Analisis Komparatif Elliptic Curve Digital Signature Algorithm (ECDSA) dengan Edward-Curve Digital Signature Algorithm (EdDSA) untuk Penandatanganan *file* Digital

Ken Azizan - 18221107
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 18221107@std.stei.itb.ac.id

***Abstract*—Makalah ini menyajikan analisis perbandingan antara *Elliptic Curve Digital Signature Algorithm* (ECDSA) dengan *Edward-Curve Digital Signature Algorithm* (EdDSA) pada pembuatan tanda tangan digital. Performa algoritma diuji berdasarkan waktu pembangkitan kunci, waktu pembuatan tanda tangan, waktu verifikasi tanda tangan, dan ukuran tanda tangan yang dihasilkan. Penelitian ini menunjukkan performa dari EdDSA lebih baik dari ECDSA pada setiap aspek yang diujikan.**

Keywords—ECDSA, EdDSA, Tanda Tangan Digital, Performa

I. Pendahuluan

Perkembangan teknologi yang terjadi secara progresif menyebabkan digitalisasi dalam berbagai aspek kehidupan sehari-hari. Pengadopsian gawai seperti komputer dan *smartphone* menjadi hal lumrah yang dilakukan masyarakat sebagai penunjang aktivitas. Pada era digital ini, kegiatan pertukaran informasi dilakukan secara digital, termasuk pengiriman dan penerimaan dokumen. Pertukaran informasi dapat dilakukan melalui berbagai *platform* seperti *email,* aplikasi percakapan, dan layanan penyimpanan awan (*cloud storage)*. Namun, seiring dengan peningkatan penggunaan dokumen digital, muncul kekhawatiran mengenai keamanan dokumen digital pada aspek keaslian (*authentication*) dan keutuhan data (*integrity*) pada dokumen tersebut.

Salah satu solusi untuk dapat memastikan keutuhan dan keaslian data dengan memberikan tanda tangan digital pada dokumen. Tanda tangan digital merupakan teknik kriptografi yang dapat memastikan bahwa tidak ada perubahan pada dokumen yang telah ditandatangani, dapat melakukan verifikasi pemberi tanda tangan pada dokumen, serta menjamin nir-penyangkalan. Terdapat beberapa algoritma tanda tangan digital, dua diantaranya adalah *Elliptic Curve Digital Signature Algorithm* (ECDSA) dan *Edward-Curve Digital Signature Algorithm* (EdDSA).

Makalah ini bertujuan untuk membandingkan performa pada penggunaan ECDSA dengan EdDSA dalam beberapa aspek, yaitu waktu pembangkitan kunci, pemberian tanda tangan pada dokumen, verifikasi tanda tangan yang terdapat pada dokumen, serta ukuran tanda tangan yang dibangkitkan. Pada kegiatan eksperimen ini ukuran kunci untuk ECDSA dan EdDSA berukuran sama yaitu sepanjang 256 bit. Sementara itu, ukuran *file* yang diuji terdiri dari tiga kategori, yaitu kecil (1 MB - 10 MB), sedang (10 MB - 100 MB), dan besar (100 MB - 10000 MB).

Penelitian yang dilakukan diharapkan dapat memberikan wawasan tentang kelebihan dan kekurangan dari penggunaan ECDSA dan EdDSA berdasarkan kriteria yang telah ditentukan. Hasil penelitian diharapkan dapat membantu dalam pemilihan algoritma tanda tangan digital yang paling sesuai dengan kebutuhan keamanan dokumen.

II. Kajian Teori

A. Elliptic Curve

Elliptic curve tidak berarti kurva yang terbentuk menyerupai elips. *Elliptic curv*e diterapkan dengan melakukan perhitungan matematika yang menghasilkan kurva halus dengan titik O yang terdefinisi pada ketakhinggaan [1]. Berikut merupakan persamaan dari elliptic curve.

$$y^2 = x^3 + ax + b. \quad (1)$$

Setiap nilai a dan b yang dimasukkan akan menciptakan bentuk kurva yang berbeda. Anggap a memiliki nilai sebesar 2 dan b memiliki nilai sebesar 6. Maka elliptic curve yang dihasilkan akan seperti pada Fig 1.

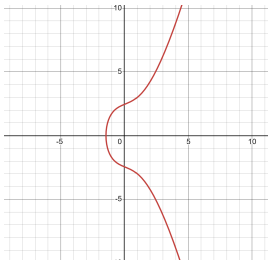

Fig. 1. Contoh *Elliptic Curve*

Dalam penerapan elliptic curve pada kriptografi digunakan perkalian titik skalar pada bentuk $k \cdot P$ dengan k sebagai nilai bilangan bulat positif dan P sebagai titik yang berada pada *elliptic curve*. Perkalian $k \cdot P$ akan menghasilkan titik Q yang juga berada pada *elliptic curve*. Jika hanya diberikan nilai P dan Q, maka sangat sukar untuk mendapatkan nilai k untuk memenuhi $Q = k \cdot P$. Permasalahan ini dikenal sebagai Elliptic Curve Discrete Logarithm Problem (ECDLP) bahkan sampai saat ini belum ada yang dapat memecahkannya [2]. Oleh karena itu, penggunaan elliptic curve sangat cocok untuk diterapkan dalam pertukaran kunci publik karena memiliki tingkat keamanan yang tinggi

B. *Edward Curve*

Edward curve merupakan salah satu jenis pengembangan dari kurva eliptik yang dikemukakan Harolds M. Edward pada tahun 2007. *Edward curve* mempunyai hukum penjumlahan yang lebih sederhana dan efisien daripada kurva eliptik pada umumnya [3]. Berikut merupakan persamaan dari *edwards curve*.

$$x^2 + y^2 = 1 + d x^2 y^2 \qquad (2)$$

Nilai d merupakan parameter dan tidak boleh bernilai 0 ataupun 1 [3]. *Edward Curve* juga memiliki persamaan yang berbentuk lebih umum yang ditunjukkan pada persamaan berikut.

$$x^2 + y^2 = c^2 (1 + d x^2 y^2) \qquad (3)$$

Pada persamaan ini terdapat dua parameter yaitu c dan d. Hasil dari $cd(1 - c^4 d)$ tidak boleh sama dengan 0 [3]. Berikut merupakan grafik *edward curve* dengan persamaan $x^2 + y^2 = 1 + (-250) x^2 y^2$.

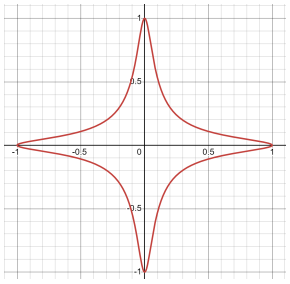

Fig. 2. Contoh *Edward Curve*

Penggunaan *edward curve* dalam kriptografi serupa dengan *elliptic curve* pada umumnya dengan melakukan perkalian titik skalar. Bentuk perkalian titik skalar berupa $k \cdot P$ dengan k sebagai nilai bilangan bulat positif dan P sebagai titik yang berada pada *edward curve* yang akan menghasilkan titik Q yang juga berada pada *edward curve*.

C. Tanda Tangan Digital

Tanda tangan merupakan salah satu cara untuk menunjukan identitas pada dokumen. Tanda tangan memiliki beberapa fungsi, yaitu memverifikasi identitas seseorang, menunjukkan persetujuan terkait isi dokumen, serta menunjukkan keaslian dokumen. Oleh karena itu setiap orang memiliki tanda tangan yang berbeda satu sama lain. Pemberian tanda tangan secara konvensional dilakukan dengan menulis menggunakan pena tinta pada kertas. Namun, seiring dengan kemajuan teknologi pemberian tanda tangan dapat dilakukan secara digital.

Tanda tangan digital adalah skema autentikasi dokumen menggunakan kriptografi. Pengirim dokumen dapat mencantumkan kode unik pada dokumen sebagai tanda tangan digital. Kode unik tersebut dibangkitkan dengan menggunakan *hash function* dan proses enkripsi [4]. Tanda tangan digital terdiri dari dua proses yaitu pemberian tandan tangan (*signing*) dan verifikasi tanda tangan (*verification*). Berikut merupakan skema dari tanda tangan digital.

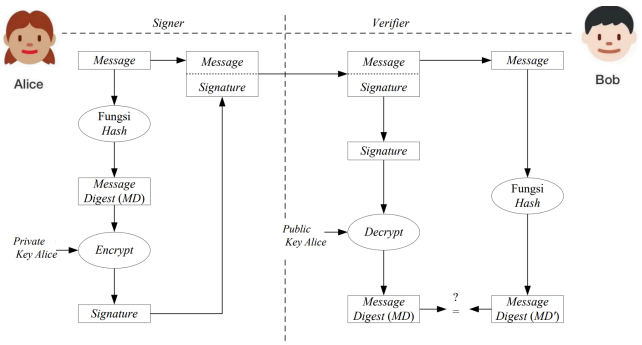

Fig. 3. Skema Tanda Tangan Digital (sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/18-Tanda-tangan-digital-2024.pdf)

Pada proses penandatanganan, pesan akan dimasukkan pada fungsi *hash*. Hasil dari fungsi *hash* berupa *message digest* akan dienkripsi menggunakan kunci privat penandatangan yang akan menghasilkan tanda tangan digital. Tanda tangan digital akan ditempel dengan pesan asli. Setelah itu, pesan dengan tanda tangan digital akan dikirimkan pada penerima.

Untuk proses verifikasi, tanda tangan pada pesan akan dilakukan dekripsi menggunakan kunci publik penandatangan untuk menghasilkan *message digest*. Kemudian, pesan yang dikirimkan akan dimasukkan ke dalam fungsi *hash* untuk mendapatkan *message digest* lainnya. Kedua *message digest* akan dibandingkan untuk memastikan autentikasi dan keutuhan pesan. Jika *message digest* sama, pesan yang dikirimkan terbukti dikirim oleh penanda tangan dan isi pesan masih utuh. Apabila tidak sama, pesan patut dicurigai keaslian dan kebenaran isinya.

D. Elliptic Curve Digital Signature Algorithm (ECDSA)

Elliptic Curve Digital Signature Algorithm (ECDSA) adalah salah satu variasi dari algoritma *elliptic curve cryptography* (ECC) untuk melakukan proses tanda tangan digital. ECDSA ditetapkan menjadi standar ANSI pada tahun 1999. Kemudian, IEEE dan NIST menetapkan ECDSA sebagai standar pada tahun 2000. ECDSA dinilai memiliki tingkat keamanan yang lebih baik daripada algoritma tanda tangan digital lainnya, seperti RSA dan *digital signature algorithm* (DSA). Berikut merupakan tahapan penggunaan ECDSA. [5]

1. Pembangkitan Kunci
   - Menentukan persamaan *elliptic curve* yang akan digunakan.
   - Menentukan nilai n sebagai orde dari persamaan.
   - Menentukan nilai P yang berada pada *elliptic curve*
   - Pemberi tanda tangan menentukan kunci privat d, nilai d harus berada di $[1, ..., n-1]$.
   - Mendapatkan kunci publik Q dari persamaan $Q = d \cdot P$.
2. Penandatanganan (*Signing*)
   - Memasukkan pesan M pada *hash function* $H(M)$ untuk mendapatkan nilai z.
   - Memilih nilai k secara acak, nilai k harus berada di $[1, ..., n-1]$.
   - Melakukan perhitungan perkalian skalar titik kurva, $(x_1, y_1) = k \cdot P$.
   - Mendapatkan nilai r dari persamaan $r = x_1 \bmod n$.
   - Mendapatkan nilai s dari persamaan $s = k^{-1}(z + rd) \bmod n$.
   - Nilai tanda tangan dari pesan berupa pasangan nilai (r, s).
3. Verifikasi
   - Nilai tanda tangan dari pesan berupa pasangan nilai (r, s).
   - Memasukkan pesan pada *hash function* $H(M)$ untuk mendapatkan nilai z
   - Melakukan perhitungan nilai $w = s^{-1} \bmod n$.
   - Melakukan perhitungan nilai $u_1 = z\,w \bmod n$ dan $u_2 = r\,w \bmod n$.
   - Melakukan perhitungan $u_1 \cdot P + u_2 \cdot Q = (x_0, y_0)$.
   - Tanda tangan (r, s) valid apabila $r = x_0 \bmod n$.

E. Edward-Curve Digital Signature Algorithm (EdDSA)

Edward-Curve Digital Signature Algorithm (EdDSA) merupakan salah satu algoritma yang mengimplementasikan *edward curve* pada skema pembuatan tanda tangan digital. Terdapat dua ukuran kunci yang dibangkitkan melalui algoritma EdDSA, yaitu 32 bytes dan 57 bytes. Berikut merupakan tahapan penggunaan ECDSA. [6]

1. Pembangkitan Kunci
   - Menentukan nilai G yang berada pada *edward curve*.
   - Menentukan nilai l sebagai orde dari persamaan.
   - Memilih kunci privat sk sebesar 32 bytes secara acak.
   - mencari nilai *hash* h dari kunci privat dimasukkan ke dalam fungsi SHA-512, $h = H(sk)$.
   - hasil *hash* kunci privat akan dibagi menjadi dua, bagian setengah pertama h_{left} untuk pembuatan kunci publik dan bagian sisanya menjadi h_{right} untuk pembuatan *nonce*.
   - Mendapatkan bilangan bulat a dari h_{left} menggunakan konsep *little-endian*.
   - Mendapatkan kunci publik A dari persamaan $A = a \cdot G$.
2. Pembuatan Tanda Tangan
   - Membuat *nonce* k dengan memasukkan pesan M dan h_{right} pada fungsi SHA-512 dengan persamaan $k = H(M \,||\, h_{right})$.
   - Mendapatkan R dari persamaan $R = k \cdot G$.
   - Mendapatkan nilai r dengan melakukan *hashing* pada M, R, dan A secara bersamaan, $r = H(R \,||\, A \,||\, M)$.
   - Mendapatkan nilai S melalui persamaan $S = k + r \cdot a \bmod l$.
   - Nilai tanda tangan dari pesan berupa pasangan nilai (R, S).
3. Verifikasi Tanda Tangan
   - Mendapatkan nilai r dengan melakukan *hashing* pada M, R, dan A secara bersamaan, $r = H(R \,||\, A \,||\, M)$.
   - Melakukan perhitungan $S \cdot G$ dan $R + r \cdot a$.
   - Tanda tangan terverifikasi apabila nilai $S \cdot G = R + r \cdot a$.

III. Pembahasan

A. Rancangan Eksperimen

Pada kegiatan eksperimen ini terdiri dari tiga tahapan utama yaitu, pembangkitan kunci, pemberian tanda tangan, dan verifikasi tanda tangan. Berikut penjelasan untuk setiap tahapan.

1. Pembangkitan Kunci
   - Melakukan pembangkitan pasangan kunci privat dan publik dengan panjang 256 bit menggunakan algoritma *Elliptic Curve Cryptography*
   - Melakukan pembangkitan pasangan kunci privat dan publik dengan panjang 256 bit

menggunakan algoritma *Edward Curve Cryptography*

- Melakukan perbandingan waktu pembangkitan pasangan kunci antara algoritma *Elliptic Curve Cryptography* dengan algoritma *Edward Curve Cryptography*

2. Pemberian Tanda Tangan
   - Melakukan penandatanganan dokumen dengan algoritma ECDSA menggunakan kunci privat. Pada tahapan ini juga dilakukan *hashing* menggunakan fungsi SHA-256.
   - Melakukan penandatanganan dokumen dengan algoritma EdDSA menggunakan kunci privat. Pada tahapan ini juga dilakukan *hashing* menggunakan jenis *hash* SHA-256.
   - Melakukan perbandingan waktu yang dibutuhkan dalam pemberian tanda tangan antara ECDSA dengan EdDSA
   - Melakukan perbandingan ukuran tanda tangan yang dihasilkan dari ECDSA dengan EdDSA
3. Verifikasi Tanda Tangan
   - Melakukan verifikasi tanda tangan digital dengan algoritma ECDSA menggunakan kunci publik dan memasukkan dokumen pada fungsi SHA-256.
   - Melakukan verifikasi tanda tangan digital dengan algoritma EdDSA menggunakan kunci publik dan memasukkan dokumen pada fungsi SHA-256.
   - Melakukan perbandingan waktu yang diperlukan dalam melakukan verifikasi antara ECDSA dengan EdDSA.

B. Implementasi Eksperimen

Eksperimen ini diimplementasikan menggunakan bahasa pemrograman Python dan menggunakan *library* cryptography yang telah disediakan oleh Python. Panjang kunci yang telah ditentukan yaitu 256 bit untuk ECDSA dan EdDSA, sehingga eksperimen yang dilakukan lebih akurat. Pada eksperimen ini tahapan pertama yang dilakukan adalah pembangkitan kunci. Berikut ini adalah kode program pembangkitan kunci.

Pembangkitan kunci publik dan kunci privat ECDSA

```
from cryptography.hazmat.primitives.asymmetric import ed25519
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import ec
import time
import os

def generate_key_ecdsa(filename):
    keys_dir = "./keys/"
    os.makedirs(keys_dir, exist_ok=True)

    # Pembangkitan pasangan kunci
    start_time = time.time()
    curve = ec.SECP256K1()
    private_key = ec.generate_private_key(curve, default_backend())
    public_key = private_key.public_key()
    end_time = time.time()


    # Penyimpanan kunci privat
    priv_key_bytes = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    with open(os.path.join(keys_dir, f"{filename}_ecdsa.priv"), "wb") as
priv_key_file:
        priv_key_file.write(priv_key_bytes)

    #Penyimpanan kunci publik
    pub_key_bytes = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
    with open(os.path.join(keys_dir, f"{filename}_ecdsa.pub"), "wb") as
pub_key_file:
        pub_key_file.write(pub_key_bytes)

    generate_time = end_time - start_time
    print("Key Generation Time:", generate_time, "seconds")
    print("ECDSA Private Key saved to:", f"{filename}_ecdsa.priv")
    print("ECDSA Public Key saved to:", f"{filename}_ecdsa.pub")
```

Pembangkitan kunci publik dan kunci privat EdDSA

```
from cryptography.hazmat.primitives.asymmetric import ed25519
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import ec
import time
import os
def generate_key_eddsa(filename):
    keys_dir = "./keys/"
    os.makedirs(keys_dir, exist_ok=True)

    # Pembangkitan pasangan kunci
    start_time = time.time()
    priv_key = ed25519.Ed25519PrivateKey.generate()
    pub_key = priv_key.public_key()
    end_time = time.time()


    # Penyimpanan kunci privat
    priv_key_bytes = priv_key.private_bytes(
        encoding=serialization.Encoding.Raw,
        format=serialization.PrivateFormat.Raw,
        encryption_algorithm=serialization.NoEncryption()
    )

    with open(os.path.join(keys_dir, f"{filename}_eddsa.priv"), "wb") as
privkey_file:
        privkey_file.write(priv_key_bytes)

    #Penyimpanan kunci publik
    pub_key_bytes = pub_key.public_bytes(
        encoding=serialization.Encoding.Raw,
        format=serialization.PublicFormat.Raw
    )
    with open(os.path.join(keys_dir, f"{filename}_eddsa.pub"), "wb") as
pubkey_file:
        pubkey_file.write(pub_key_bytes)

    generate_time = end_time - start_time
    print("Key Generation Time:",generate_time, "seconds")
    print("EdDSA Private Key saved to:", f"{filename}_eddsa.priv")
    print("EdDSA Public Key saved to:", f"{filename}_eddsa.pub")
```

Setelah berhasil melakukan pembangkitan kunci, tahapan selanjutnya adalah pemberian tanda tangan pada dokumen

yang dimasukkan menggunakan kunci privat. Berikut merupakan kode program pembuatan tanda tangan digital.

Pembuatan tanda tangan digital menggunakan ECDSA.

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric import ed25519
from cryptography.hazmat.backends import default_backend
from functionList import *
import time
import os

def sign_ecdsa_file(filename, key_filename):
    keys_dir = "./keys/"
    signatures_dir = "./signatures/"
    os.makedirs(signatures_dir, exist_ok=True)

    filename_ori = get_base_file_name(filename)

    with open (filename, "rb") as file:
        message = file.read().strip()

    # Mendapatkan kunci privat
    key_path = os.path.join(keys_dir, f"{key_filename}_ecdsa.priv")
    with open(key_path, "rb") as priv_key_file:
        priv_key_data = priv_key_file.read()
        private_key = serialization.load_pem_private_key(priv_key_data,
password=None, backend=default_backend())

    # Membuat tanda tangan
    start_time = time.time()
    signature = private_key.sign(message, ec.ECDSA(hashes.SHA256()))
    end_time = time.time()

    # Menyimpan tanda tangan pada file
    with open(os.path.join(signatures_dir,
f'{filename_ori}_signature_ecdsa.txt'), "w") as signature_file:
        signature_file.write(signature.hex())

    signing_time = end_time - start_time
    signature_size_bytes = len(signature)


    print('Signature successfully saved into:
',f'{filename_ori}_signature_ecdsa.txt' )
    print("Signing Time:", signing_time, "seconds")
    print("Signature Size:", signature_size_bytes, "bytes")
```

Pembuatan tanda tangan digital menggunakan EdDSA.

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric import ed25519
from cryptography.hazmat.backends import default_backend
from functionList import *
import time
import os

def sign_eddsa_file(filename, key_filename):
    keys_dir = "./keys/"
    signatures_dir = "./signatures/"
    os.makedirs(signatures_dir, exist_ok=True)

    filename_ori = get_base_file_name(filename)

    with open(filename, "rb") as file:
        message = file.read()

    # Mendapat kunci privat
    key_path = os.path.join(keys_dir, f"{key_filename}_eddsa.priv")
    with open(key_path, "rb") as priv_key_file:
        priv_key_data = priv_key_file.read()
        private_key =
ed25519.Ed25519PrivateKey.from_private_bytes(priv_key_data)

    # Membuat tanda tangan
    start_time = time.time()
    digest = hashes.Hash(hashes.SHA256())
    digest.update(message)
    hashed_message = digest.finalize()
    signature = private_key.sign( hashed_message)
    end_time = time.time()

    # Menyimpan file tanda tangan
    signature_path = os.path.join(signatures_dir,
f'{filename_ori}_signature_eddsa.txt')
    with open(signature_path, "w") as signature_file:
        signature_file.write(signature.hex())

    signing_time = end_time - start_time
    signature_size_bytes = len(signature)

    print('Signature successfully saved into: ',
f'{filename_ori}_signature_eddsa.txt')
    print("Signing Time:", signing_time, "seconds")
    print("Signature Size:", signature_size_bytes, "bytes")
```

Setelah melakukan pembuatan tanda tangan, tahapan berikutnya adalah verifikasi tanda tangan. Verifikasi tanda tangan dilakukan dengan menggunakan kunci publik. Berikut merupakan kode program verifikasi tanda tangan.

Verifikasi tanda tangan digital menggunakan ECDSA.

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric import ed25519
from cryptography.hazmat.backends import default_backend
from functionList import *
import time
import os

def verify_ecdsa_signature(filename, signature_filename, key_filename):
    keys_dir = "./keys/"
    signatures_dir = "./signatures/"

    filename_ori = get_base_file_name(filename)
    with open(filename, "rb") as file:
        message = file.read().strip()

    # mendapatkan kunci publik
    key_path = os.path.join(keys_dir, f"{key_filename}_ecdsa.pub")
    with open(key_path, "rb") as pub_key_file:
        pub_key_data = pub_key_file.read()
        public_key = serialization.load_pem_public_key(pub_key_data,
backend=default_backend())

    # Membaca file tanda tangan
    signature_path = os.path.join(signatures_dir,signature_filename)
    with open(signature_path, "r") as signature_file:
        signature_hex = signature_file.read().strip()
        signature = bytes.fromhex(signature_hex)

    # Verifikasi tanda tangan
    try:
        start_time = time.time()
        public_key.verify(signature, message, ec.ECDSA(hashes.SHA256()))
        end_time = time.time()
        verification_time = end_time - start_time
        print("Verification successful.")
        print("Verification Time:", verification_time, "seconds")
    except:
        print("Verification failed. Signature does not match the message.")
```

Verifikasi tanda tangan digital menggunakan EdDSA.

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.asymmetric import ed25519
from cryptography.hazmat.backends import default_backend
from functionList import *
import time
import os

def verify_eddsa_signature(filename, signature_filename, key_filename):
    keys_dir = "./keys/"
    signatures_dir = "./signatures/"

    filename_ori = get_base_file_name(filename)
    with open(filename, "rb") as file:
        message = file.read()


    # Mendapatkan kunci publik
    key_path = os.path.join(keys_dir, f"{key_filename}_eddsa.pub")
    with open(key_path, "rb") as pub_key_file:
        pub_key_data = pub_key_file.read()
        public_key = ed25519.Ed25519PublicKey.from_public_bytes(pub_key_data)

    # Membaca file tanda tangan
    signature_path =
os.path.join(signatures_dir,f"{signature_filename}_signature_eddsa.txt")
    with open(signature_path, "r") as signature_file:
        signature_hex = signature_file.read().strip()
        signature = bytes.fromhex(signature_hex)

    # Verifikasi tanda tangan
    try:
        start_time = time.time()
        digest = hashes.Hash(hashes.SHA256())
        digest.update(message)
        hashed_message = digest.finalize()
        public_key.verify(signature, hashed_message)
        end_time = time.time()
        verification_time = end_time - start_time
        print("Verification successful.")
        print("Verification Time:", verification_time, "seconds")
    except:
        print("Verification failed. Signature does not match the message.")
```

C. Hasil Eksperimen

Terdapat empat parameter yang digunakan untuk membandingkan antara ECDSA dengan EdDSA, yaitu waktu pembangkitan kunci, waktu pembuatan tanda tangan, besar tanda tangan digital yang dihasilkan, dan waktu verifikasi tanda tangan digital.

1. Perbandingan Waktu Pembangkitan Kunci
   Pada setiap jenis algoritma akan dilakukan pembangkitan kunci sebanyak lima kali. Lalu, waktu pembangkitan kunci akan dihitung rata-ratanya untuk setiap jenis algoritma.

   Berikut merupakan hasil percobaan pembangkitan kunci menggunakan ECDSA

```
Input key file name: attempt1
Key Generation Time: 0.11805224418640137 seconds
ECDSA Private Key saved to: attempt1_ecdsa.priv
ECDSA Public Key saved to: attempt1_ecdsa.pub
```

```
Input key file name: attempt2
Key Generation Time: 0.1179664134979248 seconds
ECDSA Private Key saved to: attempt2_ecdsa.priv
ECDSA Public Key saved to: attempt2_ecdsa.pub
```

```
Input key file name: attempt3
Key Generation Time: 0.11722707748413086 seconds
ECDSA Private Key saved to: attempt3_ecdsa.priv
ECDSA Public Key saved to: attempt3_ecdsa.pub
```

```
Input key file name: attempt4
Key Generation Time: 0.12179327011108398 seconds
ECDSA Private Key saved to: attempt4_ecdsa.priv
ECDSA Public Key saved to: attempt4_ecdsa.pub
```

```
Input key file name: attempt5
Key Generation Time: 0.14278745651245117 seconds
ECDSA Private Key saved to: attempt5_ecdsa.priv
ECDSA Public Key saved to: attempt5_ecdsa.pub
```

Mean pembangkitan kunci: 0,12356527

Berikut merupakan hasil percobaan pembangkitan kunci menggunakan EdDSA

```
Input key file name: attempt1
Key Generation Time: 0.10909318923950195 seconds
EdDSA Private Key saved to: attempt1_eddsa.priv
EdDSA Public Key saved to: attempt1_eddsa.pub
```

```
Input key file name: attempt2
Key Generation Time: 0.09785890579223633 seconds
EdDSA Private Key saved to: attempt2_eddsa.priv
EdDSA Public Key saved to: attempt2_eddsa.pub
```

```
Input key file name: attempt3
Key Generation Time: 0.10806703567504883 seconds
EdDSA Private Key saved to: attempt3_eddsa.priv
EdDSA Public Key saved to: attempt3_eddsa.pub
```

```
Input key file name: attempt4
Key Generation Time: 0.10596346855163574 seconds
EdDSA Private Key saved to: attempt4_eddsa.priv
EdDSA Public Key saved to: attempt4_eddsa.pub
```

```
Input key file name: attempt5
Key Generation Time: 0.10868263244628906 seconds
EdDSA Private Key saved to: attempt5_eddsa.priv
EdDSA Public Key saved to: attempt5_eddsa.pub
```

Mean pembangkitan kunci: 0,10593158

Dari hasil percobaan diperoleh bahwa pembangkitan kunci menggunakan EdDSA lebih cepat daripada ECDSA dengan ukuran kunci yang sama. Rata-rata waktu pembangkitan kunci dengan menggunakan EdDSA adalah 0.10593158 sekon sedangkan rata-rata waktu pembangkitan kunci menggunakan ECDSA adalah 0.12356527 sekon.

2. Perbandingan Waktu Pembuatan Tanda Tangan dan Besar Tanda Tangan

Pada setiap kategori ukuran file, pembuatan tanda tangan digital dilakukan dengan menggunakan algoritma ECDSA dan EdDSA. Untuk setiap algoritma, pembuatan tanda tangan diulang sebanyak tiga kali. Berikut hasil pembuatan tanda tangan digital pada file yang ukurannya termasuk kategori kecil.

ECDSA

```
input file name: file_kecil.pdf
input key file name: attempt1
Signature succesfully saved into:  file_kecil_signature_ecdsa.txt
Signing Time: 0.005998373031616211 seconds
Signature Size: 70 bytes
```

```
input file name: file_kecil.pdf
input key file name: attempt1
Signature succesfully saved into:  file_kecil_signature_ecdsa.txt
Signing Time: 0.0050013065338134766 seconds
Signature Size: 70 bytes
```

```
input file name: file_kecil.pdf
input key file name: attempt1
Signature succesfully saved into:  file_kecil_signature_ecdsa.txt
Signing Time: 0.006254434585571289 seconds
Signature Size: 70 bytes
```

Mean *signing* ECDSA: 0,00575136 sekon
Ukuran *digital signature*: 70 bytes

EdDSA

```
input file name: file_kecil.pdf
input key file name: attempt1
Signature successfully saved into:  file_kecil_signature_eddsa.txt
Signing Time: 0.0020046234130859375 seconds
Signature Size: 64 bytes
```

```
input file name: file_kecil.pdf
input key file name: attempt1
Signature successfully saved into:  file_kecil_signature_eddsa.txt
Signing Time: 0.0029993057250976562 seconds
Signature Size: 64 bytes
```

```
input file name: file_kecil.pdf
input key file name: attempt1
Signature successfully saved into:  file_kecil_signature_eddsa.txt
Signing Time: 0.0030052661895751953 seconds
Signature Size: 64 bytes
```

Mean *signing* EdDSA: 0,00266973 sekon
Ukuran *digital signature*: 64 bytes

Berikut hasil pembuatan tanda tangan digital pada file yang ukurannya termasuk kategori sedang.

ECDSA

```
input file name: file_sedang.pdf
input key file name: attempt1
Signature succesfully saved into:  file_sedang_signature_ecdsa.txt
Signing Time: 0.0330047607421875 seconds
Signature Size: 71 bytes
```

```
input file name: file_sedang.pdf
input key file name: attempt1
Signature succesfully saved into:  file_sedang_signature_ecdsa.txt
Signing Time: 0.03499937057495117 seconds
Signature Size: 71 bytes
```

```
input file name: file_sedang.pdf
input key file name: attempt1
Signature succesfully saved into:  file_sedang_signature_ecdsa.txt
Signing Time: 0.031621456146240234 seconds
Signature Size: 71 bytes
```

Mean *signing* ECDSA: 0,0332085 sekon
Ukuran *digital signature*: 71 bytes

EdDSA

```
input file name: file_sedang.pdf
input key file name: attempt1
Signature successfully saved into:  file_sedang_signature_eddsa.txt
Signing Time: 0.03155946731567383 seconds
Signature Size: 64 bytes
```

```
input file name: file_sedang.pdf
input key file name: attempt1
Signature successfully saved into:  file_sedang_signature_eddsa.txt
Signing Time: 0.030846595764160156 seconds
Signature Size: 64 bytes
```

```
input file name: file_sedang.pdf
input key file name: attempt1
Signature successfully saved into:  file_sedang_signature_eddsa.txt
Signing Time: 0.031036853790283203 seconds
Signature Size: 64 bytes
```

Mean *signing* EdDSA: 0,0311476 sekon
Ukuran *digital signature*: 64 bytes

Berikut hasil pembuatan tanda tangan digital pada file yang ukurannya termasuk kategori besar.

ECDSA

```
input file name: file_besar.mp4
input key file name: attempt1
Signature succesfully saved into:  file_besar_signature_ecdsa.txt
Signing Time: 0.2975747585296631 seconds
Signature Size: 71 bytes
```

```
input file name: file_besar.mp4
input key file name: attempt1
Signature succesfully saved into:  file_besar_signature_ecdsa.txt
Signing Time: 0.2759721279144287 seconds
Signature Size: 71 bytes
```

```
input file name: file_besar.mp4
input key file name: attempt1
Signature succesfully saved into:  file_besar_signature_ecdsa.txt
Signing Time: 0.2769956588745117 seconds
Signature Size: 71 bytes
```

Mean *signing* ECDSA: 0,28351414 sekon
Besar *digital signature*: 71 bytes

EdDSA

```
input file name: file_besar.mp4
input key file name: attempt1
Signature successfully saved into:  file_besar_signature_eddsa.txt
Signing Time: 0.2812654972076416 seconds
Signature Size: 64 bytes
```

```
input file name: file_besar.mp4
input key file name: attempt1
Signature successfully saved into:  file_besar_signature_eddsa.txt
Signing Time: 0.27354884147644043 seconds
Signature Size: 64 bytes
```

```
input file name: file_besar.mp4
input key file name: attempt1
Signature successfully saved into:  file_besar_signature_eddsa.txt
Signing Time: 0.2731201648712158 seconds
Signature Size: 64 bytes
```

Mean *signing* EdDSA: 0,275978 sekon
Ukuran *digital signature*: 64 bytes

Hasil percobaan menunjukkan semakin besar *file* yang ditandatangani, maka semakin lama waktu penandatanganan. Pada setiap kategori ukuran *file*, EdDSA selalu lebih cepat daripada ECDSA dalam pembuatan tanda tangan. Pada file berukuran kecil, EdDSA memerlukan waktu rata-rata 0,00266973 sekon sedangkan ECDSA memerlukan waktu rata-rata 0,00575136 sekon. Untuk *file* berukuran sedang, EdDSA membutuhkan waktu rata-rata sebesar 0,0311476 sekon dan ECDSA memerlukan waktu rata-rata 0,0332085 sekon. Pada *file* berukuran besar, EdDSA membutuhkan waktu rata-rata 0,275978 sekon sedangkan ECDSA membutuhkan waktu rata-rata 0,28351414 sekon. EdDSA juga selalu menghasilkan tanda tangan yang berukuran lebih kecil daripada ECDSA pada setiap kategori ukuran *file*. Ukuran tanda tangan yang dihasilkan dengan EdDSA adalah 64 bytes sedangkan tanda tangan hasil ECDSA adalah 70 bytes untuk kategori *file* kecil dan 71 bytes untuk kategori *file* sedang dan besar.

3. Perbandingan Waktu Verifikasi
Pada setiap kategori ukuran file, verifikasi tanda tangan digital dilakukan dengan menggunakan algoritma ECDSA dan EdDSA. Untuk setiap algoritma, verifikasi tanda tangan diulang sebanyak tiga kali. Berikut hasil verifikasi tanda tangan digital pada file yang ukurannya termasuk kategori kecil.

ECDSA

```
input file name: file_kecil.pdf
input key file name: attempt1
input file signature name: file_kecil
Verification successful.
Verification Time: 0.006964921951293945 seconds
```

```
input file name: file_kecil.pdf
input key file name: attempt1
input file signature name: file_kecil
Verification successful.
Verification Time: 0.006989717483520508 seconds
```

```
input file name: file_kecil.pdf
input key file name: attempt1
input file signature name: file_kecil
Verification successful.
Verification Time: 0.005982398986816406 seconds
```

Mean verifikasi ECDSA: 0,00664568 sekon

EdDSA

```
input file name: file_kecil.pdf
input key file name: attempt1
input file signature name: file_kecil
Verification successful.
Verification Time: 0.003912687301635742 seconds
```

```
input file name: file_kecil.pdf
input key file name: attempt1
input file signature name: file_kecil
Verification successful.
Verification Time: 0.0029990673065185547 seconds
```

```
input file name: file_kecil.pdf
input key file name: attempt1
input file signature name: file_kecil
Verification successful.
Verification Time: 0.0029981136322021484 seconds
```

Mean verifikasi EdDSA: 0,0033032892 sekon

Berikut hasil verifikasi tanda tangan digital pada file yang ukurannya termasuk kategori sedang.

ECDSA

```
input file name: file_sedang.pdf
input key file name: attempt1
input file signature name: file_sedang
Verification successful.
Verification Time: 0.03600192070007324 seconds
```

```
input file name: file_sedang.pdf
input key file name: attempt1
input file signature name: file_sedang
Verification successful.
Verification Time: 0.034010410308837891 seconds
```

```
input file name: file_sedang.pdf
input key file name: attempt1
input file signature name: file_sedang
Verification successful.
Verification Time: 0.03416013717651367 seconds
```

Mean verifikasi ECDSA: 0,0347235796 sekon

EdDSA

```
input file name: file_sedang.pdf
input key file name: attempt1
input file signature name: file_sedang
Verification successful.
Verification Time: 0.03203630447387695 seconds
```

```
input file name: file_sedang.pdf
input key file name: attempt1
input file signature name: file_sedang
Verification successful.
Verification Time: 0.031281471252441406 seconds
```

```
input file name: file_sedang.pdf
input key file name: attempt1
input file signature name: file_sedang
Verification successful.
Verification Time: 0.03094482421875 seconds
```

Mean verifikasi EdDSA: 0,0314208 sekon

Berikut hasil verifikasi tanda tangan digital pada file yang ukurannya termasuk kategori besar.

ECDSA
input file name: file_besar.mp4 input key file name: attempt1 input file signature name: file_besar Verification successful. Verification Time: 0.2792315483093261 7 seconds
input file name: file_besar.mp4 input key file name: attempt1 input file signature name: file_besar Verification successful. Verification Time: 0.2735304832458496 seconds
input file name: file_besar.mp4 input key file name: attempt1 input file signature name: file_besar Verification successful. Verification Time: 0.2741727828979492 seconds
Mean verifikasi ECDSA: 0,275645 sekon
EdDSA
input file name: file_besar.mp4 input key file name: attempt1 input file signature name: file_besar Verification successful. Verification Time: 0.2716286182403564 5 seconds
input file name: file_besar.mp4 input key file name: attempt1 input file signature name: file_besar Verification successful. Verification Time: 0.2655317783355713 seconds
input file name: file_besar.mp4 input key file name: attempt1 input file signature name: file_besar Verification successful. Verification Time: 0.2675268650054931 6 seconds
Mean verifikasi EdDSA: 0,268229 sekon

Hasil percobaan menunjukkan semakin besar *file* yang diverifikasi, maka waktu yang dibutuhkan semakin banyak. Pada setiap kategori ukuran *file*, EdDSA selalu lebih cepat daripada ECDSA dalam melakukan verifikasi tanda tangan. Pada *file* berukuran kecil, EdDSA memakan waktu rata-rata 0,0033032892 sekon sedangkan ECDSA memerlukan waktu rata-rata 0,00664568 sekon. Untuk *file* berukuran sedang, EdDSA membutuhkan waktu rata-rata 0,0314208 sekon dan ECDSA memerlukan waktu rata-rata 0,0347235796 sekon. Pada *file* berukuran besar, EdDSA membutuhkan waktu rata-rata 0,268229 sekon sedangkan ECDSA membutuhkan waktu rata-rata 0,275645 sekon.

IV. Kesimpulan & Saran

D. Kesimpulan dan Saran

Berdasarkan hasil percobaan, EdDSA (Edwards-curve Digital Signature Algorithm) menunjukkan performa yang lebih baik daripada ECDSA (Elliptic Curve Digital Signature Algorithm). EdDSA unggul pada setiap aspek yang diuji, seperti waktu pembankitan kunci yang lebih cepat, pembuatan tanda tangan lebih cepat, verifikasi tanda tangan lebih cepat, dan ukuran tanda tangan yang dihasilkan selalu lebih kecil. Ukuran tanda tangan yang dihasilkan dengan EdDSA selalu berukuran sama, terlepas dari ukuran file yang ditandatangani. Ini memberikan keuntungan dalam hal efisiensi penyimpanan dan transmisi tanda tangan digital.

Untuk penelitian ke depannya, diharapkan tidak hanya melakukan perbandingan pada performa saja, tetapi juga melakukan perbandingan dari segi keamanan antara ECDSA dan EdDSA. Meskipun hasil percobaan menunjukkan bahwa EdDSA memiliki performa yang lebih baik daripada ECDSA, aspek keamanan merupakan faktor yang sangat penting dalam pemilihan algoritma tanda tangan digital. Perbandingan yang komprehensif dapat memberikan hasil yang lebih lengkap dan berimbang dalam menentukan algoritma tanda tangan digital.

Video Link at Youtube *(Heading 5)*

https://youtu.be/nJaJtTFwerk

Acknowledgment

Penulis ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa sebagai pemberi ilmu, sehingga penulis dapat menyelesaikan makalah ini. Terima kasih yang sebesar-besarnya kepada Bapak Dr. Rinaldi Munir, M.T. sebagai dosen yang telah memberikan materi mengenai kriptografi. Penulis juga mengucapkan terima kasih kepada kakak tingkat yang telah memberikan referensi pembuatan makalah ini, serta para penulis lainnya yang telah mempublikasikan hasil penelitian mereka yang menjadi bahan referensi penting. Semoga makalah ini bermanfaat bagi pembaca dan dapat memberikan kontribusi dalam pengembangan ilmu kriptografi.

References

[1] R. B. Deokar, “File Transfer on Cloud using Diffie-Hellman Key Exchange in Conjunction with AES Encryption,” Master’s Thesis, Dublin, National College of Ireland, 2023. [Online]. Available: https://norma.ncirl.ie/6467/

[2] M. Amara and A. Siad, “ELLIPTIC CURVE CRYPTOGRAPHY AND ITS APPLICATIONS,” *Int. Workshop Syst. Signal Process. Their Appl. WOSSPA*, Jun. 2011, doi: 10.1109/WOSSPA.2011.5931464.

[3] Y. El Housni, “Edwards curves,” Dec. 2018. [Online]. Available: https://hal.science/hal-01942759

[4] R. Kaur and A. Kaur, “Digital Signature,” in *2012 International Conference on Computing Sciences*, Phagwara, India: IEEE, Sep. 2012, pp. 295–301. doi: 10.1109/ICCS.2012.25.

[5] A. Khalique, K. Singh, and S. Sood, “Implementation of Elliptic Curve Digital Signature Algorithm,” *Int. J. Comput. Appl.*, vol. 2, no. 2, pp. 21–27, May 2010, doi: 10.5120/631-876.

[6] S. Josefsson and I. Liusvaara, “Edwards-Curve Digital Signature Algorithm (EdDSA),” no. 8032. in Request for Comments. RFC Editor, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8032

[7] 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/2023-2024/18-Tanda-tangan-digital-2024.pdf, Diakses pada: 11 Juni 2024.

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, 12 Juni 2024

Ken Azizan - 18221107