Jurnal Computech & Bisnis, Vol. 1, No. 2, Desember 2007, 105-118
ISSN 1978-9629

SISTEM PENGKODEAN DATA PADA FILE TEKS PADA KEAMANAN INFORMASI DENGAN MENGGUNAKAN METODE SKIPJACK

Suprianto
STMIK Mardira Indonesia, Bandung 40235
supriaja@yahoo.com

Abstract

At this time the internet has become an important part of the IT world, with the number of Internet users began large corporations to private use this also makes the data traffic becomes very much. With the increasing number of traffic data is then began to appear are the people who want to know the data of others for the benefit of themselves, these people are called hacker. This can be prevented by encrypting a document or data to be transmitted so as to be read by unauthorized parties, there are several ways to encrypt the data, the authors chose the method of data encryption skipjack. This method is to encrypt a text file that is sourced from a network application examples Internet e - mail service and have 32 times the stage with the inclusion of the permutation formula and rule a, b rule that is part of the method, the method of data to be transmitted over the Internet is encrypted by sender. Then after I got to the receiver data is encrypted or decrypted the ciphertext back into plaintext data is original. Use of skipjack method with "Security Software Skipjack" made, the encryption and decryption process is proven to help maintain the validity of the information derived from the sender to the receiver.

Keywords*: Hacker, Skipjack Method, "Security Software Skipjack", Valid Information*

Abstrak

Pada saat ini internet telah menjadi bagian penting dalam dunia TI, dengan banyaknya pengguna internet dari mulai perusahaan besar hingga penggunaan pribadi hal ini juga membuat lalu lintas data menjadi sangat banyak. Dengan makin banyaknya lalu lintas data maka mulai bermunculan pula orang-orang yang ingin mengetahui data orang lain demi keuntungan diri sendiri, orang tersebut disebut *hacker.* Hal ini dapat dicegah dengan mengenkripsi dokumen atau data yang akan dikirimkan sehingga dapat dibaca oleh pihak yang tidak berkepentingan, ada beberapa cara untuk mengenkripsi data, penulis memilih enkripsi data dengan metode skipjack. Metode ini adalah mengenkripsi file text yang bersumber dari suatu aplikasi jaringan internet contoh e-mail service dan memiliki 32 kali tahapan dengan disertakannya rumus permutasi dan rule a, rule b yang merupakan bagian dari metode tersebut, dengan metode tersebut data yang akan dikirimkan melalui internet dienkripsi oleh *sender*. Kemudian setelah sampai kepada *receiver* data yang dienkripsi atau *chipertext* didekripsi kan kembali menjadi data semula atau plaintext. Penggunaan metode skipjack dengan "Perangkat Lunak Keamanan *Skipjack*" yang dibuat, proses enkripsi dan dekripsi terbukti dapat membantu menjaga kevalidan informasi yang berasal dari *sender* ke *receive.*

Kata Kunci: *Hacker*, *Metode Skipjack*, "Perangkat Lunak Keamanan *Skipjack*", Informasi Yang Valid

I. PENDAHULUAN

1.1 Latar Belakang Masalah

Dengan berkembangnya sistem komputer dan interkoneksinya melalui jaringan telah meningkat, tentu saja dalam hal ini sangat membutuhkan keamanan data yang sangat handal agar supaya terhindar dari penyadapan yang tidak diinginkan. Dalam hal ini keamanan data merupakan permasalahan yang sangat penting, karena data yang tidak terjamin kerahasiaannya akan dapat dengan mudah dimanfaatkan atau diambil oleh orang yang tidak berhak.

Dalam menjaga keamanan data, tentunya sangat diperlukan suatu sistem yang menjaga agar data tersebut terjamin kerahasiaan dan keutuhannya. Dalam pengiriman data yang melalui jaringan, diperlukan suatu metode agar data tersebut tidak hilang atau utuh sampai tujuan. Dalam penyimpanan data diperlukan suatu keamanan agar data yang disimpan tidak dapat dibuka, dibaca bahkan diambil oleh orang yang bukan haknya untuk mencoba membukanya

Dalam system keamanan data dikenal sebuah metode enkripsi yang mempunyai kode-kode pengamanan untuk mengacak data dan juga mempunyai kode- kode untuk mengembalikan data yang teracak ke data yang sebenarnya. Salah satu metoda yang akan dibahas pada tugas akhir ini adalah Algoritma *Skipjack*.

Dari latar belakang masalah yang ada, maka penulis dalam melakukan penelitian ini mengambil judul " **Sistem Pengkodean Data File Teks Pada Keamananan Informasi Menggunakan Metode *Skipjack***"

1.2. Identifikasi Masalah

Sesuai dengan latar belakang yang ada, maka penulis mengidentifikasi masalah tersebut yaitu :

1. Sering terjadi data yang tidak dienkripsi sehingga dapat dengan mudah dibaca kerahasiaannya
2. Sering terjadi data yang didistribusikan melalui jaringan atau melalui internet walaupun sudah dienkripsi, namun mudah dibaca kode kerahasiaanya karena sistem keamanannya masih kurang.

1.3. Batasan Masalah

Dari identifikasi yang telah penulis sampaikan, penulis memberikan batasan masalah yang akan penulis teliti yaitu :

1. Sistem keamanan data pada file yang berekstensi .txt atau *file* teks
2. Proses Enkripsi dan Dekripsi Pada *File* teks menggunakan Algoritma *SkipJack*
3. Blok *cipher* menggunakan *input/output* 64 bit, dengan panjang kunci 80 bit, dan banyaknya pengulangan 32 kali.

1.4 Tujuan Dan Kegunaan Penelitian

1.4.1. Tujuan

Adapun tujuan dari Penelitian ini adalah :

1. Menganalisis dan membuat sistem keamanan pada *file text* menggunakan metode *Skipjack* sehingga file tersebut terjaga kerahasiaanya.
2. Menggunakan algoritma Skipjack yang sulit untuk di tebak kode kerahasiaanya karena dilakukan proses sebanyak 32 kali putaran

II. LANDASAN TEORI

2.1 Kriptografi

Berbagai macam layanan komunikasi tersedia di internet, di antaranya adalah web, e-mail, newsgroups, dan lain sebagainya. Dengan semakin maraknya orang memanfaatkan layanan komunikasi di internet tersebut, maka permasalahan pun bermunculan, apalagi ditambah dengan adanya *hacker* dan *cracker*. Banyak orang kemudian berusaha menyiasati bagaimana cara mendeteksi keaslian dari informasi yang diterimanya.

Pesan atau informasi yang dapat dibaca disebut sebagai *plaintext* atau *cleartext*. Teknik untuk membuat pesan menjadi tidak dapat dibaca disebut sebagai enkripsi. Pesan yang tidak dapat dibaca disebut sebagai *Chipertext*. Proses yang merupakan kebalikan dari enkripsi disebut sebagai dekripsi. Jadi dekripsi akan membuat *Chipertext* menjadi *plaintext*.

Plaintext dinyatakan dengan M (*message*) atau P (*plaintext*). Pesan dapat berupa aliran bit, file teks, bitmap, gambar video digital dan sebagainya. Namun apabila semua pesan tadi berbentuk digital, maka semua pesan tadi dapat dianggap sebagai data biner. Dan data biner juga dapat diperlakukan sebagai sistem bilangan biner. Karena itulah diperlukan matematika untuk menganalisnya. *Plaintext* dapat disimpan maupun dikirim ke jaringan. Dalam sembarang kasus, M merupakan pesan yang akan dienkripsi.

Kriptografi adalah ilmu yang mempelajari bagaimana supaya pesan atau dokumen aman, tidak bisa dibaca oleh pihak yang tidak berhak. Dalam perkembangannya, kriptografi juga digunakan untuk identifikasi pengirim pesan dengan tanda tangan digital dan keaslian pesan dengan sidik jari digital (*fingerprint*). Tugas utama kriptografi adalah untuk menjaga agar baik *plaintext* maupun kunci ataupun keduanya tetap terjaga kerahasiaannya dari penyadap (disebut juga sebagai lawan, penyerang, pencegat, penyelundup pesan, musuh, attacker dan sebagainya). Pencegat pesan rahasia mempunyai akses yang lengkap ke dalam saluran komunikasi antara pengirim dan penerima. Ini sangat mudah terjadi pada jalur internet dan saluran telepon.

2.2 Algoritma dan Kunci

Algoritma kriptografi selalu terdiri dari dua bagian yaitu fungsi enkripsi dan dekripsi. Bila keamanan algoritma tergantung pada kerahasiaan algoritma bekerja, maka algoritma tersebut dikatakan algoritma terbatas (terbatas kemampuannya).

Kriptografi modern dapat menyelesaikan masalah algoritma terbatas ini dipecahkan dengan merahasiakan kunci (*key*) tanpa harus menyembunyikan algoritmanya sendiri. Kunci (K) dapat juga disebut sebagai *password*. Keamanan enkripsi hanya tergantung pada kunci, dan tidak bergantung apakah algoritmanya dilihat orang lain atau tidak. Kunci pada proses enkripsi dan dekripsi tersebut dapat berupa sekumpulan nilai yang panjangnya dibatasi oleh ukuran kunci tersebut. Semakin panjang ukuran kunci maka semakin banyak kombinasi kunci yang ada, tetapi hanya satu kombinasi kunci yang akan digunakan sehingga menambah tingkat pengiriman data.

Bila keseluruhan keamanan algoritma ini tergantung kunci dan tak satu pun yang didasarkan pada detail algoritma maka algoritma ini dapat dipublikasikan dan dianalisis oleh semua orang. Produk-produk yang menggunakan algoritma tersebut dapat diproduksi secara masal. Tidak masalah jika penyadap mengetahui algoritma tersebut, jika tidak diketahui kunci rahasianya, maka tetap tidak bisa membuka pesan yang ada.

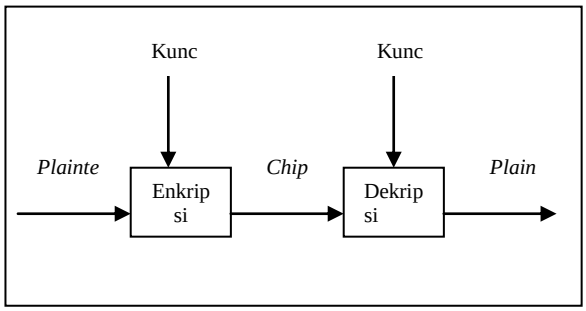
2.3 Algoritma Simetri dan Algoritma Asimetri

Terdapat dua jenis algoritma kriptografi berdasar jenis kuncinya yaitu algoritma simetri (konvensional) dan algoritma asimetri (kunci-publik).

2.3.1 Algoritma Simetri

Algoritma simetri disebut juga sebagai algoritma konvensional adalah algoritma yang menggunakan kunci enkripsi yang sama dengan kunci dekripsinya. Algoritma simetri sering juga disebut algoritma kunci rahasia, algoritma kunci tunggal atau algoritma satu kunci, dan mengharuskan pengirim dan penerima menyetujui suatu kunci tertentu sebelum

dapat dilakukan komunikasi dengan aman. Keamanan algoritma simetri tergantung pada kunci, membocorkan kunci berarti bahwa orang lain dapat mengenkripsi dan mendekripsi pesan, agar komunikasi tetap aman maka kunci harus tetap dirahasiakan. Yang termasuk algoritma kunci simetri adalah *DES*, *RC2*, *Blowfish*, dan sebagainya.



Gambar 2.1. Kriptografi Simetri

Gambar 2.1 memperlihatkan kriptografi simetri yang biasa disebut juga sebagai kriptografi kunci konvensional. Dalam algoritma simetrik, pengirim data dan penerima data menggunakan kunci yang sama. Proses enkripsi dengan menggunakan suatu kunci akan mengubah *plaintext* menjadi *Chipertext* dan proses dekripsi dengan menggunakan kunci yang sama dengan kunci yang digunakan untuk proses dekripsi, maka akan diperoleh data yang tidak sama dengan data asli. Jika kunci untuk proses enkripsi diketahui maka kunci untuk proses dekripsi dapat diperoleh karena menggunakan kunci yang sama.

Fungsi matematika dari proses enkripsi dan proses dekripsi algoritma simetri :
Ek(P) = C
Dk(C) = P
Dk(Ek(P)) = P
dengan P adalah *plaintext*, C adalah *Chipertext*, Ek adalah proses enkripsi menggunakan kunci, dan Dk adalah proses dekripsi menggunakan kunci.
berikut ini adalah kriptografi yang merupakan kelompok Algoritma Simetri
a. Algoritma *Skipjack*

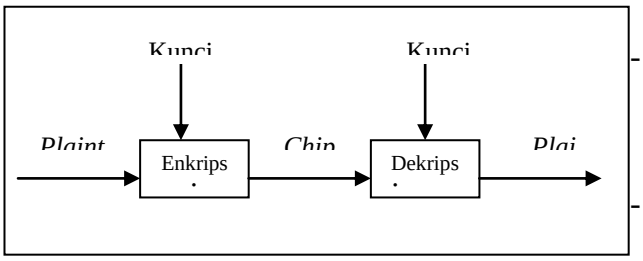
Skipjack merupakan salah satu algoritma simetrik di mana *Skipjack* menggunakan kunci yang sama untuk proses enkripsi dan proses dekripsinya. *Skipjack* memiliki blok data masukan (*Plaintext*) berukuran 64 bit yang kemudian data tersebut diubah menjadi kumpulan blok-blok data yang berukuran 64 bit yang diproses dengan kunci yang sama untuk menghasilkan *Chipertext*. Kunci yang digunakan berukuran 80 bit. Dalam proses enkripsi dan dekripsinya *Skipjack* memiliki 32 putaran artinya algoritma utamanya diputar sebayak 32 kali untuk menghasilkan *Chipertext*. Kombinasi dari 32 putaran tersebut membuat algoritma *Skipjack* memiliki tingkat keamanan yang tinggi.
b. Algoritma *BlowFish*
Algoritma *BlowFish* diciptakan oleh Bruce Scheiner pada tahun 1993 sebagai suatu alternatif dari berbagai macam algoritma dalam kriptografi didesain dengan instruksi proses 32 bit dan dinyatakan lebih cepat disbanding DES
c. Algoritma *Stream Cheaper* dan Block Cheaper
Algoritma *Stream Cheaper* ini adalah cara yang digunakan untuk mengenkripsi data yaitu dengan cara mengenkripsi plaintext dalam 1 bit demi satu bit. Hal ini menjaga agar plaintext dan ciphertext berukuran sama. Berbeda dengan Block Cheaper yang mengenkripsi data dalam 64 bit, Stream Cheaper bekerja lebih cepat dan bila ada data yang hilang dalam satu bit maka hanya hilang satu bit sementara block chipper harus hilang dalam 64 bit atau satu block.

2.3.2 Algoritma Asimetri

Algoritma asimetri atau algoritma kunci publik adalah algoritma kriptografi yang menggunakan kunci yang berbeda pada proses enkripsi dan dekripsinya. Contoh algoritma kriptografi yang menggunakan algoritma simetri adalah RSA dan Diffie-Hellman. Dalam algoritma kunci publik, pengirim data menggunakan kunci yang berbeda dengan kunci yang digunakan oleh penerima data. Kunci yang digunakan

untuk proses dekripsi dinamakan kunci pribadi karena hanya orang-orang tertentu yang berhak menggunakannya.

Proses enkripsi dengan menggunakan kunci publik akan mengubah *plaintext* menjadi *Chipertext* dan proses dekripsi dengan menggunakan kunci pribadi akan mengubah *Chipertext* menjadi *plaintext* kembali. Ukuran kunci publik tidak sama
dengan ukuran kunci pribadi. Algoritma kunci publik memiliki tingkat keamanan data lebih tinggi dibandingkan dengan algoritma simetri. Jika kunci untuk proses dekripsi yaitu kunci pribadi tidak dapat diperoleh sehingga data tetap aman kerahasiaannya. Walaupun algoritma kunci publik memiliki tingkat keamanan data yang lebih tinggi, tetapi waktu untuk proses algoritma ini jauh lebih lambat dibandingkan dengan algoritmasimetri. Algoritma kunci publik dapat dilihat pada gambar 2.2.



Gambar 2.2. Kriptografi Asimetri

Fungsi matematika dari proses enkripsi dan proses dekripsi algoritma kunci publik:

Ek1(P) = C
Dk2(C) = P
Dk2(Ek1(P)) = P

Dengan P adalah *plaintext*, C adalah *Chipertext*, Ek1 adalah proses enkripsi menggunakan kunci, dan Dk2 adalah proses dekripsi menggunakan kunci.

III. ANALISIS SISTEM

3.1. Flow Chart Proses Enkripsi

Plaintext berupa kumpulan karakter yang diperoleh dari memo yang belum dienkripsi, dibagi menjadi blok-blok data berukuran 64 bit yang diolah melalui program pengolahan *Plaintext* secara serial(berurutan). Diagram alir pengolahan *Plaintext* dapat dilihat pada gambar 3.1.

Gambar 3.1. Proses Enkripsi

a. Enkripsi

- Hitung Panjang *Plaintext* jika di mod 8 ada sisa sisipkan karakter 000 sebanyak sisa bagi misalkan plaintext SAYA menjadi SAYA0000
- Potong-potong menjadi 8 *word* atau 64 bit lakukan proses berikut sebanyak 32 putaran x jumlah potongan 8 *word* I = 32 putaran dan J jumlah potongan 8 word

  w1[i]=permutasi(w1[i],j) XOR W4[i] XOR j;

  w2[i]=permutasi(w1[i],j)

  w3[i]=w2[i]

  w4[i]=w3[i]
- Didalam 32 Putaran lakukan proses rule A untuk putaran 1..8 dan 17..24 , Rule B untuk putaran 9..16 dan 25..32
- Dalam Proses *Rule A* dan B lakukan Proses Permutasi yang melibatkan Kunci dan Tabel F
- Sehingga didapat putaran terakhir ke 32 yang dikonversikan ke dalam karakter.
- Hasil akhir didapat ciphertext dari konversi putaran 8 word sebanyak jumlah potongan yang dikonversikan kedalam karakter.
- Periksa kembali ciphertext yang dihasilkan , jika ada karakter (chr(0),chr(10,chr(13) maka sisipkan ‘nilstr’ untuk chr(0), ‘notret’ untuk chr(10) dan ‘bknret’ untuk chr(13)

b. Dekripsi
- Ambil ciphertext yang akan diproses dan dibagi ke dalam potongan 8 word
- Selama panjang ciphertext lakukan pengecekan jika ada char nlstr,bknret,notret dan ubah menjadi chr(0)
- Lakukan putaran sebanyak jumlah potongan dikali 32 putaran menurun .untuk proses w1[32],w2[32],w3[32],w4[32]
- Lakukan Proses G_Invers, Rule A_Invers dan Rule B Invers, sebanyak 32 kali
- Lakukan konversi ke karakter dari hasil akhir w1[0],w2[0],w3[0],w4[0]
- Gabungkan ke dalam karakter dari w1[0],w2[0],w3[0],w4[0] yang dikurangi jumlah sisipan. Sehingga di dapat plaintext kembali.
-

3.2 Mode Operasi Enkripsi Blok *Ciphertext*

Bagian ini membahas mengenai macam-macam operasi yang biasa digunakan dalam kriptografi. Semua algoritma kriptografi konvensional dapat digunakan pada metode operasi ini. Mode-mode digunakan dengan tujuan untuk mengatasi keamanan cara penyandian. Mode-mode lain masih sangat beraneka ragam jenisnya.

Pada setiap mode, pesan *Plaintext* (P) yang panjang dipecah menjadi satuan unit data yang disebut blok. Misalkan saja panjang setiap blok 64 bit, maka blok itu terlebih dahulu harus ditambah dengan bit padding (tambahan) agar jumlah totalnya mencapai 64 bit.

Padding dapat dilakukan dengan penambahan bit "0" hingga mencapai panjang yang diinginkan.

3.3. Enkripsi Dan Dekripsi Dengan Metoda *Skipjack*

Skipjack merupakan salah satu algoritma simetrik di mana *Skipjack* menggunakan kunci yang sama untuk proses enkripsi dan proses dekripsinya. *Skipjack* memiliki blok data masukan (*Plaintext*) berukuran 64 bit yang kemudian data tersebut diubah menjadi kumpulan blok-blok data yang berukuran 64 bit yang diproses dengan kunci yang sama untuk menghasilkan *Chipertext*. Kunci yang digunakan berukuran 80 bit. Dalam proses enkripsi dan dekripsinya *Skipjack* memiliki 32 putaran artinya algoritma utamanya diputar sebayak 32 kali untuk menghasilkan *Chipertext*. Kombinasi dari 32 putaran tersebut membuat algoritma *Skipjack* memiliki tingkat keamanan yang tinggi.

3.3.1 Notasi *Skipjack*

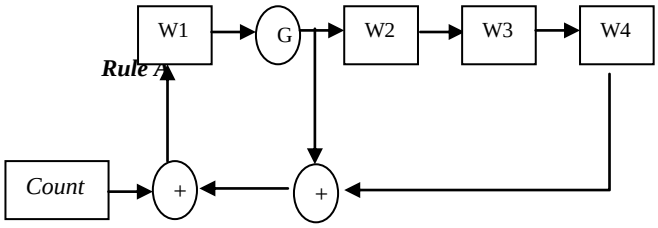
Dalam *Skipjack* terdapat beberapa istilah yang digunakan yaitu antara lain :
• Word : berisi 16 bit
• Byte : berisi 8 bit
• X.Y : XOR dari X dan Y

3.3.2 Struktur Dasar

Skipjack mengenkripsi data sebanyak 4-word (terdiri atas 8-byte, 1-byte = 16-bit). *Skipjack* memiliki 2 macam aturan yaitu *Rule A* dan *Rule B*, aturan ini digunakan secara bergantian dalam proses enkripsi untuk mengubah *Plaintext* menjadi *Chipertext* dan dalam proses dekripsi untuk mengubah *Chipertext* menjadi *Plaintext*.

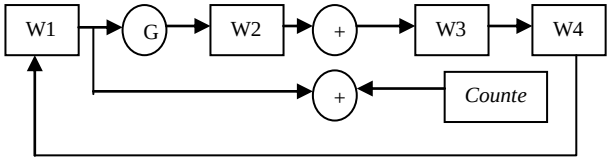
Langkah-langkah dari *Rule A* (ditunjukkan pada gambar 3.4.) :
• G melakukan permutasi untuk w1
• w1 yang baru di XOR dengan G, *counter* dan w4
• w2 dan w3 kemudian bergeser satu register ke kanan, w2 menjadi w3, dan w3
menjadi w4.
• w2 yang baru menjadi G output
• *counter* naik satu pada setiap kenaikan langkahnya

Langkah-langkah *Rule B* (ditunjukkan pada gambar 3.5.) :
• w2 merupakan hasil permutasi w1 oleh G
• w2 kemudian di XOR dengan w3 dan *counter*
• w4 merupakan w3 yang digeser satu register ke kanan menjadi w4
• *counter* naik satu pada setiap kenaikan langkahnya



Gambar 3.4. *Rule A*

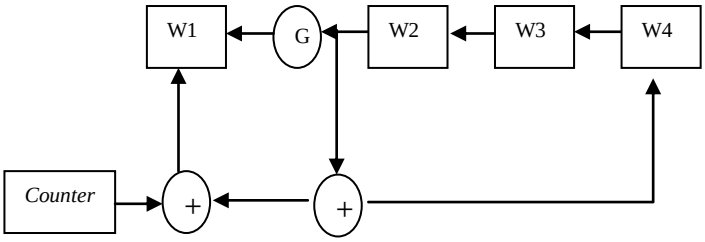
Untuk proses dekripsi *Skipjack Rule A* dan *Rule B* dibalik menjadi *Rule* A^{-1} dan *Rule* B^{-1} untuk menghasilkan kembali *Plaintext*. Gambar 3.6 menunjukkan langkah-langkah dari *Rule* A^{-1} yang merupakan pembalikan dari *Rule A*.



Rule B

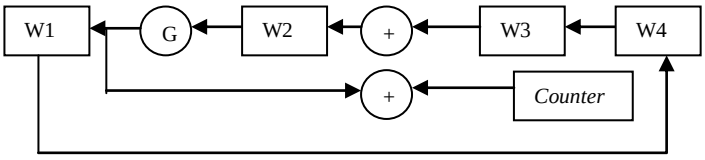
Gambar 3.5. *Rule B*

Gambar 3.7 menunjukkan langkah-langkah dari *Rule* B^{-1} yang merupakan pembalikan dari *Rule B*.



Rule A^{-1}

Gambar 3.6. *Rule* A^{-1}



Rule B^{-1}

Gambar 3.7. *Rule* B^{-1}

3.3.3 Langkah Perhitungan *Rule*

Pada perhitungan di bawah ini, huruf / angka yang ditulis di atas menunjukkan angka kenaikan tiap langkah .

ENKRIPSI

Rule A Rule B

w1k + 1 = Gk(w1k) . w4k . *counter*k w1k + 1 = w4k

w2k+1 = Gk(w1k) w2k+1 = Gk(w1k)

w3k+1 = w2k w3k+1 = w1k . w2k . *counter*k

w4k+1 = w3k w4k+1 = w3k

DEKRIPSI

Rule A-1 *Rule B*-1

w1k – 1 = [Gk-1]-1 (w2k) w1k – 1 = [Gk-1]-1 (w2k)

w2k – 1 = w3k w2k – 1 = [Gk-1]-1 (w2k) . w3k . *counter*k

w3k – 1 = w4k w3k – 1 = w4k

w4k – 1 = w1k. w2k . *counter*k w4k – 1 = w1k

3.3.4 Langkah Proses Pergantian *Rule*

Algoritma *Skipjack* membutuhkan 32 langkah untuk proses enkripsi dan dekripsinya

Proses enkripsi :

- Input wi0, 1 = i = 4 (k = 0, untuk langkah pertama)
- Dimulai saat hitungan pertama (*counter* = 1)
- Proses enkripsi dimulai dengan *Rule A* melakukan putaran sebanyak 8 kali kemudian dilanjutkan oleh *Rule B* sebanyak 8 putaran. Proses enkripsi berlanjut dengan putaran sebanyak 8 kali dari *Rule A* yang kemudian diakhiri oleh 8 putaran dari *Rule B*. Pada setiap kenaikan langkah *counter* naik 1.
- Output yang dihasilkan wi32, 1 = i = 4
- Pada proses enkripsi ini total langkah yang dipergunakan sebanyak 32 kali untuk menghasilkan *Chipertext*.

Proses dekripsi :

- Input wi32, 1 = i = 4 (k = 32, untuk langkah pertama)
- Proses dekripsi dimulai oleh putaran sebanyak 8 kali dari *Rule B*-1, yang kemudian dilanjutkan oleh 8 kali putaran *Rule A*-1, kemudian proses

dekripsi dilanjutkan dengan 8 kali putaran *Rule B*-1 dan diakhiri oleh 8 kali putaran *Rule A*-1. *Counter* berkurang 1 pada setiap langkah berikutnya.

• Output yang dihasilkan wi0, 1 = i = 4

• Untuk proses dekripsi total langkah yang diperlukan sebanyak 32 langkah untuk kembali menjadi *Plaintext*.

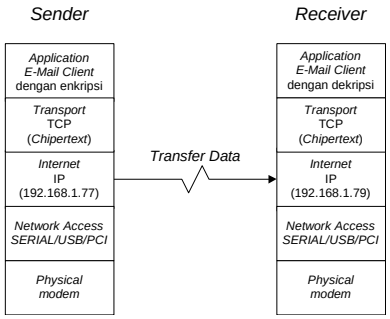
3.3.5 Permutasi G

secara matematik permutasi G yaitu :
Gk (w = g1¦g2) = g5¦g6,
gi = F(gi-1 . cv4k+i–3) . gi-2, k adalah angka setiap penambahan langkah (angka
pertama adalah 0), F merupakan tabel subtitusi, dan cv4k+i–3 merupakan (4k+i-3) byte pada urutan kriptografi yang ada.
g3 = F(g2 .cv4k) . g1
g4 = F(g3 . cv4k+1) . g2
g5 = F(g4 . cv4k+2) . g3
g6 = F(g5 . cv4k+3) . g4
Untuk G invers, [Gk]-1(w = g5¦g6) = g1¦g2, gi-2 = F(gi-1 . cv4k+i–3) . gi

IV. PERANCANGAN SISTEM

4.1. Perancangan Pengiriman *File* Melalui *Internet* (*TCP/IP*)

Pengiriman *file* teks melalui internet dengan menggunakan *protocol TCP/IP*



Gambar 4.1 Pengiriman

file teks melalui internet

Dokumen (*file teks*) yang akan dikirim berupa *plaintext* yang kemudian dienkripsi melalui perangkat lunak pengiriman *e-mail* (*e-mail client*) contohnya MS Outlook Express, Eudora Mail dan lain-lain. Setelah dienkripsi, *file* teks (*plaintext*) berubah menjadi *chipertext*, *chipertext* tersebut dikirimkan melalui protokol TCP/IP *Sender* yang selanjutnya dikirimkan ke protokol TCP/IP *Receiver*. Pada protokol TCP/IP *Receiver*, *chipertext* tersebut didekripsi menjadi *plaintext* kembali melalui perangkat lunak penerima *e-mail* (*e-mail client*).

V. IMPLEMENTASI DAN PENGUJIAN SISTEM

Implementasi ini sesuai dengan perancangan pada Bab 4 Perancangan Sistem, yaitu implementasi antarmuka (*interface*) dan akan dibahas pengujian terhadap sistem.

5.1. Antarmuka Menu Utama

Sesuai dengan Bab IV perancangan sistem, berikut adalah implementasi antarmuka menu utama.

Gambar 5.1. Implementasi Antarmuka Menu Utama

5.2. Antarmuka Enkripsi

Program enkripsi di realisasikan menggunakan bahasa pemrogramman Borland Delphi. Tampilan program Enkripsi *skipjack* dapat dilihat pada gambar sebagai berikut :
- Antarmuka Proses Enkripsi 1

Pada langkah ini dilakukan proses mengambil *file* teks dari media

penyimpanan, dengan mengklik tombol *next* maka kita harus memasukan kunci enkripsi pada antarmuka yang ditampilkan.

- Antarmuka Proses Enkripsi 2

Gambar 5.3. Implementasi Antarmuka

Proses Enkripsi 2

Pada langkah ini dilakukan proses memasukkan kata kunci untuk proses enkripsi, dengan mengklik tombol *next* maka akan ditampilkan antarmuka untuk memilah kalimat ke dalam 64 bit.

- Antarmuka Proses Enkripsi 3

Proses Enkripsi 3

Pada langkah ini dilakukan proses memilah kalimat ke dalam 64 bit yang dapat dilakukan dengan menekan tombol proses. Maka dari gambar 5.4. ini kita telah mendapatkan *file* yang berukuran habis dibagi 64 *bit* atau 8 *word* yang merupakan aturan dari algoritma skipjack, dengan mengklik tombol *next* maka akan ditampilkan antarmuka untuk melakukan proses putaran 32 kali yang di dalamnya meliputi proses Rule A, Rule B, pengecekan karakter (enter, baris baru, kosong (*nill*)) dan Permutasi.

- Antarmuka Proses Enkripsi 4

Gambar 5.5. Implementasi Antarmuka

Proses Enkripsi 4

Pada langkah ini dilakukan proses putaran 32 kali meliputi proses Rule A, Rule B, pengecekan karakter (enter, baris baru, kosong (*nill*)) dan Permutasi. Listing Program Enkripsi adalah sebagai berikut :

```
.....
..
Function ENCRYPT(plain : string; key :
string) : string;
var
j,n,i,longplain,sisip,m,longcipher:integer
;
ciphertext,chip:string;
begin
k:=key;
ciphertext:='';
longplain:=length(plain);
if (longplain mod 8) <> 0 then
begin
sisip:=8 - (longplain mod 8);
for n:=1 to sisip do
plain:=plain+'0';
end;
longplain:=length(plain);
for j:=1 to (longplain div 8) do
begin
w1[0]:=StrTo Word (plain[(((j-
1)*8)+1)]+plain[(((j-1)*8)+2)]);
w2[0]:=StrTo Word (plain[(((j-
1)*8)+3)]+plain[(((j-1)*8)+4)]);
w3[0]:=StrTo Word (plain[(((j-
1)*8)+5)]+plain[(((j-1)*8)+6)]);
w4[0]:=StrTo Word (plain[(((j-
1)*8)+7)]+plain[(((j-1)*8)+8)]);
for i:=1 to 32 do
begin
case i of
1..8 : a_proc(i);
```

```
9..16 : b_proc(i);
Text Box: Komponen dalam form enkripsiText Box: Tabel 3.1.
17..24 : a_proc(i);
25..32 : b_proc(i);
end;
end;
ciphertext:=ciphertext+WordToStr(w1[32])+WordToStr(w2[32])+WordToStr(w3[32])+WordToStr(w4[32]);
end;
longcipher:=length(ciphertext);
i:=1;
while i<longcipher+1 do
begin
if ciphertext[i] = chr(0) then
begin
chip:=copy(ciphertext,1,i-1)+'nilstr'+copy(ciphertext,i+1,longcipher-i);
ciphertext:=chip;
end
else
if ciphertext[i] = chr(10) then
begin
chip:=copy(ciphertext,1,i-1)+'notret'+copy(ciphertext,i+1,longcipher-i);
ciphertext:=chip;
end
else
if ciphertext[i] = chr(13) then
begin
chip:=copy(ciphertext,1,i-1)+'bknret'+copy(ciphertext,i+1,longcipher-i);
ciphertext:=chip;
end;
longcipher:=length(ciphertext);
inc(i);
end;
ciphertext:=ciphertext+IntToStr(sisip);
ENCRYPT:=ciphertext;
end;
.....
.
```

Listing program enkripsi ini digunakan untuk mengubah *plaintext* menjadi *chipertext* dengan cara mengubah *plaintext* menjadi 4 *word* yang diolah oleh pergantian *rule A* dan *rule B*. Jika pada *plaintext* ditemukan adanya karakter 0(*nilstring*) , 10 (*return*) dan 13 (*enter*) maka program akan otomatis mengganti karakter nol menjadi *nilstring*, karakter 10 menjadi *notret* dan karakter 13 menjadi *bknret*. Hal ini untuk mencegah terjadinya *error* pada program akibat adanya karakter 0 dan 13 pada *plaintext*.

Pada listing program enkripsi terdapat 2 macam *Rule* yang digunakan secara bergantian. Prosedur *Rule A* dan *Rule B* ini ditampilkan dalam listing program berikut :

```
.....
.
Procedure a_proc(a :integer);
begin
w1[a]:=g(w1[a-1],a) XOR w4[a-1] XOR a;
w2[a]:=g(w1[a-1],a);
w3[a]:=w2[a-1];
w4[a]:=w3[a-1];
end;
procedure b_proc(a :integer);
begin
w1[a]:=w4[a-1];
w2[a]:=g(w1[a-1],a);
w3[a]:=w1[a-1] xor w2[a-1] xor a;
w4[a]:=w3[a-1];
end;
```

Pada permulaan prosedur *Rule A* dijalankan, a pada w1[a] bernilai 1. Dalam penggunaan *Rule A* dan *Rule B* terdapat permutasi g, untuk lebih jelasnya penggunaan permutasi g ini dilampirkan dalam listing program berikut :

```
function G(inp : word; i :integer) : word;
var K0,k1,k2,k3 : byte;
A,B,C,D : word;
inpL,inpH : byte;
outL,outH : byte;
w_cely : word;
w_left : word;
w_right : word;
begin
w_cely:=inp;
w_left:=w_cely shr 8;
w_right:= (w_cely shl 8) shr 8;
inpl:=w_right;
inph:=w_left;
k0:=Ord(k[((4*(i-1)) mod 10)+1]);
k1:=Ord(k[(((4*(i-1))+1) mod 10)+1]);
```

```
k2:=Ord(k[(((4*(i-1))+2) mod 10)+1]);
k3:=Ord(k[(((4*(i-1))+3) mod 10)+1]);
A:=F[inpL XOR k0];
B:=F[inpH XOR A XOR k1];
C:=F[inpL XOR B XOR k2];
D:=F[inpH XOR A XOR C XOR k3];
outh := (inpH XOR A XOR C);
outl := (inpL XOR B XOR D);
w_cely:=(outh*256) + outl;
g:=w_cely;
end;
.....
.
```

Pada listing permutasi g terdapat proses pengolahan kunci. Kunci di atas didefinisikan sebagai k0, k1, k2 dan k3. Permutasi g ini melibatkan penggunaan tabel F yang diperoleh dari hasil pengolahan kunci tersebut .

Untuk menampilkan hasil ciphertext pada memo dilampirkan pada listing program berikut :

```
Cipher.Lines.Add(ENCRYPT(Plain.Tex
t,edPass.Text));
```

Komponen yang digunakan dalam program proses enkripsi ditampilkan pada tabel 5.1 di bawah ini.

Tabel 5.1. Komponen Antarmuka Proses Enkripsi

Object	Kegunaan
Label	Untuk penamaan suatu object
Edit	Masukan kunci dan visualisasi
	Menunjukan waktu awal sebelum program dieksekusi
	Menunjukan waktu akhir setelah program dieksekusi
	Waktu yang dihasilkan
Memo	Memasukan data awal (*plaintext*)
	Hasil akhir setelah program dieksekusi
Command Button	Proses Next (Form selanjutnya)
	Membuka *File*
	Menyimpan *File*

5.3. Antarmuka Dekripsi

Diagram alir program dekripsi dapat direalisasikan dengan menggunakan program Borland Delphi 5. Tampilan program dekripsi *Skipjack* dapat dilihat pada gambar 5.6 dan gambar 5.7.

- Antarmuka Proses Dekripsi 1

Gambar 5.6. Implementasi Antarmuka

Proses Dekripsi 1

Pada gambar 5.6. dikerjakan proses pengecekan karakter, dimana jika terdapat karakter Notret maka diganti dengan chr(10) dan karakter bknret diganti dangan karakter chr(13). dengan mengklik tombol *next* maka kita harus memasukkan kata kunci untuk proses dekripsi pada antarmuka yang ditampilkan.

- Antarmuka Proses Dekripsi 2

Gambar 5.7. Implementasi Antarmuka

Proses Dekripsi 2

Listing program dekripsi secara singkat dilampirkan di bawah ini :

```
.....
.
Function DECRYPT(cipher : string; key
: string) : string;
var j,i,longcipher,sisip:integer;
plaintext,ciphertext,chip,tmp:string;
begin
k:=key;
plaintext:='';
```

```
sisip:=StrToInt(RightStr(cipher,1));
longcipher:=length(cipher);
ciphertext:=cipher;
i:=1;
while i<longcipher+1 do
begin
tmp:=copy(ciphertext,i,6);
if tmp = 'nilstr' then
begin
chip:=copy(ciphertext,1,i-
1)+chr(0)+copy(ciphertext,i+6,longciph
er);
ciphertext:=chip;
end
else
if tmp = 'notret' then
begin
chip:=copy(ciphertext,1,i-
1)+chr(10)+copy(ciphertext,i+6,longcip
her);
ciphertext:=chip;
end
else
if tmp = 'bknret' then
begin
chip:=copy(ciphertext,1,i-
1)+chr(13)+copy(ciphertext,i+6,longcip
her);
ciphertext:=chip;
end;
longcipher:=length(ciphertext);
inc(i);
end;

cipher:=LeftStr(ciphertext,length(ciphert
ext)-1);
longcipher:=length(cipher);
for j:=1 to (longcipher div 8) do
begin
w1[32]:=StrTo(Word)(cipher[(((j-
1)*8)+1)]+cipher[(((j-1)*8)+2)]);
w2[32]:=StrTo(Word)(cipher[(((j-
1)*8)+3)]+cipher[(((j-1)*8)+4)]);
w3[32]:=StrTo(Word)(cipher[(((j-
1)*8)+5)]+cipher[(((j-1)*8)+6)]);
w4[32]:=StrTo(Word)(cipher[(((j-
1)*8)+7)]+cipher[(((j-1)*8)+8)]);
for i:=32 downto 1 do
begin
case i of
1..8 : ainv_proc(i);
9..16 : binv_proc(i);
17..24 : ainv_proc(i);
25..32 : binv_proc(i);
end;
end;
plaintext:=plaintext+WordToStr(w1[0])
+WordToStr(w2[0])+WordToStr(w3[0]
)+WordToStr(w4[0]);
end;
DECRYPT:=LeftStr(plaintext,length(pl
aintext)-sisip);
end;
```

.....

Program dekripsi ini mula-mula mencari ada atau tidaknya karakter *nilstr*, *bknret*, *notret* jika ditemukan maka program dekripsi ini akan mengubah karakter di atas menjadi 0, 13, 10. Setelah proses pencarian selesai dan karakternya telah diubah, proses selanjutnya adalah membagi *ciphertext* menjadi 4 *word*. Setelah dibagi menjadi 4 *word* kemudian diproses dengan mengunakan prosedur *Rule* A^{-1} dan *Rule* B^{-1}. Hal ini dilakukan untuk menghasilkan *plaintext*. Listing program prosedur *Rule* A^{-1} dan *Rule* B^{-1} ditampilkan di bawah ini :

.....

```
Procedure ainv_proc(a :integer);
begin
w1[a-1]:=gi(w2[a],a);
w2[a-1]:=w3[a];
w3[a-1]:=w4[a];
w4[a-1]:=w1[a] xor w2[a] xor a;
end;
Procedure binv_proc(a :integer);
begin
w1[a-1]:=gi(w2[a],a);
w2[a-1]:=gi(w2[a],a) xor w3[a] xor a;
w3[a-1]:=w4[a];
w4[a-1]:=w1[a];
end;
```

.....

Pada program dekripsi pertama-tama digunakan prosedur *Rule* B_{-1} yang kemudian dilanjutkan oleh prosedur *Rule* A^{-1} dan selanjutnya. Pada kedua *Rule* diatas terdapat permutasi G *invers*. Permutasi G invers ini menggunakan tabel F dalam proses permutasinya.

Listing program dari permutasi G *invers* ditampilkan di bawah ini :

```
.....
.
function GI(inp : word; i :integer) : word;
var K0,k1,k2,k3 : byte;
A,B,C,D : byte;
inpL,inpH : byte;
outL,outH : byte;
w_cely,w_left,w_right : word;
begin
w_cely:=inp;
w_left:=w_cely shr 8;
w_right:=(w_cely shl 8) shr 8;
inpl:=w_right;
inph:=w_left;
k0:=Ord(k[((4*(i-1)) mod 10)+1]);
k1:=Ord(k[(((4*(i-1))+1) mod 10)+1]);
k2:=Ord(k[(((4*(i-1))+2) mod 10)+1]);
k3:=Ord(k[(((4*(i-1))+3) mod 10)+1]);
A:=F[inph XOR k3];
B:=F[inpl XOR A XOR k2];
C:=F[inph XOR B XOR k1];
D:=F[inpl XOR A XOR C XOR k0];
outl := (inpl XOR A XOR C);
outh := (inph XOR B XOR D);
w_cely:=(outh*256) + outl;
gi:=w_cely;
end;
.....
.
```

Pada permutasi G *invers* terdapat proses pengolahan kunci, kunci yang digunakan pada proses dekripsi sama dengan kunci yang digunakan untuk proses dekripsi. Permutasi G invers ini juga melibatkan penggunaan tabel F yang diperoleh dari hasil pengolahan kunci tersebut . Untuk menampilkan hasil ciphertext pada memo dilampirkan pada listing program berikut :

```
.....
.
Plain.Lines.Add(DECRYPT(ciphertext,
edPass.Text));
.....
.
```

VI.Kesimpulan Dan Saran

6.1. Kesimpulan

Setelah kita melakukan penelitian dari bab awal sampai bab 5, maka dapat ditarik kesimpulan sebagai berikut :

1. Dengan sistem keamanan data, file yang mudah di enkripsi akan sulit dibaca dan terjaga kerahasiaanya
2. Dengan Algoritma *Skipjack*, data sangat aman karena algoritma ini yang paling aman dengan 32 kali putaran algoritma dibandingkan dengan algoritma yang lain sehingga data sulit untuk dibuka kerahasiaanya tanpa didekripsi.
3. Dari pengujian sistem yang dilakukan maka dapat diketahui bahwa dengan proses enkripsi-dekripsi pada algoritma skipjack dengan kata kunci yang berbeda antara sender dan receiver, *plaintext* yang sama akan menghasilkan *chipertext* yang berbeda, tetapi hasil *plaintext* tetap sama.
4. Demikian pula bila pada proses enkripsi-dekripsi pada algoritma skipjack dengan kata kunci sama antara sender dan receiver dengan *plaintext* yang berbeda maka akan dihasilkan *chipertext* yang berbeda pula.
5. Bila diinginkan informasi pada receiver dalam proses enkripsi-dekripsi pada algoritma skipjack maka diperlukan dekripsi dengan kata kunci yang sama dengan proses enkripsi pada sender.

6.2. Saran-saran

Tentunya apa yang dilakukan penulis tentang sistem ini, memiliki kekurangan-kekurangan untuk itu penulis menerima masukan dan kritikannya atas kesalahan atau kekeliruan yang penulis perbuat dalam menulis dan meneliti karya ilmiah ini. Itu semua karena kekurangan pengetahuan dan pengalaman penulis dalam membuat sebuah penelitian seperti ini.

Algoritma skipjack ini sebenarnya dapat digunakan dalam file selain file text, untuk itu penulis menyarankan untuk digunakan atau dicoba pada file selain

file text yang diterapkan pada distribusi email. Selain itu dengan algoritma ini pula ukuran file ternyata dari hasil penelitian penulis, ukuran file ada yang berkurang ada juga yang bertambah.

DAFTAR PUSTAKA

http://csrc.nist.gov/crypto toolkit/skipjack/penerbit, kota, tahun http://csrc.nist.gov/crypto Toolkit/skipjack/clarification.pdf

http://www.austinlinks.com/crypto/skipjack.review.html

http://www.faqs.org/rfcs/rfc2876.html

http://www.kremlinencrypt.com/crypto/algorithms.html

http://www.cs.technion.ac.il/~biham/Reports/SkipJack/

http://www.epic.org/crypto/clipper/skipjack_interim_review.html

http://www.mirror386.com/gnupg/contrib/do-not-use/skipjack.c

Pawling, J.(2000). Use of the KEA and SKIPJACK Algorithms in CMS rfc2876, http://www.ietf.org/rfc/

Jogiyanto, H.M, Analysis dan Desain, 1990.

Pressman, Roger S.(1997). Rekayasa Perangkat Lunak, Andi Offset Yogyakarta.

Yao Tung, Khoe. (1997), Teknologi Jaringan Intranet, Andi Offset Yogyakarta.

W. Purbo, Onno (2000), TCP/IP Standar, Desain, dan Implementasi, PT. Gramedia, Jakarta.

Subyantara, Didik (2004), Instalasi dan Konfigurasi Jaringan Microsoft Windows, PT. Elex Media Komputindo, Jakarta.