Teknik Konversi Berbagai Jenis Arsip ke Dalam bentuk Teks Terenkripsi

Dadan Ramdan Mangunpraja[1)]
1) Jurusan Teknik Informatika, STEI ITB, Bandung, email: if14087@if.itb.ac.id

Abstract *–Konversi berbagai jenis arsip ke dalam bentuk teks merupakan suatu cara agar arsip bisa disimpan dalam bentuk teks yang berisi karakter-karakter yang lazim digunakan manusia sehingga arsip bisa disimpan dalam bentuk dokumen fisik (kertas). Aplikasi berbasis teks biasa pada dasarnya mampu menerjemahkan arsip menjadi karakter-karakter teks namun dengan rentang karakter yang besar sehingga banyak memunculkan karakter-karakter yang tidak lazim.*

Makalah ini membahas bagaimana cara mengkonversi arsip menjadi teks yang berisi karakter-karakter yang lazim digunakan manusia. Proses yang ada di dalamnya meliputi pemilahan arsip tiap 5 bit untuk membatasi karakter yang muncul hanya karakter tertentu saja dengan efek pembesaran arsip sebesar 60%, kompresi menggunakan algoritma Huffman untuk meminimalkan pembesaran arsip akibat pemilahan 5 bit tersebut, serta enkripsi menggunakan Vigénere Cipher untuk mengamankan mekanisme konversi.

Kata Kunci*: Vigénere Cipher,Huffman, Konversi, Arsip Teks*

1. PENDAHULUAN

Dalam sejarah perkembangan kriptografi, pesan rahasia dalam bentuk teks merupakan bentuk pesan rahasia paling awal. Seiring perkembangan teknologi, teknik dan metode penyampaian pesan rahasia pun semakin beragam. Berbagai bentuk pesan rahasia di samping pesan teks seperti pesan citra, pesan audio, dan pesan video sudah umum digunakan. Seperti halnya pesan teks dalam menjaga kerahasiaannya, pesan non-teks juga memerlukan teknik-teknik enkripsi yang sebisa mungkin sederhana tapi sukar dipecahkan.

Dalam dunia kriptografi terdapat konsep "pengecohan", yakni menyamarkan sesuatu yang bisa menggiring kriptanalis ke dalam suatu kesalahan persepsi misalnya mengkonversi pesan rahasia jenis apapun (gambar, suara, video, dll) ke dalam bentuk teks di mana karakter-karakter yang muncul dalam teks merupakan karakter yang lazim digunakan oleh manusia. Dengan demikian hasil dari teknik konversi tersebut akan memberikan kesan bahwa pesan tersebut merupakan pesan rahasia hasil enkripsi pesan teks menggunakan algoritma sederhana (Caesar Cipher atau Vigénere Cipher), atau bisa dikatakan teknik konversi tersebut memiliki kelebihan yaitu menyamarkan jenis arsip. Jika kriptanalis tergiring pada kesalahan dengan menganggap cipherteks yang akan dipecahkan adalah hasil enkripsi teks, maka kriptanalis akan menggunakan teknik-teknik pemecahan yang hanya cocok untuk kriptanalisis teks sehingga cipherteks tidak akan bisa dipecahkan sama sekali.

Selain untuk menerapkan konsep pengecohan tadi, teknik konversi ini juga bisa digunakan untuk pesan rahasia dalam bentuk arsip (dokumen fisik). Dengan teknik konversi arsip menjadi bentuk teks, pengiriman pesan bisa dilakukan dalam bentuk dokumen fisik yang berisi karakter-karakter yang lazim digunakan manusia sehingga pesan rahasia bisa didigitalkan kembali oleh pihak yang berkepentingan secara manual (input karakter satu per satu) ataupun dengan teknologi (misalnya dengan pemindai yang mampu menerjemahkan gambar karakter) untuk kemudian dikembalikan lagi menjadi pesan semula.

Mekanisme konversi arsip menjadi bentuk teks tergolong sederhana. Arsip teks pada dasarnya bisa dihasilkan dari arsip nonteks dengan menggunakan aplikasi berbasis teks biasa seperti notepad. Aplikasi tersebut menerjemahkan arsip ke dalam bentuk teks dengan mengkonversi nilai byte-bytenya menjadi karakter sesuai aturan ASCII. Akan tetapi karena rentang nilai yang bisa dihasilkan dalam 1 byte berjumlah 256, sedangkan jumlah karakter yang lazim digunakan manusia (abjad, angka, dan beberapa karakter khusus) berjumlah jauh lebih kecil dari itu, maka pada hasil konversi tersebut besar kemungkinan terdapat karakter-karakter yang tidak bisa dipahami manusia (bisa dikatakan juga tidak tersedia dalam keyboard sebagai *input device* untuk komputer yang sudah umum digunakan). Untuk membatasi hasil konversi, beberapa aplikasi tidak mengkonversi tiap-tiap byte-nya (8 bit) menjadi 1 karakter (8 bit), tetapi kurang dari itu misalnya mengkonversi tiap 6 bit menjadi 1 karakter. Cara tersebut (mengkonversi tiap 6 bit pada arsip menjadi 8 bit) akan membatasi rentang jumlah karakter yang dihasilkan menjadi 64 karakter, sehingga bisa mengeliminasi kemunculan karakter-karakter yang tidak lazim digunakan manusia. Tapi oleh sebab itu, hasil dari konversi memberikan ukuran arsip yang lebih besar menjadi 8/6-nya.

Karena mekanisme konversi yang diterapkan tersebut tergolong sederhana, sedangkan dalam pemecahan suatu pesan rahasia selalu diasumsikan kriptanalis mengetahui mekanisme yang digunakan untuk merahasiakan pesan, maka teknik konversi ini tergolong kurang aman. Kriptanalis bisa dengan mudah mengembalikan pesan rahasia menjadi pesan semula dengan menggunakan mekanisme balikannya

Dalam makalah ini, akan dibahas mekanisme konversi dengan menggabungkan proses yang membatasi kemunculan karakter, proses minimalisasi pembesaran ukuran arsip, serta proses peningkatan keamanannya.

2. PENJELASAN ALGORITMA

2.1 Substitusi Bit-Bit Arsip Menjadi Karakter

Rentang bit yang akan digunakan dalam teknik konversi ini adalah 5 bit. Bit-bit pada arsip yang akan disubstitusi dengan pemilahan 5 bit menjadi 1 *byte* dimaksudkan agar menghasilkan jangkauan jumlah karakter sebanyak 2^5 atau 32. Jumlah tersebut cukup untuk menghasilkan karakter-karakter yang lazim digunakan orang yaitu karakter abjad ditambah karakter angka.

Langkah-langkah substitusi adalah sebagai berikut:

1. Arsip secara keseluruhan dipilah per 5 bit.
2. Ubah bilangan biner yang direpresentasikan dalam 5 bit tadi menjadi suatu bilangan desimal.
3. Substitusi bilangan desimal tadi menjadi karakter berdasarkan Tabel 1.

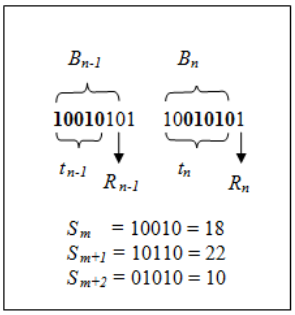
Tabel 1 Substitusi Bilangan Desimal Menjadi Karakter

Angka	Karakter	Angka	Karakter	Angka	Karakter
0	A	12	M	24	Y
1	B	13	N	25	Z
2	C	14	O	26	0
3	D	15	P	27	1
4	E	16	Q	28	2
5	F	17	R	29	3
6	G	18	S	30	4
7	H	19	T	31	5
8	I	20	U	32	6
9	J	21	V	33	7
10	K	22	W	34	8
11	L	23	X	35	9

Apabila *tool* untuk pembangunan perangkat lunak tidak memfasilitasi operasi pemilahan bit, maka untuk memudahkan operasi pemilahan bit, dibuat mekanisme agar hal tersebut bisa dilakukan dalam operasi desimal sebagai berikut:

B_n → *Byte* ke *n* dalam bentuk desimal
t_n → jarak pemilahan bit pada *Byte* ke *n*
R_n → barisan bit sisa pemilahan dalam bentuk desimal
S_m → bilangan desimal ke *m* untuk disubstitusi

di mana variabel-variabel tersebut lebih jelasnya bisa dilihat pada Gambar 1di bawah ini:



Gambar 1 Variabel-Variabel Untuk Proses Pemilahan *Byte*

Jika $t_{n-1} \leq 6$, yang berarti dalam satu *byte* bisa dibentuk dua bilangan desimal, maka:

$t_n = 5-(8-t_{n-1})+5$
$= 2 + t_{n-1}$

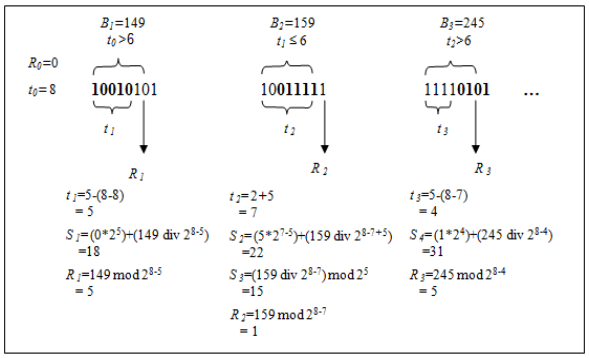
serta $S_m = (R_{n-1}) * 2^{tn-5} + (B_n \text{ div } 2^{8-tn+5})$
dan $S_{m+1} = (B_n \text{ div } 2^{8-tn}) \bmod 2^5$

Jika $t_{n-1} > 6$, yang berarti dalam satu *byte* hanya bisa dibentuk satu bilangan desimal, maka:

$S_m = (R_{n-1}) * 2^{tn} + (B_n \text{ div } 2^{8-tn})$
$t_n = 5-(8-t_{n-1})$
$= t_{n-1} -3$

Untuk keduanya berlaku $R_n = B_n \bmod 2^{8-tn}$

Contoh dari mekanisme substitusi yang menerapkan aturan tersebut bisa dilihat pada Gambar 2.



Gambar 2 Contoh Mekanisme Pemilahan *Byte*

2.2 Penanggulangan Peningkatan Ukuran Arsip

Seperti yang telah dijelaskan pada subbab 2.1, substitusi 5 bit menjadi 1 *byte* akan mengakibatkan peningkatan ukuran arsip sebesar 60% (3/5). Untuk menanggulangi masalah tersebut digunakan algoritma untuk mengkompresi arsip yaitu algoritma Huffman.

Setelah proses substitusi dilakukan, hasilnya akan dikompresi. Agar efek dari kompresi arsip oleh algoritma Huffman tidak akan menghasilkan banyak karakter yang tidak lazim, pengkodean pada algoritma Huffman dilakukan bukan untuk menhasilkan 1 *byte*, tapi untuk menghasilkan 5

bit. Mekanisme substitusi pada awal proses diharapkan bisa mengoptimalkan proses kompresi, karena dari jangkauan 256 karakter yang disimpan menjadi hanya 32 karakter saja. Setelah dilakukan pengujian akan dianalasis sejauh mana proses kompresi optimal dalam meminimalkan pembesaran ukuran arsip dalam proses konversi.

Oleh karena operasi substitusi menggunakan bilangan desimal baik sebelum kompresi maupun sesudah kompresi, maka untuk memudahkan operasi tersebut, proses kompresi pun dilakukan dalam operasi desimal. Berdasarkan dasar teori, kompresi dilakukan dengan mengkodekan string biner karakter semula menjadi string biner baru berdasarkan kamus Huffman yang dibentuk. Mekanisme kompresi dalam operasi desimal dilakukan sebagai berikut:

B_n → String biner pada kamus Huffman untuk karakter ke-n

D_n → Nilai desimal untuk string biner

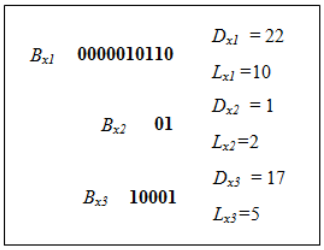
L_n → panjang string biner

t_m → jumlah bit yang akan dibaca untuk byte ke-m

R_m → barisan bit sisa pemilahan 5 bit untuk *byte* ke-m dalam bentuk desimal

S_m → bilangan desimal ke m hasil kompresi

Kompresi menggunakan operasi desimal membutuhkan string biner yang direpresentasikan dalam bilangan desimal juga serta informasi panjang stringnya. Variabel-variabel tersebut bisa dilihat pada Gambar 3.



Gambar 3 Variabel-Variabel Untuk Proses Kompresi

Mekanisme operasi selanjutnya mirip dengan mekanisme pemilahan *byte*. Karakter yang dikodekan dalam string biner dipilah-pilah sepanjang 5 karakter untuk dijadikan elemen array hasil (bilangan desimal 5 bit). Untuk setiap iterasi pengkodean karakter, hitung

$R_m = R_m * 2L_n + D_n$

$t_m = t_m + L_n$

jika nilai $t_m > 5$, terusakan iterasi untuk karakter berikutnya. Jika t_m sudah ≥ 5, yang berarti sudah bisa dibentuk sebuah *byte*, hitung

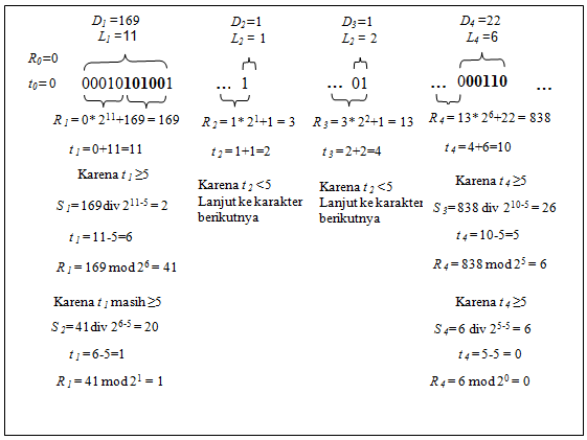
$t_m = t_m - 5$

$S_m = R_m \text{ div } 2t_m$

$R_m = R_m \text{ mod } 2t_m$

kemudian ulangi hingga $t_m<5$ untuk bisa memproses karakter berikutnya.

Contoh dari mekanisme kompresi yang menerapkan aturan tersebut bisa dilihat pada Gambar 4.



Gambar 4 Contoh Penggunaan Mekanisme Kompresi

2.3 Analisis Pemilihan Vigénere Cipher untuk Algoritma Enkripsi

Seperti telah dijelaskan dalam latar belakang pada bab Pendahuluan, walaupun mekanisme konversi akan menghasilkan arsip yang berbeda dan tidak bisa dipahami dibandingkan arsip sebelumnya, tetapi dalam penyandian pesan selalu diasumsikan bahwa seseorang yang berusaha memecahkan pesan rahasia (kriptanalis) telah mengetahui algoritma atau mekanisme penyandian yang diterapkan. Berdasarkan hal tersebut, jika mekanisme penyandian arsip hanya dilakukan sampai proses konversi atau kompresi saja, maka kriptanalis akan dengan mudah mengubah pesan rahasia menjadi pesan semula dengan melakukan balikan/invers terhadap mekanisme yang diterapkan.

Faktor penting dalam penyandian sebuah pesan rahasia adalah adanya sesuatu yang tidak diketahui oleh kriptanalis, yaitu kunci untuk memecahkan pesan. Penerapan algoritma enkripsi yang menggunakan suatu kunci diperlukan agar mekanisme penyandian dengan konversi tidak lagi mudah untuk dipecahkan.

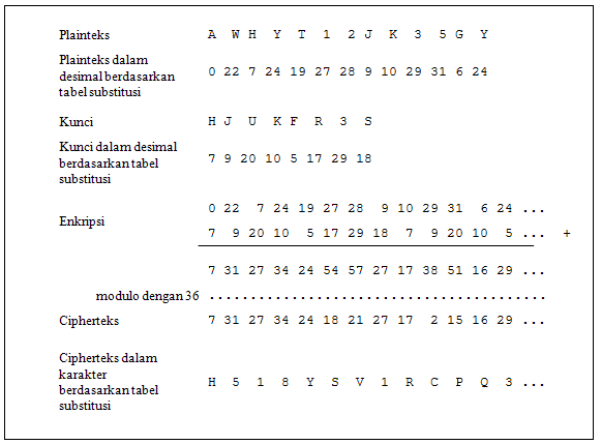
Dari sekian banyak algoritma untuk mengenkripsi, tidak sembarang algoritma enkripsi bisa diterapkan. Algoritma enkripsi yang bisa digunakan adalah algoritma yang tidak mengubah kondisi pesan di mana karakter-karakter yang muncul merupakan karakter-karakter yang lazim digunakan oleh manusia. Algoritma enkripsi modern yang secara umum orientasi operasinya menggunakan mode bit tidak bisa digunakan karena akan memunculkan karakter-karakter yang tidak lazim sehingga merusak spesifikasi pesan yang sudah ditentukan.

Berdasarkan hal tersebut, algoritma enkripsi yang bisa digunakan adalah algoritma kriptografi klasik yang orientasi operasinya menggunakan mode karakter seperti cipher substitusi atau cipher transposisi. Algoritma kriptografi klasik yang akan digunakan adalah cipher substitusi dan dipilih *Vigénere Cipher*. *Cipher* transposisi hanya menggunakan suatu angka sebagai kunci di mana angka tersebut merepresentasikan selang jumlah karakter yang ditransposisikan, sehingga terlalu mudah untuk dipecahkan secara brute force, selain itu *Cipher* transposisi hanya mengacak posisi karakter sehingga rentang karakter yang dihasilkan oleh proses substitusi sebanyak 32 tidak bisa bertambah menjadi 36. Hal tersebut berbeda dengan *Vigénere Cipher* yang variabilitas kuncinya jauh lebih tinggi dan *Vigénere Cipher* juga memungkinkan terjadinya penyebaran karakter dari hasil proses substitusi yang hanya 32 karakter menjadi 36 karakter yang diinginkan (26 karakter abjad ditambah 10 karakter angka).

Algoritma enkripsi *Vigénere Cipher* telah ditemukan metode pemecahannya yaitu metode kasiski yang memanfaatkan frekuensi kemunculan karakter secara umum pada penggunaan kalimat atau kata. Meskipun demikian, penggunaan *Vigénere Cipher* pada teknik konversi ini belum bisa dikatakan tidak aman, karena arsip yang dienkripsi bukan merupakan arsip teks sehingga metode kasiski untuk teknik konversi ini belum tentu bisa digunakan. Di luar hal tersebut, penggunaan algoritma enkripsi *Vigénere Cipher* pada teknik konversi ini dilakukan untuk menguji apakah dengan algoritma enkripsi yang lemah bisa membuat teknik konversi ini aman atau tidak. Jika aman, maka penggunaan algoritma enkripsi yang lebih kuat akan memberikan hasil yang lebih baik pula.

Pada implementasinya, Bujursangkar *Vigénere* harus direpresentasikan sebagai variabel global dalam bentuk matriks $V_{36,36}$. Ada cara yang lebih efektif agar program tidak memakan memori yang tidak perlu, yaitu dengan memanfaatkan variabel tabel substitusi. Pada tabel substitusi terdapat indeks yang pada dasarnya juga mereprentasikan nilai desimal dari karakter-karakternya (A=0, B=1, C=2, …). Dengan menggunakan indeks tersebut, cipherteks dapat diperoleh dengan menjumlahkan indeks karakter pada plainteks dan indeks karakter pada kunci kemudian dimodulo dengan 36. Untuk lebih jelasnya, bisa dilihat pada contoh pada Gambar III-9.

Pada Gambar 5 yang menunjukkan contoh penggunaan *Vigénere Cipher* dengan memanfaatkan Tabel Substitusi, plainteks dan kunci diubah dulu menjadi suatu bilangan desimal yang merupakan indeks karakter tersebut dari tabel substitusi. Enkripsi dilakukan dengan menjumlahkan plainteks dengan kunci yang sudah direpresentasikan dalam bentuk nilai desimalnya berdasarkan tabel substitusi, kemudian dimodulo dengan 36.



Gambar 5 Penggunaan Mekanisme Enkripsi

Bilangan desimal hasil dari modulo tersebut diubah kembali menjadi karakter di mana nilai tersebut merupakan indeks karakter yang dicari pada tabel substitusi. Terdapat dua cara untuk mendapatkan indeks dari karakter pada proses enkripsi, yaitu:

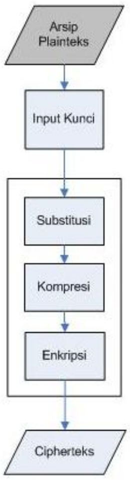
1. *Exhaustive search*
   Pada cara ini indeks untuk proses enkripsi dan dekripsi didapatkan dengan membandingkan satu-persatu karakter yang dicari dengan karakter pada tiap elemen tabel substitusi sampai ditemukan karakter yang sama. *Exhaustive search* tidak akan terlalu menimbulkan masalah jika dilakukan untuk data yang sedikit. Jika datanya banyak, waktu yang harus dibuang untuk proses pencarian dengan cara ini akan terasa signifikan.
2. Memanfaatkan tabel ASCII
   Pada cara ini tiap karakter diambil nilai ASCIInya, kemudian nilai tersebut disesuaikan dengan tabel substitusi. Data nilai ASCII untuk karakter abjad A sampai Z secara berurutan adalah dari 65 hingga 90 dan untuk karakter angka dari 0 sampai 9 secara berurutan adalah dari 48 hingga 57. Pada tabel substitusi, karakter abjad secara berurutan berada pada indeks 0 sampai 25 dan untuk karakter angka pada indeks 26 sampai 35. Dengan demikian nilai karakter abjad pada tabel substitusi bisa diperoleh dari nilai ASCIInya dikurangi dengan 65, dan untuk karakter angka bisa diperoleh dari nilai ASCII-nya dikurangi dengan 22.

Dari kedua cara tersebut terlihat bahwa cara kedua akan lebih efektif dari cara pertama karena menghindari prosedur pencarian. Penggunaan prosedur pencarian akan memakan banyak waktu apabila jumlah karakter dalam satu arsip yang

harus diproses sangat banyak.

2.4 Skema Global Algoritma

Berdasarkan analisis-analisis yang sudah dilakukan pada subbab-subbab sebelumnya, skema global algoritma untuk proses konversi terdiri dari proses substitusi, kompresi, dan enkripsi. Skema global untuk aplikasi ini bisa dilihat pada Gambar 6.



Gambar 6 Skema Global Algoritma

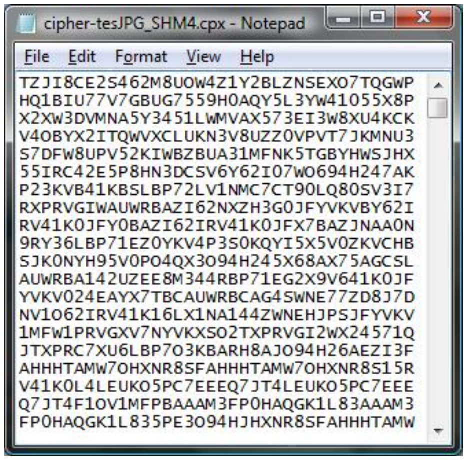
Skema global pada Gambar 6 memperlihatkan urutan proses untuk mekanisme konversi. Untuk mekanisme konversi, prosesnya dilakukan dengan menggabungkan proses secara berurutan mulai dari substitusi, kompresi, dan enkripsi dengan memasukan arsip yang akan diproses dan kunci simetris pada awal proses.

3. HASIL DAN PEMBAHASAN

3.1 Contoh Pengujian

Gambar 7 merupakan contoh arsip yang dikonversi. Setelah dikonversi menggunakan kunci kriptografi, potongan cipherteks yang dihasilkan bisa terlihat pada Gambar 8.

Gambar 7 Arsip Citra tesJPG_SHM4.jpg Sebelum Dikonversi

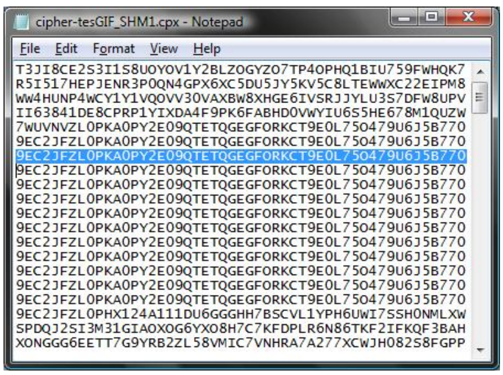


Gambar 8 Arsip Citra tesJPG_SHM4.jpg Setelah Dikonversi

3.2 Data Hasil Pengujian

1. Pada kebanyakan arsip proses kompresi yang dilakukan setelah pemilahan *byte* akan menghasilkan karakter-karakter dengan frekuensi terbanyak ‘5’, ’4’, ’3’, ’2’, ’1’ hampir secara berurutan. Hal tersebut terjadi karena untuk arsip yang penyebaran *byte* nya merata (atau sudah terkompresi sejak awal), penyebaran karakternya relatif merata sehingga pohon Huffman yang dibentuk merupakan pohon biner yang seimbang, di mana karakter paling sering muncul secara berurutan akan dikodekan menjadi string ‘11111’, ’11110’, ’11101’, ’11100’, ‘11011’ dan seterusnya yang tidak lain adalah karakter ‘5’, ‘4’, ‘3’, ‘2’, ‘1’. Hal ini akan menjadi peluang bagi kriptanalis untuk memecahkan kunci dengan metode kasiski. Pemecahan kuncinya dilakukan terhadap cipherteks cipher-tesGIF_SHM1.cpx adalah sebagai berikut:

   Pada Gambar 9 yang merupakan cipherteks cipher-tesGIF_SHM1.cpx bisa dilihat dengan mudah adanya perulangan *string* `9EC2JFZL0PKA0PY2E09QTETQGEGFORKCT9E0L750479U6J5B770` dengan panjang 51 karakter.



Gambar 9 Contoh Perulangan *String* Pada Cipherteks

Kunci `kriptografi` dalam barisan desimal berdasarkan spesifikasi aplikasi adalah [13 13 25 6 18 28 3 20 13 29 19 23 4 24 11 6 13] dengan panjang 17 karakter. Jika diasumsikan kriptanalis berhasil mengetahui panjang kunci dengan mudah yaitu 17 (51 merupakan kelipatan 17), maka kriptanalis bisa menerapkan metode Kasiski untuk memecahkan kunci.

Ada hal yang berbeda dengan penerapan metode Kasiski antara enkripsi untuk teks biasa dengan teks hasil konversi. Pada enkripsi teks normal karakter paling sering muncul (untuk penggunaan kalimat pada bahasa Inggris) adalah 'e' dengan frekuensi kemunculan 13% dan nilainya terpaut cukup jauh dengan karakter paling sering muncul berikutnya. Pada teks hasil konversi, kemunculan karakter paling tinggi sebelum enkripsi (yaitu setelah kompresi) rata-rata ditempati karakter '5' namun dengan frekuensi kemunculan jarang melebihi 5% dan tidak terlalu terpaut jauh dengan karakter paling sering muncul berikutnya. Jika dianggap 5 karakter paling sering muncul cukup untuk pengujian, maka kriptanalis berhasil mengurangi kemungkinan kunci dari 173^6 menjadi hanya 17^5.

2. Efektifitas kompresi bisa dilihat dari seberapa besar perubahan arsip yang terjadi. Pada bab analisis dijelaskan pemilahan byte tiap 5 bit akan mengakibatkan pembesaran ukuran arsip sebesar 60%, maka untuk menyimpulkan kompresi efektif, perubahan ukuran arsip harus kurang dari 60%. Berdasarkan data hasil, persentase perubahan arsip berkisar antara 50% sampai 60%. Dengan demikian efektifitas kompresi rata-rata kurang dari 10%, dan untuk arsip yang dari awal sudah terkompresi (misalnya arsip citra berformat jpg) sudah tidak bisa dikompresi lagi (perubahannya tetap 60%). Hal tersebut dikarenakan pohon biner yang dibentuk seimbang. Dengan demikian efektifitas kompresi tidak terlalu besar dalam menanggulangi pembesaran arsip. Akan tetapi kompresi terlihat signifikan untuk arsip dokumen teks (kecuali dokumen dengan format pdf) di mana rata-rata berhasil mengurangi efek pembesaran hingga kurang dari 50%, bahkan ada anomali arsip hasil konversi lebih kecil dari arsip semula.

4. KESIMPULAN

1. Secara statistik, analisis frekuensi kemunculan karakter terhadap berbagai arsip yang sudah mengalami tahapan proses konversi sebelum tahap enkripsi memberikan adanya kemunculan karakter yang menjadi karakter paling banyak muncul untuk sebagian besar arsip uji, hal tersebut mengindikasikan ada sedikit celah untuk membuat metode Kasiski mungkin untuk dilakukan untuk kebanyakan arsip. Akan tetapi hal tersebut namun butuh pengujian lebih lanjut apakah metode Kasiski benar-benar bisa memecahkan kunci atau tidak karena frekuensi distribusi karakter sebelum proses enkripsi semuanya merata (tidak seperti frekuensi kemunculan huruf pada penggunaan kalimat dalam bahasa Inggris).
2. Perbandingan persentase pembesaran ukuran arsip antara perkiraan pembesaran untuk proses konversi tanpa kompresi dengan hasil pengujian proses konversi yang mengimplementasikan proses kompresi tidak terlalu jauh berbeda. Dengan demikian pada dasarnya pembesaran ukuran arsip sulit dihindari untuk proses mengkonversi *byte-byte* karakter pesan menjadi *byte-byte* karakter yang lazim digunakan manusia. Namun proses kompresi yang diterapkan cukup berguna dalam mengacak kemunculan karakter dan menjadikan penyebaran karakter sebelum dienkripsi lebih merata.

DAFTAR REFERENSI

[1] Munir, Rinaldi, "*Diktat Kuliah IF15054, Kriptografi*", 2006, Bandung

[2] Mangunpraja, Dadan R. *Konversi Citra ke dalam Bentuk Teks Terenkripsi dengan Memanfaatkan Cipher Abjad Majemuk* Departemen Teknik Informatika, Institut Teknologi Bandung. *http://informatika.org/~rinaldi/Kriptografi/20072008/Makalah1/MakalahIF5054-2007-A-073.pdf* . Diakses terakhir tanggal 1 Februari 2008

[3]*makcoder.sourceforge.net/demo/vigenere.php.* Diakses terakhir tanggal 15 Mei 2008

[4]*http://cs.colgate.edu/faculty/nevison/Core139Web/tools/kasiski.html.*Diakses terakhir tanggal 15 Mei 2008.

[5] *http://www.pgp.org*. Diakses terakhir tanggal 20 Maret 2008.