Perbandingan Kecepatan Algoritma Tanda Tangan Digital Edwards Curve Digital Signature dengan Elliptic Curve Digital Signature

Jan Meyer Saragih / 13517131
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517131@std.stei.itb.ac.id

***Abstrak*—Dengan perkembangan teknologi, semakin banyak algoritma tanda tangan digital yang dapat digunakan sekarang. Salah satu di antaranya adalah algoritma tanda tangan digital yang berbasis kurva eliptik. Algoritma berbasis kurva eliptik sering digunakan pada mesin yang memiliki daya komputasi rendah karena memiliki ukuran kunci yang lebih kecil dibandingkan dengan algoritma tanda tangan digital yang paling umum digunakan, RSA. Contoh algoritma kurva eliptik adalah *Elliptic Curve Digital Signature Algorithm* (ECDSA) dan *Edwards-Curve Digital Signature Algorithm* (EdDSA). Makalah ini akan membandingkan implementasi dan kecepatan dari kedua algoritma tersebut.**

***Kata Kunci—Tanda tangan digital; ECDSA; EdDSA; Kriptografi kunci publik*.**

I. PENDAHULUAN

Perkembangan teknologi yang pesat hingga sekarang mengakibatkan munculnya sebuah bentuk tanda tangan baru, yaitu tanda tangan digital. Berbeda dengan tanda tangan yang biasa digunakan untuk tanda tangan kontrak hingga awal abad ke-20, tanda tangan ini tidak menggunakan tinta sebagai bukti keabsahannya. Tanda tangan ini menggunakan teknik kriptografi. Teknik kriptografi yang paling umum digunakan adalah kriptografi kunci publik. Pihak yang menandatangani tanda tangan tersebut memiliki kunci privat dan memberikan kunci publik ke pihak lain. Dengan demikian, pihak yang diberikan kunci publik tersebut mampu memverifikasi apakah tanda tangan yang diberikan benar-benar dari pihak yang memiliki kunci privat.

Gagasan untuk skema tanda tangan digital dibuat pada tahun 1976 oleh Diffie-Helman (Whitfield Diffie dan Martin Hellman), penggagas algoritma pertukaran kunci simetri Diffie-Hellman. Setelah itu, muncul implementasi dengan menggunakan kriptografi kunci publik untuk tanda tangan digital pertama kali dilakukan oleh Ron Riveest, Adi Shamir, dan Leonard Adleman dengan algoritma kriptografi kunci publik RSA.

Adapun algoritma kriptografi kunci publik terus berkembang dan muncul algoritma-algoritma kriptografi kunci publik baru yang dapat digunakan untuk menggantikan RSA dalam skema tanda tangan digital. Salah satu algoritma yang menggantikan algoritma RSA adalah algoritma yang berbasis kurva eliptik. Contoh algoritma tanda tangan digital yang berbasis kurva eliptik adalah Elliptic Curve Digital Signature Algorithm (ECDSA) dan Edward's Curve Digital Signature Algorithm (EdDSA).

Algoritma tanda tangan digital yang berbasis kurva eliptik dikatakan lebih ramah terhadap memori dibandingkan algoritma RSA dikarenakan algoritma yang berbasis kurva eliptik membutuhkan ukuran kunci yang lebih kecil dibandingkan algoritma RSA. Dengan demikain, algoritma ini membutuhkan kemampuan komputasi yang lebih kecil dibandingkan algoritma yang berdasarkan operasi aritmatika.

Walaupun sama-sama berbasiskan kurva eliptik, terdapat beberapa perbedaan antara implementasi algoritma tanda tangan digital ECDSA dan algoritma tanda tangan digital EdDSA. Maka dari itu, makalah ini bertujuan untuk membandingkan kedua algoritma tersebut dan meneliti perbedaan-perbedaan yang terdapat pada kedua algoritma tersebut. Setelah mengetahui perbedaan tersebut, maka akan dicoba dibandingkan kecepatan enkripsi antara

II. LANDASAN TEORI

A. Tanda Tangan Digital

Tanda tangan digital merupakan sebuah skema atau algoritma yang digunakan untuk membuktikan keabsahan sebauh dokumen digital. Sebuah tanda tangan digital merupakan nilai kriptografis yang bergantung pada isi pesan dan kunci. Selain digunakan untuk dokumen digital, tanda tangan digital juga dapat digunakan dalam surat elektronik untuk membuktikan keabsahan surat elektronik tersebut beserta bukti integritas surat elektronik tersebut.

Terdapat 3 aspek keamanan dari kriptografi yang dipenuhi oleh tanda tangan digital. Aspek-aspek tersebut adalah sebagai berikut [1].

1. Otentikasi
   Tanda tangan digital harus mampu memberikan bukti bahwa sebuah pesan atau dokumen benar-benar ditandatangani oleh pihak yang memiliki kunci.
2. Keaslian pesan
   Tanda tangan digital harus mampu memberikan bukti bahwa pesan atau dokumen tersebut tidak diubah setelah

ditandatangani oleh pemilik kunci hingga sampai ke pihak yang melakukan verifikasi.

3. Anti-penyangkalan
   Pihak yang sudah menandatangani pesan atau dokumen tidak dapat membantah fakta bahwa pihak tersebut sudah menandatangani pesan atau dokumen yang bersangkutan.

B. Kriptografi Kunci Publik

Kriptografi kunci publik diciptakan pada bulan Mei 1975 untuk menyelesaikan 2 permasalahan utama pada kriptografi kunci simetrik [2].

1. Masalah distribusi kunci. Dengan menggunakan kriptografi kunci publik, kedua pihak yang belum bertemu dapat berkomunikasi karena terdapat kunci yang disebarkan secara massal (kunci publik).
2. Masalah tanda tangan digital. Dengan menggunakan kriptografi kunci publik, dimungkinkan untuk pihak yang tidak mengetahui penandatangan kunci mengotentikasi tanda tangan tersebut.

Pada kriptografi kunci publik. Terdapat 2 jenis kunci, yaitu kunci publik dan kunci privat. Kunci privat dimiliki hanya oleh satu pihak dan tidak diketahui oleh pihak-pihak lain (rahasia). Sementara itu, kunci publik disebarkan kepada publik dan bukan merupakan sebuah rahasia.

Kriptografi kunci publik dapat digunakan secara 2 arah. Pertama, digunakan untuk membuat pesan yang dapat ditulis oleh siapa pun namun hanya dapat dibaca oleh pemilik kunci privat. Kedua, digunakan untuk membuat sebuah pesan yang hanya dapat ditulis oleh pemilik kunci privat, namun dapat dibaca oleh publik. Fungsi kedua inilah yang digunakan dalam tanda tangan digital.

Seperti yang dijelaskan pada bagian pendahuluan, teknik kriptografi yang sering digunakan untuk tanda tangan digital adalah kriptografi kunci publik (kriptografi kunci asimetrik). Hal ini dikarenakan kriptografi kunci simetrik membutuhkan algoritma pertukaran kunci untuk melakukan verifikasi tanda tangan dan dinilai kurang efisien karena proses ini harus diulangi berkali-kali untuk setiap pihak yang harus memverifikasi pesan tersebut. Selain itu, algoritma ini tidak dapat menangani aspek anti-penyangkalan dari tanda tangan digital. Hal ini dikarenakan kunci yang digunakan untuk tanda tangan diketahui oleh kedua pihak.

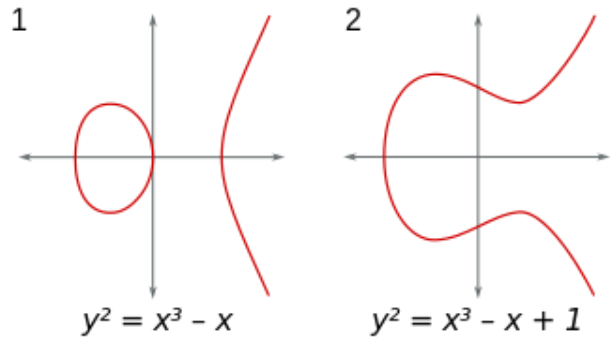
Di sisi lain, kriptografi kunci publik memungkinkan pihak penanda tangan dapat menyebarkan kunci publik sekali kepada orang banyak dan setiap orang dapat menggunakan kunci publik itu untuk memverifikasi pesan atau dokumen yang ditandatangani oleh pihak pemilik kunci privat. Selain itu, kunci privat hanya dimiliki oleh pihak yang menandatangani dokumen tersebut. Dengan demikian, hal ini memenuhi aspek anti-penyangkalan dari sebuah tanda tangan digital.

Algoritma kriptografi kunci publik yang paling sering digunakan adalah RSA, yang berbasis operasi aritmatika seperti eksponensial, modulus, perkalian, dan penjumlahan. Namun, seiring perkembangan zaman, bermunculan algoritma kriptografi kunci publik baru yang berusaha meningkatkan efisiensi dari algoritma kriptografi kunci publik RSA. Salah satu contohnya adalah algoritma kriptografi kunci publik berbasis kurva eliptik.

C. Kurva Eliptik

Kurva eliptik merupakan bagian penting dalam kriptografi kunci publik berbasis kurva eliptik. Penggunaan kriptografi berdasarkan kurva eliptik bermula dari terbentuknya sistem yang berdasarkan grup Abelian terbatas beserta kesuksesan H.W. Lentra dalam melakukan faktorisasi bilangan bulat pada kurva eliptik [3]. Hal ini memungkinkan penelitian untuk melakukan kriptografi memanfaatkan titik-titik pada kurva eliptik yang dibatasi dengan bilangan prima besar.

Contoh dari kurva eliptik digambarkan pada gambar di bawah ini.



Gambar 1. Contoh Kurva Eliptik
Sumber: https://en.wikipedia.org/wiki/Elliptic_curve

Rumus dasar yang digunakan untuk merepresentasikan sebuah kurva eliptik adalah sebagai berikut.

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

Terdapat variasi pada bentuk kurva eliptik tergantung pada nilai a dan b. Dengan menggunakan variasi nilai a dan b, maka didapatkan kurva eliptik dengan bentuk yang berbeda-beda. Hal ini dapat dilihat pada Gambar 1, yaitu perbedaan bentuk kurva 1 dan kurva 2 karena perbedaan nilai b. Namun, terdapat batasan pada pemilihan nilai a dan b, yaitu sebagai berikut.

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

Jumlah pasangan bilangan bulat (x, y) yang memenuhi (1) ada tak hingga. Namun, dengan membatasi nilai x dan y dengan sebuah bilangan prima besar p (menggunakan operasi modulus), maka didapatkan sebuah kurva eliptik yang diwakili dengan pasangan bilangan bulat (x, y) yang berjumlah banyak, namun masih berhingga. Titik-titik inilah yang digunakan untuk melakukan kriptografi kunci publik berbasis kurva eliptik. Sebuah pesan direpresentasikan oleh salah satu titik pada kurva ini dan diubah saat melakukan proses enkripsi. Kemudian titik yang diubah tersebut dikembalikan ke titik semula pada proses dekripsi.

Terdapat 2 jenis operasi yang dapat dilakukan kepada titik-titik yang terdapat pada kurva eliptik, yaitu sebagai berikut [4].

1. Penjumlahan titik
   Proses ini menjumlahkan 2 buah titik (p dan q) pada kurva eliptik dan menghasilkan 1 titik (r). Proses ini memanfaatkan penghitungan gradien, yang dihitung dengan cara berikut.
   $$m = \frac{y_p - y_q}{x_p - x_q} \qquad (3)$$
   Setelah itu, gradien ini digunakan untuk menentukan titik hasil penjumlahan (r) dengan rumus berikut.

$$x_r = m^2 - x_p - x_q \quad (4)$$
$$y_r = m(x_p - x_r) - y_p \quad (5)$$

2. Penggandaan titik
   Proses ini menggandakan 1 titik (p) pada kurva eliptik dan menghasilkan 1 titik (r). Proses ini memanfaatkan penghitungan gradien dengan cara berikut [4].
$$m = \frac{3x_p^2 + a}{2y_p} \quad (6)$$
   Gradien digunakan untuk menentukan titik hasil penggandaan (r) dengan rumus berikut.
$$x_r = m^2 - 2x_p \quad (7)$$
$$y_r = m(x_p - x_r) - y_p \quad (8)$$

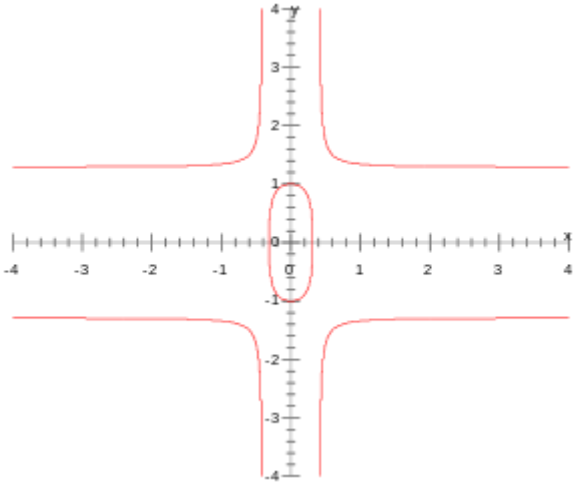
D. Kurva Edward Melengkung

Kurva Edward melengkung *(twisted Edwards* curve) merupakan variasi dari kurva eliptik. Jenis kurva ini lebih umum digunakan pada algoritma EdDSA dibandingkan ECDSA. Adapun rumus dasar dari kurva ini adalah sebagai berikut.

$$ax^2 + y^2 = 1 + dx^2y^2 \quad (9)$$

Kurva tersebut memiliki 2 buah variabel, yaitu a dan d untuk menghasilkan variasi pada parameter kurva tersebut, layaknya variabel a dan b pada kurva eliptik biasa. Kurva ini ekuivalen dengan kurva Montgomery [5] yang memiliki rumus umum sebagai beikut.

$$By^2 = x^3 + Ax^2 + x \quad (10)$$



Gambar 2. Contoh Kurva Edward Melengkung ($10x^2 + y^2 = 1 + 6x^2y^2$)
Sumber: https://en.wikipedia.org/wiki/Twisted_Edwards_curve

Sama seperti kurva eliptik pada umunya, terdapat 2 buah persamaan umum yang digunakan pada kurva Edward melengkung, yaitu sebagai berikut [5].

1. Penjumlahan titik
   Proses ini menjumlahkan 2 buah titik (p dan q) pada kurva Edward melengkung menjadi 1 titik (r). Hasil penjumlahan kedua titik dapat ditemukan menggunakan rumus berikut.
$$x_r = \frac{x_p y_q + y_p x_q}{1 + d x_p x_q y_p y_q} \quad (11)$$
$$y_r = \frac{y_p y_q - a x_p x_q}{1 - d x_p x_q y_p y_q} \quad (12)$$
2. Penggandaan titik
   Proses ini menggandakan 1 titik (p) pada kurva Edward melengkung menjadi 1 titik (r). Hasil penggandaan titik dapat ditemukan menggunakan rumus berikut.
$$x_r = \frac{2x_p y_p}{ax_p^2 + y_p^2} \quad (13)$$
$$y_r = \frac{y_p^2 - ax_p^2}{2 - ax_p^2 - y_p^2} \quad (14)$$

E. Persoalan Diskrit Logaritmik Kurva Eliptik

Persoalan ini merupakan hal yang mendasari penggunaan kurva eliptik dan variasinya untuk digunakan pada sebuah kritpografi kunci publik. Diberikan dua buah titik, P dan Q yang berada pada sebuah kurva eliptik. Carilah sebuah bilangan bulat k sedemikan sehingga.

$$Q = kP \quad (15)$$

Dikarenakan terdapat banyak titik yang terdapat pada sebuah kurva eliptik, sulit untuk menemukan nilai k yang memenuhi persoalan tersebut. Maka dari itu nilai k digunakan sebagai kunci privat dalam sebuah algoritma kriptografi kunci publik berbasis kurva eliptik. Dengan demikian, hanya pemiliki kunci privat yang mampu menghasilkan titik Q dari titik P pada sebuah kurva eliptik yang sudah ditentukan.

E. ECDSA

ECDSA (*Elliptic Curve Digital Signature Algorithm*) merupakan sebuah algoritma kriptografi kunci publik yang digunakan sebagai proses tanda tangan digital. Algoritma ini memanfaatkan kurva eliptik sebagai dasar perhitungannya.

Dalam mengimplementasikan ECDSA, terdapat 3 buah proses yang dilakukan, yaitu sebagai berikut [6].

1. Pembangkitan kunci (*Key Generation*)
   Proses ini merupakan proses yang dilakukan untuk membangkitkan kunci privat (*d*) dan kunci publik (*Q*) yang nanti akan digunakan pada saat proses pembangkitan tanda tangan dan verifikasi tanda tangan digital. Proses pembangkitan kunci mengikuti algoritma berikut.
   a. Ambil bilangan bulat acak yang bernilai antara 1 hingga *n-1* di mana *n* merupakan *order* dari sebuah kurva eliptik. *Order* dari kurva eliptik adalah jumlah titik yang berada pada kurva eliptik tersebut.
   b. Setelah mengambil bilangan bulat acak. Kalikan basis kurva (*G*) sejumlah *d* kali untuk mendapatkan kunci publik. Dengan kata lain $Q = dG$.
2. Pembangkitan tanda tangan (*Signing*)
   Proses ini menghasilkan tanda tangan yang bergantung pada kunci privat dan pesan yang diberikan. Proses pembuatan tanda tangan mengikuti algoritma berikut.
   a. Memilih nilai acak k, di mana $k < n$. Dengan demikian, dipastikan k bukan merupakan titik tak hingga dalam kurva eliptik.
   b. Menghitung titik dari k tersebut dengan melakukan perkalian kG.
   c. Mengambil absis dari titik kG dengan nilai x1dan menghitung nilai $r = x_1 mod\, n$.
   d. Jika $r = 0$, maka kembali ke tahap 1.
   e. Menggunakan algoritma *hash* untuk melakukan *hashing* dari *message* yang masuk. Nilai yang dihasilkan adalah *e*.

f. Menghitung nilai $s = k^{-1}(e + dr)\ mod\ n$. Jika $s = 0$, maka kembali ke tahap 1.
g. Jika s tidak 0, masukkan nilai r dan s menjadi *signature* dari *message* tersebut.
h. Jika diperlukan, ubah nilai *integer* r dan s ke dalam hexadecimal.

3. Verifikasi tanda tangan (*Verifying*)
Proses ini menerima tanda tangan, pesan, dan kunci publik. Hasil dari proses ini adalah apakah tanda tangan yang diberikan dan pesan yang diberikan sesuai atau tidak. Proses verifikasi tanda tangan mengikuti algoritma berikut.
a. Pastikan nilai r dan s minimal bernilai 1 dan maksimal bernilai $n - 1$.
b. Menggunakan algoritma *hash* untuk melakukan *hashing* dari *message* yang masuk. Nilai yang dihasilkan adalah *e*.
c. Menghitung nilai $w = (s - 1)\ mod\ n$.
d. Menghitung nilai $u_1 = (ew)\ mod\ n$ dan $u_2 = (rw)\ mod\ n$.
e. Menghitung nilai $X = u_1 G + u_2 Q$.
f. Jika X = O (nilai tak hingga), maka tanda tangan tidak valid.
g. Jika X bukan nilai tak hingga, maka ambil absis dari X, katakan x_1 lalu hitung nilai $v = x_1\ mod\ n$.
h. Tanda tangan dinilai valid apabila nilai $v = r$.

F. EdDSA

EdDSA (*Edwards-Curve Digital Signature Algorithm*) merupakan algoritma tanda tangan digital yang menggunakan kurva Edward melengkung sebagai dasar perhitungannya. Perlu diperhatikan bahwa algoritma EdDSA, terutama yang paling terkenal Ed25519, pada umumnya menggunakan *hash* SHA-512 dan umumnya hasil *hash* tersebut akan dibagi menjadi 2 untuk diproses pada tahap berikutnya.

Dalam implementasi sederhana, terdapat 3 buah proses yang dilakukan, yaitu sebagai berikut [7].

1. Pembangkitan kunci (*Key Generation*)
Proses ini merupakan proses untuk membangkitkan kunci privat (*d*) dan kunci publik (*Q*) yang nanti akan digunakan pada proses pembangkitan tanda tangan dan verifikasi tanda tangan digital. Proses pembangkitan kunci mengikuti algoritma sebagai berikut.
a. Pilih sebuah *seed k*.
b. Lakukan *hash* pada *k*, setelah itu bagi menjadi 2 buah bagian dengan panjang bit yang sama (*a* dan *b*). Bagian *a* akan digunakan sebagai kunci privat (*d*).
c. Buat kunci publik dengan mengalikan basis kurva sebanyak *d* kali ($Q = dG$).

2. Pembangkitan tanda tangan (*Signing*)
Proses ini menghasilkan tanda tangan yang bergantung pada kunci privat (*d*), nilai *b*, yaitu sisa dari kunci privat, serta kunci publik Q yang akan diambil absisnya. Masukan yang diterima untuk memulai proses ini adalah kunci tersebut beserta pesan (*M*). Proses pembuatan tanda tangan mengikuti algoritma berikut.
a. Lakukan *hash* pada nilai *b* yang digabungkan dengan pesan yang diberikan.

$$r = Hash(b, M)$$

b. Mencari titik *R*, yaitu titik basis *G* dikali sebanyak *r* kali ($R = rG$).
c. Mencari nilai *h*, yaitu dengan menggabungkan hasil *hash* pesan (M), absis dari kunci publik (Q), dan absis dari titik *R*. Setelah itu, lakukan operasi *hash*.

$$h = Hash(R, Q, M)$$

d. Cari nilai *s*, yaitu $s = (r + h * d)\ mod\ n$.
e. Titik *R* dan nilai *s* merupakan hasil tanda tangan digital.

3. Verifikasi tanda tangan (*Verifying*)
Proses ini menerima tanda tangan (*R* dan *s*), pesan (*M*), dan kunci publik (*Q*). Hasil dari proses ini adalah apakah tanda tangan yang diberikan dan pesan yang diberikan sesuai atau tidak. Proses verifikasi tanda tangan mengikuti algoritma berikut.
a. Mencari nilai *h*, yaitu dengan menggabungkan hasil *hash* pesan (M), absis dari kunci publik (Q), dan absis dari titik *R*. Setelah itu, lakukan operasi *hash*.

$$h = Hash(R, Q, M)$$

b. Hitung *p1*, yaitu dengan mengalikan titik basis *G* sebanyak s kali ($p1 = sG$).
c. Hitung *p2* dengan melakukan operasi pada kurva eliptik, yaitu $p2 = R + h * Q$.
d. Tanda tangan dinilai valid apabila *p1* = *p2*.

III. PERENCANAAN PERCOBAAN

Sebelum melakukan perbandingan pada kecepatan secara langsung (diukur waktu eksekusinya), maka ada beberapa hal yang perlu diperhatikan agar perbandingan yang dihasilkan lebih objektif. Hal ini disebabkan perbedaan-perbedaan lain selain dari perbedaan proses algoritma yang digunakan untuk mengimplementasikan ECDSA dan EdDSA.

Perbedaan pertama yang perlu diperhatikan adalah perbedaan algoritma yang dilakukan untuk melakukan *hash*. Terdapat banyak algoritma *hash* yang dapat digunakan dalam mengimplementasikan ECDSA dan EdDSA, beberapa contoh di antaranya adalah SHA-1, SHA-256, dan SHA-512. Untuk melakukan pengujian kecepatan yang lebih akurat, maka disamakan algoritma *hashing* yang digunakan oleh ECDSA dan EdDSA.

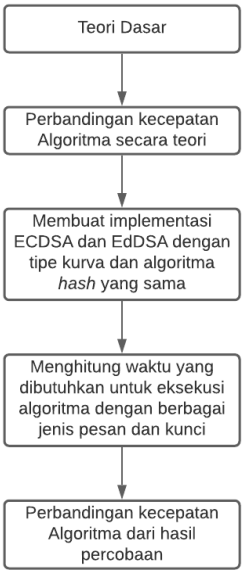
Perbedaan kedua yang perlu diperhatikan adalah bentuk kurva yang digunakan untuk proses ECDSA dan EdDSA. ECDSA biasa menggunakan kurva eliptik sederhana sementara EdDSA menggunakan kurva eliptik yang bertipe kurva Edward melengkung (*twisted Edwards curve*). Dengan demikian kecepatan kedua algoritma tersebut sulit untuk dibandingkan apabila kedua algoritma tersebut menggunakan kurva yang berbeda.

Maka dari itu, untuk memiliki perbandingan yang sesuai dan tidak bergantung pada bentuk kurva dan fungsi *hash*, maka disamakan tipe kurva dan fungsi *hash* yang digunakan. Untuk penelitian ini, *hash* menggunakan salah satu teknik SHA-2, yaitu SHA-256. Sementara itu, kurva eliptik yang digunakan dipilih dari kurva-kurva eliptik yang digunakan untuk ECDSA ataupun untuk EdDSA.

Setelah itu, dilakukan penghitungan waktu baik untuk proses pembangkitan tanda tangan (*signing*) dan proses verifikasi hasil

tanda tangan (*verifying*). Hasil yang didapatkan dari kedua proses tersebut akan dibandingkan sehingga dapat diputuskan algoritma mana yang lebih efisien.

Dari pertimbangan-pertimbangan tersebut, maka dibuat rancangan alur penelitian untuk mengukur algoritma yang lebih cepat antara ECDSA dan EdDSA. Berikut adalah rancangan eksekusi untuk membandingkan kecepatan eksekusi algoritma ECDSA dan EdDSA pada makalah ini.



Gambar 3. Rancangan eksekusi penelitian

Percobaan dimulai dengan meneliti teori dasar dari algoritma ECDSA dan EdDSA yang sudah dijelaskan pada bagian II. Setelah itu, mulai dari bagian berikutnya, akan dilakukan perbandingan kecepatan algoritma secara teori. Bagian ini menelaah tahap-tahap yang dibutuhkan untuk mengeksekusi algoritma tersebut lalu memutuskan algoritma mana yang lebih cepat secara teori untuk menunjukkan hipotesis.

Setelah itu, akan dibuat implementasi ECDSA dan EdDSA dengan kurva yang sama dan algoritma *hash* yang sama. Hal ini menurut penjelasan pada awal bagian ini, yaitu perbedaan waktu yang dibutuhkan untuk operasi dalam sebuah kurva dan waktu yang dibutuhkan untuk melakukan operasi *hash* akan berpengaruh pada perhitungan sehingga perbandingan yang dihasilkan kurang akurat.

Setelah mengimplementasikan algoritma ECDSA dan EdDSA dengan tipe kurva dan algoritma *has* yang sama, maka akan dihitung waktu yang dibutuhkan untuk melakukan eksekusi setiap algoritma tersebut dengan variasi pesan dan kunci. Setelah itu, akan dihitung rata-rata waktu yang dibutuhkan untuk eksekusi tersebut.

Setelah menghtiung rata-rata waktu yang dibutuhkan, maka akan dibandingkan kecepatan algoritma dari hasil percobaan sehingga dapat dibuktikan apakah hipotesis yang sudah dibuat tersebut benar atau salah.

Rata-rata waktu tersebut merupakan hal yang dibutuhkan untuk mencapai kesimpulan dari makalah ini, yaitu memutuskan algoritma yang lebih cepat antara ECDSA dan EdDSA yang keduanya merupakan algoritma berbasis kurva eliptik.

IV. Perbandingan Kecepatan Algoritma ECDSA dan EdDSA Secara Teori

Seperti yang dijelaskan pada bagian sebelumnya, bagian ini akan membahas perbandingan kecepatan algoritma apabila dibandingkan secara teoretis. Dari perbandingan ini, akan muncul hipotesis mengenai algoritma yang lebih cepat untuk melakukan proses pembangkitan tanda tangan (*signing*) dan verifikasi (*verifying*).

Jika dilihat dari algoritma pembangkitan tanda tangan (*signing*), didapati bahwa proses yang dilakukan oleh ECDSA lebih rumit dibandingkan proses yang dilakukan oleh EdDSA. Hal ini dilihat dari jumlah tahap dan kerumitan tahap yang dimiliki ECDSA dibandingkan EdDSA.

Hal utama yang membuat ECDSA lebih rumit dibandingkan dengan EdDSA adalah dibutuhkannya pengacakan bilangan untuk mendapatkan nilai *k* pada awal proses pembangkitan tanda tangan. Jika *k* yang dihasilkan berujung pada nilai *r* dan *s* menjadi 0, maka seluruh proses pembangkitan tanda tangan ECDSA diulangi dari awal. Di sisi lain, pembangkitan nilai *r* pada algoritma EdDSA bersifat deterministik sehingga tidak dibutuhkan sebuah usaha untuk mengambil bilangan bulat secara acak. Dengan demikian, proses akan lebih cepat.

Alasan kedua mengapa ECDSA lebih rumit dibandingkan dengan EdDSA adalah proses invers modulus. Proses ini tidak ada dalam algoritma pembangkitan tanda tangan EdDSA. Invers modulus pada bilangan yang besar membutuhkan waktu. Walaupun kemungkinan besar memiliki perbedaan waktu yang sedikit, keberadaan operasi invers modulus dapat memengaruhi kecepatan antara ECDSA dengan EdDSA.

Namun, di sisi lain, ECDSA memiliki 1 bagian di mana algoritma ini lebih cepat dibandingkan EdDSA, yaitu pada bagian *hash*. Algoritma ECDSA hanya melakukan proses *hash* sebanyak 1 kali sementara algoritma EdDSA melakukan operasi *hash* sebanyak lebih dari 1 kali pada saat pembangkitan tanda tangan digital.

Dari informasi di atas, maka dapat dibuat hipotesis bahwa dari segi proses pembangkitan tanda tangan, menggunakan algoritma ECDSA membutuhkan waktu yang lebih lama apabila dibandingkan dengan menggunakan algoritma EdDSA.

Jika dilihat dari algoritma verifikasi tanda tangan, dilihat algoritma ECDSA juga memiliki kerumitan yang lebih tinggi dibandingkan algoritma EdDSA dilihat dari jumlah tahap yang harus dikerjakan untuk verifikasi.

Proses verifikasi tanda tangan digital pada algoritma EdDSA hanya terdiri dari 4 tahap, yaitu proses *hash* sehingga menghasilkan *hash* yang sama pada saat proses pembangkitan tanda tangan, menghitung nilai *p1*, menghitung nilai *p2*, lalu melihat apakah kedua titik tersebut sama atau tidak pada kurva eliptik. Sementara itu, proses verifikasi tanda tangan digital pada algoritma ECDSA memiliki tahap yang lebih rumit, yang juga mencakup invers modulus dan nilai antara seperti *w*, u_1, dan u_2. Proses menghitung nilai antara ini akan memakan waktu sehingga proses verifikasi pada ECDSA akan dihitung lebih lambat

Dari informasi di atas, maka dapat dibuat hipotesis bahwa dari segi proses verifikasi tanda tangan, menggunakan algoritma ECDSA membutuhkan waktu yang lebih lama dibandingkan dengan menggunakan algoritma EdDSA.

Dari kedua hipotesis tersebut, maka diharapkan dari hasil percobaan, waktu yang dibutuhkan untuk eksekusi proses pembangkitan tanda tangan menggunakan EdDSA akan sedikit lebih cepat dibandingkan dengan menggunakan ECDSA. Sementara itu, waktu yang dibutuhkan untuk eksekusi proses verifikasi tanda tangan menggunakan EdDSA diharapkan lebih cepat secara signifikan dibandingkan dengan menggunakan ECDSA.

V. Rancangan Percobaan Perbandingan Kecepatan ECDSA dengan EdDSA

A. Pilihan Hash dan Kurva Eliptik

Sebelumnya, akan dijelaskan kembali bahwa algoritma ECDSA dan EdDSA yang dibandingkan di sini memiliki algoritma *hash* yang sama dan kurva eliptik yang sama untuk mencegah ketidakakuratan pengukuran akibat perbedaan waktu untuk melakukan *hash* dan melakukan operasi penjumlahan, penggandaan, dan perkalian (gabungan dari penjumlahan dan penggandaan) pada kurva eliptik.

Dikarenakan kebutuhan tersebut, maka sebelumnya dibutuhkan untuk memilih algoritma *hash* dan tipe kurva eliptik yang sama. Untuk algoritma *hash*, selama algoritma tersebut sama, maka tidak akan ada masalah. Untuk ini, digunakan salah satu algoritma *hash* SHA-256 Dikarenakan pada umumnya hasil *hash* dari algortima EdDSA dibagi menjadi 2 bagian, maka diperlukan sedikit modifikasi pada algoritma EdDSA sehingga tidak membutuhkan pemisahan bagian *hash*.

Untuk kurva eliptik, dapat digunakan kurva eliptik dengan jenis apa saja, termasuk kurva Edward melengkung (*twisted Edwards curve*) karena merupakan salah satu variasi dari kurva eliptik. Namun, dikarenakan kurva Edward melengkung (*twisted Edwards curve*) merupakan subset dari kurva eliptik, maka digunakan kurva eliptik sederhana yang merupakan bagian dari ECDSA.

B. Modifikasi Algoritma EdDSA

Dikarenakan diperlukan usaha untuk menyamakan algoritma *hash* dan kurva eliptik, maka diperlukan sedikit perubahan pada cara mengimplementasikan algoritma pada ECDSA atau EdDSA pada bagian algoritma yang berhubungan dengan *hash* atau kurva eliptik. Karena ECDSA bersifat lebih umum dibandingkan dengan EdDSA (terutama dari segi kurva eliptik yang digunakan), maka perubahan dilakukan pada EdDSA.

Perubahan pada kurva eliptik EdDSA tidak signifikan. Hal ini disebabkan operasi-operasi yang dilakukan pada kurva eliptik biasa dan kurva eliptik pada umumnya dan kurva Edward melengkung (*twisted Edwards curve*) mempunyai operasi yang sama, yaitu operasi penjumlahan, penggandaan, dan perkalian (gabungan dari penjumlahan dan penggandaan).

Sementara itu, perubahan pada algoritma *hash* membutuhkan sedikit modifikasi pada algoritma EdDSA jika ingin menggunakan algoritma *hash* yang sama. Modifikasi ini digunakan agar hasil dari proses *hash* tidak perlu dipisah menjadi 2 bagian selama eksekusi algoritma.

Algoritma EdDSA yang digunakan setelah modifikasi masih tetap melakukan proses yang mirip dengan algoritma EdDSA awal (yang biasa digunakan oleh Ed25519). Berikut adalah algoritma hasil modifikasi tersebut.

1. Pembangkitan kunci (*Key Generation*)
   a. Pilih sebuah *seed k*.
   b. Lakukan *hash* pada *k*. Hasil *hash* tersebut merupakan kunci privat *d*.
   c. Buat kunci publik Q dengan mengalikan basis kurva sebanyak *d* kali ($Q = dG$).
2. Pembangkitan tanda tangan (*Signing*)
   a. Lakukan *hash* pada kunci privat *d* yang digabungkan dengan pesan yang diberikan.
   $$r = Hash(d, M)$$
   b. Mencari titik *R*, yaitu titik basis *G* dikali sebanyak *r* kali ($R = rG$).
   c. Mencari nilai *h*, yaitu dengan menggabungkan hasil *hash* pesan (M), absis dari kunci publik (Q), dan absis dari titik *R*. Setelah itu, lakukan operasi *hash*.
   $$h = Hash(R, Q, M)$$
   d. Cari nilai *s*, yaitu $s = (r + h * d) \bmod n$.
   e. Titik *R* dan nilai *s* merupakan hasil tanda tangan digital.
3. Verifikasi tanda tangan (*Verifying*)
   a. Mencari nilai *h*, yaitu dengan menggabungkan hasil *hash* pesan (M), absis dari kunci publik (Q), dan absis dari titik *R*. Setelah itu, lakukan operasi *hash*.
   $$h = Hash(R, Q, M)$$
   b. Hitung *p1*, yaitu dengan mengalikan titik basis *G* sebanyak s kali ($p1 = sG$).
   c. Hitung *p2* dengan melakukan operasi pada kurva eliptik, yaitu $p2 = R + h * Q$.
   d. Tanda tangan dinilai valid apabila *p1 = p2*.

Seperti yang dilihat, perubahan algoritma EdDSA yang digunakan tidak mengubah cara kerja algoritma secara keseluruhan. Perubahan yang dilakukan hanya tidak memisahkan hasil pembangkitan kunci menjadi *a* dan *b*. Namun, seluruh hasil *hash* langsung dijadikan sebagai kunci *privat*. Setelah itu, kunci privat digunakan 2 kali pada saat pembangkitan tanda tangan. Yang pertama adalah pada saat mencari nilai *r* (seharusnya dikerjakan oleh *b*) dan kedua adalah saat mencari nilai *s*, yang sebelumnya juga dilakukan oleh kunci privat.

C. Rancangan Pengujian

Pengujian akan dilakukan pada proses pembangkitan tanda tangan (*signing*) dan proses verifikasi tanda tangan (*verifying*). Perbandingan kecepatan tidak dilakukan pada proses pembangkitan kunci dikarenakan proses yang dilakukan sangat mirip, yaitu memilih sebuah bilangan bulat yang akan dijadikan kunci privat. Setelah itu, dilakukan proses perkalian pada kurva eliptik untuk mendapatkan kunci publik. Perbedaan utama hanyalah pada algoritma ECDSA, angka tersebut dipilih langsung secara acak, sementara pada algoritma EdDSA, angka tersebut didapatkan dari melakukan operasi *hash* pada *seed*.

Dikarenakan waktu yang dibutuhkan untuk melakukan proses tanda tangan dan verifikasi sangat singkat dan pada proses ECDSA, lama waktu verifikasi juga bergantung pada bilangan acak yang dihasilkan pada saat proses mencari nilai acak *k*, maka akan dilakukan pengujian kecepatan untuk eksekusi proses pembangkitan tanda tangan dan verifikasi tanda tangan

sebanyak 10 kali, 50 kali, 100 kali, dan 500 kali dan dicari total waktu yang dibutuhkan.

Selain itu, dilakukan variasi panjang pesan dan kunci yang digunakan untuk melakukan proses pembangkitan tanda tangan dan verifikasi tanda tangan untuk memastikan kinerja dari kedua algoritma tersebut dari variasi pesan dan kunci.

Terakhir, digunakan juga variasi kurva eliptik untuk memastikan kinerja dari kedua algoritma tersebut pada kurva eliptik yang berbeda-beda.

VI. Perbandingan Kecepatan Algoritma ECDSA dan EdDSA Melalui Percobaan

Percobaan yang dilakukan menggunakan bahasa pemrograman Python versi 3.8. Operasi *hash* yang dilakukan menggunakan SHA-256 dengan jumlah bit yang variatif. Kurva eliptik yang digunakan adalah kurva elitpik sederhana yang digunakan pada ECDSA. Kurva eliptik ini akan digunakan sebagari representasi kurva eliptik pada algoritma ECDSA dan EdDSA.

A. Percobaan 1

Percobaan pertama adalah menggunakan kurva eliptik dengan parameter-parameter sebagai berikut.

- $a = 0$
- $b = 7$
- $p = 115792089237316195423570985008687907853269984665640564039457584007908834671663$
- $n = 115792089237316195423570985008687907852837564279074904382605163141518161494337$
- $G = (55066263022277343669578718895168534326250603453777594175500187360389116729240,32670510020758816978083085130507043184471273380659243275938904335757337482424)$

Pada percobaan tersebut, digunakan dibuat terlebih dahulu kunci privat, kunci publik, serta pesan yang diberikan, yaitu sebagai berikut.

- Kunci privat
  0xc14a5ec9ae144ad77eb63e15926c559f2f7d98482566b43f9d525f803c2efccc
- Kunci publik
  0xf3dfed68ffe86fdc8e68222c622df36b4091fd572c0792147e508ae2c570f623
  0x9345d4d00be64d1a65aca9378eb3d10e22462324a62e5c952507bb5ff102438f
- Pesan
  Aku ingin makan nasi uduk karena nasi uduk itu enak.

Dari percobaan tersebut, berikut adalah waktu yang dibutuhkan untuk melakukan eksekusi pada proses pembangkitan tanda tangan.

Tabel 1. Hasil percobaan 1 proses pembangkitan tanda tangan

Jumlah Eksekusi	10	50	100	500
ECDSA	0,262 sec	1,319 sec	2,598 sec	13,02 sec
EdDSA	0,261 sec	1,312 sec	2,589 sec	12,96 sec

Percobaan pertama kemudian dicoba untuk proses verifikasi tanda tangan. Hasilnya dilihat pada tabel berikut.

Tabel 2. Hasil percobaan 1 proses verifikasi tanda tangan

Jumlah Eksekusi	10	50	100	500
ECDSA	0,512 sec	2,580 sec	5,169 sec	25,92 sec
EdDSA	0,510 sec	2,572 sec	5,140 sec	25,82 sec

Dari hasil kedua percobaan tersebut, dapat disimpulkan bahwa pada umumnya algoritma EdDSA lebih cepat dibandingkan algoritma ECDSA. Walaupun demikian, selisih kecepatan antara kedua algoritma tersebut tidak terlalu signifikan seperti hipotesis pada selisih kecepatan EdDSA dan ECDSA pada proses verifikasi. Namun, selisih kecepatan algoritma EdDSA dan ECDSA pada proses verifikasi masih lebih besar daripada saat proses penandatanganan.

B. Percobaan 2

Untuk mencoba kemungkinan lain, maka akan digunakan ukuran pesan yang lebih besar dan melihat selisih antara kecepatan kedua algoritma tersebut. Pesan yang digunakan adalah sebagai berikut.

> Miusov, as a man man of breeding and deilcacy, could not but feel some inwrd qualms, when he reached the Father Superior's with Ivan: he felt ashamed of havin lost his temper. He felt that he ought to have disdaimed that despicable wretch, Fyodor Pavlovitch, too much to have been upset by him in Father Zossima's cell, and so to have forgotten himself. "Teh monks were not to blame, in any case," he reflceted, on the steps. "And if they're decent people here (and the Father Superior, I understand, is a nobleman) why not be friendly and courteous withthem? I won't argue, I'll fall in with everything, I'll win them by politness, and show them that I've nothing to do with that Aesop, thta buffoon, that Pierrot, and have merely been takken in over this affair, just as they have."

Kunci privat dan kunci publik yang akan digunakan sama. Selain itu, parameter-parameter kurva yang digunakan sama. Dari percobaan tersebut, berikut adalah waktu yang dibutuhkan untuk melakukan eksekusi pada proses pembangkitan tanda tangan.

Tabel 3. Hasil percobaan 2 proses pembangkitan tanda tangan

Jumlah Eksekusi	10	50	100	500
ECDSA	0,286 sec	1,339 sec	2,635 sec	13,27 sec
EdDSA	0,273 sec	1,417 sec	2,617 sec	13,21 sec

Setelah dilakukan proses pembangkitan tanda tangan, dilakukan juga proses verifikasi. Hasilnya sebagai berikut.

Tabel 4. Hasil percobaan 2 proses verifikasi tanda tangan

Jumlah Eksekusi	10	50	100	500
ECDSA	0,520 sec	2,584 sec	5,210 sec	26,10 sec
EdDSA	0,514 sec	2,574 sec	5,189 sec	25,82 sec

C. Percobaan 3

Percobaan ketiga menggunakan kunci privat, kunci publik, dan pesan yang sama dengan percobaan 1. Yang diubah pada percobaan ketiga adalah parameter dari kurva. Adapun kurva masih menggunakan kurva eliptik sederhana yang digunakan pada ECDSA dengan parameter yang berukuran lebih kecil, yaitu sebagai berikut.

- a = -3
- b = 245515554600894381774029391519745178476910 8058161191238065
- p = 627710173538668076383578942320766641608390 8700390324961279
- n = 627710173538668076383578942317605901376719 4773182842284081
- G = (602046282375688656758213480587526111916698976636884684818, 174050332293622031404857552280219410364023488927386650641)

Dengan kurva sederhana, selain dapat mengukur perbandingan waktu, terdapat tujuan lain. Dengan kurva yang lebih sederhana, diharapkan waktu yang dibutuhkan untuk komputasi lebih rendah. Dengan demikian, diharapkan dapat dibuktukan bahwa dibutuhkan kurva yang sama untuk menguji kedua model ini. Adapun hasil yang didapatkan untuk proses pembangkitan tanda tangan adalah sebagai berikut.

Tabel 5. Hasil percobaan 3 proses pembangkitan tanda tangan

Jumlah Eksekusi	10	50	100	500
ECDSA	0,154 sec	0,725 sec	1,419 sec	7,011 sec
EdDSA	0,140 sec	0,696 sec	1,397 sec	6,975 sec

Sama dengan percobaan pertama dan percobaan kedua, dilakukan juga pengujian pada proses verifikasi tanda tangan. Berikut adalah hasilnya.

Tabel 6. Hasil percobaan 3 proses verifikasi tanda tangan

Jumlah Eksekusi	10	50	100	500
ECDSA	0,307 sec	1,463 sec	2,964 sec	14,26 sec
EdDSA	0,279 sec	1,392 sec	2,780 sec	13,90 sec

D. Analisis Hasil Percobaan

Dari hasil percobaan 3 dibandingkan percobaan 1, ditemukan bahwa kompleksitas sebuah kurva eliptik sangat mempengaruhi waktu eksekusi. Dengan demikian, sebelum membandingkan kecepatan pembangkitan tanda tangan dan verifikasi tanda tangan dari kedua algoritma, diperlukan penyamaan kurva eliptik yang digunakan.

Selain itu, dari hasil percobaan 2 dibandingkan dengan percobaan 1, ditemukan bahwa panjang pesan tidak terlalu mempengaruhi waktu pembangkitan dan verifikasi tanda tangan. Selisih waktu yang terjadi antara percobaan 1 dan percobaan 2 merupakan selisih waktu yang diperlukan untuk SHA-256 untuk melakukan *hashing* pada pesan.

Selain perubahan pada pesan dan kompleksitas kurva, diperlukan juga perbandingan pada kecepatan eksekusi EdDSA dan ECDSA yang merupakan tujuan dari penelitian ini. Jika dibandingkan dari hipotesis, maka hipotesis pertama cukup tepat sasaran karena selisih antara kecepatan pembangkitan tanda tangan algoritma ECDSA dengan EdDSA cukup kecil, namun algoritma EdDSA sedikit lebih cepat dibandingkan ECDSA. Sementara itu, hipotesis kedua dinilai kurang benar. Hal ini disebabkan selisih waktu yang dihasilkan oleh algoritma ECDSA dan EdDSA pada saat melakukan verifikasi tanda tangan hanya berkisar antara 0,1 detik per 100 tanda tangan. Dengan demikian, tidak terdapat selisih yang signifikan antara kecepatan verifikasi tanda tangan algoritma ECDSA dan EdDSA. Walaupun demikian, algoritma EdDSA masih lebih cepat dibandingkan ECDSA dalam hal verifikasi tanda tangan.

Dari kedua hal tersebut, dapat disimpulkan bahwa algoritma ECDSA pada umumnya lebih lambat dibandingkan algoritma EdDSA pada saat proses pembangkitan tanda tangan dan verifikasi tanda tangan digital. Namun, hal tersebut sangat bergantung pada kompleksitas kurva eliptik.

VII. KESIMPULAN

Kesimpulan yang didapatkan adalah algoritma EdDSA pada umumnya lebih cepat dibandingkan dengan algoritma ECDSA pada saat proses pembangkitan tanda tangan dan proses verifikasi tanda tangan. Selisih waktu yang diberikan antara kedua algoritma tersebut untuk proses pembangkitan tanda tangan dan verifikasi tanda tangan tidak terlalu jauh, yaitu sekitar 0,001 detik hingga 0,002 detik per operasi pembangkitan atau verifikasi tanda tangan. Namun, algoritma EdDSA masih lebih cepat dibandingkan ECDSA.

Selain itu, untuk membandingkan kecepatan kedua algoritma tersebut, diperlukan kurva eliptik yang sama karena kompleksitas kurva eliptik sangat mempengaruhi waktu yang dibutuhkan untuk pembangkitan dan verifikasi tanda tangan digital.

VIII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan YME atas penyertaannya selama menuliskan makalah ini. Penulis juga mengucapkan terima kasih kepada orang tua penulis atas dukungannya dalam menulis makalah ini. Tak lupa penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir sebagai dosen mata kuliah Kriptografi yang mengajarkan pengetahuan yang dibutuhakan agar penulis mampu menuliskan makalah ini. Terakhir, penulis mengucapkan terima kasih kepada teman-teman penulis atas dukungannya selama menuliskan makalah ini.

REFERENSI

[1] A. J. Menezes, P. C. Van Oorschot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[2] W. Diffie, "The first ten years of public-key cryptography," in *Proceedings of the IEEE*, vol. 76, no. 5, pp. 560-577, May 1988, doi: 10.1109/5.4442.

[3] Koblitz, N. (1987). *Elliptic Curve Cryptosystems*. *4*(177), 203–209.

[4] R. Munir, "Elliptic Curve Cryptography (ECC) Bagian 2," Teknik Informatika STEI - ITB, Bandung.

[5] Bernstein, D. J., Birkner, P., Jove, M., Lange, T., & Peters, C. (2008). Twisted Edwards Curves, 5023 LNCS, 389-405. https://doi.org/10.1007/978-3-540-68164-9_26.

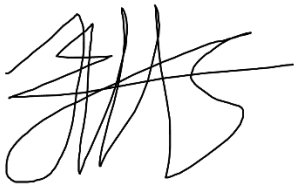
[6] Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm Validation System (ECDSAVS).

International Journal of Information Security, *1*(1), 36–63. http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf.
[7] Josefsson, S.; Liusvaara, I. (January 2017). *Edwards-Curve Digital Signature Algorithm (EdDSA)*. Internet Engineering Task Force. doi:10.17487/RFC8032.

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, 3 Desember 2020



Jan Meyer Saragih
13517131