Transaksi Web dengan Protokol SSL Menggunakan Algoritma ECC

Hari Bagus Firdaus – NIM: 13506044

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
Email: if16044@students.if.itb.ac.id

Abstrak

Elliptic Curve Cryptography (ECC) adalah sebuah pendekatan kriptografi kunci publik yang berbasiskan struktur aljabar dari kurva elips pada sebuah bidang. Algoritma kriptografi ini muncul sebagai alternatif yang cukup menarik dari kriptografi kunci publik tradisional seperti RSA, DSA, dan *Diffie-Hellman*. Keuntungan dari ECC adalah bahwa ECC menawarkan keamanan yang setara tetapi dengan ukuran kunci yang lebih kecil sehingga menghasilkan proses komputasi yang lebih cepat dan konsumsi daya listrik yang lebih rendah, serta menghemat memori dan *bandwidth*.

Dewasa ini, melakukan transaksi pada sebuah *website* secara *online* sudah bukan hal baru dalam dunia internet. Contoh yang paling mudah adalah membeli suatu barang secara *online* pada sebuah situs *e-commerce* seperti *amazon* dan *e-bay*. Protokol yang digunakan tentulah berbeda dari protokol yang digunakan *browser* untuk melakukan *request* biasa kepada sebuah *web-server*. Protokol yang paling populer untuk mengatasi hal ini adalah SSL atau *Secure Socket Layer*. Pada makalah ini akan dibahas mengenai penerapan dan performansi algoritma ECC dalam SSL sehingga berakibat pada pemrosesan transaksi web yang lebih cepat.

Makalah ini akan membahas *overview* mengenai ECC, karakteristiknya, serta kelebihan dan kekurangannya dibandingkan dengan algoritma kunci publik lainnya. Selain itu makalah ini juga akan menjelaskan secara singkat tentang SSL dan juga cara kerjanya dalam suatu transaksi web. Dan yang terakhir adalah penerapan ECC pada protokol SSL serta analisis dan perbandingan performansi yang terjadi dengan algoritma kunci publik yang dominan digunakan pada SSL, algoritma RSA. Perbandingan ini sedikit mengarah kepada *benchmarking* atara ECC dan RSA, yang dilakukan dengan berbagai parameter dan jenis input data.

Kata kunci: ECC, SSL, transaksi web, kriptografi kunci publik

1. Pendahuluan

Komunikasi yang aman adalah suatu persyaratan utama pada transaksi *online* di dunia saat ini. Dalam hal bertukar informasi finansial, bisnis, ataupun informasi pribadi, orang ingin mengetahui dengan siapa ia berkomunikasi (otentikasi *user*) dan mereka berharap untuk memastikan bahwa informasi yang dipertukarkan tidak dimodifikasi (integritas data) dan tidak diketahui (kerahasiaan data) pada saat *transit*. *Secure Socket Layer* (SSL) adalah pilihan yang paling populer untuk mengatasi masalah ini.

Protokol SSL merupakan protokol yang tidak bergantung pada jenis aplikasi, secara konseptual. Artinya, aplikasi yang sedang berjalan diatas TCP dapat juga berjalan siatas SSL. Ini adalah alasan penting mengapa pengembangan SSL menjadi sangat pesat bila dibandingkan dengan protokol keamanan lainnya seperti SSH, S/MIME, dan SET. Ada banyak contoh protokol aplikasi seperti TELNET, FTP, IMAP, dan LDAP yang dapat beroperasi diatas SSL. Akan tetapi, penggunaan SSL yang paling umum adalah untuk keamanan protokol utama dalam sistem terdistribusi World Wide Web, yaitu HTTP.

Saat ini, SSL dipercaya untuk keamanan aplikasi-aplikasi yang sensitif, seperti perbankan web, perdagangan saham, dan perdagangan *online* atau *e-commerce*. Sayangnya, penggunaan SSL juga dibarengi dengan penurunan performansi yang cukup signifikan pada web *server*. Penelitian melaporkan bahwa *web server* dengan SSL beroperasi 3,4 sampai 9 kali lebih lambat dibanding dengan web *server* biasa, dengan *platform* perangkat keras yang sama. Waktu respon yang lama adalah penyebab utama para pengunjung situs *e-commerce* sering membatalkan belanjanya pada saat *check-out*. Hal ini cukup disoroti karena nilai dari transaksi yag batal tersebut cukup merugikan perusahaan penjual.

Pada penggunaan biasa, SSL mengutilisasi enkripsi RSA dalam pengiriman pesan rahasia acak yang digunakan untuk menurunkan kunci untuk enkripsi data dan autentikasi. Operasi dekripsi RSA adalah bagian yang proses komputasinya paling intensif dari sebuah transaksi SSL pada suatu *secure web*

server. Beberapa *vendor* terkemuka dunia sudah menawarkan solusi yaitu *hardware* khusus untuk meng-*offload* komputasi RSA dan membantu memperbaiki performansi *server*.

Makalah ini mengeksplorasi pemakaian *Elliptic Curve Cryptography* (ECC), sebuah alternatif dari RSA, sebagai alat untuk mengingkatkan performa SSL tanpa perlu menggunakan *hardware* khusus yang berharga mahal. Sejak pertama kali diperkenalkan oleh Victor Miller dan Neal Koblitz pada pertengahan 1980, ECC sudah berkembang menjadi sebuah sistem kriptografi kunci publik yang cukup diandalkan.

Dibandingkan dengan para pendahulunya, ECC memberikan tingkat keamanan yang sama menggunakan kunci yang jauh lebih kecil. Hal ini berdampak pada proses komputasi yang lebih cepat dan penghematan kapasitas pada memori, daya listrik, dan *bandwidth*, yang cukup penting pada beberapa lingkungan dengan *contraint* tinggi seperti ponsel, PDA, dan *smart card*. Dan yang terpenting, keuntungan dari ECC dari para kompetitornya meningkat seiring meningkatnya kebutuhan akan sistem keamanan dari waktu ke waktu.

2. Overview ECC

Fondasi dari setiap kriptografi kunci publik adalah masalah matematis yang rumit yang secara komputasi akan sulit dipecahkan. Tingkat kesulitan relatif dalam memecahkan masalah ini menentukan kekuatan keamanan dari sistem kriptografi tersebut. Kriptografi kunci publik seperti RSA, *Diffie-Hellman*, dan DSA memungkinkan dilakukannya serangan yang menerapkan algoritma sub-eksponensial, namun serangan terhadap ECC membutuhkan waktu yang eksponensial. Dengan alasan ini, maka ECC dapat memberikan keamanan yang setara dengan ukuran kunci yang lebih kecil.

Tabel 1. Perbandingan algoritma kunci publik

Kunci Publik	Masalah Matematis
RSA, Rabin-Williams	Diberikan sebuah bilangan *n*, temukan faktor-faktor primanya
Diffie-Hellman, DSA, ElGamal	Diberikan sebuah bilangan prima *n*, dan bilangan *g* dan *h*, temukan *x* dimana $h = g^x \bmod n$
ECDH, ECDSA	Diberikan sebuah kurva elips *E* dan titik *P* dan *Q* pada *E*, temukan *x* dimana $Q = xP$

Skema kunci publik sendiri secara khusus digunakan untuk transportasi atau pertukaran kunci dari kriptografi kunci simetri, dan kerja yang dibutuhkan untuk memecahkan kunci simetri itu harus relatif sama dengan yang dibutuhkan untuk memecahkan sistem kunci publik yang digunakan dalam pertukaran kunci. Tabel berikut menunjukkan ekuivalensi secara komputasi dari ukuran kunci simetri dan kunci publik.

Tabel 2. Ekuivalensi ukuran kunci (dalam bit)

Simetri	ECC	RSA/DH/DSA
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

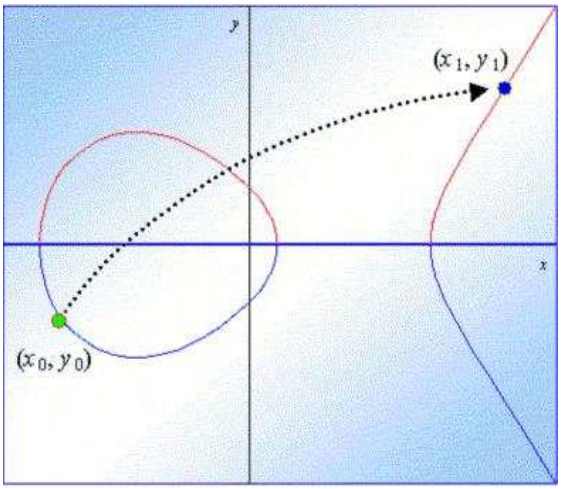
Terlihat bahwa penggunaan RSA 1024 bit tidak sesuai dengan 128 bit atau bahkan 112 bit tingkat keamanan yang sekarang digunakan untuk *cipher* simetri pada SSL. Ini menandakan perlunya untuk berpindah pada RSA dengan ukuran kunci yang lebih besar untuk mengantarkan keamanan penuh dari algoritma kunci simetri dengan kunci lebih dari 80 bit. Dengan semakin besarnya ukuran kunci, masalah performansi RSA akan menjadi semakin buruk. Karena itu, migrasi kepada ECC mungkin dapat memberikan alternatif.

Tidak seperti kriptografi kunci publik konvensional yang didasari oleh aritmatika pada bidang yang terbatas (*finite field*), ECC beroperasi pada kumpulan poin pada sebuah kurva elips yang didefinisikan diatas sebuah bidang terbatas. Operasi kriptografi utamanya adalah *scalar point multiplication*, yang akan menghitung $Q = k\ P$ (sebuah titik *P* dikalikan dengan suatu bilangan bulat *k* menghasilkan sebuah titik *Q* pada kurva). Perkalian skalar dilakukan lewat sebuah kombinasi dari penambahan titik dan *doubling* titik. Sebagai contoh, $11P$ dapat dijabarkan menjadi $11P = (2((2(2P)) + P)) + P$.

Keamanan ECC bersandar pada tingkat kesulitan dalam memecahkan Masalah Logaritma Diskrit dari Kurva Elips (*Elliptic Curve Discrete Logarithm Problem* / ECDLP), yang menyatakan bahwa jika diberikan *P* dan $Q = kP$, akan sangat susah untuk menemukan *k*. Selain persamaan kurva, parameter penting lain dalam kurva elips adalah titik awal (*base point*), *G*, yang bernilai tetap untuk setiap kurva. Pada ECC, suatu bilangan bulat acak bernilai besar *k* bertindak sebagai kunci privat, sedangkan hasil perkalian kunci privat *k* dengan titik awal kurva *G* berkorespondensi sebagai kunci publiknya.

Contoh ECC sederhana dapat digambarkan sebagai berikut. Alice dan Bob sepakat pada sebuah kurva elips, sebut saja $y^2 = x^3 + ax + b$, dan mengambil sebuah titik awal *G* (x_0,y_0) pada kurva ini. Mereka dapat melakukan ini tanpa kerahasiaan. Alice

kemudian memilih sebuah bilangan acak *As* dan menghitung *Ap* = *As***G* (*Ap* (x_1,y_1) merupakan sebuah titik pada kurva). *As* akan menjadi kunci privat Alice, *Ap* akan menjadi kunci publiknya. Bob juga akan melakukan hal yang sama dan menghasilkan *Bs* dan *Bp*.



Gambar 1. Titik *G* dan *Ap* pada kurva elips

Alice dan Bob kemudian sepakat pada sebuah kunci yang akan digunakan pada enkripsi pesan dari Bob ke Alice. Kunci tersebut adalah perkalian dari kunci privat Bob dan kunci publik Alice (yang akan sama dengan perkalian kunci privat Alice dan kunci publik Bob): *As*Bp = As*Bs*G = Bs*Ap*. Apabila Eve ingin memecahkan kunci itu, maka ia harus merekonstruksi satu dari kunci privat yang ada. Ini berarti ia harus menghitung *As* setelah mengetahui *Ap* dan *G* (ECDLP), yang tentu saja sangat sulit. Jika *G* berukuran besar, misalnya 160 bit , maka Eve membutuhkan sekitar 2^{80} operasi: jika dihitung dengan *billion operations per second*, hal ini memakan waktu sekitar 38 juta tahun.

Ada beberapa jenis skema ECC, diantaranya pertukaran kunci *Elliptic Curve Diffie-Hellman* (ECDH) yang berbasis algoritma *Diffie-Hellman*, *Elliptic Curve Digital Signature Algorithm* (ECDSA) yang berbasis algoritma tandatagan digital, dan pertukaran kunci ECMQV yang berbasis skema pertukaran kunci MQV. Contoh yang dipaparkan diatas adalah salah satu contoh ECDH.

3. Overview Protokol SSL

Pengamanan data dalam sebuah transaksi web harus menggunakan hal-hal berikut ini:

- Skema kriptografi asimetris
  Komputer yang akan berkomunikasi menggunakan data terenkripsi harus memiliki dua buah kunci untuk mengenkripsi data dan mendekripsinya. Kunci publik untuk siapapun yang ingin melakukan komunikasi dengannya, dan kunci privat yang bersifat rahasia.

- Sertifikat
  Untuk memastikan apakah data yang diperoleh adalah bnar dari pihak yang memiliki otoritas, bukan dari pihak lain yang tidak berkepentingan. Dibutuhkan pihak ketiga untuk memverifikasi pesan yang dikirim, yang berisi semacam *signature* sebagai tanda identitas pengirim.

SSL hanya digunakan untuk komunikasi yang bersifat *connection-oriented* dan hanya mengenkripsi data yang dikirim lewat HTTP. SSL berperan sebagai protokol tambahan dimana kunci dan sertifikat dari suatu situs *e-commerce* akan ditransfer ke *browser* atau *server* lain, yang akan melakukan verifikasi sertifikat dari situs tersebut sehingga dapat mengetahui identitas pengirim yang sebenarnya.

SSL adalah protokol keamanan internet yang paling banyak digunakan saat ini. SSL memberikan layanan enkripsi, autentikasi sumber, dan proteksi integritas data serta cukup fleksibel dalam penggunaan algoritma kriptografi yang berbeda untuk pertukaran kunci, enkripsi, dan *hashing*. Spesifikasi dari kombinasi algoritma ini disebut dengan *cipher suite*. Contoh *cipher suite* adalah *TLS_RSA_WITH_RC4_128_SHA*, yang diartikan menggunakan RSA untuk pertukaran kunci, RC4 128 bit untuk enkripsi bit dalam jumlah besar (*bulk encryption*), dan SHA untuk *hashing*.

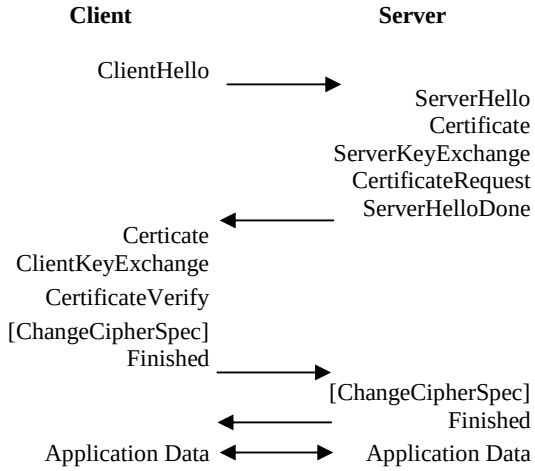
Dua komponen utama SSL adalah *Handshake* dan *Record Layer*. Protokol *Handshake* mengijinkan klien dan server SSL untuk menentukan *cipher suite* yang dipakai, mengotentikasi satu sama lain, dan membangun kunci rahasia (*master secret*) dengan algoritma kunci publik. Protokol *Record Layer* menurunkan kunci-kunci simetri dari *master secret* dan menggunakannya pada algoritma kunci simetri untuk enkripsi bit dalam jumlah besar dan autentikasi data.

4. Penerapan ECC dalam SSL

Sistem kriptografi kunci publik yang paling populer digunakan saat ini adalah RSA. Hal yang akan dilakukan adalah menganalisis dampak dan perubahan performansi yang terjadi apabila RSA digantikan dengan ECC. Suatu transaksi web via HTTPS tidak hanya melibatkan kriptografi kunci publik, tetapi juga operasi-operasi lainnya yaitu enkripsi simetris, *hashing*, *message parsing*, dan akses terhadap *file system*, sehingga diasumsikan bahwa *workload* untuk masing-masing operasi lainnya adalah sama.

4.1 *Handshake* berbasis RSA

Merupakan metode kriptografi kunci publik yang paling umum digunakan dalam pembangunan suatu *master secret*. Berikut ini diperlihatkan garis besar mekanismenya:



Gambar 2. Mekanisme *Handshake* dengan RSA

Klien dan server pertama-tama bernegosiasi tentang *cipher suite* dengan pesan *ClientHello* dan *ServerHello*. Server lalu mengirim kunci publik RSA-nya (*e*,*n*) dalam pesan *Certificate*. Klien melakukan verifikasi kunci RSA server dan menggunakannya untuk mengenkripsi 48-*byte* bilangan acak (*pre-master secret*, $r^e \bmod n$), yang dikirimkan lewat pesan *ClientKeyExchange*. Server menggunakan kunci privat RSA-nya untuk mendekripsi *pre-master secret* ($r = (r^e \bmod n)^d \bmod n$). Keduanya lalu menggunakan ini untuk membangkitkan *master secret*, yang kemudian digunakan untuk menurunkan kunci *cipher*, *initialization vector*, dan MAC untuk enkripsi bit dalam jumlah besar dan autentikasi oleh *Record Layer*.

4.2 *Handshake* berbasis ECC

Mekanisme yang dilakukan pada *Handshake* berbasis ECC secara umum sama dengan *Handshake* berbasis RSA. Lewat 2 pesan pertama, klien dan server bernegosiasi mengenai *cipher suite* ECC. Pesan *Certificate* pada server berisi kunci publik ECDH server yang ditandatangani oleh otoritas sertifikasi menggunakan ECDSA. Setelah memvalidasi tandatangan ECDSA, klien mengirim kunci publik ECDH-nya ke server pada pesan *ClientKeyExchange*. Selanjutnya, baik klien dan server menggunakan kunci privat ECDH masing-masing dan kunci publik yang diterimanya tadi untuk melakukan sebuah operasi ECDH yang menghasilkan sebuah *pre-master secret*. Penurunan *master secret* dan kunci-kunci simetri, serta operasi sisanya sama seperti *Handshake* pada RSA.

4.3 Kriptografi Kunci Publik dalam SSL

Pada *Handshake* berbasis RSA, klien melakukan 2 operasi kunci publik RSA – satu untuk verifikasi sertifikat server dan satu lagi untuk mengenkripsi *premaster secret* dengan kunci publik server. Server melakukan satu operasi kunci privat RSA untuk mendekripsi pesan *ClientKeyExchange* dan mengembalikan *premaster secret*.

Pada *Handshake* berbasis ECDH-ECDSA, klien melakukan sebuah verifikasi ECDSA terhadap sertifikat server dan kemudian sebuah operasi ECDH menggunakan kunci privat ECDH miliknya dan kunci publik ECDH milik server untuk menghitung *premaster secret*. Server hanya perlu melakukan operasi ECDH untuk menghasilkan *master secret* yang sama.

Tabel 3. Operasi kriptografi pada *Handshake* SSL

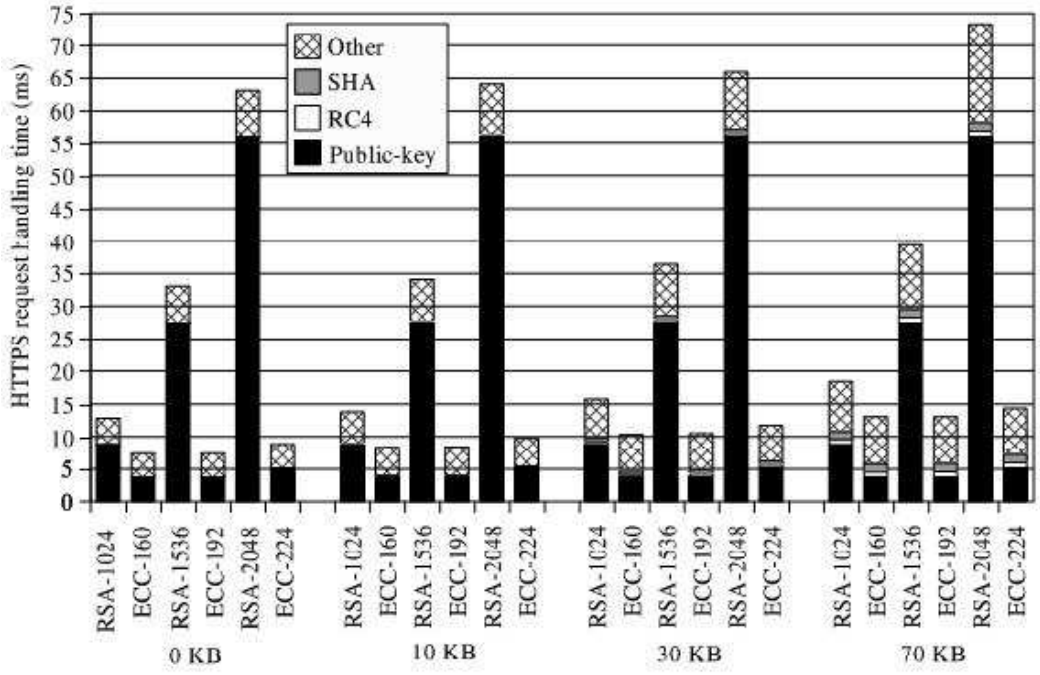
	RSA	ECC
Klien	RSA_{verify} + $RSA_{encrypt}$	$ECDSA_{verify}$ + $ECDH_{op}$
Server	$RSA_{decrypt}$	$ECDH_{op}$

4.4 Analisis Keunggulan ECC dari RSA

Keunggulan ECC dari RSA diperlihatkan dengan melakukan percobaan untuk membandingkan keduanya pada sebuah *Handshake* SSL. Dua cipher suite digunakan, *TLS_RSA_WITH_RC4_128_SHA* dan *TLS_ECDH_ECDSA_WITH_RC4_128_SHA* untuk mengindikasikan RSA dan ECC. Tiga tingkat keamanan yang ekuivalen berdasarkan Tabel 2 digunakan yaitu 1024, 1536, dan 2048 bit untuk RSA, dan 160, 192, dan 224 bit untuk ECC. Empat ukuran file sebagai data suatu transaksi web yang berbeda, 0KB, 10KB, 30KB, 70KB, akan dilewatkan diatas SSL.

Dengan spesifikasi diatas, didapatkan bahwa waktu *Handshake* dengan ECC jauh lebih cepat apabila dibandingkan dengan RSA. Untuk RSA, rata-rata waktu yang dibutuhkan dari ketiga tingkat keamanan untuk menyelesaikan proses *Handshake* adalah 30,8 ms, sedangkan rata-rata waktu untuk ECC adalah 4,23 ms. Sehingga, didapatkan rasio performansi rata-rata adalah 7,3 : 1 untuk ECC. Ukuran kunci pada 3 jenis tingkat keamanan ECC juga jauh lebih kecil dibandingkan ekuivalensinya pada RSA. Rasio antara ECC-160 dan RSA-1024 adalah 1 : 6,4; rasio ECC-192 dan RSA-1536 adalah 1 : 8; rasio ECC-224 dan RSA-2048 adalah 1 : 9,1. Terlihat bahwa ECC cukup unggul dari segi performansi dan ukuran kunci.

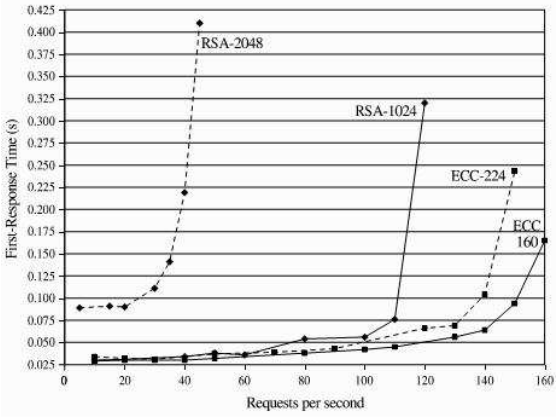
Ditinjau dari sisi server, pengukuran performansi sebuah server adalah kecepatan dalam menangani koneksi HTTPS. Gambar 3 menunjukkan waktu



Gambar 3. Perbandingan waktu penanganan *request* HTTPS pada server

rata-rata yang dibutuhkan server untuk memenuhi HTTPS *request* untuk beberapa ukuran halaman/data dan kunci publik dan *session* yang digunakan selalu baru. Dari empat ukuran data yang digunakan, waktu server untuk penanganan *request* RSA selalu lebih tinggi dibandingkan *request* yang menggunakan ECC. Lamanya waktu yang dibutuhkan ini akibat dari proses dekripsi kunci RSA yang memakan waktu sekitar 63% - 88% dari keseluruhan waktu penanganan, tergantung tingkat keamanan. Hal ini sangat kontras dengan penggunaan tingkat keamanan ECC, yang bahkan untuk ukuran data 70KB memakan waktu dibawah 50% dari keseluruhan waktu penanganan.

Dari sisi klien, efisiensi sistem dapat dilihat dari tingkat *latency* dan *throughput*. Web server yang efisien memilik kemampuan untuk menangani



Gambar 4. Perbandingan *latency* dan *throughput*

request transaksi dengan ukurandata lebih besar tetapi dengan delay yang lebih rendah. Gambar 4 menunjukkan waktu *delay* untuk respon pertama yang diterima klien (model *latency* pada komputer *user*) terhadap jumlah *request* yang ditangani server pada setiap detik (*throughput*).

Dapat terlihat bahwa penggunaan ECC membuat server dapat menangani jumlah *request* yang lebih banyak (30% - 270%) daripada RSA. Pada RSA-1024 dan ECC-160 yang dianggap memiliki tingkat keamanan setara, klien mengalami *latency* yang hampir sama. Hal ini dikarenakan sangat cepatnya waktu yang dibutuhkan untuk operasi kunci publik dibandingkan dengan waktu penghantaran dan pemrosesan data melalui SSL. Perbedaan waktu *delay* yang cukup besar baru terasa saat menggunakan RSA-2048 dan ECC-224, dimana operasi kunci publik yang dibutuhkan RSA-2048 memakan waktu yang cukup besar bila dibandingkan dengan ECC-224, sehingga berdampak pada *delay* yang cukup signifikan pada klien. Sehingga terlihat bahwa semakin banyak *request* transaksi secara konkuren dari klien-klien yang ditangani oleh server SSL, maka akan lebih baik apabila menggunakan ECC dalam pemrosesannya karena akan mengurangi waktu *delay* klien.

5. Kesimpulan

Elliptic Curve Cryptography adalah sebuah kriptografi kunci publik yang menggunakan kombinasi titik-titik pada kurva elips yang memenuhi ECDLP sebagai pasangan kuncinya.

Karena memenuhi ECDLP, maka kunci-kunci ECC akan sangat susah dipecahkan, bahkan lebih susah dari faktorisasi *integer* pada RSA dan logaritma diskrit pada Diffie-Hellman atau DSA. Selain itu ukuran kunci yang dihasilkan ECC juga akan lebih kecil dari algoritma kunci publik lainnya.

Salah satu pemanfaatan dari ECC yang bisa dilakukan adalah menerapkannya pada protokol SSL. Protokol SSL adalah protokol yang paling banyak digunakan untuk transaksi data, khususnya data pribadi dan data kartu kredit, melalui internet. SSL menawarkan keamanan komunikasi sehingga banyak digunakan oleh website *e-commerce*, pemerintah, dan organisasi lainnya. Kebanyakan SSL saat ini menerapkan algoritma RSA dalam aktivitas enkripsi dan verifikasi yang dilakukan antar server dan klien.

Dari analisis diatas, dapat disimpulkan bahwa penggunaan ECC dalam proses transasksi web melalui protokol SSL memberikan peningkatan performansi yang cukup signifikan baik pada klien maupun server, bila dibandingkan dengan sistem kriptografi kunci publik yang populer digunakan pada SSL saat ini, RSA. Hal ini dikarenakan ECC akan membangkitkan kunci dengan ukuan yang jauh lebih kecil daripada yang dihasilkan RSA untuk tingkat keamanan yang sama.

Daftar Pustaka

[1] http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Gupta.pdf
Tanggal akses: 20 Mei 2009 pukul 23.00.

[2] Munir, Rinaldi. (2006). Diktat Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[3] http://en.wikipedia.org/wiki/Elliptic_curve_cryptography
Tanggal akses: 20 Mei 2009 pukul 23.00.

[4] http://st101.blogspot.com/2007/11/ecc-cryptography.html
Tanggal akses: 21 Mei 2009 pukul 10.00