

SISTEM KEAMANAN DATA MENGGUNAKAN ALGORITMA BLOWFISH DENGAN KUNCI SIMETRIK

Taufik Rachman

Dosen Teknik Informatika, STT STIKMA Internasional Malang

ABSTRAK

Keamanan data sangat mutlak diperlukan untuk menjaga keberadaan data dari berbagai gangguan ancaman dari luar.

Adapun untuk mengamankan data ini terdapat berbagai cara dilakukan, yaitu dengan cara enkripsi. Metode untuk mengamankan data dilihat dari sifat kerahasiaan (secrecy) data pada aspek ancaman keamanan penyadapan (interception), yaitu salah satunya menggunakan algoritma blowfish dengan kunci simetrik. Blowfish merupakan blok chiper 64-bit dengan panjang kunci variabel. Algoritma ini terdiri dari dua bagian: key expansion dan enkripsi data.

enkripsi dengan metode part dihasilkan data terenkripsi yakni jika direpresentasikan pada notepad menjadi data yang tidak dapat dibaca manusia. Pengujian yang dilakukan dengan diberikan noise pada file enkripsi, data tersebut masih dapat dikembalikan dengan proses dekripsi selama noise yang ada tidak merubah header file enkripsi. Dalam hal ini data yang dapat dikembalikan lagi merupakan data yang tidak terkena noise.

1. PENDAHULUAN

1.1. Latar Belakang

Enkripsi data merupakan salah satu cara untuk melindungi data dari gangguan ancaman keamanan. Salah satu aspek ancaman keamanan data tersebut yaitu penyadapan (*interception*), dimana ancaman terhadap kerahasiaan (*secrecy*) data yang mengandung informasi penting dan sensitif keberadaannya bila diketahui oleh banyak orang, misalnya data keuangan dan inteligen. Terdapat tiga kategori enkripsi data yaitu, kunci enkripsi rahasia (kunci simetrik), kunci enkripsi publik (kunci asimetrik) dan fungsi satu arah (fungsi *one-way*).

Untuk mengamankan data dari penyadapan digunakan algoritma kunci enkripsi rahasia (kunci simetrik). Terdapat beberapa algoritma kunci enkripsi rahasia yang berkembang saat ini. Salah satunya algoritma DES (*Data Encryption Standard*) dengan kunci 64 bit yang termasuk dalam kunci enkripsi rahasia (kunci simetrik). Algoritma tersebut sudah dikenal dan banyak digunakan masyarakat luas untuk mengamankan data. Sehingga untuk melanjutkan perkembangan pengetahuan tentang keamanan data maka digunakan algoritma lain yang juga dapat digunakan sebagai enkripsi data, yaitu algoritma blowfish. Untuk implementasinya mudah karena terdiri dari operasi sederhana, yaitu operasi penambahan (*addition*) dan XOR serta tambahan penelusuran tabel. Kunci dari algoritma blowfish ini dapat mencapai 448 bit (56 byte), dimana juga termasuk dalam kunci enkripsi rahasia (kunci simetrik). Algoritma blowfish ini digunakan pada pada sistem operasi linux dan unix untuk sistem keamanannya, selain menggunakan algoritma DES dan MD5.

1.2. Rumusan Masalah

Bagaimana mengimplementasikan suatu algoritma Blowfish sebagai salah satu metode keamanan data?

1.3. Batasan Masalah

Untuk menghindari agar pembahasan tidak melebar dari fokus permasalahan yang dirumuskan, ada beberapa batasan masalah meliputi :

- Metode yang digunakan Blowfish algoritma.
- Data yang diamankan berupa data yang umum digunakan saat ini, misalnya data teks (*.txt dan *.RTF), gambar (*.bmp, *.gif dan *.jpg), audio (*.mp3, *.MID dan *.WAV), video (*.DAT, *.AVI dan *.MPG).

1.4. Tujuan dan Manfaat

Tujuan dari pembuatan aplikasi ini adalah memberikan salah satu solusi dalam mengamankan data yang sifatnya rahasia untuk menghindari adanya gangguan tangan hacker maupun craker yang selalu ingin mengetahui data yang bukan miliknya.

Dan diharapkan bisa digunakan setiap orang atau instansi untuk mengamankan data miliknya dari penyadapan.

1.5. Metodologi

1. Studi literature
2. Desain dan perancangan system
3. Pembuatan perangkat lunak (*software*)
4. Uji coba program

2. TEORI DASAR

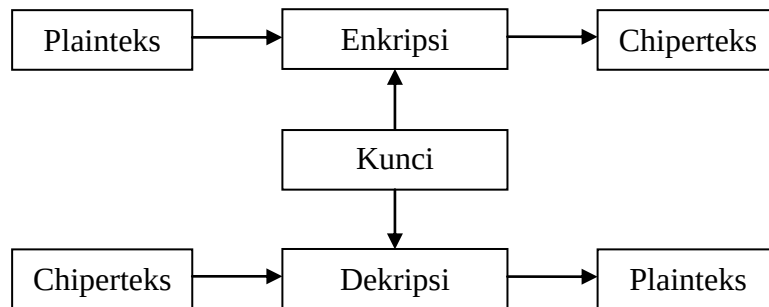
2.1.1. Ancaman Keamanan Data

Aspek yang berkaitan dengan ancaman keamanan antara lain:

1. **Interupction.** Merupakan ancaman terhadap *availability* yaitu data dan informasi yang berada dalam sistem komputer dirusak atau dibuang sehingga menjadi tidak ada dan tidak berguna. Contohnya, hard disk yang dirusak, memotong line komunikasi dan lain-lain.
2. **Interception.** Merupakan ancaman terhadap *secrecy* yaitu orang yang tidak berhak namun berhasil mendapatkan akses informasi dari dalam sistem komputer. Contohnya, dengan menyadap data yang melalui jaringan publik (*wiretapping*) atau menyalin secara tidak sah *file* atau program.
3. **Modification.** Merupakan ancaman terhadap integritas yaitu orang yang tidak berhak yang tidak hanya berhasil mendapatkan akses informasi dari dalam sistem komputer, tetapi juga dapat melakukan perubahan terhadap informasi. Contohnya, merubah program dan lain-lain.
4. **Fabrication.** Merupakan ancaman terhadap integritas yaitu orang yang tidak berhak meniru atau memalsukan suatu objek ke dalam sistem. Contohnya, dengan menambahkan suatu record ke dalam file.

2.1.2. Algoritma simetris.

Algoritma Simetris, kadang-kadang disebut algoritma konvensional, adalah algoritma di mana kunci enkripsi dapat dihitung dari kunci dekripsi dan sebaliknya. Pada kebanyakan algoritma simetris, kunci enkripsi dan kunci dekripsi adalah sama. Algoritma ini, juga disebut algoritma kunci rahasia (*secret-key algorithms*), algoritma kunci tunggal (*single-key algorithms*), atau algoritma satu kunci (*one-key algorithms*). Diperlukan kesepakatan suatu kunci antara pengirim dan penerima sebelum dapat berkomunikasi dengan aman. Keamanan dari algoritma simetris terletak pada kerahasiaan kunci, seseorang bisa melakukan enkrip dan dekrip pesan jika telah mengetahui kunci. Selama komunikasi dianggap rahasia, kunci harus dirahasiakan.



Gambar 2.5 Algoritma kriptografi kunci rahasia [3]

2.2. (Symmetric Encryption Algorithms).

Fungsi-fungsi yang mendasar dalam kriptografi adalah enkripsi dan dekripsi. Enkripsi adalah proses mengubah suatu pesan asli (plaintext) menjadi suatu pesan dalam bahasa sandi (ciphertext). Sedangkan dekripsi adalah proses mengubah pesan dalam suatu bahasa sandi menjadi pesan asli kembali[3].

Enkripsi algoritma simetris (*chipers*) beroperasi pada plaintext (P) dari panjang plaintext (N bytes) dengan menggunakan kunci (K) dari panjang kunci (L_bytes) dan menghasilkan ciphertext (C). Operasi ini disebut enkripsi. Dan operasi kebalikan disebut dekripsi, yang menghasilkan plaintext (P) dari ciphertext (C) dan kunci (K).

2.2.1. Tipe Enkripsi Algoritma Simetris (Chipers)

Ada dua jenis tipe enkripsi algoritma simetris (*chipers*), yaitu *block chipers* dan *stream chipers*[1]. Berikut penjelasan dua jenis *chipers* tersebut:

1. Block-Cipher

Block-cipher adalah skema algoritma sandi yang akan membagi-bagi plaintext yang akan dikirimkan dengan ukuran tertentu (disebut blok) dengan panjang t, dan setiap blok dienkripsi dengan menggunakan kunci yang sama. Jika panjang plaintext bukan merupakan kelipatan dari (ukuran tiap blok) panjang t, maka tambahan karakter (*bytes*) perlu ditambahkan ke plaintext untuk membuat panjangnya merupakan kelipatan dari (ukuran tiap blok) panjang t, proses ini disebut *padding*.

Pada umumnya, *block-cipher* memproses plaintext dengan blok yang relatif panjang lebih dari 64 bit, untuk mempersulit penggunaan pola-pola serangan yang ada untuk membongkar kunci.

Contoh:

Blok cipher : 8 karakter

Plainteks : ENKRIPSI BLOCK CHIPER

length = 21

Padding : $\text{length}(\text{P}) \bmod 8 = 21 \bmod 8$

Padding = 3

Kunci : BLOCK

Ekskansi Kunci :

1	2	3	4	5	6	7	8
B	L	O	C	K	B	L	O

Pola Kunci :

1	5	7	3	4	2	6	8
---	---	---	---	---	---	---	---

Maka susunan plaintext sebagai berikut

Blok :	1	2	3	4	5	6	7	8
Blok 1 :	E	N	K	R	I	P	S	I
Blok 2 :		B	L	O	C	K		C
Blok 3 :	H	I	P	E	R	3	3	3

Dan proses enkripsi tiap blok sebagai berikut:

Indeks →	1	2	3	4	5	6	7	8
Plainteks Blok 1 :	E	N	K	R	I	P	S	I
Kunci →	1	5	7	3	4	2	6	8
Chiperteks Blok 1 :	E	I	S	K	R	N	P	I

Indeks →	1	2	3	4	5	6	7	8
Plainteks Blok 2 :		B	L	O	C	K		C
Kunci →	1	5	7	3	4	2	6	8
Chiperteks Blok 2 :		C		L	O	B	K	C

Indeks →	1	2	3	4	5	6	7	8
Plainteks Blok 3 :	H	I	P	E	R	3	3	3
Kunci →	1	5	7	3	4	2	6	8
Chiperteks Blok 3 :	H	R	3	P	E	I	3	3

Didapatkan chiperteks sebagai berikut:

EISKRNPI C LOBKCHR3PEI33

Untuk algoritma komputer modern, ukuran blok yang umum adalah 64 bits, cukup panjang untuk menghalangi analisa dan cukup kecil untuk dapat dikerjakan.

2. Stream-Cipher

Stream-cipher adalah algoritma sandi yang mengenkripsi data persatuan data, seperti *bit*, *byte*, *nible* atau per lima *bit* (saat data yang dienkripsi berupa data Boudout). Setiap mengenkripsi satu satuan data di gunakan kunci yang merupakan hasil pembangkitan dari kunci sebelum.

Contoh:

Nomer indeks untuk urutan huruf alphabetis

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

16	17	18	19	20	21	22	23	24	25	26	27
P	Q	R	S	T	U	V	W	X	Y	Z	SPASI

Jumlah karakter yaitu 27 karakter dengan spasi

Plainteks : ENKRIPSI STREAM CHIPER

length : 22

Kunci = 6

$K_0 = C_0 = 6$

Chiperteks : $C_i = (P_i + C_{i-1}) \bmod 27$

$i = 1 - \text{length}(P)$

Plainteks : E N K R I P S I S T R E

Chiperteks : K Y I I Y Q Z Z R K E

Plainteks : A M C H I P E R

Chiperteks : F W W Z Q P E J A

Chiperteks : KYI IYQZZRK EFWWZQEJA

Dengan nilai 6 sebagai kunci awal atau disebut juga sebagai kunci pembangkit (*trigger*) seterusnya dibangkitkan kunci berikutnya dari enkripsi data dengan kunci sebelumnya. Dan dari contoh tersebut diatas didapatkan chiperteks KYI IYQZZRK EFWWZQEJA.

2.2.2. Teknik Enkripsi Klasik

Sebelum komputer, algoritma secara umum dioperasikan pada plainteks setiap satu karakter pada waktu yang sama. Dapat diilustrasikan sama seperti stream chiper yang beroperasi pada stream karakter. Kriptografi yang berbasis karakter algoritma, terdiri dari substitusi dan transposisi[1].

1. *Substitusi Chiper.*

Untuk algoritma substitusi setiap karakter digantikan (disubstitusikan) dengan karakter lain sesuai dengan kunci.

Substitution Cipher dengan Caesar Cipher[4]

Salah satu contoh dari “substitution cipher” adalah Caesar Cipher yang digunakan oleh Julius Caesar. Pada prinsipnya, setiap huruf digantikan dengan huruf yang berada tiga (3) posisi dalam urutan alfabet. Sebagai contoh huruf “A” digantikan dengan huruf “D” dan seterusnya. Transformasi yang digunakan adalah:

Plain : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Cipher : D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Misalnya plainteks dari kalimat di bawah ini.

Plainteks : ENKRIPSI SUBSTITUSI CHIPER

Chiperteks : HQNULSVL VXEVLWXLV FKLSHU

Caesar cipher disebut juga “monoalphabetic ciphers” karena setiap huruf digantikan dengan sebuah huruf. Mono.

2. *Transposisi Chiper.*

Untuk algoritma transposisi setiap karakter diubah urutannya (ditransposisi) dengan karakter lain sesuai dengan kunci.

Transposisi Cipher dengan Simple Columnar

Metode terbaik transposisi yang diketahui, yaitu transposisi kolom sederhana, dimana plainteks ditulis secara horisontal pada kertas dan dibaca secara vertikal, yang bekerja sebagai berikut:

Menggunakan suatu kunci atau frase, misalnya TRANSPOSISI, lantas diberikan nomer indeks pada setiap huruf dalam kata kunci atau frase dengan aturan berikut:

1. Nomer indeks yang diberikan dimulai dengan angka 1.
2. Indeks yang pertama diberikan pada huruf dengan urutan paling awal pada urutan huruf alphabetis, misalnya A (1).
3. Kemudian nomer indeks berikutnya untuk huruf dengan urutan setelah huruf awal tadi, misalnya I (2).
4. Jika terdapat huruf yang sama pada kunci, maka diberikan nomer indeks berikutnya setelah huruf sebelumnya sesuai urutan posisinya pada kunci atau frase, misalnya I (3).

Kemudian, pesan ditulis secara mendatar dibawah kata kunci, bila panjang pesan lebih dari panjang kata kunci maka dilanjutkan dibawahnya sama secara mendatar hingga huruf terakhir pesan. Misalnya pesan “ENKRIPSI TRANSPOSISI KOLOM SEDERHANA”. Dapat digambarkan sebagai berikut:

Kunci :	T	R	A	N	S	P	O	S	I	S	I
Pola :	11	7	1	4	8	6	5	9	2	10	3
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Plainteks 1 :	E	N	K	R	I	P	S	I	T	R	A
Plainteks 2 :	N	S	P	O	S	I	S	I	K	O	L
Plainteks 3 :	O	M	S	E	D	E	R	H	A	N	A

Untuk proses enkripsi pesan ditulis ulang diambil tiap-tiap kolom dengan urutan sesuai nomer indeks pada urutan numeric, sehingga pesan teracak menjadi “KPS TKA ALA ROE SSR PIE NSM ISD IHH RON ENO”

2.2.3. Blowfish

Blowfish merupakan metode enkripsi yang mirip dengan DES (*DES like chiper*) dan diciptakan oleh Bruce Schneier yang ditujukan untuk mikroprocessor besar (32 bit ke atas dengan chace data yang besar).

1. Enkripsi Algoritma Blowfish

Blowfish adalah cipher blok 64-bit yang memiliki sebuah kunci yang panjangnya variabel. Algoritma blowfish terdiri dari dua bagian yaitu *key expansion* dan enkripsi data. Blok diagram enkripsi algoritma Blowfish dapat dilihat pada gambar 2.8 [1].

Key expansion mengkonversikan sebuah kunci sampai 448 bit ke dalam beberapa array subkey dengan total 4168 byte.

Enkripsi data terdiri dari sebuah fungsi yang sederhana dengan iterasi 16 kali. Setiap round mempunyai sebuah permutasi *key-dependent* dan sebuah *subsitusi-key* dan *data-dependent*. Semua operasi, penjumlahan dan XOR pada word 32-bit. Hanya operasi tambahan diindek empat lookup data array per round [1].

Blowfish menggunakan sejumlah subkey yang besar. Key ini harus dihitung awal sebelum enkripsi atau dekripsi.

P-array mempunyai 18 subkey 32-bit :

$P_1, P_2, P_3, \dots, P_{18}$

Empat S-box 32-bit mempunyai masing-masing 256 entry [1] yaitu :

$S_{1,0}, S_{1,1}, S_{1,2}, S_{1,3}, \dots, S_{1,255}$

$S_{2,0}, S_{2,1}, S_{2,2}, S_{2,3}, \dots, S_{2,255}$

$S_{3,0}, S_{3,1}, S_{3,2}, S_{3,3}, \dots, S_{3,255}$

$S_{4,0}, S_{4,1}, S_{4,2}, S_{4,3}, \dots, S_{4,255}$

Blowfish adalah sebuah jaringan Feistel yang mempunyai 16 round. Inputnya adalah (x) element data 64-bit. Untuk mengenkripsi (x) yaitu :

Bagi (x) dalam dua bagian 32-bit menghasilkan (x_L) dan (x_R).

Untuk $i = 1$ sampai 16 maka :

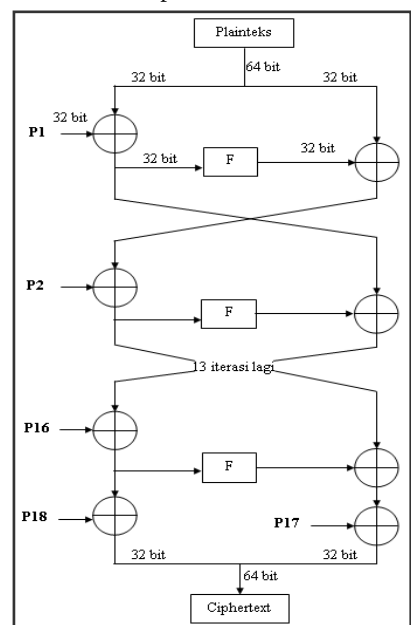
$$x_L = x_L \oplus P_i$$

$$x_R = F(x_L) \oplus x_R$$

Swap (tukar) x_L dan x_R

Swap (tukar) x_L dan x_R (mengulang swap yang

Gambar 2.8 Blok diagram Algoritma Enkripsi Blowfish



lalu)

$$x_R = x_R \oplus P_{17}$$

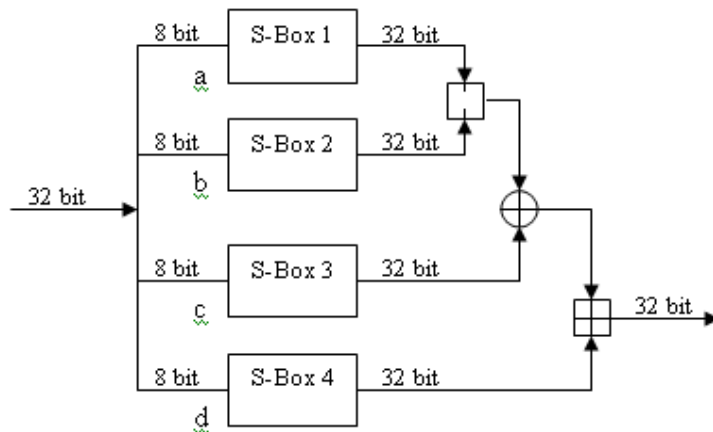
$$x_L = x_L \oplus P_{18}$$

Gabungkan kembali x_L dan x_R [1 3]

Fungsi F adalah sebagai berikut [1]:

Bagi x_L dalam empat kuartier 8-bit yaitu a, b, c dan d seperti gambar 3 maka :

$$F(x_L) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \oplus S_{3,c}) + S_{4,d} \bmod 2^{32}$$



Gambar 2.9 Fungsi F (Bruce Schenier)

Dekripsi Algoritma Blowfish

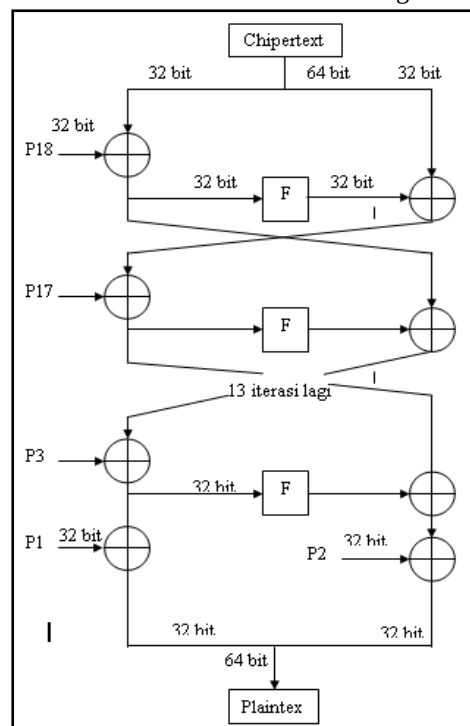
Dekripsi sama persis dengan enkripsi, kecuali bahwa $P_1, P_2, \dots, P_{18}$ digunakan pada urutan yang berbalik (reverse) [1, 3]. Blok diagram dekripsi seperti pada gambar 2.10.

2.3. Pengkodean Base 64.

Pengkodean Base 64 merupakan suatu proses tambahan setelah proses enkripsi data. Akan tetapi pengkodean base 64 ini bersifat opsional/pilihan yakni bisa dilakukan ataupun tidak tergantung pilihan *user* saat akan mengenkripsi data.

Encoding secara sederhana berarti mengubah/mengkonversi data dari satu bentuk menjadi bentuk yang lain yang berbeda, bukan hanya enkripsi data. Base 64 merupakan representasi data berdasarkan 84 karakter alfabet, seperti berikut.

Gambar 2.10 Blok Diagram



Tabel 2.1 Alfabet Base 64

Urutan	Karakter
0 ... 25	'A' ... 'Z'
26 ... 51	'a' ... 'z'
52 ... 61	'0' ... '9'
62	'+'
63	'/'

Data biner terdiri dari nilai 0 dan 1, kemudian jika dikelompokkan menjadi 6 bit maka setiap kelompok akan didapatkan nilai desimal antara 0 dan 63. Kemudian nilai-nilai desimal tersebut dikonversi dengan alfabet base 64 sesuai urutan pada tabel diatas. Setiap nilai desimal tersebut merepresentasikan huruf alfabet base 64, yakni huruf yang dapat dibaca oleh manusia.

Berikut contoh sederhana konversi data biner ke dalam alfabet base 64.

'001100110011' merupakan kumpulan data biner sebanyak 12 bit

Kemudian dikelompokkan menjadi 6 bit, sehingga mendapatkan 2 kelompok data biner sebagai berikut:

'001100' dan '110011' 2 kelompok data biner 6 bit

Setiap kelompok tersebut jika diubah kedalam nilai desimal akan didapatkan nilai 12 untuk kelompok pertama dan 51 untuk kelompok kedua. Lalu nilai desimal tersebut dikonversi kedalam alfabet base 64 menjadi 'Mz', sehingga menjadi karakter yang dapat dibaca oleh manusia.

Permasalahan yang kemudian muncul kebanyakan komputer dalam melakukan penyimpanan dan pemrosesan data menggunakan data biner sebanyak 8 bit atau 1 byte. Sehingga tidak dapat direpresentasikan menjadi satu karakter alfabet base 64. Untuk itu diperlukan minimal 24 bit (3 byte) untuk dapat dikelompokkan menjadi 6 bit. Jika banyak data biner tersebut merupakan kelipatan 24 bit (3 byte) maka tidak menjadi masalah, sebaliknya perlu menambahkan (*padding*) data biner jika tidak merupakan kelipatan 24 bit (3 byte). Ditambahkan 1 atau 2 byte data lagi untuk dapat merepresentasikan menjadi alfabet base 64. Karakter tambahan (*padding*) alfabet base 64 adalah '='.

Berikut contoh sederhana untuk data biner 8 bit (1 byte).

'00000001' merupakan kumpulan data biner sebanyak 8 bit

Kemudian ditambahkan 16 bit (2 byte) data biner agar merupakan kelipatan 24 bit (3 byte) sebagai berikut:

'00000001' '00000000' '00000000' 3 kelompok data biner 8 bit

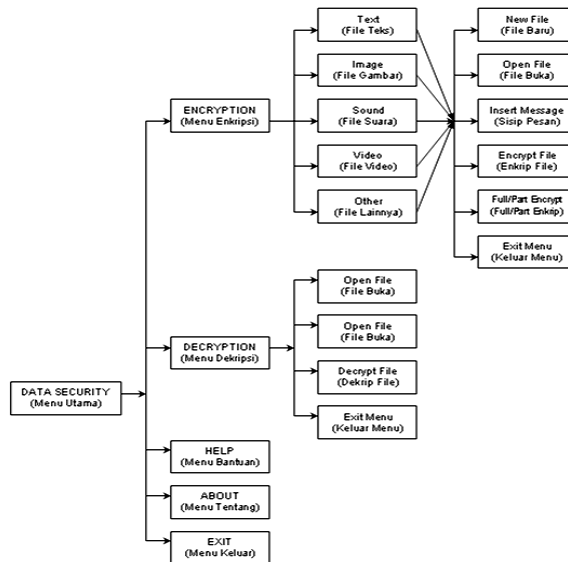
Kemudian dikelompokkan menjadi 6 bit, sehingga mendapatkan 3 kelompok data biner sebagai berikut:

'000000' '010000' '000000' '000000' 4 kelompok data biner 6 bit

Setiap kelompok tersebut jika tersebut dikonversi kedalam alfabet base 64 menjadi 'AQ= ='. Diberikan 2 tambahan karakter '=' pada akhir kumpulan karakter yang mengartikan bahwa terdapat 2 byte tambahan data. Untuk data sebanyak 16 bit (2 byte) maka diperlukan tambahan 8 bit (1 byte) untuk melengkapi menjadi kelipatan 24 bit (3 byte). Sehingga diperlukan 1 karakter '=' sebagai tambahan (*padding*) pada akhir kumpulan karakter alfabet base 64.

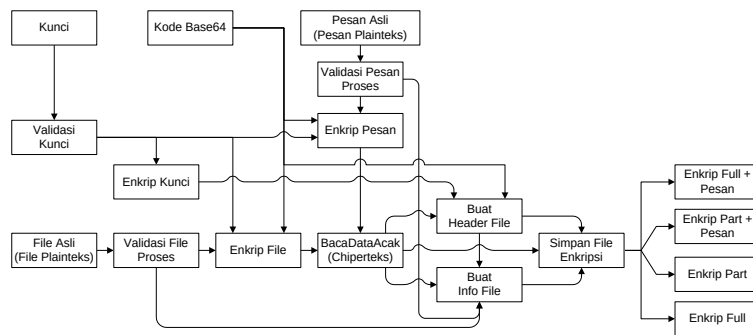
3. RANCANGAN dan DESAIN SISTEM

3.1 Desain Struktur Menu Program



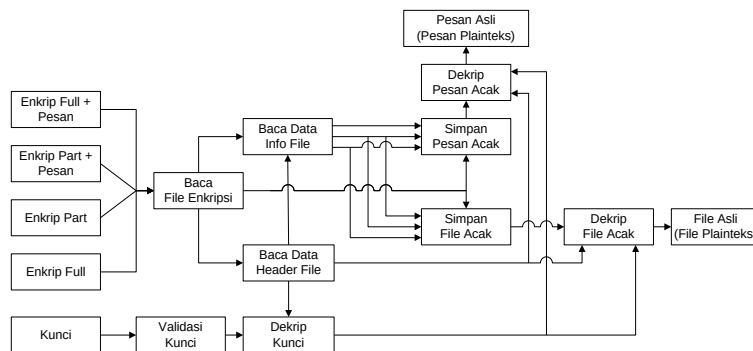
Gambar 3.1 Struktur Menu Program

3.2 Blok Diagram Enkripsi



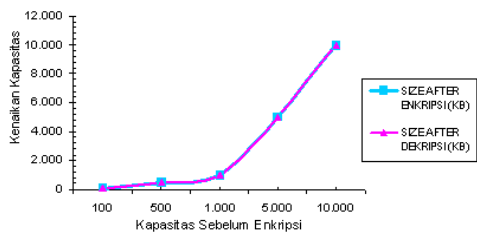
Gambar 3.3 Blok Diagram Enkripsi

Blok Diagram Dekripsi

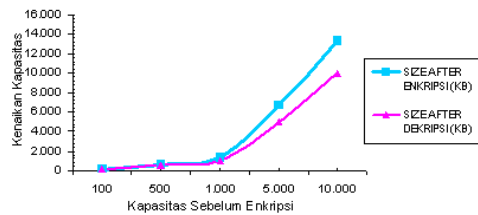


Gambar 3.4 Blok Diagram Dekrip

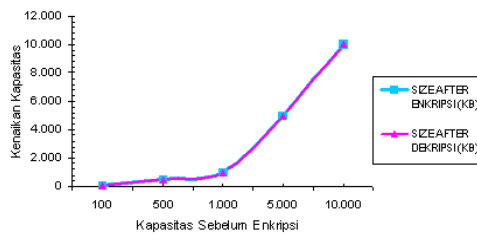
4. HASIL INPLEMENTASI



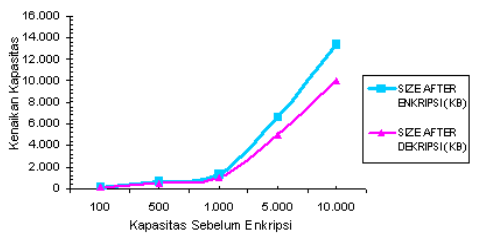
Grafik Perubahan Kapasitas File (*.txt) tanpa pengkodean



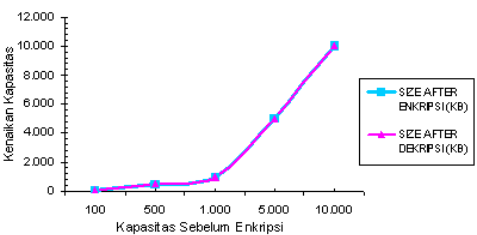
Grafik Perubahan Kapasitas File (*.txt) dengan pengkodean



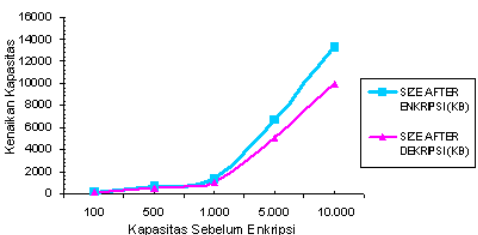
Grafik Perubahan Kapasitas File (*.jpg/ *.jpeg) tanpa pengkodean



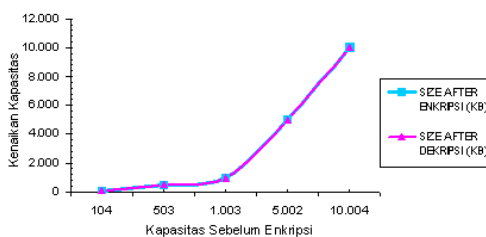
Grafik Perubahan Kapasitas File (*.jpg/ *.jpeg) dengan pengkodean



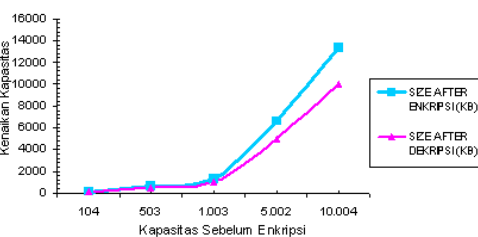
Grafik Perubahan Kapasitas File (*.mp3) tanpa pengkodean



Grafik Perubahan Kapasitas File (*.mp3) dengan pengkodean



Grafik Perubahan Kapasitas File (*.avi) tanpa pengkodean



Grafik Perubahan Kapasitas File (*.avi) dengan pengkodean

5. KESIMPULAN

Dari analisa yang telah dilakukan dapat diambil kesimpulan bahwa kenaikan file enkripsi untuk proses *part* enkripsi tidak begitu terpaut jauh dengan file aslinya. Bahkan tidak terjadi kenaikan untuk file dengan kapasitas tidak mendekati ukuran diatasnya. Kenaikan tersebut disebabkan dengan ditambahkan header file enkripsi yang nilainya tidak sampai dengan 100 byte. Sedangkan untuk *full* enkripsi terjadi kenaikan yang cukup signifikan, hal itu disebabkan selain karena ditambahkan header file enkripsi juga terdapat encode base 64 yang mengakibatkan kapasitas file enkripsi bertambah

menjadi 33% dari file sebelumnya. Data yang terenkripsi merupakan data yang acak sehingga mencegah dari berbagai gangguan. Ditambah dengan adanya encode base 64 menambah tingkat keamanan data dan data tersebut juga telah memenuhi standar *transmitting*, seperti data email atau MIME.

1. Untuk enkripsi dengan metode part dihasilkan data terenkripsi yakni jika direpresentasikan pada notepad menjadi data yang tidak dapat dibaca manusia
2. Sebaliknya dengan metode full dihasilkan data terenkripsi yakni jika direpresentasikan pada notepad menjadi data alfabet sehingga dapat dibaca manusia, tetapi terdapat kenaikan kapasitas data dalam media penyimpanan.
3. Pengujian yang dilakukan dengan diberikan noise pada file enkripsi, data tersebut masih dapat dikembalikan dengan proses dekripsi selama noise yang ada tidak merubah header file enkripsi. Dalam hal ini data yang dapat dikembalikan lagi merupakan data yang tidak terkena noise.
4. Berkaitan dengan no 3, untuk data itu sendiri terdapat header dan file sendiri yang menyertai sebagai informasi format, kapasitas, dan rangkaian informasi lainnya. Jika noise yang diberikan mengenai sekumpulan data penting tersebut maka dalam hal ini data tersebut masih dapat dikembalikan kembali tetapi tidak dapat direpresentasikan sesuai dengan formatnya semula

6. DAFTAR PUSTAKA

1. Bruce Schneier, Applied Cryptography : Protocols, Algorithms, and Source Code in C, USA, John Wiley & Sons, Inc., 1996.
 2. William Stallings, Network and Internetwork Security Principles and Practice, New Jersey, Prentice-Hall, 1995.
 3. Tim Wahana Komputer, 2003, Memahami Model Enkripsi & Security Data, Penerbit Andi, Yogyakarta.
 4. Budi Rahardjo, 1998-1999, Keamanan Sistem Berbasis Internet, PT. Insan Komunikasi Infonesia, Bandung.
 5. Kriptografi <http://id.wikipedia.org/wiki/kriptografi>
 6. Block Cipher <http://www.bkpsecurity.com/crypto.doc>
 7. Justin Hibbits, April 26, 2004, Passwords and Authentication
 8. Transposisi Cipher <http://www.quadibloc.com/crypto/ppen01.htm>
 9. William Stallings, 2000, Dasar-Dasar Komunikasi Data, Salemba Teknika, Jakarta.
 10. Error Code <http://home.ecn.ab.ca/~jsavard/crypto/jsencrypt.htm>
- Randy Charles Morin, How to Base64 <http://kbcafe.com>