Implementasi Tanda Tangan Digital pada Karya Ilustrasi Grafis

Panegak Raya Pasaribu - 18219029
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 18219029@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas tentang aplikasi Image Digital Signing yang menggabungkan steganografi dan tanda tangan digital untuk menjaga keaslian dan integritas karya ilustrasi digital. Melalui implementasi aplikasi ini, berhasil dibuktikan bahwa steganografi dapat efektif dalam menjaga kualitas karya ilustrasi digital, sedangkan validasi tanda tangan digital dengan kunci publik memungkinkan publik untuk memverifikasi keaslian dan pembuat karya tersebut. Selain itu, kunci publik dapat disimpan di media sosial ilustrator untuk diakses. Antarmuka pengguna yang sederhana membuat pengguna dapat dengan mudah mengakses fitur-fitur aplikasi ini. Namun, terdapat keterbatasan dalam akses tanda tangan digital yang kurang terbuka dan mudah diakses, serta penyimpanan kunci publik yang membutuhkan lebih banyak sumber daya untuk menjaga hak cipta. Secara keseluruhan, eksperimen ini menunjukkan potensi besar dan dapat dikembangkan menjadi sistem yang lebih luas.**

Keywords—tanda tangan digital, steganografi, hak cipta karya ilustrasi digital

I. PENDAHULUAN

Dalam era digital yang semakin berkembang, perlindungan hak cipta terhadap karya ilustrasi digital menjadi sangat penting. Ilustrator kecil seringkali menghadapi tantangan dalam menjaga keaslian dan integritas karya mereka di tengah penyebaran yang luas melalui media sosial. Hak cipta karya ilustrasi digital merupakan aspek penting yang harus dilindungi dalam dunia digital. Karya ilustrasi digital merupakan hasil dari kreativitas dan usaha ilustrator kecil yang mencurahkan waktu dan energi dalam menciptakan karya-karya unik. Dengan adanya hak cipta yang diakui dan dilindungi secara hukum, ilustrator kecil dapat merasa aman dan dihargai atas karya mereka. Perlindungan hak cipta karya ilustrasi digital memastikan bahwa ilustrator kecil dapat mengendalikan penggunaan, reproduksi, dan distribusi karya mereka, mencegah penyalahgunaan, dan memperoleh manfaat yang pantas dari hasil karyanya.

Dalam konteks ini, kriptografi dapat memberikan solusi dalam melindungi hak cipta karya ilustrasi digital dengan menggunakan teknik steganografi dan tanda tangan digital, sehingga memberikan kepercayaan, keaslian, dan integritas terhadap karya-karya tersebut. Oleh karena itu, makalah ini bertujuan untuk melakukan eksperimen kriptografi guna mencapai tujuan utama yaitu perlindungan hak cipta karya ilustrasi digital.

Makalah ini akan menjelaskan langkah-langkah eksperimen yang dilakukan untuk mengimplementasikan teknik kriptografi, khususnya steganografi dan tanda tangan digital, dalam konteks perlindungan hak cipta karya ilustrasi grafis. Eksperimen ini diharapkan dapat memberikan pemahaman yang lebih baik tentang cara-cara melindungi keaslian karya ilustrasi dan mencegah pelanggaran hak cipta yang sering terjadi di dunia digital.

Dalam eksperimen ini, algoritma SHA-256 dan algoritma RSA akan digunakan sebagai dasar kriptografi. Algoritma SHA-256 akan digunakan untuk menghasilkan hash unik dari tanda tangan digital, sementara algoritma RSA akan digunakan untuk enkripsi dan dekripsi kunci publik dan pribadi. Implementasi teknik steganografi akan dilakukan untuk menyematkan tanda tangan digital secara tersembunyi pada gambar ilustrasi dalam format PNG.

Eksperimen ini akan melibatkan langkah-langkah seperti pembuatan sistem, pengujian, dan analisis hasil. Melalui eksperimen ini, diharapkan dapat dikembangkan metode atau sistem yang efektif dalam melindungi hak cipta karya ilustrasi digital dan memastikan keaslian serta integritas karya tersebut.

Makalah ini akan menguraikan eksperimen yang dilakukan, termasuk langkah-langkah implementasi, data yang digunakan, serta hasil dan analisis eksperimen tersebut. Hasil eksperimen ini dapat menjadi landasan untuk pengembangan lebih lanjut guna mencapai tujuan utama, yaitu perlindungan hak cipta karya ilustrasi digital di tengah perkembangan teknologi dan penyebaran yang luas melalui media sosial.

II. DASAR TEORI

A. Hak Cipta Karya Ilustrasi Digital

Hak cipta karya ilustrasi digital merupakan bagian penting dari hukum kekayaan intelektual yang memberikan perlindungan hukum terhadap karya-karya tersebut. Hak cipta melindungi ekspresi asli dan kreatifitas dari ilustrator kecil, memberikan mereka kontrol eksklusif atas penggunaan dan distribusi karya mereka. Dalam konteks ilustrasi digital, hak cipta meliputi berbagai aspek penting, seperti hak reproduksi,

hak distribusi, hak modifikasi, dan hak pemanfaatan komersial.

Hak reproduksi mengacu pada hak ilustrator untuk membuat salinan atau reproduksi dari karya ilustrasi digital mereka. Hal ini termasuk pembuatan salinan fisik maupun salinan digital dari karya tersebut. Hak distribusi berkaitan dengan hak ilustrator untuk mengontrol distribusi fisik atau digital dari karya ilustrasi mereka. Ini meliputi penjualan, pengiriman, dan penyebaran karya ilustrasi kepada pihak lain.

Hak modifikasi memberikan kekuasaan kepada ilustrator untuk membuat perubahan atau modifikasi pada karya ilustrasi digital mereka. Ini mencakup kemampuan untuk mengubah, mengadaptasi, atau memodifikasi karya tersebut. Terakhir, hak pemanfaatan komersial mengacu pada hak ilustrator untuk memperoleh manfaat komersial dari karya ilustrasi digital mereka. Ini termasuk hak untuk menghasilkan keuntungan melalui penjualan, lisensi, atau pemanfaatan karya dalam konteks bisnis.

Dengan adanya hak cipta karya ilustrasi digital, ilustrator kecil dapat melindungi karya-karya mereka dari penggunaan tanpa izin, pemalsuan, dan penyalahgunaan yang merugikan. Hak cipta memberikan perlindungan hukum yang penting untuk memastikan keberlanjutan dan penghargaan terhadap kreativitas serta usaha yang ditanamkan oleh ilustrator kecil dalam menciptakan karya ilustrasi digital yang orisinal dan unik. [1]

B. Algoritma RSA

Algoritma RSA (Rivest-Shamir-Adleman) adalah salah satu algoritma kriptografi yang paling umum digunakan dalam konteks keamanan data dan tanda tangan digital. Algoritma RSA menggunakan konsep dasar matematika teori bilangan, seperti faktorisasi bilangan besar, untuk menyediakan keamanan dalam komunikasi dan enkripsi data.

Algoritma RSA terdiri dari tiga komponen utama: pembangkitan kunci, enkripsi, dan dekripsi. Pertama, dalam tahap pembangkitan kunci, sebuah pasangan kunci yang terdiri dari kunci publik dan kunci pribadi dihasilkan. Kunci publik dapat dibagikan kepada siapa pun dan digunakan untuk enkripsi data, sementara kunci pribadi harus dijaga kerahasiaannya oleh pemiliknya dan digunakan untuk dekripsi data.

Proses enkripsi dalam algoritma RSA melibatkan penggunaan kunci publik untuk mengubah data menjadi bentuk yang tidak dapat dibaca atau diartikan. Kunci publik berfungsi sebagai mekanisme enkripsi yang memodifikasi data asli menjadi bentuk yang hanya dapat didekripsi oleh kunci pribadi yang sesuai.

Proses *decryption*, di sisi lain, melibatkan penggunaan kunci pribadi yang hanya diketahui oleh penerima pesan untuk mengembalikan data yang dienkripsi menjadi bentuk aslinya. Hanya dengan menggunakan kunci pribadi yang sesuai, data yang telah dienkripsi dapat didekripsi dan dibaca.

Keamanan algoritma RSA didasarkan pada kesulitan dalam faktorisasi bilangan bulat yang besar menjadi faktor-faktor primanya. Dalam praktiknya, memecahkan masalah faktorisasi ini menjadi tugas yang sangat sulit dan membutuhkan waktu yang lama bahkan dengan menggunakan komputer yang kuat.

Algoritma RSA telah digunakan secara luas dalam implementasi tanda tangan digital karena keunggulannya dalam keamanan data dan otentikasi. Dalam konteks implementasi tanda tangan digital pada karya ilustrasi grafis, algoritma RSA digunakan untuk mengenkripsi tanda tangan digital menggunakan kunci pribadi pencipta dan untuk melakukan dekripsi menggunakan kunci publik untuk memverifikasi keaslian karya.

Dengan demikian, algoritma RSA merupakan komponen kriptografi penting dalam implementasi tanda tangan digital pada karya ilustrasi grafis, yang memberikan keamanan, integritas, dan otentikasi terhadap karya tersebut. [2]

C. Hash

Hash merupakan salah satu komponen penting dalam kriptografi yang digunakan dalam berbagai aplikasi keamanan data, termasuk implementasi tanda tangan digital pada karya ilustrasi grafis. *Hash* adalah fungsi matematika yang mengambil input data apa pun dan menghasilkan keluaran yang unik, yang disebut sebagai hash value atau digest.

Fungsi *hash* memiliki beberapa karakteristik penting. Pertama, *hash value* yang dihasilkan harus unik untuk setiap input yang berbeda. Ini berarti bahwa setiap perubahan kecil pada input data akan menghasilkan *hash value* yang berbeda secara signifikan. Keunikan ini penting untuk memastikan integritas data, karena jika ada perubahan pada data, *hash value* yang dihasilkan akan berbeda.

Kedua, fungsi *hash* harus menghasilkan *hash value* dengan ukuran yang tetap, tidak peduli seberapa besar atau kecil ukuran input data. Artinya, *hash value* yang dihasilkan tidak tergantung pada ukuran data inputnya. Ini memungkinkan penggunaan *hash value* sebagai representasi singkat dari data asli yang lebih besar, yang berguna dalam aplikasi seperti verifikasi integritas data atau tanda tangan digital.

Selain itu, fungsi *hash* juga harus memiliki sifat "*one-way*" atau tidak dapat dikembalikan. Ini berarti bahwa tidak mungkin untuk mendapatkan data asli dari *hash value* yang dihasilkan. Dalam konteks tanda tangan digital, ini adalah aspek penting karena memungkinkan penggunaan tanda tangan digital yang terenkripsi untuk memverifikasi keaslian karya ilustrasi digital tanpa harus mengungkapkan isi sebenarnya dari karya tersebut.

Algoritma *hash* yang umum digunakan dalam implementasi tanda tangan digital pada karya ilustrasi grafis adalah *Secure Hash Algorithm* (SHA). SHA menghasilkan hash value dengan panjang tetap, seperti SHA-256 yang menghasilkan *hash value* sepanjang 256-bit. Algoritma SHA memiliki sifat keamanan yang kuat dan rentan terhadap upaya pembongkaran atau perubahan data yang tidak sah.

Dalam implementasi tanda tangan digital pada karya ilustrasi grafis, *hash value* dihasilkan dari data karya ilustrasi digital, kemudian *hash value* ini dienkripsi menggunakan kunci pribadi pencipta menggunakan algoritma RSA. Hasil

enkripsi ini adalah tanda tangan digital yang unik dan terkait langsung dengan karya ilustrasi tersebut.

Dengan demikian, hash menjadi komponen kriptografi penting dalam implementasi tanda tangan digital pada karya ilustrasi grafis, yang memberikan integritas dan keaslian terhadap karya tersebut. [3]

D. Tanda Tangan Digital

Tanda tangan digital adalah metode yang digunakan untuk memverifikasi integritas, otentikasi, dan non-repudiasi suatu dokumen digital. Dalam konteks implementasi tanda tangan digital pada karya ilustrasi grafis, tanda tangan digital memiliki peran penting dalam melindungi hak cipta karya ilustrasi digital dan mengidentifikasi penciptanya.

Tanda tangan digital menggunakan algoritma kriptografi untuk menghasilkan serangkaian data yang unik, yang disebut sebagai hash, dari karya ilustrasi digital. Hash ini kemudian dienkripsi dengan menggunakan kunci pribadi dari pencipta menggunakan algoritma kriptografi yang disebut dengan RSA (Rivest-Shamir-Adleman). Hasil enkripsi ini adalah tanda tangan digital yang terkait langsung dengan karya ilustrasi tersebut.

Tanda tangan digital tidak dapat diubah atau dimanipulasi tanpa menghasilkan hash yang berbeda, sehingga dapat digunakan untuk memverifikasi keaslian dan integritas karya ilustrasi digital. Ketika tanda tangan digital diverifikasi dengan menggunakan kunci publik yang terkait, hasil dekripsi akan menghasilkan hash yang sama dengan hash asli. Dengan demikian, penerima dapat memastikan bahwa karya ilustrasi tersebut belum diubah sejak dibuat dan memiliki otentikasi dari penciptanya.

Selain itu, tanda tangan digital juga memberikan keandalan dalam konteks non-repudiasi. Non-repudiasi berarti bahwa pencipta karya ilustrasi digital tidak dapat dengan tegas menyangkal tanggung jawab atas karya tersebut. Karena tanda tangan digital menggunakan kunci pribadi yang hanya diketahui oleh pencipta, tidak mungkin bagi orang lain untuk memalsukan atau mengklaim bahwa mereka adalah pencipta karya ilustrasi digital yang memiliki tanda tangan digital yang sah.

Dalam implementasi tanda tangan digital pada karya ilustrasi grafis, algoritma SHA-256 sering digunakan untuk menghasilkan hash yang unik dan RSA digunakan untuk enkripsi dan dekripsi kunci. Dengan memadukan steganografi dan tanda tangan digital, perlindungan hak cipta karya ilustrasi digital dapat ditingkatkan, memberikan kepercayaan dan keaslian terhadap karya tersebut. [3]

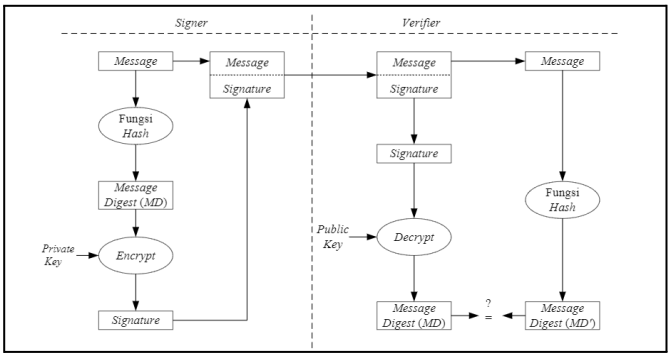

Fig. 1. Gambaran umum proses tanda tangan digital
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/)

E. Steganografi

Steganografi adalah suatu teknik yang digunakan untuk menyembunyikan pesan atau informasi rahasia dalam suatu media yang tampaknya tidak mencurigakan, seperti gambar, audio, atau teks. Dalam konteks implementasi tanda tangan digital pada karya ilustrasi grafis, steganografi memiliki peran penting dalam melindungi hak cipta karya ilustrasi digital.

Teknik steganografi pada karya ilustrasi grafis melibatkan penyisipan pesan atau data tambahan ke dalam gambar tanpa mengubah secara signifikan tampilan visual gambar tersebut. Pesan atau data tersebut dapat berupa tanda tangan digital yang mengidentifikasi pencipta karya, informasi hak cipta, atau metode lain untuk autentikasi dan verifikasi keaslian karya ilustrasi digital.

Metode steganografi yang umum digunakan dalam karya ilustrasi grafis termasuk penyisipan pesan ke dalam bit-bit yang kurang signifikan pada gambar, penggunaan teknik modifikasi bit terkecil, atau bahkan memanfaatkan perbedaan nilai piksel yang hampir tidak terlihat pada gambar. Tujuan utama steganografi dalam hal ini adalah menyembunyikan pesan secara efektif sehingga sulit untuk dideteksi oleh orang lain yang melihat gambar tersebut.

Keuntungan menggunakan steganografi dalam implementasi tanda tangan digital adalah bahwa metode ini menyediakan lapisan keamanan tambahan dengan cara menyembunyikan informasi kunci atau tanda tangan digital di dalam karya ilustrasi grafis. Dengan demikian, bahkan jika seseorang mencoba untuk memalsukan atau memodifikasi karya ilustrasi digital, tanda tangan digital yang tersembunyi akan membantu dalam memverifikasi keaslian karya tersebut.

Namun, steganografi juga memiliki beberapa tantangan. Salah satu tantangannya adalah menjaga keseimbangan antara menghasilkan pesan yang tidak terdeteksi secara visual dan kapasitas penyisipan yang cukup untuk menyimpan tanda tangan digital. Selain itu, keandalan dan ketahanan steganografi terhadap serangan atau upaya pembongkaran juga perlu dipertimbangkan dalam implementasi yang efektif. [4]

III. IMPLEMENTASI

Pada bab ini, akan dijelaskan mengenai implementasi nyata dari konsep-konsep yang telah dibahas dalam bab sebelumnya. Implementasi ini bertujuan untuk memberikan

pemahaman yang lebih konkret tentang bagaimana tanda tangan digital dapat diimplementasikan pada karya ilustrasi grafis dengan menggunakan steganografi. Bab ini akan mencakup langkah-langkah yang diperlukan untuk memasukkan tanda tangan digital ke dalam gambar ilustrasi, serta proses verifikasi keaslian karya tersebut. Selain itu, akan dibahas pula tentang algoritma-algoritma kriptografi yang digunakan, seperti algoritma SHA-256 untuk menghasilkan hash value dan algoritma RSA untuk enkripsi dan dekripsi kunci. Implementasi ini diharapkan dapat memberikan wawasan yang lebih praktis dan mendalam mengenai perlindungan hak cipta karya ilustrasi digital melalui penggunaan tanda tangan digital dan steganografi.

A. Prosedur Hashing

Prosedur hashing diimplementasikan menggunakan algoritma SHA-256 (Secure Hash Algorithm 256-bit) untuk menghasilkan hash value dari pesan yang diberikan. Algoritma SHA-256 menggunakan serangkaian fungsi dan operasi matematika yang kompleks untuk menghasilkan hash value dengan panjang tetap 256-bit.

```
#SHA-256

#const
CONST_K = [ '0x428a2f98', '0x71374491', '0xb5c0fbcf', '0xe9b5dba5', '0x3956c25b', '0x59f111f1', '0x923f82a4',
            '0xab1c5ed5', '0xd807aa98', '0x12835b01', '0x243185be', '0x550c7dc3', '0x72be5d74', '0x80deb1fe',
            '0x9bdc06a7', '0xc19bf174', '0xe49b69c1', '0xefbe4786', '0x0fc19dc6', '0x240ca1cc', '0x2de92c6f',
            '0x4a7484aa', '0x5cb0a9dc', '0x76f988da', '0x983e5152', '0xa831c66d', '0xb00327c8', '0xbf597fc7',
            '0xc6e00bf3', '0xd5a79147', '0x06ca6351', '0x14292967', '0x27b70a85', '0x2e1b2138', '0x4d2c6dfc',
            '0x53380d13', '0x650a7354', '0x766a0abb', '0x81c2c92e', '0x92722c85', '0xa2bfe8a1', '0xa81a664b',
            '0xc24b8b70', '0xc76c51a3', '0xd192e819', '0xd6990624', '0xf40e3585', '0x106aa070', '0x19a4c116',
            '0x1e376c08', '0x2748774c', '0x34b0bcb5', '0x391c0cb3', '0x4ed8aa4a', '0x5b9cca4f', '0x682e6ff3',
            '0x748f82ee', '0x78a5636f', '0x84c87814', '0x8cc70208', '0x90befffa', '0xa4506ceb', '0xbef9a3f7', '0xc67178f2']

hash_value = ['0x6a09e667', '0xbb67ae85', '0x3c6ef372', '0xa54ff53a', '0x510e527f', '0x9b05688c', '0x1f83d9ab', '0x5be0cd19']
```

Fig. 2. Inisialisasi Konstanta

Prosedur hashing dimulai dengan inisialisasi konstanta CONST_K dan nilai awal hash_value. Konstanta CONST_K merupakan rangkaian konstanta yang digunakan dalam algoritma SHA-256. Nilai awal hash_value adalah nilai hash awal yang akan berubah saat proses hashing dilakukan.

```
def translate(message):
    charcodes = [ord(c) for c in message]
    bytes = []
    for char in charcodes:
        bytes.append(bin(char)[2:].zfill(8))
    bits = []
    for byte in bytes:
        for bit in byte:
            bits.append(int(bit))
    return bits

def chunker(bits, chunk_length=8):
    chunked = []
    for b in range(0, len(bits), chunk_length):
        chunked.append(bits[b:b+chunk_length])
    return chunked
```

Fig. 3. Pemisahan chunk *message*

Selanjutnya, pesan yang diberikan diproses melalui serangkaian fungsi dan operasi matematika. Pesan diubah menjadi representasi biner menggunakan fungsi translate, lalu dipisahkan menjadi chunk-chunk dengan menggunakan fungsi preprocessMessage.

Setiap chunk pesan diproses menggunakan iterasi yang terdiri dari beberapa langkah operasi. Langkah-langkah ini melibatkan pergeseran bit, operasi XOR, operasi AND, penjumlahan bit, dan operasi matematika lainnya. Iterasi ini dilakukan sejumlah kali untuk mengubah nilai hash_value.

Terakhir, hasil hash value diubah ke dalam bentuk representasi heksadesimal dan dikembalikan sebagai output dari prosedur hashing.

Dengan menggunakan prosedur hashing ini, pesan yang diberikan dapat diubah menjadi hash value yang unik dan tidak dapat diprediksi. Hash value ini dapat digunakan untuk memverifikasi integritas pesan, keabsahan data, atau sebagai kunci untuk proses enkripsi lainnya.

B. Algoritma RSA

Pada subbab ini, penjelasan tentang algoritma RSA yang digunakan dalam implementasi enkripsi dan dekripsi pada program akan dijelaskan. Algoritma RSA (Rivest-Shamir-Adleman) adalah algoritma kriptografi asimetris yang menggunakan kunci publik dan kunci privat untuk melakukan enkripsi dan dekripsi data.

1. Pembangkitan Kunci

Pada langkah pertama, dilakukan pembangkitan kunci publik dan kunci privat. Prosedur pembangkitan kunci dilakukan sebagai berikut:

- Dua bilangan prima acak yang cukup besar, misalnya p dan q, dipilih. Dalam implementasi ini, bilangan prima tetap yaitu 47 dan 71 digunakan..
- Hitung nilai n dengan mengalikan p dan q, yaitu n = p * q. Nilai n akan menjadi modulus dalam operasi enkripsi dan dekripsi.
- Hitung nilai toitent Euler (toitent) dari n dengan rumus toitent = (p-1) * (q-1).
- Cari bilangan bulat positif e yang relatif prima dengan toitent (gcd(e, toitent) = 1). Dalam implementasi ini, bilangan e dipilih secara acak dari rentang 2 hingga toitent.
- Hitung bilangan bulat positif d yang merupakan invers modular dari e modulo toitent, yaitu d * e ≡ 1 (mod toitent). Fungsi findD pada program digunakan untuk mencari nilai d.

Setelah langkah-langkah di atas dilakukan, kunci publik (n, e) dan kunci privat (n, d) akan diperoleh.

2. Enkripsi

Langkah-langkah enkripsi pada algoritma RSA adalah sebagai berikut:

- Ubah teks yang akan dienkripsi menjadi bentuk numerik. Dalam implementasi ini, setiap karakter diubah menjadi nilai angka sesuai dengan urutan abjad (misalnya, 'a' menjadi 1, 'b' menjadi 2, dan seterusnya).
- Blokkan pesan numerik menjadi blok-blok dengan ukuran yang sesuai dengan panjang modulus n. Jika

panjang pesan tidak memenuhi ukuran blok, tambahkan angka 0 pada awal pesan hingga memenuhi ukuran blok.
- Lakukan enkripsi pada setiap blok pesan dengan menggunakan rumus enkripsi RSA: c = (m^e) % n, di mana c adalah blok pesan terenkripsi, m adalah blok pesan numerik, e adalah kunci publik, dan n adalah modulus.
- Hasil enkripsi dari setiap blok pesan digabungkan menjadi satu teks terenkripsi.

3. Dekripsi

Langkah-langkah dekripsi pada algoritma RSA adalah sebagai berikut:

- Dapatkan faktor-faktor prima dari modulus n. Dalam implementasi ini, faktor-faktor prima p dan q telah ditentukan sebelumnya yaitu 47 dan 71.
- Hitung nilai toitent Euler (toitentDecrypt) dari faktor-faktor prima dengan rumus toitentDecrypt = (pDecrypt-1) * (qDecrypt-1).
- Dapatkan nilai dDecrypt yang merupakan invers modular dari kunci publik e modulo toitentDecrypt, yaitu dDecrypt * e ≡ 1 (mod toitentDecrypt). Fungsi findD pada program digunakan untuk mencari nilai dDecrypt.
- Hitung modulus dekripsi nDecrypt dengan mengalikan faktor-faktor prima pDecrypt dan qDecrypt.
- Blokkan teks terenkripsi menjadi blok-blok dengan ukuran yang sesuai dengan panjang modulus dekripsi nDecrypt.
- Lakukan dekripsi pada setiap blok teks terenkripsi dengan menggunakan rumus dekripsi RSA: m = (c^dDecrypt) % nDecrypt, di mana m adalah blok teks terdekripsi, c adalah blok teks terenkripsi, dDecrypt adalah kunci privat, dan nDecrypt adalah modulus dekripsi.
- Hasil dekripsi dari setiap blok teks terenkripsi digabungkan menjadi satu teks terdekripsi.
- Ubah teks terdekripsi dari bentuk numerik menjadi bentuk teks sesuai dengan urutan abjad (misalnya, angka 1 menjadi 'a', angka 2 menjadi 'b', dan seterusnya).

Dengan langkah-langkah di atas, dapat dilakukan enkripsi dan dekripsi menggunakan algoritma RSA dalam implementasi program tersebut.

C. Digital Signature

Pada subbab ini, akan dijelaskan tentang implementasi tanda tangan digital menggunakan algoritma RSA dan fungsi hash SHA-256 dalam program yang telah dibuat.

Untuk implementasi tanda tangan digital, digunakan kelas Signature yang memiliki beberapa metode sebagai berikut:

- Metode __init__(self, text, key=None, mode=None): Metode ini digunakan untuk inisialisasi objek Signature. Pada metode ini, argumen text digunakan untuk menyimpan teks yang akan ditandatangani. Argumen key digunakan jika pengguna ingin mengimpor kunci publik dari file eksternal. Argumen mode digunakan untuk menentukan apakah teks akan diimpor dari file atau diinputkan secara langsung.

```
from RSA import *
from SHA256 import *

class Signature:
    def __init__(self, text, key=None, mode=None):
        #cases are if user want to import text file, or input the text itself
        if(mode == None):
            f = open(text, "r")
            self.text = f.read()
        elif(mode == "string"):
            self.text = text

        #if user import the a public key file
        if(key != None):
            f = open(key, "r")
            key = f.read()
            pubN,pubE = key.replace(" ","").split(",")
            self.n, self.e = int(pubN), int(pubE)
            f.close()
```

Fig. 4. Metode __init__

- Metode signFile(self): Metode ini digunakan untuk melakukan proses tanda tangan digital pada teks yang telah disimpan. Pada metode ini, teks dihash menggunakan fungsi hash SHA-256, kemudian hasil hash tersebut dienkripsi menggunakan algoritma RSA untuk menghasilkan tanda tangan digital. Teks asli dan tanda tangan digital digabungkan menjadi satu teks yang disimpan dalam file "signedText.txt". Metode ini mengembalikan teks yang telah ditandatangani.

```
def signFile(self):
    #apply hashing method (SHA256) to plaintext
    #encrypt hashed message using RSA to make digital signature
    signature = int(sha256(self.text), 16)
    rsa = RSA(str(signature))
    signature = rsa.encryptNumber()
    signedText = self.text + "\n\n**Digital Signature**" + signature + "**Digital Signature**"

    f_signed = open('signedText.txt', 'w')
    f_signed.write(signedText)
    f_signed.close()

    return signedText
```

Fig. 5. Metode signFile

- Metode validateSign(self): Metode ini digunakan untuk memvalidasi tanda tangan digital pada teks yang telah ditandatangani. Pada metode ini, teks dibagi menjadi dua bagian, yaitu teks asli dan tanda tangan digital. Teks asli dihash menggunakan fungsi hash SHA-256, dan tanda tangan digital didekripsi menggunakan algoritma RSA. Hasil hash dan tanda tangan digital yang telah diproses dibandingkan. Jika kedua nilai tersebut sama, maka tanda tangan digital dianggap valid, dan metode ini mengembalikan nilai True. Jika tidak, maka tanda tangan digital dianggap tidak valid, dan metode ini mengembalikan nilai False.

```python
def validateSign(self):
    #check if decrypted signature is equal with hashed message

    splittedText = self.text.split("**Digital Signature**")
    textFile = splittedText[0][:-2]
    while(textFile[-1] == '\n' or textFile[-1] == '\r'):
        textFile = textFile[:-1]
    signature = splittedText[1]

    hashedText = int(sha256(textFile), 16) % self.n
    processedSignature = (int(signature, 16) ** self.e) % self.n

    if(hashedText == processedSignature):
        return True
    else:
        return False
```

Fig. 6. Metode validateFile

Dalam implementasi ini, fungsi hash yang digunakan adalah SHA-256, yang diimplementasikan dalam modul SHA256. Kelas Signature juga menggunakan modul RSA untuk proses enkripsi dan dekripsi menggunakan algoritma RSA.

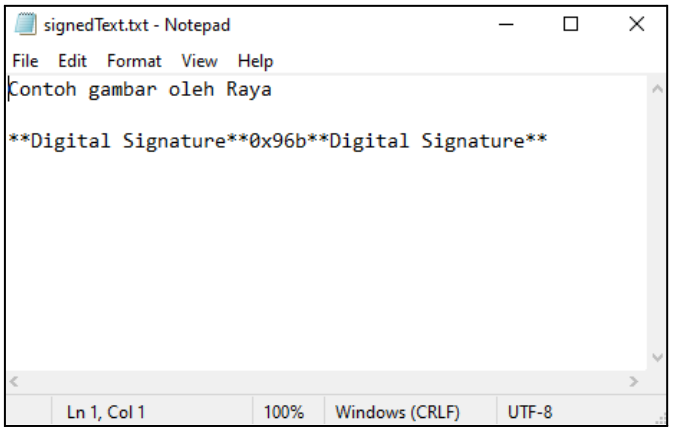

Fig. 7. Contoh Tanda Tangan Digital

Dengan menggunakan metode-metode di atas, dapat dilakukan tanda tangan digital pada teks dan tanda tangan digital tersebut dapat divalidasi dalam program yang telah dibuat.

D. Steganografi

Pada subbab ini, akan dijelaskan tentang implementasi steganografi menggunakan metode penyisipan pesan pada citra menggunakan bit terkecil (Least Significant Bit/LBS) dalam program yang telah dibuat.

Implementasi steganografi terdiri dari dua metode, yaitu Encode dan Decode, yang digunakan untuk menyisipkan pesan pada citra dan mengambil pesan tersembunyi dari citra, secara berturut-turut.

- Metode Encode(src, msg, dest): Metode ini digunakan untuk menyisipkan pesan ke dalam citra. Pada metode ini, argumen src digunakan untuk menyimpan path citra sumber yang akan digunakan sebagai media penyisipan pesan. Argumen msg berisi pesan yang akan disisipkan ke dalam citra. Argumen dest digunakan untuk menyimpan path citra hasil penyisipan pesan. Pada metode ini, citra sumber di-load menggunakan modul PIL dan dikonversi menjadi matriks array menggunakan modul numpy. Selanjutnya, pesan yang akan disisipkan diubah menjadi representasi biner menggunakan fungsi ConvertMsgToBinary. Jumlah piksel yang diperlukan untuk menyimpan pesan dihitung, dan jika pesan terlalu panjang untuk disisipkan dalam citra, maka pesan tidak dapat disisipkan. Jika pesan dapat disisipkan, maka setiap bit pesan akan disisipkan dalam bit terakhir (LSB) dari matriks array citra pada setiap saluran warna (R/G/B/A). Citra hasil penyisipan pesan disimpan dengan path yang telah ditentukan.

```python
def Encode(src, msg, dest):
    #load image
    img = Image.open(src, 'r')
    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4

    width, height = img.size
    img_arr = np.array(list(img.getdata()))
    total_pixels = img_arr.size//n

    #set end of line for hidden message
    #convert hidden message to binary
    msg += "3Offf"
    b_message = ConvertMsgToBinary(msg)
    req_pixels = len(b_message)

    #check if total length of hidden message exceeds total pixel of image file given
    if req_pixels > total_pixels:
        print("ERROR: Need larger file size")

    #insert hidden message binary into image file per pixel -> per channel(R/G/B/A)
    else:
        index=0
        for p in range(total_pixels):
            for q in range(0, 3):
                if index < req_pixels:
                    print('----------------')
                    print(b_message[index])
                    print(str(IntToBinary(img_arr[p][q])))
                    temp = list(str(IntToBinary(img_arr[p][q])))
                    temp[7] = b_message[index]
                    res = ''.join(temp)
                    img_arr[p][q] = int(res, 2)
                    print(str(IntToBinary(img_arr[p][q])))
                    print('----------------')
                    index += 1

        img_arr=img_arr.reshape(height, width, n)
        enc_img = Image.fromarray(img_arr.astype('uint8'), img.mode)
        enc_img.save(dest)
        print("Image Encoded Successfully")
```

Fig. 8. Metode Encode

- Metode Decode(src): Metode ini digunakan untuk mengambil pesan tersembunyi dari citra. Pada metode ini, argumen src digunakan untuk menyimpan path citra yang berisi pesan tersembunyi. Citra di-load menggunakan modul PIL dan dikonversi menjadi matriks array menggunakan modul numpy. Selanjutnya, setiap bit terakhir (LSB) dari matriks array pada setiap saluran warna (R/G/B/A) digabungkan untuk membentuk pesan tersembunyi dalam bentuk biner. Pesan biner tersebut dikonversi menjadi karakter menggunakan fungsi chr dan dihentikan ketika ditemukan penanda akhir pesan. Pesan tersembunyi yang berhasil diambil akan ditampilkan. Jika tidak ada pesan tersembunyi yang ditemukan, pesan akan menampilkan pesan "No Hidden Message Found".

```
def Decode(src):

    #get hidden binary from least significant bit (LSB) from each channel from each pixel until end of line reached
    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4

    total_pixels = array.size//n

    hidden_bits = ""
    for p in range(total_pixels):
        for q in range(0, 3):
            hidden_bits += (bin(array[p][q])[2:][-1])

    hidden_bits = [hidden_bits[i:i+8] for i in range(0, len(hidden_bits), 8)]

    #get hidden message until before end of line
    message = ""
    for i in range(len(hidden_bits)):
        if message[-5:] == "3OfFF":
            break
        else:
            message += chr(int(hidden_bits[i], 2))
    if "3OfFF" in message:
        print("Hidden Message:")
        print(message[:-5])
    else:
        print("No Hidden Message Found")

    return message[:-5]
```

Fig. 9. Metode Decode

Dengan menggunakan metode-metode di atas, pesan dapat disisipkan ke dalam citra menggunakan bit terkecil dan pesan tersembunyi dapat diambil dari citra dalam program yang telah dibuat.

E. Main App dan Validasi

Pada subbab ini, terdapat implementasi utama dari aplikasi berbasis GUI menggunakan framework tkinter. Aplikasi ini mencakup fitur steganografi dan validasi hash pesan tersembunyi dalam file citra menggunakan kunci publik.

Berikut adalah penjelasan mengenai metode yang dilakukan untuk mengembangkan aplikasi tersebut:

- Import library dan modul yang diperlukan, seperti tkinter, ttk, showerror, scrolledtext, askopenfilename, serta modul-modul khusus yang dibuat, yaitu Signature dan Stegano.
- Definisikan class UtilityFunction yang berisi berbagai fungsi utilitas yang digunakan dalam aplikasi. Fungsi-fungsi ini antara lain untuk membuka file, melakukan validasi, dan mengimpor file ke aplikasi.
- Class UtilityFunction juga memiliki fungsi decryptClick yang digunakan untuk memeriksa keabsahan nilai hash pesan tersembunyi dalam file citra dengan menggunakan kunci publik. Jika nilai hash valid, maka akan ditampilkan pesan "Valid", dan jika tidak valid, akan ditampilkan pesan "Invalid".
- Fungsi steganoClick digunakan untuk menerapkan steganografi pada file yang dipilih. Fungsi ini akan mengenkripsi file tanda tangan menggunakan metode SHA256 + RSA. Setelah itu, fungsi akan menanamkan tanda tangan yang telah dienkripsi ke dalam file citra. Setelah proses selesai, akan ditampilkan pesan "Signature Embedded".
- Fungsi-fungsi importKeyFile, importPNGFile, importSteganoPNGFile, dan importSignFile digunakan untuk mengimpor file kunci publik, file citra, file citra hasil steganografi, dan file tanda tangan ke dalam aplikasi, masing-masing.
- Class tkinterApp merupakan class utama yang mewarisi class Tk dari framework tkinter. Class ini bertanggung jawab untuk membuat tampilan GUI dan mengatur frame yang akan ditampilkan.
- Dalam class tkinterApp, terdapat frame-frame yang terdiri dari class Stegano dan Decrypt. Frame Stegano berisi elemen-elemen antarmuka pengguna untuk menerapkan steganografi pada file citra, sementara frame Decrypt berisi elemen-elemen antarmuka pengguna untuk melakukan validasi hash pesan tersembunyi dalam file citra.
- Di dalam masing-masing frame, terdapat elemen-elemen antarmuka pengguna seperti label, tombol, dan area teks yang digunakan untuk memasukkan atau menampilkan informasi terkait file yang diimpor dan proses yang dilakukan.
- Di bagian terakhir, terdapat pemanggilan fungsi tkinterApp() untuk menjalankan aplikasi. Aplikasi akan ditampilkan dalam jendela GUI dengan judul "Image Digital Signing".

Dengan menggunakan aplikasi ini, pengguna dapat menerapkan steganografi pada file gambar dan melakukan validasi hash pesan tersembunyi dalam file gambar dengan menggunakan kunci publik.

IV. HASIL DAN ANALISA

Pada bab ini, akan dijelaskan mengenai hasil yang diperoleh dari implementasi aplikasi Image Digital Signing menggunakan steganografi dan validasi hash. Bab ini juga akan memberikan analisis terhadap hasil tersebut dengan tujuan untuk mengevaluasi efektivitas dan kehandalan aplikasi yang telah dibangun.

Dalam proses pengembangan aplikasi, berbagai fitur telah diimplementasikan, seperti kemampuan untuk menyembunyikan tanda tangan digital dalam file citra menggunakan metode steganografi, serta melakukan validasi terhadap hash pesan tersembunyi dengan menggunakan kunci publik. Aplikasi ini juga dilengkapi dengan antarmuka pengguna berbasis GUI yang memudahkan pengguna dalam mengimpor file dan menjalankan proses steganografi serta validasi.

Hasil yang diperoleh dari pengujian aplikasi akan dianalisis untuk mengevaluasi keandalan dan kualitas kinerja aplikasi. Evaluasi akan mencakup aspek-aspek seperti keakuratan validasi hash, ketahanan terhadap manipulasi citra, dan responsivitas antarmuka pengguna. Selain itu, akan dilakukan pengujian terhadap waktu yang dibutuhkan untuk proses steganografi dan validasi pada berbagai ukuran dan jenis file citra.

Analisa hasil ini akan memberikan pemahaman yang lebih mendalam mengenai kekuatan dan keterbatasan aplikasi Image Digital Signing ini. Hal ini akan memberikan panduan dalam meningkatkan kualitas dan kinerja aplikasi serta mengidentifikasi potensi pengembangan lebih lanjut. Dengan demikian, bab ini akan memberikan informasi yang berharga dalam merumuskan kesimpulan dan rekomendasi mengenai aplikasi Image Digital Signing menggunakan steganografi dan validasi hash.

A. *GUI Aplikasi*

Aplikasi eksperimen memiliki GUI untuk memudahkan *userflow* dan penggunaan dalam implementasi tanda tangan digital pada karya ilustrasi digital. Berikut adalah tampilan antarmuka yang sudah dikembangkan:

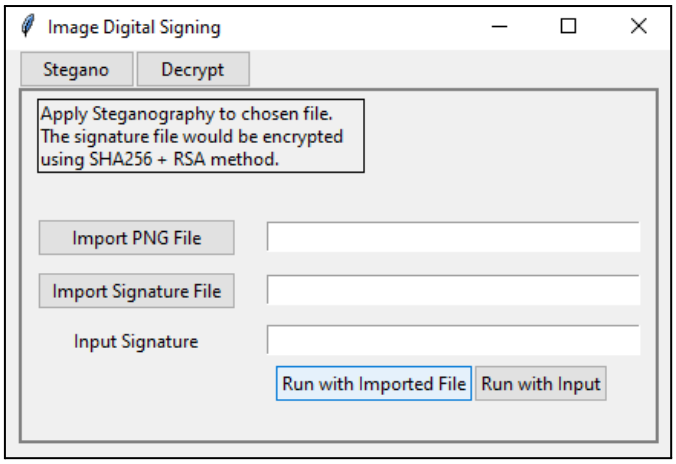

Fig. 10. GUI Aplikasi

Antarmuka pada Fig 10 menampilkan fitur steganografi yang terdiri dari tiga input yang dapat dimasukkan oleh pengguna. 'Import PNG File' yaitu input file gambar yang ingin ditandatangani secara digital, 'Import Signature File' yaitu opsi input file tanda tangan digital yang berupa .txt file dan 'Input Signature' yaitu opsi input tanda tangan berupa teks yang diketikkan secara langsung oleh pengguna. Pengguna dapat memilih untuk memproses steganografi dengan salah satu dari kedua pilihan pada dua *button* di bawah yaitu 'Run with Imported File' atau 'Run with Input'. Setelah semua input dan pilihan pemanggilan fungsi 'steganoClick' telah dilaksanakan maka akan dihasilkan output file .png yang telah di-*encode* dengan tanda tangan digital serta file *public key* untuk melakukan validasi.

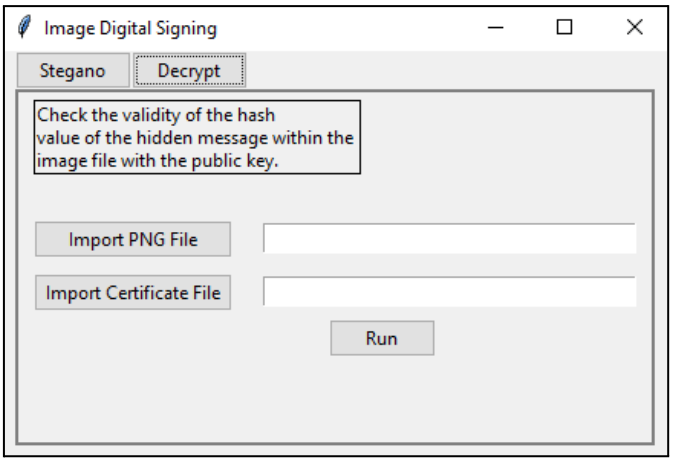

Fig. 11. GUI Decrypt

Antarmuka pada Fig. 11 menampilkan fitur validasi gambar dengan *public key* yang dimiliki oleh pengguna yang terdiri dari dua input yang dapat dimasukkan yaitu file PNG yang ingin diperiksa dan 'Import Certificate File' yaitu input file *public key* yang akan dipasangkan. Ketika pengguna menekan *button* 'Run' akan dilakukan proses validasi yang terdiri dari dekripsi serta verifikasi *hash* yang terdapat pada gambar.

B. *Pengujian*

Pengujian dilakukan dengan melakukan input gambar serta input tanda tangan digital melalui input file tanda tangan. Setelah dilakukan proses steganografi, hasil output gambar diuji validasinya dengan dua kunci yang berbeda yaitu kunci asli dan kunci palsu. Faktor yang diuji adalah kualitas gambar setelah diadakan steganografi, hasil tanda tangan yang di-*encode* ke dalam gambar, dan validasi tanda tangan digital tersebut.

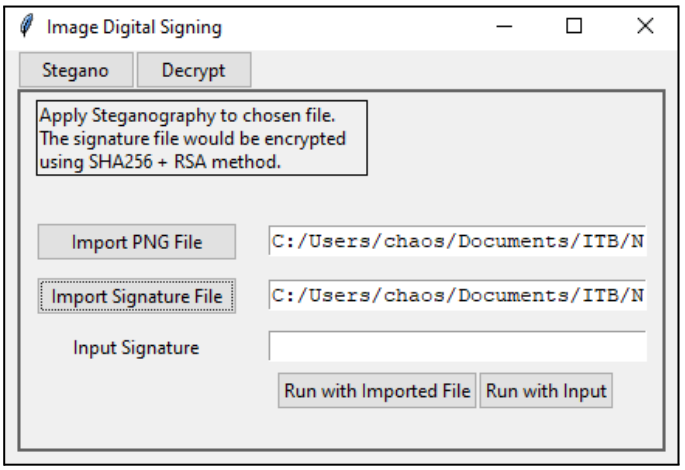

Fig. 12. Input Gambar dengan Tanda Tangan Digital

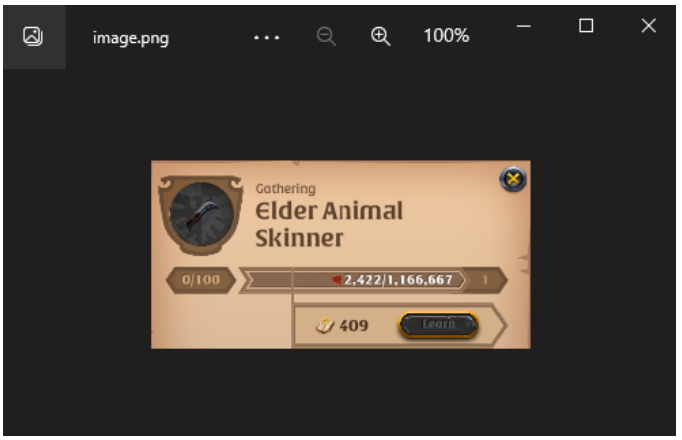

Fig. 13. Input File PNG

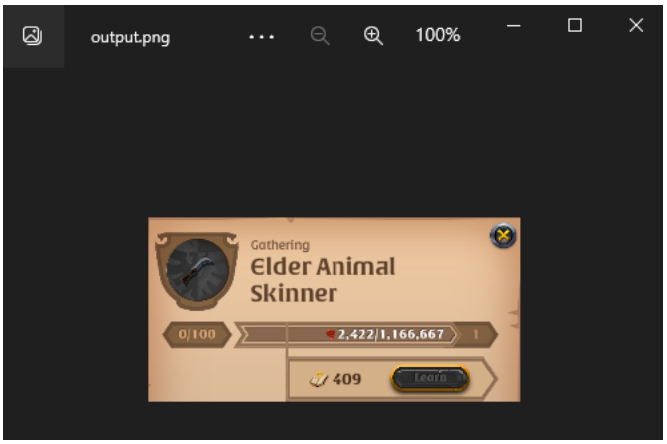

Fig. 14. Output File PNG

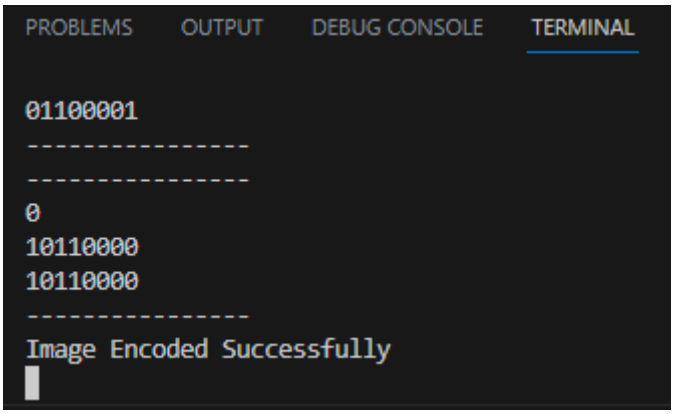

Fig. 15. Hasil *Encoding*

Hasil proses steganografi menunjukkan bahwa kualitas dan integritas gambar tetap terjaga setelah dilakukan steganografi dan pesan yaitu tanda tangan digital berhasil disisipkan dalam gambar.

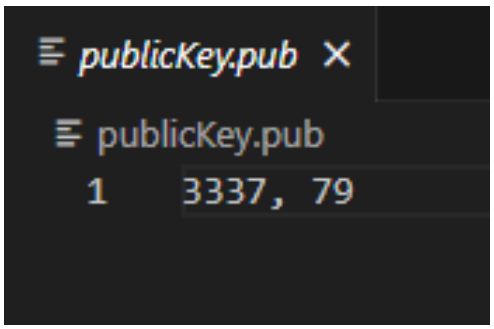

Fig. 16. Output Certificate Key

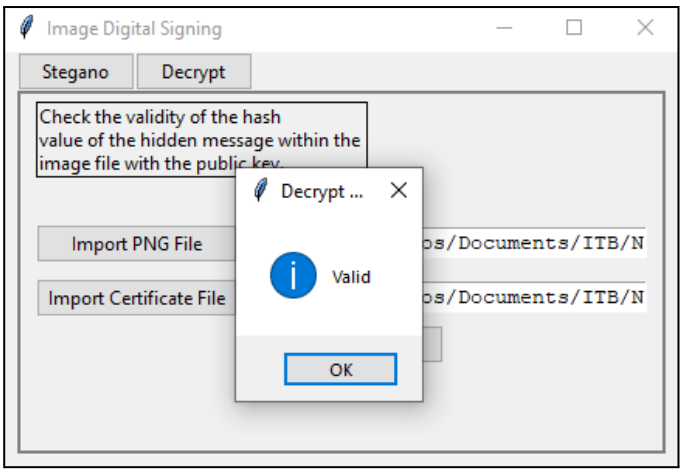

Fig. 17. Hasil Validasi dengan *Public Key*

Hasil proses validasi dengan *public key* yang benar menunjukkan bahwa gambar tersebut valid dengan pasangan sertifikat digital yang dimasukkan.

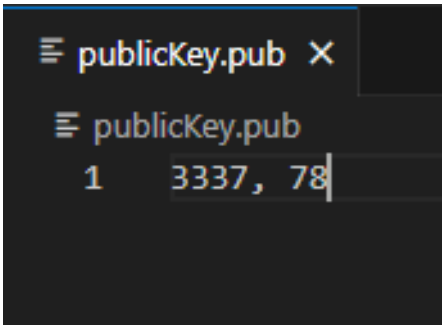

Fig. 18. Public Key Salah

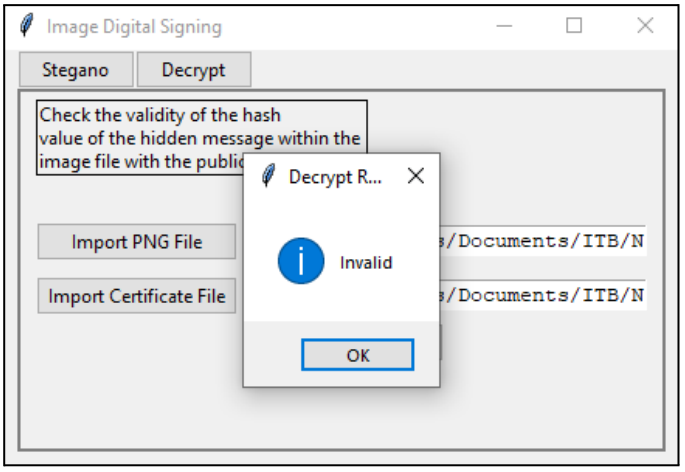

Fig. 19. Hasil Validasi Salah

Hasil proses validasi dengan *public key* yang salah menampilkan output ‘Invalid’ yang berarti pasangan gambar dan sertifikat digital tidak sesuai.

C. *Analisa*

Analisa dilakukan berdasarkan hasil pengujian yang berhasil untuk setiap *use case* yang diimplementasikan dalam eksperimen ini. Hasil pengujian menunjukkan keberhasilan aplikasi Image Digital Signing dalam menjaga keaslian dan integritas karya ilustrasi digital melalui penggabungan steganografi dan tanda tangan digital. Dalam pengujian *use case* penerapan steganografi, aplikasi berhasil menyisipkan tanda tangan digital ke dalam file gambar dengan menggunakan metode SHA256 + RSA. Hal ini terbukti dengan berhasilnya ekstraksi tanda tangan digital dari file gambar yang telah dimodifikasi.

Pada *use case* validasi tanda tangan digital dengan kunci publik, aplikasi mampu memvalidasi keaslian tanda tangan digital dengan menggunakan kunci publik yang sesuai. Publik dapat memverifikasi tanda tangan digital untuk memastikan bahwa karya ilustrasi digital tersebut benar-benar dibuat oleh pembuatnya. Dalam analisa, juga ditemukan bahwa aplikasi dapat menentukan bahwa validasi berupa "invalid" apabila sertifikat digital yang digunakan tidak cocok atau tidak valid. Hal ini menunjukkan bahwa tidak sembarang orang dapat mengakui kepemilikan hak cipta terhadap karya digital yang telah ditandatangani melalui aplikasi ini. Keberhasilan validasi tanda tangan digital ini menjadi indikator keberhasilan implementasi eksperimen.

Namun, analisa juga mengidentifikasi beberapa keterbatasan dalam implementasi eksperimen ini jika diterapkan dalam skala yang lebih besar. Pertama, akses terhadap tanda tangan digital masih terbatas dan kurang terbuka. Pengguna memerlukan akses yang lebih mudah untuk mendapatkan informasi tanda tangan digital dan memverifikasinya. Kedua, penyimpanan kunci publik yang digunakan dalam validasi tanda tangan digital membutuhkan sumber daya yang lebih besar untuk menjaga hak cipta. Dalam skala yang lebih besar, perlu dipertimbangkan infrastruktur yang memadai untuk menyimpan kunci publik secara aman dan terpercaya.

Secara keseluruhan, analisa menunjukkan bahwa implementasi eksperimen ini telah berhasil membuktikan efektivitas penggunaan steganografi dan tanda tangan digital dalam menjaga keaslian karya ilustrasi digital. Namun, dalam skala yang lebih besar, perlu dilakukan pengembangan lebih lanjut untuk meningkatkan aksesibilitas tanda tangan digital dan menyediakan infrastruktur yang dapat mengelola penyimpanan kunci publik dengan efisien dan aman.

V. KESIMPULAN

Dalam penelitian ini, telah diimplementasikan sistem yang mampu menyembunyikan tanda tangan digital dalam citra menggunakan steganografi serta melakukan validasi hash menggunakan kunci publik. Berdasarkan hasil dan analisis yang telah dilakukan, berikut adalah kesimpulan yang dapat diambil:

1. Kelestarian Kualitas Karya Ilustrasi Digital:

   Melalui penggunaan steganografi, tanda tangan digital dapat disembunyikan dalam citra tanpa mengorbankan kualitas visual dari gambar atau karya ilustrasi digital tersebut. Hal ini penting dalam menjaga integritas dan keaslian karya seni digital yang sering kali memiliki nilai estetika dan artistik yang tinggi.

2. Validasi Tanda Tangan Digital dengan Kunci Publik:

   Dengan menggunakan validasi hash menggunakan kunci publik, tanda tangan digital yang tersembunyi dalam citra dapat diverifikasi oleh publik. Hal ini memberikan transparansi kepada masyarakat umum untuk melihat keaslian dan pembuat karya tersebut. Kunci publik juga dapat disimpan di media sosial artist atau platform lainnya agar dapat diakses secara mudah oleh pengguna.

3. Antarmuka Eksperimen yang Sederhana:

   Aplikasi ini dilengkapi dengan antarmuka pengguna berbasis GUI yang sederhana dan mudah digunakan. Hal ini memungkinkan pengguna untuk melakukan eksperimen dengan aplikasi dengan mudah tanpa memerlukan pengetahuan teknis yang mendalam. Antarmuka ini dapat memperluas potensi pengguna dan memudahkan adopsi teknologi ini dalam lingkungan yang lebih luas.

4. Potensi Pengembangan Sistem yang Lebih Luas:

   Eksperimen ini dapat menjadi dasar untuk pengembangan sistem yang lebih luas. Dalam penelitian ini, fokus utama adalah pada steganografi dan validasi hash, namun konsep ini dapat diperluas untuk mencakup berbagai aspek keamanan digital, seperti pengiriman pesan terenkripsi, verifikasi dokumen, atau perlindungan hak cipta. Potensi pengembangan lebih lanjut memungkinkan aplikasi ini untuk digunakan dalam berbagai konteks dan skenario.

5. Keterbatasan dan Tantangan:

   Meskipun aplikasi ini memberikan solusi yang baik, terdapat beberapa keterbatasan yang perlu diperhatikan. Akses terhadap tanda tangan digital yang tersembunyi mungkin kurang terbuka dan mudah untuk diakses oleh masyarakat umum. Selain itu, penyimpanan kunci publik juga membutuhkan sumber daya yang lebih besar untuk menjaga keamanan dan hak cipta.

Dalam kesimpulannya, penelitian ini telah mengimplementasikan aplikasi Image Digital Signing menggunakan steganografi dan validasi hash. Hasilnya menunjukkan bahwa steganografi dapat menjaga kualitas gambar/karya ilustrasi digital, sementara validasi tanda tangan digital dengan kunci publik memungkinkan publik untuk melihat keaslian dan pembuat karya tersebut. Antarmuka eksperimen yang sederhana juga mempermudah penggunaan aplikasi ini. Namun, terdapat tantangan dalam akses tanda tangan digital dan penyimpanan kunci publik. Dengan pengembangan lebih lanjut, aplikasi ini memiliki potensi untuk digunakan dalam berbagai konteks keamanan digital dan perlindungan hak cipta.

Links

Source Code untuk eksperimen makalah ini:

https://github.com/chaosthepotato/DigitalSignatureForDigitalArt

Video Penjelasan Singkat Eksperimen:

https://youtu.be/V4oOD9LRaAA

Acknowledgment

Penulis ingin menyampaikan penghargaan yang tulus kepada teman-teman dan rekan-rekan yang telah memberikan dukungan dan bantuan berharga selama proses penelitian dan penulisan makalah ini. Dorongan dan masukan mereka sangat berarti dalam membentuk ide-ide yang disajikan di sini.

Terima kasih yang tak terhingga kepada Bapak Rinaldi Munir, instruktur kami yang terhormat dalam mata kuliah Kriptografi dan Koding, atas bimbingan dan keahliannya. Pengetahuan dan wawasan beliau telah sangat memperkaya pemahaman kami tentang materi yang dibahas dan berperan penting dalam pengembangan karya ini.

Kami juga ingin mengucapkan terima kasih kepada komunitas akademik dan institusi terkait yang telah menyediakan sumber daya dan fasilitas yang diperlukan dalam menyelesaikan penelitian ini dengan sukses.

Terakhir, kami mengucapkan apresiasi kepada semua pihak yang telah memberikan kontribusi dan dukungan dalam berbagai bentuknya. Tanpa dukungan dari teman-teman dan pembimbing, makalah ini tidak akan terwujud.

References

[1] VAVER, David. “Intellectual Property : The State of the Art.” 2000. LAW QUARTERLY REVIEW, vol. 116, no. October 2000, Thomson Reuters, 2000, pp. 621–637.

[2] MUNIR, Rinaldi. “Algoritma RSA”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/10-Algoritma-RSA-2023.pdf

[3] MUNIR, Rinaldi. “Tanda Tangan Digital”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/16-Tanda-tangan-digital-2023.pdf

[4] MUNIR, Rinaldi. “Fungsi Hash”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/13-Fungsi-hash-2023.pdf

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

Panegak Raya Pasaribu 18219029