Analisis Perbandingan Algoritma *ElGamal Signature* dan *Elliptic Curve-Digital Signature Algorithm* (ECDSA)

Gresya Angelina Eunike Leman / 18220104
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 18220104@std.stei.itb.ac.id

***Abstract*—Seiring perkembangan teknologi, semakin banyak penyebaran *file* digital melalui berbagai metode dan platform, sehingga dibutuhkan mekanisme untuk menjamin keaslian dokumen dan otektikasi pemilik *file* asli. Tanda tangan digital hadir untuk memenuhi kebutuhan tersebut dengan menyediakan layanan verifikasi keaslian *file* dan juga identitas pengirim. Secara umum, algoritma tanda tangan digital memanfaatkan teknik enkripsi dan *hashing*. Pada makalah ini, penulis ingin mempelajari dan menganalisis implementasi algoritma tanda tangan digital selain algoritma *Digital Signature Algorithm* (DSA) yang umumnya digunakan, yaitu *ElGamal Signature* dan *Elliptic Curve-Digital Signature Algorithm* (ECDSA).**

Keywords—Tanda tangan digital; ElGamal Signature; Elliptic Curve-Digital Signature Algorithm;

I. Pendahuluan

Perkembangan dan pemanfaatan teknologi sudah meningkat dengan pesat, terlebih sejak pandemi Covid-19 yang memaksa semua orang untuk bekerja dari rumah. Pada kondisi tersebut, terdapat banyak kegiatan yang biasanya terbatas waktu dan tempat yang kemudian didigitalisasi dengan pemanfaatan teknologi dan internet sehingga dapat dilakukan dari mana saja dan kapan saja. Tanda tangan sebagai persetujuan surat-surat penting tidak luput dari digitalisasi berbagai kegiatan tersebut. Namun, tanda tangan elektronik (tanda tangan yang berbentuk digital) tidak cukup aman karena sangat mudah dimodifikasi dan disalahgunakan oleh berbagai pihak. Oleh sebab itu, diperlukan sebuah metode untuk menjamin keaslian dan mengetahui pengirim dari *file* tertentu agar dapat dipertanggungjawabkan namun tidak mudah disalahgunakan oleh pihak lain. Kriptografi dapat membantu dalam mencapai tujuan tersebut dengan kombinasi penggunaan teknik enkripsi kunci-publik dan juga teknik *hashing* yang dinamakan tanda tangan digital.

Terdapat banyak algoritma yang dapat digunakan untuk melakukan tanda tangan digital, salah satu yang paling popular adalah *Digital Signature Algorithm* (DSA) yang umumnya memanfaatkan AES dan fungsi Hash SHA-1, SHA-256 atau SHA-3. Namun, terdapat pula algoritma lain yang dapat digunakan dalam membubuhkan tanda tangan digital pada sebuah dokumen, seperti *ElGamal Signature* atau *Elliptic Curve-Digital Signature Algorithm* (ECDSA). Pada makalah ini, akan dibandingkan analisis antara kedua algoritma alternatif tanda tangan digital tersebut. Analisis disertai dengan implementasi akan dilakukan untuk membandingkan kecepatan eksekusi *signing* dan *verifying* dari tanda tangan digital.

II. Dasar Teori

A. Tanda Tangan Digital

Tanda tangan digital merupakan suatu skema matematis yang digunakan untuk melakukan verifikasi dan otentikasi dari pesan atau dokumen digital. Secara garis besar, tanda tangan biasa yang di-digitasi (*digitized signature*) dan tanda tangan digital itu berbeda. Tanda tangan digital digunakan untuk mengetahui siapa pengirim asli dari sebuah pesan atau dokumen dan apakah pesan atau dokumen tersebut masih asli atau sudah mengalami modifikasi. Dengan kata lain, tanda tangan harus dapat memenuhi layanan kriptografi *data integrity, authentication,* dan juga *non-repudiation.* Tanda tangan digital pada setiap pesan atau dokumen pasti berbeda karena bergantung pada kunci yang digunakan untuk enkripsi dan juga isi dari pesan tersebut, sehingga dapat dipastikan keamanannya. Secara umum, terdapat 2 cara untuk membubuhkan tanda tangan digital, yaitu:

1) Mengenkripsi pesan
2) Menggunakan kombinasi fungsi *hash* dan kriptografi kunci-publik

Namun pada makalah ini akan digunakan metode kedua, yaitu penggunaan kombinasi fungsi hash dan kriptografi kunci-publik. Berikut adalah gambaran skema tersebut secara umum.

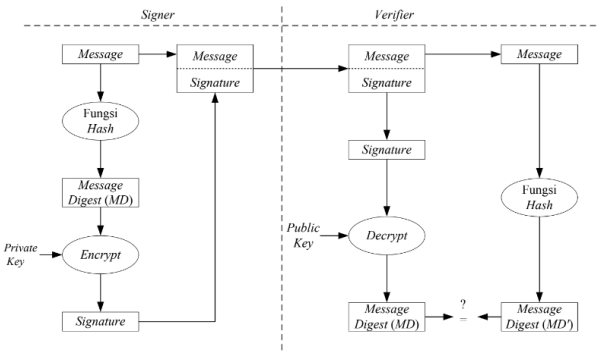

Fig. 1. Skema tanda tangan digital dengan kombinasi fungsi *hash* dan kriptografi kunci-publik.
Sumber: Slide Kuliah II4031 Kriptografi dan Koding - Rinaldi Munir

Dalam penggunaan Digital Signature Algorithm (DSA) dengan pemanfaatan algoritma RSA dan fungsi SHA-1, berikut adalah langkah-langkah penandatanganan digital yang harus dilakukan.

1) Pengirim pesan membangkitkan pasangan kunci publik K dan kunci privat R dengan algoritma RSA.
2) Pengirim membubuhkan tanda tangan digitalnya pada pesan (*Signing*) dengan langkah berikut.
   a) Menghitung nilai hash dari pesan M.
   $$h = H(M)$$
   b) Mengenkripsi pesan M menggunakan kunci privat R milik pengirim untuk menghasilkan *signature* S.
   $$S = h^R \bmod n$$
   c) Membubuhkan *signature* S pada pesan M dan mengirimkan pesan tersebut.
3) Penerima pesan memverifikasi tanda tangan digital dengan langkah berikut.
   a) Memisahkan pesan M dan *signature* S.
   b) Mendekripsi *signature* S dengan menggunakan kunci publik K milik pengirim pesan.
   $$h' = S^K \bmod n$$
   c) Menghitung nilai hash dari pesan M.
   $$h = H(M)$$
   d) Membandingkan nilai h' dengan h yang didapatkan dari tanda tangan digital dan hasil hash pesan M. Jika sama, maka verifikasi berhasil dan terbukti bahwa orang yang mengirimkan benar orang tersebut serta konten dari pesan itu masih asli.

B. Algoritma ElGamal

Algoritma ElGamal merupakan salah satu algoritma kriptografi yang dibuat oleh Taher ElGamal pada tahun 1985 bersama dengan skema tanda tangan digital ElGamal. Keamanan dari algoritma ini didasarkan pada tingkat kesulitan dalam komputasi perhitungan logaritma diskrit.

Algoritma ini terbagi menjadi tiga tahap, yaitu pembangkitan kunci, enkripsi dan dekripsi. Berikut adalah tahapan dalam proses pembangkitan kunci dengan menggunakan algoritma ElGamal.

1) Pemilihan suatu bilangan prima sembarang *p*.
2) Pemilihan dua bilangan acak, *g* dan *x*, dengan syarat berikut.
   $$g < p \text{ dan } 1 \leq x \leq (p-2)$$
3) Penghitungan nilai y dengan ketentuan berikut.
   $$y = g^x \bmod p$$
4) Hasil dari proses ini adalah.
   a) Kunci publik
   $$tripel\ (y, g, p)$$
   b) Kunci privat
   $$pasangan\ (x, p)$$

Setelah kunci telah berhasil dibangkitkan, dilanjutkan dengan proses enkripsi, dengan langkah-langkah seperti berikut.

1) Penyusunan dan pemisahan *plaintext* menjadi *n* blok dengan ketentuan *n* berada di interval $[0, (p-1)]$.
2) Pemilihan bilangan acak *k* dengan syarat berikut.
   $$1 \leq k \leq (p-2)$$
3) Pengenkripsian setiap blok hasil dari langkah 1 dengan rumus berikut.
   $$a = g^k \bmod p$$
   $$b = y^k m \bmod p$$
4) Hasil enkripsi tersebut terdiri atas *a* dan *b*, yaitu *ciphertext* untuk pesan *m,* sehingga ukuran *ciphertext* akan menjadi dua kali dari ukuran *plaintext*-nya.

Pesan yang sudah dienkripsi kemudian akan didekripsi dengan cara seperti berikut.

1) Penggunaan kunci privat *x* untuk menghitung
   $$(a^x)^{-1} = a^{p-1-x} \bmod p$$
2) Penghitungan *plaintext* dengan *m* dengan persamaan
   $$m = \frac{b}{a^x} \bmod p = b(a^x)^{-1} \bmod p$$

C. ElGamal Signature

Tanda tangan ElGamal atau *ElGamal Signature* adalah skema tanda tangan digital yang dibuat oleh Taher ElGamal pada tahun 1985. Sama seperti algoritma ElGamal, keamanannya juga didasarkan pada tingkat kesulitan dalam komputasi perhitungan logaritma diskrit.

Tanda tangan ElGamal terdiri atas tiga tahap, yaitu pembangkitan kunci, pembubuhan tanda tangan (*Signing*), dan verifikasi (*Verification*).

Tahap pembangkitan kunci pada tanda tangan ElGamal prosesnya sama dengan algoritma ElGamal biasa dan dilakukan oleh pengirim cukup sekali saja.

Pada tahap pembubuhan tanda tangan (*Signing*) pada pesan *m* dengan kunci privat (*x,p*), dilakukan langkah-langkah seperti berikut.

1) Pemilihan bilangan k sembarang yang memenuhi kedua syarat berikut.
   a) $0 < k < (p-1)$
   b) $FPB(k, (p-1)) = 1$
2) Melakukan komputasi untuk *r* sebagai komponen dari tanda tangan digital dengan ketentuan berikut.
   $$r \cong g^k \bmod p$$

3) Melakukan komputasi untuk *s* sebagai komponen dari tanda tangan digital dengan ketentuan berikut.

$$s \cong (H(m)) - xr)k^{-1} (mod\ (p - 1))$$

4) Jika $s = 0$, ulang kembali dari langkah pertama. Jika tidak, maka pasangan *(r,s)* adalah tanda tangan digital yang dihasilkan dan pengirim dapat mengulang proses ini pada setiap dokumen yang ingin dibubuhi tanda tangan digital.

Untuk melakukan verifikasi terhadap tanda tangan (*r,s*) pada sebuah pesan *m* dengan kunci publik (*p,g,y*), maka dilakukan langkah-langkah berikut.

1) Memastikan kedua syarat berikut dipenuhi.

$$0 < r < p \text{ dan } 0 < s < (p - 1)$$

2) Menghitung persamaan berikut dan memastikan bahwa syaratnya terpenuhi.

$$g^{(H(m))} \cong y^r r^s\ (mod\ p)$$

3) Jika kondisi pada langkah 1 dan 2 semuanya terpenuhi, maka tanda tangan terverifikasi dan terbukti bahwa orang yang mengirimkan benar orang tersebut serta konten dari pesan itu masih asli.

D. Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) merupakan salah satu teknik kriptografi kunci-publik yang dikembangkan oleh Neal Koblitz dan Victor S. Miller pada tahun 1985. ECC dibuat sebagai alternatif dari kriptografi kunci-publik yang menggunakan bilangan bulat seperti RSA, ElGamal, dan Diffie-Hellman yang membutuhkan tenaga komputasi yang besar dan waktu komputasi yang cukup lama dalam penyimpanan dan pemrosesan kunci serta pesan. Keamanan pada ECC didasarkan pada kesulitan pemecahan permasalahan matematis kurva eliptik (*elliptic curve*) atau yang lebih sering disebut dengan *Elliptic Curve Discrete Logarithm Problem* (*EDCLP*). ECC menawarkan keamanan yang relatif sama dengan algoritma-algoritma tersebut dengan ukuran kunci yang lebih kecil sehingga mengurangi tenaga komputasi dan waktu komputasi yang dibutuhkan, serta menghemat penyimpanan dan *bandwidth*.

Kurva eliptik adalah kurva dengan bentuk umum persamaan seperti berikut.

$$y^2 = x^3 + ax + b$$

dengan syarat

$$4a^3 + 27b^2 \neq 0$$

dan terdefinisi untuk $x, y \in R$.

Dalam ECC, kedua pihak yang akan berkomunikasi harus menyepakati parameter data sebagai berikut.

1) Persamaan kurva eliptik $y^2 = x^3 + ax + b\ mod\ p$.
2) Himpunan titik-titik yang dihitung dari persamaan kurva eliptik.
3) Titik basis (*base point*) $B(x_B, y_B)$, yang dipilih dari grup eliptik untuk operasi kriptografi.

Pembangkitan kunci pada ECC dilakukan sedemikian rupa sehingga syarat berikut terpenuhi.

1) Kunci privat (*x*)
   Kunci privat dipilih dari internal $[0, (p - 1)]$
2) Kunci publik (*Q*)
   Kunci publik adalah hasil kali antara *x* dengan titik basis B.

$$Q = x \cdot B = (x_1, y_2)$$

E. Elliptic Curve-Digital Signature Algorithm (ECDSA)

Elliptic Curve-Digital Signature Algorithm (ECDSA) merupakan algoritma tanda tangan digital yang memanfaatkan kunci dari kriptografi kurva eliptik (ECC). Saat ini, ECDS banyak dipakai dalam sistem keamanan aplikasi *messenger* dan juga merupakan dasar dari keamanan Bitcoin.

Berikut adalah visualisasi proses yang terlibat dalam ECDSA.

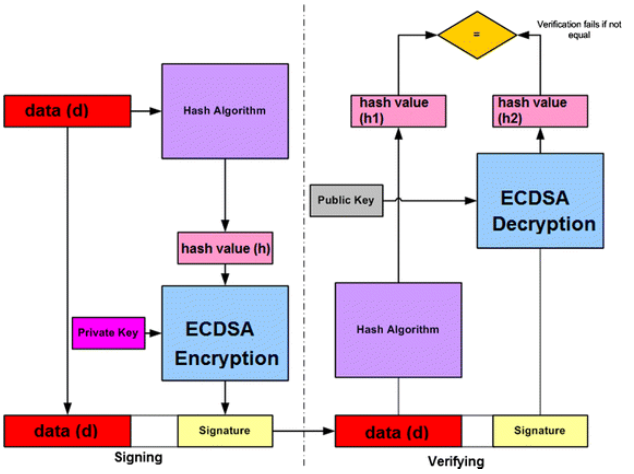

Fig. 2. Skema tanda tangan digital dengan algoritma ECDSA
Sumber: https://www.researchgate.net/figure/Signing-and-verification-phases-of-ECDSA_fig1_316174537

Pembangkitan kunci pada ECDSA memanfaatkan ECC sehingga prosesnya sama dan pengirim hanya perlu melakukan pembangkitan kunci ini satu kali saja.

Untuk proses pembubuhan tanda tangan (*Signing*) untuk pesan *m* dengan fungsi hash *H*, dilakukan langkah-langkah berikut.

1) Pemilihan bilangan bulat secara sembarang k yang berada di interval $[1, (p - 1)]$.
2) Penghitungan x_1 dengan menghitung $kB = (x_1, y_2)$
3) Penghitungan $r = x_1\ mod\ p$. Jika $r = 0$, maka kembali ke langkah 1.
4) Penghitungan $k^{-1}\ mod\ p$
5) Penghitungan *digest hash* dengan persamaan berikut.

$$e = H(m)$$

6) Penghitungan tanda tangan digital *s* dengan persamaan berikut.

$$s = k^{-1}(e + x \cdot r) \bmod p$$

7) Tanda tangan yang dihasilkan adalah pasangan (r, s)

Untuk melakukan verifikasi terhadap tanda tangan (*r,s*) pada sebuah pesan *m* dengan kunci publik (*Q*), maka dilakukan langkah-langkah berikut.

1) Verifikasi bahwa r dan s adalah bilangan bulat pada interval $[1, (p - 1)]$.
2) Penghitungan *digest hash* dengan persamaan berikut.

$$e = H(m)$$

3) Penghitungan $w = k^{-1} \bmod p$
4) Penghitungan $u_1 = ew \bmod p$ dan $u_2 = rw \bmod p$
5) Penghitungan x_1 yang memenuhi persamaan berikut.

$$u_1 \cdot B + u_2 \cdot Q = (x_1, y_1)$$

Jika $x_1 = (0, 0)$, maka tanda tangan ditolak. Jika tidak, maka akan dihitung $v = x_1 \bmod p$ dan koordinat x_1 akan diubah menjadi integer.

6) Verifikasi apakah Tanda tangan akan diterima jika dan hanya jika $v = r$.

III. Rancangan Program

Analisis terhadap algoritma *ElGamal Signature* dan *Elliptic Curve Digital Sign Algorithm (ECDSA)* akan dilakukan dengan membandingkan waktu eksekusi dari kedua algoritma tersebut. Penulis menggunakan bahasa pemrograman Python dalam membuat program untuk membandingkan kinerja kedua algoritma tersebut. Bahasa Python digunakan karena memiliki *library* kriptografi yang cukup luas dan relatif mudah untuk dimengerti kodenya.

Berikut adalah kesamaan spesifikasi program tanda tangan digital untuk implementasi kedua algoritma yang diuji.

1) Dokumen yang diuji dapat memiliki format (.txt), (.jpg), (.mp3), maupun (.mp4).
2) Pasangan kunci publik dan kunci privat yang digunakan selama pengujian dibangkitkan sebelum proses *signing* dan *verification*, namun pada satu *runtime* yang sama.
3) Program tidak memiliki GUI.
4) Program dapat mengukur *execution time* setiap proses tanda tangan digital, termasuk pembangkitan kunci (*key generation*), pembubuhan tanda tangan (*signing*), dan juga verifikasi (*verification*).

Program akan dijalankan pada perangkat keras yang dimiliki penulis, yaitu dengan spesifikasi seperti berikut.

1) *Operating System* : Windows 10
2) *Processor* : AMD Ryzen 5 4000 Series
3) RAM : 8 GB
4) *System type* : 64-bit *operating system*, x64-*based processor*

IV. Implementasi dan Pengujian

A. Implementasi

Kedua program tanda tangan digital diimplementasikan dengan menggunakan Bahasa Python pada dua *file* yang berbeda. Berikut adalah kode programnya dari implementasi ECDSA.

```
import time
import hashlib
from ecdsa import SigningKey, SECP256k1

# Membangkitkan pasangan kunci privat
dan publik
def generate_key_pair():
    start_time = time.time()
    sk =
SigningKey.generate(curve=SECP256k1)
    vk = sk.get_verifying_key()
    end_time = time.time()
    ex_time = end_time - start_time
    return sk, vk, ex_time

# Membubuhkan tanda tangan digital pada
pesan
def sign_message(sk, message):
    start_time = time.time()
    signature = sk.sign(message)
    end_time = time.time()
    ex_time = end_time - start_time
    return signature, ex_time

# Melakukan verifikasi tanda tangan
digital dari pesan
def verify_signature(vk, message,
signature):
    start_time = time.time()
    result=False
    try:
        vk.verify(signature, message)
        result = True
    except:
        result = False
    end_time = time.time()
    ex_time = end_time - start_time
    return result, ex_time

# Fungsi untuk membaca file
def read_file(file_path):
    with open(file_path, 'rb') as file:
        content = file.read()
    return content

# Main program
private_key, public_key, keygen_time =
generate_key_pair()
file_path = 'example.txt'
message = read_file(file_path)
signature, sign_time =
sign_message(private_key,
hashlib.sha256(message).digest())

verification_result, verify_time =
verify_signature(public_key,
```

```
hashlib.sha256(message).digest(),
signature)

print("Private Key : ",
str(private_key.to_pem()))
print("Public Key : ",
str(public_key.to_pem()))
print("Verification result : ",
verification_result)
print("Key Generation time : ",
keygen_time, "seconds")
print("Signing time : ", sign_time,
"seconds")
print("Verification time : ",
verify_time, "seconds")
```

Berikut adalah kode programnya dari implementasi algoritma *ElGamal Signature*.

```
import random
import timeit
import hashlib

# Extended Euclidean Algorithm
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
    else:
        gcd, x, y = extended_gcd(b, a %
b)
        return gcd, y, x - (a // b) * y

# Modular Multiplicative Inverse
def mod_inverse(a, m):
    gcd, x, _ = extended_gcd(a, m)
    if gcd == 1:
        return x % m
    else:
        raise ValueError("Modular
inverse does not exist.")

# Pembangkitan Kunci ElGamal
def elgamal_keygen():
    # Membangkitkan bilangan prima besar
    prime = 2
    while pow(2, prime - 1, prime) != 1
or pow(3, prime - 1, prime) != 1:
        prime = random.getrandbits(16)

    # Membangkitkan private key
sembarang
    private_key = random.randint(1,
prime - 2)

    # Hitung public key
    generator = random.randint(1, prime
- 1)
    public_key = pow(generator,
private_key, prime)

    return prime, generator, public_key,
private_key
```

```
# Membubuhkan tanda tangan ElGamal
def elgamal_sign(message, prime,
generator, private_key):
    # Pilih ephemeral key sembarang
    k = random.randint(2, prime - 2)

    # Hitung ephemeral public key
    ephemeral_key = pow(generator, k,
prime)

    # Komputasi hash dari pesan
    message_hash =
int.from_bytes(hashlib.sha256(message).d
igest(), 'big')

    # Komputasi komponen tanda tangan
digital
    r = pow(generator, k, prime)
    s = (mod_inverse(k, prime - 1) *
(message_hash - private_key * r)) %
(prime - 1)

    return (r, s)

# Verifikasi tanda tangan ElGamal
def elgamal_verify(message, signature,
prime, generator, public_key):
    r, s = signature

    # Komputasi hash dari pesan
    message_hash =
int.from_bytes(hashlib.sha256(message).d
igest(), 'big')

    # Verifikasi tanda tangan
    left = (pow(public_key, r, prime) *
pow(r, s, prime)) % prime
    right = pow(generator, message_hash,
prime)

    return left == right

# Fungsi untuk membaca file
def read_file(file_path):
    with open(file_path, 'rb') as file:
        content = file.read()
    return content

# Main program
file_path = 'example.txt'
message = read_file(file_path)

keygen_time = timeit.timeit(lambda:
elgamal_keygen(), number=1)
prime, generator, public_key,
private_key = elgamal_keygen()
print("Public Key:", public_key)
print("Private Key:", private_key)
print("Key Generation Time:",
keygen_time, "seconds")

sign_time = timeit.timeit(lambda:
elgamal_sign(message, prime, generator,
```

```
private_key), number=1)
signature = elgamal_sign(message, prime,
generator, private_key)
print("Signature:", signature)
print("Signature Generation Time:",
sign_time, "seconds")

verify_time = timeit.timeit(lambda:
elgamal_verify(message, signature,
prime, generator, public_key), number=1)
is_valid = elgamal_verify(message,
signature, prime, generator, public_key)
print("Signature Verification:",
is_valid)
print("Verification Time:", sign_time,
"seconds")
```

B. Pengujian

Dilakukan percobaan pemberian tanda tangan digital terhadap sebuah pesan file example.txt dengan isi pesan sebagai berikut.

```
Test 123 ENHYPEN COMEBACK 22 MAY 2023
- STREAM BITE ME!
```

Pengujian dilakukan sebanyak lima kali pada kedua algoritma tersebut dengan merekam waktu eksekusi kode dari 3 bagian utama dalam tanda tangan digital, yaitu pembangkitan kunci (*Key Generation*), pembubuhan tanda tangan digital (*Signing*), dan verifikasi tanda tangan digital (*Verification*). Hasil percobaan dirangkum pada kedua tabel berikut dengan satuan detik.

TABLE I. HASIL PERCOBAAN ALGORITMA ECDSA

Percobaan ke-	***Key Generation***	***Signing***	***Verification***
1	0.008994102478	0.001004695892	0.002006530762
2	0.008003234863	0	0.004006385803
3	0.008995294571	0.001002550125	0.002001285553
4	0.008988380432	0.0009968280792	0.002999067307
5	0.008991241455	0.00200843811	0.002004384995
Rata-Rata	0.00879445076	0.001002502441	0.002603530884

TABLE II. HASIL PERCOBAAN ALGORITMA ELGAMAL

Percobaan ke	***Key Generation***	***Signing***	***Verification***
1	0.000030399999999 9998	0.0000486000000 000028	0.00004860000000 00028
2	0.000014699999999 9994	0.0000611000000 000014	0.00006110000000 00014
3	0.000015599999999	0.0000584999999	0.00005849999999
4	0.000040399999999 9959	0.0000684999999 999991	0.00006849999999 99991
5	0.000015599999999 9975	0.0000536999999 999968	0.00005369999999 99968
Rata-Rata	0.000023339999999 9980	0.0000580799999 999992	0.00005807999999 99992

Pada percobaan kedua yang dilakukan dengan algoritma ECDSA, *signing execution time* yang tercatat adalah 0 dikarenakan nilai yang seharusnya tercatat terlalu kecil sehingga terhalang oleh keterbatasan *library* Python yang digunakan pada kode program tersebut. Pada kode program untuk algoritma ElGamal digunakan *library* yang berbeda sehingga dapat merekam *execution time* yang sangat kecil tersebut.

Untuk memudahkan perbandingan antara kedua algoritma yang diteliti, berikut disajikan visualisasi data dari data yang terekam pada percobaan yang dilakukan.

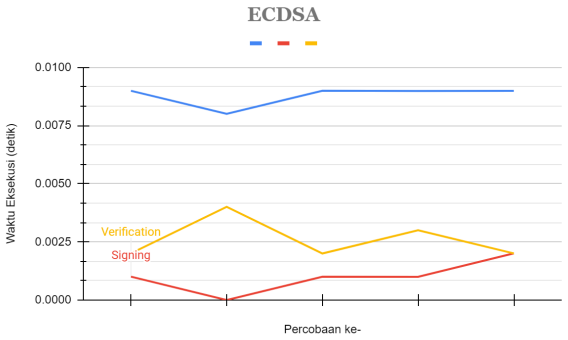

Fig. 3. Visualisasi hasil percobaan tanda tangan digital dengan algoritma ECDSA
Sumber: pustaka penulis

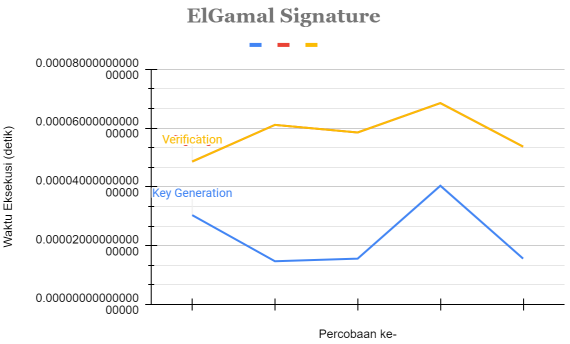

Fig. 4. Visualisasi hasil percobaan tanda tangan digital dengan algoritma ElGamal Signature
Sumber: pustaka penulis

Berikut juga disajikan visualisasi data hasil percobaan dengan membandingkan hasil percobaan kedua algoritma yang diteliti.

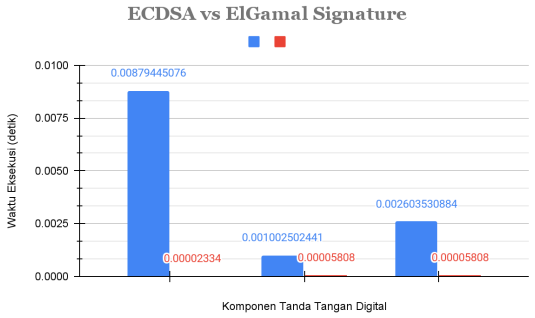

Fig. 5. Visualisasi perbandingan hasil percobaan tanda tangan digital dengan algoritma ECDSA dan ElGamal Signature
Sumber: pustaka penulis

Dari hasil percobaan yang dilakukan dengan asumsi dan keterbatasannya, dapat disimpulkan bahwa algoritma *ElGamal Signature* memiliki *execution time* yang jauh lebih cepat dibandingkan algoritma ECDSA. Namun, hasil ini masih mungkin terpengaruh dengan beberapa faktor, termasuk penggunaan *library* pada program implementasi ECDSA namun pada program implementasi *ElGamal Signature* tidak digunakan.

Satu faktor yang perlu diperhatikan juga adalah implementasi algoritma *ElGamal Signature* yang digunakan pada program memiliki banyak ruang untuk dikembangkan lebih lanjut karena tingkat keberhasilan program dalam melakukan tiga proses tanda tangan digital cukup rendah. Hal ini berarti seringkali ketika program melakukan komputasi muncul *error* yang menyebabkan program berhenti dieksekusi. Sehingga dapat dikatakan bahwa walaupun *ElGamal Signature* lebih cepat dieksekusi, ECDSA lebih stabil dalam eksekusinya dengan artian bahwa tingkat kegagalan dalam prosesnya sangat rendah.

V. Kesimpulan dan Saran

Tanda tangan digital harus dijaga keamanannya, namun juga perlu untuk diperhatikan performanya. Terdapat beberapa alternatif algoritma yang dapat digunakan untuk penyematan tanda tangan digital, dua diantaranya adalah ECDSA dan *ElGamal Signature.* Berdasarkan percobaan yang digunakan, didapatkan waktu eksekusi ketiga komponen tanda tangan digital pada algoritma *ElGamal Signature* lebih cepat dieksekusi oleh komputer, namun tingkat keberhasilan proses pada ECDSA lebih tinggi dibandingkan *ElGamal Signature.* Dalam percobaan yang dilakukan ECDSA selalu berhasil dieksekusi dengan lima percobaan berturut-turut berhasil dilakukan. Sedangkan data yang diambil untuk lima percobaan berhasil pada percobaan algoritma *ElGamal Signature* dilakukan dalam dua puluh lima kali percobaan secara total.

Saran untuk pengembang berikutnya adalah untuk memperbaiki program implementasi kedua algoritma agar dapat lebih seimbang dan penggunaan variabel kontrol yang lebih baik. Salah satu hal yang dapat diperbaiki dari program implementasi algoritma *ElGamal Signature* adalah dengan menggunakan *exception handling* yang lebih baik.

Ucapan Terima Kasih

Puji dan syukur kepada Tuhan Yang Maha Esa, atas berkat dan rahmat-Nya makalah ini dapat diselesaikan dengan baik. Terima kasih untuk keluarga saya yang senantiasa memberikan dukungan kepada saya. Terima kasih juga kepada Dr. Ir. Rinaldi Munir, M.T. yang telah membagikan pengetahuan dan pengalamannya melalui mata kuliah II4031 Kriptografi dan Koding selama semester genap tahun ajaran 2022/2023 dengan cara yang mudah dipahami. Kepada asisten dosen pada mata kuliah ini, terima kasih sudah suportif dan membantu kegiatan perkuliahan. Tidak lupa juga terima kasih pada rekan sekelas yang saling membantu dalam pembuatan makalah ini, kakak tingkat yang telah memberikan referensi, serta penulis-penulis yang menyediakan referensi berupa buku, jurnal, maupun blog di internet.

Referensi

[1] Munir, Rinaldi. Slide Kuliah II3041 Kriptografi dan Koding: Kriptografi Kunci-Publik (Public-key Cryptography) [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/09%20-%20Kriptografi-Kunci-Publik-2023.pdf

[2] Munir, Rinaldi. Slide Kuliah II3041 Kriptografi dan Koding: Algoritma ElGamal [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/Algoritma-ElGamal-(2018).pdf

[3] Munir, Rinaldi. Slide Kuliah II3041 Kriptografi dan Koding: Tanda Tangan Digital [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/16-Tanda-tangan-digital-2023.pdf

[4] Munir, Rinaldi. Slide Kuliah II3041 Kriptografi dan Koding: Elliptic Curve Cryptography (ECC) [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/12-ECC-2023.pdf

[5] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, July 1985, doi: 10.1109/TIT.1985.1057074.

[6] D. Johnson, A. Menezes, and S. Vanstone, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” International Journal of Information Security, vol. 1, no. 1, pp. 36–63, Aug. 2001, doi: https://doi.org/10.1007/s102070100002.

[7] A. Triwinarko, Elliptic Curve Digital Signature Algorithm (ECDSA) Departemen Teknik Informatika ITB [Online]. Available:

https://informatika.stei.itb.ac.id/~rinaldi.munir/TA/Makalah_TA%20Andy%20T.pdf. [Accessed: 22-May-2023]

PERNYATAAN

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

Bandung, 22 Mei 2023

Gresya Angelina Eunike Leman
18220104