IMPLEMENTASI ALGORITMA KRIPTOGRAFI PERMUTASI SUBSTITUSI XOR DINAMIS PADA PROSES DOWNLOAD FILE

Rancak Taufik Hidayat [1)], Linda Mora [2)]
Departemen Ilmu Komputer, Program Pascasarjana Ilmu Komputer, Universitas Budiluhur
Jl. Ciledug Raya, Petukangan Utara, Jakarta Selatan, 12260
Telp : (021) 5853753
rthide1977@gmail.com[1)], lindamora.pln@gmail.com[2)]

Abstract

Currently, Internet is growing so rapidly that almost anyone can use it in many areas of life. With internet users can quickly get the information they want. In the process of getting the data there are times when the data can be retrieved by others who are not eligible for example hackers. Therefore we need a method that can be used to secure our data that can not be retrieved by unauthorized parties. One method I use to secure data is cryptography using permutation substiutsi xor (PSX) dynamically during the process of getting data from a computer server of an agency. The encryption process is done on the server side during the process of downloading the desired file and dsimpan takes place on the client computer. To open the data required applications installed on the client computer. So that not everyone can use it and the result is that data can be obtained safely.

Abstrak

Saat ini internet sangat berkembang dengan pesatnya sehingga hampir semua orang dapat menggunakannya dalam berbagai bidang kehidupan. Dengan internet para pengguna dapat dengan cepat mendapatkan informasi yang mereka inginkan. Pada proses mendapatkan data tersebut ada kalanya data tersebut dapat diambil oleh pihak lain yang tidak berhak misalnya hacker. Oleh karena itulah diperlukan suatu cara yang dapat digunakan untuk mengamankan data kita agar tidak dapat diambil oleh pihak yang tidak berhak. Salah satu metode yang penulis pakai untuk mengamankan data ialah kriptografi dengan menggunakan metode permutasi substiutsi xor (PSX) dinamis pada saat proses mendapatkan data tersebut dari komputer server suatu instansi.Proses enkripsi dilakukan pada sisi server pada saat proses download file yang diinginkan berlangsung dan dsimpan pada komputer client. Untuk membuka data tersebut diperlukan aplikasi yang di-install pada komputer client. Sehingga tidak semua orang dapat memakainya dan hasilnya data tersebut dapat diperoleh dengan aman.

Kata Kunci : *kriptografi, PSX, permutasi, substitusi, xor, instansi*

1. PENDAHULUAN

Data merupakan "harta" yang tidak ternilai harganya untuk suatu instansi. Agar data tersebut dapat sampai pada suatu tujuan diperlukan suatu jalur yang salah satunya bernama internet. Internet dengan berbagai layanannya dapat mempermudah untuk saling bertukar informasi. Akan tetapi didapatkan suatu kendala dimana adakalanya data yang didapatkan tidak utuh atau didapatkan sebagian saja. Hal ini dapat menjadi bumerang terlebih pada instansi pelayanan jasa misalnya perusahaan telekomunikasi yang memiliki data pelanggan. Berangkat dari permasalahan itulah penulis membuat suatu pemecahan masalah berupa teknik penyandian (kriptografi) data, agar pada saat proses mendapatkan data tersebut, data tersebut didapatkan secara utuh.

Kata kriptografi berasal dari bahasa Yunani yaitu *kriptos* (rahasia) dan *graphein* (menulis), sehingga arti keseluruhannya penulisan tersembunyi.Dari pengertian tersebut beberapa ahli mengemukakan pendapatnya :

a. Bruce Schneier [4] : "Kriptografi adalah seni dan ilmu pengetahuan mengenai cara mengamankan berita atau data".
b. Alfred Menezes [1] : "Kriptografi adalah studi teknik matematika yang berhubungan dengan aspek-aspek pengamanan informasi seperti *confidentiality, data integrity* dan *authentication*".
c. Dorothy Elizabeth Robling Denning [7] : "Kriptografi adalah ilmu pengetahuan dan studi penulisan rahasia".

Menurut Dorothy Elizabeth Robling Denning [7] , *cipher* adalah metode penulisan rahasia dimana *plaintext* diubah menjadi *ciphertext*. Proses perubahan dari *plaintext* menjadi *ciphertext* disebut enkripsi. Sedangkan proses pengembalian *ciphertext* menjadi *plaintext* disebut dekripsi. Enkripsi dan dekripsi dikendalikan oleh kunci.

Dalam notasi matematika proses enkripsi [4] dapat digambarkan sebagai berikut :

$$E(M)=C$$

Proses dekripsi :

$$D(C)=M$$

Bila kedua notasi tersebut digabung :

$$D(E(M))=M$$

Dimana : E = fungsi enkripsi
D = fungsi dekripsi
C = teks yang telah ter-enkrip
M = teks asli

Tipe Dasar Enkripsi

a.Substitusi

penyandian dengan mengganti bit atau karakter *plaintext* dengan memakai aturan tertentu. Metode ini terbagi 2 :

1.*Monoalphabetic cipher* : dengan tabel korespondensi untuk men-substitusi sebuah karakter dengan karakter lain. Menurut Alfred Menezes [1] didefinisikan : Misalkan A merupakan alfabet dari simbol terbatas dan M merupakan himpunan semua string dengan panjang t atas A. Misalkan K merupakan himpunan semua permutasi dalam himpunan A. Tetapkan untuk setiap e $\in$K transformasi enkripsi Ee sebagai :
Ee(m) = (e(m1) e(m2) … e(mt)) = (c1 c2 … ct) = c
Dimana m = (m1 m2 … mt) $\in$M.
Dengan kata lain, untuk setiap simbol dalam tuple t, gantikan (substitusi) dengan simbol lain dari A berdasarkan beberapa permutasi tetap e. untuk mendekrip c= (c1 c2 … ct) hitung inversi permutasi d = e-1 dan
Dd(c) = (d(c1) d(c2) … d(ct)) = (m1 m2 … mt) = m
Keuntungan : kapasitas kunci yang besar (255 kemungkinan)
Kerugian : mudah dipecahkan untuk known-plaintext attack
Contoh : Caesar Cipher

2. Polyalphabetic Cipher : dengan dasar peng-enkrip-an meliputi urutan nomor yang panjang tanpa perulangan acak. Contoh : Vernam Cipher yang dibuat oleh Gilbert Vernam untuk AT & T.

b. Permutasi

Perpindahan posisi tiap karakter atau bit plaintext dengan pola-pola yang telah ditentukan dan menjadi acuan dalam

peletakan posisi karakter atau bit yang akan di-enkripsi. Plaintext mengalami perubahan letak posisi tiap karakter sehingga plaintext menjadi susah untuk dipahami.

Menurut Alfred Menezes [1] didefinisikan : misalkan S merupakan anggota himpunan terbatas. Permutasi p pada S adalah bijeksi (jika fungsi f:X→Y merupakan pemetaan satu-satu dan implikasi(f)=Y) dari S pada dirinya sendiri (p:S→S).

Keuntungan : mudah diterapkan

Kerugian : mudah dikenali dari adanya pengulangan karakter yang sama dilihat dari hasil enkripsi, tidak bisa enkripsi berulang-ulang, mudah dipecahkan untuk known-plaintext attack, mudah dipecahkan dengan cipher-only attack.

Metode XOR

Metode awalnya adalah dengan menjumlahkan karakter, karena diperlukan keyword, maka yang dijumlahkan adalah karakter pada input-an dengan karakter pada keyword.

Bentuk umum : Ci= Mi xor Ki

Dimana Ci= ciphertext yang ke-i,

Mi=plaintext yang akan disandikan ke-i,

Ki=kunci yang ke-i, i=0,1, … , 255.

Keuntungan : mudah diterapkan

Kerugian : mudah dipecahkan(dengan meng-xor-kan kembali hasil enkripsi)

Teknik Enkripsi

a.Teknik Substitusi (*Substitution Technique*)

Dengan cara menggantikan setiap elemen data atau karakter dengan karakter lainnya. Langkah pertama yang harus dilakukan untuk melaksanakan teknik ini adalah membuat tabel substitusi. Tabel ini berisi kumpulan berbagai macam karakter pengganti, misalnya karakter A digantikan dengan karakter K, karakter B digantikan dengan M, dan seterusnya. Semakin variatif isi data, semakin lengkap pula tabel substitusi yang harus dibuat. Penggantian karakter tersebut bisa dilakukan melalui urutan tertentu ataupun secara acak (random).

b. Teknik Blok

Dengan cara mengkelompokkan beberapa karakter ke dalam blok-blok, dimana pada setiap bloknya dapat berisi beberapa karakter. Langkah pertama yang harus dilakukan sebelum proses bloking adalah mengurutkan data menurut aturan baris (horisontal) atau aturan kolom (vertikal).

c. Teknik Permutasi

Dengan cara menukar atau merubah letak beberapa karakter. Langkah awal yang perlu dilakukan adalah membagi data ke dalam blok-blok (seperti teknik blok).

d. Teknik Ekspansi

Dengan cara menambahkan suatu karakter atau kata ke dalam data. Proses enkripsi dengan teknik ini cukup mudah untuk dipatahkan. Oleh karena itu, biasanya teknik ini lebih sering digunakan bersama-sama dengan teknik lainnya (dikombinasikan).

e. Teknik Pemadatan

Dilakukan dengan menghilangkan sejumlah karakter dalam data menurut suatu aturan tertentu sehingga data tersebut menjadi berkurang besarnya atau dengan kata lain dipadatkan.

2. METODE PENELITIAN

Metode yang dilakukan ialah dengan mengadakan pengamatan langsung di lapangan pada bagian customer service di instansi tersebut yang mana terjadi proses pengiriman data pelanggan misalnya, dari pusat ke cabang tertentu. Berdasarkan hal tersebutlah dibuatkan algoritma aplikasi kriptografi-nya yaitu dengan menggunakan permutasi, substitusi dan xor. Setelah aplikasi tersebut jadi user diminta untuk mencoba aplikasi tersebut.

3. LANDASAN TEORI

3.1 Pengertian Kriptografi

Kata ‘kriptografi’ berasal dari bahasa Yunani yaitu *kriptos* dan *graphein*. *Kriptos* berarti tersembunyi, rahasia. *Graphein* berarti menulis. Arti keseluruhannya adalah penulisan tersembunyi. Dan kriptografi telah ada sejak jaman Caesar. Untuk lebih jelas mengenai sejarah kriptografi bisa dilihat pada tabel dibawah ini menurut William Stalling(1996).

3.2 Tipe Dasar Ekripsi

Pada dasarnya ada dua tipe dasar enkripsi yaitu metode subsitusi dan metode permutsi. Metode subsitusi adalah pnyandian dengan mengganti bit atau karakter plaintext dengan memakai aturan tertentu. Sedangkan, metode permutasi adalah perpindahan posisi tiap karakter atau bit plaintext dengan pola-pola yang telah ditentukan dan menjadi acuan dalam peletakan posisi karakter atau bit yang akan di-enkripsi.

3.3 Metode XOR

Metode awalnya adalah dengan menjumlahkan karakter, karena diperlukan *keyword*, maka yang dijumlahkan adalah karakter pada input-an dengan karakter pada *keyword*.

Bentuk umum : $C_i = M_i$ xor K_i

Dimana C_i= ciphertext yang ke-i, M_i=plaintext yang akan disandikan ke-i, K_i=kunci yang ke-i, i=0,1, … , 255.

Keuntungan : mudah diterapkan

Kerugian : mudah dipecahkan (dengan meng-xor-kan kembali hasil enkripsi, untuk lebih jelas lihat tabel 1.

Tabel 1 : Tabel Kebenaran XOR

A	B	A XOR B	(A XOR B) XOR B
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Dari tabel di atas dapat dibuat asumsi bahwa jika pada kolom A merupakan *plaintext* dan kolom B adalah *keyword* yang kita pakai, maka untuk mengembalikan data seperti semula, maka dapat dipakai operator logika yang yang sama yaitu XOR dari hasil XOR yang pertama.

Contoh :

Plaintext : TAUFIK

Password : JKT

Kita harus merubah kode desimal dari ASCII menjadi kode biner 8-bit.

T = 84 = 01010100 I = 73 = 01001001

A = 65 = 01000001 K = 75 = 01001011

U = 85 = 01010101 J = 74 = 01001010

F = 70 = 01000110

Selanjutnya di-XOR-kan masing-masing karakter dengan karakter password. Akan dihasilkan berupa angka biner, lalu dirubah ke dalam desimal kembali dan diambil kode ASCII dari angka desimal tersebut.

T XOR J = 01010100 XOR 01001010 = 00011110 = 30 = ▲

A XOR K = 01000001 XOR 01001011 = 00001010 = 20 = ¶

U XOR T = 01010101 XOR 01010100 = 00000001 = 1 = ☺

F XOR J = 01000110 XOR 01001010 = 00001100 = 12 = ♀

I XOR K = 01001001 XOR 01001011 = 00000010 = 2 = ☻

K XOR T = 01001011 XOR 01010100 = 00011111 = 31 = ▼

Maka dapat disimpulkan bahwa dengan *plaintext* "TAUFIK" dengan *password* "JKT" jika di-enkrip akan menjadi "▲¶☺♀☻▼".

4. Tinjuan Studi

Penelitan yang dilakukan oleh Bayu Setiaji [9] yang dipublikasikan pada tahun 2015 dengan melakukan penelitian pada penyandian data dengan menggunakan algoritma RSA dan LUC. LUC dapat digunakan untuk penyandian data, signature, dan pembangkitan kunci. Algoritma LUC berdasarkan bilangan bulat besar pada deret Lucas. Deret Lucas adalah dua bilangan bulat P dan Q. Teori ini secara umm pertama kali dikembangkan oleh Edouard Lucas pada tahun 1878. Fokus utama yang menjadi perhatian dalam pemakaia deret Lucas adalah untuk pengujian bilangan prima. Pada penelitian ini dilakukan komparasi terhadap dua algoritma tersebut pada sisi ukuran file dan tipe file. Hasilnya ialah algoritma RSA jauh lebih baik daripada algoritma LUC.

Penelitian terkait lainnya ialah yang dilakukan oleh Halim Agung dan Budiman [10] pada

tahun 2015 dengan melakukan penelitian menggunakan Affine Cipher dan RC4 pada sebuah file. Enkripsi pada Affine menggunakan Affine Transformation dengan rumus :

$$C = aP + b \pmod n$$ untuk enkripsi

dan

$$P = a\text{-}1\ C - a\text{-}1b \pmod n$$ untuk dekripsi

RC4 merupakan stream cipher yang dirancang di RSA Security oleh Ron Rivest tahun 1987. Pada mulanya cara kerja RC4 dirahasiakan oleh RSA Security , akan tetapi ini dibocorkan di internet tahun 1994 di milis Cyberpunks.

Hasil yang didapat dari penelitian tersebut ialah ukuran file ikut mempengaruhi lama proses enkripsi dan dekripsi yang dilakukan, algoritma Affine Cipher membutuhkan waktu lebih lama proses enkripsi dan dekripsi.

Suriski Sitinjak, Yuli Fauziah dan Juwairiah [11] melakukan penelitian pada 16 jenis file berbeda. Hasilnya penelitian yang didapat bahwa ukuran file mempengaruhi proses enkripsi dan dekripsi file. Waktu proses untuk enkripsi dan dekripsi berbeda diakibatkan ukuran antara file plainteks dan file cipherteks-nya sedikit sedikit lebih besar dibandingkan plainteks-nya

5. Permasalahan

Dalam proses pengiriman data, unit ini masih memakai *e-mail* sebagai alat untuk mengirimkan data dari pusat ke cabang. Dimana, dalam perjalanannya bisa saja ada orang yang berusaha untuk mengambil data tersebut. Yang nantinya berdampak negatif bagi perusahaan. Dengan kata lain, dengan e-mail keamanan data belum terjamin. Untuk lebih meningkatkan keamanan, sebaiknya data dienkrip dulu.

6. Strategi Pemecahan Masalah

Untuk mencegah masalah diatas perlu diadakan proteksi data. Dimana, semakin tinggi nilai data semakin tinggi tingkat keamanan datanya.

Salah satu cara untuk mengamankan data adalah dengan penyandian (enkripsi). Teknik enkripsi yang nanti dikembangkan nantinya adalah enkripsi dengan metode permutasi, substitusi dan xor dengan dinamis.

Pada proses *upload*, yang dilakukan *administrator*, dimana data yang telah di-*upload* adalah berupa data mentah (*file* teks) yang akan didistribusikan langsung ke cabang-cabang yang mana nantinya data tersebut diolah pada cabang bersangkutan. Untuk mendapatkan data tersebut *user* harus *login* terlebih dahulu sebelum bisa memilih *file* yang akan di-*download*. Pada saat *user* meng-klik *file* yang dikehendaki proses akan berjalan sesuai pilihan *user*. Nantinya akan ada dua pilihan yaitu "*Open this file from its current location*" dan "*Save this file to disk*". Pada pilihan pertama saat *user* memilih atau meng-klik *file* proses enkripsi berlangsung dan selang beberapa waktu kemudian tampil sebuah *form* untuk melakukan proses dekripsi. Sedangkan untuk pilihan kedua *file* yang sudah di-klik akan disimpan dulu ke dalam disk/harddisk lalu untuk melakukan proses dekripsi ada dua cara yaitu dengan klik ganda dan klik kanan | *Open* ini dapat dilakukan dari jendela *explorer*. Dan untuk tampilan selanjutnya sama seperti pada pilihan pertama, muncul *form* untuk melakukan dekripsi.

Adapun gambaran umum mengenai proses yang dilakukan *user* adalah sebagai berikut :

a. *User* melakukan *login* pada situs *web*, setelah berhasil lalu memilih *file* yang akan di-*download* misalnya file dt_jbtk.txt

b. Setelah di-klik *file* yang akan di-*download*, *web server* akan melakukan enkripsi dengan menambahkan ekstensi .psx pada *file* yang telah di-*download* menjadi dt_jbtk.txt.psx.

c. Setelah menerima *file* dt_jbtk.txt.psx *user* selanjutnya memakai program untuk men-dekrip *file* .psx tadi agar dapat dipakai.

7. Algoritma Enkripsi Permutasi Subsitusi XOR

1. Proses Enkripsi

Pada proses ini, yang pertama kali dilakukan adalah membuka *file* asal dan *file* tujuan. Dimana *file* asal haruslah merupakan *file* yang belum pernah di-enkrip dengan program ini. Nama *file* asal harus berbeda dengan *file* tujuan terutama pada ekstensi *file*-nya. Selanjutnya program akan mempersiapkan dan menulis header *file* enkripsi ke *file* tujuan, kemudian dilakukan pengacakan tabel ASCII,

substitusi dengan tabel acuan dan akhirnya hasil substitusi di-xor dengan *password* yang sebelumnya telah dimasukkan oleh *user*. Di bawah ini adalah algoritma tentang proses tersebut. Untuk mendapatkan data mengenai ChkDgt maka program akan menjalankan fungsi GetChkDgt, untuk mendapatkan informasi mengenai ShfDgt program akan menjalankan fungsi GetShfDgt dan untuk proses permutasi substitusi xor, program akan menjalankan fungsi SEnkrip.

Fungsi ChkDgt akan menghitung nilai ASCII setiap karakter dari *password* yang digunakan. Ini dimaksudkan untuk menghindari penulisan *password* pada *file* tujuan. Fungsi ini akan mengubah setiap karakter *password* ke nilai desimal ASCII lalu ditulis ke *file* tujuan.

Pertama kali variabel r dibuat 0 agar dapat menampung nilai r akhir. Program akan menghitung panjang *password* dan akan melakukan perulangan sebanyak panjang *password* tersebut. Saat perulangan berlangsung proses konversi berlangsung. Pada proses ini peng-konversi-an dikalikan dengan nilai i (hitungan perulangan) lalu ditambahkan dengan r. Hasil akhir yaitu nilai variabel result akan ditulis pada *file* tujuan yang berupa nilai desimal ASCII. Misalkan *password*-nya : **RWMANGUN** (8 karakter).

Dengan mengacu pada daftar ASCII dapatlah kita ketahui nilai ASCII dari karakter-karakter tersebut. Tabel 2 menunjukkan nilai ASCII dari *password* yang dimaksud.

Tabel 2 : Tabel Nilai ASCII dari karakter password

Password	R	W	M	A	N	G	U	N
ASCII	82	87	77	65	78	71	85	78

Ketika fungsi GetChkDgt ini dipanggil akan didapatkan nilai :

Tabel 3 : Tabel Perhitungan Fungsi GetChkDgt

I	Kar	ASCII	R + I * ASCII[I]	Hasil
1	R	82	0 + 1 * 82	82
2	W	87	82 + 2 * 87	256
3	M	77	256 + 3 * 77	487
4	A	65	487 + 4 * 65	747
5	N	78	747 + 5 * 78	1137
6	G	71	1137 + 6 * 71	1563
7	U	85	1563 + 7 * 85	2158
8	N	78	2158 + 8 * 78	**2782**

Dari fungsi ini didapat nilai 2782, nilai inilah yang ditulis pada *file* tujuan.

Sedangkan fungsi GetShfDgt akan mengacak nilai untuk mendapatkan ShfDgt. Nilai ShfDgt ini didapat melalui proses pengacakan yang dilakukan komputer. Dan jika ada orang yang berusaha untuk mendapatkan nilai ini maka kemungkinan besar ia akan gagal karena yang mengacak adalah komputer dan orang itu tidak tahu berapa nilai yang sebenarnya karena bersifat dinamis (selalu berubah-ubah). Berikut adalah algoritma mengenai fungsi GetShfDgt

```
1. function GetShfDgt → smallint
2. Randomize    // pengacakan aktif
3. Result ← Random(2000000000) //
   acak dari 0-2miliar
4. return
```

Selanjutnya program akan menjalankan prosedur SEnkrip. Dibawah ini adalah algoritma lengkap SEnkrip.

```
1.  Procedure SEnkrip(FSIn, FSOut: TStream; Pwd: string;
        Geser: integer)
2.  const BufSize=32768 //max pembacaan/blok
3.  //var
4.  Buf : array[0..BufSize-1] of byte //menampung hasil
        pembacaan ke mem.
5.  Terbaca, i : Long Integer
6.  Panggil prosedur PermutasiASCIIEnkrip(Geser)
7.  repeat
8.      isi var Terbaca dari pembacaan file asal dgn max
        baca/blok sebesar BufSize tampung ke Buf
9.    if Terbaca > 0 then // jika ada sisa
10.       for i ← 0 to Terbaca - 1 // iterasi dari 0 sampai sisa
          terbaca
11.           isi var Buf ke-i dari proses substitusi dengan hasil
              PermutasiASCIIEnkrip
12.           isi var Buf ke-i dari hasil substitusi dengan hasil
              PermutasiASCIIEnkrip xor nilai ascii tiap pwd
13.       endfor
14.       Tulis hasil enkripsi ke file tujuan
15.     endif
16. until Terbaca < BufSize
17. return
```

Pada baris ke-7 akan melakukan iterasi sampai nilai Terbaca < BufSize. Misalkan ada *file* sebesar 50 KB (51200 byte). Nilai Terbaca didapat dari pembacaan *file* pertama yaitu sebesar banyaknya BufSize (32768). Dijalankan baris ke-5 dengan pengecekan kondisi dimana 32768 > 0. Dilakukan eksekusi baris-6. Dilakukan perulangan dari i=0 sampai 32768 – 1. Baris ke-11 dimana ASCII[Buf[0]]. Baris ini akan mendapatkan informasi tentang nilai tersebut dari prosedur PermutasiASCIIEnkrip. Dimisalkan isi dari Buf[0]=10, maka nilai ASCII[10]=138. Maka nilai Buf[0]=138. Baris ke-8 nilai dari Buf[0] (138) di-xor-kan dengan *password*. Dimisalkan *password* : RWMANGUN. 138 xor asc(Pwd[0 mod 8 +1]) = 216 (isi Buf[0]). Selanjutnya dilakukan penulisan karakter tersebut ke *file* tujuan.

Pembacaan selanjutnya didapat dari pengurangan ukuran *file* awal (51200 byte) dengan yang telah terbaca diawal (32768) didapat hasil 18432. Dilakukan pengecekan 18432 > 0. Baris dibawahnya dijalankan i=1 sampai 18432 – 1. ASCII[Buf[1]] dimisalkan berisi 19. Maka ASCII[19]=147 (Buf[1]). Selanjutnya nilai 147 xor asc(Pwd[1 mod 8 +1]) = 196 (Buf[1]). Nilai karakter ASCII ini yang ditulis pada *file* tujuan. Jika dibuatkan tabel seperti tampak pada tabel 4.

Tabel 4 : Tabel Output Prosedur Senkrip

Besar File	Buf	Terbaca	Buf[i] (subs)	Buf[i] (XOR)
51200	32768	32768	Buf[0] = 138	Buf[0] = 216
51200	18432	18432	Buf[1] = 147	Buf[1] = 196

Disini jelas bahwa dengan *file* sebesar 50 KB dilakukan iterasi sebanyak 2 (dua) kali. Selanjutnya pada baris ke-2 prosedur SEnkrip akan memanggil prosedur PermutasiASCIIEnkrip. Berikut algoritma prosedur PermutasiASCIIEnkrip

```
1.  procedure PermutasiASCIIEnkrip(Geser:integer)
2.  //var
3.  i : integer
4.  Geser ← Geser mod 256 // dapatkan nilai bulat dari acak
    tabel
5.  if Geser = 0 then   // jika hasil acak tabel=0
6.      Geser ← 128    // isi var acak tabel=128
7.  endif
8.  for i ← 0 to 255   // lakukan iterasi dari range ascii
9.      ASCII[i] ← (i + Geser) mod 256  // acak tiap ascii
10. endfor
11. return
```

Pada prosedur ini variabel geser pertama akan berisi 0 dan di modulus dengan 256 didapat hasil 0. Selanjutnya variabel geser berisi 128. Lalu dilakukan iterasi dari 0 sampai 255 dimana selama iterasi dilakukan pengacakan tabel ASCII.

Dari tabel 5 dijelaskan bahwa pada prosedur ini terjadi pengacakan tabel ASCII dimana pada iterasi pertama yang ke-0 didapat karakter ASCII yang ke-128, iterasi kedua yang ke-1 didapat karakter ASCII yang ke-129 dan seterusnya hingga pada akhir iterasi didapat karakter ASCII yang ke-127.

Tabel 5 : Gambaran Prosedur PermutasiASCIIEnkrip

I	ASCII[I]	Nilai
0	ASCII[0]	128
1	ASCII[1]	129
…… ……	…… ……	…… ……
254	ASCII[254]	126
255	ASCII[255]	127

Setelah selesai akan kembali ke fungsi Enkrip (pemanggil awal) dan selanjutnya penutupan *file* tujuan dan *file* asal.

Proses Dekripsi

Pada proses ini ialah pembalikan proses enkripsi yang telah dilakukan. *File* yang sudah dienkrip yang berisi header ID, tanggal, attribut, ChkDgt, ShfDgt dan panjang nama *file* dibaca. Selain *header* juga terdapat nama *file* asal dari *file* yang terenkrip. Pertama kali yang dilakukan ialah mengecek apakah *file* yang terenkrip memakai program ini ataukah tidak. Bila ya kemudian *user* diminta memasukan *password* untuk mendekrip *file*. Selanjutnya membaca *header* ID dan ChkDgt dari *file* tujuan (terenkrip) yang pada proses ini menjadi *file* asal (FAsal). Jika sama maka informasi tentang *file* asal yaitu nama *file* asal dan panjang nama *file* asal dibaca lalu dijalankan prosedur SDekrip dan set tanggal dan attribut dari *file* tujuan.

8. IMPLEMENTASI

Program ini diimplementasikan pada proses pendistribusian *file* / data *customer VIP* dari kantor pusat ke kantor cabang yang berada di Jalan Daan Mogot Km. 11 Jakarta ke kantor cabang yang tersebar di seluruh Indonesia. Program ini berjalan pada saat suatu *file* di-*download* dari situs *web*. Program ini terbagi dua bagian yaitu program yang bekerja pada sisi *server* / *Active Server Encryptor* (*ASE*) dan program yang bekerja pada sisi *client*. Kedua program ini saling terkait.

Bagian pertama program merupakan program berbentuk *ActiveX Library* yaitu *file* Server.dll. Program ini berfungsi untuk melakukan enkripsi pada *file* yang akan di-*download*. Adapun yang melakukan proses ini ialah *web server* itu sendiri tanpa adanya campur tangan dari manusia sebagai *user* sebelum *file* tersebut dikirim ke *client*.

Bagian kedua berisi rutin dekripsi berbentuk *Executable File* (*.exe) yaitu Klien.exe. Program ini dijalankan pada sisi *client* dan berjalan dibawah sistem operasi Windows 9.x/NT, Windows 2000, Windows ME maupun pada versi Windows terbaru. Program ini berfungsi untuk melakukan dekripsi terhadap *file* yang telah di-*download* dari situs *web* dalam keadaan terenkrip.

9. INSTALASI PROGRAM

Karena program yang dibuat terdiri dari dua bagian, yaitu pada sisi *server* dan sisi *client*, maka instalasi harus juga dilakukan pada kedua sisi (*server* dan *client*).

Sebaiknya instalasi pada sisi server dilakukan memakai komputer Pentium 166 ke atas dengan RAM minimal 16 MB (kebutuhan minimum). Selain itu server minimal memakai sistem operasi Windows 95 dan Internet Explorer 4.0 ke atas. Untuk menghindari hal-hal yang diluar kendali (*hang*, *server down*, dan lain-lain) maka sebaiknya besarnya data dibatasi / idealnya 1 byte sampai 10 MByte.

Pada sisi *client* perangkat komputer yang dipakai adalah komputer dengan spesifikasi komputer minimal Pentium 75 dan RAM minimal 16 MB. Sistem operasi yang digunakan minimal Windows 95 ke atas dengan Internet Explorer 3.01 ke atas. Berikut akan dijelaskan langkah-langkah instalasi pada sisi *server* dan sisi *client*.

Instalasi Pada Sisi Server

Instalasi pada sisi *server* dilakukan dengan cara melakukan registrasi *file ActiveX Library* yang berisi fungsi proses enkripsi. Dalam hal ini *file* yang akan di-registrasi adalah *file* Server.dll. Tujuan registrasi ini adalah untuk menjalankan fungsi enkripsi yang telah dibuat agar dapat dipakai pada *Active Server Page* (*ASP*).

Registrasi dilakukan dengan menjalankan *file* Regsvr32.exe yang merupakan program yang disertakan sewaktu instalasi program Windows 9.x/NT. Dalam hal ini terletak pada direktori Winnt.

Instalasi Pada Sisi Client

Instalasi pada sisi ini dimaksudkan untuk memudahkan *user* menjalankan program dekriptor dalam beberapa kondisi.

Instalasi ini untuk memudahkan pemakai dalam hal penentuan ekstensi *file* pada saat *file* akan di-dekrip lewat jendela explorer dengan nama ekstensi yaitu .psx.

Instalasi ini memanfaatkan *registry* dimana *user* tidak perlu memodifikasi sendiri karena di dalam program executable (Klien.exe) ada prosedur khusus untuk itu.

10. Hasil Pengujian

Uji coba dilakukan dengan berbagai jenis file dalam berbagai ukuran dari 2 byte hingga 4 MB. Pada uji coba ini waktu enkripsi merupakan waktu perkiraan (estimated / ellapsed time) sedangkan waktu dekripsi diambil rata-ratanya saja. Hasil pengujian didapat bahwa ukuran file berpengaruh terhadap waktu enkripsi, sedangkan untuk proses dekripsinya file berukuran 1 KB – 8 MB berkisar antara 0 – 3 detik.

.

Tabel 6 : Tabel Hasil Pengujian

No.	File Asal		File Enkripsi		
	Nama File	Ukuran (bytes)	Nama File	Ukuran (bytes)	Waktu Enkripsi (ms)
1.	Test.asp	1.323	Test.asp.psx	1.353	0.000
2.	Data20.txt	10.096	Data20.txt.psx	10.124	0.000
3.	Dt110.rtf	175.677	Dt110.rtf.psx	175.702	0.000
4.	Klien.exe	433.664	Klien.exe.psx	433.689	0.000
5.	Data.zip	1.153.842	Data.zip.psx	1.153.868	0.017
6.	Data.txt	4.132.389	Data.txt.psx	4.132.413	0.050
7.	Data1.doc	8.872.960	Data1.doc.psx	8.872.987	0.133

11. Perbandingan dengan Algoritma Lain

Berdasarkan dari hasil pengujian yang didapat dari tinjauan studi sebelumnya dapat diberikan bahwa :

a. Pada algoritma Blowfish [4] serta Permutasi Substitusi dan XOR ukuran file ikut mempengaruhi lamanya proses hasil enkripsi dan dekripsi artinya ialah secara algoritmis masih kurang efisien dan efektif.
b. Hasil komparasi antara algoritma RSA dan LUC, algoritma RSA lebih baik dibandingkan dengan LUC

Dari kedua point diatas dapat disimpulkan bahwa algoritma permutasi substitusi xor masih perlu diperbaiki dengan menggabungkan dengan algoritma yang lain atau dengan meng-efisien-kan algoritma yang ada.

12. KESIMPULAN

a. Salah satu bentuk pengamanan data agar tidak jatuh ke tangan yang tidak berhak ialah dengan teknik penyandian.
b. Kerahasiaan data cukup terjamin jika tidak ada program untuk membuka (dekrip) *file* hasil enkripsi tersebut dimana diperlukan *password* untuk membukanya.
c. Nilai ShfDgt yang diacak komputer memungkinkan seseorang tidak dapat membobol enkripsi ini karena nilai ini tidak diketahui secara pasti dan yang mengacak nilai tersebut komputer.
d. Proses enkripsi *file* yang dilakukan di *web server* dilakukan secara otomatis dengan memakai *ActiveX Library* berisikan tentang objek *Active Server Enkripsi* berfungsi melakukan enkripsi terhadap *file* yang dipilih (*download*). *Library* ini harus diimplementasikan pada bahasa *scripting ASP*.

DAFTAR PUSTAKA

1. Alfred Menezes, Handbook Of Applied Cryptography, CRC Press, New York, 1997.
2. Andi Kurniawan, Belajar Sendiri Microsoft Active Server Pages, Elex Media Komputindo, Jakarta, 2000.
3. Antony Pranata, Pemrograman Borland Delphi, Penerbit Andi, Yogyakarta, 1997.
4. Bruce Schneier, Applied Cryptography Protocols Algorithm and Source In C, John Wiley And Sons, Second Edition, USA, 1996.
5. D.S. Steve, Mengenal Teknik-teknik Enkripsi Data, Mikrodata volume 1, seri 12, hal 45, Januari 1997.
6. Dipa Pamitrapati, Krisdianto Siahaan, Trik Pemrograman Delphi, Elex Media Komputindo, Jakarta, 2000.
7. Dorothy Elizabeth, Robling Denning, Cryptography and Data Security, Addison-Wesley Publishing Company, Purdue University, January 1982.
8, Steve Teixeria, Xavier Pacheco, Delphi 5 Developer's Guide, Sams Publishing, USA, 1999.
9. Setiaji, Bayu. 2015, Analisis dan Implementasi Algoritma Kriptografi Kunci Publik RSA dan LUC Untuk Penyandian Data, Jurnal Ilmiah DASI, Vol. 16 No. 3 September 2015, hlm 27 – 36.
10. Agung, Halim dan Budiman. 2015, Implementasi Affine Chifer dan RC4 Pada Enkripsi File Tunggal, Prosiding SNATIF ke-2 Tahun 2015, ISSN : 978-602-1180-21-1
11. Sitinjak, Suriski, Fauziah, Yuli dan Juwairiah. 2010, Aplikasi Kriptografi File Menggunakan Algoritma Blowfish, Seminar Nasional Informatika (semnasIF) 2010, ISSN : 1979-2328