PERBANDINGAN ALGORITMA BERBASIS ELLIPTIC CURVE CRYPTOGRAPHY DENGAN RSA DAN DSA PADA TANDA TANGAN DIGITAL

Dicky Wizanajani R 13503124

Program Studi Teknik Informatika,
Sekolah Tinggi Elektro dan Informatika,
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13124@students.if.itb.ac.id

Abstraksi

Tanda tangan digital dapat dimasukkan ke dokumen atau pesan untuk mengecek apaka sebuah data valid dan pemilik telah menandatanganinya. Kegunaannya sama seperti tandatangan seperti tanda tangan tertulis. Tanda tangan digital dapat membuktikan secara matematis bahwa isi dari sebuah dokumen tidak berubah. Tanda tangan digital adalah string of bits yang dikalkulasi secara unik dari fungsi yang diterapkan ke data yang ditandatangan dan kunci privat yang menandatangan. Hal ini mengakibatkan penandatangan tidak dapat menyangkal bahwa data yang ditandatangan valid.

Skema tanda tangan yang digunakan sekarang umumnya adalah RSA dan DSA, akan tetapi ada algoritma yang lebih baru dan efisien yaitu algoritma yang berbasis ECC Elliptic Curve Cryptography (ECC). Salah satu dari algoritma ECC adalah Elliptic Curve Digital Signature Algorithm (ECDSA) dan Elliptic-Curve Pintsov-Vanstone Signatures (ECPVS).

Di dalam makalah ini, akan dibahas perbedaan antara ketiga algoritma diatas yaitu DSA, RSA, dan ECC-based algorithma. Skala perbandingan yang akan dibahas diantara lain adalah kecepatan memberi tanda tangan, kecepatan memverifikasi tanda tangan, besar data yang dibutuhkan untuk tanda tangan digital, keumuman digunakannya algoritma, keunggulan-keunggulan dalam menghadapi serangan-serangan yang biasa dilakukan terhadap tanda tangan digital,

1. Pendahuluan

Peningkatan penggunaan jaringan untuk mengkomunikasikan data yang rahasia dan transaksi yang penting menyebabkan munculnya kebutuhan akan confidential pada identitas setiap orang, komputer, atau pelayanan yang terkait pada komunikasi tersebut. Tanda tangan digital dan kriptografi kunci publik menyediakan autentikasi dan privasi yang lebih tinggi untuk komunikasi digital yang tidak bisa didapatkan dari metode password.

Ada tiga algoritma dari tipe kunci yang biasa digunakan untuk tanda tangan digital: RSA, DSA, dan ECDSA (Elliptic Curve Digital Signature Algorithm). RSA dan DSA telah digunakan secara luas didalam aplikasi-aplikasi, dari mulai aplikasi anti-kloning sampai aplikasi *secure firmware*.

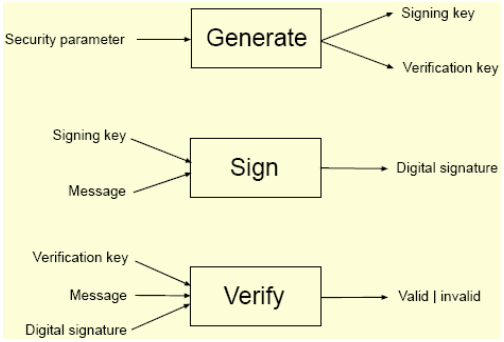
Akan tetapi, perubahan kebutuhan performansi dan keamanan seperti perubahan ke *mobile device* yang membutuhkan tanda tangan digital yang lebih kecil ukuran dan cepat pembuatannya. Kebutuhan ini bisa dipenuhi oleh tanda tangan berbais ECC yang memiliki keunggulan ukuran dan performansi.

Pada makalah ini akan dibahas ketiga algoritma kunci publik diatas yaitu RSA, DSA, dan ECC-Based Algoritma. Setelah penjelasan tersebut akan dibandingkan perbedaannya dari beberapa segi yaitu kecepatan, ukuran, dan lain-lain.

2. Tanda Tangan Digital

Telah disepakati bersama bahwa kemampuan untuk menyediakan layanan yang tidak dapat disangkal (non-repudiation services) adalah kebutuhan yang fundamental untuk aplikasi *e-commerce*. Tanda tangan digital adalah salah satu pilihan mekanisme keamanan yang menyediakan layanan tersebut. Tanda tangan digital juga dapat digunakan untuk menyediakan layanan keamanan juga seperti layanan autentikasi dan integritas data.

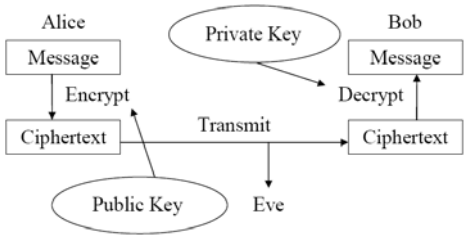
Tanda tangan digital mengacu kepada "data yang diembed ke atau transformasi kriptografi dari unit data yang mengizinkan penerima data untuk menguji sumber dari data dan integritas data sehingga melindungi data dari kepalsuan". Berikut adalah sistematika pembuatan standar tanda tangan digital



Gambar 1. Sistematika Tanda Tangan Digital

3. Kriptografi Kunci Public

Sistematika dari kriptografi kunci-publik adalah sebagai berikut, kunci kriptografi dibuat sepasang, satu kunci untuk enkripsi dan satu kunci untuk dekripsi. Kunci untuk enkripsi bersifat publik (public-key) dan kunci dekripsi bersifat rahasia (private key atau secret key). Kunci-kunci diciptakan dengan beberapa cara sehingga tidak mungkin menurunkan kunci rahasia dari kunci publik. Sistematika pengiriman pesan dengan menggunakan algoritma tersebut bisa dilihat di Gambar 2



Gambar 2. Sistematika Kriptografi Kunci Publik

Dengan adanya sistematika kriptografi kunci publik, pengiriman kunci rahasia melalui saluran komunikasi khusus seperti pada sistem kriptografi simetri tidak lagi diperlukan

Keamanan sistem kriptografi kunci publik terletak pada dua hal:

1. Sulitnya menurunkan kunci rahasia dari kunci publik.
2. Sulitnya menurunkan plainteks dari cipherteks.

Sedangkan kelemahannya terletak pada hal-hal berikut:

1. Proses enkripsi dan dekripsi jauh lebih lambat dari sistem simetri karena proses nya melibatkan operasi exponensial yang besar.
2. Ukuran dari cipherteks lebih besar daripada plainteks
3. Kunci publik diketahui secara luas dan dapat digunakan setiap orang, maka cipherteks tidak memberikan informasi mengenai otentikasi pengirim.

Dibawah ini akan dijelaskan tiap-tiap algoritma yang akan dibandingkan, dimulai dari RSA, DSA, lalu algoritma kriptografi berbasis *elliptic curve*.

3.1. RSA

Algoritma ini dipublikasikan pada tahun 1977 oleh Ron Rivest, Adi Shamir dan Len Adleman di MIT. **RSA** adalah inisial dari nama depan mereka. Algoritma RSA adalah yang paling populer digunakan. RSA melibatkan kunci publik dan kunci private. Kunci publik digunakan untuk mengenkripsi pesan dan didekripsi dengan mengunakan kunci privat. Skema tanda tangan dengan RSA menyediakan message recovery.

Keamanan dari skema RSA bergantung kepada tingkat interaktibilitas dari masalah

faktorisasi integer. Pada RSA, algoritma enkripsi dan dekripsi identik, sehingga proses signature dan verifikasi juga identik atau dengan kata lain transformasi enkripsi dilakukan secara bijeksi, tanda tangan digital dapat dibuat dengan membalik peran dari enkripsi dan dekripsi.

Pembuatan kunci pada skema tanda tangan RSA (asumsikan bahwa A adalah yang miliki data):

1. Ciptakan dua nilai prima besar yang berbeda p and q, panjangnya kira-kira sama
2. Hitung n = pq, n digunakan sebagai modulus untuk kunci privat dan kunci publik.
3. Hitung $\Phi(n) = (p - 1)(q - 1)$.
4. Pilih angka acak e, $1 < e < \Phi(n)$, dimana gcd(e, $\Phi(n)$) = 1, artinya e dengan $\Phi(n)$ saling prima satu dengan yang lain.
5. Gunakan algoritma Euclidian untuk menghitung integer unik d. $1 < d < \Phi$, dimana $ed \equiv 1 \pmod{\Phi(n)}$.

Kunci publik mengandung modulus n dan eksponen e.

Kunci privat mengandung modulus n dan eksponen d. Untuk effisiensi kunci privat juga boleh berisi (hanya untuk kemudahan perhitungan):

1. p dan q. Bilangan prima untuk membuat kunci
2. d mod (p-1) dan d mod (q-1) sering disebut *dmp1* dan *dmq1*
3. q^{-1} mod (p). Sering disebut iqmp

Cara untuk menciptakan tanda tangan digital oleh A adalah:

1. Hitung (~m) = R(m), sebuah integer pada kisaran [0, n-1]
2. Hitung $s = (\sim m)^d \bmod n$.
3. Tanda tangan A untuk pesan m adalah s

Asumsikan dari proses pembuatan kunci tersebut A telah menandatangani pesan m dengan kunci publik B. Dan B mendapat pesan m dari A. B dapat melakukan verifikasi terhadap tanda tangan digital tersebut.

Cara untuk memverifikasi tanda tangan digital oleh B adalah dengan cara sebagai berikut:

1. Ambil kunci publik yaitu n dan e
2. Hitung $\sim m = s^e \bmod n$
3. dengan mengetahui ~m, B dapat mendapatkan m.

Tanda tangan digital berfungsi pada kasus sebagai berikut: jika s adalah tanda tangan untuk pesan m, maka $s \equiv (\sim m)d \bmod n$ dimana (~m) = R(m). Karena $ed \equiv 1 \pmod{\Phi(n)}$, $s^e \equiv \sim m^{ed} \equiv \sim m \pmod n$. Terakhir, $R^{-1}(\sim m) = R^{-1}(R(m)) = m$

Berikut diataas adalah sistematika tanda tangan digital dengan message recovery. Jika sistematika DSA hanya akan melakukan *insert appendix* maka proses penyisipan dan pengecekan tanda tangan digital dilakukan dengan:

1. Ketika A akan menandatangani pesan m, A menghitung $s \equiv m^d \pmod n$.
2. B memverifikasi bahwa A telah menandatangani pesan dengan menghitung $s^e \pmod n$ dan membandingkannya dengan m

Dengan memilih nilai exponen publik yang kecil, kecepatan verifikasi tanda tangan bisa menjadi lebih cepat daripada kecepatan menandatanganinya.

3.2. DSA

DSA dikembangkan dari algoritma ElGamal. DSA tidak dapat digunakan untuk enkripsi tetapi dispesifikasikan khusus untuk tanda-tangan digital. Pada DSA, algoritma signature dan verifikasi berbeda. DSA menggunakan dua buah kunci, yaitu kunci publik dan kunci privat. Pembentukan tanda-tangan menggunakan kunci rahasia privat, sedangkan verifikasi tanda-tangan menggunakan kunci publik pengirim.

DSA menggunakan fungsi hash SHA (Secure Hash Algorithm) untuk mengubah pesan menjadi message digest yang berukuran 160 bit

Tingkat keamanan DSA bergantung kepada dua masalah logaritma yang jauh akan tetapi saling berkaitan. Pertama adalah masalah logaritmik di Zp dimana metode index kalkulus yang kuat diterapkan. Yang kedua adalah masalah logaritma pada *cyclic*

subgroup pada urutan q, dimana metode berjalan pada waktu yang "square-root"

Mekanisme tanda tangan menggunakan sebuah fungsi hash h: (0,1)* → Zq untuk sebuah integer q.

Pembuatan kunci untuk algoritma DSA adalah sebagai berikut:

1. Ambil sebuah angka prima q (160-bit) dimana $2^{159} < q < 2^{160}$
2. Ambil t dimana $0 \leq t \leq 8$
3. Ambil bilangan prima p dimana $2^{511 + 64t} < q < 2^{512 + 64t}$, dengan tambahan bahwa q dapat dibagi oleh (p - 1)
4. Pilih sebuah generator α dari unique cyclic group berdasar urutan q di Zp*
   4.1. Pilih sebuah elemen $g \in Zp^*$ dan hitung $\alpha = g^{(p-1)/q} \bmod p$
   4.2. Jika $\alpha = 1$ ulangi langkah diatas
5. Ambil bilangan integer acah dimana $1 \leq a \leq q-1$
6. Hitung $y = \alpha^a \bmod p$

Kunci publik mengandung p, q, α, y

Kunci privat adalah a.

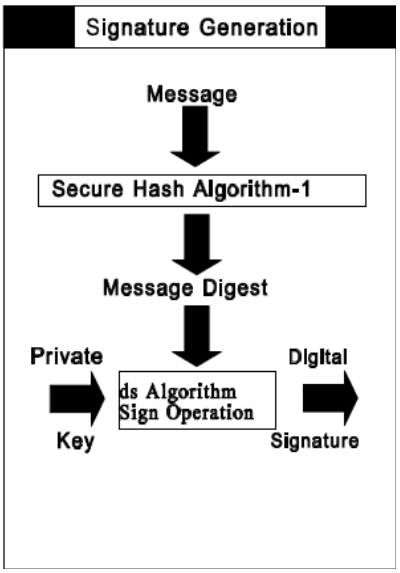
Tanda tangan digital diperoleh dengan melakukan langkah-langkah berikut (skema dalam gambar dapat dilihat pada **Gambar 3**):

1. memilih sebuah angka rahasia k, dimana $0 < k < q$
2. Hitung $r = (\alpha^k \bmod p) \bmod q$
3. Hitung $k^{-1} \bmod q$
4. Hitung $s = k^{-1}\{h(m) + ar\} \bmod q$

Tanda tangan A untuk pesan m adalah pasangan (r,s).

Tanda tangan digital dapat diverifikasi dengan langkah-langkah sebagai berikut (skema dalam gambar bisa dilihat pada):

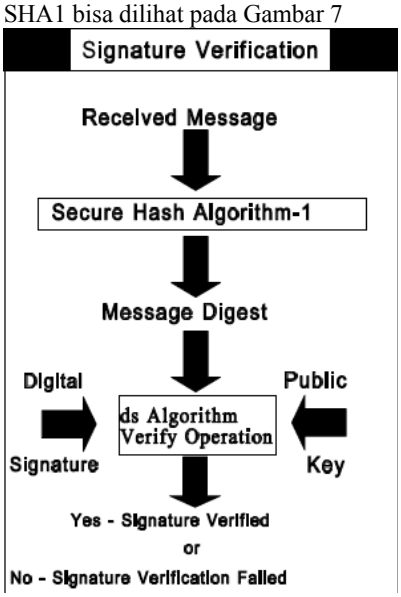
1. Ambil kunci publik (p, q, α, y)
2. Verifikasi bahwa $0 < r < q$ dan $0 < s < q$; jika tidak maka tolak tanda tangan digital tersebut.
3. Hitung $w = s^{-1} \bmod q$ dan h(m)
4. Hitung:
   $u_1 = w.h(m) \bmod q$
   $u_2 = rw \bmod q$
5. Hitung $v = (\alpha^{u1} y^{u2} \bmod p) \bmod q$
6. Terima tanda tangan jika dan hanya jika v=r



Gambar 3. Pembuatan tanda tangan pada DSA

SHA sendiri adalah fungsi hash satu-arah yang didasarkan pada MD4. Algoritma SHA menerima masukan berupa pesan dengan ukuran maksimum 2^{64} bit dan menghasilkan message digest yang panjangnya 160 bit.

Skema pembuatan message digest dengan SHA1 bisa dilihat pada Gambar 7



Gambar 4. verifikasi tanda tangan pada DSA

3.3. ECC

ECC memiliki keamanan berbasis pada salah satu masalah matematika yaitu *elliptic curve* (kurva berbentuk elips). *elliptic curvei* merupakan operasi yang menyediakan fungsi satu atah yang bisa digunakan untuk menghasilkan sistem kriptografis yang efisien. Fungsi satu arah yang digunakan pada ECC disebut *elliptic curve discrete logarithm problem* (ECDLP).

Operasi kriptografi yang utama pada ECC adalah scalar point multiplication dimana komputer menghitung Q = [a]P, point P dimultiplikasi oleh integer a yang menghasilkan point Q di kurva. Multiplikasi skalar dilakukan melalui kombinasi penambahan point dan menggandaan point.

Grup elliptic curve yang digunakan pada kriptografi didefinisikan menjadi dua jenis:

1. GF(p), dimana p adalah prima
2. GF(2m) dimana setiap elemen adalah polinomial biner tingkat m (yang direpresentasikan sebagai m-bit string karena tiap koefisien hanya bisa bernilai 0 atau 1)

Parameter umum yang digunakan:

- F_q = field terbatas dengan q elemen
- E = elliptical curve pada F_q, *group order* n
- P = generator dari *group of order* l *dengan* l|n.

Inisiasi awal:

1. Memilih a dengan acak,
2. Menghitung Q = [a]P

Kunci publik adalah Q.
Kunci Privat adalah a.

Langkah-langkah yang dilakukan ketika memberi tanda tangan:

1. Memilih k
2. Menghitung K = [k]P
3. Menghitung $s \equiv (k^{-1}(h(m) - ah(K)))$ mod l
4. Tanda tangan adalah (K,s)

Langkah-langkah yang dilakukan ketika memverifikasi tanda tangan:

1. Mengambil Q
2. menghitung $R^1 = [h(K)]Q \oplus [s]K$
3. menghitung $R^2 = [h(m)]P$
4. Tanda tangan diterima jika R^1 sama dengan R^2

4. Perbandingan Antara RSA, DSA, dan ECC

Untuk perbandingan algoritma berbasis ECC yang digunakan adalah ECDSA (Elliptic Curve Digital Signature Algorithm)

4.1. Kecepatan

Algorithm	Key Generation * 1(ms.)	Sign * 100 (ms.)	Verify*100(ms.)
RSA 512	544.61	915	160
RSA 1024	1120.46	4188	263
DSA 512	6.62	634	988
DSA 1024	17.87	1775	3397

Gambar 5. Perbandingan kecepatan pembuatan kunci, proses tanda-tangan, proses verifikasi tanda tangan antara RSA dan DSA

Dari tabel diatas ada beberapa hal yang bisa disimpulkan:

1. Kecepatan pembuatan kunci DSA lebih cepat dari RSA
2. Kecepatan penandatanganan DSA lebih cepat dari RSA tapi perbedaannya tidak berfaktor terlalu besar.
3. Kecepatan verifikasi algoritma RSA jauh lebih cepat dari DSA

Berikut adalah perbandingan antara kecepatan RSA dan ECC:
Kecepatan Algoritma RSA

- Kecepatan membuat tanda tangan berdasar:
  Komputer menghitung hash dan menghitung 1 modular exponential.
- Kecepatan memverifikasi tanda tangan:
  Komputer menghitung hash dan menghitung 1 modular exponential.

Kecepatan Algoritma berbasis ECC

- Kecepatan membuat tanda tangan berdasar:
  Komputer penandatangan melakukan hash, 1 multiplikasi skalar dan 1 multiplikasi modulo.
- Kecepatan memverifikasi tanda tangan:
  Komputer menghitung hash, 1 multiplikasi skalar dan 1 multi eksponensial

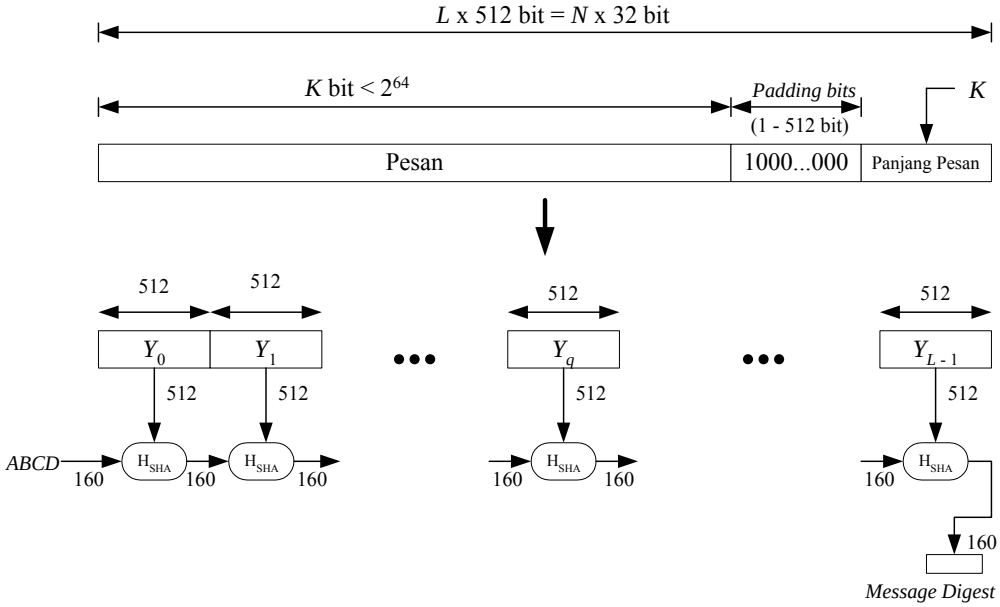
ECC key size	RSA key size	Symmetric Key	ECDSA sign (sigs/min)	RSA Sign (sigs/min)	ECC Benefit (sign)	ECDSA verify (sigs/min)	RSA verify (sigs/min)	ECC Benefit (verify)
224 bit	2048 bit	3-DES (112-bit)	105840	2940	3600%	47520	26880	177%
256 bit	3072 bit	AES-128	54000	480	11250%	22800	11280	202%
384 bit	7680 bit	AES-192	30960	60	51600%	11040	2160	511%
521 bit	15360 bit	AES 256	14400	60	24000%	5280	480	1100%

Gambar 6. Perbandingan ECC dan RSA dalam kecepatan memberi dan memverifikasi tanda tangan

Dari fakta di atas kita bisa lihat bahwa RSA menggunakan perhitungan modular eksponensial pada saat memberi tanda tangan ataupun saat memverifikasi tanda tangan. Sedangkan pada ECC, pada menandatangan hanya melakukan multiplikasi modulo dan skalar saja dan saat memverifikasi menggunakan satu kali multi eksponensial. Bisa kita katakan bahwa proses ECC lebih cepat dari RSA

Dapat dilihat dari Gambar 6 bahwa kecepatan menandatangan dan memverifikasi tanda tangan oleh algoritma berbasis ECC jauh lebih cepat daripada yang menggunakan algoritma RSA

Tanda tangan berbasis ECC pada sertifikat bersifat lebih kecil dan lebih cepat diciptakan dan kunci publik dimana tersimpan pada sertifikat juga lebih kecil dan *agile.* Verifikasi juga lebih cepat terutama pada kekuatan kunci yang lebih tinggi.



Gambar 7. Skema pembuatan message digest oleh SHA

Pada implementasinya, untuk aplikasi dimana performansi penandatanganan adalah yang paling penting seperti SSL pada aplikasi web server. Sistematika pembentukan kunci dan penandatangan yang cepat dibutuhkan. Sedangkan, jika kecepatan verifikasi tanda tangan sangat cepat maka skema tandatangan cocok digunakan untuk situasi dimana verifikasi tanda tangan adalah operasi yang cukup penting atau cukup dominan yang dilakukan. Misalnya, ketika pihak ketiga yang dipercata menciptakan sertifikat kunci publik untuk A, ini mungkin memerlukan satu kali pembentukan tanda tangan, tapi tanda tangan perlu untuk diverifikasi berkali-kali oleh banyak entitas.

4.2. Ukuran

Ukuran Kunci Publik:
Kunci public RSA mengandung (n,e) dimana n adalah modulus dan e adalah eksponen publik. Di 1024-bit RSA. N akan memiliki 1024 bit. Dan nilai untuk eksponen publik $e=2^{16}+1$. Lalu, kunci publik RSA akan memiliki 128 byte untuk modulud dan 3 byte untuk ekponen publik, total adalah 131 byte.

Kunci publik ECC mengandung point dari *elliptic curve*. Setiap point direpresentasikan dengan elemen (x,y) masing-masing dengan 192 bit. Untuk 192 bit *elliptic curve*, kunci publik akan direpresentasikan oleh 48byte.

Ukuran tanda tangan:
Tanda tangan RSA mengandung 1024 bit = 128 byte.
Tanda tangan ECDSA mengandung dua 192 bit = 48 byte

4.3. Serangan yang bisa dilakukan dan kekuatan algoritma

Serangan yang paling dikenal untuk RSA dan DSA adalah *Number Field Sieve (NFS)*. NFS adalah apa yang diketahui sebagai **sub-exponential time method**. Artinya, masalah tersebut bisa dikategorikan sulit dipecahkan tetapi tidak sesulit masalah yang hanya menyediakan penyelesaian exponensial (artinya masalah bukan penuh masalah eksponensial).

Sedangkan disisi lain, menyelesaikan masalah ECDLP dianggap lebih sulit daripada memfaktorkan integer atau menyelesxaikan masalah logaritma diskrit. Karena struktur berada dalam *elliptic curve,* tipe solusi NFS tidak bisa diterapkan pada ECDLP. Metode yang dikenal untuk menyelesaikan ECDLP adalah versi penyerangan *elliptic curve* yang dikenal sebagai metode *parallel collision search*. Metode tersebut bertipe exponensial penuh, yang berarti ECDLP dapat dianggap sebagai salah satu masalah yang paling sulit untuk dipecahkan.

Pada kenyataannya, 170-bit *elliptic curve system* memiliki equivalensi keamanan dengan 1024-bit RSA *system.*

Public-key system	Mathematical Problem	Best known method for solving (running time)
Integer factorization *e.g.* RSA	Given a number *n*, find its prime factors	Number field sieve: $\exp[1.923(\log n)^{1/3} (\log \log n)^{2/3}]$ (Sub-exponential)
Discrete logarithm *e.g.* DH, DSA	Given a prime *n*, and numbers *g* and *h*, find *x* such that $h = g^x \bmod n$	Number field sieve: $\exp[1.923(\log n)^{1/3} (\log \log n)^{2/3}]$ (Sub-exponential)
Elliptic curve Discrete logarithm *e.g.* ECDH,ECDSA	Given an elliptic curve and points P and Q find *k* such that Q = *k*P	Pollard-rho algorithm: sqrt(n) (Fully exponential)

Gambar 8. Perbandingan kekuatan ECC, DSA dan RSA

5. Kesimpulan

Kecepatan algoritma berbasis ECC dalam proses pemberian dan verifikasi tanda tangan digital jauh lebih cepat dibandingkan dengan algoritma DSA dan RSA. Ukuran dari Algritma berbasis ECC juga lebih kecil dari RSA dan DSA akan tetapi tidak dalam nilai yang besar. Yang paling utama adalah penyelesaian masalah algoritma berbasis ECC dilakukan secara eksponensial penuh dan lebih sulit dipecahkan daripada masalah DSA dan RSA, hal ini membuktikan bahwa ECC lebih kuat daripada DSA dan RSA.

Jadi, berdasarkan perbandingan yang dilakukan. Bisa disimpulkan bahwa algoritma ECC memiliki banyak keunggulan dibandingkan dengan algritma RSA dan DSA.

Pada implementasinya ECC cocok digunakan untuk aplikasi pada mobile phone, PDA, karena aplikasi tersebut membutuhkan ukuran kunci dan kecepatan pemberian tanda tangan serta verifikasi yang cepat.

6. Referensi

[1]Zuccherato, Robert.“Elliptic Curve Cryptography Support in Entrust”.

[2]Certicom ECC Certificate Utility

[3] Chang, Sheueling, Eberle, Hans, Gupta, Vipul, Gura, Nils.”Elliptic Curve Cryptography – How it Works”

[4] A. Menezes, P. van Oorschot, and S. Vanstone. “Handbook of Applied Cryptography”

[5] Lange, Tanja. “Public Key Cryptography - Performance Comparison and Benchmarking“

[6]Wikipedia: RSA, DSA, ECC