Kriptografi Elliptic Curve Dalam Digital Signature

Ikmal Syifai 13508003[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*ikmal.s@students.itb.ac.id*

***Abstract*—*Kriptografi digunakan untuk menyampaikan pesan rahasia tanpa diketahui pihak yang tidak berhak. Salah satu penggunaannya adalah dalam* digital signature. *Dalam makalah ini akan dijelaskan tentang* digital signature *yang menggunakan* elliptic curve cryptography *yang dianggap sebagai generasi lanjutan dari public key* cryptography.

***Kata Kunci*—** *digital signature*, *elliptic curve,* kriptografi.

I. PENDAHULUAN

Digital signature atau tanda tangan digital adalah suatu transformasi pesan yang menggunakan sebuah sistem kripto asimetrik sehingga orang yang memunyai pesan awal dan *public key* mampu menentukan secara akurat: (1) apakah transformasi tersebut diciptakan menggunakan *private key* yang cocok dengan *public key* milik penandatangan; dan (2) apakah pesan awal berubah sejak transformasi dilakukan (Minnesota Statutes, section 325K). Secara sederhana, *digital signature* didefinisikan sebagai sebuah skema matematika untuk membuktikan keaslian sebuah pesan atau dokumen digital.

Digital signature biasa digunakan pada pendistribusian *software*, transaksi finansial, dan dalam hal-hal yang membutuhkan pendeteksian untuk pemalsuan dan perusakan. Ada beberapa alasan mengapa untuk menandatangani hash (disebut juga *message digest*) daripada seluruh dokumen:

1. Kompatibilitas: dokumen biasa dalam bentuk teks, sebuah fungsi hash dapat digunakan pada input apapun ke bentuk yang cocok.
2. Integritas: tanpa fungsi hash, teks yang akan di-*sign* mungkin harus dipotong dalam blok-blok kecil yang cukup agar algoritma *digital signature* dapat langsung diaplikasikan.
3. Efisiensi: *signature* akan lebih pendek dan menghemat waktu karena proses *hashing* biasanya lebih cepat daripada proses *signing*.

Biasanya skema *digital signature* terdiri dari tiga algoritma:

1. Algoritma untuk membangkitkan *private key* dan *publik key*-nya.
2. Algoritma untuk memberi *digital signature* pada dokumen jika disediakan dokumen dan *private key*.
3. Algoritma untuk verifikasi tanda tangan digital jika disediakan dokumen, *public key*, dan *digital signature*.

Agar dapat bekerja dengan baik maka ada dua kondisi yang harus dipenuhi, yaitu:

1. *Digital signature* yang dibangkitkan dari dokumen dan *private key* harus bisa memverifikasi dokumen yang disertai *public key*.
2. Tidak boleh ada kemungkinan untuk membangkitkan *digital signature* yang valid dari sebuah dokumen tanpa ada *private key* yang seharusnya.

Merujuk pada fungsi hash kriptografi, *digital signature* juga harus memenuhi sifat-sifat berikut:

1. Tahan terhadap *preimage attack*: jika ada suatu hash h, maka sulit dicari m dimana $h = hash(m)$. *Preimage* adalah suatu himpunan yang berisi tepat semua elemen domain dari suatu fungsi.
2. Tahan terhadap *second preimage attack*: jika ada input m_1, maka sulit dicari m_2 dimana $m_1 \neq m_2$ sehingga $hash(m_1) = hash(m_2)$.
3. Tahan terhadap *collision*: penggabungan dua hal di atas sehingga sulit mencari m_1 dan m_2 dimana $hash(m_1) = hash(m_2)$.

II. ALGORITMA DIGITAL SIGNATURE

Digital signature algorithm adalah standar untuk *digital signature* dari *Federal Information Processing Standard*. Diajukan oleh NIST (*National Institute of Standards and Technology*) pada tahun 1991.

Terdapat tiga tahap dalam aplikasi *digital signature* yaitu:

1. Pembangkitan *private key* dan *public key*.
2. Pemberian *digital signature.*
3. Verifikasi *digital signature*.

1. Pembangkitan *private key* dan *public key*.

Terdiri dari dua tahap: pemilihan parameter algoritma dan penentuan *private key* dan *publik key*.

Tahap pemilihan parameter algoritma:

a. Memilih fungsi hash kriptografi, biasanya digunakan *SHA-1* atau *SHA-2*.
b. Menentukan panjang kunci L dan N, L adalah kelipatan 64 antara 1024 (pada DSS sebelumnya), biasanya menggunakan

pasangan (1024,160), (2048,224), (2048,256), dan (3072,256).

c. Pilih sebuah bilangan prima *N*-bit q. *N* harus lebih kecil dari atau sama dengan panjang keluaran hash.
d. Pilih sebuah bilangan prima *L*-bit modulus *p* sehingga (*p*-1) adalah sebuah perkalian dari *q*.
e. Pilih *g*, sebuah bilangan yang memunyai perkalian modulo *p* adalah *q*. Hal ini bisa dilakukan dengan mengatur $g = h^{(p-1)/q}$.

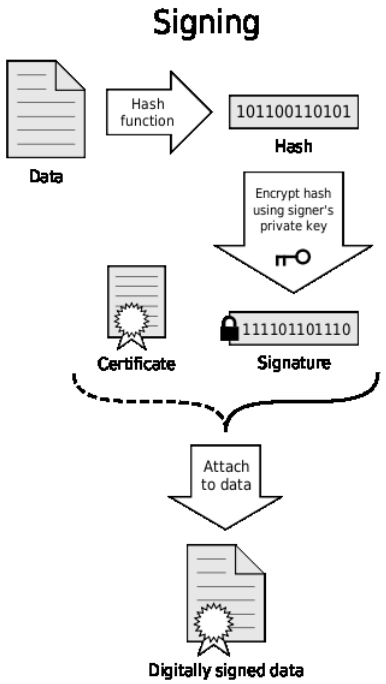
Tahap penentuan *private key* dan *publik key*:

a. Pilih *x* dengan metode acak dimana $0 < x < q$.
b. Hitung $y = g^x \bmod p$.
c. *Public key* adalah (*p, q, g, y*). *Private key* adalah *x*.

2. Pemberian *digital signature.*

Jika H adalah fungsi hash dan m adalah dokumen.:

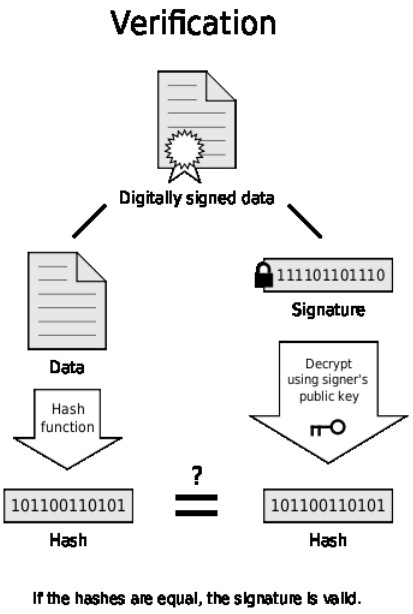
a. Bangkitkan nilai *k*, yaitu nilai acak tiap dokumen dimana $0 < k < q$.
b. Hitung $r = (g^k \bmod p) \bmod q$.
c. Hitung $s = (k^{-1}(H(m) + x*r)) \bmod q$.
d. Hitung ulang *signature* di kasus yang jarang terjadi yaitu $r = 0$ atau $s = 0$.
e. *Digital signature*-nya adalah (*r, s*).



Gambar 1 – Pemberian *Digital Signature*

3. Verifikasi *digital signature*.

a. Tolak *signature* jika $0 < r < q$ atau $0 < s < q$ tidak terpenuhi.
b. Hitung $w = (s)^{-1} \bmod q$.
c. Hitung $u1 = (H(m)*w) \bmod q$.
d. Hitung $u2 = (r*w) \bmod q$.
e. Hitung $v = ((g^{u1} * y^{u2}) \bmod p) \bmod q$.
f. *Digital signature* valid jika $v = r$.

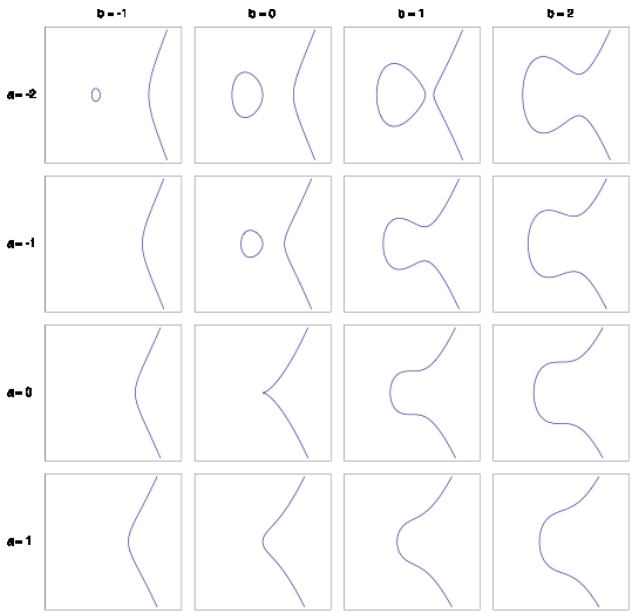


Gambar 2 – Verifikasi *Digital Signature*

III. KRIPTOGRAFI ELLIPTIC CURVE

Sebuah *elliptic curve* didefinisikan sebagai sebuah diagram Kartesius dua dimensi dengan sebuah persamaan:

$$y^2 = x^3 + ax + b$$

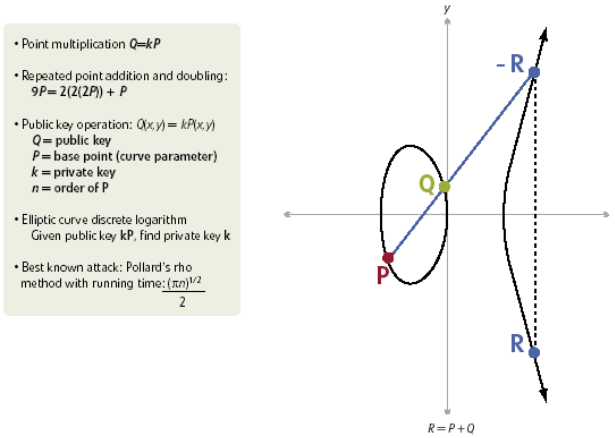


Gambar 3 – Berbagai Macam Elliptic Curve

Elliptic curves digunakan dalam berbagai bidang penting dalam riset. Sebagai contoh, *elliptic curve* digunakan sebagai bukti pada pembuktian oleh Andrew Wiles (dibantu Richard Taylor) pada *Fermat's Last Theorem*. Selain itu, ditemukan juga aplikasi *elliptic curve* pada kriptografi dan faktorisasi bilangan bulat.

Kriptografi *elliptic curve* adalah sebuah pendekatan

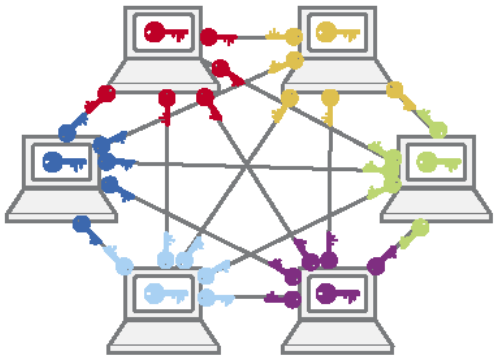
pada kriptografi *public key* berdasarkan struktur aljabar dari *elliptic curve* dari *finite fields*.



Gambar 4 – Ilustrasi Elliptic Curve Cryptography

Definisi lain mengenai *elliptic curve cryptography* adalah sebuah pendekatan untuk melakukan *asymmetric cryptography*. Algoritma *asymmetric cryptography* memunyai suatu sifat yaitu kita tidak bisa menggunakan sebuah kunci tunggal seperti pada *algoritma symmetric cryptography* (contohnya *Advanced Encryption Standard*). Pada *asymmetric cryptography* kita harus menggunakan sepasang kunci. Sebuah kunci yaitu *public key* digunakan untuk enkripsi sedangkan *private key*-nya digunakan untuk dekripsi.

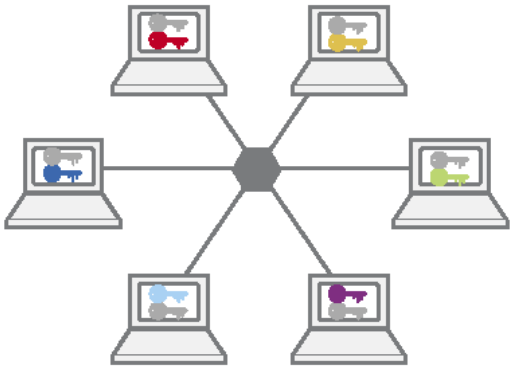
Keuntungan menggunakan sepasang kunci *public key* dan *private key* adalah seorang *cryptanalyst* tidak akan bisa memperoleh salah satu kunci dari kunci lainnya.



SYMMETRIC

Symmetric cryptography has an equation of $\frac{n \times n-1}{2}$ for the number of keys needed. In a situation with 1000 users, that would mean **499,500 keys**.

Gambar 5 – Symmetric Cryptography



ASYMMETRIC

Asymmetric cryptography, using key pairs for each of its users, has *n* as the number of key pairs needed. In a situation with 1000 users, that would mean **1000 key pairs**.

Gambar 6 – Asymmetric Cryptography

Penggunaan kurva elips dalam kriptografi diusulkan oleh Neal Koblitz dan Victor S. Miller pada 1985.

Kriptografi *public key* didasarkan pada sulit dipecahkannya beberapa masalah matematis. Sistem *public key* terbaru seperti algoritma RSA tergolong aman dengan mengasumsikan bahwa sulit untuk memfaktorkan sebuah bilangan bulat besar yang terdiri dari dua atau lebih faktor prima.

IV. ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

Elliptic curve digital signature algorithm adalah variasi dari algoritma *digital signature* menggunakan kriptografi *elliptic curve*.

Sebagaimana pada *digital signature*, terdapat tiga tahap dalam aplikasi *elliptic curve digital signature* ini, yaitu:

1. Pembangkitan *private key* dan *public key*.
2. Pemberian *digital signature*.
3. Verifikasi *digital signature*.

1. Pembangkitan *private key* dan *public key*.

Bit dari *public key* yang dibutuhkan oleh *elliptic curve digital signature algorithm* kira-kira dua kali dari tingkat keamanannya dalam bit. Jika seorang *cryptanalayst* membutuhkan 2^{80} pembangkitan *signature* untuk mencari *private key*, ukuran *public key* pada *digital signature algorithm* minimal adalah 1024 bits, sedangkan pada *elliptic curve digital signature algorithm* hanya sekitar 160 bit.

Sebelum membangkitkan *public key* dan private key, harus ditentukan parameter kurva elips terlebih dulu: (*q*,*FR*,*a*,*b*,[*DomainParameterSeed*,]*G*,*n*,*h*); *q* adalah ukuran medan; *FR* adalah basis yang digunakan; *a* dan *b* adalah dua elemen yang mendefinisikan persamaan kurva; *DomainParameterSeed* adalah bit string opsional; *G* adalah basis point; *n* adalah orde dari *G*; dan *h* adalah kofaktor (sama dengan orde dari kurva dibagi *n*).

Private key d_A adalah bilangan bulat acak dalam

interval [1,n − 1]. *Public key* Q_A adalah d_AG.

Elliptic-Curve Digital Signature Algorithm (ECDSA)

NIST Guidelines for Public Key Sizes for AES			
ECC key size (bits)	RSA key size (bits)	Key size ratio	AES key size (bits)
163	1,024	1:6	
256	3,072	1:12	128
384	7,680	1:20	192
512	15,360	1:30	256

Supplied by NIST to ANSI X9F1

Table 1

Tabel 1 – NIST Guidelines Untuk Ukuran Public Key

2. Pemberian *digital signature.*

Jika H adalah fungsi hash dan m adalah dokumen.:

a. Hitung $e = hash(m)$.
b. Pilih bilangan bulat acak k dari [1,n − 1].
c. Hitung $r = x_1(mod\ n)$, dimana $(x_1,y_1) = kG$. Jika $r = 0$, kembali ke langkah b.
d. Hitung $s = k - 1(z + rd_A)(mod\ n)$. Jika $s = 0$, kembali ke langkah b.
e. Signature yang diperoleh adalah pasangan (r,s).

3. Verifikasi *digital signature*.

Terdiri dari dua tahap: verifikasi sumber Q_A dan verifikasi hash.

Verifikasi sumber Q_A:

a. Cek apakah Q_A tidak sama dengan O dengan koordinat-koordinatnya, jika tidak, maka valid.
b. Cek apakah Q_A berada pada kurva.
c. Cek apakah $nQ_A = O$

Verifikasi hash:

a. Cek apakah r dan s adalah integer dalam interval [1,n − 1]. Jika tidak, signature tidak valid.
b. Hitung $e = hash(m)$.
c. Hitung $w = s - 1(mod\ n)$.
d. Hitung $u_1 = zw(mod\ n)$ dan $u_2 = rw(mod\ n)$.
e. Hitung $(x_1,y_1) = u_1G + u_2Q_A$.
f. Signature valid jika $r = x_1(mod\ n)$, dan tidak valid jika tidak demikian.

Berikut adalah contoh pengaplikasian *elliptic curve digital signature algorithm*:

1. Perangko digital

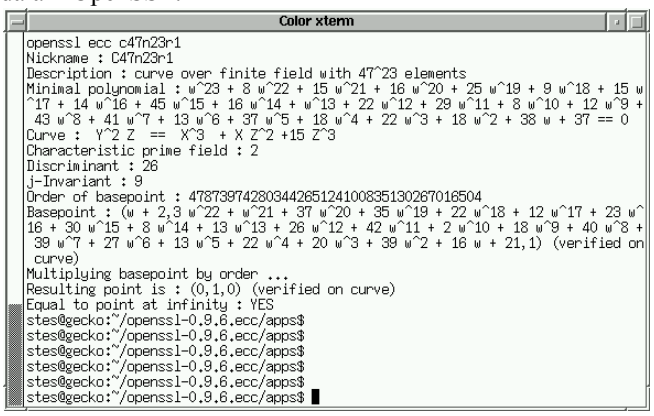
Dibandingkan dengan RSA, ECDSA menggunakan lebih sedikit ruang.

Gambar 7 – Penerapan ECDSA Pada Perangko Digital

Gambar 8 – Penerapan RSA Pada Perangko Digital

2. SSL

Penggunaan ECDSA pada yang diimplementasikan dalam OpenSSL.



Gambar 9 – ECDSA yang Diimplementasikan Dalam Open SSL

V. CONCLUSION

Elliptic curve digital signature algorithm layak menjadi standar dalam *digital signature algorithm* di masa depan karena:

1. Ukuran *publik key* yang kecil, dengan

perbandingan minimum 1:6.

2. Efisiensi lebih tinggi yang akan berefek domino pada kurangnya konsumsi daya dan sebagainya.

REFERENCES

http://www.mnhs.org/preserve/records/electronicrecords/erglossary.html ,waktu akses: 16 Desember 2010

http://www.deviceforge.com/articles/AT4234154468.html ,waktu akses: 16 Desember 2010

http://www.eetimes.com/design/communications-design/4025631/Understanding-elliptic-curve-cryptography

http://users.telenet.be/stes/ecc.html ,waktu akses: 16 Desember 2010

http://en.wikipedia.org/wiki/Cryptography ,waktu akses: 16 Desember 2010

http://en.wikipedia.org/wiki/Digital_Signature_Algorithm ,waktu akses: 16 Desember 2010

http://en.wikipedia.org/wiki/Elliptic_Curve_DSA ,waktu akses: 16 Desember 2010

http://en.wikipedia.org/wiki/Elliptic_curve ,waktu akses: 16 Desember 2010

http://en.wikipedia.org/wiki/Elliptic_curve_cryptography ,waktu akses: 16 Desember 2010

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, 17 Desember 2010

ttd

Ikmal Syfiai 13508003