Implementasi Algoritma RSA dan Fungsi Hash SHA-3 Untuk Pembangkitan Tanda Tangan Digital Pada Faktur Elektronik (*Electronic Invoice)*

Rachmad Hidayat 18220049
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email : rachmadhi26@gmail.com

***Abstract*—Faktur (*invoice*) merupakan sebuah dokumen yang menunjukan jumlah yang berhak ditagih kepada pelanggan atau orang tertentu yang menunjukan informasi kuantitas, harga, dan jumlah tagihan. Sementara itu, faktur elektronik (*electronic invoice*) adalah dokumen faktur yang dibuat, dikirimkan, dan diterima dalam bentuk elektronik, umumnya dalam format digital. Faktur elektronik dapat diamankan dengan menggunakan tanda tangan digital (*digital signature*). Tanda tangan digital (*digital signature*) merupakan salah satu teknik kriptografi yang digunakan untuk menjaga keamanan faktur elektronik. Pada makalah ini akan dibahas mengenai pembuatan tanda tangan digital untuk faktur elektronik menggunakan algoritma RSA dan fungsi *hash* SHA-3.**

Keywords—Electronic Invoice, tanda tangan digital, kriptografi asimetris

I. Pendahuluan

Di era serba digital seperti saat ini, kegiatan-kegiatan yang dilakukan sehari-hari banyak yang mengadopsi teknologi untuk meningkatkan efektivitas dan efisiensi. Salah satunya yaitu kegiatan bisnis, di zaman sekarang kegiatan bisnis terutama yang berhubungan dengan transaksi banyak dilakukan melalui digital, sebagai contoh misalnya pengiriman faktur elektronik (*electronic invoice*).

Electronic invoice digunakan untuk melakukan penagihan pembayaran kepada pihak lain secara digital. *Electronic invoice* ini sangat membantu dalam kegiatan bisnis karena dapat meningkatkan efisiensi administrasi dengan mengurangi ketergantungan pada proses manual seperti pencetakan, pengiriman, dan pengarsipan faktur fisik. Selain itu, dengan adanya *electronic invoice* memudahkan dalam akses dan pencarian karena faktur dapat dicari melalui sistem digital sehingga mempermudah dalam pengelolaan dan pemantauan riwayat transaksi.

Seiring berkembangnya teknologi, *electronic invoice* harus dapat dikirimkan dengan aman dan keaslian *electronic invoice* harus terjamin. Salah satu teknologi yang dapat membantu mengamankan *electronic invoice* yaitu tanda tangan digital (*digital signature*). Tanda tangan digital memungkinkan penerima memverifikasi keaslian dokumen *electronic invoice* dan memastikan integritas dari dokumen tersebut. Dalam buku *Information Security Management Handbook* dijelaskan bahwa terdapat tiga aspek yang perlu diperhatikan dalam keamanan informasi yaitu kerahasiaan (*confidentiality*), integritas (*integrity*), dan ketersediaan (*availability*). Selain tiga aspek tersebut, terdapat dua aspek tambahan yaitu autentikasi (*authentication*) dan anti penyangkalan (*non-repudiation*). Dengan adanya tanda tangan digital pada *electronic invoice*, memungkinkan untuk memenuhi aspek integritas, autentikasi, dan anti penyangkalan.

Tanda tangan digital dapat dibuat dengan beberapa algoritma kriptografi seperti algoritma RSA, algoritma ECC(*Eliptic Curve Cryptography*), dan lain-lain. Salah satu algoritma kriptografi asimetris yang paling populer untuk membangkitkan tanda tangan digital adalah algoritma kriptografi RSA. Algoritma RSA merupakan algoritma kriptografi kunci publik yang cukup aman untuk membangkitkan tanda tangan digital. Dalam membangkitkan tanda tangan digital fungsi *hash* juga diperlukan untuk mengambil input berupa pesan dan menghasilkan output yang memiliki ukuran yang seragam dan tetap. Fungsi *hash* yang sering digunakan yaitu SHA (Secure Hash Algorithm). Fungsi SHA yang dipilih pada makalah ini yaitu SHA-3. SHA-3 dipilih karena ketahanannya terhadap kolisi. Dalam tanda tangan digital, nilai hash yang dihasilkan oleh fungsi *hash* SHA-3 akan ditandatangani menggunakan kunci privat RSA untuk menghasilkan tanda tangan yang unik.

Dalam makalah ini, penulis akan membahas implementasi algoritma RSA dan fungsi *hash* SHA-3 untuk membangkitkan tanda tangan digital pada *electronic invoice.* Penulis akan menjelaskan langkah-langkah yang diperlukan untuk menghasilkan tanda tangan digital yang valid dan memberikan contoh kode program menggunakan bahasa python. Diharapkan penelitian ini dapat memberikan pemahaman yang lebih mendalam mengenai impelementasi algoritma RSA dan fungsi hash SHA-3 untuk pembangkitan tanda tangan digital pada *electronic invoice*.

II. Dasar Teori

A. Faktur Elektronik (Electronic Invoice)

Faktur (*invoice*) merupakan sebuah dokumen yang menunjukan jumlah yang berhak ditagih kepada pelanggan atau orang tertentu yang menunjukan informasi kuantitas, harga, dan jumlah tagihan. [1]

Faktur digunakan sebagai bukti pembelian dan sebagai dasar untuk proses administrasi, seperti pembayaran, pencatatan akuntansi, dan pemantauan inventaris. Faktur umumnya mencantumkan informasi seperti nama penjual, nama pembeli, tanggal transaksi, deskripsi barang atau jasa, jumlah, harga, total tagihan yang harus dibayar, dan rincian pembayaran lainnya.

Faktur elektronik (*electronic invoice*) adalah dokumen faktur yang dibuat, dikirimkan, dan diterima dalam bentuk elektronik, umumnya dalam format digital seperti PDF dan XML. Faktur elektronik memiliki struktur yang sama dengan faktur fisik namun faktur elektronik memiliki keunggulan dalam hal efisiensi dan keamanan. Keunggulan tersebut misalnya dalam efisiensi administrasi dengan mengurangi penggunaan kertas, penghematan biaya pencetakan dan pengiriman fisik faktur, dan mempermudah dalam mengelola dan mencari informasi faktur.

B. Algoritma RSA

Algoritma RSA adalah sebuah algoritma kriptografi asimestris yang menggunakan sepasang kunci yaitu kunci publik dan kunci privat, untuk melakukan enkripsi dan dekripsi terhadap data atau pesan. Algoritma ini diciptakan oleh tiga orang dari Massachusetts Institute of Technology (MIT) yaitu Ron Rivest, Adi Shamir, dan Len Adleman pada tahun 1977. Algoritma RSA didasarkan pada kesulitan dalam memfaktorkan bilangan bulat yang besar. Keamanan algoritma RSA ini sangat bergantung pada kesulitan dalam memecahkan persamaan faktorisasi dari dua buah bilangan prima. Algortima RSA dapat dianggap aman jika nilai dari dua buah bilangan prima tersebut besar. Berikut merupakan cara pembangkitakn kunci, enkripsi, dan dekripsi menggunakan algoritma RSA :

1. Cara Pembangkitan Pasangan Kunci
   a. Pilih dua buah bilangan prima besar, p dan q.
   b. Hitung nilai dari n dimana
      $$n = p*q$$
   c. Hitung nilai dari
      $$\varphi(n) = (p - 1) * (q - 1),$$
      dimana φ merupakan fungsi Euler totient.
   d. Selanjutnya, pilih bilangan bulat e yang lebih kecil dari φ(n) dan relatif prima dengan φ(n).
   e. Hitung bilangan bulat d yang memenuhi persamaan
      $$(d * e) \bmod \varphi(n) = 1.$$
   f. Dengan demikian akan tebentuk pasangan kunci yang terdiri dari kunci publik (e,n) dan kunci privat (d,n)
2. Cara Enkripsi
   a. Ambil pesan atau plaintext (p) yang akan dienkripsi
   b. Bagi kedalam beberapa blok yang lebih kecil: m1, m2, ... (syarat 0 mi< n)
   c. Repesentasikan dalam bentuk angka m, dengan $0 \leq m < n$.
   d. Hitung ciphertext (c_i) dengan menggunakan rumus
      $$c_i = (m_i \text{^} e) \bmod n.$$
3. Cara Dekripsi
   a. Setiap blok cipherteks c_i, hitung nilai mi menggunakan kunci privat d dengan rumus
      $$m_i = (c_i \text{^} d) \bmod n.$$

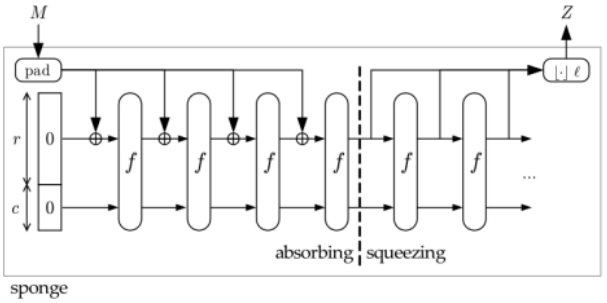
C. Fungsi Hash SHA-3

SHA-3 (Secure Hash Algorithm 3) merupakan sebuah fungsi hash kriptografik yang digunakan untuk mengubah suatu data atau pesan dengan panjang yang bervariasi menjadi sebuah nilai hash yang tetap. Fungsi hash SHA-3 merupakan salah satu varian dari fungsi hash Secure Hash Algorithm (SHA) yang dikembangkan oleh National Security Agency (NSA) dan ditebitkan oleh National Institute of Standards and Technology (NIST) di Amerika Serikat. SHA-3 diusulkan sebagai pengganti SHA-2 dengan tujuan meningkatkan keamanan dan kinerja fungsi hash.

Fungsi hash SHA-3 banyak diaplikasikan dalam keamanan informasi, misalnya penggunaan dalam pembangkitan tanda tangan digital, verifikasi integritas data, penyimpanan kata sandi dalam bentuk hash, dan masih banyak lagi. Fungsi hash SHA-3 juga menjadi salah satu standar industri yang banyak digunakan dalam berbagai protokol dan aplikasi kriptografi.

Fungsi hash SHA-3 dirancang untuk memiliki sifat ketahanan terhadap kolisi jadi untuk menemukan dua input yang berbeda untuk menghasilkan nilai hash yang sama dianggap sulit dan membutuhkan banyak waktu.

Fungsi hash SHA-3 dibuat menggunakan kontsuksi ‘spons’, fungsi hash SHA-3 menggunakan fungsi non-kompresi untuk menyerap dan kemudian ‘memeras’ digest.



sponge

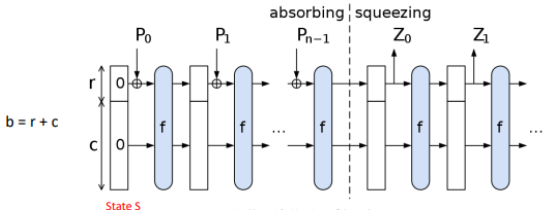
Gambar 1 Konstruksi Spons (Sumber: Slide Kuliah II4031 Kriptografi dan Koding - Rinaldi Munir)

Fungsi SHA-3 melibatkan tiga tahapan yaitu pra-proses, *absorbing,* dan *squeezing*. Berikut merupakan penjelasan dari tahapan-tahapan tersebut.

1. Pra-proses

   Misalkan panjang digest yang diingikan adalah d bit. Pesan *M* akan ditambah dengan padding agar menjadi string *P* yang habis dibagi dengan *r* atau *n =length(P)/r*. Selanjutnya *P* dipotong menjadi blok blok *Pi* berukuran *r*-bit. Setelah itu *b-bit* dari peubah status *S* diinisialisasi menjadi 0.

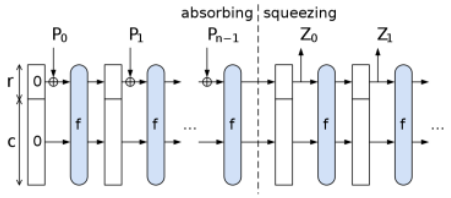
2. *Absorbing*



Gambar 2 Tahapan *Absorbing* (Sumber: Slide Kuliah II4031 Kriptografi dan Koding - Rinaldi Munir)

   Pada tahapan ini setiap blok *Pi* berukuran *r*-bit akan di XOR dengan *r*-bit pertama *state S*, dan hasilnya dimasukan ke fungsi permutasi *f* hasilnya menjadi *state* baru *S*.

3. Squeezing



Gambar 3 Tahapan Squeezing (Sumber: Slide Kuliah II4031 Kriptografi dan Koding - Rinaldi Munir)

   Pada tahapan ini, message digest akan disimpan dalam *Z* (*Z* diinisialisasi dengan string kosong). Selanjutnya, jika panjang *Z* belum sama dengan *d*, *r*-bit pertama dari *S* akan disambungkan ke *Z*. Setelah itu, jika panjang *Z* belum sama dengan *d*, masukkan ke dalam permutasi *f* untuk menghasilkan *state* baru *S*.

D. Tanda Tangan Digital

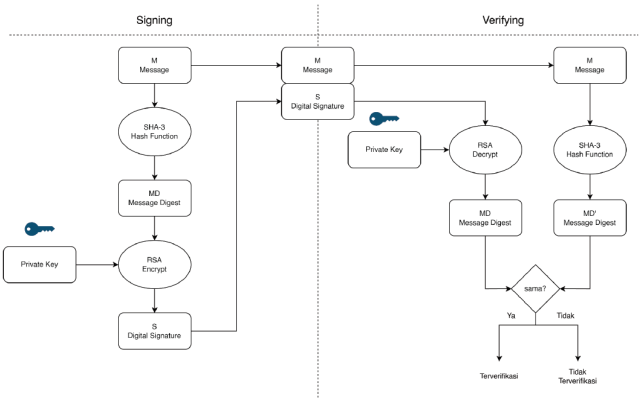
Tanda tangan digital (*digital signature*) merupakan teknik kriptografi yang digunakan untuk menjaga keamanan dokumen digital. Tanda tangan digital bukan tanda tangan seseorang yang discan, melainkan nilai kriptografis yang dibangkitkan dengan algoritma tertentu dan bergantung pada pesan dan pengirim pesan. Pada dasarnya, tanda tangan digital melibatkan beberapa proses berikut:

1. Pembangkitan pasangan kunci privat dan kunci publik. Dimana kunci privat harus dijaga kerahasiaanya dan kunci publik dapat diberikan ke orang lain.
2. Pembangkitan tanda tangan digital menggunakan algoritma tertentu dan kunci privat pengirim untuk menghasilkan tanda tangan digital.
3. Verifikasi tanda tangan digital dengan menggunakan kunci publik terkait untuk memastikan validitas tanda tangan digital dan integritas dari pesan.

Tanda tangan digital memiliki keunggulan dalam hal kemudahaan penggunaan, kecepatan, dan keamanan dibandingkan dengan tanda tangan konvensional.

Pada makalah kali ini, tanda tangan digital dibangkitkan menggunakan algoritma kriptografi RSA dan fungsi hash SHA-3 yang sudah dijelaskan sebelumnya. Algoritma kriptografi kunci-publik RSA digunakan untuk membangkitkan pasangan kunci privat dan kunci publik. Sementara itu, fungsi *hash* digunakan untuk menghasilkan sebuah nilai *hash* dari pesan agar menjadi sebuah *string* yang unik dengan panjang yang tetap. Nilai hash ini yang selanjutnya akan dienkripsi menggunakan algoritma RSA dan kunci privat penulis pesan untuk membangkitkan sebuah tanda tangan digital yang unik.

Proses pemverifikasian tanda tangan digital, dilakukan dengan cara penerima pesan yang ditandatangani memisahkan pesan dan tanda tangan digitalnya. Selanjutnya, penerima mendekripsi tanda tangan digital menggunakan algoritma RSA dan kunci publik dari pengirim pesan. Setelah itu, penerima harus menghitung nilai hash dari pesan. Jika nilai hash pesan sama dengan hasil dekripsi tanda tangan digital maka tanda tangan digital tersebut valid dan pesan terjaga integritasnya. Sebaliknya, jika tidak maka mungkin pesan telah mengalami perubahan selama pengiriman. Berikut diagram yang mengambarkan proses penandatanganan pesan dan verifikasi pesan.



Gambar 4 Diagram Proses Penandatangan dan Verifikasi Tanda Tangan Digital (Sumber: Gambar Pribadi)

III. RANCANGAN

Perancangan implementasi algorima RSA dan fungsi *hash* SHA-3 untuk pembangkitan tanda tangan digital ini akan dibagi kedalam beberapa modul yaitu pembangkitan kunci,

algoritma RSA, dan *main*. Berikut merupakan detail untuk setiap modulnya.

1. Modul Pembangkitan Kunci

   Fungsi utama dari modul ini yaitu pembangkitan pasangan kunci publik dan kunci privat. Selain itu, terdapat fungsi pembantu untuk mempermudah dalam membangkitkan kunci publik maupun kunci privat yaitu fungsi inisiasi yang menghasilkan n dan totient, fungsi pengecekan bilangan prima, fungsi pembangkitan bilangan prima acak, fungsi pencarian *greatest common divisor* (GCD), dan fungsi pengecekan bilangan relatif prima.

2. Modul Algoritma RSA

   Pada modul algoritma RSA, terdapat dua fungsi utama yaitu fungsi enkripsi untuk membangkitkan tanda tangan digital dan fungsi dekripsi untuk melakukan verifikasi terhadap tanda tangan digital.

3. Modul *main*

   Modul ini menjadi modul utama. Modul ini digunakan untuk menjalankan program. Pada modul ini terdapat fungsi untuk menghitung nilai *hash* suatu file menggunakan fungsi *hash* SHA-3. Selain itu, terdapa fungsi untuk membuat tanda tangan digital, fungsi untuk menyimpan tanda tangan digital dalam file terpisah, dan menyimpan kunci privat serta kunci publik.

IV. Implementasi dan Pengujian

A. Implementasi

Berikut merupakan implementasi program pembangkitan tanda tangan digital menggunakan algoritma RSA dan fungsi *hash* SHA-3.

1. Modul Pembangkitan Kunci

   Diimplemnetasikan dalam *file* key_generator.py

```
import random

def is_prime(num):
    if num <= 1 :
        return False

    for i in range (2, int(num**0.5)+1):
        if num % i == 0 :
            return False

    return True

def random_prime():
    num = random.randrange(1,
1000000000)
    while not(is_prime(num)):
        num = random.randrange(1,
1000000000)
    return num

def initiate(p,q):
    n = p*q
    totient = (p-1)*(q-1)

    return n, totient

def generate_public_key(n, totient):
    e = random.randrange(1, totient)
    while greatest_common_divisor(e,
totient) != 1:
        e = random.randrange(1, totient)

    return (e,n)

def generate_private_key(totient,
pubkey):
    key, n = pubkey
    d_old, d_new = 0, 1
    r_old, r_new = totient, key
    while r_new != 0:
        quotient = r_old // r_new

        temp = r_old
        r_old = r_new
        r_new = temp - quotient * r_new

        temp1 = d_old
        d_old = d_new
        d_new = temp1 - quotient * d_new

    d = 0
    if r_old == 1 :
        d = d_old % totient

    return (d, n)

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

def check_relative_prime(a, b):
    is_relative_prime = False
    while b != 0 :
        temp = a
        a = b
        b = temp % b

    if a == 1 :
        is_relative_prime = True
```

```
    return is_relative_prime
```

2. Modul Algoritma RSA

   Diimplemnetasikan dalam *file* RSA_algoritma.py

```
import hashlib

def encrypt_digest(prikey, plaintext):
    key, n = prikey
    ciphertext = []

    for char in plaintext:

ciphertext.append(pow(ord(char),key,n))

    hex_sign = f''
    for char in ciphertext:
        hex_sign = hex_sign + hex(char)

    return (hex_sign)

def decrypt_digest(pubkey, ciphertext):
    plaintext_arr = []
    key, n = pubkey
    ciphertext = ciphertext.strip()
    if ciphertext.startswith('0x'):
        parts = ciphertext.split('0x')[1:]
        for part in parts:
            plaintext_arr.append(int(part, 16))

    plaintext =""
    for char in plaintext_arr:
        plaintext += (chr(pow(char, key, n)))

    return plaintext
```

3. Modul *main*

   Diimplemnetasikan dalam *file* main.py

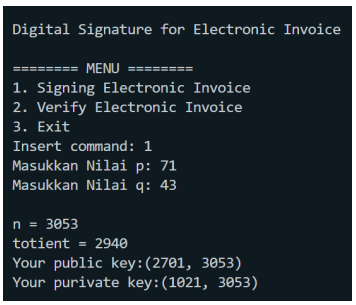
```
import hashlib
from RSA_algoritma import *
from key_generator import *

BUF_SIZE = 65536

def hashfile(filename):
    sha = hashlib.sha3_512()
    with open(filename, 'rb') as f:
        while True:
            data = f.read(BUF_SIZE)
            if not data:
                break
            sha.update(data)
    return sha.hexdigest()

def sign_file(file_path, private_key):
    file_hashed = hashfile(file_path)
    return encrypt_digest(private_key,
file_hashed)

def save_signature_file(signature):
    with
open('Invoice_Digital_Signature.txt', 'w')
as file:
        file.write(
f"""
***BEGIN OF DIGITAL
SIGNATURE***
{signature}
***END OF DIGITAL SIGNATURE***

Digital Signature Algorithm :
RSA with SHA-256
""")

    print("Invoice digital signature saved
successfully")
    file.close

def save_key_file(pubkey, prikey):
    with open('Public Key.pub', 'w') as file:
        file.write(pubkey)

    print("Public Key saved successfully")
    file.close

    with open('Private Key.pri', 'w') as file:
        file.write(prikey)

    print("Private Key saved successfully")
    file.close

def __main__():
    print("Digital Signature for Electronic
Invoice")
    print("")
    Exit = False
    while Exit == False:
        print("========= MENU
=========")
        print("1. Signing Electronic
Invoice")
        print("2. Verify Electronic Invoice")
        print("3. Exit")
        command = int(input("Insert
command: "))
        if command == 1 :
            # key generator
            p = int(input("Masukkan Nilai p:
```

```
"))
        q = int(input("Masukkan Nilai q:
"))
        n, totient = initiate(p,q)
        print("")
        print("n = " + str(n))
        print("totient = " + str(totient))

        pubkey =
generate_public_key(n,totient)
        prikey =
generate_private_key(totient, pubkey)
        print("Your public key:" +
str(pubkey))
        print("Your purivate key:" +
str(prikey))
        print('')

        # Signing Electronic Invoice
        file = str(input("Insert filename:
"))
        filename = ""+file
        message_digest =
hashfile(filename)
        print(message_digest)
        print('')

        digital_sign =
encrypt_digest(prikey, message_digest)
        print("Signing successful")
        print('')

        save_signature_file(digital_sign)

save_key_file(str(pubkey),str(prikey))
    elif command == 2 :
        # Verify Electronic Invoice
        file = str(input("Insert filename:
"))
        filename = ""+file
        digital_sign_file =
str(input("Insert digital signature
filename: "))
        digital_signature =
""+digital_sign_file
        pubkey = str(input("Insert public
key filename: "))
        pubkey_name = ""+pubkey

        #Read digital signature
        with open(digital_signature, 'r') as
file:
            lines = file.readlines()
            if len(lines) >= 2:
                digital_sign = lines[2]

        #Read digital signature
        with open(pubkey_name, 'r') as
file:
            public_key = file.read()
            public_key =
public_key.replace("(", "").replace(")", "")
            pubkey = tuple(map(int,
public_key.split(",")))

        #Verify Electronic Invoice Digital
Signature
        message_digest =
hashfile(filename)
        verify =
decrypt_digest(pubkey,digital_sign)
        print('')
        if verify == message_digest :
            print("Verify Result: True")
        else :
            print("Verify Result: False")
    else :
        Exit = True
        exit()

__main__()
```

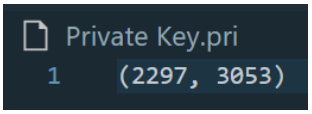
B. Pengujian

Berikut merupakan pengujian yang dilakukan terhadap program.

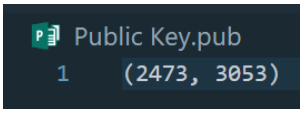
1. Pembangkitan Pasangan Kunci



Gambar 5 Pembangkitan Pasangan Kunci



Gambar 6 Contoh File Kunci Privat



Gambar 7 Contoh File Kunci Publik

2. Pembangkitan Tanda Tangan Digital

Gambar 8 Contoh File Faktur Elektronik

```
Your public key:(2473, 3053)
Your purivate key:(277, 3053)

Insert filename: Invoice.pdf
274249ffe7b3567f47e736cbcfa746acdd5010e6a7955ea6943b3813d9a60a82fde7cf0276
006834f6c71b3505

Signing successful
```

Gambar 9 Penandatanganan File Faktur Elektronik

```
Invoice_Digital_Signature.txt
1
2   ***BEGIN OF DIGITAL SIGNATURE***
3   0x6b20xc90x5a20x6b20x5a20x3d10x2c00x2c00x29d0xc
4   ***END OF DIGITAL SIGNATURE***
5
6   Digital Signature Algorithm :
7   RSA with SHA-256
```

Gambar 10 File Tanda Tangan Digital

3. Verifikasi Tanda Tangan Digital

```
======== MENU ========
1. Signing Electronic Invoice
2. Verify Electronic Invoice
3. Exit
Insert command: 2
Insert filename: Invoice.pdf
Insert digital signature filename: Invoice_Digital_Signature.txt
Insert public key filename: Public Key.pub

Verify Result: True
```

Gambar 11 Verifikasi Tanda Tangan Digital

```
======== MENU ========
1. Signing Electronic Invoice
2. Verify Electronic Invoice
3. Exit
Insert command: 2
Insert filename: Invoice.pdf
Insert digital signature filename: Invoice_Digital_Signature.txt
Insert public key filename: Public Key.pub

Verify Result: False
```

Gambar 12 Verifikasi Tanda Tangan Digital Jika Tanda Tangan Digital Diubah

V. KESIMPULAN

Berdasarkan penelitian dan percobaan yang telah dilakukan, dapat disimpulkan bahwa tanda tangan digital menggunakan algoritma RSA dan fungsi hash SHA-3 dapat membantu dalam memenuhi aspek kemanan informasi yaitu *non-repudiation* dan *integrity* pada *electronic invoice*. Dengan menggunakan kedua algoritma tersbebut, risiko pemalsuan dan perubahan data dapat dikurangi, dan kepercayaan dalam *electronic invoice* dapat ditingkatkan.

Diharapan pengembangan tanda tangan didigital pada faktur elektronik ini dapat ditingkatkan lagi dengan membubuhkan tanda tangan digital langsung pada *file* faktur elektronik berformat pdf.

KODE PROGRAM

https://drive.google.com/file/d/1ewj6yLXhOI231KrQgD-7eRqO6TGsn2rk/view?usp=share_link

UCAPAN TERIMAKASIH

Terima kasih saya ucapkan kepada Tuhan Yang Maha Esa atas limpahan rahmat dan hidayahnya sehingga saya dapat menyelesaikan makalah yang berjudul "Implementasi Algoritma RSA dan Fungsi Hash SHA-3 Untuk Pembangkitan Tanda Tangan Digital Pada Faktur Elektronik (Electronic Invoice)" untuk memenuhi tugas kuliah II4031 Kriptografi dan Koding. Saya juga berterima kasih kepada seluruh pihak yang turut membantu, terutama Bapak Dr. Ir. Rinaldi Munir, M.T. yang memberikan ilmu mengenai kriptografi dan koding.

REFERENCES

[1] La Midjan, Sistem Informasi Akuntansi I, Edisi kedelapan, Penerbit Lingga Jaya, Bandung Definisi Faktur, 2001.

[2] M. Ray Rizaldy, "Perbandingan Tanda Tangan Digital RSA dan DSA Serta Implementasinya untuk Antisipasi Pembajakan Perangkat Lunak," Bandung, 2009..

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

[4] Munir, Rinaldi. 2023. Slide Kuliah II4031 Kriptografi dan Koding: Tanda Tangan Digital

[5] National Institute of Standards and Technology (NIST). (2020). Federal Information Processing Standards Publication (FIPS) 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.

[6] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

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

Rachmad Hidayat