Perbandingan Kinerja Algoritma *Digital Signature* berbasis RSA & *Elliptic Curve*

Yoel Susanto - 13517014
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517014@std.stei.itb.ac.id

***Abstrak*—Untuk bertukar informasi secara aman, digunakan skema tanda tangan digital berbasis algoritma kriptografi. Dua algoritma tanda tangan digital yang umum digunakan adalah algoritma RSA dan ECDSA. Dengan volume komunikasi yang begitu besar, keunggulan kinerja salah satu algoritma yang dipilih dapat menghasilkan efisiensi yang signifikan. Dalam tulisan ini, akan dilakukan analisis terhadap perbedaan kinerja antara algoritma tanda tangan digital RSA dan ECDSA.**

***Keywords*—RSA, ECDSA, kinerja, tanda tangan digital.**

I. Pendahuluan

Saat ini, internet digunakan untuk memfasilitasi banyak jenis kegiatan diantaranya mengkonsumsi konten, mencari suatu informasi dan melakukan transaksi pembelian atau penjualan. Semua kegiatan tersebut melibatkan proses pertukaran informasi antara dua atau lebih pihak. Dalam proses pertukaran informasi, data akan disalurkan melalui jaringan internet yang menghubungkan kedua belah pihak yang terlibat, pada saat ini lah informasi yang dipertukarkan dapat berada dalam ancaman.

Terdapat dua jenis informasi berdasarkan tingkat kerahasiaannya, terdapat informasi non-sensitif dan informasi sensitif. Informasi non-sensitif dapat berupa isi dari sebuah halaman publik yang dapat diakses oleh semua orang. Contoh dari informasi sensitif adalah kata sandi dari akun seseorang, informasi yang dipertukarkan saat seseorang menggunakan internet *banking* dan lain sebagainya.

Untuk menjaga kerahasiaan informasi sensitif, maka komunikasi dan pertukaran informasi di Internet harus dilakukan dengan skema Hypertext Transfer Protocol Secure (HTTPS). Dalam protokol HTTPS, digunakan algoritma *digital signature* untuk melakukan pertukaran kunci sesi. Semakin besar volume pertukaran informasi yang dilakukan, semakin penting untuk meninjau kinerja dari algoritma tanda tangan digital yang digunakan. Dalam tulisan ini, akan dilakukan analisis terhadap perbedaan kinerja antara algoritma tanda tangan RSA dan ECDSA.

II. Dasar Teori

A Digital Signature

Digital Signature atau tanda tangan digital adalah sebuah mekanisme untuk melakukan verifikasi terhadap keaslian sebuah pesan. Terdapat beberapa tipe implementasi dari algoritma tanda tangan digital namun umumnya beroperasi pada dasar yang sama yaitu menggunakan konsep matematika dan fungsi hash satu arah untuk mendeteksi apakah isi sebuah pesan telah mengalami manipulasi atau masih merupakan pesan asli.

Terdapat tiga fungsi utama dalam melakukan verifikasi tanda tangan digital:

1 *Authentication*
Tanda tangan digital memberikan keyakinan pada penerima pesan bahwa pesan benar diubah oleh pihak yang berotoritas. Saat melakukan verifikasi tanda tangan digital, pesan diverifikasi dengan menggunakan kunci publik yang disediakan oleh pengirim pesan. Apabila pesan berhasil diverifikasi dengan menggunakan kunci publik tersebut, maka penerima pesan dapat yakin bahwa pesan yang diterima benar-benar dibuat oleh pengirim yang mempunyai kunci privat. Hal ini disebabkan kunci publik dan kunci privat dari sebuah mekanisme tanda tangan digital bersifat pasangan dan tidak akan bekerja apabila nilai dari kunci publik atau kunci privat nya diubah.

2 *Data Integrity*
Mekanisme tanda tangan digital juga memungkinkan penerima pesan untuk melakukan pemeriksaan terhadap keaslian dari isi pesan. Hal ini penting dilakukan karena isi pesan yang dikirimkan melalui medium tertentu beresiko mengalami rekayasa sehingga isi pesan yang diterima sudah berbeda dengan pesan asli yang dikirimkan. Dalam domain tertentu seperti dunia perbankan, hal ini menjadi sangat krusial.

3 *Non-Repudiation*
Non repudiation dalam mekanisme verifikasi tanda tangan digital artinya seseorang tidak dapat melakukan penyangkalan terhadap pesan yang pernah dikirimkan. Hal ini berbasis pada poin pertama yaitu sebuah pesan hanya dapat dikirimkan oleh pengirim yang memiliki kunci privat, oleh sebab itu identitas pengirim dapat dipastikan dan pengirim tidak dapat menyangkal hal tersebut.

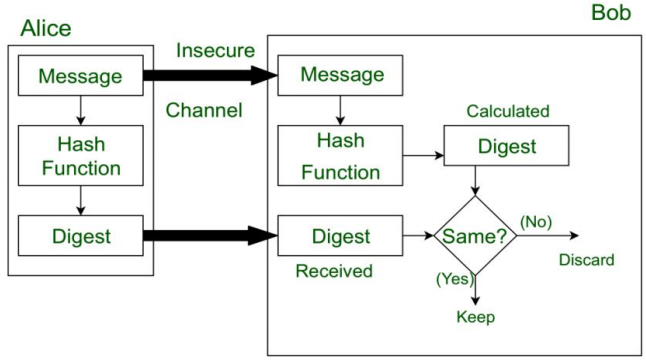
B Algoritma RSA

Algoritma RSA merupakan algoritma enkripsi kunci publik.

Dengan demikian, algoritma RSA menggunakan kunci yang berbeda saat melakukan proses enkripsi dan dekripsi. Terdapat pasangan kunci publik dan kunci privat pada algoritma RSA. Sesuai dengan namanya, kunci publik adalah bagian kunci yang bebas tersedia kepada publik sedangkan kunci privat adalah bagian kunci yang hanya diketahui oleh pengirim pesan. Apabila kunci privat diketahui oleh pihak eksternal, maka keseluruhan skema algoritma RSA sudah tidak aman.

Skema kerja algoritma RSA dalam pengiriman pesan oleh Alice kepada Bob adalah sebagai berikut:

1. Alice memilih dua buah angka prima yang besar yaitu p dan q.
2. Alice menghitung n = p * q.
3. Alice memilih sebuah angka e sebagai kunci publik yang bukan merupakan faktor dari (p-1) * (q-1). Nilai e diberikan oleh Alice kepada Bob
4. Alice memilih sebuah angka d sebagai kunci private yang memenuhi persamaan (d*e) mod (p1)*(q-1)=1
5. Alice menghitung message digest dari pesan yang akan dikirimkan, message digest ini lalu dienkripsi dengan kunci private yang tidak dibagikan kepada siapapun
6. Bob menerima pesan yang dikirimkan oleh Alice lalu melakukan dekripsi message digest dengan kunci publik yang diberikan oleh Alice.
7. Setelah melakukan dekripsi terhadap message digest, Bob menghitung message digest berdasarkan isi pesan yang diterimanya. Apabila kedua message digest sama, maka pesan telah terverifikasi.
8. Apabila message digest yang diterima Bob berbeda dengan hasil perhitungannya maka pesan yang diterima Bob sudah mengalami rekayasa.



Gambar 1. Skema Algoritma RSA

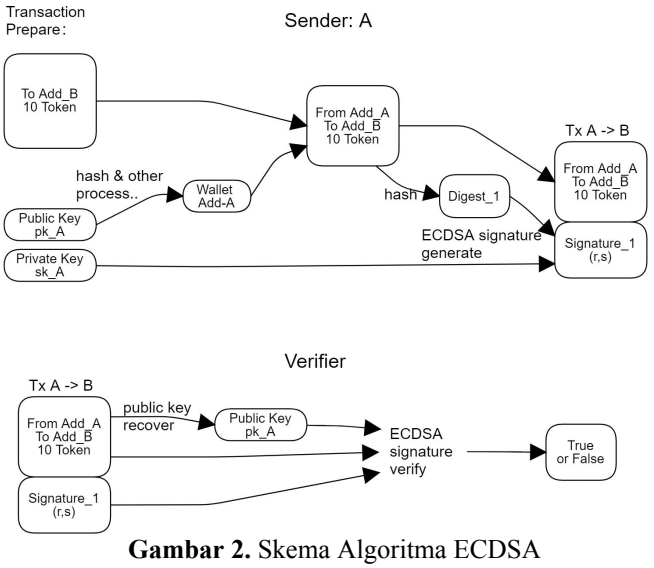
C Algoritma ECDSA

Algoritma ECDSA adalah algoritma tanda tangan digital yang berbasis pada konsep matematika *elliptic curve*. *Elliptic curve* memiliki persamaan $y^2=x^3+ax+b$ dengan syarat $4a^3+27b^2\neq0$. Salah satu keunggulan dari algoritma ECDSA dibandingkan dengan RSA adalah untuk menghasilkan tingkat keamanan yang sama, kebutuhan komputasi yang diperlukan oleh algoritma ECDSA lebih kecil dibandingkan dengan algoritma RSA. Berikut merupakan cara kerja algoritma ECDSA pada pihak pengirim pesan:

1. Alice dan Bob membuat kesepakatan tentang parameter yang digunakan pada algoritma ECDSA. Parameter yang perlu disepakati adalah persamaan kurva yang akan digunakan, titik G sebagai basis dari kurva dan juga sebuah angka n dimana n x G = O, dengan O merupakan titik pada tak terhingga. Selain itu, Alice juga menentukan sebuah nilai kunci private dengan simbol pvt.
2. Alice melakukan perhitungan message digest e dengan menggunakan fungsi hash tertentu.
3. Alice memilih sebuah nilai k berupa integer dengan nilai antara 1 sampai dengan n-1.
4. Alice menghitung sebuah titik pada kurva (x,y) dimana titik tersebut merupakan perkalian antara k dengan titik G yang telah disepakati di awal.
5. Alice lalu menghitung sebuah nilai r = x mod n.
6. Selanjutnya Alice menghitung sebuah nilai $s = k^{-1}*(z+r*pvt) \bmod n$ dengan z merupakan beberapa bits paling kiri dari nilai e.
7. Pasangan nilai r dan s yang dihitung oleh Alice menjadi tanda tangan dari pesan dan akan dikirimkan bersamaan dengan pesan.

Setelah Bob menerima pesan yang dikirimkan oleh Alice, terdapat serangkaian tahapan yang harus dilakukan oleh Bob untuk memverifikasi pesan yang diterimanya benar berasal dari Alice:

1. Bob melakukan pemeriksaan apakah pasangan nilai r dan s yang diterima dalam pesan Alice berada antara nilai 1 sampai n-1. Jika tidak berada pada batasan tersebut maka tidak perlu dilakukan perhitungan lebih jauh karena dapat dipastikan pasangan nilai tersebut merupakan pasangan yang tidak valid.
2. Bob melakukan perhitungan message digest dengan menggunakan fungsi hash yang sama dengan yang digunakan oleh Alice. Message digest yang dihasilkan disimbolkan dengan e.
3. Bob melakukan kalkulasi nilai $u1=z*s^{-1} \bmod n$ dan $u2=r*s^{-1} \bmod n$ dengan z merupakan beberapa bit terkiri dari nilai e.
4. Selanjutnya Bob melakukan kalkulasi titik x,y=u1 x G + u2 x pub. Pub merupakan kunci publik Alice yang dapat diakses secara bebas.
5. Tanda tangan pesan terverifikasi apabila terdapat r bernilai sama dengan x mod n.



Gambar 2. Skema Algoritma ECDSA

D Perbandingan Keamanan RSA dan ECDSA

Algoritma RSA dan ECDSA memiliki prinsip dasar matematis yang berbeda. Oleh sebab itu, keamanan yang dihasilkan oleh masing-masing algoritma pun berbeda. Berdasarkan publikasi dari National Institute of Standards and Technology berikut merupakan tingkat keamanan dari algoritma RSA dan ECDSA pada berbagai panjang kunci yang digunakan:

Security Strength	RSA Key Size	ECDSA Key Size
80	1024	160-223
112	2048	224-255
128	3072	256-383
192	7680	384-511
256	15360	512+

Gambar 3. Perbandingan Keamanan Panjang Kunci Algoritma RSA & ECDSA

Melalui tabel diatas, pada tingkat keamanan yang sama ECDSA menggunakan panjang kunci yang jauh lebih pendek oleh sebab itu kinerja dari algoritma ECDSA diharapkan akan lebih baik dari RSA.

III. Implementasi

Dalam melakukan eksperimen, dibuat dua buah program sederhana dalam bahasa python. Program pertama berisi kode untuk melakukan simulasi eksekusi algoritma RSA. Program kedua berisi kode untuk melakukan simulasi terhadap algoritma ECDSA. Pada masing-masing algoritma, digunakan fungsi hash SHA256. Pada makalah ini penulis menggunakan kakas pycryptodome untuk mendapatkan implementasi paling optimal dari masing-masing algoritma.

Program Algoritma RSA:

```
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature.pkcs1_15 import
PKCS115_SigScheme
from time import time


def sign(message, keyPair):
    hash = SHA256.new(message)
    signer = PKCS115_SigScheme(keyPair)
    signature = signer.sign(hash)
    return signature


def verify(message, pubKey, signature):
    hash = SHA256.new(message)
    verifier = PKCS115_SigScheme(pubKey)
    verifier.verify(hash, signature)


def simulate(iteration, keyLength, message):
    def run():
        keyPair = RSA.generate(bits=keyLength)
        publicKey = keyPair.publickey()

        signature = sign(message, keyPair)
        verify(message, publicKey, signature)

    start = time()
    for i in range(iteration):
        run()
    end = time()
    durationInMS = (end-start) * 1000
    print("RSA: {} iteration, took: {:.2f}ms. Individual
Average: {:.2f}ms".format(
        iteration, durationInMS, durationInMS/iteration))


message_10bytes = b'0123456789'
message_100bytes = 10*message_10bytes
message_1000bytes = 10*message_100bytes
simulate(10, 3072, message_10bytes)
```

Program Algoritma ECDSA:

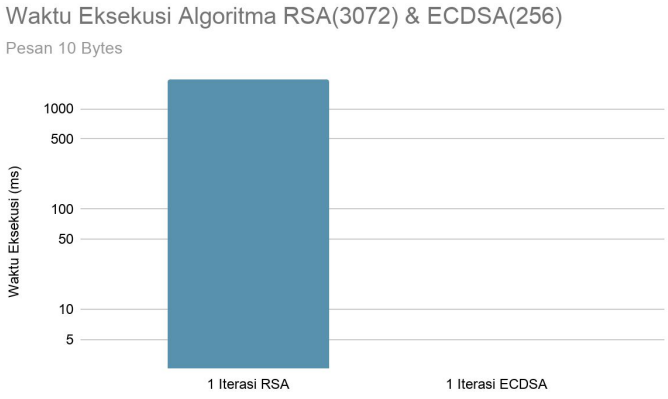
```
from Crypto.Hash import SHA256
from Crypto.PublicKey import ECC
from Crypto.Signature import DSS
from time import time


def sign(message, key):
    digest = SHA256.new(message)
    signer = DSS.new(key, 'fips-186-3')
    signature = signer.sign(digest)
    return signature


def verify(message, key, signature):
    digest = SHA256.new(message)
    verifier = DSS.new(key, 'fips-186-3')
    verifier.verify(digest, signature)


def simulate(iteration, keyLength, message):
    def run():
        key = ECC.generate(curve='P-{}'.format(keyLength))
        signature = sign(message, key)
        verify(message, key, signature)

    start = time()
    for i in range(iteration):
        run()
    end = time()
    durationInMS = (end-start) * 1000
    print("ECDSA: {} iteration, took: {:.2f}ms. Individual
Average: {:.2f}ms".format(
        iteration, durationInMS, durationInMS/iteration))


message_10bytes = b'0123456789'
message_100bytes = 10*message_10bytes
```

```
message_1000bytes = 10*message_100bytes
simulate(10, 256, message_10bytes)
```

IV. Eksperimen

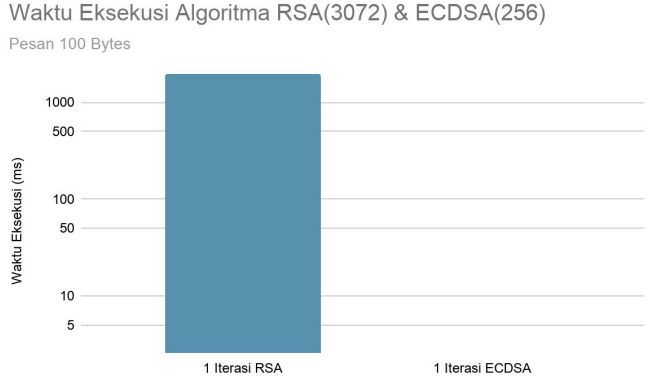
Eksperimen dilakukan dengan mengeksekusi algoritma RSA dan ECDSA sebanyak 10 iterasi per kasus uji. Eksperimen dilakukan pada berbagai tingkat panjang kunci untuk mensimulasikan kerja algoritma pada tingkat keamanan tertentu serta panjang pesan yang berbeda-beda. Hardware yang digunakan untuk melakukan eksperimen adalah processor 7th Generation Intel® Core™ i5-7200U dengan 20GB RAM DDR4.

A. Tingkat Keamanan 128 dengan Pesan 10 Bytes
Pada kasus ini, algoritma RSA menggunakan panjang kunci 3072 bits dan ECDSA menggunakan panjang kunci 256 bits. Algoritma RSA melakukan eksekusi sepanjang 1936.4ms dan ECDSA sepanjang 2.56ms. Pesan yang diuji berukuran 10 bytes. Dalam kasus uji ini, algoritma ECDSA lebih cepat 774.56x.



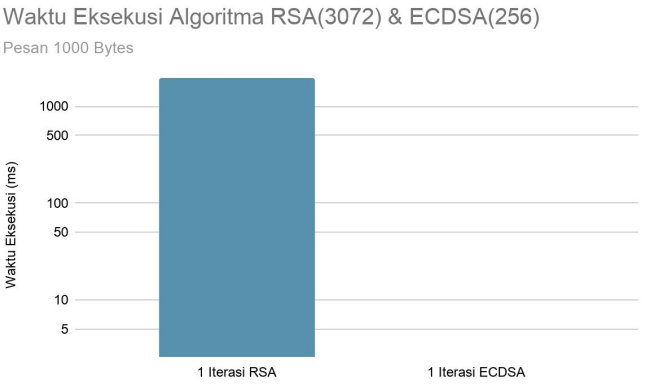
Gambar 4. Perbandingan Waktu Eksekusi Algoritma RSA dan ECDSA pada Panjang Pesan 10 Bytes

B. Tingkat Keamanan 128 dengan Pesan 100 Bytes
Pada kasus ini, algoritma RSA menggunakan panjang kunci 3072 bits dan ECDSA menggunakan panjang kunci 256 bits. Algoritma RSA melakukan eksekusi sepanjang 1958.6ms dan ECDSA sepanjang 2.71ms. Pesan yang diproses berukuran 100 bytes. Dalam kasus uji ini, algoritma ECDSA lebih cepat 722.7x.



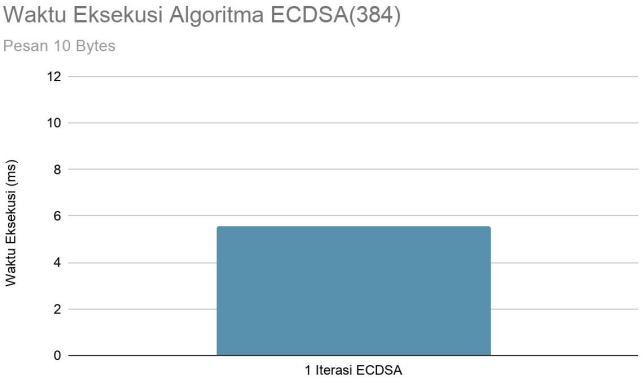
Gambar 5. Perbandingan Waktu Eksekusi Algoritma RSA dan ECDSA pada Panjang Pesan 100 Bytes

C. Tingkat Keamanan 128 dengan Pesan 1000 Bytes
Pada kasus ini, algoritma RSA menggunakan panjang kunci 3072 bits dan ECDSA menggunakan panjang kunci 256 bits. Algoritma RSA melakukan eksekusi sepanjang 1985.6ms dan ECDSA sepanjang 3.81ms. Pesan yang diproses berukuran 1000 bytes. Dalam kasus uji ini, algoritma ECDSA lebih cepat 521.0x.



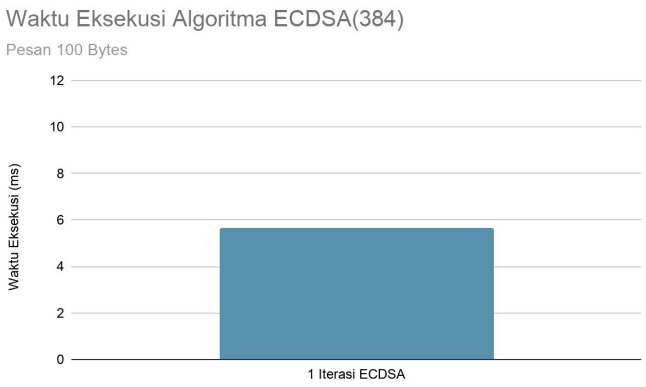
Gambar 6. Perbandingan Waktu Eksekusi Algoritma RSA dan ECDSA pada Panjang Pesan 1000 Bytes

D. Tingkat Keamanan 192 dengan Pesan 10 Bytes
Pada kasus ini, algoritma RSA menggunakan panjang kunci 7680 bits namun program berjalan sangat lama dan tidak dapat selesai. ECDSA menggunakan panjang kunci 384 bits dan harus menggunakan fungsi hash SHA512. Eksekusi algoritma ECDSA membutuhkan waktu sepanjang 5.55ms. Pesan yang diproses berukuran 10 bytes.



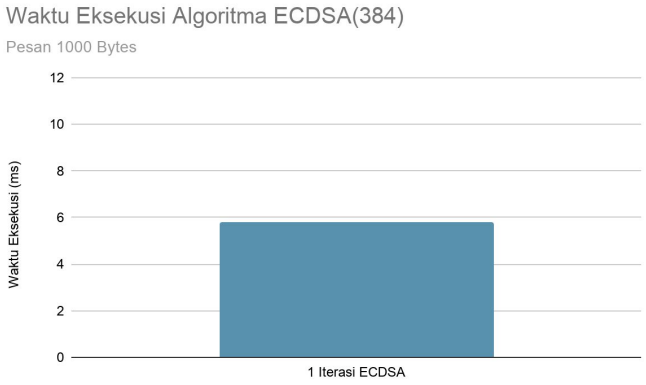
Gambar 7. Waktu Eksekusi Algoritma ECDSA pada Panjang Pesan 10 Bytes

E. Tingkat Keamanan 192 dengan Pesan 100 Bytes
ECDSA menggunakan panjang kunci 384 bits dan menggunakan fungsi hash SHA512. Eksekusi algoritma ECDSA membutuhkan waktu sepanjang 5.65ms. Pesan yang diproses berukuran 100 bytes.



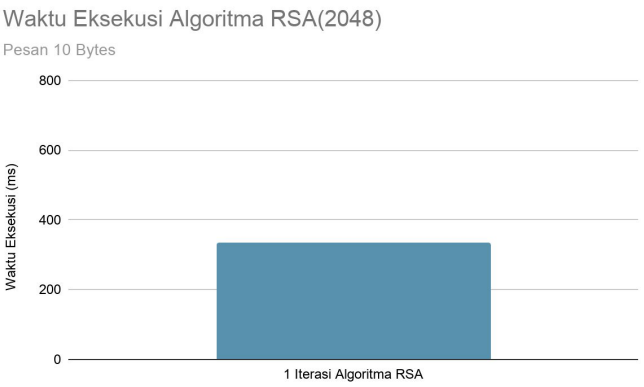
Gambar 8. Waktu Eksekusi Algoritma ECDSA pada Panjang Pesan 100 Bytes

F. Tingkat Keamanan 192 dengan Pesan 1000 Bytes
ECDSA menggunakan panjang kunci 384 bits dan menggunakan fungsi hash SHA512. Eksekusi algoritma ECDSA membutuhkan waktu sepanjang 5.79ms. Pesan yang diproses berukuran 1000 bytes.



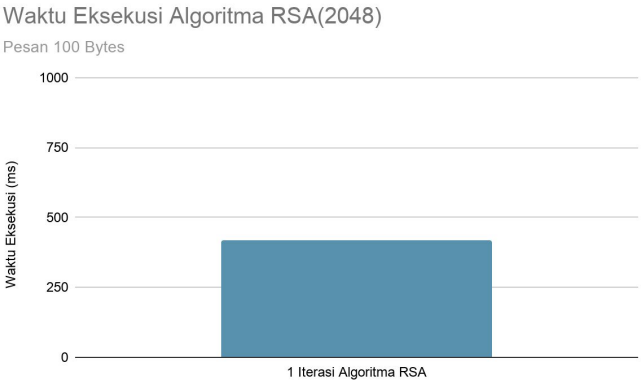
Gambar 9. Waktu Eksekusi Algoritma ECDSA pada Panjang Pesan 1000 Bytes

G. Tingkat Keamanan 112 dengan Pesan 10 Bytes
Algoritma RSA menggunakan kunci dengan panjang 2048 bits. ECDSA menggunakan panjang kunci 224 bits namun belum ada kurva yang diimplementasikan pada panjang kunci tersebut. Oleh sebab itu eksperimen tingkat keamanan 112 hanya dilakukan dengan algoritma RSA. Eksekusi algoritma RSA membutuhkan waktu sepanjang 334.87ms. Pesan yang diproses berukuran 10 bytes.



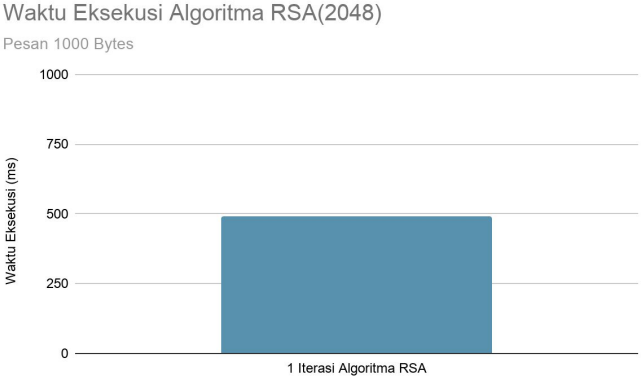
Gambar 10. Waktu Eksekusi Algoritma RSA pada Panjang Pesan 10 Bytes

H. Tingkat Keamanan 112 dengan Pesan 100 Bytes
Algoritma RSA menggunakan kunci dengan panjang 2048 bits. Eksekusi algoritma RSA membutuhkan waktu sepanjang 420.65ms. Pesan yang diproses berukuran 100 bytes.



Gambar 11. Waktu Eksekusi Algoritma RSA pada Panjang Pesan 100 Bytes

I. Tingkat Keamanan 112 dengan Pesan 1000 Bytes
Algoritma RSA menggunakan kunci dengan panjang 2048 bits. Eksekusi algoritma RSA membutuhkan waktu sepanjang 489.80ms. Pesan yang diproses berukuran 1000 bytes.



Gambar 12. Waktu Eksekusi Algoritma RSA pada Panjang Pesan 1000 Bytes

V. Analisis

Pada ketiga eksperimen pertama, algoritma RSA dan ECDSA dijalankan untuk menghasilkan tingkat keamanan yang sama. Karena perbedaan mendasar pada konsep matematika yang digunakan, algoritma RSA membutuhkan panjang kunci yang jauh lebih besar untuk mencapai tingkat keamanan yang sama dengan algoritma ECDSA. Hal ini berdampak pada waktu komputasi algoritma RSA yang sangat lama dibanding algoritma RSA. Secara rata-rata algoritma ECDSA lebih cepat 672.7x dibandingkan dengan algoritma ECDSA.

Pada eksperimen D, E dan F, algoritma ECDSA dijalankan pada panjang kunci 384. Meskipun panjang kunci yang digunakan dibuat semakin panjang, waktu eksekusi algoritma ECDSA tidak jauh berbeda dengan pada eksperimen A, B dan C. Oleh sebab itu jika dibutuhkan keamanan dengan tingkat yang lebih tinggi, menggunakan panjang kunci yang lebih tinggi untuk algoritma ECDSA tidak akan menjadi masalah.

Pada eksperimen G, H dan I, algoritma RSA dijalankan pada panjang kunci 2048. Waktu rata-rata eksekusi pada ketiga eksperimen tersebut adalah 415.1 ms. Pada eksperimen A,B dan C dimana algoritma RSA dijalankan dengan panjang kunci 3072, rata-rata waktu eksekusi adalah 1960.2ms. Panjang kunci pada algoritma RSA sangat berpengaruh terhadap kinerja algoritma tersebut. Hal ini membuat penggunaan panjang kunci yang lebih besar tidak selalu menjadi solusi yang praktikal.

VI. Kesimpulan

Algoritma RSA dan algoritma ECDSA adalah algoritma yang dapat digunakan untuk melakukan skema tanda tangan digital. Algoritma ECDSA mempunyai efisien yang sangat tinggi, pada tingkat keamanan 128, algoritma ECDSA mempunyai keunggulan waktu eksekusi sebesar 672.7x dibandingkan dengan algoritma RSA. Dengan demikian, proses tanda tangan yang masih menggunakan algoritma RSA dapat menikmati peningkatan efisiensi yang masif jika melakukan migrasi kepada algoritma ECDSA. Dengan sifat algoritma ECDSA yang begitu efisien, algoritma ini juga sangat cocok digunakan dalam perangkat yang tidak mempunyai sumber daya komputasi yang besar misalkan perangkat IoT dan smartphone. Pada server yang melayani request di Internet, algoritma ECDSA dapat memungkinkan process session key exchange yang lebih cepat sehingga dapat melayani lebih banyak pengguna.

VII. Ucapan Terima Kasih

Penulis mengucapkan syukur dan terima kasih kepada Tuhan Yang Maha Esa atas berkat dan penyertaan Nya dalam proses penulis menempuh pendidikan di Institut Teknologi Bandung.

Penulis juga mengucapkan terima kasih kepada orang tua dan keluarga penulis yang memberikan doa dan dukungan kepada penulis selama melalui proses perkuliahan. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir atas dedikasi dan ilmu yang diajarkan selama penulis menempuh mata kuliah Kriptografi.

Tidak lupa, penulis juga mengucapkan terima kasih kepada teman-teman seperjuangan Informatika 2017 yang telah memberikan dukungan untuk menyelesaikan berbagai tugas perkuliahan terutama makalah ini.

Referensi

1 NIST. “Recommendation for Key Management”, May 2020.
2 Ali. Al Imem. “Comparison and Evaluation of Digital Signature Scheme Employed in NDN Network”, June 2015.
3 Levi. Sharon. “Performance and Security of ECDSA”. 2015.
4 Milanov. Evgeny, “The RSA Algorithm”, June 2009.
5 Johnson, D., Menezes, A. & Vanstone, S. “The Elliptic Curve Digital Signature Algorithm (ECDSA)”, August 2001.
6 Nakov. Svetlin, “RSA: Sign / Verify - Examples”, https://cryptobook.nakov.com/digital-signatures/rsa-sign-verify-examples, diakses pada 21 Desember 2020.
7 “Elliptic Curve Cryptography”, https://www.pycryptodome.org/en/latest/src/public_key/ecc.html, diakses pada 21 Desember 2020.
8 “Digital Signature Algorithm (DSA and ECDSA)”, https://pycryptodome.readthedocs.io/en/latest/src/signature/dsa.html, diakses pada 21 Desember 2020.

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.

Medan, 19 Desember 2020

Yoel Susanto - 13517014