APLIKASI OTOMATISASI MAINTENANCE PERANGKAT LUNAK DENGAN FUNGSI HEURISTIC INTEGRITY CHECKERS DAN LOGIKA FUZZY C-MEANS

Muhammad Rofiq[1] , Agam Dhany Saputro[2]
[1] Jurusan Teknik Informatika, DosenSTMIK ASIA Malang
[2] Jurusan Teknik Informatika, STMIK ASIA Malang
e-mail: muhammad.rofiq.09@gmail.com[1]

ABSTRAK

Masalah dalam kerusakan file perangkat lunak di dalam laboratorium STMIK ASIA Malang dideteksi dengan pendeteksian integritas data dan ditambahkan dengan fungsi heuristic untuk menambah akurasi pencarian file, penanganan kerusakan ditentukan dalam kelas-kelas solusi keadaan. Dengan menggunakan pendeteksian checksum error pada metode CRC32 ditentukan kerusakan pada file perangkat lunak. Dalam penanganan kerusakan file perangkat lunak digunakan Fuzzy C-Means yaitu algoritma pengklusteran untuk menentukan kondisi kerusakan terhadap perangkat lunak dengan membagi kerusakan dalam beberapa cluster penanganan kerusakan. Pengujian menunjukkan hasil prosentase keakurasian sebesar 100% dalam menangani kerusakan file yang dilakukan pada komputer A11 dan juga pengujian terhadap Lab A, B, C dan D.

Kata kunci: Kerusakan perangkat lunak, Checksum Error, Cyclic Redundancy Check 32bit, Fuzzy Clustering, Fuzzy C-Means, Fungsi Heuristic.

ABSTRACT

Problems in file damage software in a laboratory STMIK ASIA Malang detected with Poor data integrity detection and heuristic functions added to increase the accuracy of searches files, handling damage classes specified in the solution of the situation. By using a checksum error detection on the specified method on the file CRC32 checksum error. In the handling of file damage software used Fuzzy C-Means clustering algorithm to determine the condition of damage of the software by dividing damage into several clusters of handling damage. Testing shows the results of 100% percent accuracy in dealing with the damage done on a computer A11 and also testing of Lab A, B, C and D.

Keywords: Error Of Software, Checksum Error, Cyclic Redundancy Check 32bit, Fuzzy Clustering, Fuzzy C-Means, Heuristic function

PENDAHULUAN

Kegiatan belajar mengajar di STMIK ASIA Malang tidak lepas dengan penggunaan laboratorium komputer. Dimana laboratorium komputer dituntut untuk selalu dalam keadaan baik dan siap untuk digunakan praktikum. Tidak terlepas dengan kondisi laboratorium yang harus baik maka dilakukan maintenance komputer secara berkala. Pelaksanaan kegiatan maintenance komputer ada beberapa tahapan diantaranya adalah maintenance perangkat lunak. Maintenance perangkat lunak dilakukan dengan cara dibuatkan daftar program yang akan di maintenance, dilakukan checking program satu-persatu setiap komputer dan ketika terjadi kerusakan maka akan dilakukan penanganan masalah secara manual. Kegiatan maintenance perangkat lunak yang dilakukan ada 3 tahapan dan dilakukan secara manual ini memerlukan waktu dan tenaga yang tidak sedikit sehingga hal ini mempengaruhi kegiatan maintenance perangkat lunak terhadap kegiatan belajar mengajar.

Perangkat lunak dalam sebuah komputer memiliki bagian penyusun file dan pada umumnya memiliki lebih dari satu file yang ber-ekstensi EXE maupun DLL, dimana file EXE merupakan file eksekusi yang berguna untuk menjalankan perangkat lunak dan file DLL adalah file library windows yang merupakan kode yang sudah dikompilasi dan dapat digunakan oleh lebih dari satu file EXE secara bersamaan. Seperti halnya file-file perangkat lunak yang tidak terlepas dari bahaya kerusakan. Beberapa kerusakan yang sering terjadi pada file-file perangkat lunak adalah kerusakan yang diakibatkan oleh serangan virus komputer maupun pemadaman komputer secara paksa baik disengaja maupun yang tidak. Hal ini mengakibatkan file-file perangkat lunak

mengalami perubahan struktur cluster-cluster dalam sistem berkas. Salah satu metode yang menggunakan fungsi hash dalam membaca sebuah struktur dalam sebuah file dalam transmisi atau penyimpanan sebuah data adalah CRC32 (cyclic redundancy check 32 bit). CRC32 dapat digunakan untuk mendeteksi error (kerusakan) pada sebuah data dalam file yang mungkin terjadi pada saat transmisi data atau pengiriman data. Metode ini menghitung nilai checksum dari panjang bit sebuah data yang kemudian membandingkannya dengan aturan CRC dengan menggunakan kunci 32 bit untuk mendeteksi apakah data tersebut mengalami perubahan (kerusakan) atau tidak. Pemanfaatan metode ini akan ditambahkan fungsi heuristic dimana proses pendeteksian akan dilakukan kepada file-file mengalami perubahan secara langsung, sehingga kerusakan dapat ditangani lebih dini dan tidak menyebar ke file-file yang lain. Teknik ini lebih dikenal dengan teknik heuristic integrity checkers.

Setelah melakukan kegiatan pendeteksian file untuk menentukan kerusakan pada file-file perangkat lunak maka dilakukan sebuah metode yang dapat menggolongkan keadaan file dimana segala kemungkinan yang terjadi dalam pendeteksian di klasifikasikan menurut penanganan yang akan dilakukan, metode ini dikenal dengan metode fuzzy c-means.

Berdasarkan uraian diatas, dibuatlah sebuah aplikasi dengan menggunakan teknik heuristic integrity checkers untuk melakukan deteksi kerusakan file-file dan dilakukan penanganan masalah menggunakan Fuzzy C-means pada aplikasi yang terinstal di komputer.

Berdasarkan pada permasalahan tersebut diatas, maka rumusan masalah dalam penelitian ini adalah bagaimana membuat aplikasi komputer yang dapat mendeteksi dan menangani kerusakan terhadap perubahan pada *file-file* perangkat lunak dengan menggunakan teknik *heuristic integrity checkers* dan logika *fuzzy c-means*.

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Format perangkat lunak yang digunakan untuk pencarian checksum error berupa file ekstensi EXE dan DLL.
2. Penelitian dilakukan pada sistem operasi Windows Xp SP3.
3. Pencarian file-file error terhadap perangkat lunak hanya pada drive C:\Program Files\ (Default System Drive).
4. Penelitian ini dilakukan pada laboratorium komputer di STMIK ASIA Malang.
5. Aplikasi dibatasi hanya untuk mendeteksi kerusakan pada file, dan menangani kerusakan file yang terjadi perubahan dengan file yang tidak mengalami kerusakan.

Tujuan dari penelitian ini adalah untuk merancang dan membuat aplikasi otomatisasi maintenance perangkat lunak dengan fungsi *heuristic integrity checkers* dan logika *fuzzy* c-means.

Manfaat dari penelitian yang akan dilakukan adalah:

1. Membantu menyelesaikan masalah perangkat lunak yang ada pada laboratorium komputer STMIK ASIA Malang.
2. Hasil penggunaan metode dalam aplikasi dapat digunakan sebagai acuan untuk pengembangan penyelesaian masalah pada laboratorium komputer.
3. Mempercepat proses maintenance perangkat lunak pada laboratorium komputer.

KAJIAN TEORI

Perangkat Lunak

Pengertian perangkat lunak komputer adalah instruksi-instruksi (program komputer) yang jika dieksekusi/ dijalankan akan menghasilkan fungsi dan unjuk kerja yang diinginkan, struktur data yang memungkinkan program dapat memanipulasi informasi, dokumen-dokumen yang menjelaskan dan menggunakan dari program (Pressman, 2001).

Menurut Roger S (2002) mendefinisikan perangkat lunak sebagai Perintah program komputer yang bila di eksekusi memberikan fungsi dan unjuk kerja seperti yang di inginkan. Sedangkan menurut Melwin (2007) mendefinisikan perangkat lunak Berfungsi sebagai pengatur aktivitas kerja komputer dan semua intruksi yang mengarah pada sistem komputer. Perangkat lunak menjembatani interaksi user dengan computer yang hanya memahami bahasa mesin.

Sistem *File* atau berkas

Menurut Muflih (2008) sistem *file* (berkas) dan akses menyangkut sistem pengorganisasian, pengelolaan dan penyimpanan data pada alat penyimpanan eksternal dengan organisasi *file* tertentu.

Sedangkan berkas menurut Noor (2005) adalah sebuah unit tempat menyimpan informasi. Berkas ini dapat diakses lebih dari satu proses, dapat dibaca, dan bahkan menulis yang baru. Informasi yang disimpan dalam berkas harus persisten, dalam artian tidak hilang sewaktu proses berhenti. Berkas-berkas ini diatur oleh sistem operasi, bagaimana strukturnya, namanya, aksesnya, penggunaannya,

perlindungannya, dan implementasinya. Bagian dari sistem operasi yang mengatur masalah-masalah ini disebut sistem berkas.

File atau juga berkas memiliki nama yang unik dalam direktori dimana *file* tersebut berada. Alamat direktori dimana suatu berkas ditempatkan diistilahkan dengan *pathfile*. Nama berkas (*Filename*) akan memberikan sebuah nama terhadap sebuah berkas agar dapat dikelola dengan mudah. Berkas adalah sebuah koleksi informasi berkaitan yang diberi nama dan disimpan di dalam *secondary storage*. Biasanya sebuah berkas merepresentasikan data atau program. Adapun jenis-jenis dari berkas:

1. *Text file*: yaitu urutan dari karakter-karakter yang diatur menjadi barisan dan mungkin halaman. (Contoh: txt, rtf, doc, ppt, odt, pdf)
2. *Source file*: yaitu urutan dari berbagai *subroutine* dan fungsi yang masing-masing kemudian diatur sebagai deklarasi-deklarasi diikuti oleh pernyataan-pernyataan yang dapat di-*execute*. (Contoh: c, cpp, java, pas, asm)
3. *Object file*: yaitu urutan dari byte-byte yang diatur menjadi blok-blok yang dapat dipahami oleh penghubung sistem. (Contoh: Obj, o)
4. *Executable file*: adalah kumpulan dari bagian-bagian kode yang dapat dibawa ke memori dan di eksekusi. (Contoh: exe, com, bin)
5. *Library* file: adalah file yang mengandung *kode-fungsi* yang dapat dipanggil dari *kode-executable* (.exe) lain (baik aplikasi atau DLL lain). Sebuah Dynamic Link Library (.DLL) adalah library yang berisi kode dan data (kumpulan program kecil) yang dapat digunakan oleh lebih dari satu program pada waktu yang bersamaan.

Basis Data

Menurut Connoly dan Begg (2002), basis data (database) adalah suatu kumpulan data secara logikal saling terkait yang dirancang untuk mendapatkan informasi yang dibutuhkan oleh suatu organisasi.

Menurut Fathansyah (2004) dalam bukunya yang berjudul Basis Data, menjelaskan bahwa basis data adalah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.

Sedangkan menurut Hariyanto (2004) dalam bukunya yang berjudul Sistem Manajemen Basis data Permodelan, Perancangan, dan Terapannya menjelaskan bahwa basis data adalah kumpulan data (elementer) yang secara logika berkaitan dalam mempresentasikan fenomena/fakta secara terstuktur dalam domain tertentu untuk mendukung aplikasi pada sistem tertentu. Basis data adalah kumpulan data yang saling berhubungan yang merefleksikan fakta-fakta yang terdapat diorganisasi.

Integrity Checking

Salah satu cara untuk menguji integritas sebuah data adalah dengan memberikan *checksum* atau tanda bahwa data tersebut tidak berubah. Cara yang paling mudah dilakukan adalah dengan menjumlahkan karakter-karakter atau data-data yang ada sehingga apabila terjadi perubahan, hasil penjumlahan menjadi berbeda. Cara ini tentunya mudah dipecahkan dengan menggunakan kombinasi data yang berbeda akan tetapi menghasilkan hasil penjumlahan yang sama (Hendrawan, 2004).

Menurut Raharjo (1999) pada sistem digital biasanya ada beberapa mekanisme pengujian integritas seperti antara lain:

1. parity checking
2. *checksum*
3. hash function

Hash function merupakan fungsi yang bersifat satu arah dimana jika kita masukkan data, maka dia akan menghasilkan sebuah "*checksum*" atau "*fingerprint*" (sidik jari) dari data tersebut. Ada beberapa *hash function* yang umum digunakan, salah satunya adalah *Cyclic*

Checksum

Checksum adalah jumlah yang dikalkulasikan sebagai fungsi dari suatu *message*, *message* itu sendiri adalah masukan data yang telah melewati proses *checksum*. Atau *checksum* bisa dianalogikan sebagai sidik jari pada manusia (Shadewa, 2006).

Menurut Prihardhanto (2009) *Checksum* atau hash sum adalah suatu data dengan ukuran tetap yang dihitung dari suatu blok data digital dengan tujuan untuk mendeteksi kesalahan yang mungkin terjadi saat proses transmisi atau penyimpanan. Integritas data dapat diperiksa pada langkah selanjutnya dengan menghitung pula *checksum* dan membandingkannya dengan data sumber / asli. Jika *checksum* tidak sama, maka hampir dipastikan bahwa data tersebut telah berubah, baik disengaja maupun tidak disengaja.

Checksum adalah teknologi untuk menandai sebuah *file*, dimana setiap *file* yang sama harus memiliki *checksum* yang sama, dan bila nilai *checksum*nya berbeda meskipun satu bit saja, maka *file* tersebut merupakan *file* yang berbeda walaupun memiliki nama *file* yang sama. (Harahap, 2010).

Cyclic Redundancy Check 32

Menurut Wijayanto (2007) prinsip kerja *CRC* adalah menganggap suatu *file* yang diproses

sebagai suatu string yang besar dan terdiri dari bit-bit. Kemudian operasikan suatu bilangan *polynomiai* (pernyataan yang terbentuk dari satu atau lebih variabel dan konstanta) yang sangat besar. Untuk menghitung nilai *CRC*, membagi bilangan *polynomial* sebagai representasi dari *file*, dengan suatu bilangan *polynomial* kecil yang sudah terdefinisi untuk *CRC*. Nilai *CRC* adalah sisa hasil bagi tersebut yang biasa disebut dengan *checksum*.

Setiap pembagian pasti menghasilkan suatu sisa hasil bagi (meskipun bernilai 0), tetapi ada perbedaan dalam melakukan pembagian pada perhitungan *CRC*. Secara umum (prinsip aljabar biasa), pembagian dapat dilakukan dengan mengurangi suatu bilangan dengan pembaginya secara terus-menerus sampai menghasilkan suatu sisa hasil bagi (yang lebih kecil dari bilangan pembagi). Dari nilai hasil bagi, sisa hasil bagi dan bilangan pembagi bisa mendapat bilangan yang dibagi dengan mengalikan bilangan pembagi dengan hasil bagi dan menambahkan dengan sisa hasil bagi (Wijayanto, 2007).

Dalam perhitungan *CRC*, operasi pengurangan dan penjumlahan dilakukan dengan mengabaikan setiap *carry* (merupakan angka lebih besar dari batas dan tambahan akan dipindahkan ke sebelah kiri bilangan) yang didapat. Tentu saja hal ini juga akan berpengaruh pada proses pembagian yang menjadi dasar utama dalam melakukan perhitungan *CRC*. Operasi dalam *CRC* juga hanya melibatkan nilai 0 dan 1, karena secara umum perhitungan beroperasi dalam level bit. Secara notasi aljabar CRC32 tuliskan pada persamaan (1):

$$\boldsymbol{a(x).xN} = \boldsymbol{b(x).p(x)} + \boldsymbol{r(x)} \qquad (1)$$

a(*x*) adalah bilangan *polynomial* yang merepresentasikan data, *xN* merupakan nilai 0 sebanyak *W* (banyaknya 0), b(x) hasil bagi yang didapat, p(x) poly dan r(x) sisa hasil bagi yaitu nilai CRC (Wijayanto, 2007).

Cyclic Redundancy Check 32 bit *(CRC32)* melambangkan panjang *checksum* dalam bit (ukuran terkecil data dalam sebuah komputer). bentuk CRC yang disediakan untuk algoritma sesuai dengan ide pembagian *polynomial* dan digunakan untuk memperhitungkan *checksum* yang sama dari seluruh algoritma CRC (Shadewa, 2007).

Algoritma perhitungan CRC32 menurut Shadewa (2007) dari suatu *file* dimulai dari:

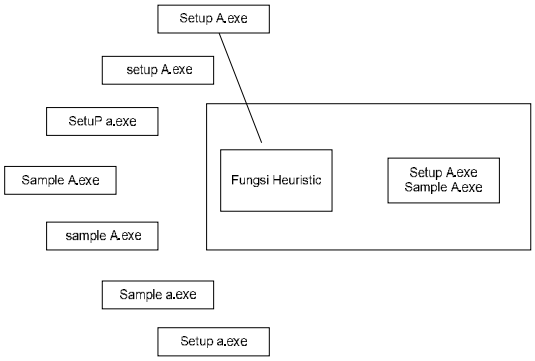
1. Mengambil informasi dari *file* tersebut yaitu nama dan ukuran *file*.
2. Kemudian aplikasi akan membuat tabel perhitungan CRC32 yang disimpan dalam *array* (struktur data yang terdiri atas banyak variabel dengan tipe data sama). Tabel tersebut berisi nilai tiap bit yang akan dibandingkan dengan nilai tiap bit *file* yang ingin dihitung nilai *checksum*-nya.
3. *File* yang ingin dihitung nilai *checksum*-nya dibagi menjadi 8 byte (1 byte merupakan kumpulan dari 8 bit), yaitu dengan mengambil ukuran *file* dan membandingkan dengan Hexa (bilangan yang terdiri dari 16 bilangan, yaitu 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) FFFFFFFF, bit terakhir sebuah nilai adalah F, karena disini pembaginya ingin dijadikan 8 byte maka F tersebut harus ada 8 kali.
4. Aplikasi melakukan penghitungan dari bit yang terakhir, yaitu dengan cara membandingkan tabel CRC32 (Lampiran 1) yang dihasilkan dan FF atau hexa dari 255 dengan buffer *file* (area memori yang menyimpan data) yang telah dipecahkan menjadi 8 bagian.
5. Kemudian nilai *checksum*-nya diambil dari hasil yang tidak sama dengan tabel crc32 yang dihasilkan oleh aplikasi.

Heuristics

Thomas A. Knox (2003) berpendapat mengenai *heuristic:*

"*Heuristics is a difficult concept to explain, but it generally means to apply knowledge gained previously to a new problem. In order to fool a spam solution that uses keyword filtering someone might take the word "badword" and change it somewhat to be b-a-d-w-o-r-d. Since this obfuscation might not be in the list of keywords the spam solution may not trigger on this word. Heuristics would come into play here and allow the spam solution to see past the literal typing of the word and see it for what it is*" (Knox, 2003).

Pengertian dari pendapat Thomas A. Knox (2003) mengenai *Heuristic* adalah menerapkan penelusuran pengetahuan yang diperoleh sebelumnya dalam masalah baru. Dapat dianalogikan dalam pemilihan solusi yang menggunakan kata kunci penyaringan kata dalam pencarian *filename* (nama *file*) pada suatu *software* (perangkat lunak) dengan kata "*Setup A.exe*" dan "*Sample A.exe*" pada *software* "*Program File*" dan perubahan menjadi "*Setup a.exe*" dan "*Sample a.exe*" pada *software* "*Program File*". Karena perubahan ini dimungkin tidak terdapat dalam daftar kata kunci pada solusi yang telah ada, sehingga solusi tidak ditemukan. *Heuristic* berperan dalam kemungkinkan solusi dari perubahan yang terjadi dan menelusuri kejadian mulai dari awal pencarian berdasarkan *filename* dan mengurutkan kemungkinan hasil akhirnya. Pola *heuristic* dapat digambarkan pada Gambar 2.8.


Setup A.exe
setup A.exe
SetuP a.exe
Sample A.exe
sample A.exe
Sample a.exe
Setup a.exe
Fungsi Heuristic
Setup A.exe
Sample A.exe

PEMBAHASAN

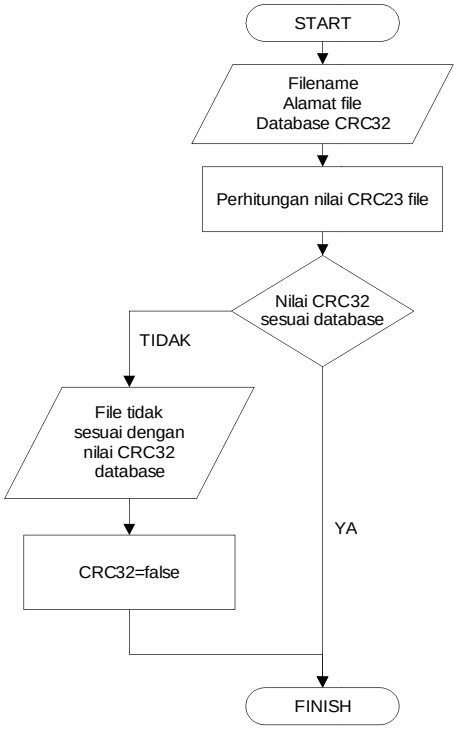
Deskripsi Sistem

Tujuan pembuatan sistem ini adalah untuk menangani kerusakan yang terjadi pada *file-file software* berdasarkan nilai integritas *file*, *pathfile* (alamat *file*) serta ekstensinya dan melakukan penanganan masalah secara otomatis.

Flowchart berikut merupakan implementasi dari beberapa metode yang digunakan dalam penanganan masalah pada *software* menggunakan metode *heuristic integrity checkers* dan *fuzzyfikasi* (logika *fuzzy c-means*) yang ditunjukkan pada Gambar 3.2.

proses yang dikakukan berulang-ulang hingga *file* yang di cari menurut *database* telah selesai *scanning* (pencarian *file*), setelah proses *scanning* selesai maka dilanjutkan pada tahap pendeteksian *file*.

Pada tahap proses pendeteksian dilakukan proses perhitungan nilai *CRC32* terhadap *file* yang ditemukan dan nilai *CRC32* akan di cocokkan nilainya dengan nilai *CRC32 database*, flowchart proses CRC32 dapat ditunjukkan pada Gambar 3.4.



Gambar 3.4 Flowchart *Integrity Check*

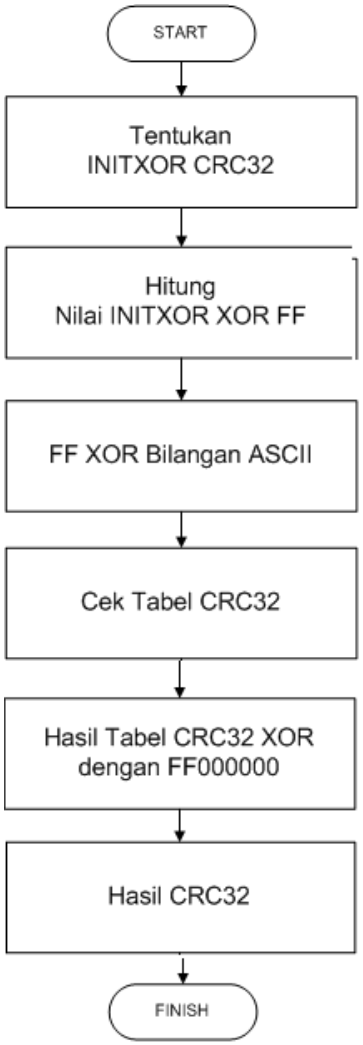
Proses pada Gambar 3.3 file yang ditemukan akan dihitung nilai *CRC32* dan dicocokkan dengan *database*, jika nilai *CRC32* sesuai maka file otomatis masuk dalam perhitungan *fuzzyfikasi*, jika nilai *CRC32* tidak sesuai dengan *database* maka *file* dinyatakan mengalami perubahan atau mengalami kerusakan sehingga atribut *file* akan ditandai dengan nilai *FALSE*.

Dalam perhitungan CRC32 ini akan menjelaskan dimana perhitungan CRC32 ini terdapat banyak bilangan yang akan dioperasiakan. Sebelum melakukan perhitungan dimulai dengan mengambil sampel sebuah file .EXE yang memiliki *filename* "setup.exe". Didalam file setup.exe memiliki isi berupa data karakter "a" (tanpa tanda kutip). Kemudian cek dengan CRC32 generator untuk mengetahui nilai CRC32 dari file setup.exe secara umum. CRC32 generator dapat didownload pada link berikut:
http://virologi.info/download/sourcecodewav.zip

Cara untuk mengetahui CRC32 dari file setup.exe menggunakan CRC32 generator dengan menjalankan aplikasi CRC32 generator dan cari file setup.exe, setelah itu akan langsung muncul nilai CRC32 dari file setup.exe.

Gambar 3.5 Aplikasi CRC32 Generator

Dari Gambar 3.5 dapat dilihat nilai dari file setup.exe adalah E8B7BE43. Untuk lebih jelas dalam perhitungan nilai CRC32 dapat dilakukan dengan langkah-langkah sebagai berikut.



Gambar 3.6 Flowchart perhitungan CRC32

a. Pertama program akan melakukan operasi XOR FFFFFFFF dengan FF, kenapa harus FFFFFFFF dan FF, karena FFFFFFFF adalah INITXOR dari CRC32 (konstanta INITXOR CRC32 adalah FFFFFFFF).

b. Nilai FF digunakan sebagai XOR dari FFFFFFFF Karena FF adalah Decimal dari 255 atau dengan kata lain karena CRC32 mempunyai rumus X pangkat n-1 dan total tabel CRC32 itu ada 256 kolom, maka n - 1 = 255.

c. Untuk mempermudah penghitungan dalam operasi AND, dapat di hitung dengan operasi logika pada MS Windows: FFFFFFFF AND FF = FF. Untuk lebih jelas dapat dilihat pada gambar 3.7.

d.

Operasi AND	0	1
0	0	0
1	0	1

Gambar 3.7 Operasi AND

Hubungan antara Operasi AND dengan FFFFFFFF dan FF adalah sebagai perhitungan secara biner, maka untuk perhitungan diperlukan konversi FFFFFFFF dan FF ke dalam biner:

FFFFFFFF = 1111 1111 1111 1111 1111 1111 1111 1111

FF = 1111 1111

Karena jumlah digit dari konversi biner FFFFFFFF adalah 32, maka FF yang konversi binernya hanya 8 digit, harus disamakan jumlah digitnya, sehingga konversi menjadi:

0000 0000 0000 0000 0000 0000 0000 0000
1111 1111
---------- XOR
0000 0000 0000 0000 0000 0000 1111 1111

sehingga jika dilakukan operasi AND

1111 1111 1111 1111 1111 1111 1111 1111
0000 0000 0000 0000 0000 0000 1111 1111
----------AND
0000 0000 0000 0000 0000 0000 1111 1111

e. Kemudian setelah menjadi FF, FF akan dilakukan operasi XOR dengan 61, nilai 61 itu adalah Hexa dari karakter 'a', dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Nilai ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p

f. FF XOR 61 = 9E , kemudian 9E dikonversi ke desimal, menjadi 158 Desimal. Setelah menjadi 158, lihat di tabel CRC32 (lampiran 1) di bit yang ke 158 adalah 17B7BE43.

g. Setelah mendapat nilai 17B7BE43 lakukan operasi dengan FF000000 yang didapat dari menggeser nilai 000000FF menjadi sebanyak 8 bit:

FF000000 XOR 17B7BE43 = E8B7BE43

atau hitungan manual seperti berikut (lihat gambar 3.6 Operasi AND) 0 AND 0 = 0
konversikan FF000000 dan 17B7BE43 menjadi biner:
1111 1111 0000 0000 0000 0000 0000 0000 ->
dari konversi FF000000
0001 0111 1011 0111 1011 1110 0100 0011 -> dari konversi 17B7BE43
----------XOR
1110 1000 1011 0111 1011 1110 0100 0011

setelah itu 1110 1000 1011 0111 1011 1110 0100 0011 dikonversikan menjadi hexadesimal yaitu: E8B7BE43

Nilai E8B7BE43 adalah nilai CRC32 dari file setup.exe, dengan cara diatas adalah perhitungan dimana terjadi perhitungan operasi XOR dan operasi AND dimana menjadi logika perhitungan utama dalam metode CRC32.

3.2.3 *Fuzzyfikasi* (*Fuzzy C-Means*)

Pada tahap ini dilakukan perhitungan terhadap data-data yang diperoleh dari proses *heuristic scan*, maka didapat data dari hasil penghitungan dari kesalahan pada *file software*, dari sini akan dibagi beberapa kelas-kelas untuk menentukan tindakan penanganan masalah. *Flowchart* tahapan proses *fuzzyfikasi* dapat ditunjukkan pada Gambar 3.7.


START

pusat *cluster* yang tepat. Sehingga diperoleh kecenderungan data untuk masuk ke *cluster* mana pada tahap akhir berdasarkan nilai yang terbesar berdasarkan persamaan (5).

Perubahan matrik akan didapat dari proses fungsi objektif dan menghasilkan matrik yang baru dan nilai yang berbeda berdasarkan persamaan (6), selanjutnya hasil dari matrik yang baru akan cek kondisi berhenti berupa *iterasi* yang ditentukan apakah sudah mencapai batas atau tidak, jika masih belum mencapai batas maka kembali pada proses hitung pusat *cluster* pada persamaan (4).

Dari hasil beberapa proses iterasi yang telah ditentukan, maka didapat besarnya pusat *cluster* pada iterasi terakhir dan menghasilkan matrik akhir dan dari matrik tersebut dapat disimpulkan pengelompokan data setelah *iterasi* tertentu, dalam penghitungan *iterasi* sebelumnya akan menghasilkan pusat *cluster* yang baru, pusat *cluster* inilah yang menjadi acuan dalam menentukan data akan masuk pada *cluster* yang sesuai dengan nilai terbesar.

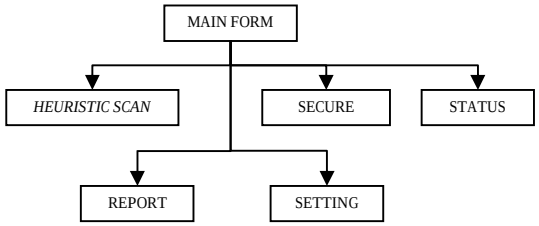
Setelah dilakukan perhitungan hingga data ke-*i* maka akan menghasilkan *cluster*, dimana dari *Cluster1*, *Cluster2* dan *Cluster3* akan di ambil nilai yang paling besar, sehingga data ke-i dapat ditentukan akan masuk pada salah satu *cluster*.

Reporting

Tahap akhir dari sistem ini dilakukan *reporting* aplikasi, dimana hasil dari pencarian kerusakan dan penanganan masalah *software* akan dilaporkan berupa tabel yang menunjukkan kondisi *software* secara keseluruhan dan dapat dilakukan *printout* yang bertujuan sebagai evaluasi dalam *maintenance software*.

Desain menu program HIPO (*Hierarchy Input Process Output*)

Sistem dalam aplikasi dapat digambarkan secara garis besar dengan HIPO dan merupakan alat bantu untuk mendesain dan teknik dokumentasi dalam siklus pengembangan sistem. HIPO aplikasi yang dirancang adalah sebagai berikut:



Gambar 3.9 HIPO Menu Utama

Dari Gambar 3.9 aplikasi utama terdiri dari 1 *MAIN FORM* dan 5 sub menu program, sub menu program tersebut adalah *HEURISTIC SCAN*, *SECURE*, *STATUS*, *REPORT* dan*SETTING*.

1. *HEURISTIC SCAN* adalah sub menu untuk memunculkan awal aplikasi dan memunculkan proses pendeteksian.
2. *SECURE* berfungsi memunculkan proses dalam penanganan masalah berdasarkan hasil *fuzzyfikasi*.
3. *STATUS* untuk menampilkan kondisi sistem komputer setelah dilakukan *scanning*.
4. *REPORT* untuk memunculkan hasil dari proses *SECURE* dan *HEURISTIC SCAN*.
5. *SETTING* untuk menampilkan pengaturan dalam aplikasi.

IMPLEMENTASI DAN PENGUJIAN

Implementasi Program

Pada tahapan ini akan dijelaskan mengenai implementasi inisialisasi *database*, implementasi *heuristic scan*, *fuzzyfikasi* dan implementasi *reporting*.

Inisialisasi *database*

Implementasi inisialisasi *database* menjelaskan mengenai struktur tabel beserta atribut yang ditunjukkan pada tabel 4.2.

Tabel 4.2 Inisialisasi *Database*

No	Tabel Implementasi	Data Type	Atribut
1	LIST PROGRAM	Nama : Text	String
		Lokasi : Memo	String
		Type : Text	String
		CRC32 : Text	String
		NamaSoft : Text	String
		LokasiBackup : Memo	String
2	MUSER	id_user : Text	String
		nama_user : Text	String
		katasandi : Text	String
3	PROFILES CAN RESULT	cfilename : Text	String
		cnamasoft : Text	String
		cpathfile : Memo	String
		nnilaifile : Number	Integer
		nnilaicrc32 : Number	Integer
		nnilaisecure : Number	Integer
		nresult1 : Number	Double
		nresult2 : Number	Double
		nresult3 :	Double

		Number	
		nstatuscheck : Number	Integer
		ddatetime : Date/Time	Date
		crc32scanresult : Text	String
		nurut : Number	Long Integer
		cpathbackup : Memo	String
4	REPORT RESULT	cfilename : Text	String
		cnamasoft : Text	String
		cpathfile : Memo	String
		nstatuscheck1 : Number	Integer
		nstatuscheck2 : Number	Integer
		ddatetime : Date/Time	Date
		nurut : Number	Long Integer
5	TIME ADVANCE	nsettime : Number	Long Integer
		nsettime2 : Number	Long Integer

Pada tahap inisialisasi database dilakukan pencarian seluruh file .EXE dan .DLL yang berada di "C:\Program Files\" dan setiap file yang ditemukan akan dimasukkan ke dalam tabel LISTPROGRAM, dimana setiap file akan dicatat berdasarkan nama, lokasi file, nilai CRC32 dan nama perangkat lunak. Serta proses inisialisasi database ini meliputi *BackUp file* (Peng-Copy-an file yang ditemukan pada "C:\Program Files\" ke dalam "C:\ BackupProgramFiles\").

Implementasi *Heuristic scan*

Pencarian *file* secara *heuristic* berdasarkan data yang ada pada *database LIST PROGRAM* perangkat lunak, maka pada pencarian file dilakukan terhadap data pada *field filename* (nama), *pathfile* (lokasi), *crc32* dan *pathfile backup* (Lokasi *Backup*). Data yang dicari akan terus dilakukan perulangan hingga total *record* data pada *database LISTPROGRAM*.

```
lCRC32 = cCRC32.GetFileCrc32(cstream)
vbandingCRC32 = Trim(Hex(lCRC32))
If vbandingCRC32 = Trim(Me.ListData.ListItems(vjumdata) _
_.SubItems(3)) Then
vpathbackup = Trim(Me.ListData.ListItems(vjumdata)._
_SubItems(5)) & "\" & Trim(Me.ListData.ListItems(vjumdata).Text)
```

Pada proses *heuristic scan* dilakukan pencarian, pendeteksian dan memberi aitribut pada *file* yang dicari berdasarkan bobot yang telah di tentukan sebelumnya berdasarkan *urgenitas* (prioritas atribut) data dalam pengolahan pada aplikasi ini.

Dari pembobotan pada proses pendeteksian *file* diberi nilai 5 jika *file* yang dicari sesuai dengan *filename* pada *database* dan nilai 4 jika tidak ditemukan pada *pathfile*. Dalam arti kertika aplikasi ini berjalan dan dilakukan proses *heuristic scan*, maka aplikasi ini dapat menentukan kondisi *file* yang akan dicari berdasarkan atribut tersebut.

Pendeteksian *file* pada aplikasi ini menggunakan metode integritas nilai *CRC32*, dimana setiap *file* akan dihitung nilai hash *file*nya dan akan di cocokkan nilai *CRC32* dengan nilai yang ada pada *database*.

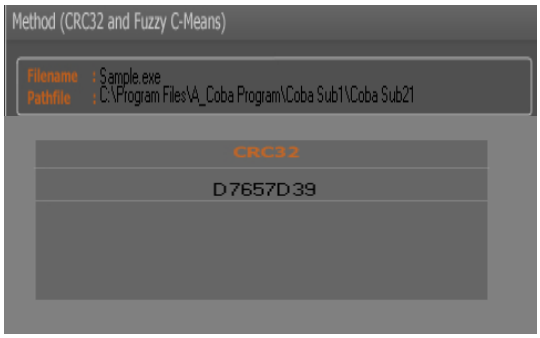
Dari *database LISTPROGRAM* pencarian *file* menandakan bahwa jika atribut *file* adalah 4, maka secara otomatis nilai CRC32 tidak akan ditemukan atau nilai atribut 6 (tidak sesuai), karena syarat untuk perhitungan nilai CRC32 adalah dengan ukuran dan *bitstring* dari *file* itu sendiri.

Ketika atribut pada tahap pencarian *file* adalah 5 maka pada proses *CRC32* akan memanggil *ClassModules* dari integritas *CRC32* dan akan dibandingkan hasil nilai *CRC32* dari *file* yang dicari dengan nilai yang ada pada *List* pada *database*, jika sesuai maka atribut untuk nilai *CRC32* adalah 8 dan jika tidak sesuai maka nilainya 6.

Berikut adalah listing program untuk menghitung nilai *CRC32* berdasarkan *hash* dari *file* yang ditemukan.

```
(ClassModules)
    iLookup = (crc32Result And &HFF) Xor buffer(i)
    crc32Result = ((crc32Result And &HFFFFFF00) \ &H100) And
    16777215        crc32Result = crc32Result Xor crc32Table(iLookup)
```

Ketika aplikasi memanggil *ClassModules* ini maka akan muncul keterangan dari nilai *CRC32* dari *file* tersebut yang di tunjukkan pada Gambar 4.2.



Gambar 4.2 Nilai *CRC32* pada *File*

Setelah selesai pada tahap 2 yaitu pendeteksian nilai *CRC32*, maka proses *scan* akan lanjut pada tahap 3 yaitu pencarian *BackupProgramFiles* yang terdiri dari *file-file* backup dari *file* perangkat lunak. Jika *file backup* ada maka atributnya adalah 7, jika *file backup* tidak ada maka bernilai 3.

Implementasi *Fuzzyfikasi* (*Fuzzy C-Means*)

Implementasi logika *fuzzy c-means* dalam aplikasi ini ada 6 tahapan, dimana data untuk atribut telah diperoleh dari proses *heuristic scan* dapat ditunjukkan pada Gambar 4.3. Logika *fuzzy* ini sebagai dasar untuk menentukan langkah penanganan masalah pada *file-file* perangkat lunak.

File Name	Nilai Bobot: Nilai File	Nilai CRC32	File Secure	SECURE C1	FIX C2	ERROR C3
mute.exe	5	8	7	7.222668397	6.730273961	6.623604614
AccessibleMarsh...	5	6	3	4.229374237	5.988509223	5.999882986
browsercomps.dll	5	8	7	7.222668397	6.730273961	6.623604614
crashreporter.exe	4	6	7	4.036937101	6.00544815	4.273181096
D3DCompiler_43...	5	8	7	7.222668397	6.730273961	6.623604614
firefox.exe	5	8	7	7.222668397	6.730273961	6.623604614
freebl3.dll	5	8	7	7.222668397	6.730273961	6.623604614
glmediias.dll	5	8	7	7.222668397	6.730273961	6.623604614
helper.exe	5	8	7	7.222668397	6.730273961	6.623604614
libEGL.dll	5	8	7	7.222668397	6.730273961	6.623604614
libGLESv2.dll	5	8	7	7.222668397	6.730273961	6.623604614

Gambar 4.3 Hasil atribut *heuristic scan*

Dari hasil pada gambar 4.3 disajikan nilai atribut dari *file* yang di *scan* dan nilai hasil *clustering* (C1, C2 dan C3).

Tahap awal dalam perhitungan *fuzzy c-mean* adalah menentukan atribut dalam matrik *x* yang di dapat dari proses *heuristic scan*, berikutnya menghitung normalisasi dari bilangan acak yang ditunjukkan pada gambar 4.5.

Normalisasi		
Ui1	Ui2	Ui3
0.411764706	0.294117647	0.294117647
0.357142857	0.142857143	0.5
0.285714286	0.333333333	0.380952381

Gambar 4.5 Normalisasi bilangan acak

Langkah selanjutnya dari aplikasi maintenance dalam menentukan *cluster* pada *file* adalah menghitung pusat *cluster* awal.

Dari pusat cluster akan menghitung fungsi objektif serta memunculkan matrik baru untuk menentukan hasil dari pusat cluster, dimana dari hasil pusat cluster yang nilai terbesar akan menjadi acuan peng-*cluster*-an.

Seluruh tahapan pada fuzzy c-mean dalam implementasinya dalam progran dapat dilihat pada listing program berikut:

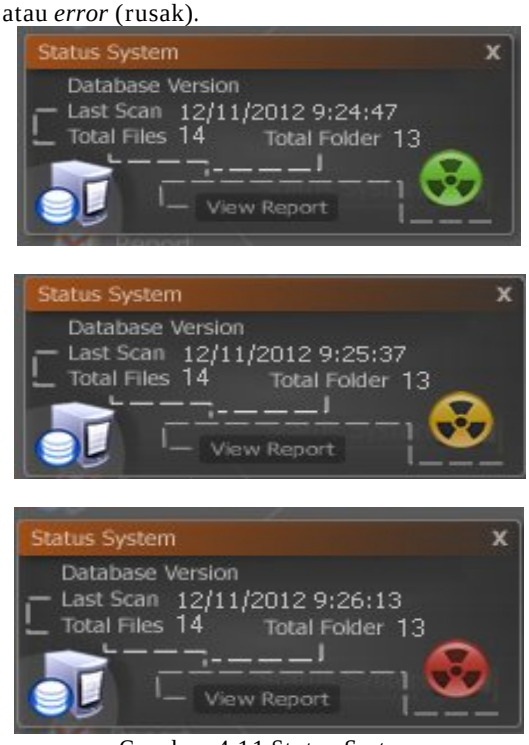
```
Me.Text2.Text = Val(Me.Text1(15).Text) + Val(Me.Text1(16).Text) +
Val(Me.Text1(17).Text)
Me.Text3.Text = Val(Me.Text1(12).Text) + Val(Me.Text1(13).Text) +
Val(Me.Text1(14).Text)
Me.Text4.Text = Val(Me.Text1(9).Text) + Val(Me.Text1(10).Text) +
Val(Me.Text1(11).Text)
...
...

Lihat Lampiran 6
```

Proses berikutnya adalah proses secure dimana melakukan tindakan penanganan masalah berdasarkan cluster.

a. Untuk *cluster* 1 (C1) yaitu *cluster secure*, dimana kondisi *file* adalah tidak terdapat masalah.
b. *Cluster* 2 (C2) yaitu *cluster Fix*, dimana kondisi *file* mengalami kerusakan atau kesalahan data, maka tindakan yang dilakukan adalah me-*restore* dari *file BackupProgramFiles* ke *file* yang dituju.
c. Cluster C3 adalah keadaan dimana *file* yang mengalami kerusakan tidak dapat di tangani oleh aplikasi, sehingga pada hasil laporan akan menunjukkan nilai error.

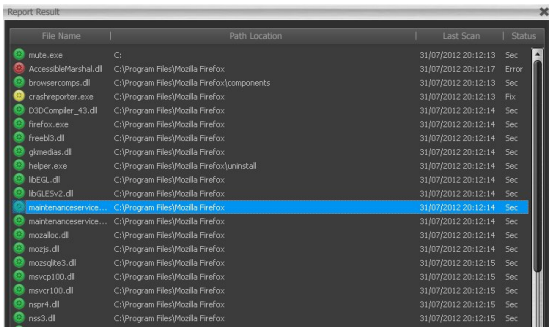
Implementasi *Reporting system*

Pada tahap *reporting* ini pada dasarnya adalah mengambil kesimpulan dari keseluruhan proses, dimana meliputi proses *Heuristic scan* dan proses *secure*. Data pada *database* akan di cek kondisi awal dan kondisi setelah dilakukan penanganan masalah, apakah sistem komputer dalam keadaan *secure* (aman), *fix* (perbaikan) atau *error* (rusak).



Gambar 4.11 Status *System*

Untuk menampilkan tabel *result system* dengan menekan tombol "*View Report*" dan akan muncul *form* yang ditunjukkan pada Gambar 4.12.



Gambar 4.12 *View Report system*

Pengujian

Berikut adalah pengujian *system* yang menunjukkan kinerja aplikasi *mantenance* perangkat lunak menggunakan *heuristic integrity check* dan *fuzzy c-means*. Proses pengujian *system* dalam aplikasi *maintenance* perangkat lunak dilakukan pada data *file* yang berada pada Laboratorium komputer di STMIK ASIA meliputi Lab A, Lab B, Lab C dan Lab D. Masing-masing lab di ambil sampel 15 komputer. Teknik pengujian menggunakan aplikasi bantu berupa program untuk memanipulasi file berdasarkan prosentase dari total file pada database secara acak, sehingga manipulasi file dapat dijalankan dengan sengaja sebagai simulasi kerusakan pada file. Manipulasi file berupa penghapusan file secara acak dari prosentase yang ditentukan.

Pengujian *File* EXE dan DLL

Pada proses pengujian *file* EXE dan DLL akan dibuat tabel perbandingan dari keadaan *file* sebelum terjadi kerusakan dan keadaan file sesudah dilakukan *maintenance*. Kerusakan *file* dirandom dengan jenis-jenis kerusakan meliputi menghapus *file* (*DELETE*), merubah *filename* (*RENAME*), menghapus *file* pada *BackupProgramFiles* (*DELLBACKUP*) dan merubah isi *file* atau mengganti dengan *file* lain (*CHANGE FILE*). Pengujian ini bertujuan untuk mengetahui keakuratan dari *system* proses *maintenance* perangkat lunak, setelah dilakukan pengujian data maka hasil dari proses *maintenance* dapat dilihat pada tabel 4.4.

Tabel 4.4 Pengujian *file* EXE dan DLL

Nama perangkat lunak	Nama file	Status awal	Keadaan file	Status akhir
Acronis	aszbrowsehelper.exe	Secure	DELETE	Fix
Acronis	prl_stat.exe	Secure	DELETE	Fix
Acronis	tishell.dll	Secure	DELETE	Fix
Acronis	TrueImage.exe	Secure	RENAME	Fix
Acronis	TrueImageLauncher.exe	Secure	DELETE	Fix
Acronis	resource.dll	Secure	DELETE	Fix
Acronis	ulxmlrpcpp.dll	Secure	RENAME	Fix
Adobe	apdboot.dll	Secure	DELETE	Fix
Adobe	Bridge.exe	Secure	DELETE	Fix
Adobe	Photodownloader.exe	Secure	DELETE	Fix
Adobe	opera.dll	Secure	DELETE	Fix
Adobe	AdobeUpdater.dll	Secure	RENAME	Fix
Adobe	zlib1.dll	Secure	DELETE	Fix
Adobe	flash.dll	Secure	RENAME	Fix
Adobe	Director.exe	Secure	DELETE	Fix
Adobe	LogSession.dll	Secure	CHANGE FILE	Fix
Adobe	SketchUpReader.dll	Secure	CHANGE FILE	Fix
Adobe	Wintdlst.exe	Secure	DELETE	Fix
Adobe	AGM.dll	Secure	DELETE	Fix
Adobe	msvcr80.dll	Secure	DELETE	Fix
Adobe	Photoshop.dll	Secure	CHANGE FILE	Fix
Adobe	Plugin.dll	Secure	DELETE	Fix
Adobe	About.dll	Secure	DELETE	Fix
Adobe	Adobe Premiere Pro.exe	Secure	RENAME	Fix
Adobe	AMEFoundation.dll	Secure	DELETE	Fix
Adobe	EMUL.dll	Secure	DELETE	Fix
Adobe	libmmd.dll	Secure	DELETE	Fix
Adobe	msvcp80.dll	Secure	DELETE	Fix
Adobe	pxwave.dll	Secure	DELETE	Fix
Adobe	TitleLayout.dll	Secure	DELETE	Fix
Adobe	rtam3290.dll	Secure	DELETE	Fix
Adobe	DeClicker1.dll	Secure	DELETE	Fix
Adobe	pncrt.dll	Secure	DELETE	Fix
Adobe	encn3260.dll	Secure	DELETE	Fix
Adobe	sdpp3260.dll	Secure	DELETE	Fix
AutoCAD 2004	aseloc.dll	Secure	DELETE	Fix
AutoCAD 2004	cspenures.dll	Secure	DELETE	Fix
AutoCAD 2004	msvci70.dll	Secure	DELETE	Fix
AutoCAD 2004	pm8.dll	Secure	DELETE	Fix
AutoCAD 2004	sfl201as.dll	Secure	DELETE	Fix
Adobe	DeHummer6.dll	Secure	DELETE	Fix
Adobe	Phaser6.dll	Secure	DELETE	Fix
Adobe	BEER.dll	Secure	DELETE	Fix
Adobe	AcroRd32.dll	Secure	DELETE	Fix
Adobe	icucnv36.dll	Secure	DELETE	Fix
AnswerWorks 4.0	awApi4.dll	Secure	CHANGE FILE	Fix
AnswerWorks 4.0	LtSpynEn30.dll	Secure	DELETE	Fix
ASIO4ALL v2	asio4all.dll	Secure	DELETE	Fix
AutoCAD 2004	acadinet.dll	Secure	DELETE	Fix

4.1.1 Hasil Pengujian

Dari hasil uji coba *maintenance* perangkat lunak pada komputer A11 dengan *heuristic integrity check* dan *fuzzy* c-*mean* terhadap *file* perangkat lunak didapat hasil penanganan kerusakan sebesar 100% dari total kerusakan file perangkat lunak.

Hasil pengujian terhadap laboratorium Lab A, Lab B, Lab C dan Lab D dapat di tunjukkan pada Tabel 4.5, pengujian dilakukan 2 (dua) kali pada setiap komputer, dimana setiap komputer dilakukan manipulasi file dengan kondisi 20% (dilakukan penghapusan 20% dari total file pada database secara acak) dan 50% (Lihat Lampiran 7).

Tabel 4.5 Pengujian Lab A, B, C dan D

No	Komputer	Manipulasi Data (ke-1)	Manipulasi Data (ke-2)	Hasil ke-1	Hasil ke-2
				%	%
1	A30	20%	50%	100%	100%
2	A29	20%	50%	100%	100%
3	A28	20%	50%	100%	100%
4	A27	20%	50%	100%	100%
5	A26	20%	50%	100%	100%
6	A25	20%	50%	100%	100%
7	A22	20%	50%	100%	100%
8	A23	20%	50%	100%	100%
9	A24	20%	50%	100%	100%
10	A13	20%	50%	100%	100%
11	A14	20%	50%	100%	100%
12	A15	20%	50%	100%	100%
13	A04	20%	50%	100%	100%
14	A05	20%	50%	100%	100%
15	A06	20%	50%	100%	100%
16	B01	20%	50%	100%	100%
17	B02	20%	50%	100%	100%
18	B03	20%	50%	100%	100%
19	B04	20%	50%	100%	100%
20	B05	20%	50%	100%	100%
21	B06	20%	50%	100%	100%
22	B07	20%	50%	100%	100%
23	B08	20%	50%	100%	100%
24	B09	20%	50%	100%	100%
25	B10	20%	50%	100%	100%
26	B11	20%	50%	100%	100%
27	B12	20%	50%	100%	100%
28	B13	20%	50%	100%	100%
29	B14	20%	50%	100%	100%
30	B15	20%	50%	100%	100%
31	C02	20%	50%	100%	100%
32	C03	20%	50%	100%	100%
33	C04	20%	50%	100%	100%
34	C05	20%	50%	100%	100%
35	C05	20%	50%	100%	100%
36	C07	20%	50%	100%	100%
37	C08	20%	50%	100%	100%
38	C09	20%	50%	100%	100%
39	C11	20%	50%	100%	100%
40	C12	20%	50%	100%	100%
41	C14	20%	50%	100%	100%
42	C15	20%	50%	100%	100%
43	C17	20%	50%	100%	100%
44	C18	20%	50%	100%	100%
45	C19	20%	50%	100%	100%
46	D04	20%	50%	100%	100%
47	D05	20%	50%	100%	100%
48	D06	20%	50%	100%	100%
49	D10	20%	50%	100%	100%
50	D11	20%	50%	100%	100%
51	D12	20%	50%	100%	100%
52	D16	20%	50%	100%	100%
53	D17	20%	50%	100%	100%
54	D18	20%	50%	100%	100%
55	D22	20%	50%	100%	100%
56	D23	20%	50%	100%	100%
57	D24	20%	50%	100%	100%
58	D25	20%	50%	100%	100%
59	D26	20%	50%	100%	100%
60	D27	20%	50%	100%	100%
Total Rata-Rata Prosentase				100%	100%

Hasil uji pada Tabel 4.5 menunjukkan bahwa perlakuan penanganan masalah dengan pendeteksian file berdasarkan CRC32 dan menentukan kelas-kelas keadaan menggunakan logika Fuzzy C-means dapat menghasilkan akurasi rata-rata 100%. hal ini sesuai dengan pendapat Harahap (2010) bahwa nilai CRC32 dapat menandai sebuah *file*, dimana setiap *file* yang sama harus memiliki *CRC32* yang sama, dan bila nilai *CRC32* berbeda meskipun satu bit saja, maka *file* tersebut merupakan *file* yang berbeda walaupun memiliki nama *file* yang sama. Serta menurut Daulay (2006) bahwa logika Fuzzy C-means dapat menentukan kelas optimal, dimana hal ini sebagai acuan dalam menentukan penanganan masalah pada aplikasi otomatisasi maintenance perangkat lunak.

PENUTUP

Kesimpulan

Berdasarkan aplikasi yang telah dibuat beserta ujicoba yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut :

a. Pendeteksian *file* dengan *heuristic integrity check* berdasarkan *checksum error* dapat diterapkan untuk menghitung nilai CRC32 dari sebuah *file*
b. Metode CRC32 hanya dapat membaca pola dari isi sebuah *file* untuk menentukan *checksum* berdasarkan nama *file* dan ukuran byte dari *file*.
c. Ketika terjadi kerusakan *file* berupa *RENAME file*, sistem dapat mengembalikan *file* dengan yang *file*name aslinya, akan tetapi tidak dapat menghapus *file* yang mengalami *RENAME* tersebut, maka ada dua *file* dengan memiliki *checksum* sama tetapi *filename* berbeda.
d. Hasil uji coba *maintenance* perangkat lunak dengan *heuristic integrity check* dan *fuzzy* c-*mean* terhadap *file* perangkat lunak didapat hasil penanganan kerusakan sebesar 100% dari total kerusakan *file* perangkat lunak pada komputer A11 di laboratorium komputer dan sebesar 100% untuk pengujian pada Lab A, B, C dan D.

Saran

Dari hasil pengujian sistem beberapa pengembangan sistem yang diharapkan adalah sebagai berikut:

a. Format ekstensi *file* perangkat lunak yang digunakan untuk pencarian *checksum error* dilakukan dalam semua *file* perangkat lunak dan sistem komputer, sehingga pencarian *file* dapat di optimalkan pada keseluruhan *file*.
b. Pencarian *file-file* perangkat lunak diharapkan dapat dilakukan pada drive C:\ (*Default System Drive*) dan lokasi-lokasi dimana *file* sistem berada, semisal lokasi pencarian terhadap registri sistem komputer.
c. Pada proses *Update Database* diharapkan dapat mengenali *software* yang baru terinstall pada *windows*, sehingga *database* selalu ter-*Update otomatis* ketika penambahan *software*.

DAFTAR PUSTAKA

1. Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. NY. Plenum. 1981

2. Connolly TM dan Begg C. *Database System: A Practical Approaach to Design, Implementation and Management, third edition*. Essex: Pearson Education Ltd. 2002.

3. Daulay, aisyah marlian. *Segmentasi pasar produk mie cepat saji Menggunakan fuzzy c-means*. Bogor. Institut pertanian bogor. 2006

4. Erniwati. *Upaya Meningkatkan Kemampuan Pemecahan Masalah Matematika Siswa Kelas Viii Smp Negeri 2 Depok Dengan Menggunakan Lks Berbasis Pmr Melalui Model Pembelajaran Kooperatif Tipe Stad Pada Pokok Bahasan Panjang Garis Singgung Lingkaran*. Yogyakarta. Universitas Negeri Yogyakarta. 2011.

5. Fathansyah. *Basis Data.* Bandung. Informatika. 2004

6. Harahap, Putri Hartary. *Teknik Pendeteksian Kerusakan File Dokumen Dengan Metode Cyclic Redundancy Check 32 (Crc32).* Medan. Universitas Sumatera Utara. 2010.

7. Hariyanto, Bambang. *Sistem Manajemen BasisData Pemodelan, Perancangan dan Terapannya*. Bandung. Informatika. 2004

8. Hendrawan, Leo. *Keamanan Sistem Informasi*. Departemen Teknik Elektro Bandung. Institut Teknologi Bandung. 2004

9. Knox, Thomas A. *Technologies to Combat Spam*. GIAC *Security Essentials Certification* (SEC) *Practical Assignment*. 2003

10. Melwin Syafrizal Daulay, *Mengenal Hardware-Software dan Pengelolaan Instalasi Komputer*, Yogyakarta. Penerbit C.V ANDI OFFSET. 2007.

11. Muflih, M. *Pengalamatan File Pada Metode Akses Secara Acak Menggunakan Fungsi Hashing*. Al'Ulum. 2008.

12. Noor, rinaldi. Effendi. *File management system: sistem berkas*. Universitas indonesia. 2005.

13. Pressman, Roger S. *Software Engineering A Practitioner's Approach*. New York. McGraw-Hill. 2001

14. Prihardhanto, Muhammad Dhito. *Studi Perbandingan Beberapa Fungsi Hash dalam Melakukan Checksum Berkas*. Bandung. institut Teknologi Bandung. 2009

15. Raharjo, Budi. *Keamanan Sistem Informasi Berbasis Internet*. Bandung. PT Insan Komunikasi Infonesia. 1999.

16. Sadewa, Aat. *Mengenali Virus Lewat Checksum Error dengan metode CRC32*. Yogyakarta. DSI Publishing. 2007

17. Sadewa, Aat. *Rahasia Membuat Antivirus Menggunakan Visual Basic*. Yogyakarta. DSI Publishing. 2006

18. Wijayanto, Indra Sakti. *Penggunaan CRC32 dalam Integritas Data.* Bandung. Institut Teknologi Bandung. 2007