Analisis Perbandingan Kinerja dan Efektivitas *Digital Signature Algorithm* (DSA) dan *Elliptic Curve-Digital Signature Algorithm* (ECDSA) untuk Penandatanganan Sertifikat Digital

Khafifanisa - 18220056
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: khffnsa52@gmail.com

***Abstract*—Makalah ini bertujuan untuk menganalisis dan membandingkan kinerja dan efektivitas dua algoritma tanda tangan digital, yaitu *Digital Signature Algorithm* (DSA) dan *Elliptic Curve Digital Signature Algorithm* (ECDSA) dalam penandatanganan sertifikat digital. Pengujian yang dilakukan adalah melakukan perbandingkan waktu pembangkitan kunci, waktu pemberian tanda tangan dan waktu verifikasi tanda tangan pada sertifikat digital. Hasil penelitian menunjukkan ECDSA cenderung memiliki kinerja yang lebih baik dibandingkan dengan DSA dilihat dari lebih sedikitnya waktu yang digunakan ECDSA.**

Keywords—RSA; ECDSA; SHA-256; tanda tangan digital; sertifikat digital; kinerja

I. PENDAHULUAN

Sertifikat digital merupakan salah satu komponen dari *Public Key Infrastructure* (PKI) untuk menjaga keamanan komunikasi online. Salah satu hal yang dapat menjaga keamanan dari sertifikat digital adalah dengan pemberian tanda tangan pada sertifikat digital tersebut [1].

Terdapat beberapa algoritma kriptografi yang dapat digunakan untuk pembangkitan kunci publik dan penandatangan sertifikat digital, diantaranya adalah *digital signature algorithm* (DSA) dan *elliptic curve digital signature* (ECDSA).

Penelitian ini dimaksudkan untuk mengkaji perbandingan kinerja dan efektivitas algoritma pada DSA dan ECDSA mulai dari waktu pembangkitan pasangan kunci publik dan kunci privat, waktu penandatangan sertifikat digital, waktu verifikasi tanda tangan sertifikat digital, serta ukuran tanda tangan yang dihasilkan. Skenario pengujian akan dilakukan pada masing-masing 3 kunci dengan panjang yang berbeda untuk DSA dan ECDSA. Untuk ECDSA akan diuji pada kunci dengan panjang 256 bit. Sementara pada DSA akan diuji untuk panjang kunci 1024 bit.

Penelitian ini dilakukan dengan metode eksperimental yang melibatkan implementasi kode algoritma untuk membangkitkan kunci, penandatanganan dan verifikasi tanda tangan, dan pengujian dengan membandingkan waktu dan ukuran tanda tangan yang dihasilkan. Adapun batasan dari penelitian ini terbatas pada tanda tangan untuk sertifikat digital dengan panjang kunci yang telah ditentukan sebelumnya.

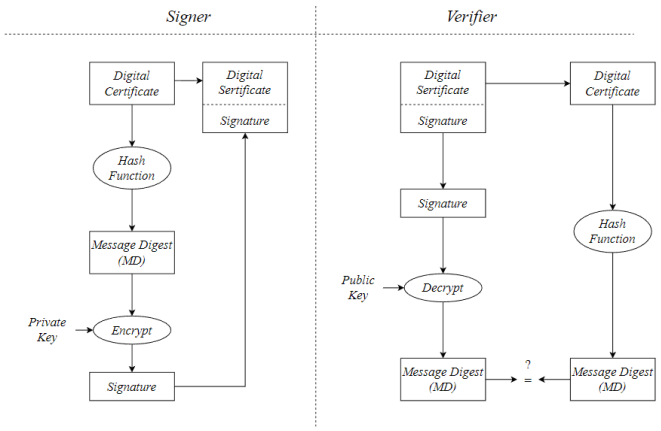
II. TEORI DASAR

A. Sertifikat Digital

Sertifikat digital merupakan dokumen yang menyatakan kepemilikan seseorang atau suatu badan terhadap kunci publik yang dimilikinya. Sertifikat dikeluarkan oleh suatu badan atau institusi yang memiliki kewenangan untuk menerbitkannya (*issued*). Badan ini disebut dengan *Certification Authority* (CA). Melalui penerbitan sertifikat digital oleh CA, sertifikat digital ini akan ditandatangani menggunakan kunci privat CA [2].

Sertifikat digital menjadi penting karena kunci publik tidak bersifat rahasia dan dapat diketahui oleh semua orang sehingga perlu disertifikasi agar kunci publik tidak disalahgunakan secara tidak bertanggung jawab [2].

Berikut adalah alur penandatangan dan verifikasi tanda tangan untuk sertifikat digital:



Gbr 1. Alur *signing* - *verify* tanda tangan pada sertifikat digital
Sumber: II4031 Kriptografi dan Koding - Tanda Tangan Digital - Rinaldi Munir

Berdasarkan diagram alur di atas, proses penandatanganan dilakukan dengan mengkombinasikan algoritma kunci publik dengan fungsi *hash*. Informasi dalam sertifikat digital diubah dengan fungsi *hash* menjadi *Message Digest*. Kemudian, dengan menggunakan *private key* milik CA, *message digest* akan dienkripsi. Hasil enkripsi inilah yang akan menjadi tanda tangan [3].

Untuk melakukan verifikasi, tanda tangan pada sertifikat akan kembali didekripsi menggunakan kunci publik miliki CA menjadi *message digest*. Sementara itu, informasi pada sertifikat digital juga diubah menggunakan fungsi *hash* menjadi *message digest*. Apabila *message digest* dari informasi sertifikat digital dan hasil dekripsi tanda tangan memiliki nilai yang sama, maka tanda tangan terverifikasi valid [3].

B. Digital Signature Algorithm *(DSA)*

DSA (*Digital Signature Algorithm*) merupakan algoritma kriptografi kunci publik untuk tanda tangan digital yang populer. Algoritma pada DSA menggunakan matematika dasar seperti operasi modulo dan operasi eksponensial [4].

1. Bilangan prima

   Pada DSA, dibutuhkan dua bilangan prima, yakni bilangan prima p dan bilangan prima q untuk pembangkitan kunci. Bilangan prima q yang dipilih merupakan bilangan prima yang sangat besar untuk meningkatkan keamanan, sedangkan bilangan prima p harus memenuhi q adalah faktor dari p-1 [5].

2. Grup multiplicative modulo p

   Selain bilangan prima p dan q, pada DSA juga dibutuhkan grup bilangan bulat positif kurang dari p yang saling relatif prima dengan p [4].

C. Elliptic Curve Digital Signature *(ECDSA)*

Elliptic Curve Digital Signature (ECDSA) adalah algoritma kriptografi untuk membuat tanda tangan digital yang menggunakan teori kurva eliptik sebagai dasar algoritmanya [6]. Terdapat 3 proses utama dalam ECDSA, yaitu pembangkitan kunci publik dan kunci privat, *signing*, dan verifikasi tanda tangan. ECDSA dianggap tahan terhadap serangan dan memiliki ukuran kunci yang lebih pendek dibandingkan dengan kriptografi kunci publik konvensional [7].

1. Kurva eliptik (*eliptic curve*)

   Kurva eliptik merupakan kurva eliptik yang didefinisikan oleh persamaan matematika $y^2 = x^3 + ax + b$, dengan a dan b adalah konstanta, dan x, y adalah koordinat pada kurva [8].

2. Grup kurva eliptik

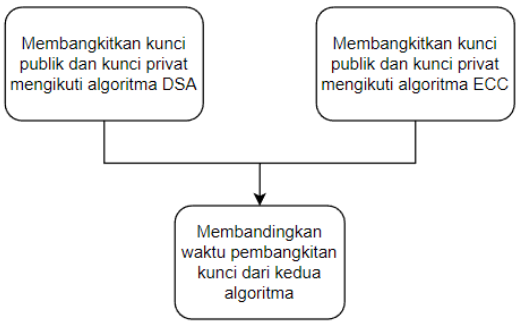
   Grup kurva eliptik adalah kurva eliptik yang membentuk grup abelian atas titik-titiknya [8].

III. PEMBAHASAN

A. Rancangan Implementasi dan Pengujian

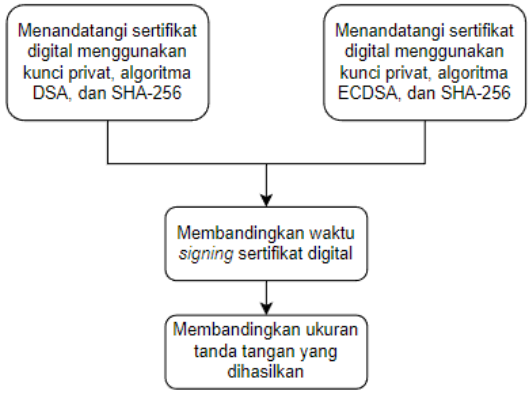
Terdapat 3 rancangan implementasi dan pengujian yang akan dilakukan untuk membandingkan kinerja dan efektivitas *digital signature algorithm* (DSA) dan *elliptic curve digital signature* (ECDSA), yaitu:

1. Pembangkitan Kunci
   Berikut adalah rancangan alur implementasi pembangkitan kunci dan pengujian waktu pembangkitan kunci.



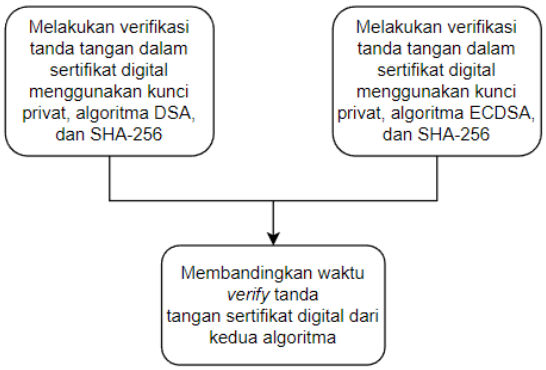
Gbr 2. Alur implementasi dan pengujian pembangkitan kunci

2. *Signing*
   Berikut adalah rancangan alur implementasi proses *signing* sertifikat digital dan pengujian perbandingan waktu dan ukuran tanda tangan.



Gbr 3. Alur implementasi dan pengujian proses *signing*

3. *Verify*
   Berikut adalah rancangan alur implementasi proses verifikasi tanda tangan pada sertifikat digital dan pengujian perbandingan waktu verifikasi.



Gbr 4. Alur implementasi dan pengujian proses *verify*

B. Implementasi

Implementasi dilakukan menggunakan bahasa pemrograman *python* dan menggunakan *library cryptography* yang sudah tersedia pada bahasa pemrograman *python*.

Untuk DSA, akan menghasilkan kunci berukuran 1024 bit, dan ECDSA menghasilkan kunci berukuran 256 bit. Hal ini disesuaikan dengan panjang kunci yang disediakan oleh *library cryptography*.

1. Pembangkitan Kunci
   Berikut ini adalah implementasi kode program untuk pembangkitan kunci.

   Pembangkitan kunci dengan DSA, panjang kunci 1024

```python
from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.primitives import serialization
import time


# Fungsi untuk generate key
def generate_dsa_1024_keypair(file_name):
    private_key = dsa.generate_private_key(key_size=1024)
    public_key = private_key.public_key()


    private_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    public_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKe
yInfo
    )


    # Simpan key
    with open(f"{file_name}_dsa_1024.pri", "wb") as
pri_file:
        pri_file.write(private_pem)
    with open(f"{file_name}_dsa_1024.pub", "wb") as
pub_file:
        pub_file.write(public_pem)
    print("Kunci publik dan kunci privat DSA 1024
berhasil disimpan.")


# Test
file_name = input("Masukkan nama file: ")


start_time = time.time()
generate_dsa_1024_keypair(file_name)
end_time = time.time()


execution_time = end_time - start_time
print(f"Waktu    pembangkitan    kunci:    {execution_time}
detik")
```

   Pembangkitan kunci dengan ECDSA, panjang kunci 256

```python
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives import serialization
import time


# Fungsi untuk generate key
def generate_ecdsa_p256_keypair(file_name):
    curve = ec.SECP256R1()
    private_key = ec.generate_private_key(curve)
    public_key = private_key.public_key()


    private_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    public_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKe
yInfo
    )


    # Simpan key
```

```
    with open(f"{file_name}_ecdsa_p256.pri", "wb") as
pri_file:
        pri_file.write(private_pem)
    with open(f"{file_name}_ecdsa_p256.pub", "wb") as
pub_file:
        pub_file.write(public_pem)
    print("Kunci publik dan kunci privat ECDSA P-256
berhasil disimpan.")



# Test
file_name = input("Masukkan nama file: ")

start_time = time.time()
generate_ecdsa_p256_keypair(file_name)
end_time = time.time()

execution_time = end_time - start_time
print(f"Waktu    pembangkitan    kunci:    {execution_time}
detik")
```

2. *Signing*
   Berikut ini adalah implementasi kode program untuk proses *signing*.

   Proses *signing* dengan DSA

```
import time
import sys
import os
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.primitives.asymmetric.utils
import decode_dss_signature
from cryptography.hazmat.primitives import serialization

# Fungsi untuk signing dengan DSA
def sign_certificate_with_dsa(private_key_filename,
certificate_filename, signature_filename):
    private_key_path = f"{private_key_filename}.pri"
    certificate_path = f"{certificate_filename}.txt"

    # Membaca file private key
    with open(private_key_path, "rb") as key_file:
        private_key = serialization.load_pem_private_key(
            key_file.read(),
            password=None
        )

    # Membaca file sertifikat
    with open(certificate_path, "rb") as cert_file:
        certificate = cert_file.read().strip()

    # Menghitung waktu untuk signing
    start_time = time.time()

    # Menandatangani sertifikat
    signature = private_key.sign(
        certificate,
        hashes.SHA256()
    )

    end_time = time.time()
    signing_time = end_time - start_time

    # Enkripsi tanda tangan
    r, s = decode_dss_signature(signature)
    encoded_signature = r.to_bytes(48, 'big') +
s.to_bytes(48, 'big')

    # Simpan tanda tangan
    signature_file_path = f"{signature_filename}.txt"
    with open(signature_file_path, "w") as
signature_file:
        signature_file.write(encoded_signature.hex())

    print(f"Tanda tangan berhasil disimpan di file:
{signature_file_path}")
    print(f"Waktu signing: {signing_time} detik")
    print(f"Ukuran tanda tangan:
{sys.getsizeof(signature)} bytes")

# Test
private_key_filename = input("Masukkan nama file kunci
privat: ")
certificate_filename = input("Masukkan nama file
sertifikat: ")
signature_filename = input("Masukkan nama file tanda
tangan: ")

sign_certificate_with_dsa(private_key_filename,
certificate_filename, signature_filename)
```

   Proses *signing* dengan ECDSA

```
import time
import sys
import os
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric.utils
import decode_dss_signature
from cryptography.hazmat.primitives import serialization

# Fungsi untuk signing dengan ECDSA
def sign_certificate_with_ecdsa(private_key_filename,
certificate_filename, signature_filename):
    private_key_path = f"{private_key_filename}.pri"
    certificate_path = f"{certificate_filename}.txt"

    # Membaca file private key
    with open(private_key_path, "rb") as key_file:
        private_key = serialization.load_pem_private_key(
            key_file.read(),
            password=None
        )

    # Membaca file sertifikat
    with open(certificate_path, "rb") as cert_file:
        certificate = cert_file.read().strip()

    # Menghitung waktu untuk signing
    start_time = time.time()

    # Menandatangani sertifikat
    signature = private_key.sign(
        certificate,
        ec.ECDSA(hashes.SHA256())
    )
```

```
    end_time = time.time()
    signing_time = end_time - start_time

    # Enkripsi tanda tangan
    r, s = decode_dss_signature(signature)
    encoded_signature = r.to_bytes(32, 'big') +
s.to_bytes(32, 'big')

    # Simpan tanda tangan
    signature_file_path = f"{signature_filename}.txt"
    with open(signature_file_path, "w") as
signature_file:
        signature_file.write(encoded_signature.hex())

    print(f"Tanda tangan berhasil disimpan di file:
{signature_file_path}")
    print(f"Waktu signing: {signing_time} detik")
    print(f"Ukuran tanda tangan:
{sys.getsizeof(signature)} bytes")



# Test
private_key_filename = input("Masukkan nama file kunci
privat: ")
certificate_filename = input("Masukkan nama file
sertifikat: ")
signature_filename = input("Masukkan nama file tanda
tangan: ")

sign_certificate_with_ecdsa(private_key_filename,
certificate_filename, signature_filename)
```

Untuk proses *signing*, file tanda tangan disimpan pada file berbeda untuk menghindari kesalahan saat verifikasi.

3. *Verify*
   Berikut ini adalah implementasi kode program untuk verifikasi tanda tangan digital pada sertifikat digital.

Proses verifikasi tanda tangan dengan DSA

```
import time
import sys
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.primitives.asymmetric.utils
import encode_dss_signature
from cryptography.hazmat.primitives import serialization

# Fungsi untuk verifikasi tanda tangan dengan DSA
def verify_signature_with_dsa(public_key_filename,
certificate_filename, signature_filename):
    public_key_path = f"{public_key_filename}.pub"
    certificate_path = f"{certificate_filename}.txt"
    signature_file_path = f"{signature_filename}.txt"

    # Membaca file public key
    with open(public_key_path, "rb") as key_file:
        public_key = serialization.load_pem_public_key(
            key_file.read()
        )
    # Membaca file sertifikat
    with open(certificate_path, "rb") as cert_file:
        certificate = cert_file.read().strip()

    # Membaca file tanda tangan
    with open(signature_file_path, "r") as
signature_file:
        encoded_signature = signature_file.read()

    # Menghitung waktu verifikasi tanda tangan
    start_time = time.time()

    # Dekripsi tanda tangan
    signature_bytes = bytes.fromhex(encoded_signature)
    r = int.from_bytes(signature_bytes[:48], 'big')
    s = int.from_bytes(signature_bytes[48:], 'big')
    signature = encode_dss_signature(r, s)

    try:
        # Verifikasi tanda tangan
        public_key.verify(
            signature,
            certificate,
            hashes.SHA256()
        )
        print("Tanda tangan valid.")
    except:
        print("Tanda tangan tidak valid.")

    end_time = time.time()
    verification_time = end_time - start_time

    print(f"Waktu verifikasi: {verification_time} detik")



# Test
public_key_filename = input("Masukkan nama file kunci
publik: ")
certificate_filename = input("Masukkan nama file
sertifikat: ")
signature_filename = input("Masukkan nama file tanda
tangan: ")

verify_signature_with_dsa(public_key_filename,
certificate_filename, signature_filename)
```

Proses verifikasi tanda tangan dengan ECDSA

```
import time
import sys
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric.utils
import encode_dss_signature
from cryptography.hazmat.primitives import serialization

# Fungsi untuk verifikasi tanda tangan dengan ECDSA
def verify_signature_with_ecdsa(public_key_filename,
certificate_filename, signature_filename):
    public_key_path = f"{public_key_filename}.pub"
    certificate_path = f"{certificate_filename}.txt"
    signature_path = f"{signature_filename}.txt"

    # Membaca file public key
    with open(public_key_path, "rb") as key_file:
        public_key = serialization.load_pem_public_key(
```

```
            key_file.read()
        )

    # Membaca file sertifikat
    with open(certificate_path, "rb") as cert_file:
        certificate = cert_file.read().strip()

    # Membaca file tanda tangan
    with open(signature_path, "r") as signature_file:
        encoded_signature =
bytes.fromhex(signature_file.read().strip())

    # Menghitung waktu verifikasi tanda tangan
    start_time = time.time()

    # Dekripsi tanda tangan
    r = int.from_bytes(encoded_signature[:32], 'big')
    s = int.from_bytes(encoded_signature[32:], 'big')
    signature = encode_dss_signature(r, s)



    # Verifikasi tanda tangan
    try:
        public_key.verify(
            signature,
            certificate,
            ec.ECDSA(hashes.SHA256())
        )
        print("Verifikasi tanda tangan berhasil: Valid")
    except:
        print("Verifikasi tanda tangan gagal: Tidak
valid")

    end_time = time.time()
    verification_time = end_time - start_time

    print(f"Waktu verifikasi: {verification_time} detik")



# Test
public_key_filename = input("Masukkan nama file kunci
publik: ")
certificate_filename = input("Masukkan nama file
sertifikat: ")
signature_filename = input("Masukkan nama file tanda
tangan: ")

verify_signature_with_ecdsa(public_key_filename,
certificate_filename, signature_filename)
```

C. Pengujian

Untuk pengujian kali ini, setiap algoritma akan diuji sebanyak 3 kali mulai dari pembangkitan kunci, tanda tangan, dan verifikasi tanda tangan.

Berikut ini adalah skenario pengujian yang dilakukan untuk membandingkan DSA dan ECDSA.

1. Pengujian waktu pembangkitan kunci

   Pengujian DSA

```
Masukkan nama file: key1
Kunci publik dan kunci privat DSA 1024 berhasil disimpan.
Waktu pembangkitan kunci: 0.4755704402923584 detik
```

```
Masukkan nama file: key2
Kunci publik dan kunci privat DSA 1024 berhasil disimpan.
Waktu pembangkitan kunci: 0.13568353652954102 detik
```

```
Masukkan nama file: key3
Kunci publik dan kunci privat DSA 1024 berhasil disimpan.
Waktu pembangkitan kunci: 0.183074951171875 detik
```

Rata-rata waktu pembangkitan kunci:
0.26477

Pengujian ECSA

```
Masukkan nama file: key10
Kunci publik dan kunci privat ECDSA P-256 berhasil disimpan.
Waktu pembangkitan kunci: 0.09924054145812988 detik
```

```
Masukkan nama file: key11
Kunci publik dan kunci privat ECDSA P-256 berhasil disimpan.
Waktu pembangkitan kunci: 0.09136009216308594 detik
```

```
Masukkan nama file: key12
Kunci publik dan kunci privat ECDSA P-256 berhasil disimpan.
Waktu pembangkitan kunci: 0.09376239776611328 detik
```

Rata-rata waktu pembangkitan kunci:
0.09478

Berdasarkan hasil pengujian perbandingan waktu pembangkitan kunci publik dan kunci privat menggunakan DSA dan ECDSA, dapat dilihat bahwa ECDSA membangkitkan kunci lebih cepat, yaitu dalam rata-rata waktu 0.09478, sementara DSA membangkit kunci dalam rata-rata waktu 0.26477.

2. Pengujian waktu *signing* dan perhitungan ukuran tanda tangan

   Pengujian DSA

```
Masukkan nama file kunci privat: key1_dsa_1024
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed
Tanda tangan berhasil disimpan di file: digital_crtf_signed.txt
Waktu signing: 0.004227161407470703 detik
Ukuran tanda tangan: 80 bytes
```

```
Masukkan nama file kunci privat: key2_dsa_1024
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed2
Tanda tangan berhasil disimpan di file: digital_crtf_signed2.txt
Waktu signing: 0.002000570297241211 detik
Ukuran tanda tangan: 80 bytes
```

```
Masukkan nama file kunci privat: key3_dsa_1024
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed3
Tanda tangan berhasil disimpan di file: digital_crtf_signed3.txt
Waktu signing: 0.0020170211791992188 detik
Ukuran tanda tangan: 80 bytes
```

Rata-rata waktu *signing*:
0.00274

Ukuran tanda tangan:
80 bit

Pengujian ECDSA dengan panjang kunci 256 bit

```
Masukkan nama file kunci privat: key10_ecdsa_p256
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed10
Tanda tangan berhasil disimpan di file: digital_crtf_signed10.txt
Waktu signing: 0.002000093460083008 detik
Ukuran tanda tangan: 104 bytes
```

```
Masukkan nama file kunci privat: key11_ecdsa_p256
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed11
Tanda tangan berhasil disimpan di file: digital_crtf_signed11.txt
Waktu signing: 0.0019991397857666016 detik
Ukuran tanda tangan: 104 bytes
```

```
Masukkan nama file kunci privat: key12_ecdsa_p256
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed12
Tanda tangan berhasil disimpan di file: digital_crtf_signed12.txt
Waktu signing: 0.0020017623901367188 detik
Ukuran tanda tangan: 105 bytes
```

Rata-rata waktu *signing*:
0.00199

Ukuran tanda tangan:
104 bit

Berdasarkan hasil pengujian perbandingan waktu *signing* menggunakan DSA dan ECDSA, dapat dilihat bahwa ECDSA menandatangani sertifikat digital lebih cepat dalam rata-rata waktu 0.00199, sedangkan DSA dalam rata-rata waktu 0.00274.

Sementara itu, untuk ukuran tanda tangan yang dihasilkan, DSA memiliki ukuran tanda tangan yang lebih kecil yaitu 80 bit, sedangkan ECDSA memiliki tanda tangan berukuran 104 bit.

Dari sini dapat dilihat bahwa ECDSA menandatangani sertifikat digital lebih cepat meskipun ukuran tanda tangan yang dihasilkan lebih besar.

3. Pengujian waktu verifikasi tanda tangan

Pengujian DSA dengan panjang kunci 1024 bit

```
Masukkan nama file kunci publik: key1_dsa_1024
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed1
Tanda tangan valid.
Waktu verifikasi: 0.0031425952911376953 detik
```

```
Masukkan nama file kunci publik: key2_dsa_1024
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed2
Tanda tangan valid.
Waktu verifikasi: 0.0010001659393310547 detik
```

```
Masukkan nama file kunci publik: key3_dsa_1024
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed3
Tanda tangan valid.
Waktu verifikasi: 0.0009937286376953125 detik
```

Rata-rata waktu *verify*:
0.00171

Pengujian ECDSA dengan panjang kunci 256 bit

```
Masukkan nama file kunci publik: key10_ecdsa_p256
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed10
Tanda tangan valid.
Waktu verifikasi: 0.0009999275207519531 detik
```

```
Masukkan nama file kunci publik: key11_ecdsa_p256
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed11
Tanda tangan valid.
Waktu verifikasi: 0.0010039806365966797 detik
```

```
Masukkan nama file kunci publik: key12_ecdsa_p256
Masukkan nama file sertifikat: digital_crtf
Masukkan nama file tanda tangan: digital_crtf_signed12
Tanda tangan valid.
Waktu verifikasi: 0.0009999275207519531 detik
```

Rata-rata waktu *verify*:
0.00099

Berdasarkan hasil pengujian perbandingan waktu verifikasi tanda tangan pada sertifikat digital, diketahui bahwa ECDSA memverifikasi tanda tangan lebih cepat dalam rata-rata waktu 0.00099, sedangkan DSA memverifikasi tanda tangan dalam rata-rata waktu 0.00171.

Pada hal ini dalam diketahui bahwa ECDSA memverifikasi tanda tangan digital lebih cepat meskipun ukuran tanda tangannya lebih besar.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, *Elliptic Curve Digital Signature Algorithm* (ECDSA) memiliki kinerja yang lebih efektif dan efisien dalam menandatangani sertifikat digital serta memverifikasi tanda tangan pada sertifikat digital dibandingkan dengan *Digital Signature Algorithm*.

ECDSA menghasilkan kunci publik dan kunci privat dengan ukuran lebih kecil daripada DSA, namun menghasilkan tanda tangan digital yang lebih panjang. Meskipun demikian, dari segi waktu, mulai dari pembangkitan tanda tangan, proses *signing*, hingga proses verifikasi tanda tangan, ECDSA memiliki rata-rata waktu lebih cepat dari pada DSA.

B. Saran untuk Penelitian Selanjutnya

- Untuk hasil yang lebih akurat, sebaiknya hasilkan kunci dengan ukuran yang sama untuk DSA dan ECDSA.
- Untuk hasil yang lebih akurat, sebaiknya file sertifikat yang akan ditandatangani berukuran lebih besar agar perbedaan waktu *signing* dan verifikasi tanda tangan antara DSA dan ECDSA dapat terlihat lebih besar.

VIDEO LINK AT YOUTUBE

Penjelasan makalah ini dapat dilihat melalui tautan berikut: https://youtu.be/LXEwf2WuPyY

DAFTAR PUSTAKA

[1] R. Munir, "Public Key Infrastructure (PKI) Bahan Kuliah II4031 Kriptografi dan Koding," Informatika STEI ITB, Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/19-PKI-2023.pdf, Diakses pada: 15 Mei 2023.

[2] R. Munir, "Sertifikat Digital Bahan Kuliah II4031 Kriptografi dan Koding," Informatika STEI ITB, Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/18-Sertifikat-digital-2023.pdf, Diakses pada: 15 Mei 2023.

[3] R. Munir, "Tanda Tangan Digital Bahan Kuliah II4031 Kriptografi dan Koding," Informatika STEI ITB, Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/16-Tanda-tangan-digital-2023.pdf, Diakses pada: 15 Mei 2023.

[4] A. Wahyuni, "Aplikasi Kriptografi untuk Pengamanan E-Dokumen dengan Metode Hybrid: Biometrik Tanda Tangan dan DSA," Tesis, M.Sc Sistem Informasi, Universitas Diponegoro, Semarang, 2011.

[5] M. Rizaldy, "Perbandingan Tanda Tangan Digital RSA dan DSA serta Implementasinya untuk Antisipasi Pembajakan Perangkat Lunak," Informatika STEI ITB, Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2008-2009/Makalah2/MakalahIF3058-2009-b001.pdf, Diakses pada: 17 Mei 2023.

[6] A. Nadzifarin, Asmunin, "Penerapan Elliptic Curve Digital Signature Algorithm pada Tanda Tangan Digital dengan Studi Kasus Dokumen Surat-menyurat," Journal of Informatics and Computer Science, vol. 04, no. 01, pp. 1-9, 2022.

[7] A. Saepulrohman, T. Negara, "Implementasi Algoritma Tanda Tangan Digital Berbasis Kriptografi Kurva Eliptik," Jurnal Ilmiah Ilmu Komputer dan Matematika, vol. 18, no. 1, pp. 22-28, Januari 2021.

[8] R. Munir, "Elliptic Curve Cryptography (ECC) Bahan Kuliah II4031 Kriptografi dan Koding," Informatika STEI ITB, Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/12-ECC-2023.pdf, Diakses pada: 17 Mei 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

Khafifanisa
(18220056)