Peancangan Algoritma Kriptografi Huffman-Monoalphabetic

Satrio Ajie Wijaya [1)]

1) Jurusan Teknik Informatika ITB, Bandung , email: if14092@students.id.itb.ac.id

Abstract *– Ada beberapa jenis algoritma monoalphabetic yang telah dikenal diantaranya Caesar chipper. Dalam makalah ini akan dibahas mengenai algoritma kriptografi Huffman-Monoalphabetic yang memanfaatakan kode Huffman sebagai key dalam melakukan enkripsi dan dekripsi.Disamping itu juga akan dibahas mengenai kekurangan dan kelebihan yang dimiliki oleh algoritma kriptografi ini serta analisis terhadap ketanguhan algoritma kriptografi ini dalam menghadapi serangan kriptoanalisis.*

Kata Key*: Kode Huffman, Monoalphabetic, chipertex, plaintext, enkripsi, dekripsi*

1. PENDAHULUAN

Kode Huffman dikenal sebagi kode yang digunakan untuk melakukan proses kompresi data. Kode ini dibentuk dengan memanfaatkan frekuensi kemunculan karakter pada berkas yang ingin dikompresi. Kode yang diperoleh ialah berupa bilangan biner yang kemudian diubah ke dalam karakter yang bersesuaian dengan bilangan biner tersebut.

Algoritma Kriptografi Huffman-Monoalphabetic merupakan suatu algoritma monoalphabetic yang memanfaatkan kode Huffman sebagai pembentuk *key* untuk melakukan enkripsi. Yang membedakan antara algoritma kriptografi monoalphabetic lainnya dengan algoritma kriptografi Huffman-monoalphabetic adalah subtitusi karakter dilakukan dengan memanfaatkan kode Huffman yang dibentuk dari plaintex. Kode Huffman tersebut kemudian digunakan sebagai *key* dalam algoritma ini. Pada algoritma ini, pensubtitusian huruf dilakukan bukan dengan memanfaatkan karakter, melainkan dengan menggunakan bilangan biner yang diperoleh dari kode Huffman.

Ada beberapa karakteristik dalam algoritma kriptografi Huffman-Monoalphabetic

1. Jika pada beberapa algoritma monoalphabetic lainnya karakter tanda baca dan spasi tidak ikut dienkripsi, pada algoritma Huffman-Monoalphabetic karakter berupa spasi dan tanda baca ikut dienkripsi dengan tujuan untuk mempersulit dalam melakukan kriptoanalisis.
2. Huruf besar dan huruf kecil daru sutu huruf yang sama akan dikodekan menjadi suatu kde yang sama
3. *Chipertext* yang terbentuk berupa deretan bilangan biner
4. *Key* yang digunakan adalah Huffman code yang dibentuk dari berkas *plaintext*
5. *Key* hanya digunakan untuk mengenkripsi atu berkas *plaintext* saja dan tidak dapat digunakan untuk berkas *plaintext* lainnya. Perubahan kecil yang terjadi pada berkas *plaintext* seperti penghilangan tanda baca atau spasi dapat menyebabkan perubahan pada *key* yang digunakan.

2. ALGORITMA KRIPTOGRAFI HUFFMAN-MONOALPHABETIC

2.1. Algoitma Enkripsi

Dalam melakukan pengenkripsian *plaintext* ada beberpa tahapan yang digunkan :

1. Melakukan pembentukan kode Huffman dari berkas *plaintext* yaitu dengan tahapan :
    a. Lakukan pembacaan terhadap karakter yang terdapat pada berkas *plaintext* untuk menghitung frekuensi kemunculan setiap karakter. Setiap karakter penyusu dinyatakan sebagai suatu pohon bersimpul tunggal. Setiap simpul kemudian di-*assign* dengan frekuensi kemuncula karakter tersebut.
    b. Gabungkan dua buah pohon yang mempunyai frekuensi terkecil pada sebuah akar. Dalm melakukan penggabungan ini, anak sebelah kiri dari akar di-*assign* dengan pohon yang memiliki frekuensi yang lebih kecil dan anak yang di sebelah kanan akar di-*assign* dengan pohon yang memilki frekuensi yang lebi besar . Akar memiliki frekuensi yang merupakan jumlah dari frekuensi dua buah pohon penyusunnya.
    c. Ulangi langkah pada point b hingga tersisa hanya satu buah pohon biner
    d. *Assign* tiap kaki kiri yang terdapat pada pohon dengan angka 0 dan kaki kanan yang terdapat pada pohon dengan nilai 1. Nilai-nilai inilah yang kemudian digunakan unuk membentuk kode Huffman.
    e. Kode Huffman diperoleh dengan menelususri pohon mulai dari akar sampai ditemukan daun yang berisi karakter yang ingin dicari kode

Huffman nya . Nilai 0 dan 1 yang diperoleh dengan menelusuri jalur mulai dari akar hinga daun tersebut disusun dengan uruan nilai yang terdapat pada kaki akar diletakkan pada urutan paling kiri dan dilanjutkan dengan nilai yang terdapat pada jalur yang dilalui dan nilaipaling kanan merupakan nilai yang terdapat pada kaki yang menghubungkan daun yang berisi karakter yang dimaksud dengan node di atasnya.

2. Kode Huffman yang telah dibentuk tersebut kemudian dijadikan sebagai *key* dalam proses enkripsi ini
3. Berdasarkan kode Huffman tersebut kemudian dilakukan pensubtitusian karakter yang terdapat pada *plaintext* dengan karakter yang bersesuaian pada kode Huffman yang dibentuk dari berkas *plaintext* tersebut
4. Berkas hasil pensubtitusian tersebut kemudian menjadi berkas *chipertext*

2.2. Algoritma Dekripsi

Dekripsi dilakukan dengan melakukan subtitssi terhadap deretan bilangan biner yang terdapat pada *chipertext* dengan karakter yang bersesuaian yang tedapat pada *key*. *Key* yang digunakan adalah kode Huffman yang dibentuk dari *plaintext*. Hasil subtitusi ini kemudian menjadi berkas paintext.

3. CONTOH KASUS

3.1. Enkripsi

Terdapat suatu berkas *plaintext*

Tabel 1. *Plaintext*

There is more supplementary material that we want to offer to the instructor, but we have decided to do it through the medium of the world wide web rather than through a CD or printed instructor' s manual. The idea is that this solution manual contains the material that must be kept secret from students, but the Web site contains material can be updated and added to more timely fashion. The address for the web site is http: //aima.cs.berkley.edu.

Langkah awal yang dilakukan adalah melakukan pembentukan kode Huffman yang digunakan sebagai *key* dalam melakukan enkripsi.

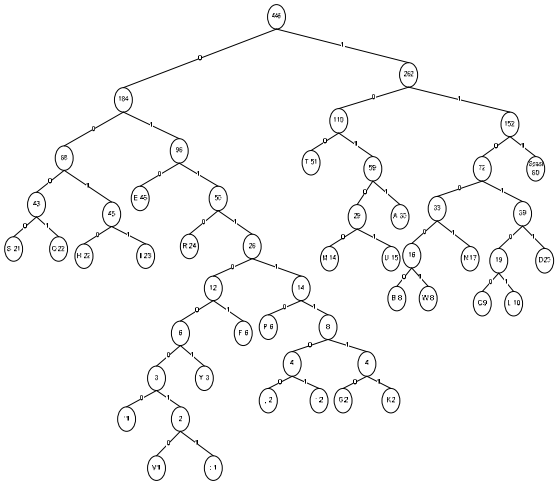
Dengan melakukan perhitungan terhadap frekuensi kemunculan karakter pada *plaintext* maka diperoleh tabel frekuensi karakter

Tabel 2. Tabel Frekuensi Kemunculan

karakter	frekuensi kemunculan
a	30
b	8
c	9
d	20
e	46
f	6
g	2
h	22
i	23
j	0
k	2
l	10
m	14
n	17
o	22
p	6
q	0
r	24
s	21
t	51
u	15
v	1
w	8
x	0
y	3
z	0
spasi	80
,	2
.	6
:	1
/	2
‘	1

Berdasarkan tabel frekeuensi karakter diatas maka dapat dibentuk pohon Huffman

Gambar 1 Pohon Huffman



Dengan memanfaatkan pohon Huffman yang telah dibentuk maka dapat diperoleh kode Huffman. Kode Huffman ini diunakan sebagai *key* dalam algoritma enkripsi ini

Tabel 3. Tabel Kode Huffman

karakter	Kode Huffman
a	1011
b	110000
c	110100
d	11011
e	010
f	011101
g	01111110
h	0010
i	0011
k	01111111
l	110101
m	10100
n	11001
o	0001
p	011110
r	0110
s	0000
t	100
u	10101
v	011100010
w	110001
y	0111001
spasi	111
,	01111100
.	0111101
:	011100011
/	01111101
‘	01110000

Terakhir, seluruh karakter yang terdapat di berkas *plaintext* disubtitusi dengan deretan bilangan biner yang bersesuaian berdasarkan kode Huffman yang terbentuk

Tabel 4. *Chipertext*

100001001001100101110011000011101000001011
001011100001010101111001111011010101010100
101100110010110110011100111110100101110001 0
011000111011110101111111100001010111001111 1
000101011111000110111100110011110000011110 0
010111010111010100110111100000111110000100 1
011100111100100001000110101011101001000001 0
110011111001111100001010110011111100010101 11
001010110111000100101111101101011010000111 1
011010110111111000001111111011000111100111 00
111100001001100001101010111111000101111000 0
100101111010001011011001110101101001110001 0
111011111000010010111110001000101101101011 1
011111110001001111011010111110001010110000 1
110110101110000100100110111100001010111001 1
111100001001100001101010111111000101111011 1
111010011011111000101101110111100110001111 0
001100010110111110011110010000100011010101 1
1010010000010110’11100001111010010111100110 1
011011110101011110111110000100101110011110 1
101010111110011000011110000101011100111100 0
010001100001110000000111010110101100001100 0
111001111101001011110011010110111101011111 1
010000011100110010110011111001000011110000 10
010111101001011100010011000111011110101111 1
000010101110011110100101010000100111110000 0
101110111111101001111010011100000101101000 1
100101001110111010110000110100111000010010 1
011101101011001100000001111100111111000010 10
110011110000100101111100010101100001110000 0
011100010111110100000111001100101100111001 1
000011110100101110001001100011101111010111 1
110100101111001111110000010111101010111011 1
011101110001011011111101111001110111111011 1
101111011010110111111000001111101000001011 0
010111100001110100010110101011100111101110 1
101100000010001100011100101111011111000010 0
101111011110111101101100100000000011101110 1
000101101111000010010111110001010110000111 0
000001110001011100110000111001010010000111 0
011100011111011111010111110110110010011101 0
110111101110100000001111011100000100110011 1
111111010101001110010111101010110111010101 1
1101

Hasil subtitusi ini kemudian menjadi *chipertext*.

3.2. Dekripsi

Untuk melakukan dekripsi dibutuhkan *key* yang berupa kode Huffman yang dibentuk dari berkas *plaintext*. Dengan memanfaatkan kode ini kemudian dilakukan subtitui balik terhadap berkas cipertex dengan memngunakan *key* tersebut.

Tabel 5. *Chipertext*

```
1000010010011001011100110000111101000001011
0010111000010101011110011110110101010101000
1011001100101101100111001111101001011100010
0110001110111101011111111000010101110011111
0001010111110001101111001100111100000111100
0101110101110101001101111000001111100001001
0111001111001000010001101010111010010000010
1100111110011111000010101100111110001010111
0010101101110001001011111011010110100001111
0110101101111110000011111101100011110011100
1111000010011000011010101111110001011110000
1001011110100010110110011101011010011100010
1110111110000100101111100010001011011010111
0111111100010011110110101111100010101100001
1101101011100001001001101111000010101111001
1111000010011000011010101111110001011110111
1111010011011111000101101110111100110001111
0011000101101111100111100100001000110101011
1010010000010110’1110000111101001011110011011
0110111101010111101111100001001011100111101
1010101111100110000111100001010111001111000
0100011000011100000001110101101011000011000
1110011111010010111100110101101111010111111
0100000111001100101100111100100001111000010
0101111010010111000100110001110111101011111
0000101011100111101001010100001001111100000
1011101111111010011110100111000001011010001
1001010011101110101100001101001110000100101
0111011010110011000000011111001111100001010
1100111100001001011111000101010110000011100000
0111000101111101000001110011001011001111001
0000111101001011100010011000111011110101111
1101001011110011111100000101111010101111011
0111011100010110111111011110011101111110111
1011110110101101111110000011111010000010110
0101111000011101000101101010111001111011101
1011000000100011000111001011110111110000100
1011110111101111011011001000000000111011101
0001011011110000100101111100010101100001110
0000011100010111001100001110010100100011110
0111000111110111101011111011011001110100 10
1101111011101000000011110111000001001100111
1111110101010011100101111010101101110101011
1101
```

Sehingga diperoleh berkas plaintest kemabali

Tabel 6. *Plaintext* Hasil Dekripsi

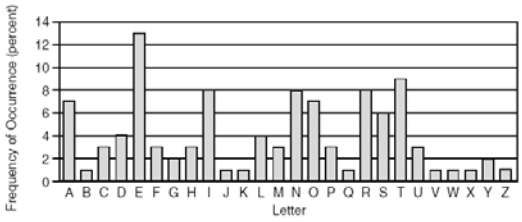
There is more supplementary material that we want to offer to the instructor, but we have decided to do it through the medium of the world wide web rather than through a CD or printed instructor' s manual. The idea is that this solution manual contains the material that must be kept secret from students, but the Web site contains material can be updated and added to more timely fashion. The address for the web site is http: //aima.cs.berkley.edu.

4. KRIPTOANALISIS ALGORITMA HUFFMAN-MONOALPHABETIC

Ada beberapa keunggulan yang dimiliki oleh algoritma Huffman-monoalphabetic jika dibandingkan dengan

1. Karena enkripsi dilakukukan terhadap seluruh karakter, maka kriptoanalisis terhadap berkas *chipertext* akan lebih sulit. Hal ini karena adanya perubahan kemunculan karakter secara umum. Tabel frekuensi kemunculan huruf secara umum tidak dapat digunakan karena karakter yang memiliki kemungkinan paling sering muncul (dalam *plaintext* berbahas Inggris) bukan lagi karakter “e” melainkan karakter spasi.

Gambar 2 Tabel frekuensi kemunculan huruf



2. *Key* yang digunakan berbeda untuk tiap berkas *plaintext* sehingga penggunaan *key* untuk melakukan dekripsi terhadap *chipertext* yang berbeda akan menghasilkan berkas *plaintext* yang tidak memiliki makna.
3. Setiap karakter akan disubtitusi dengan panjang bilangan biner yang berbeda-beda sehingga akan mempersulit ketika kriptianalis melakukan analisis terhadap panjang karakter pensubtitusi

Disamping keunggulan yang dimiliki ada beberpa kekurangan yang dimiliki oleh algoritma Huffman-Monoalphabetic yaitu panjang *chipertext* yang dihasilkan akan lebih panjang dari panjang *plaintext*. Disamping itu, pendistribusian *key* harus dilakukan dengan hati-hati agar *key* tidak jatuh ke tangan ornag yang salah

Serangan kriptoanalisis dengan menggunakan metoda

terkaan hampir tidak mungkin dilakukan. Hal ini dikarenakan banyak kemungkinan *key* yang mungkin adalah tak terbatas karena setiap karakter dan tanda baca yang terdapat di dalam *plaintext* akan ikut di enkripsi. Jadi, jumlah kemungkinan *key* adalah ∞.

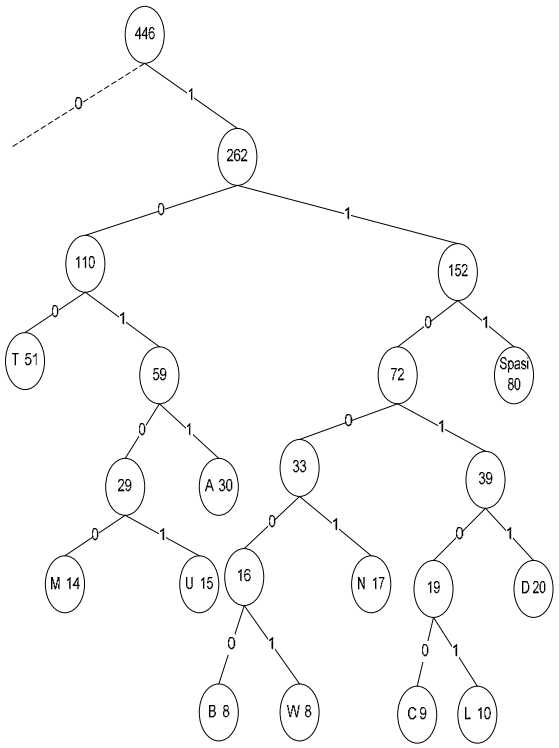
Dikarenakan setiap karakter dan tanda baca yang terdapat pada *plaintext* akan dienkripsi menjadi bilangan biner yang memilki panjang yang berbeda-beda maka akan mempersulit dalam menentukan panjang hasil subtitusi untuk masing- masing karakter. Dalam satu *chipertext* terdapat kemungkinan panjang dereten bilangan biner yang mengsubtitusi satu karekter pada *plaintext* sebanyak jumlah karakter didalam berkas *plaintext*. Namun, jumlah kemungkinan dapat direduksi dengan memanfaatkan sifat dari pohon Huffman. Untuk suatu berkas *chipertext* yang mengandung lebih dari 1 karakter, maka kemungkinan panjang bilangan biner tidak lebih dari setengah jumlah karakter yang terdapat pada *chipertext*. Semakin panjang karakter yang terdapat pada berkas *chipertext* maka semakin pendek panjang kemungkinan deretan bilangan biner yang digunakan untuk mensubtitusi satu karakter pada *plaintext*.

Tabel frekuensi kemunculan huruf untuk *plaintext* berbahasa inggris (gambar 2) tidak dapat digunakan untuk melakukan kriptoanalisis terhadap *chipertext* hal ini dikarenakan table frekuensi tersebut hanya memperhatikan kemunculan huruf saja sedangkan pada algoritma kriptografi Huffman-Monoalphabetic karakter bukan huruf seperti tanda baca dan karakter yang lainnya akan ikut di enkripsi.

Serangan yang mungkin dilakukan terhadap algoritma Huffman-Monoalphabetic adalah dengan memanfaatkan karakteristik dari pohon Huffman. Sebagian deretan bilangan biner akan mengalami pengulangan pada awal derertan untuk tiap satu karakter. Hal ini diakibatkan karena dalam melakukan pembentukan kode Huffman ada jalur pohon sama yang dilalui dalam melakukan pengkodean beberapa karakter yang berbeda

Perhatikan potongan pohon Huffman yang dibentuk dari *plaintext* pada table 1

Gambar 3 potongan pohon huffman



Karakter “B” akan dienkripsi dengan dengan “110000” dan karakter “W” akan dienkripsi dengan kode “110001”. Jika diperhatikan, hasil enkripsi karakter “B” dan “W” mengalami pengulangan deretan bilangan biner yaitu “11000”. Hal ini juga ditemui pada hasil enkripsi karakter”A” dan dan “M” yaitu terdapat pengulangan deretan bilangan biner “101”.Hal ini dapat dijadikan sebagai dasar untuk melakukan analisis untuk memperkirakan panjang masing-masing deretan bilangan biner yang digunakan untuk melakukan subtitusi karakter pada *plaintext*.

4. KESIMPULAN

Kode Huffman yang sebelumnya digunakan untuk megkompresi data ternyata dapat digunakan sebagai sarana untuk melakukan enkripsi dan dekripsi suatu teks. Algoritma enkripsi yang menggunakan kode Huffman tersebut diberi nama algoitma Huffman-Monoalphabetic. Beberapa keunggukan yang algoritma kriptografi Huffman-Monolaphabetic adalah tiap karakter disubtitusi dengan karakter bilangan biner yang memilki panjang yang berbeda-beda dan untuk tiap-tiap *plaintext* akan memilki *key* yang berbeda-beda. Disamping keungulan tersebut terdapat juga kekurangan yang dimilki algoritma ini diantaranya adalah *chipertext* yang dihasilkan kana lebih panjang dari berkas *plaintext.*

DAFTAR REFERENSI

[1] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Institut Teknologi Bandung, 2006.

[2] Munir, Rinaldi, *Diktat Kuliah IF2251 Strategi Algoritmik*, Institut Teknologi Bandung, 2006.