Analisis Efisiensi Algoritma RSA pada Database Kependudukan (e-KTP)

Jutono Gondohanindijo[1], Eko Sediyono[2]
[1]*Fakultas Ilmu Komputer,UNAKI, Semarang*
E-mail : jude_mei_2007@yahoo.co.

[2]*Fakultas Teknologi Informasi,UKSW, Salatiga*
E-mail : ekosed1@yahoo.com

ABSTRAK

Analisis Algoritma membantu kita memahami skalabilitas dan unjuk kerja sebuah program dengan melalui uji kompleksitasnya dengan mengukur atau menentukan problem size-nya, menentukan operasi yang dominan, menentukan fungsi yang digunakan. Algoritma yang akan dibahas diambil dari program komputer Enkripsi RSA yang diimplementasikan dalam Single Identification Number (SIN) e-KTP di Indonesia yang merupakan sebuah nomor identitas unik yang terintegrasi dengan gabungan data dari berbagai macam instansi pemerintahan dan swasta.Program Enkripsi RSA digunakan untuk menjaga keamanan dan kerahasiaan pesan, data, atau informasi dalam suatu jaringan komputer, agar tidak dapat di baca atau di mengerti oleh sembarang orang. Dalam tulisan ini, penulis menganalisis aplikasi sistem kriptografi yang layak diterapkan pada basis data terintegrasi di SIN (e-KTP). Penilaian algoritma didasarkan pada : waktu eksekusi (dari fungsi yang paling dominan), penggunaan memori/sumber daya, kesederhanaan dan kejelasan algoritma.Langkah-langkah analisis algoritma dikembangkan dengan menentukan jenis/sifat data input,mengukur Waktu Eksekusi (Running Time) dari setiap fungsi, menganalisis secara matematis dengan uji data Big-O dengan Grafik dan Tabel Polynomial untuk menentukan average case_nya.

Kata kunci : Algoritma, Enkripsi, big O, Kompleksitas, Running Time, e-KTP

1.PENDAHULUAN

Banyak algoritma enkripsi digunakan untuk keamanan data. Algoritma-algoritma tersebut sudah mapan dan menjadi standar, contohnya adalah : Data Encryption Standard (DES), Advanced Encryption Standard (AES), Rivert-Shamir-Adelman (RSA), Elliptic curve cryptography (ECC), Message-Digest algorithm 5 (MD5) [3]. Analisis Algoritma diperlukan untuk mengukur unjuk kerja program melalui uji validasi program dan waktu eksekusi (Running Time). Algoritma yang efisien ialah algoritma yang meminimumkan kebutuhan waktu dan ruang.

Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (input), yang menyatakan jumlah data yang diproses beserta waktu prosesnya.

Seringkali programer mengalami kendala untuk mengembangkan model alat bantu untuk uji efisiensi dari program yang sudah dibuat.Programer juga dituntut untuk mengoptimalkan program dari aspek struktur program maupun struktur data yang digunakan sehingga didapat program yang efisien.

Penelitian ini akan menganalisis algoritma program Enkripsi RSA terhadap waktu eksekusi (running time) fungsi dan uji kompleksitasnya untuk mengetahui kinerjanya. Data yang digunakan untuk uji efisensi algoritma enkripsi ini adalah Key enkripsi yang digunakan pada Nomor Induk Kependudukan KTP sebagai SIN (Single Identification Number).

Jenis file (berkas) yang akan dienkrip adalah bertipe text. Pentingnya melindungi SIN adalah SIN terkait dengan data kependudukan, kepegawaian, perpajakan, imigrasi, perbankan dan upaya penegakan hukum serta mempermudah kerjasama antar lembaga pemerintah dalam pelaksanaan tugasnya.

2. ANALISIS ALGORITMA

Sebuah Algoritma adalah prosedur terbatas untuk menyelasaikan sebuahpermasalahan dengan langkah yang terbatas dalam waktu yang terbatas.Istilah algoritma pertama kali ditulis oleh al-Khowarizmi matematikawanarab abad 19, dalam bukunya Kitab al-jabr w'al muquabala. Beberapa sifat umum algoritmaadalah : Input, Output, Definiteness, Finiteness, Effectiveness.

2.1 Fungsi Pertumbuhan

Diberikan sebuah program komputer dengan n bilangan bulat. Pertimbangan terpenting yang berkaitan dengan kegunaan program tersebut adalah berapa lama sebuah komputer menyelesaikan problem tersebut. Untuk menganalisa kegunaan pada program, dibutuhkan pengertian bagaimana kecepatan fungsi pertumbuhan dengan nilai n yang bertumbuh. Notasi yang sering digunakan untuk menganalisa fungsi pertumbuhan disebut notasi big-O. Besaran ini digunakan untuk menunjukkan running time dari suatu program.

Diberikanfungsi f dan g dari himpunan bilangan bilangan bulat atauhimpunan bilangan real pada suatu himpunan bilangan real. Dikatakan f(x) adalah O(g(x)) jika terdapat sebuah konstanta C dan k sedemikian sehingga [6]:

$| f(x) | \leq C | g(x) |$, dimana $x > k$. Dibaca f(x) adalah "big-oh" pada g(x) (1).

Untuk menunjukkan f(x) adalah O(g(x)), hanya dibutuhkan untuk menemukan satu pasangan konstanta C dan k sedemikian sehingga $| f(x) | \leq C | g(x) |$jika $x > k$. Pasangan C, k yang memenuhi definisi tidak pemah tunggal. Selanjutnya, jika satu pasangan ada, maka terdapat tak tarbatas pasangan yang lain. Sebuahcara sederhana untuk melihat hal tersebut adalah, jika C, k adalah satu pasangan,pasangan yang lain C', k' dengau $C < C'$ dan $k < k'$ juga memenuhi definisi, jika

$| f(x) | \leq C | g(x) | \leq C' | g(x)|$ dimana$x > k' > k$ (2).

Notasi big-O digunakan pada bidang matematika, khususnya pada ilmuKomputer untuk analisa algoritma.

Diberikan$f(x) = a_n x_n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$, dimana $a_0, a_1, ..., a_{n-1}$,
a_n adalah bilangan real. Maka f(x) adalah $O(x^n)$
Bukti :

Dengan menggunakan pertidaksamaan segitiga, jika $x > 1$, diperoleh :

$$
\begin{aligned}
&|a_n | x_n + |a_{n-1} | x^{n-1} + ... + | a_1 | x + |a_0 | \\
&f(x)| = | a_n x_n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 | \\
&\quad \leq |a_n | x_n + |a_{n-1} | x^{n-1} + ... + | a_1 | x + |a_0 | \\
&\quad = x^n (|a_n| + |a_{n-1}|/x + ... + |a_1|/x^{n-1} + | a_0 | / x^n) \\
&\quad \leq x^n (|a_n| + |a_{n-1}| + ... + |a_1| + |a_0|).
\end{aligned}
$$

Terlihat bahwa $| f(x) | \leq C x^n$
dimana $C = |a_n| + |a_{n-1}| + ... + |a_1| + |a_0|$ untuk $x > 1$. Oleh karena itu,
f(x) adalah $O(x^n)$ (3).

Jika ditemukan beberapa algoritma yang berbeda untuk menyelesaikanpermasalahan yang sama, maka harus dipilih salah satu yang sesuai denganpermintaannya. Alat yang paling utama untuk tujuan tersebut adalah AnalisisAlgoritma.

2.2 Kompleksitas Algoritma

Salah satu alat yang digunakan untuk mengetahui efisiensi sebuahalgoritma adalah waktu yang digunakan oleh komputer untuk menyelesaikansebuah permasalahan dengan menggunakan algoritma tersebut, ketika sebuahnilai input ukurannya telah dispesifikasikan [2].
Pertimbangan kompleksitaswaktu dan ruang pada sebuah algoritma adalah penting ketika sebuah algoritmaakan diimplementasikan.
Terminologi yang digunakan untuk Kompleksitas Algoritma, O(1) : Kompleksitas Konstanta,O(log n) : Kompleksitas Logaritma, O(n) = Kompleksitas Linier, O(n log n) = Kompleksitas n log n, $O(n^b)$ = Kompleksitas Polynomial [6].

3.METODE PENELITIAN

3.1 Obyek Penelitian

Obyek yang diteliti adalah Database Kependudukan dengan *field* data NIK (Nomor Induk Kependudukan), NAMA (Nama Penduduk) dan KDKELURAHAN (Kode Kelurahan Penduduk) menggunakan format text masing- masing sebesar 20 byte, 30 byte dan 15 byte, totalnya adalah sebesar 65 byte dengan panjang Kunci Enkripsi sebesar 512 bit. Data ini selanjutnya akan dienkripsi

dan dekripsi ke dalam bentuk byte/character/text atau bit. Metode Enkripsi yang digunakan adalah RSA.

3.2 Alat Penelitian

Alat penelitian yang digunakan dalam proses penelitian ini sebagai berikut:

1. Perangkat Keras berupa seperangkat computer dengan spesifikasi PentiumIntel(R) Core i3-3120M CPU@2.50GHz, 2M Bof RAM.
2. Perangkat Lunak berupa Microsoft Windows 7, ApacheWebServerV.2.2.4,PHP Script LanguageV. 5.2.3, My SQL Data Base V. 5.0.4.5, phpMyAdmin DataBaseManager V.2.10,2, GoogleChrome 2.1, EditorPHPNuSphereV.7, Javascriptlib.V.4.2, Visustin v7.03

3.3 Jalan Penelitian

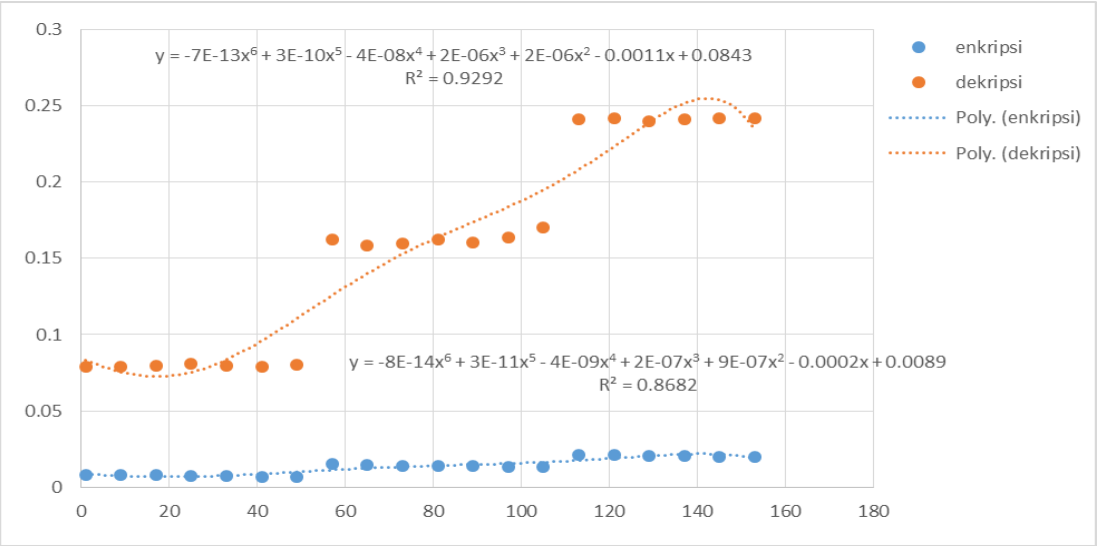
Algoritma enkripsi RSA diimplementasikan pada komputer *server* untuk proses enkripsi *query* data penduduk dan algoritma dekripsi RSA diimplementasikan pada computer *client* untuk proses dekripsi *query* data penduduk. Analisis Data pada Algoritma RSA ini menggunakan analisis regresi polynomial.

4. HASIL DAN PEMBAHASAN

Analisis Algortma dilakukan dengan mengukur kompleksitas program RSA melalui uji Big O dengan serangkaian inputan data karakter (byte) dengan jangkauan / range dari 1 sampai dengan 153 karakter (byte). Nilai ini diambil dari jangkauan total data / field dari data yang akan dienkripsi (NIK, NAMA, KDKELURAHAN).
Hasil uji dapat dilihat pada Tabel 1, tabel ini menjelaskan hasil uji waktu enkrip dan dekrip yang dibutuhkan untuk serangkaian data inputan berupa karakter. Dari dua variabel atau nilai x (panjang / banyak data) dan y (waktu) kemudian dibuatkan grafik korelasi regresi pynomialnya untuk melihat nilai korelasi (R^2), lihat Gambar 1 grafik korelasi regresi polynomial.

Tabel 1. Waktu enkrip dan dekrip dengan variasi panjang karakter

# Percobaan	# Byte	Kar. Enkrip	Byte Enkrip	Lama Enkrip (detik)	# Byte	Kar. Dekrip	Lama Dekrip (detik)
1	1	o	128	0.008265018	1	o	0.079177856
2	9	fF6h7scdC	128	0.007961035	9	fF6h7scdC	0.078781128
3	17	L9GgOpV7qJ	128	0.007781029	17	L9GgOpV7qJ	0.079634905
4	25	7Ewqx4hYe2(	128	0.007630825	25	7Ewqx4hYe2(	0.080622196
5	33	HhRvYVyyrZ	128	0.007356167	33	HhRvYVyyrZ	0.07959199
6	41	jLDu5PzUfmd	128	0.006965876	41	jLDu5PzUfmd	0.079143047
7	49	UTqRyIQVQI	128	0.00673914	49	UTqRyIQVQI	0.080024958
8	57	t0BibaQ8noh	256	0.015287161	57	t0BibaQ8noh	0.162219048
9	65	u43hPXdIngr	256	0.014369011	65	u43hPXdIngr	0.158661127
10	73	xQoWidGFbC	256	0.014062881	73	xQoWidGFbC	0.159669161
11	81	ZgUecONf1C	256	0.01422286	81	ZgUecONf1C	0.162406921
12	89	6xBPTjeQAM	256	0.013820171	89	6xBPTjeQAM	0.160511017
13	97	VmIGtzbcWA	256	0.013274193	97	VmIGtzbcWA	0.16383791
14	105	WmvELX9Qy	256	0.0132761	105	WmvELX9Qy	0.169917107
15	113	rIX8aYawFgF	384	0.021315098	113	rIX8aYawFgF	0.241089821
16	121	ufYA38n1YR'	384	0.021156073	121	ufYA38n1YR	0.241681099
17	129	CdlcEIfU5yO	384	0.02064395	129	CdlcEIfU5yO	0.23990202
18	137	bteqGOLLV1y	384	0.020835161	137	bteqGOLLV1y	0.241327047
19	145	JilaNBruaNGI	384	0.020053864	145	JilaNBruaNGI	0.241987944
20	153	eWC8WJQkr'	384	0.020042896	153	eWC8WJQkr'	0.242069006



Gambar 1. Grafik korelasi regresi polynomial

Nampak dari hasil korelasi R^2, masing-masing untuk Enkripsi sebesar 0,8682 dan Dekripsi sebesar 0,9292. Kedua nilai ini mendekati angka 1 yang menunjukkan adanya hubungan korelasi yang kuat diantara x dan y [1].
Selanjutnya dari uji size (ukuran dari banyak datanya) akan dilanjutkan dengan uji data R^2nya untuk menentukan orde dari f(n) atau big O atau O(n) dengan melakukan serangkaian uji data lanjutan dengan mensubstitusikan nilai X dengan jangkauan sejumlah data yang menjadi proses enkripsi dari sebuah field. Dari sample data diambil KDKELURAHAN sebagai nilai jangkauan tertinggi yaitu 30 karakter, lihat Tabel 2 (uji derajat X untuk big O Enkripsi) dan Tabel 3 (uji derajat X untuk big O Dekripsi) .

Tabel 2. Uji Derajat X untuk big O Enkripsi

R^2 = 0.8682		y = -8E-14 X^6 + 3E-11 X^5 - 4E-09 X^4 + 2E-07 X^3 + 9E-07 X^2 - 0.0002 X				
		X				
Derajat X	Koef. X	1	9	17	25	30
6	-8E-14	-0.0000000000000800	-0.0000000425152800	-0.0000019310055200	-0.0000195312500000	-0.0000583200000000
5	3E-11	0.0000000000300000	0.0000017714700000	0.0000425957100000	0.0002929687500000	0.0007290000000000
4	-4E-09	-0.0000000040000000	-0.0000262440000000	-0.0003340840000000	-0.0015625000000000	-0.0032400000000000
3	0.0000002	0.0000002000000000	0.0001458000000000	0.0009826000000000	0.0031250000000000	0.0054000000000000
2	0.0000009	0.0000009000000000	0.0000729000000000	0.0002601000000000	0.0005625000000000	0.0008100000000000
1	-0.0002	-0.0002000000000000	-0.0018000000000000	-0.0034000000000000	-0.0050000000000000	-0.0060000000000000

Tabel 3. Uji Derajat X untuk big O Dekripsi

$R^2 = 0.9292$		$y = -7E\text{-}13\ X^6 + 3E\text{-}10\ X^5 - 4E\text{-}08\ X^4 + 2E\text{-}06\ X^3 + 2E\text{-}06\ X^2 - 0.0011\ X$				
		X				
Derajat X	Koef. X	1	9	17	25	30
6	-7E-13	-0.0000000000007000	-0.0000003720087000	-0.0000168962983000	-0.0001708984375000	-0.0005103000000000
5	3E-10	0.0000000003000000	0.0000177147000000	0.0004259571000000	0.0029296875000000	0.0072900000000000
4	-0.00000004	-0.0000000400000000	-0.0002624400000000	-0.0033408400000000	-0.0156250000000000	-0.0324000000000000
3	0.000002	0.0000020000000000	0.0014580000000000	0.0098260000000000	0.0312500000000000	0.0540000000000000
2	0.000002	0.0000020000000000	0.0001620000000000	0.0005780000000000	0.0012500000000000	0.0018000000000000
1	-0.0011	-0.0011000000000000	-0.0099000000000000	-0.0187000000000000	-0.0275000000000000	-0.0330000000000000

Nampak dari hasil perhitungan, nilai X akan selalu muncul dengan nilai tertinggi pada kisaran derajat X = 3 baik pada uji derajat X untuk big O Enkripsi maupun Dekripsinya. Berdasarkan persamaan 3, kita menetapkan parameter X yang terbesar ini sebagai nilai yang signifikan sebagai derajat Kompleksitasnya [7]. Notasi big O pada uji Kompleksitas derajat X ini (substitusi nilai X) adalah O (n^3) dengan nama fungsi pertumbuhan Cubic.

Dari notasi big O yang ada, Kompleksitas disusun dari yang berderajat rendah sampai tinggi, mulai dari Kompleksitasdengan derajat big O =1 (konstan) atau big O(1) sampai dengan big O(2^n).Kompleksitas algoritma bergerak dari derajat big O konstan (1) sampai pada eksponensial (2^n). Sehingga kompleksitas dengan derajat paling rendah membutuhkan waktu proses yang sedikit jika dibandingkan dengan banyak datanya, sedangkan kompleksitas eksponensial adalah tingkat atau derajat yang paling tinggi, yang berarti bahwa algoritma program semakin kompleks dengan perbandingan bahwa dengan relatif data yang sedikit membutuhkan waktu proses yang banyak.

Kegiatan selanjutnya adalah mengukur running time dari program RSA. Program RSA ini menjalankan beberapa fungsi didalamnya. Hasil dari pengukuran nampak pada Tabel 4 (Running Time fungsi RSA).

Tabel 4. Running Time Fungsi RSA

Nomor	Fungsi	Running Time (detik)
1	_generateMinMax	0.000006199
2	createKey	1.655412912
3	_convertPrivateKey	0.002645969
4	_encodeLength	0.000000954
5	_convertPublicKey	0.000717878
6	_parseKey	0.000014067
7	loadKey	0.000710011
8	_random	0.001905203
9	_exponentiate	0.000002861
10	_i2osp	0.000686884
11	_rsaes_pkcs1_v1_5_encrypt	0.008409023
12	encrypt	0.008425951
13	_rsaes_pkcs1_v1_5_decrypt	0.077547073
14	decrypt	0.077579021

Dengan mengetahui lama proses semua fungsi yang dikerjakan kita dapat mengkritisi fungsi yang dominan dengan proses terlama tersebut untuk sebagai cara mengambil tindakan validasi program dengan mereformulasikan coding/programnya kembali sehingga didapat solusi algoritma yang terbaik.
Tidak ada teknik pemrograman khusus untuk memperbaiki coding /program, kita bisa melakukan pendekatan dari reformulasi struktur program maupun struktur datanya. Reformulasi dilakukan dengan serangkaian uji trial error untuk mendapatkan solusi algoritma terbaik.
Setelah itu program bisa diuji lagi berdasarkan analisis algoritmanya kembali dengan mengukur Kinerja waktu Eksekusi (big O) untuk mengukur kompleksitasnya dan identifikasi running time fungsi terbesar untuk bisa di reformulasi programnya. Proses ini berulang sampai didapatkan nilai kinerja program yang lebih baik, baik dari segi kompleksitasnya dan efisiennya (terhadap waktu dan memory yang digunakan).

5. PENUTUP

Dengan Analisis Algoritma melalui uji big O kita bisa mengukur kinerja suatu program apakah Kompleks atau tidak. Kompleks disini tidak berhubungan langsung dengan efisiensi sebuah program. Kompleksitas berhubungan dengan besaran data yang dimasukkan atau diproses. Semakin besar semakin membutuhkan waktu proses yang lama.
Dengan melihat kompleksitas program yang kita buat, kita bisa menilai performance suatu program. Selanjutanya, dengan mengukur Running Time masing-masing fungsi, kita memberikan perhatian yang lebih besar untuk mencari solusi terbaik dari program yang kita buat agar lebih efisien dalam penggunaan waktu dan memorynya melalui formulasi struktur programnya maupun struktur datanya.

DAFTAR PUSTAKA

[1] Agus Setiawan, “Pengantar Metode Numerik”, Penerbit Andi, Yogyakarta, 2006.
[2] Cormen, T. H., Leiserson, C. E., & Stein, R. L. 2001. Introduction to Algorithms, Second Edition. New York: MIT Press.
[3] Luc, Bouganim, Yanli, Guo, 2009, “DatabaseEncryption”, France.
[4] Lusmiarwan, Driana, 2006, ”Perancangan Prototype Single Identity Number(SIN) Untuk Menunjang E- Government”, Bandung.
[5] Munir, R., 2006, “Kriptografi”, Informatika, Bandung.
[6] P. Danziger, “Big-O Notation”, http://web.mit.edu/big_o.pdf, Diakses tanggal 3 September 2013.
[7] R. K. Sembiring, “Analisis Regresi”, ITB, Bandung, 2005.
[8] Suharno, 2005, “Menuju Terciptanya SingleIdentification Number di Indonesia”, Jakarta