Implementasi Tanda Tangan Digital dengan Algoritma RSA dan Steganografi LSB pada *File Audio*

Tubagus Baraka Kautsar Sofiuddin 18220095 (*Author*)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): barakakautsar@gmail.com

***Abstract*— Dengan berkembangnya teknologi AI, kita dapat dengan mudah membuat rekaman suara kemudian membuat suara pada rekaman tersebut menjadi serupa dengan suara orang lain. Hal ini tentu saja berpotensi untuk dieksploitasi seperti untuk serangan pemalsuan identitas. Oleh karena itu, diusulkan sebuah solusi untuk menjaga integritas dan memeriksa autentikasi dari sebuah rekaman suara menggunakan tanda tangan digital berdasarkan algoritma RSA dan fungsi hash SHA-256. Untuk memudahkan pengecekan tanda-tangan steganografi LSB dapat digunakan untuk memasukannya langung ke file utama.**

Keywords—tanda-tangan digital, algoritma RSA, hash SHA-256, steganografi LSB.

I. PENDAHULUAN

Rekaman suara telah menjadi bukti penting dalam berbagai konteks, termasuk peradilan, investigasi, dan komunikasi yang kritis. Namun, dengan kemajuan teknologi kecerdasan buatan (AI), semakin mudah untuk memalsukan rekaman suara dengan kualitas yang sangat realistis. Hal ini menimbulkan ancaman serius terhadap keaslian dan integritas rekaman suara yang digunakan sebagai bukti atau sumber informasi.

Dalam upaya untuk melawan pemalsuan rekaman suara yang semakin canggih, tanda tangan digital telah digunakan sebagai solusi untuk memverifikasi keaslian dan integritas data digital, termasuk file audio. Algoritma RSA, sebagai salah satu algoritma kriptografi yang kuat, telah digunakan secara luas dalam implementasi tanda tangan digital. Namun, meskipun tanda tangan digital dapat memberikan tingkat keamanan yang tinggi, seringkali sulit untuk mengidentifikasi apakah rekaman suara telah dipalsukan atau tidak.
Dalam konteks ini, kami mengusulkan penggunaan kombinasi algoritma RSA dan steganografi LSB sebagai metode untuk memperkuat tanda tangan digital pada *file audio*. Algoritma RSA digunakan untuk menghasilkan tanda tangan digital yang unik dan autentik, sementara steganografi LSB digunakan untuk menyembunyikan tanda tangan digital langsung pada file tersebut.

II. DASAR TEORI

A. File Audio

File audio merupakan representasi digital dari suara atau musik yang dapat disimpan dan diputar pada perangkat elektronik. Setiap file audio memiliki parameter-parameter yang mendefinisikan karakteristiknya, seperti format file, ukuran file, tingkat sampel, dan resolusi bit. Pemahaman tentang parameter-parameter ini penting dalam memahami perbedaan antara format file audio yang berbeda.

Dua format file audio yang umum digunakan adalah WAV (*Waveform Audio File Format*) dan MP3 (MPEG-1 *Audio Layer* 3). Perbedaan mendasar antara kedua format ini terletak pada kompresi data yang digunakan. Format file WAV adalah format audio mentah yang umum digunakan dalam produksi musik dan rekaman. File WAV menyimpan data audio dalam bentuk tak terkompresi (*lossless*), yang berarti bahwa tidak ada data yang hilang saat proses penyimpanan. Format ini menghasilkan kualitas audio yang sangat tinggi, tetapi juga menghasilkan ukuran file yang besar. Parameter-parameter yang tersedia dalam format file WAV mencakup tingkat sampel (*sample rate*), resolusi bit (*bit depth*), dan jumlah saluran (*channel*).

Dalam maklah ini, kami akan melihat lebih dalam mengenai penerapan tanda tangan digital pada kedua format file WAV terhadap keaslian dan integritas file audio tersebut.

B. Tanda Tangan Digital

Tanda tangan digital adalah metode kriptografi yang digunakan untuk memverifikasi keaslian dan integritas data digital. Dalam konteks kriptografi, tanda tangan digital berfungsi seperti tanda tangan konvensional yang digunakan untuk mengautentikasi dokumen fisik atau surat. Namun,

tanda tangan digital memanfaatkan prinsip-prinsip kriptografi untuk memberikan tingkat keamanan yang lebih tinggi.

Prinsip dasar dari tanda tangan digital melibatkan penggunaan sepasang kunci kriptografi, yaitu kunci privat (*private key*) dan kunci publik (*public key*). Kunci privat hanya diketahui oleh pemiliknya dan harus tetap rahasia. Kunci publik, di sisi lain, dapat didistribusikan secara bebas kepada pihak lain.

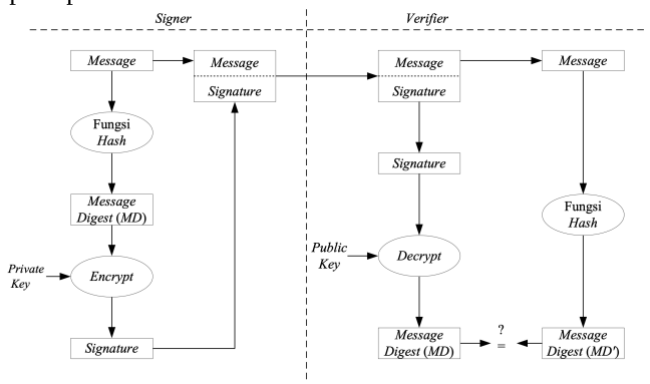

Fig. 1. Diagram tahapan tanda tangan digital (Sumber: Slide Kuliah II4031 Kriptografi dan Koding: Tanda-tangan digital oleh Rinaldi Munir)

C. Algoritma RSA

Algoritma RSA (Rivest-Shamir-Adleman) adalah salah satu algoritma kriptografi yang banyak digunakan dalam implementasi tanda tangan digital. Algoritma ini didasarkan pada penggunaan sepasang kunci kriptografi, yaitu kunci privat dan kunci publik, serta prinsip matematika yang melibatkan operasi aritmetika modular.

Algoritma RSA didasarkan pada teori bilangan dan operasi aritmetika modular. Keamanan algoritma RSA bergantung pada faktor-faktor seperti kompleksitas perhitungan faktorisasi bilangan besar dan penggunaan kunci yang cukup panjang. Dalam konteks tanda tangan digital, kekuatan algoritma RSA terletak pada kunci privat yang hanya diketahui oleh pemilik tanda tangan. Kunci privat ini digunakan untuk membuat tanda tangan yang unik, sehingga hanya pemilik yang sah yang dapat menghasilkan tanda tangan yang valid.

D. Hashing

Hashing adalah proses yang digunakan dalam kriptografi untuk mengubah data input menjadi nilai hash yang unik dan tetap panjang. Nilai hash merupakan representasi numerik tetap dari data input yang memiliki ukuran yang tetap, terlepas dari ukuran data asli. Salah satu fungsi hash yang banyak digunakan adalah SHA-256.

SHA-256 (Secure Hash Algorithm 256-bit) adalah salah satu algoritma hashing yang termasuk dalam keluarga fungsi hash Secure Hash Algorithm (SHA). SHA-256 menggunakan panjang nilai hash sebesar 256-bit, yang menghasilkan nilai hash yang terdiri dari 64 karakter heksadesimal. Algoritma ini dikembangkan untuk menghasilkan nilai hash yang sangat unik dan resisten terhadap perubahan data yang kecil.

Fungsi hash SHA-256 memiliki berbagai aplikasi, termasuk dalam keamanan informasi, tanda tangan digital, dan verifikasi integritas data. Dalam konteks tanda tangan digital, SHA-256 digunakan untuk menghasilkan nilai hash dari data yang ditandatangani, yang kemudian dienkripsi dengan kunci privat untuk membentuk tanda tangan digital. Nilai hash SHA-256 juga digunakan dalam verifikasi tanda tangan, di mana data yang ditandatangani akan di-hash ulang menggunakan SHA-256, dan nilai hash tersebut dibandingkan dengan nilai hash asli yang terkait dengan tanda tangan digital.

E. Steganografi

Steganografi adalah sebuah teknik yang digunakan untuk menyembunyikan pesan atau informasi rahasia dalam media yang tampaknya tidak berhubungan, seperti gambar, audio, atau video. Salah satu teknik steganografi yang umum digunakan adalah steganografi LSB (Least Significant Bit).

Pada steganografi LSB, pesan atau informasi rahasia disisipkan ke dalam *media cover* dengan memanfaatkan bit terkecil (*least significant bit*) dari setiap piksel atau sampel dalam media tersebut. Setiap piksel atau sampel umumnya direpresentasikan oleh beberapa bit dalam format digital, seperti RGB untuk gambar atau PCM untuk audio. Dengan mengganti bit terkecil dari piksel atau sampel dengan bit pesan, informasi rahasia dapat disisipkan tanpa mengubah secara signifikan tampilan visual atau kualitas *media cover*.

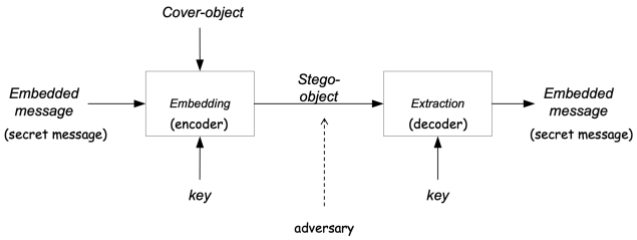

Fig. 2. Diagram Proses Steganografi (Sumber: Slide Kuliah II4031 Kriptografi dan Koding: Steganografi oleh Rinaldi Munir)

III. Rancangan Solusi dan Implementasi

Berdasarkan teori-teori yang telah dibahas pada bab sebelumnya, berikut merupakan rancangan dan implementasi dari solusi yang diusulkan.

A. Deskripsi Umum Solusi

Sebuah tanda tangan digital dapat memastikan otentikasi dan integritas dari sebuah rekaman suara. Namun, Tidak seperti file teks yang tanda tangan dapat langsung dimasukan pada file asli, seringkali penyimpanan tanda tangan untuk file suara disimpan di file terpisah. Untuk memastikan tanda tangan

selalu bersama dengan rekaman dan memudahkan proses verifikasi, tanda tangan dapat dimasukan kedalam rekaman tersebut menggunakan LSB. Untuk memastikan proses pemasukan tanda tangan tidak mempengaruhi hasil hash yang dibutuhkan dalam algoritma RSA, tanda tangan akan dibuat menggunakan hasil hash rekaman dengan *least-significant-bit* di-*set* ke nilai 0.

Dengan solusi ini seseorang dapat memasukan tanda tangannya kedalam *file audio* hanya dengan kunci privatnya. Kemudian untuk melakukan verifikasi keaslian dari *file*, seseorang hanya membutuhkan kunci publik dari penanda tangan.

B. Rancangan Solusi

Solusi dapat didekomposisi menjadi 2 tahap atau bagian. Bagian pertama adalah untuk membuat dan memasukan tanda tangan kedalam *file*. Berikut merupakan Langkah-langkah yang dilakukan untuk bagian tersebut:

- Apabila belum punya, pengguna membuat pasangan kunci public dan privatnya.
- Pengguna memasukan *file audio* yang ingin ditanda-tangan beserta dengan kunci privatnya.
- Program akan mengubah LSB dari setiap *frame file audio* menjadi 0.
- Algortima SHA-256 digunakan untuk menghasilkan *hash digest* dari hasil modifikasi.
- Algoritma RSA digunakan untuk menekripsi pesan menggunakan kunci privat dan menghasilkan tanda tangan digital berbentuk *bytes*.
- Tanda tangan digital dikonversi menjadi biner dan dihitung panjang nya.
- Informasi panjang tanda tangan dimasukan sebelum tanda tangan digital menghasilkan *binary message* untuk dimasukan ke file.
- Dilakukan pengecekan untuk memastikan jumlah frame cukup untuk dimasukan *binary message*.
- LSB dari setiap *frame* kemudian diubah menjadi setiap *bit* pada *binary message*.
- *Frames* yang sudah dimodifikasi kemudian dibuat kembali menjadi *file* .wav

Berikut merupakan diagram tahapan memasukan tanda tangan digital kedalam *file audio*.

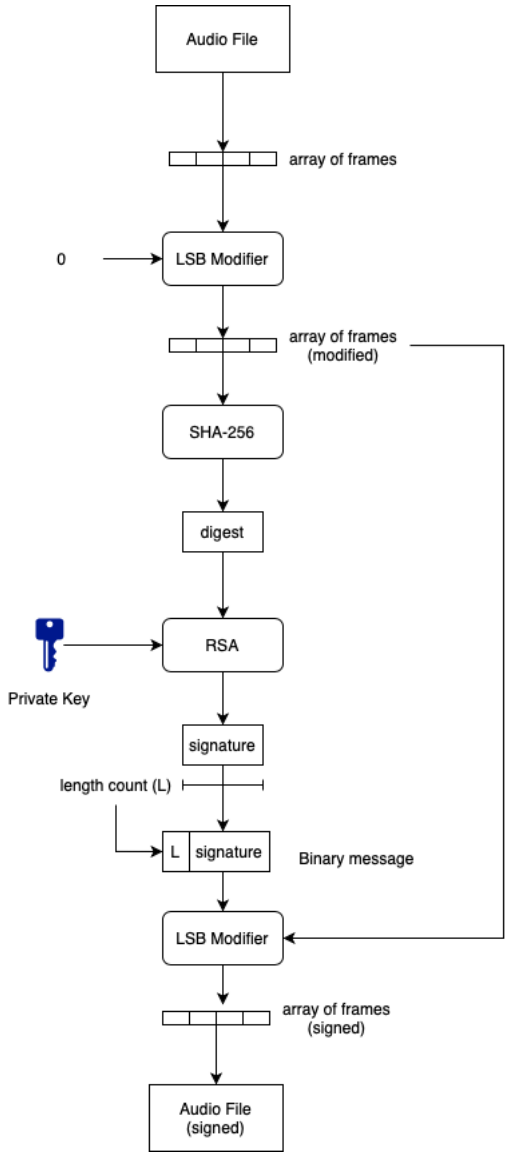

Fig. 3. Diagram tahapan memasukan tanda tangan digital kedalam *file audio*

Bagian kedua dari sistem solusi merupakan sistem untuk melakukan verifikasi tanda tangan digital yang telah dimasukan ke *file audio*. Berikut merupakan Langkah-langkah dari proses tersebut.

- Pengguna akan memasukan kunci publik milik penandatangan beserta *file audio* yang telah ditanda tangan.
- Program akan membaca LSB 16 frame pertama untuk menentukan panjang *signature.*
- Program kemudian mendapatkan *signature* setelah membaca LSB sesuai panjang yang telah didapatkan sebelumnya.

- Setiap frame audio kemudian dimodifikasi LSB nya menjadi 0 kemudian dilakukan hash dengan SHA-256
- Hasil hash kemudian diverifikasi dengan *signature* yang telah didapatkan.

Berikut merupakan diagram tahapan melakukan verifikasi file audio yang telah di tanda tangan.

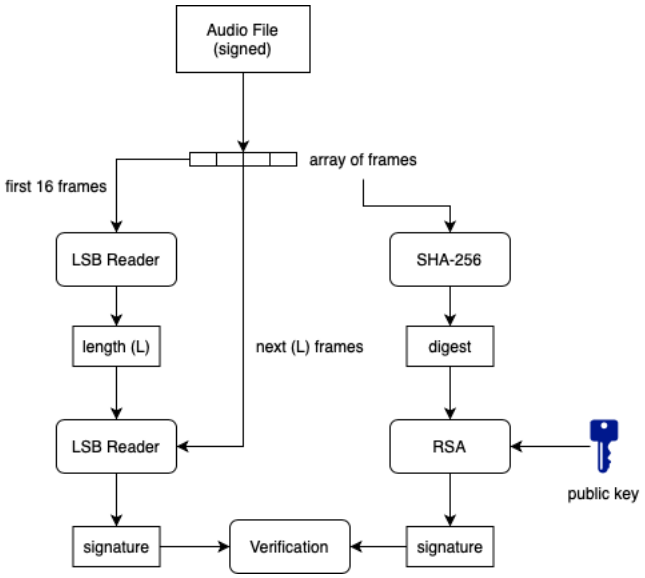

Fig. 4. Diagram melakukan verifikasi file audio yang telah di tanda tangan.

C. *Implementasi*

Berikut merupakan implementasi rancangan solusi subab sebelumnya yang dibuat menggunakan bahasa pemrograman python.

1) *Pembangkitan Kunci dan Enkripsi*

```
import rsa


def generateKeys():
    (pubkey, privkey) = rsa.newkeys(512)
    with open('keys/publicKey.pem', 'wb') as p:
        p.write(pubkey.save_pkcs1('PEM'))
    with open('keys/privateKey.pem', 'wb') as p:
        p.write(privkey.save_pkcs1('PEM'))
    return pubkey, privkey



def loadKeys():
    with open('keys/publicKey.pem', 'rb') as p:
        publicKey = rsa.PublicKey.load_pkcs1(p.read())
    with open('keys/privateKey.pem', 'rb') as p:
        privateKey = rsa.PrivateKey.load_pkcs1(p.read())
    return publicKey, privateKey


def sign(message, privateKey):
    return rsa.sign(message, privateKey, 'SHA-256')


def verify(message, signature, publicKey):
    verified = rsa.verify(message, signature, publicKey)
    if verified == 'SHA-256':
        return True
    else:
        return False
```

2) *Fungsi Hash untuk Array*

```
import hashlib
import struct


def calculate_array_hash(array):
    sha256_hash = hashlib.sha256()

    # Convert the array of short values to bytes in chunks
    for i in range(0, len(array), 4096):
        chunk = array[i:i+4096]
        byte_data = bytearray()
        for value in chunk:
            byte_data.extend(struct.pack('H', value))

        sha256_hash.update(byte_data)

    array_hash = sha256_hash.digest()

    return array_hash
```

3) *Tanda tangan File Audio*

```
def sign_audio(audio_file, key):
    print("\n ///SIGNED AUDIO EMBEDDING/// \n")
    print("opening audio file...")
    audio = wave.open(audio_file, 'rb')

    # Read the audio file parameters
```

```
    num_channels = audio.getnchannels()
    sample_width = audio.getsampwidth()
    frame_rate = audio.getframerate()
    num_frames = audio.getnframes()
    total_samples = num_frames * num_channels

    # Read the audio samples & hashing modified samples
    samples = array.array('h', audio.readframes(num_frames))
    modified_LSB_samples = [set_lsb_to_zero(sample) for sample in
samples]
    print("hashing...")
    hash_value =
hashbrown.calculate_array_hash(modified_LSB_samples)

    print("creating signature...")
    signature = enkrisisturrr.sign(hash_value, key)

    # Convert the signature bytes into binary & insert the length of the
signature at the beginning
    binary_signature = bin(int.from_bytes(signature, byteorder='big'))[2:]
    lenght = format(len(binary_signature),"016b")
    binary_message = lenght + binary_signature

    # Ensure the audio file is long enough to hold the secret message
    if len(binary_message) > total_samples:
        raise ValueError("Secret message is too long to embed in the
audio file.")

    # Embed the secret message into the audio samples
    for i in range(len(binary_message)):
        # Get the current sample
        sample = samples[i]
        modified_sample = sample & 0xFFFE
        modified_sample |= int(binary_message[i])

        samples[i] = modified_sample

    output_file = audio_file[:-4] + "(signed).wav"
    output = wave.open(output_file, 'wb')


    output.setnchannels(num_channels)
    output.setsampwidth(sample_width)
    output.setframerate(frame_rate)

    # Write the modified samples to the output file
    output.writeframes(samples.tobytes())
    audio.close()
    output.close()

    print("Audio signed successfully!")
    print("signed audio saved at:",output_file)
```

4) *Verifikasi Tanda tangan File Audio*

```
def verify_audio(audio_file, key):
    print("\n ///SIGNED AUDIO VERIFICATION/// \n")

    # Open the audio file
    print("opening audio file...")
    audio = wave.open(audio_file, 'rb')

    # Read the audio file parameters
    num_channels = audio.getnchannels()
    num_frames = audio.getnframes()

    # Calculate the total number of samples in the audio file
    total_samples = num_frames * num_channels

    # Read the audio samples
    samples = array.array('h', audio.readframes(num_frames))

    # Extract signature length
    bitlength = ''
    for i in range(0, 16):
        sample = samples[i]
        bitlength += str(sample & 1)
    length = int(bitlength,2)

    # Extract the signature from the audio samples
    print("extracting signature...")
    binary_signature = ''
    for i in range(16, length+16):
        sample = samples[i]
        binary_signature += str(sample & 1)
```

```
    signature = int(binary_signature,2).to_bytes((len(binary_signature) +
7) // 8, byteorder='big')

    # Hash audio file
    print("hashing...")
    modified_LSB_samples = [set_lsb_to_zero(sample) for sample in
samples]
    hash_value =
hashbrown.calculate_array_hash(modified_LSB_samples)

    if (enkrisisturrr.verify(hash_value, signature, key)):
        print("Signature verified!")
    else:
        print("Signature not verified!")
```

IV. Pengujian

A. *Pengujian Steganografi*

Salah satu aspek penting dari steganografi adalah pemasukan informasi tambahan tidak boleh mengubah konten asli secara signifikan. Berikut merupakan perbandingan *file audio* sebelum dan sesudah di tanda tangan.

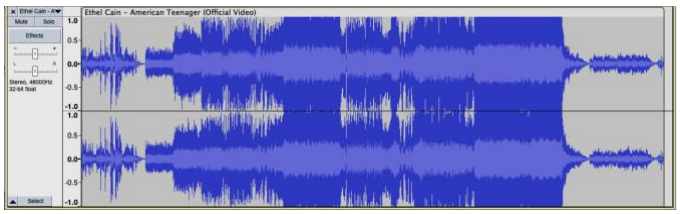

Fig. 5. *File audio sebelum tanda tangan*

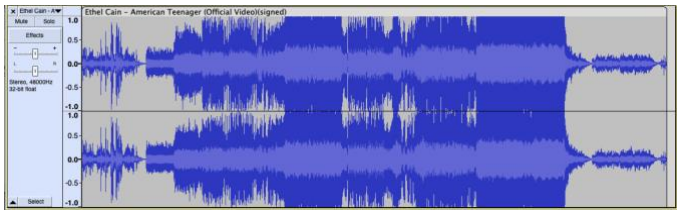

Fig. 6. *File audio setelah tanda tangan*

B. *Pengujian Tanda Tangan*

Perlu dipastikan juga bahwa tanda tangan dapat diekstraksi dan diverifikasi apabila file benar dan apabila fike telah di ubah. Berikut merupakan pengujian verifikasi tanda tangan menggunakan pada beberapa kasus.

TABLE I. Pengujian Tanda Tangan

No.	Deskripsi Kasus	Output
1.	Menggunakan file original dan kunci sesuai	Signature verified!
2.	Menggunakan file diubah dengan kunci sesuai	Signature not verified!
3.	Menggunakan file original dan kunci tidak sesuai	Signature not verified!

V. Kesimpulan dan Saran

A. *Kesimpulan*

Berdasarkan pengujian yang telah dilakukan dapat disimpulkan bahwa solusi yang diimplementasikan dapat menjamin *authentication*, *integrity*, dan *non-repudation* dari sebuah *file audio*. Dapat dilihat *authentication* dan *non-repudation* didapatkan dari implementasi tanda tangan digital menggunakan algoritma RSA dikarenakan adanya kunci public dan kunci privat yang hanya dimiliki sang penandatangan. Sedangkan *non-repudation* didapatkan dari implementasi fungsi hash SHA-256 yang akan menghasilkan digest berbeda apabila input diubah.

Selain itu pemanfaatan steganografi LSB membuat tanda tangan dapat langsung dimasukan ke file utama sehingga meningkatkan kemudahan. Namun hal ini menyebabkan beberapa batasan dari pengiriman file audio karena apabila dikompresi secara lossy memungkinkan tanda tangan digital menjadi tidak dapat dibaca.

B. *Saran*

Berikut merupakan beberapa saran untuk implementasi solusi serupa kedepannya.

- Membuat program dapat menerima *file* audio selain .wav seperti .mp3
- Menambah GUI untuk memudahkan penggunaan program

Link Repository Github

https://github.com/BarakaKautsar/Kripto_Makalah

Referensi

[1] Munir, Rinaldi. 2023. Slide Kuliah II4031 Kriptografi dan Koding: Steganografi

[2] Munir, Rinaldi. 2023. Slide Kuliah II4031 Kriptografi dan Koding: Kriptografi Kunci-Publik (Public-key Cryptography)

[3] Munir, Rinaldi. 2023. Slide Kuliah II4031 Kriptografi dan Koding: Algoritma RSA

[4] Munir, Rinaldi. 2023. Slide Kuliah II4031 Kriptografi dan Koding: Fungsi hash

[5] Munir, Rinaldi. 2023. Slide Kuliah II4031 Kriptografi dan Koding: Tanda-tangan digital (digital signature)

[6] Laurentia, Nadya, “Implementasi Algoritma Kunci Publik RSA, Fungsi Hash Keccak, dan Steganografi untuk Memberikan Tanda Tangan Digital pada Hasil Tes COVID- ”, Institut Teknologi Bandung, Mei 2022

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

Tubagus Baraka Kautsar Sofiuddin
18220095