Perbandingan Digital Signature Algorithm dan Elliptic Curve Digital Signature Algorithm

Rama Febriyan 13511067[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511067@std.stei.itb.ac.id

***Abstract*—Pada makalah ini, dilakukan perbandingan antara dua algoritma pembangkit tanda tangan digital, yaitu Digital Signature Algorithm dan Elliptic Curve Digital Signature Algorithm. Perbandingan dilihat dari sisi kecepatan tiap algoritma dalam membangkitkan tanda tangan serta validasi tanda tangan.**

***Index Terms*—DSA, ECDSA, hash, kriptografi kunci publik**

I. PENDAHULUAN

Kriptografi merupakan sebuah kegiatan atau dapat disebut sebagai karya seni untuk menyembunyikan informasi dalam bentuk yang berbeda. Informasi tersebut dapat berupa pesan teks, gambar, suara, atau video. Kriptografi digunakan untuk menjaga kerahasiaan pesan, mnjaga orisinalitas serta keaslian pesan.

Pada dunia nyata keaslian satu pesan ditandai dengan dibubuhkannya tanda tangan dari pemilik atau orang yang bertanggung jawab atas pesan tersebut. Layaknya dokumen dunia nyata, dokumen digital juga perlu untuk dijaga keasliannya. Akan tetapi, tanda tangan tidak dapat diberikan dengan cara yang sama dengan dokumen nyata. Tanda tangan digital atau yang lebih dikenal dengan *Digital Signature* merupakan solusi untuk mengatasi masalah keaslian dari dokumen digital.

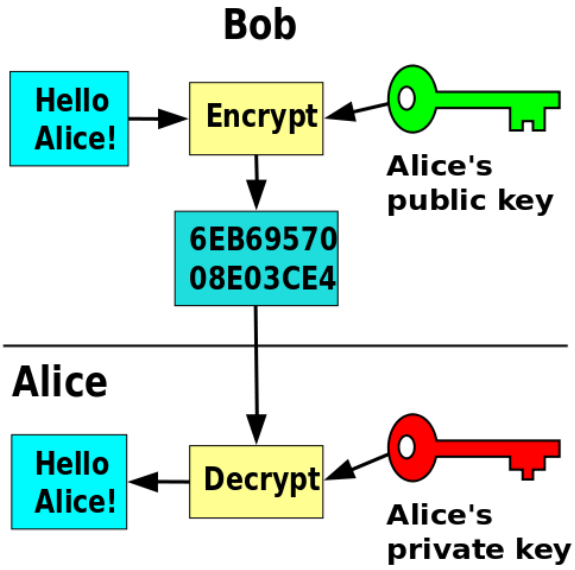
Pembuatan tanda tangan digital dilakukan dengan memanfaatkan algoritma kriptografi kunci publik. Tujuannya adalah agar tanda tangan dapat di periksa keabsahannya oleh semua orang namun hanya dapat dibuat oleh orang tertentu. Terdapat banyak jenis algoritma yang dapat digunakan dalam pembuatan tanda tangan digital. Beberapa diantara algoritma tersebut adalah *Digital Signature Algorithm, Elliptic Curve Algorithm, Edwards-curve Digital Signature Algorithm.* Algoritma-algortma tersebut pada dasarnya menggunakan prinsip yang sama, yaitu menggunakan nilai *hash* dari pesan untuk membuat pasangan tanda tangan. Hanya saja, setiap algoritma memiliki cara tersendiri dalam hal mengkalkulasi nilai dari tanda tangan.

.

II. TEORI DASAR

A. Kriptografi Kunci Publik

Algoritma kriptografi kunci publik merupkan algoritma dasar yang digunakan dalam pembuatan tanda tangan digital. Layaknya algoritma kriptografi biasa, algoritma kriptografi kunci publik dibuat untuk menyembunyikan pesan asli. Ide dari pembuatan algoritma kriptografi kunci publik adalah kunci dibuat sepasang, terdiri dari kunci untuk melakukan enkripsi dan kunci untuk dekripsi.



Gambar 1 Ilustrasi kriptografi kunci publik (id.wikipedia.com)

Kunci publik merupakan kunci yang dipakai untuk melakukan enkripsi pesan, sedangkan kunci privat digunakan untuk dekripsi pesan. Hal ini sama seperti surat yang dimasukkan ke dalam kotak pos pribadi. Surat dapat dimasukkan oleh semua orang yang mengetahui lokasi kotak pos, namun surat yang berada di dalam kotak hanya dapat diambil oleh pemilik yang memiliki kunci kotak. Misal terdapat fungsi enkripsi dan dekripsi E dan D, serta pasangan kunci publik-privat (e,d) maka pesan *p* dienkripsi dengan persamaan berikut:

$$E_e(p) = c$$

Sedangkan untuk proses dekripsi, dilakukan perhitungan berikut:

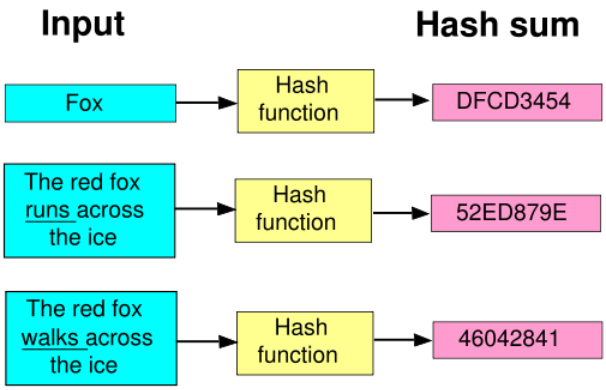
$$D_d(c) = p$$

Kunci publik sebelumnya didistribusikan secara bebas agar setiap orang dapat menggunakannya, namun pesan yang telah dienkripsi hanya dapat dibuka oleh pemilik kunci publik dengan menggunakan kunci privat miliknya.

B. Fungsi Hash

Fungsi *hash* merupakan salah satu bentuk dari teknik kriptografi yang lebih dikenal sebagai teknik kriptografi tanpa kunci. Fungsi *hash* juga dikenal sebagai *one-way function* karena pesan yang sudah dikenai fungsi tidak dapat dikembalikan menjadi pesan semula.

Fungsi *hash* yang cukup banyak dikenal adalah MD5, SHA, dan SHA-1. Sebenarnya masih banyak jenis fungsi *hash*, seperti MD2, MD4, SHA-2, *Snefru, N-hash, RIPE-MD* dan masih banyak lagi. Fungsi *hash* banyak digunakan sebagai identitas pesan. Alasannya adalah perubahan pada pesan akan meberikan perubahan yang cukup besar pada pesan keluaran (*message digest*). Dengan membandingkan *message digest* dari dua buah pesan, dapat diketahui apakah pesan tersebut sama atau tidak. Hal ini sangat berguna untuk melihat adanya perubahan yang sangat kecil pada satu pesan besar.



Gambar 2 Ilustrasi mengenai fungsi hash (blog.ugm.ac.id)

Selain itu, fungsi *hash* juga digunakan untuk menyimpan kata sandi dalam basis data. Ini bertujuan agar kata sandi tersebut tidak dapat dibaca oleh siapapun yang membuka basis data tersebut.

C. Digital Signature Algorithm

Digital Signature Algorithm (DSA) merupakan salah satu algoritma yang digunakan dalam pembuatan tanda digital. Algoritma ini juga dapat digunakan untuk memeriksa keabsahan dari tanda tangan.

Algoritma DSA dikembangkan dari algoritma kunci publik ElGamal. Seperti algoritma kunci publik pada unumnya, DSA juga menggunakan dua buah kunci. Kunci privat digunakan untuk membuat tanda tangan, dan kunci publik digunakan untuk memeriksa keabsahan dari tanda tangan serta isi dokumen. Algoritma DSA diawali dengan pembangkitan kunci sebagai berikut:

1. Tentukan bilangan prima p dan q di mana $(p-1)$ $mod\ q=0$.
2. Hitung $g = h^{(p-1)/q} \bmod p$, yang dalam hal ini $1 < h < p - 1$ dan $h^{(p-1)/q} \bmod p > 1$.
3. Tentukan kunci privat x Diana $x<q$.
4. Hitung kunci publik $y=g^x\ mod\ p$.

Setelah kunci dibangkitkan, tanda tangan dibangitkan dengan langkah berikut:

1. Ubah pesan m menjadi *message digest* dengan fungsi *hash* SHA. SHA digunakan karena DSA dibuat sepasang dengan SHA.
2. Tentukan bilangan acak $k < q$.
3. Hitung r dan s sebagai berikut:

$$r = (g^k \bmod p) \bmod q$$

$$s = (k^{-1}(H(m) + x * r)) \bmod q$$

4. Kirim pesan m beserta tanda-tangan r dan s.

Untuk melakukan verifikasi dari tanda tangan, dulakukan langkah berikut:

1. Hitung

$$w = s^{-1} \bmod q$$

$$u_1 = (H(m) * w) \bmod q$$

$$u_2 = (r * w) \bmod q$$

$$v = ((g^{u1} * y^{u2}) \bmod p) \bmod q)$$

2. Jika $v = r$, maka tanda-tangan sah

D. Elliptic Curve Digital Signature Algorithm

ECDSA sendiri merupakan pengembangan dari algoritma Elliptic Curve Cryptography. Pada algoritma ini, tanda tangan digital dibuat dengan melakukan perkalian terhadap salah satu titik pada elips. Berikut langkah pembuatan tanda tangan digital dengan menggunakan algoritma ECDSA.

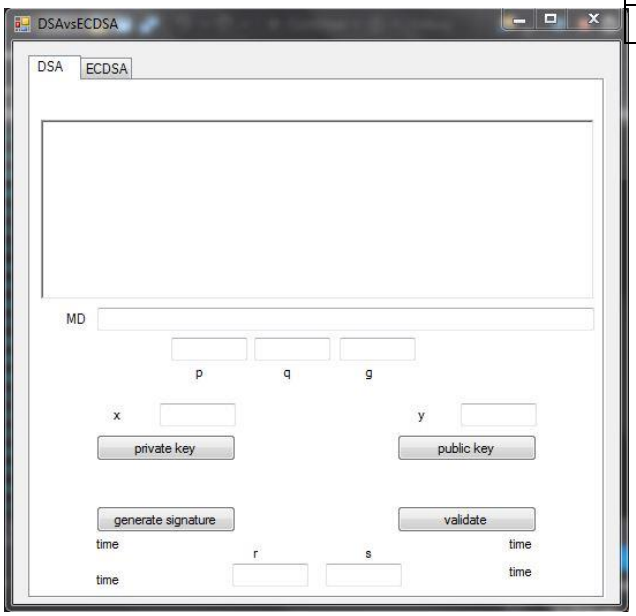
1. Tentukan sebuah kunci privat dA.
2. Hitung nilai e = HASH(m) dengan HASH merupakan fungsi untuk melakukan hash, pada kasus ini menggunakan SHA-1
3. Pilih salah satu bilangan acak k dari [1, n-1] dengan n merupakan orde dari kurva elips.
4. Hitung r = x1 (mod n), dengan (x1,y1) = k*G, dan G merupakan titik basis yang telah disepakati kedua belah pihak. Jika r = 0, ulangi langkah 3.
5. Hitung $s = k^{-1}(e + dA*r)(\text{mod } n)$. Jika s = 0, ulangi langkah 3
6. Pasangan tanda tangan adalah (r,s)

Sedangkan untuk melakukan verifikasi dari tanda tangan, penerima harus memiliki kunci publik Q dari pengirim.

1. Pastikan jika r dan s berada dalam selang 1 sampai n-1. Jika tidak, tanda tangan tidak sah.
2. Hitung e = HASH(m) dengan menggunakan fungsi hash yang sama.
3. Hitung w = s^{-1} (mod n)
4. Hitung u1 = e*w (mod n) dan u2 = r*w (mod n)
5. Hitung (x1, y1) = u1*G + u2*Q
6. Tanda tangan akan sah jika x1 = r (mod n)

III. Analisis dan Pembahasan

Untuk melakukan pengujian, algoritma diimplementasikan dengan menggunakan bahasa C#. Program hasil implementasi dapat melakukan *hash* terhadap teks, membangkitkan tanda tangan serta melakukan pengecekan terhadap keabsahan tanda tangan.



Gambar 3 Tampilan program

Pengujian dilakukan lima kali. Untuk setiap pengujian, nilai MD sama antara algoritma DSA dengan ECDSA.

Tabel 1 Nilai MD pada percobaan

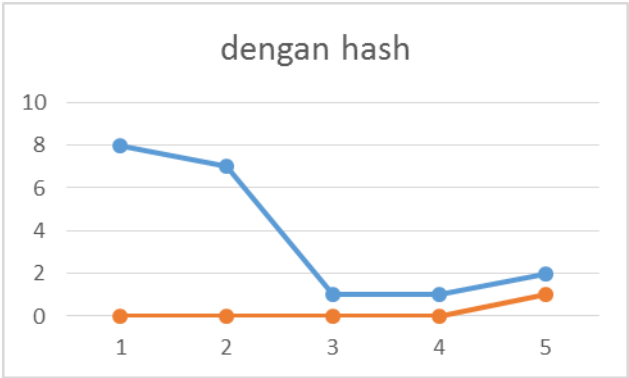
no	MD
1	A94A8FE5CCB19BA61C4C0873D391E987982FBBD3
2	4EB76CE9DA8CC53BEDC3567225CC2AD4B869F58F
3	AB7B4FEA12B998F03FA844E031B5A9051137331F
4	9605CA44984DC1DB70EBBF7E4A12CE54C124B04C
5	8B52CBB253E7822775756BC571053E21D7A43EE5

Tabel 2 Waktu pembangkitan tanda tangan dalam milisekon

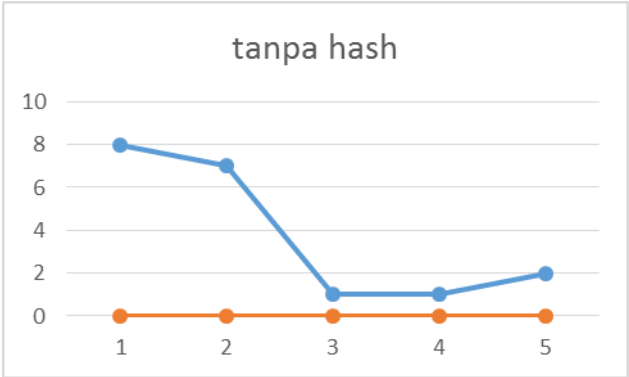
	DSA		ECDSA	
	Hash	non-hash	hash	nonhash
1	8	8	0	0
2	7	7	0	0
3	1	1	0	0
4	1	1	0	0
5	2	2	1	0

Tabel 3 Waktu validasi tanda tangan dalam milisekon

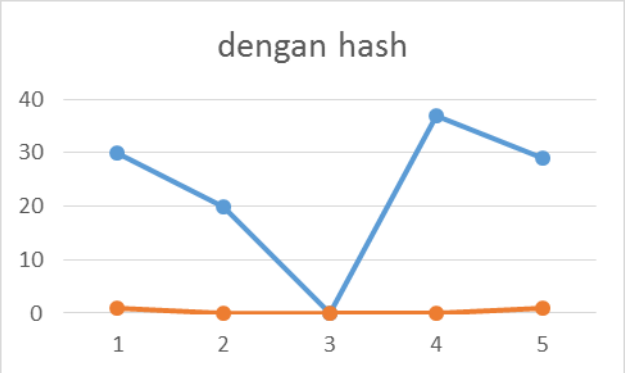
	DSA		ECDSA	
	hash	non-hash	hash	nonhash
1	30	30	1	1
2	20	20	0	0
3	0	0	0	0
4	37	37	0	0
5	29	29	1	1



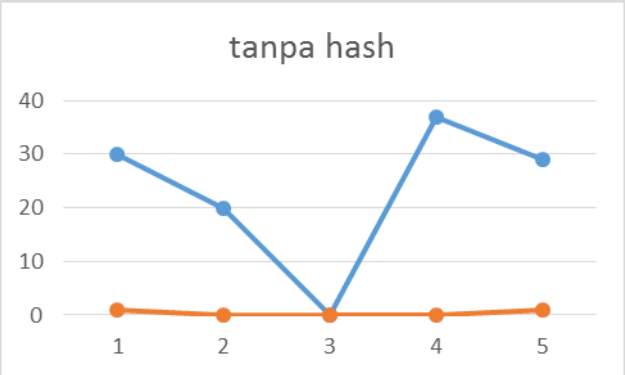
Gambar 4 Perbandingan pembangkitan tanda tangan beserta hash



Gambar 5 Perbandingan pembangkitan tanda tangan tanpa hash



Gambar 6 Perbandingan waktu validasi tanda tangan beserta hash



Gambar 7 Perbandingan waktu validasi tanda tangan tanpa hash

Dari data pada tabel 1 diketahui bahwa algoritma ECDSA bekerja lebih cepat jika dibandingkan dengan DSA. Untuk proses validasi tanda tangan, ECDSA jauh lebih cepat jika dibandingkan dengan DSA yang mencapai puluhan milisekon.

Jika dilihat dari proses kalkulasi yang dilakukan, algoritma DSA memiliki kalkulasi yang lebih kompleks jika dibandingkan dengan ECDSA. Kalkulasi yang kompleks ini memakan waktu yang cukup besar, sehingga proses pembangkitan serta validasi dari tanda tangan lebih lama. Selain itu, algoritma DSA mengharuskan setiap parameter memiliki nilai yang sangat besar. Jika kalkulasi dilakukan dengan nilai yang besar, Wat yang dihabiskan akan semakin besar. Hal ini menjadi kelebihan dari algoritma ECDSA karena tidak ada ketentuan untuk menggunakan bilangan yang besar sebagai parameter.

IV. KESIMPULAN

Algoritma DSA jika dibandingkan dengan algoritma ECDSA berjalan jauh lebih lambat. Namun hal ini terbayarkan dengan adanya ketentuan agar parameter DSA merupakan bilangan yang sangat besar. Dengan adanya parameter yang besar, proses analisis untuk mengetahui kunci privat dari algoritma sangat sulit sehingga keamanan lebih terjamin.

Algoritma ECDSA memiliki kelebihan dari sisi kecepatan. Selain itu ECDSA memiliki parameter yang cenderung lebih mudah diingat karena berupa persamaan dan titik koordinat karteisan. Bandingkan dengan DSA yang memiliki parameter hanya berupa variabel dan nilai yang sulit diingat.

REFERENSI

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/ECC_Tut_v1_0.pdf
[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2013-2014/Makalah2-2014/MakalahKripto2-2014-004.pdf
[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Fungsi%20Hash%20(2015).ppt
[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/SHA%20(2015).ppt

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, 11 Mei 2015

Rama Febriyan/13511067