

Pemeliharaan Keamanan Pada Sistem *Real-Time Database*

**Suharto Sisar
NIM 23203131**

**Program Magister Teknik Elektro
Bidang Khusus Teknologi Informasi
Institut Teknologi Bandung
2004**

Abstrak

Pada kebanyakan sistem real time database yang diterapkan pada Institusi Militer dan Lembaga Pemerintahan, dimana informasi yang dihasilkannya hanya dapat diakses oleh kalangan tertentu saja. Untuk menjaga keamanan informasi ini diperlukan kendali pada hak akses (*mandatory access*). Berdasarkan konsep *real time database* yang mempunyai batasan waktu, maka pada sistem *real time database* juga mempunyai batasan waktu untuk keamanannya (*security constraints*).

Model *multi-level secure database* yang konvensional tidaklah memadai untuk aplikasi yang mempunyai waktu kritis dan model *real time database* yang konvensional juga mendukung adanya batasan waktu pada keamanannya. Tujuan dari tulisan ini adalah untuk mengimplementasikan batas waktu keamanan pada *real time database* sedemikian rupa sehingga bukan saja tingkat keamanannya tercapai, tetapi juga tidak mengurangi kinerja sistem *real time*.

Pada tulisan ini ditawarkan suatu algoritma *concurrency control* baru yang diterapkan pada *firm real time database*. Hasilnya menunjukkan bahwa algoritma ini bekerja dengan baik dari segi keamanan dan segi waktu *deadline* jika dibandingkan dengan algoritma yang tidak aman. Penulis berpendapat bahwa untuk meningkatkan keamanan tidak perlu mengorbankan kinerja sistem *real time*.

1. Pendahuluan

Sistem *real time database*[3,4] seperti pada kendali komando Militer, komunikasi, sistem kendali pada pesawat terbang (*avionic*), radar dan pengaturan pabrik yang berjalan secara otomatis mempunyai batasan waktu (*deadline*) untuk menyelesaikan suatu transaksi tertentu. Pada suatu lingkungan dimana banyak *user* menggunakan *database* yang sama, hak akses untuk setiap user berbeda (ada user yang tidak diperkenankan untuk mengakses data-data tertentu), oleh karena itu dibutuhkan suatu *real time database* yang aman. Model keamanan database yang konvensional tidak dapat diterapkan pada aplikasi yang mempunyai waktu kritis[5]. Model keamanan pada *real time database* harus mempunyai kinerja yang baik dari segi batasan waktu (*timing constraints*) dan batasan keamanan (*security constraints*).

Banyak penelitian yang difokuskan pada keamanan database tanpa batasan waktu, pada akhir-akhir ini mulai dilakukan penelitian keamanan pada *real time database* [2,5,6]. Tujuan dari tulisan ini adalah untuk mempelajari faktor-faktor yang terkait dengan keamanan *concurrency control*, pengembangan algoritma yang sesuai dengan *concurrency control*, implementasi dan menguji kinerja algoritma ini dengan simulasi sistem *real time database*.

Kendali pada mandatory access telah diimplementasikan pada *multilevel secure (MLS) database*[7,8,9] dan dapat digunakan pada semua tingkatan informasi seperti pada Institusi Militer, Lembaga Pemerintah, Rumah Sakit dan lain-lain. Kendali *mandatory access* dapat diterapkan pada informasi dalam jumlah yang besar yang membutuhkan perlindungan. Pada MLS ini subjek dikelompokkan kedalam pemeriksaan (*clearances*) dan objek dikelompokkan kedalam klasifikasinya. Pada model *multi-level secure database* objek mempunyai tingkatan keamanan dan subjek mempunyai tingkatan pemeriksaan. Menurut Bell-LaPadula subjek mempunyai dua buah properti[7] yaitu :

1. *simple property*,

subjek dapat melakukan operasi membaca data-data tertentu dari objek, jika tingkat pemeriksaan subjek lebih dominan dari pada tingkat keamanan objek.

2. *star property*.

Subjek dapat melakukan operasi menulis kepada objek, jika tingkat pemeriksaan subjek lebih dominan dari pada tingkat keamanan objek

Propertis Bell-LaPadula mencegah aliran informasi dari objek dan atau subjek yang mempunyai tingkat pemeriksaan yang tinggi ke subjek dengan tingkatan yang lebih rendah, hal inilah yang menjadi dasar dari semua model MLS.

Real time database yang aman mempunyai dua tujuan yaitu; menyediakan keamanan dan memastikan persentase *output* sistem melewati *deadline* (*miss deadline percentage*/MDP) seminimal mungkin. Pada banyak kesempatan untuk mencapai salah satu tujuan, maka akan mengorbankan yang lain. Misalnya, seandainya kita mempunyai transaksi dengan prioritas yang tinggi pada tingkat keamanan yang tinggi dan transaksi dengan prioritas yang rendah pada tingkat keamanan yang rendah ada terjadi konflik data antar keduanya. Jika transaksi dengan prioritas yang tinggi mendapatkan data dan memblok transaksi dengan prioritas yang lebih rendah. Pada transaksi tingkat keamanan yang rendah dapat mengetahui ada transaksi dengan tingkat keamanan yang lebih tinggi yang sedang dikerjakan, tetapi transaksi dengan tingkat keamanan yang rendah ini masih dapat informasi selama terjadi penundaan (*delay*) pada transaksi dengan tingkat keamanan yang lebih tinggi. Bila hal ini dibalik maka tingkat keamanan transaksi tetap tetapi melanggar prioritas transaksksi yang ada. Bila terjadi konflik maka sistem yang menentukan mana yang didahulukan, prioritas atau keamanan. Jika sistem menghendaki prioritas yang didahulukan maka keamanan dapat diabaikan tetapi bila sistem memungkinkan untuk berkompromi antara prioritas dan keamanan, maka yang harus didahulukan meningkatkan keamanan sedemikian rupa tanpa meningkatkan MDP.

Pada sistem *firm real time database* transaksi akan digeser pada saat transaksksi itu melewati *deadlinenya*. Pada tulisan ini dibahas mengenai keamanan pada sistem *firm real time database*, yang dititik beratkan pada bagaimana mengurangi MDP tanpa mengurangi tingkat keamanannya. Tujuan dari tulisan ini dapat simpulkan sebagai berikut :

1. pada tulisan ini diperkenalkan algoritma *concurrency control* baru. Algoritma ini dapat digunakan pada sistem yang keamanannya dapat dikompromikan dengan prioritasnya ataupun sebaliknya,
2. pada tulisan ini diperkenalkan dua buah matrik yang digunakan untuk mengukur tingkat keamanan pada sistem keamanan *real time database*,
3. pada tulisan ini diperlihatkan, dengan menggunakan algoritma ini dimungkinkan untuk mencapai hampir 100% keamanan sistem pada kanal yang terbuka tanpa harus mengorbankan kinerja dari sistem *real time*.

2. Keamanan *concurrency control* pada Sistem Real Time

Transaksi pada sistem real time mempunyai *soft*, *firm* dan *hard deadline* [3,10]. Transaksi pada *soft deadline*, apabila melewati *deadlinenya* output dari sistem ini tetap masih ada nilainya (nilai *output* ini semakin berkurang seiring dengan dilewatinya

deadline). Transaksi dengan jenis *soft deadline* akan memproses transaksi sampai selesai walaupun *deadlinenya* telah terlewati. Sebaliknya pada *hard deadline*, apabila *deadlinenya* terlewati akan akibatnya berbahaya. *Output* dari *hard deadline* adalah negatif jika *deadline* terlewati. Pada *firm deadline* konsekuensi dari dilewatinya *deadline* lebih ringan jika dibandingkan dengan *hard deadline*, tetapi *output* dari sistem tidak dapat digunakan dan nilainya adalah nol.

2.1 Penelitian Lain yang Terkait

Tidak banyak penelitian yang dilakukan pada keamanan *real time database* seperti yang dijelaskan pada [2,5,6]. Pada [5] strategi *concurrency control* dijelaskan dengan pertukaran (*trades off*) keamanan untuk meningkatkan kinerja sistem jika sistem tidak memenuhi MDP yang diinginkan. Pada penelitian ini diperkenalkan “*capacity*” sebagai variabel yang digunakan mengatur dibukanya kanal untuk mendapatkan kinerja yang baik dari sistem *real time*.

Pada [6] algoritma *two-phase locking* digunakan untuk memungkinkan bagian pelanggaran keamanan untuk meningkatkan kinerja sistem. Keputusan diambil dengan mempertimbangkan *trade off* antara keamanan dan terpenuhinya *deadline*, keputusan ini dibuat dengan membandingkan hasil pengukuran faktor keamanan dan faktor terlewatinya *deadline*. Perbandingan ini digunakan untuk menentukan apakah transaksi dengan tingkat yang rendah dibatalkan atau diproses jika terjadi konflik dengan transaksi dengan tingkat yang lebih tinggi. Jika tingkat keamanan yang dipentingkan maka yang diproses adalah transaksi dengan prioritas yang rendah jika sebaliknya maka transaksi dengan prioritas yang tinggi didahulukan diproses.

Pada [2] penelitian ini, keamanan dilihat sebagai standar kebenaran dan jumlah dari MDP. Penelitian ini tidak melakukan *trade off* antara MDP dan keamanan, tetapi membuat suatu mekanisme yang dapat meminimalakan MDP tanpa mengurangi tingkat keamanan, dengan menerapkan strategi *concurrency control* yang sesuai. Strategi *concurrency control* terdiri dari tiga skema yaitu :

- a. *two-phase locking priority*,
- b. *prioritized optimistic concurrency control algorithm*,
- c. *dual approach*.

2.2 Faktor Keamanan Real Time

Algoritma *concurrency control* untuk keamanan *real time database* harus menggunakan tingkat keamanan dan *deadline* untuk mencegah terjadinya konflik. Jika satu

transaksi mempunyai tingkat keamanan yang tinggi dan satunya lagi mempunyai tingkat keamanan yang lebih rendah dan ada *covert channel* antara kedua transaksi tersebut. Adanya *covert channel* akan lebih banyak dari yang lain tergantung dari tingkat keamanannya. Pada suatu sistem *real time database* yang aman, dimana *timeliness* merupakan salah satu tujuan yang akan dicapai. Tujuan ini tidak dapat dikorbankan apabila *covert channel* tidak dapat melayani dengan baik. Hal inilah yang menyebabkan tingkat keamanan menjadi hal yang sangat penting untuk menentukan apakah *covert channel* ditutup untuk mengorbankan *timeliness*.

Variabel "*covert channel property*" merupakan variabel yang digunakan untuk konsekwensi dari dilanggarnya keamanan. *covert channel property* adalah perbedaan yang besar terhadap tingkat keamana antara dua transaksi yang mengalami konflik data, konsekwensi yang besar atas *covert channel* yang ada.

Covert channel property menunjukkan perbedaan yang besar terhadap tingkatan hak akses atau keamanan, hal ini lebih penting dari pada menjaga keamana dan menutup *covert channel*. Jika dua transaksi yang terjadi konflik pada tingkat keamanan yang ekstrim, maka konsekwensi pembukaan *covert channel* akan maksimum, jika terjadi tingkat keamanan yang sebaliknya maka konsekwensinya adalah minimum. Disini diperkenalkan suatu matrik yang digunakan konsekwensi dari digunakannya *covert channel* yang dikenal dengan *covert channel factor* (CCF). CCF didapatkan dengan menormalisasi perbedaan tingkatan akses antara dua konflik transaksi.

$$\begin{aligned} \text{CCF} &= \frac{\text{Difference in access levels}}{\text{Maximum difference possible}} \\ &= \frac{\text{Difference in access levels}}{\# \text{ of access levels} - 1} \end{aligned}$$

Nilai maksimum dari CCF adalah 1 ketika dua transaksi berada pada dua terminal tingkatan akses. Nilai minumum dari CCF adalah $\frac{1}{\# \text{ of access} - 1}$ ketika dua transaksi pada tingkatan keamanan yang berurutan. Nilai CCF adalah 0 bearti tidak ada *covert channel*. Pada tulisan ini diasumsikan untuk model MLS sangat sesuai dengan *single tree*.

3. Algoritma Keamanan *concurrency control*

Sistem real time database yang aman harus mempunyai batasan keamanan dan batasan waktu yang baik. Keamanan dapat dipertimbangkan sebagai kriteria kebenaran atau dapat juga dikatakan sebagai kriteria pengkompromian antara keamanan dengan

batasan waktu dimana keamanan dapat dikorbankan untuk mendapatkan MDP yang kecil. Algoritma yang diperkenalkan disini adalah sesuai dengan kriteria kedua. Sebelum dibahas mengenai algoritma yang tidak aman yaitu *Optimistic Concurrency Control* (OPT)[11]

3.1 Secure OPT

Pada algoritma tradisional *optimistic*, transaksi dimungkinkan untuk membaca dan merubah data tanpa ada batasan. Semua perubahan data ini dilakukan secara permanen selama transaksi ini valid untuk dikerjakan dan tidak terjadi konflik pada transaksi[11]. Transaksi akan diulangi jika hasil tes validasi gagal. Tujuan utama dari algoritma real time *optimistic* adalah mencegah transaksi dengan prioritas dikerjakan jika terjadi konflik dengan transaksi dengan prioritas yang lebih tinggi terjadi.

Perbedaan strategi *optimistic* pada transaksi *real time* adalah *priority sacrifice* dan *priority wait*. Pada *priority sacrifice* atau OPT-SACRIFICE, strategi ini melakukan ulang (*restarts*) validasi transaksi jika transaksi dengan prioritas tinggi benar-benar ada pada kumpulan transaksi yang mengalami konflik. Pada *priority wait* atau OPT-WAIT, strategi ini akan memvalidasi *transaction waits* jika ada transaksi dengan prioritas tinggi ada pada kumpulan transaksi yang mengalami konflik. Algoritma ini tidak mengecek tingkat keamanan pada transaksi dan akan dapat diberikan *covert channels* ketika validasi transaksi dilakukan dan membatalkan transaksi pada kumpulan transaksi yang mengalami konflik.

Algoritma *priority sacrifice* dapat dicontohkan sebagai berikut, pada validasi transaksi dalam tingkatan hak akses dan prioritas lebih tinggi dari transaksi yang lain maka transaksi ini diperkenankan untuk menggunakan *covert channel*, ketika hasil dari validasi transaksi adalah untuk membatalkan seluruh transaksi yang mengalami konflik, maka digunakan algoritma *priority wait* dimana jika pada kumpulan transaksi itu terdapat transaksi dengan prioritas yang tinggi maka ia diperkenankan menggunakan *covert channel*. Pada tulisan ini membahas berbagai jenis konflik yang akan terjadi antara *validating transaction*(VT) dengan *conflicting set*(CS), dan hasilnya dapat dilihat dibawah ini :

- a. $\forall T \in CS \text{ if } deadline(T) > deadline(VT) \text{ and } \forall T \in CS \text{ if } access\ level(T) > access\ level(VT) \text{ then kerjakan VT dan batalkan semua } T \text{ pada CS. Karena VT mempunyai}$

prioritas yang paling tinggi dan tingkat keamanan yang paling rendah, maka prioritas dan keamanannya tetap dijaga agar tidak berubah.

- b. $\forall T \in CS$ if $deadline(T) < deadline(VT)$ and $\forall T \in CS$ if $access\ level(T) < access\ level(VT)$ then blok VT. VT adalah mempunyai prioritas yang paling rendah dan tingkat keamanan yang paling tinggi. Jika mempertahankan prioritas maka transaksi ini tidak dapat menggunakan *covert channel*.
- c. Jika dua kondisi diatas tidak ditemui, maka *covert channel* akan dapat digunakan tanpa memperhatikan pihak mana yang membatalkan atau memblok kecuali nilai dari CS dibuat menjadi satu. Pada kasus ini, karena menggunakan strategi *optimistic* dimana sistem tidak dapat membuat *covert channel* 100% bebas, jika beberapa hasil tahap validasi gagal dan ada lebih dari satu transaksi yang berada pada CS. Permasalahan ini dapat dipecahkan dengan mengurangi CCF dan prioritas dapat diidentifikasi setiap saat.
- 1) Meminimalkan CCF. Disini CCF akan dihitung ketika validasi transaksi dibatalkan dan CCF akan dihitung ketika CS dibatalkan. Kedua cara penghitungan CCF ini dibandingkan untuk menentukan mana yang dapat menggunakan *covert channel*.
 - 2) Mempertimbangkan Prioritas (*Consider the priority*). Disini akan dipertimbangkan prioritas dari transaksi pada CS untuk tidak mengurangi kinerja sistem *real time*, untuk melakukan hal ini diperlukan pengecekan sederhana pada CS apabila ada ada transaksi dengan prioritas tinggi maka akan dilakukan VT.

Berdasarkan kedua kasus diatas, konflik dapat dipecahkan dengan beberapa cara. pada tulisan ini akan meminimalkan CCF dan mengidentifikasi prioritas. Pada VT dimungkinkan untuk melakukan.

If (CCF for aborting CS $<$ CCF for aborting VT)

OR

(no high priority transaction exists in CS)

Then

Commit the VT and abort the CS.

Else

Abort the VT.

Berikut merupakan contoh dari penggunaan algoritma ini. Diasumsikan transaksi $\{T1=(4, 10.40), T2=(5, 11.57), T3=(2, 12.67)\}$ merupakan CS dimana setiap transaksi ditampilkan sebagai pasangan (*access level* dan *deadline*) dan juga diasumsikan $VT=(3, 10.54)$. Jika total angka dari *access level* adalah 6 maka untuk membatalkan CCF pada CS adalah $CS=(3-2)/5= 1/5$ karena pada T3 terjadi konflik keamanan, pembatalan pada T1 dan T2 tidak akan mengakibatkan *covert channel* pada tingkat

keamanan yang lebih tinggi. CCF akan membatalkan VT jika $VT=(4-3)/5+(5-3)=1/5+2/5=3/5$. Pada *if* yang pertama kondisinya sudah memuaskan dan dilakukan VT. Pilihan ini mengidentifikasi baik keamanan maupun prioritas dan jika nilai CCF minimal (kasus 1) atau memperetimbangkan prioritas (kasus 2), VT dimungkinkan untuk dikerjakan. Tujuan dari pilihan ini adalah untuk sebanyak mungkin melakukan VT dan meningkatkan kinerja dari sistem *real time*. Jika pilihan yang dilakukan adalah pada tingkat keamanan maka *if* yang pertam sudah cukup, keadaan ini disebut *secure option*.

If (CCF for aborting CS < CCF for aborting VT)
(no high priority transaction exists in CS)
Else
Abort the VT.

3.2 Pemeliharaan Matrik Keamanan

Pada bagian ini diperkenalkan dua matrik atau faktor keamanan (*security factor*) untuk mengukur pemeliharaan keamanan pada sistem. Satu matrik adalah tetap mencatat nilai dari keamanan yang dipelihara dan matrik yang lain untuk mengidentifikasi perbedaan antara tingkatan hak akses.

$$security\ factor\ 1 = \frac{\#of\ times\ security\ is\ ma\ int\ ained}{Total\ number\ of\ security\ conflicts}$$

$$security\ factor\ 2 = \frac{Sum\ of\ the\ diffrence\ in\ access\ levels\ for\ conflicts\ having\ security\ ma\ int\ ained}{Sum\ of\ the\ diffrence\ in\ access\ levels\ in\ all\ security\ conflicts}$$

kedua matrik diatas dapat digunakan untuk mengukur kinerja sistem. Penggunaan faktor tergantung pada sistemnya, jika yang diperhatikan adalah jumlah konflik maka faktor yang pertama yang digunakan. Jika perbedaan tingkatan hak akses yang diperhatikan maka faktor yang kedua. Pada tulisan ini yang digunakan adalah faktor yang kedua.

3.3 Pemeliharaan Matrik Prioritas

Dalam rangka untuk menampilkan tingkatan prioritas pada suatu sistem digunakan faktor berikut.

$$Priority\ maintenance\ factor = \frac{\#of\ times\ priority\ is\ ma\ int\ ained}{Total\ number\ of\ data\ conflicts}$$

Matrik ini digunakan untuk menentukan bagaimana dampak dari menjaga keamanan dari sistem *real time*.

4. Model Simulasi

Pada bagian ini akan dibahas mengenai struktur dan detail dari model simulasi yang digunakan untuk mengevaluasi kinerja algoritma *concurrency control* untuk sistem *real time database*. Pusat dari model ini adalah *single-site main memory database system* yang beroperasi pada satu prosesor. *Database* dimodelkan sebagai kumpulan *data pages* pada memori. Model simulasi ini terdiri dari tiga komponen utama yaitu :

1. *Transaction Manager*(TM),

TM bertanggung jawab menyediakan permintaan *lock*.

2. *CPU Manager*(CM),

CM digunakan untuk mengizinkan hak akses pada CPU.

3. *Log Manager*(LM).

LM digunakan untuk *log* pada akses *database*.

Service discipline digunakan pada antrian (*queues*) adalah Earliest Deadline First(EDF)[11] tanpa preemption. Tiap transaksi terdiri dari banyak operasi, dimana setiap operasinya dapat melakukan operasi baca atau tulis. Jika operasi yang dilakukan adalah membaca maka *page* tidak diupdate. Jika operasinya adalah menulis maka akan dilakukan *update* pada *page* dan penulisan ini akan dimasukkan kedalam *log buffer*.

Ketika sebuah operasi dari sebuah transaksi dilakukan maka operasi akan meminta hak akses data yaitu *lock request* pada objek data yang ada pada TM. Permintaan ini akan diteruskan pada *concurrency control* untuk menghindari *lock* pada data objek. Jika permintaan ini disetujui transaksi akan meminta hak pada CPU untuk mengakses CM. Jika CPU tidak mengerjakan apa-apa maka permintaan ini disetujui dan transaksi diproses oleh CPU. Setelah CPU melakukan proses, jika masih ada operasi lain maka operasi akan dikerjakan dan akan melakukan *lock request* pada TM. Jika tidak ada transaksi lain maka transaksi akan meminta akses pada log disk pada LM dan jika hak ini diizinkan maka transaksi ini akan menulis *log buffer* ke *log disk*.

Jika permintaan untuk *lock* tidak dikabulkan maka transaksi ini akan diletakkan kembali ke *block queue*.

4.1 Parameter pada Model Simulasi

Tabel 1 merupakan resource parameters yang ada pada model simulasi pada tulisan ini. Parameter CPU_TIME adalah waktu yang dibutuhkan untuk memproses sebuah *page* oleh CPU. Pada simulasi ini tidak secara eksplisit menghitung waktu yang dibutuhkan untuk mengakses TM, CM dan LM. Hal ini diasumsikan waktu itu termasuk dalam waktu yang dibutuhkan untuk mengakses sumber daya seperti CPU dan *log disk*.

Tabel 1 : Resource Parameters System

Parameter	Explanation	Value
DBSIZE	Number of data pages in the database	400
CPU_TIME	CPU time for processing a data page	5 msec
WRITE_PROB	Probability that an operation is write	0.5
MAXACCESS	# access levels	6

Tabel 2 : Workload Parameters

Parameters	Meaning	Value
Rate	Arrival rate of the transactions	[5,50]
TransSize	Average transaction size	6
LogDelay	Overhead for writing log buffer to the log disk	5 msec
RestartDelay	Overhead for restarting	5 msec
MinSlack	Minimum slack factor	2
MaxSlack	Maximum slack factor	8

Tabel 2 berisikan *workload parameter* yang merupakan karakteristik dari transaksi dan *system workload*. Transaksi *inter-arrival rates* disistribusikan secara eksponensial. *Rate* parameter adalah nilai rata-rata dari kedatangan transaksi. *TransSize* menentukan nilai tengah dari suatu operasi yang didapat dari transaksi yang berupa distribusi normal. *Actual data object* atau *pages accessed* pada setiap operasi dalam bentuk distribusi yang sama pada setiap database. *LogDelay* adalah waktu yang dibutuhkan untuk menulis *log buffer* pada *log disk*. *RestartDelay* adalah *overhead* dari *rol back* ketika

transaksi digagalkan. Parameter *MinSlack* dan *MaxSlack* digunakan untuk batas atas dan bawah *deadline* suatu transaksi. Berikut ini adalah rumus yang digunakan untuk menentukan *deadline*.

$$Deadline = Arrival\ time + Uniform(MinSlack, MaxSlack) * Execution\ time$$

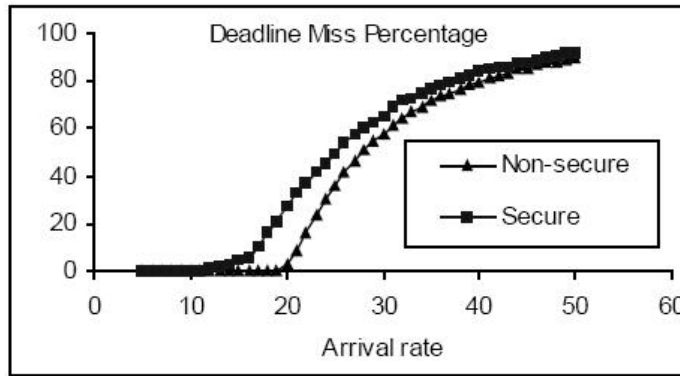
Arrival time adalah waktu kedatangan setiap transaksi. *Execution time* dihitung berdasarkan seluruh data yang diperlukan pada setiap operasi dengan menggunakan *TranSize*, *CPU_TIME* dan *LogDelay*.

4.2 Percobaan dan Hasil

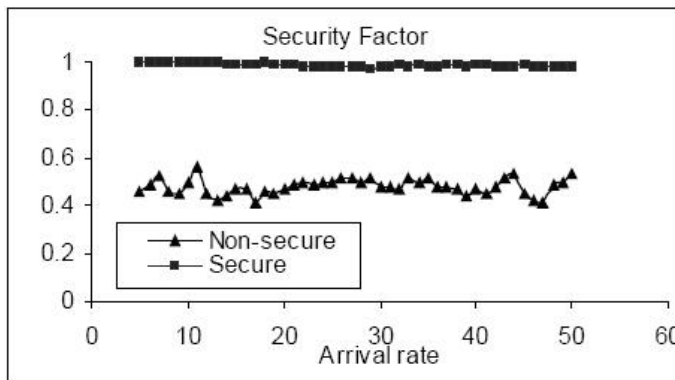
Program simulasi dibuat dengan menggunakan Bahasa C++ dengan menggunakan strategi simulasi berikut dan menjalankan 5000 transaksi. Different random seeds digunakan pada panggilan yang berbeda untuk membuat bilangan acak dan untuk memastikan transaksi yang datang mempunyai algoritma yang berbeda satu dengan yang lain. Dalam rangka simulasi pada firm real time system, pada awal setiap kejadian sistem mengecek untuk melihat apakah ada transaksi yang melewati *deadlinenya* dan jika ada maka transaksi itu dihilangkan dari sistem. Disini dipelajari secara terperinci simulasi pada firm database system dengan menggunakan algoritma yang dibuat dan dibandingkan dengan algoritma yang tidak aman. Disini juga dibahas tentang keefektifan algoritma yang diajukan dalam rangka memelihara keamanan dan meminimalkan MDP dan juga dipelajari *restart ratios* untuk kedua algoritma.

Gambar 1 memperlihatkan MDP pada algoritma *non-secure* OPT dan *secure* OPT dengan keamanan sebagai pilihannya. Algoritma *non-secure* yang digunakan adalah OPT-SACRIFACE. OPT-SACRIFACE adalah algoritma yang sangat memperhatikan prioritas dan oleh karena itu mempunyai kinerja lebih baik dari pada *secure* OPT. *Non-secure* OPT mempunyai nilai MDP rata-rata dibawah 20 sedangkan pada *secure* OPT transaksi mulai melewati *deadlinenya* pada waktu kedatangan 15 tetapi mempunyai MDP yang kecil jika waktu rata-rata kedatangannya kecil dari 20. Perbedaan utama antara kedua algoritma ini adalah antara waktu kedatangan 15 dan 25, setelah transaksi ini maka kedua algoritma akan melewati *deadlinenya*.

Algoritma *secure* OPT mengidentifikasi adanya *covert channels* oleh karena itu faktor keamanan pada sistem bertambah jika algoritma ini digunakan. Hasilnya dapat dilihat pada Gambar 2, dimana pada gambar ini memperlihatkan perbandingan faktor keamanan antara algoritma yang aman dan yang tidak aman.



Gambar 1 : Perbandingan *Miss Deadline Percentages*(MDP)

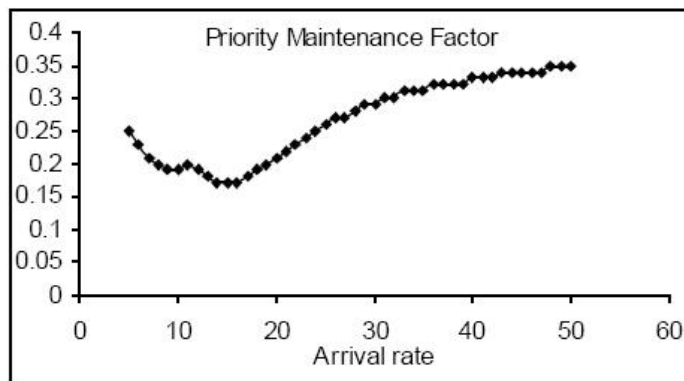


Gambar 2 : Perbandingan Faktor Keamanan

Pada algoritma yang aman, faktor keamanan mendekati angka 1, sebaliknya dengan algoritma yang tidak aman faktor keamanannya berubah-ubah disekitar angka 0,5. Menarik untuk diperhatikan pada Gambar 1, walaupun keamanannya telah ditingkatkan dengan menggunakan secure OPT dan tidak mengorbankan kinerja dari sistem real time. Hal ini disebabkan banyaknya konflik data untuk memaksa tingkat keamanan tidak melanggar tingkat prioritas yang didasarkan pada *deadline*. Meningkatkan keamanan berarti banyak konflik data diselesaikan pada transaksi dengan tingkat keamanan yang rendah dengan mengabaikan *deadlinenya*. Jika transaksi dengan tingkat keamanan yang rendah mempunyai *deadline* yang lebih lama, apabila dipaksakan dari sisi keamanannya maka transaksi ini akan kehilangan prioritasnya. Bagaimanapun jika hal ini terjadi pada transaksi dengan prioritas yang rendah maka tingkat keamanannya dapat dijaga. Oleh karena itu untuk mencapai tingkat keamanan yang diinginkan tidak berarti dengan kelihalangan semua prioritas.

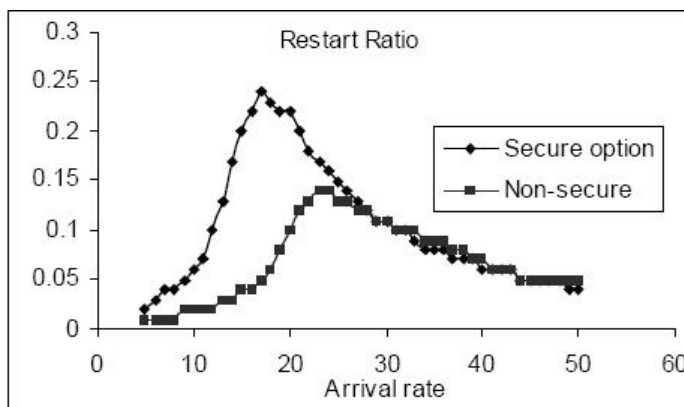
Gambar 3 memperlihatkan *priority maintenance factor* ketika faktor keamanan sangat dekat ke angka 1,0. Pada kasus ini *priority maintenance factor* nilainya tidak nol tapi bervariasi antara 0,15 sampai dengan 0,35. Hal inilah yang menyebabkan kinerja

sistem *real time* dengan *secure* OPT pada Gambar 1 sama dengan non-secure OPT. Hal ini merupakan ciri-ciri (*feature*) sangat penting pada algoritma *secure* OPT.



Gambar 3 : *Priority Maintenance Factor* pada *Secure* OPT

Gambar 4 memperlihatkan perbandingan *restart ratios* antara algoritma non-secure dan *secure* OPT. Pada setiap algoritma *restart ratios* naik mengikuti naiknya arrival rate dan akan turun pada jarak tertentu dari arrival rate (15-16 untuk *secure* OPT dan 21-22 untuk non-secure OPT). Hal ini dapat dijelaskan sebagai berikut. Kebanyakan transaksi mencapai deadlinenya setelah *arrival rate* melewati 15-16 untuk *secure* OPT dan 21-22 untuk non-secure OPT. Jika *arrival ratenya* rendah maka akan terjadi sedikit konflik dan konsekuensi restart. Bagaimanapun pada saat transaksi mulai melewati *deadlinenya*, beberapa transaksi yang ditunda telah melewati *deadlinenya* dan tidak dapat direstart karena transaksi ini telah dipindahkan dari sistem sesat transaksi ini melewati *deadlinenya*. Dengan tingginya *arrival time*, jumlah transaksi yang dibatalkan(*aborted*) akan meningkat, yang akan menyebabkan turunnya jumlah *restarts*. Dengan demikian meningkatnya *restart ratios* seiring dengan meningkatnya *arrival rate* samapai dengan sistem mulai melewati *deadlinenya*, setelah ini *restart ratio* akan turun dan akan terus turun seiring dengan bertambahnya *arrival rate*.



Gambar 4 : Perbandingan *Restart Ratios*

5. Kesimpulan

Pada tulisan ini diidentifikasi *covert channel* pada *secure real time database* dan pada tulisan ini diusulkan algoritma *secure optimistic concurrency* baru. Pada tulisan ini *secure* dan *non-secure* algorithm telah diimplementasikan dan dipelajari kinerjanya secara detail dengan menggunakan model simulasi sistem *real time database*. Pembahasan ini mencakup *firm real time database*. Pada tulisan ini juga diperkenalkan matrik yang digunakan untuk mengukur keamanan pada sistem *real time database*. Hasilnya sangat jelas memperlihatkan algoritma ini berkinerja sangat baik dalam *trade off* antara keamanan dan *timeline* yang dibandingkan dengan algoritma *non-secure*.

Pada tulisan ini juga diperlihatkan bahwa untuk mencapai tingkat keamanan yang diinginkan tidak berarti dengan mengorbankan kinerja dari sistem *real time*. Sistem dapat dibuat hampir 100% bebas dari *covert channel* tetapi hanya dapat dilakukan jika MDP diatas 20.

Referensi

- [1] Abbott, R. and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation" *ACM Transactions on Database Systems* 17, 1992, pp. 513-560.
- [2] George, B. and J. Haritsa, "Secure Transaction Processing in Firm Real-Time Database Systems," *Proceedings SIGMOD*, Phoenix, AZ, 1997, pp. 462-473.
- [3] Ozsoyoglu, Gultekin, and Richard T. Snodgrass, "Temporal and Real-Time Databases: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7 no. 4, Aug. 1995, pp. 513-532.
- [4] Ramamritham, Krithi, "Real-Time Databases," *International Journal of Distributed and Parallel Databases*, 1993, pp. 199-226.
- [5] Rasikan, David, Sang H. Son and Ravi Mukkamala, "Supporting Security requirements in Multilevel Real-Time Databases," *Proceedings IEEE Symposium on Security and Privacy*, Oakland, CA, May 1995, pp. 199-210.
- [6] Son, Sang H., Rasikan David and Bhavani Thuraisingham, "An Adaptive Policy for Improved Timeliness in Secure Database Systems," *Proceedings of Annual IFIP WG 11.3 Conference of Database Security*, Aug. 1995.
- [7] Bell D. E., and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model," *Technical Report*, MITRE Corporation, 1974
- [8] Denning, Dorothy E., "The Sea View Security Model," *Proceedings IEEE Symposium on Security and Privacy*, Oakland, Ca, April, 1988.

- [9] Jajodia, Sushil and R. Sandhu, “*Toward a Multilevel Secure Relational Data Model*,” *Proceedings ACM SIGMOD*, Denver, Colorado, May, ACM, New York, 1991, pp. 50-59.
- [10] Wolfe, V. F. and L. C. DePippo. “*Real-Time Database Systems*” in *Database Systems Handbook*, (Paul J. Fortier, ed.), McGraw Hill Publishers, 1997.
- [11] Haritsa, J. R., M. J. Carey and M. Livny, “*Data Access Scheduling in Firm Real-Time Database Systems*”, *The Journal of Real-Time Systems*, 4, 203-241 (1992).
- [12] Liu, C. L. and J. W. Layland, “*Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment*,” *Journal of the ACM*, vol. 20, no. 1, Jan 1979, pp. 46-61.
- [13] Moskowitz, I. S. and Allen R. Miller, “*Simple Timing Channels*”, *Proceedings of the IEEE Symposium on Security*, 1994, pp.56-64.
- [14] Qian, Xialoei, “*Inference Channel-Free Integrity Constraints in Multilevel Relational Databases*”, *Proceedings of the IEEE Symposium on Security*, 1994, pp. 158-167.