakin didukung oleh pesatnya bantuan teknologi untuk mempermudah komunikasi dan akses informasi. Dengan adanya teknologi sekarang, industri teknologi informasi menjembatani manusia dalam berkomunikasi yang dimana kemudahan ini tidak hanya meningkatkan efisiensi berbagai aktivitas, tetapi juga menciptakan era baru di mana data menjadi salah satu aset paling berharga. Namun, di balik semua manfaat yang ditawarkan, ancaman terhadap privasi dan keamanan informasi juga meningkat secara signifikan. Serangan siber, peretasan, dan penyadapan menjadi isu global yang memerlukan perhatian serius.

Untuk menjawab tantangan ini, berbagai metode telah dikembangkan guna memastikan keamanan data, salah satunya adalah kriptografi. Sebagai ilmu yang berfokus pada pengamanan data melalui transformasi informasi, kriptografi telah menjadi solusi andal dalam menjaga kerahasiaan pesan. Salah satu algoritma yang menonjol dalam bidang ini adalah algoritma RSA, sebuah metode kriptografi kunci asimetris yang terkenal karena kekuatannya dalam melindungi data. Dengan menggunakan pasangan kunci publik dan privat, RSA memungkinkan proses enkripsi dan dekripsi berjalan dengan tingkat keamanan yang tinggi.

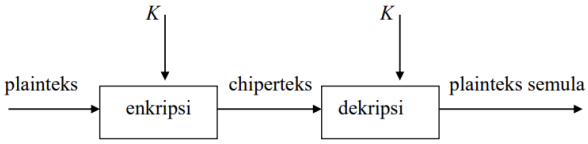
Dalam konteks komunikasi berbasis suara, kebutuhan akan keamanan menjadi semakin penting. Pesan suara sering digunakan dalam aplikasi perpesanan instan, panggilan VoIP, maupun komunikasi profesional. Mengingat sifat pesan suara yang rentan terhadap penyadapan, diperlukan langkah-langkah inovatif untuk melindungi integritas dan kerahasiaannya. Salah satu pendekatan yang dapat diadopsi adalah penerapan algoritma RSA, yang mampu menjamin bahwa hanya pihak yang berwenang dapat mengakses konten pesan.

II. TEORI DASAR

A. Kriptografi

Kriptografi, atau dalam bahasa Inggris disebut cryptography, berasal dari dua kata dalam bahasa Yunani: kryptos yang berarti "tersembunyi" atau "rahasia" dan graphia yang berarti "tulisan." Ilmu ini berfokus pada cara melindungi informasi dengan menggunakan teknik-teknik matematika. Kriptografi menjadi salah satu elemen penting dalam keamanan informasi modern, dengan mencakup aspek-aspek seperti kerahasiaan, integritas, autentikasi, dan keabsahan data.

Dalam penerapannya, kriptografi bekerja dengan dua jenis pesan: plaintext yang merupakan pesan asli sebelum disandikan, dan ciphertext yang merupakan hasil pesan setelah melalui proses penyandian. Agar ciphertext dapat dikembalikan menjadi plaintext, dibutuhkan kunci khusus dan metode enkripsi yang digunakan. Proses penyandian dari plaintext menjadi ciphertext dikenal sebagai enkripsi, sedangkan pengembalian ciphertext ke plaintext disebut dekripsi. Dengan mekanisme ini, hanya pihak yang berwenang yang dapat membaca pesan asli, sehingga keamanan dan privasi informasi tetap terjaga.



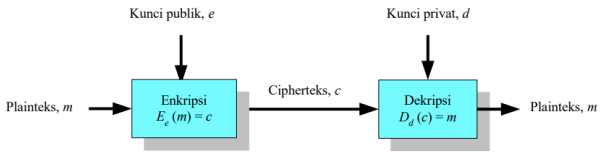
Gambar 1. ilustrasi enkripsi dan dekripsi (sumber :https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/17-Teori-Bilangan-Bagian3-2024.pdf)

B. Algoritma RSA

Algoritma RSA merupakan salah satu algoritma kriptografi asimetris yang dikembangkan pada tahun 1976 oleh tiga peneliti dari Massachusetts Institute of Technology (MIT), yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman. Algoritma ini dikenal sebagai metode kriptografi kunci publik (public-key cryptography), di mana kunci untuk enkripsi berbeda dengan kunci untuk dekripsi. Setiap pengguna dalam RSA memiliki sepasang kunci, yaitu kunci publik (e) dan kunci privat (d). Kunci publik digunakan untuk mengenkripsi pesan dan bersifat terbuka sehingga dapat diumumkan kepada publik. Sebaliknya,

kunci privat digunakan untuk mendekripsi pesan dan bersifat rahasia, hanya diketahui oleh pemilik kunci.

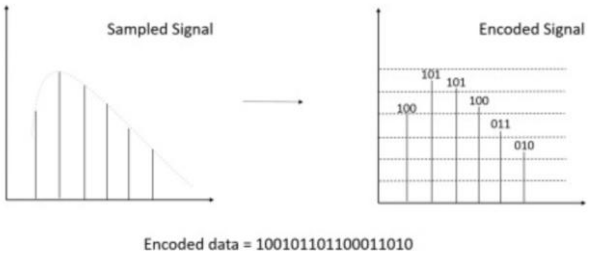
Proses pembangkitan pasangan kunci dalam algoritma RSA dimulai dengan memilih dua bilangan prima, p dan q, yang bersifat rahasia. Nilai n dihitung sebagai hasil perkalian p dan q, dengan n ini akan menjadi bagian dari kunci publik. Selanjutnya, nilai m dihitung sebagai (p−1)(q−1), yang tetap dirahasiakan. Kemudian, sebuah bilangan bulat e dipilih sebagai kunci publik, dengan syarat e relatif prima terhadap m, atau PBB(e,m)=1. Setelah itu, nilai d sebagai kunci privat dihitung dengan memenuhi persamaan ed≡1 (mod m), di mana d adalah invers modular dari e terhadap m.



Gambar 2. Ilustrasi RSA (sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/17-Teori-Bilangan-Bagian3-2024.pdf)

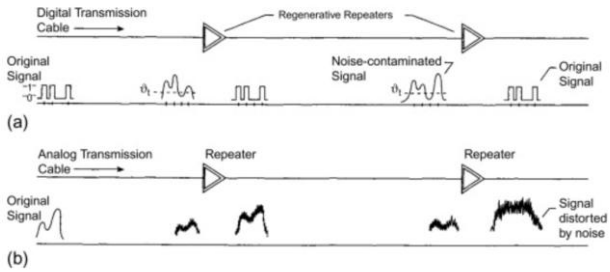
C. Pulse Code Modulation

Pulse Code Modulation (PCM) adalah teknik modulasi digital yang digunakan untuk mengubah informasi analog menjadi urutan biner, yaitu kombinasi angka 1 dan 0. Proses PCM melibatkan pengubahan sinyal pesan menjadi bentuk diskrit dalam domain waktu dan amplitudo, sehingga sinyal analog dapat direpresentasikan oleh serangkaian pulsa berkode.



Gambar 3. ilustrasi konversi sinyal (sumber : Martin Plonus, in Electronics and Communications for Scientists and Engineers (Second Edition) 2020)

Keunggulan utama dari PCM terletak pada fakta bahwa pesan yang dikirimkan berupa rangkaian pulsa dengan tinggi yang sama, sehingga dapat diregenerasi hampir sempurna sepanjang jalur propagasi. Informasi dalam PCM tidak bergantung pada tinggi atau bentuk spesifik pulsa, tetapi lebih pada keberadaan pulsa itu sendiri. Sebuah pulsa yang mengalami deformasi bentuk atau ukuran masih dapat dikenali di sisi penerima sebagai keberadaan sebuah bit, yaitu "1" atau "0". Sebelum deretan pulsa menjadi terlalu terdistorsi sehingga sulit membedakan antara "1" dan "0", rangkaian pulsa dapat diregenerasi kembali ke bentuk aslinya menggunakan pengulang regeneratif (regenerative repeater).



Gambar 4. Perbandingan transmisi sinyal digital dan analog (sumber : Sri Krishnan, in Biomedical Signal Analysis for Connected Healthcare, 2021)

Regenerative repeater bekerja dengan cara mengambil sampel dari sinyal yang terdistorsi pada titik tengah periode simbol digital dan memutuskan apakah sinyal tersebut di atas atau di bawah ambang batas tegangan tertentu (υt). Jika sampel di atas ambang batas, sinyal dianggap sebagai "1", sedangkan jika di bawah, sinyal dianggap sebagai "0". Dengan demikian, pengulang PCM bukanlah penguat konvensional, melainkan sirkuit yang dirancang untuk mendeteksi keberadaan atau ketidakhadiran pulsa listrik. Tambahan noise dalam jumlah moderat tidak akan memengaruhi kemampuan deteksi pulsa, sedangkan noise dalam jumlah besar diperlukan untuk memberikan indikasi palsu pada pulsa yang sebenarnya tidak ada.

Proses PCM dimulai dengan menyaring sinyal analog menggunakan low-pass filter untuk memastikan tidak ada frekuensi di atas $fmax$, yang disebut sebagai antialiasing filter. Kemudian, sinyal analog diambil sampelnya pada laju sampling Nyquist ($2fmax$) oleh sebuah rangkaian clock yang menghasilkan pulsa dengan frekuensi tersebut. Setelah itu, proses kuantisasi dilakukan untuk membagi sinyal ke dalam $2n$ level dan membulatkan nilai setiap sampel. Hasil dari sampling dan kuantisasi ini disebut sebagai digitalisasi, menghasilkan sinyal berupa pulsa dengan amplitudo yang bervariasi pada laju sampling, yang dikenal sebagai sinyal Pulse-Amplitude Modulation (PAM).

Langkah berikutnya adalah mengonversi sinyal PAM ini menjadi sinyal digital dengan cara mengkodekan setiap nilai kuantisasi ke dalam bilangan biner. Bit-bit dari bilangan biner tersebut kemudian direpresentasikan dalam bentuk pulsa sederhana on-off, yang kemudian dikelompokkan dan ditransmisikan secara berurutan. Karena semua pulsa yang dikirimkan memiliki amplitudo yang sama, sinyal ini lebih kebal terhadap noise aditif, karena penerima hanya perlu mendeteksi keberadaan atau ketidakhadiran pulsa.

Di sisi penerima, proses ini dibalik. Data biner yang diterima didekodekan kembali menjadi sinyal analog "berbentuk tangga" yang kemudian dihaluskan menggunakan low-pass filter dengan bandwidth $fmax$, sehingga sinyal asli berhasil direkonstruksi. PCM inilah yang menjadi dasar komunikasi digital modern, memungkinkan integrasi yang mudah antara jaringan komunikasi digital dan komputer, sehingga memunculkan teknologi seperti Local Area Networks (LANs) dan Internet.

III. PEMBAHASAN

A. Perancangan Kode

Rancangan kode ini dibuat menggunakan bahasa python dengan memanfaatkan *library* PyCrytodome untuk pustaka kunci RSA dan *wave* untuk pustaka input dan output file audio. Berikut adalah implementasi dari kode algoritma RSA ini :

1. Header

   Header pada kode di atas memiliki fungsi untuk mendukung pengolahan data audio dan implementasi enkripsi RSA. Modul wave digunakan untuk membaca, memodifikasi, dan menulis file audio dalam format WAV, sementara numpy mempermudah manipulasi data numerik dalam bentuk array, seperti konversi data audio menjadi array tipe int16. Untuk enkripsi dan dekripsi, Crypto.PublicKey.RSA digunakan untuk membuat dan mengelola kunci RSA, sedangkan Crypto.Cipher.PKCS1_OAEP menyediakan metode enkripsi dan dekripsi yang aman menggunakan padding OAEP. Selain itu, Crypto.Random.get_random_bytes menghasilkan byte acak yang diperlukan untuk operasi kriptografi agar lebih aman. Kombinasi semua header ini memungkinkan proses membaca audio, memprosesnya sebagai data numerik, lalu mengenkripsi dan mendekripsi data menggunakan algoritma RSA.

```
import wave
import numpy as np
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Random import get_random_bytes
```

Gambar 5. Header

2. Fungsi read_audio

   Fungsi ini membaca file audio dalam format WAV dan mengembalikan data audio beserta parameter file. Fungsi ini membuka file dalam mode baca biner menggunakan modul wave, kemudian mengambil parameter file seperti jumlah channel, sampel per detik, dan lainnya melalui fungsi getparams(). Setelah itu, frame audio dibaca sebanyak jumlah frame yang tersedia menggunakan readframes().

```
def read_audio(file_path):
    with wave.open(file_path, 'rb') as wf:
        params = wf.getparams()
        frames = wf.readframes(params.nframes)
        audio_data = np.frombuffer(frames, dtype=np.int
    return audio_data, params
```

Gambar 6. Fungsi read_audio

3. Fungsi write_audio

   Fungsi ini digunakan untuk menulis data audio ke dalam file WAV baru. Fungsi ini membuka file dalam mode tulis biner menggunakan modul wave, lalu menetapkan parameter file (seperti jumlah channel dan sampel rate) menggunakan fungsi setparams(). Data audio dalam bentuk array NumPy dikonversi menjadi byte menggunakan tobytes() dan ditulis ke file melalui fungsi writeframes().

```
def write_audio(file_path, audio_data, params):
    with wave.open(file_path, 'wb') as wf:
        wf.setparams(params)
        wf.writeframes(audio_data.tobytes())
```

Gambar 7. Fungsi write_audio

4. Fungsi generate_rsa_keys

   Fungsi ini berfungsi untuk membuat pasangan kunci RSA, yaitu kunci privat dan kunci publik. Fungsi ini menggunakan modul Crypto.PublicKey.RSA untuk menghasilkan kunci RSA dengan panjang 2048-bit. Kunci privat diekspor dalam format byte menggunakan metode export_key(), sedangkan kunci publik diperoleh dari kunci privat melalui metode publickey().export_key().

```
def generate_rsa_keys():
    key = RSA.generate(2048)
    private_key = key.export_key()
    public_key = key.publickey().export_key()
    return private_key, public_key
```

Gambar 8. Fungsi generate_rsa_keys

5. Fungsi encrypt_audio

   Fungsi ini mengenkripsi data audio menggunakan kunci publik RSA. Fungsi ini memanfaatkan algoritma PKCS1_OAEP untuk enkripsi, yang diimplementasikan dengan membuat objek cipher dari kunci publik. Data audio dipecah menjadi potongan kecil (chunk) dengan ukuran 190 byte, sesuai batas ukuran data yang dapat dienkripsi RSA (2048-bit dikurangi padding). Setiap potongan data dikonversi menjadi byte menggunakan tobytes(), lalu dienkripsi menggunakan metode cipher.encrypt(). Hasil dari semua potongan data yang telah dienkripsi digabungkan menjadi satu dengan b"".join(), menghasilkan data terenkripsi dalam bentuk byte.

```python
def encrypt_audio(audio_data, public_key):
    cipher = PKCS1_OAEP.new(RSA.import_key(public_key))
    encrypted_chunks = []
    chunk_size = 190  # RSA 2048-bit limit for OAEP padding

    audio_data_bytes = audio_data.tobytes()

    for i in range(0, len(audio_data_bytes), chunk_size):
        chunk = audio_data_bytes[i:i + chunk_size]
        encrypted_chunk = cipher.encrypt(chunk)
        encrypted_chunks.append(encrypted_chunk)

    encrypted_data = b"".join(encrypted_chunks)
    print("\n[INFO] Data yang terenkripsi:")
    print(encrypted_data[:500], "...")  # Menampilkan sebagian (500 byte perta
    return encrypted_data
```

Gambar 9. Fungsi encrypt_audio

6. Fungsi decrypt_audio

   Fungsi ini berfungsi untuk mengenkripsi data audio yang telah terenkripsi menggunakan kunci privat RSA. Fungsi ini membuat objek cipher menggunakan kunci privat RSA dan memecah data terenkripsi menjadi potongan-potongan dengan ukuran 256 byte, sesuai panjang kunci RSA. Setiap potongan data didekripsi menggunakan metode cipher.decrypt(), dan hasilnya diubah menjadi array NumPy dengan tipe data int16 menggunakan np.frombuffer(). Potongan data yang telah didekripsi kemudian digabungkan menjadi satu array dengan np.concatenate(), sehingga menghasilkan data audio yang telah kembali ke bentuk aslinya.

```python
def decrypt_audio(encrypted_data, private_key):
    cipher = PKCS1_OAEP.new(RSA.import_key(private_key))
    chunk_size = 256  # RSA 2048-bit key size
    decrypted_chunks = []

    for i in range(0, len(encrypted_data), chunk_size):
        encrypted_chunk = encrypted_data[i:i + chunk_size]
        decrypted_chunk = cipher.decrypt(encrypted_chunk)
        decrypted_chunks.append(decrypted_chunk)

    decrypted_audio_bytes = b"".join(decrypted_chunks)
    print("\n[INFO] Data yang telah didekripsi (sebagian):")
    print(decrypted_audio_bytes[:500], "...")  # Menampilkan sebagian
    return np.frombuffer(decrypted_audio_bytes, dtype=np.int16)
```

Gambar 10. Fungsi decrypt_audio

7. Fungsi main

   Fungsi ini menjalankan keseluruhan proses enkripsi dan dekripsi data audio. Fungsi ini dimulai dengan membaca file audio input (input.wav) menggunakan read_audio. Selanjutnya, kunci RSA (privat dan publik) dihasilkan menggunakan generate_rsa_keys. Data audio kemudian dienkripsi menggunakan kunci publik dan hasilnya disimpan ke dalam file (encrypted_audio.bin). Setelah itu, data terenkripsi dibaca kembali dari file, didekripsi menggunakan kunci privat melalui fungsi decrypt_audio, dan hasilnya disimpan dalam file audio output (decrypted.wav). Fungsi ini memastikan seluruh proses berjalan secara terstruktur, mulai dari membaca file asli hingga menyimpan file hasil dekripsi.

```python
def main():
    input_audio_file = 'input.wav'
    encrypted_audio_file = 'encrypted_audio.bin'
    decrypted_audio_file = 'decrypted.wav'

    # Langkah 1: Baca file audio asli
    print("[INFO] Membaca file audio...")
    audio_data, params = read_audio(input_audio_file)

    # Langkah 2: Generate kunci RSA
    print("[INFO] Membuat kunci RSA...")
    private_key, public_key = generate_rsa_keys()

    # Langkah 3: Enkripsi data audio
    print("[INFO] Mengenkripsi data audio...")
    encrypted_data = encrypt_audio(audio_data, public_key)

    # Simpan data yang terenkripsi ke file
    with open(encrypted_audio_file, 'wb') as ef:
        ef.write(encrypted_data)
    print(f"[INFO] Data terenkripsi disimpan di: {encrypted_audio_file}")

    # Langkah 4: Dekripsi data audio
    print("[INFO] Mendekripsi data audio...")
    with open(encrypted_audio_file, 'rb') as ef:
        encrypted_data = ef.read()

    decrypted_audio = decrypt_audio(encrypted_data, private_key)

    # Langkah 5: Simpan file audio hasil dekripsi
    print("[INFO] Menyimpan file audio hasil dekripsi...")
    write_audio(decrypted_audio_file, decrypted_audio, params)
    print(f"[INFO] File audio hasil dekripsi disimpan di: {decrypted_audio_file}")

    print("[INFO] Kunci Public RSA:")
    print(public_key.decode())
    print("\n[INFO] Kunci Private RSA:")
    print(private_key.decode())

if __name__ == "__main__":
    main()
```

Gambar 11. Fungsi main

B. Pengujian Kode

```
PS C:\Users\IRGIANSYAH\OneDrive\Documents\enkripsi_suara> python rsa.py
[INFO] Membaca file audio...
[INFO] Membuat kunci RSA...
[INFO] Mengenkripsi data audio...
[INFO] Data terenkripsi disimpan di: encrypted_audio.bin
[INFO] Mendekripsi data audio...
[INFO] Menyimpan file audio hasil dekripsi...
[INFO] File audio hasil dekripsi disimpan di: decrypted.wav
```

Gambar 12. Pengujian pertama

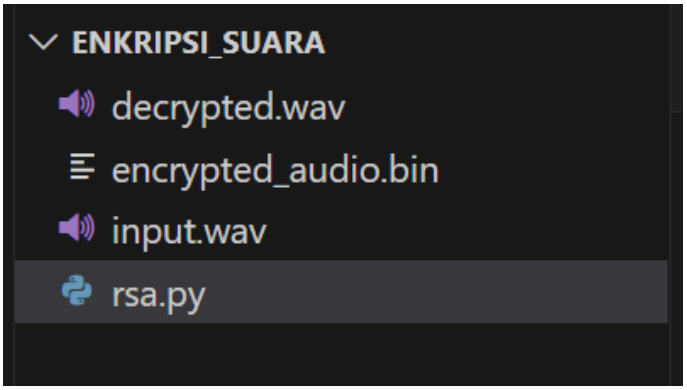
```
[INFO] Data yang terenkripsi:
b'+\xae\tGr\x96>\xa9fh\xf5|E\xa5\xe1\xdcZMl\x01\x00k\x11\xb1Gj\xd4ty6]\xe8\xcb+\xaa\xb1\
0f\xff\xc0\x9bH\xc7S\xc8v\xbf\x9f"$\x0b\xee\x0b6BA(\xf3\xf4\xf6\xa93~^\xe7\xee\xd0\xea\x
xbc\x86\xf0\xdf!\x99M\xbc\x1dH\xe3\\?\x1d\xf7\n\xb0\xbc(XFs-\x1e{\x8e0\xbe\xa4p?\x0b\x1f
D\xf7\xfd&\xf9\xdaP\xc0\x87\xa0\x954\x0e[\x8eYB\xf3\xe6\xb0\x81:i?\xdd\x86\xf9\x94]_\xaf
0\x99q#d\xbb\nN\xc8=\xbf1\x04\x04B\xd4\xe3p\x9ea\x16L\xd6\xce\x07\xf2\x1aK#\xb8\xbd\xe81
xfe\xe84\xaf\xa8\x95\x8by|\x05\x91\xc1A\xc51w#sg\x82\x1a\x07\xda\x1e\xf4^\x16^\xe7\x7f\x
2\x8cKb\xaa\x01$\x19\x06\xf9\xf2p\x99' ...
[INFO] Data terenkripsi disimpan di: encrypted_audio.bin
```

Gambar 13. Data yang terenkripsi

```
[INFO] Data yang telah didekripsi (sebagian):
b'uvvxz~\x83\x87\x88\x87\x84\x81\x80\x80\x82\x82\x82\x80|yz~\x83\x88\x89
\x8f\x8f\x8e\x8c\x8a\x88\x85\x82\x80~}|||~~~\x81\x82\x85\x86\x87\x86\x84
\x89\x8a\x88\x87\x86\x84\x83\x81\x80\x7f~}||}~}|zyxxyxxxxyz|\x80\x82\x85
rv|\x82\x86\x86\x85\x85\x86\x8b\x91\x95\x96\x92\x8a\x81{y{\x7f\x80|vsuz\
\x91\x91\x8e\x88\x81{vsrpmllllnruwz{||{|~~\x7f\x80\x80\x81\x82\x84\x85\
\x84\x82\x82\x81\x81\x81~|zxxxxxwusrrtvyz|\x80\x85\x8b\x90\x93\x92\x8f\
[INFO] Menyimpan file audio hasil dekripsi...
[INFO] File audio hasil dekripsi disimpan di: decrypted.wav
```

Gambar 14. Data yang didekripsi

Gambar 15. Kunci publik



Gambar 16. Hasil enkripsi tersimpan di file decrypted.wav

Gambar 17. Kunci privat

Pada pengujian pertama, compiler berhasil dan input.wav yang dimasukan memenuhi format. Setelah itu kode melakukan membuat kunci dan mengenkripsi data yang hasilnya disimpan file baru bernama decrypted.wav.

C. Analisis hasil

Berdasarkan hasil dari pengujian, program telah berhasil mengenkripsi dan mendekripsi data audio menggunakan RSA dengan panjang kunci 2048-bit. Hasil log menunjukkan bahwa proses enkripsi menghasilkan data terenkripsi dalam bentuk biner yang ditampilkan sebagian, dan data dekripsi juga berhasil diubah kembali ke bentuk semula (dengan tampilan sebagian). Kunci publik dan privat RSA juga tercetak dengan jelas dalam format PEM, yang sesuai untuk digunakan kembali. Hal ini menunjukkan bahwa implementasi RSA bekerja dengan baik untuk data chunk kecil yang sesuai dengan batas ukuran RSA (190 byte per chunk untuk PKCS1_OAEP).

IV. KESIMPULAN

Perkembangan teknologi yang pesat membawa tantangan dalam pengamanan data, terutama dalam menjaga privasi dan kerahasiaan informasi. Penerapan algoritma kriptografi pada data audio menjadi solusi untuk menjamin keamanan komunikasi, termasuk dalam pesan suara. Penulis memilih algoritma RSA untuk mengenkripsi dan mendekripsi data suara yang diolah menggunakan Pulse Code Modulation (PCM). Penerapan algoritma RSA menunjukkan bagaimana prinsip Matematika Diskrit, khususnya teori bilangan dan aritmetika modular, dapat diterapkan pada bidang keamanan data. RSA menjadi pilihan algoritma kriptografi yang kuat dalam melindungi data suara dari akses tidak sah, sekaligus memberikan dasar yang kokoh untuk penerapan lebih lanjut pada aplikasi yang membutuhkan transmisi data audio yang aman dan terpercaya.

Dengan menerapkan algoritma RSA untuk mengenkripsi dan mendekripsi pesan suara yang diolah menggunakan Pulse Code Modulation (PCM) dan kunci RSA 2048-bit, data audio dapat dienkripsi menjadi data terenkripsi yang aman dan dikembalikan ke bentuk semula melalui proses dekripsi. Pendekatan ini membuktikan efektivitas algoritma RSA dalam memastikan kerahasiaan data suara. Namun, RSA lebih optimal untuk mengenkripsi data kecil seperti kunci simetris. Untuk meningkatkan efisiensi pada pengolahan pesan suara berukuran besar, kombinasi RSA dan algoritma simetris seperti AES dapat digunakan, di mana RSA mengenkripsi kunci AES, sementara AES menangani enkripsi data audio. Hasil ini memberikan solusi dasar untuk melindungi data suara dari akses tidak sah, namun implementasi di lingkungan nyata memerlukan pengelolaan kunci yang aman dan optimisasi algoritma untuk performa yang lebih baik.

V. UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya, sehingga penulis dapat menyelesaikan makalah yang berjudul “Enkripsi RSA pada Pesan Suara Menggunakan Pulse Code Modulation (PCM)” tepat pada waktunya. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., selaku dosen pengampu mata kuliah Matematika Diskrit Kelas 1, atas bimbingan, arahan, dan pengajaran yang diberikan selama proses pembelajaran. Terakhir, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada orang tua, keluarga, dan teman-teman, serta seluruh pihak yang telah memberikan

dukungan moral maupun material sehingga makalah ini dapat terselesaikan dengan baik.

REFERENSI

[1] Rinaldi Munir, "Teori Bilangan Bagian 3" , https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/17-Teori-Bilangan-Bagian3-2024.pdf, diakses pada 5 January pukul 09:00.

[2] Tutorialspoint Team https://www.tutorialspoint.com/digital_communication/digital_communication_pulse_code_modulation.htm , diakses pada 5 January pukul 09.00.

[3] Martin Plonus. (2020). Electronics and Communications for Scientists and Engineers (Second Edition), Chapter 9.

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, 8 January 2024

Irgiansyah Mondo - 13521167