Perbandingan ECDSA dan Algoritma RSA dalam Verifikasi Keaslian Pesan

Faisal Ibrahim Hadiputra / 13509048
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
faisal.hadiputra@students.itb.ac.id

***Abstrak*—Makalah ini membahas tentang implementasi algoritma ECDSA dan RSA pada pengiriman pesan dengan *digital signature*. *Digital signature* adalah salah satu layanan keamanan pada kriptografi yang memberikan jaminan kepada pihak penerima pesan (*receiver*). Jaminan yang diberikan yaitu bahwa pihak pengirim pesan adalah *sender* bukan pihak ketiga (*eyesdropper*) dan pesan yang diterima masih asli. Metode yang dibahas dalam makalah ini adalah *Elliptic Curve Digital Signature Algortihm* (ECDSA) dan Algoritma RSA dan keduanya akan dibandingkan tingkat keamanannya.**

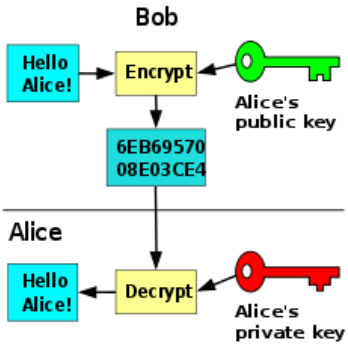
***Kata Kunci*—*Digital signature*, ECDSA, Algoritma RSA, *sender*, *receiver*, enkripsi, dekripsi, autentikasi, *public key cryptography*.**

I. PENDAHULUAN

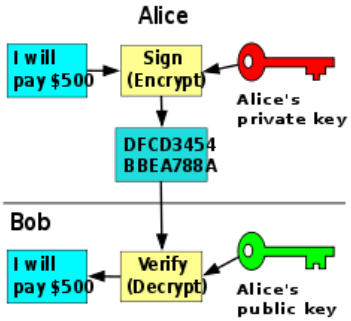
Pada zaman sekarang ini, perkembangan teknologi informasi sudah terjadi secara pesat. Seiring perkembangan teknologi informasi ini, sempat terjadi perbincangan mengenai keamanan dan kebenaran informasi. Saat ini, sudah tidak asing lagi bagi kita untuk mengirim informasi atau pesan dalam bentuk elektronik. Banyak orang memilih media ini karena lebih mudah, cepat, dan sederhana. Adanya internet pun telah menjadi pemicu berkembangnya aliran informasi dalam bentuk elektronik. Banyak kalangan bisnis dan ekonomi memanfaatkan media ini. Hal ini karena mereka dituntut untuk bergerak lebih cepat dalam mengembangkan usahanya. Namun, terjadinya perkembangan elektronik ini mengakibatkan terjadinya rentannya keamanan informasi pada proses komunikasi. Banyak sekali terjadi manipulasi informasi yang terjadi di dunia maya.

Proses komunikasi pada dasarnya melibatkan hanya dua pihak yaitu pengirim (*sender*) dan penerima (*receiver*). Dalam hal ini, terkadang informasi yang dikirim memiliki kadar rahasia yang hanya boleh diketahui oleh kedua belah pihak. Yang jadi masalah adalah apabila terdapat pihak ketiga yang memanipulasi informasi dan menyamar menjadi pengirim. Hal ini menjadi sangat merugikan bagi kedua belah pihak yaitu pengirim dan penerima informasi. Masalah ini dikenal dengan *Man In The Middle Attack*. Hal ini merupakan masalah besar dalam keamanan dunia informasi. Oleh karena itu, diperlukan metode untuk mengetahui keaslian sebuah informasi.

Cara untuk menjaga keamanan data yang dikirim agar tetap utuh dan asli disebut autentikasi. Autentikasi dilakukan untuk membuktikan asli atau tidaknya sebuah informasi atau pesan yang dipakai oleh user (orang yang berhak atas data tersebut). Pembuktian sebuah dokumen atau data ini asli atau tidak juga merupakan dasar untuk pelayanan keamanan pada kepentingan tertentu. Salah satu metode yang ditawarkan dalam konsep authenticity adalah *digital signature* atau tanda tangan digital.



***Gambar 1* Skema enkripsi algoritma enkripsi kunci publik**



***Gambar 2* Skema enkripsi tanda tangan digital**

Digital signature didasarkan pada algoritma kriptografi kunci publik (*public key cryptography*) yaitu terdapat kunci enkripsi dan kunci dekripsi yang berbeda. Pada aplikasinya, penggunaan *digital signature* ini menggunakan metode yang berlawanan dengan algoritma enkripsi kunci publik. Dengan demikian, pengguna melakukan *signing* (enkripsi) dengan kunci pribadi dan

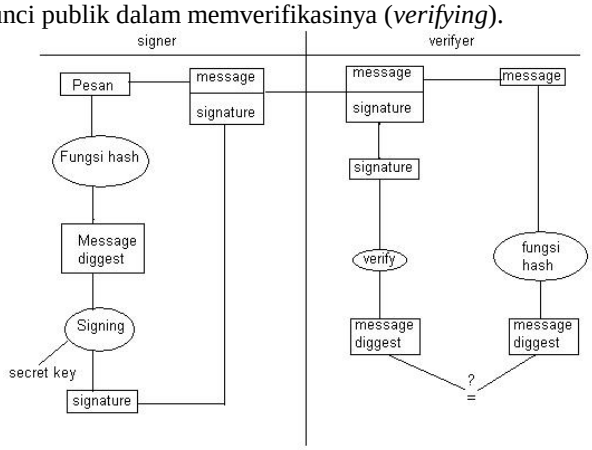
pengguna melakukan *verifying* (dekripsi) dengan kunci publik. Hal ini berbeda dengan algoritma enkripsi kunci publik yang melakukan enkripsi dengan kunci publik dan dekripsi dengan kunci pribadi.

II. DASAR TEORI

A. *Digital Signature*

Tanda tangan digital atau *digital signature* dari kata "tanda tangan" tentu kita memperoleh arti yang merupakan penanda pesan dari pengirim pesan. Berbeda dengan *handwritten signature* atau tanda tangan tulisan tangan, *digital signature* digunakan untuk menandatangani surat atau pesan yang dikirimkan secara elektronik. Namun demikian, sebenarnya fungsi dari keduanya tidak terlalu berbeda jauh.

Pada perkembangannya, banyak sekali algoritma *digital signature*dan semuanya berjenis algoritma kriptografi kunci publik yaitu menggunakan kunci pribadi (*private key*) dalam penandatanganan (*signing*) dan menggunakan kunci publik dalam memverifikasinya (*verifying*).



Gambar 3 **Skema tanda tangan digital**

Berdasarkan skema tersebut, secara umum, penggunaan tanda tangan digital dilakukan dalam 3 proses utama: pembangkitan kunci (*key generating*), pemberian tanda tangan digital (*signing*), dan verifikasi keabsahan tanda tangan digital tersebut (*verifying*).

Pada pembahasan kali ini, proses tersebut akan dilakukan oleh dua algoritma kunci pubik, yaitu algoritma RSA dan ECDSA.

B. Algoritma RSA

Algoritma RSA dijabarkan pada tahun 1977 oleh tiga orang: Ron Rivest, Adi Shamir, dan Len Addleman dari Massachusetts Institute of Technology. Nama Algoritma RSA berasal dari inisial nama belakang mereka, yaitu (**R**ivest—**S**hamir—**A**ddleman).

Pada implementasinya, Algoritma RSA dilakukan sesuai dengan proses penandatanganan digital, yaitu 3 proses utama yang tediri dari:

- *Key Generating*
  1. Pilih dua buah bilangan prima p dan q dengan p ≠ q secara acak dan terpisah untuk p dan q. Hitung N = p.q. N hasil perkalian p dan q.
  2. Hitung φ = (*p*-1)(*q*-1).
  3. Pilih bilangan bulat (integer) antara 1 dan φ (1<e< φ) yang juga relatif prima dengan φ.
  4. Hitung d dengan $d.e \equiv 1 \pmod{\varphi}$.

Dari proses tersebut didapat kedua kunci:

- Kunci Publik:
  - N, modulus yang digunakan.
  - e, eksponen publik.
- Kunci Pribadi:
  - N, modulus pada kunci publik.
  - d, eksponen pribadi.

- *Signing(Encryption)*

Untuk melakukan penandatanganan digital pada pesan m, dengan N dan d sebagai kunci pribadi yang telah didapat. Hal ini dilakukan dengan menghitung:

$$\sigma \equiv m^d \pmod{N} \quad (1)$$

Dengan σ adalah *digital signature* dari pengirim pesan (*sender*).

- *Verifying(Decryption)*

Selanjutnya, penerima melakukan verifikasi pesan m dengan parameter kunci publik yang diberikan, yaitu N dan e dengan menngecek kebenaran kongruensi berikut:

$$\sigma^e \equiv m \pmod{N} \quad (2)$$

Apabila kekongruenan tersebut benar, maka pesan yang diterima oleh penerima (*receiver*) adalah valid.

Pada praktiknya, metode ini terkadang kurang aman, untuk menambah keamanannya, pengirim dan penerima pesan dapat melakukan perjanjiang menggunakan fungsi Hash dalam mengenkripsi dan mendekripsi tanda tangan digitalnya. Penjelasan mengenai fungsi hash akan dijabarkan pada subbab D.

C. *Elliptic Curve Digital Signature Algorithm* (ECDSA)

ECDSA adalah salah satu tipe digital signature yang menggunakan metode *Elliptic Curve Cryptography*[1] (ECC). ECDSA diperkenalkan pertama kali pada tahun 1992 oleh Scott Vanstone dan pada tahun 1998 mendapatkan standar ISO (*International Standard Organization*) yaitu ISO 14888-3. Tahun 1999 diterima sebagai standar ANSI (*American National Standard Institue*) ANSI X9.62 dan tahun 2000 sebagai standar IEEE (*Institute of Electrical and Electronic Engineers*) IEEE 1363-2000 serta standar NIST (*National Institute of Standards and Technology*) yaitu FIPS 186-2.

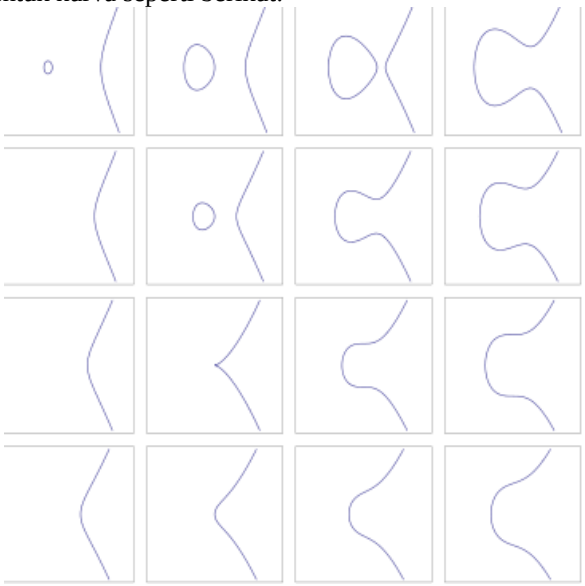
ECDSA menggunakan kurva eliptik sebagai dasar dalam perhitungannya. Kurva eliptik merupakan kurva mulus dan termasuk salah satu *projective algebraic curve*[2]dengan persamaan berikut:

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

[1] http://en.wikipedia.org/wiki/Elliptic_Curve_Cryptography

[2] *Projective algebraic curve* adalah kurva simetri yang ekspansinya dapat dinyatakan dalam persamaan aljabar.

Apabila digambarkan secara grafis, diperoleh beberapa bentuk kurva seperti berikut:



***Gambar 4* Macam-macam kurva eliptik**

Pada penggunaan ECDSA, pihak yang melakukan *digital signature* mempunyai parameter domain kurva eliptik berupa D = {q,FR,a,b,G,n,h} yang telah disepakati oleh kedua pihak, dimana q adalah ukuran batas daerah (*size of finite field*), FR adalah basis yang digunakan, a dan b adalah dua parameter yang mendefinisikan kurva, G adalah titik basis (*base point*) dari orde tertentu kurva G = (x_G,y_G), n adalah orde dari titik G, h adalah kofaktor yang sama dengan orde kurva dibagi n. Selain itu, pihak pengirim juga memiliki pasangan kunci pribadi d_A dan kunci publik Q_A. Kemudian, pihak yang akan melakukan verifikasi *digital signature* memiliki salinan dokumen D yang otentik dan kunci publik Q_A. Proses yang dilakukan pada ECDSA adalah sebagai berikut:

- *Key Generating*
  1. Pilih sebuah bilangan bulat random d_A, yang nilainya antara 1 dan n-1 ($1<d_A<n-1$).
  2. Hitung $Q_A = d_A . G$.
  3. Kunci pribadi adalah d_A.
  4. Kunci pribadi adalah Q_A.
- *Signing (Encryption)*
  1. Hitung e = HASH (m), dimana HASH adalah fungsi kriptografi hash.
  2. Pilih sebuah bilangan bulat random k yang nilainya antara 1 dan n-1 ($1<k<n-1$).
  3. Hitung $k . G = (x_1,y_1)$ dan r = x1 mod n, apabila r = 0, kembali ke langkah 2.
  4. Hitung $s = k^{-1}(e+d_A.r) \bmod n$, apabila s =0, kembali ke langkah 2.
  5. *Digital signiture*-nya adalah (r,s).
- *Verifying (Decryption)*
  1. Verifikasi r dan s berada pada rentang [1,n-1]. Apabila tidak benar, maka tidak valid.
  2. Hitung e = HASH(m), dimana fungsi HASH yang digunakan adalah sesuai yang digunakan pada enkripsi.
  3. Hitung $w = s^{-1} \bmod n$.
  4. Hitung $u_1 = ew \bmod n$ dan $u_2 = rw \bmod n$.
  5. Hitung $(x_1,y_1) = u_1G + u_2Q_A$.
  6. Apabila $r = x_1 \bmod n$, maka *digital signature* valid, dan sebaliknya.

Terkadang pada proses verifikasi, kunci publik Q_A tidak dipercayai. Oleh karena itu, terdapat cara pengecekan keabsahan kunci publik tersebut dengan membutuhkan O sebagai kunci validasi, yaitu:

1. Cek bahwa Q_A tidak sama dengan O apabila tidak sama, maka kunci publik valid.
2. Cek bahwa Q_A terletak pada kurva $y^2 \bmod q = x^3 + ax + b \bmod q$.
3. Cek bahwa $nQ_A = O$

D. Fungsi Hash

Fungsi *Hash* (dilambangkan dengan *h(k)*) bertugas untuk mengubah k (*key*) menjadi suatu nilai dalam interval [0....X], dimana "X" adalah jumlah maksimum dari *record-record* yang dapat ditampung dalam tabel. Jumlah maksimum ini bergantung pada ruang memori yang tersedia. Fungsi *Hash* yang ideal adalah mudah dihitung dan bersifat *random*, agar dapat menyebarkan semua *key*. Dengan *key* yang tersebar, berarti data dapat terdistribusi secara seragam bentrokan dapat dicegah. Sehingga kompleksitas waktu model *Hash* dapat mencapai O(1), di mana kompleksitas tersebut tidak ditemukan pada struktur model lain.

Pada praktiknya, fungsi hash terdapat beberapa macam yang relatif sederhana, yaitu:

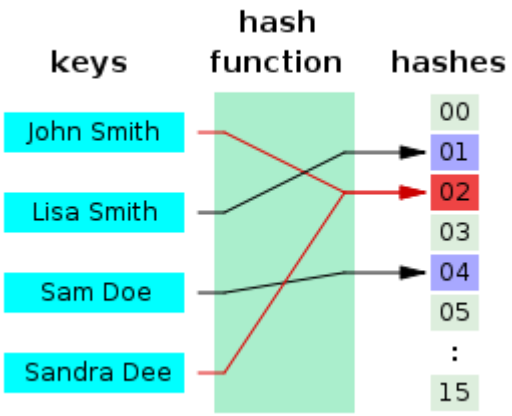
1. Metode Pembagian Biasa (*division-remainder method*)

   Secara umum, rumus dari metode ini adalah h(k) = k mod m. Dalam hal ini m adalah jumlah lokasi yang tersedia. Fungsi hash ini menempatkan *record* dengan kunci k pada suatu lokasi memori yang beralamat h(k).
2. Melipat (*folding*)

   Metode ini membagi nilai asli ke dalam beberapa bagian, kemudian menambahkan nilai-nilai tersebut, dan mengambil beberapa angka terakhir sebagai nilai hashnya.
3. Transformasi Radiks (*radix transformation*)

   Karena nilai dalam bentuk digital, basis angka atau radiks dapat diganti sehingga menghasilkan urutan angka-angka yang berbeda. Contohnya nilai desimal (basis 10) bisa ditransformasikan ke dalam heksadesimal (basis 16). Digit atas hasilnya bisa dibuang agar panjang nilai hash dapat seragam.
4. Pengaturan ulang digit (*digit rearrangement*)

   Metode ini mengubah urutan digit dengan pola tertentu. Contohnya mengambil digit ketiga sampai ke enam dari nilai aslinya, kemudian membalikan urutannya dan menggunakan digit yang terbalik sebagai nilai hash.



Gambar 5 **Fungsi hash yang merelasikan nama dengan bilangan bulat [0..15]**

Pada *digital signature*, fungsi hash digunakan untuk menambah nilai keamanan dari pesan yang dikirimkan oleh penandatangan digital. Kedua pihak, baik pengirim dan penerima harus mengetahui dan menggunakan metode fungsi hash yang sama dalam penggunaan *digital signature* ini. Dengan demikian tingkat keamanan dari pesan dapat dijaga dengan cukup baik.

III. PEMBAHASAN

Pada Bab ini akan dilakukan contoh sederhana penggunaan algoritma RSA dan ECDSA dalam verifikasi keaslian pesan. Setelah itu, akan dilakukan pembandingan antara keamanan dan efisiensi dari kedua metode ini.

Contoh permasalahan yang akan diambil adalah sebagai berikut:

Misalkan Ani akan mengirimkan pesan m kepada Budi. Dengan m = 3458999. Selanjutnya, ia akan menggunakan *digital signature* pada pengirimannya kepada Budi. Selain itu, iya juga menggunakan fungsi hash secara metode pembagian biasa dengan basis pembaginya adalah 2980, yaitu h(k) = k mod 2980.

A. Penyelesaian dengan Algoritma RSA

Langkah pertama yang harus dilakukan adalah melakukan pembangkitan kunci (*key generating*):

1. Misal p = 509 dan q = 487 dengan demikian N = (509).(487) = 247883.
2. Lalu φ = (509-1)(487-1) = 246888.
3. Ambil e = 157 relatif prima terhadap 246888.
4. Hitung d.157 ≡ 1 (mod 246888). Diperoleh d = 179269.

Diperoleh kunci pubik

- N = 247883 dan
- e = 157.

Kunci pribadi

- N = 247883 dan
- d = 179269.

Pada Algoritma RSA ini ternyata diperoleh kunci pribadi dengan d sepanjang 6 digit. Langkah selanjutnya adalah melakukan *signing (encryption)*, yaitu:

1. Pertama kali temukan m' sebagai fungsi hash dari m:

$$m' = h(m) = 3458999 \bmod 2980$$

$$m' = 2199$$

2. Selanjutnya, kita akan mencari *digital signature*-nya, yaitu :

$$\sigma = 2199^{179269} \bmod 247883$$

$$\sigma = 175477$$

Dengan demikian signature yang diperoleh adalah 175477. Yang perlu dikirimkan oleh Ani adalah m = 345899, N = 247883 dan e = 157, serta signature σ = 175477 .

Selanjutnya setelah pesan terkirim, Budi harus mengecek keabsahan (*verifying*) dari pesan yang dikirimkan oleh Ani kepadanya. Ia akan mengecek kongruensi berikut:

$$1745477^{157} \equiv 2199 (\bmod 247883)$$

Hasil dari pengecekan dari kongruensi ini adalah benar, sehingga Budi mendapatkan pesan yang benar.

B. Penyelesaian dengan ECDSA

Pada penggunaan *digital signature* dengan ECDSA, mungkin tidak semudah melakukannya dengan Algoritma RSA, langkah-langkah pada ECDSA agak lebih rumit. Langkah pertama yang harus dilakukan Ani adalah menentukan parameter domain yang dibutuhkan pada kurva eliptik. Penentuan parameter yang dibutuhkan ini memiliki tingkat kesulitan yang cukup tinggi.

Misalkan parameter yang diambil adalah sebagai berikut:

- Batasan daerah : q adalah bilangan prima 23
- *Field Representation* (FR) :

$$y^2 \bmod 23 = (x^3 + ax + b) \bmod 23$$

$$4a^3 + 27b^2 \bmod 23 \neq 0 \qquad (3)$$

- Parameter kurva : a = 1, b = 4
- Ambil titik generator G : (0,2)
- Orde n adalah bilangan prima harus lebih besar dari 4*sqrt(23) ≈ 19.18, maka ambil n = 23.
- Kofaktor h :

$$h = \#E(F_{23}) / n$$

$$h = 29/23 \approx 1.261 \approx 1$$

Sehingga didapatkan parameter yang harus disetujui oleh Ani dan Budi D={q,FR,a,b,G,n,h}= {23,F_{23},1,4,(0,2),23,1}. Langkah selanjutnya adalah proses pembangkitan kunci (*key generating*):

1. Misalkan ambil d_A= 18.
2. Dengan demikian Q_A= 18G = (10,18) sesuai definisi *Point Multiplication* dari kurva eliptik.
3. Diperoleh kunci pribadi : d_A = 18.
4. Kunci publik : Q_A= (10,18).

Setelah didapatkan kedua kunci tersebut, langkah selanjutnya adalah melakukan *signing (encryption)*, yaitu:

1. Kembali kita menggunakan Hash dan diperoleh e=m'= 2199.
2. Misalkan kita pilih k = 4.
3. $(x_1,y_1) = 4.(0,2) = (1,12)$ dan r = 1 mod 23 = 1.
4. Perhitungan: $s = 4^{-1}(2199+18.14) \pmod{23} = 9$.
5. Diperoleh digital signature adalah (1,9).

Dengan demikian digital signature yang diperoleh adalah (14,16). Yang diperlukan untuk Ani kirimkan adalah m = 345899, D ={23,F_{23},1,4,(0,2),23,1}, dan Q_A=(10,18), serta signature (r,s)=(1,9).

Selanjutnya, setelah pesan terkirim, Kembali lagi Budi harus mengecek keabsahan (*verifying*) dari pesan yang dikirimkan oleh Ani kepadanya. Langkah-langkah verifikasi adalah sebagai berikut:

1. Cek r dan s berada pada [1, n-1], ternyata benar 14 dan 16 berada pada [1,22].
2. Hitung e = HASH(345899) = 2199.
3. Hitung $w = 9^{-1} \pmod{23} = 18$.
4. Hitung u_1 = 2199 . 18 (mod 23) = 22 dan u_2 =1 . 18 (mod 23) = 18.
5. $(x_1,y_1) = 22G + 18Q_A = (15,17) + (11,9) = (1,11)$
6. Diperoleh $r \equiv x_1 \pmod{23}$ terpenuhi.

Hasil dari penggunaan digital signature dengan menggunakan ECDSA telah dilakukan dengan benar.

C. Perbandingan Algoritma RSA dan ECDSA

Dari kedua penyelesaian dengan metode Algoritma RSA dan ECDSA terdapat perbedaan bahwa Algoritma RSA membutuhkan digit yang lebih banyak pada *digital signature*-nya dibandingkan dengan ECDSA dengan tingkat keamanan yang relatif sama. Selain itu, kompleksitas algoritma dari ECDSA relatif lebih sulit dibandingkan Algoritma RSA, hal ini mengakibatkan tingkat keamanan ECDSA lebih tinggi dibandingkan Algoritma RSA pada taraf panjang digit yang relatif sama.

Berdasarkan penelitian internasional didapatkan perbandingan antara kedua algoritma tersebut yang tertera pada tabel berikut:

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

***Gambar 6* Perbandingan jumlah bit Algoritma RSA dan ECDSA dengan tingkat keamanan sama.**

Pada Gambar 6 tersebut, terdapat perbandingan antara Algoritma RSA (kolom 2) dan ECDSA (kolom 3). Untuk tingkat keamanan yang sama, RSA membutuhkan 1024 bit, sedangkan ECDSA membutuhkan 160 bit. Hal ini menunjukkan bahwa keamanan ECDSA lebih baik daripada Algoritma RSA, namun tingkat kesulitan dalam melakukan perhitungan ECDSA jauh lebih tinggi daripada Algoritma RSA.

IV. KESIMPULAN

Berdasarkan uraian dan analisis, serta perbandingan diatas, dapat diambil beberapa kesimpulan:

1. Dalam dunia informasi, keamanan dari pesan dan informasi merupakan hal yang sangat penting untuk menghindari hal-hal yang tidak diinginkan.
2. Untuk memverifikasi keaslian pesan, serta menjaga keamanannya dapat menggunakan beberapa macam metode, seperti Algoritma RSA dan ECDSA.
3. Algoritma RSA memiliki kompleksitas yang lebih mudah dibandingkan ECDSA
4. ECDSA memiliki tingkat kesulitan yang lebih tinggi dibandingkan dengan RSA, namun lebih aman untuk digunakan.
5. Kekuatan dan keamanan dari sebuah sistem verifikasi keaslian pesan ditentukan oleh jumlah bit dari *digital signature*. Semakin panjang bit-nya, semakin tinggi tingkat keamanannya.

DAFTAR PUSTAKA

[1] AP, Pualam Sendi dkk. "Implementasi Algoritma ECDSA Untuk Pengamanan E-Mail".www.eepis-its.edu. September 2007.

[2] http://en.wikipedia.org/wiki/Elliptic_curve diakses pada tanggal: 15 Desember 2010.

[3] http://en.wikipedia.org/wiki/Elliptic_curve_DSA diakses pada tanggal: 14 Desember 2010.

[4] http://en.wikipedia.org/wiki/Hash_function diakses pada tanggal: 14 Desember 2010.

[5] http://en.wikipedia.org/wiki/Public-key-cryptography diakses pada tanggal: 14 Desember 2010.

[6] http://id.wikipedia.org/wiki/Kriptografis diakses pada tanggal: 14 Desember 2010.

[7] http://id.wikipedia.org/wiki/RSA diakses pada tanggal: 15 Desember 2010.

[8] Johnson, Don dkk. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". Certicom. Desember 2001.

[9] MS, Anoop.*Elliptic Curve Cryptography*. An Implementation Guide, September 2007.

[10] Munir, Rinaldi.2008.*Diktat Kuliah Struktur Diskrit*.Bandung: Penerbit Informatika ITB.

[11] Prabanco, Puthut. "Fungsi Hash Kriptografis".informatika.org, Desember 2007.

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, 15 Desember 2010

Faisal Ibrahim Hadiputra / 13509048